IMPLEMENTATION OF DATA ANALYTICS LEARNING IN PYTHON PROGRAMMING LEARNING ASSISTANCE SYSTEM

THESIS

By:

NOORA AULIA HIDAYAT Student ID. 2041720046

INFORMATICS ENGINEERING STUDY PROGRAM

INFORMATION TECHNOLOGY DEPARTMENT

STATE POLYTECHNIC OF MALANG

2024

APPROVAL SHEET

IMPLEMENTATION OF DATA ANALYTICS LEARNING IN PYTHON PROGRAMMING LEARNING ASSISTANCE SYSTEM

by:

NOORA AULIA HIDAYAT NIM. 2041720046

This thesis proposal was tested on July 29, 2024

Approved by:

1.	Supervisor	:	Yan Watequlis Syaifudin, ST., MMT., PhD. NIP. 19810105 200501 1 005	.....
2.	Supervisor II	:	Retno Damayanti, S.Pd., M.T. NIP. 19891004 201903 2 023	.....
3.	Examiner I	:	- -	.....
4.	Examiner II	:	- -	.....

Confirmed by,

Head of Information Technology Department	Head of Informatics Engineering Study Program
Dr. Rosa Andrie Asmara, ST., MT.	Dr. Ely Setyo Astuti, S.T., MT.
NIP. 19801010 200501 1 001	NIP. 19760515 200912 2 001

STATEMENT

I hereby declare that, to the best of my knowledge, this thesis does not contain any work or opinion that has been written or published by others, other than those that are cited in writing within the manuscript and mentioned in the citation/reference list. It also does not contain any work, in whole or in part, that has ever been submitted for the award of an academic degree at any higher education institution.

Malang, 29 April 2024

Noora Aulia Hidayat

ABSTRACT

***Noora Aulia Hidayat**. "Implementation of Data Analytics Learning in Python Programming Learning Assistance System".*

Supervisor: Yan Watequlis Syaifudin, ST., MMT., PhD., Co–Supervisor: Retno Damayanti, S.Pd., M.T.

Thesis, Informatics Engineering Study Program, Department of Information Technology, State Polytechnic of Malang, 2024.

The demand for Data Analytics in the business world is increasing. In 2023, there were 1,072 job postings related to the most sought-after skills, techniques, and degrees, as well as the current job market conditions on LinkedIn (Yosifova, 2023). This paper presents a Python Programming Learning Assistance System (PYPLAS) to assist students in self-learning Python programming for Data Analytics. The system is designed to facilitate students in independently learning Python programming and reduce the workload for university instructors. The test was conducted on 40 students from the Faculty of Information Technology at Politeknik Negeri Malang. The results show that all students successfully completed the Data Analytics learning task using Python and provided 85.71% positive feedback. From some topics there are relatively difficult experiments among others are chapter 3 experiments 2, chapter 3 experiments 3, chapter 3 experiments 4, and chapter 4 experiments 1. The longest average student takes to complete an experiment is 508 seconds and the highest submission is 3 times. The testing results indicate that the learning system with the topic of Data Analytics learning with Python provided has been helpful for students.

Keywords: Data Analytics, Python Programming, Codewars Test Framework

PREFACE

We give thanks and appreciation to Allah SWT/God Almighty for His presence, which allowed the author to finish this thesis, " **IMPLEMENTATION OF DATA ANALYTICS LEARNING IN PYTHON PROGRAMMING LEARNING ASSISTANCE SYSTEM**". The completion of the Diploma IV study program in Informatics Engineering at Politeknik Negeri Malang's Department of Information Technology requires the preparation of this thesis.
We acknowledge that the successful completion of this final report would not have been possible without the assistance and collaboration of numerous stakeholders. Therefore, we would like to thank the following people:

1. Mr. Yan Watequlis Syaifudin, S.T., M.MT., Ph.D., as the primary advisor who has provided guidance, direction, and motivation throughout the preparation of this thesis.
2. Mrs. Retno Damayanti, S.Pd., M.T., as the co-advisor who has assisted Mr. Yan in providing guidance and support.
3. Mr. Dr. Eng Rosa Andrie Asmara, ST., MT., as Head of Information Technology Department
4. Mrs. Dr. Ely Setyo Astuti, ST., MT., as Head of Informatics Engineering Study Program
5. To all the respected lecturers at Politeknik Negeri Malang who have imparted valuable knowledge and experience during my studies.
6. To my beloved family who have always provided prayers, support, and motivation.
7. To my fellow students in the Informatics Engineering Study Program who have always given encouragement and companionship throughout my studies.
8. And everyone else, whom we are unable to name individually, who assisted and encouraged the seamless compilation of this Final Report from start to finish.

The author admits that there are still a lot of flaws and issues with the way this final report was put together, both in terms of language usage and writing style. For this reason, the author appreciates helpful advice and criticism from a range of

sources in order to make this report better. This report should be beneficial to both the author and readers at large. Lastly, the author would like to thank everyone very much.

Malang, 29 July 2024

Noora Aulia Hidayat

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER I. INTRODUCTION

1.1. Background

Currently, data has become one of the most valuable assets for businesses. While some organizations build their business models entirely based on data, others regularly collect, store, and analyze large amounts of data to identify convincing patterns, gain insights, predict business outcomes, track consumer behavior, or enhance customer interactions. Gartner found that businesses are increasingly inclined to make data-driven decisions rather than intuition-based decisions (Gartner, 2022). Gartner also believes that 60% of organizations will use composable analytics technology (Goasduff, 2022). The current challenge lies in the fact that a significant number of organizations are struggling to analyze the vast amount of data they collect to discover patterns and trends within unstructured data that are not immediately apparent (Amarnath, 2023).

Therefore, the demand for Data Analytics in the business world is increasing. In 2023, there were 1,072 job postings related to the most sought-after skills, techniques, and degrees, as well as the current job market conditions on LinkedIn (Yosifova, 2023). Data Analytics is the application of computer systems in analyzing large datasets to support decision-making (Runkler, 2020). Data Analytics is a highly interdisciplinary field, adopting aspects from various other disciplines such as statistics, machine learning, pattern recognition, systems theory, operations research, or artificial intelligence.

In programming learning without validation, students find it difficult to measure how well they understand the material. This can hinder the growth of their confidence in solving coding problems. Similarly, identifying errors, finding mistakes in long and complex code without the help of validation tools can be very difficult, especially for beginners. The understanding of concepts is also lacking. Without an explanation of the location of the error, students tend to focus only on fixing the code syntax so that the program can run. There for the lecturer recognizes the significance of independent learning in mastering data analytics. To facilitate this, they have taken the initiative to develop a web-based learning platform equipped with automated code validation (Watequlis Syaifudin, 2021). This system will provide students with practical assignments and enable them to instantly check their answers. Students can complete practical assignments according to the guidelines available on the website. After completing the tasks, the results are submitted for evaluation using the Test-driven

Development method. The output validation results of student work will automatically appear on the web page, with the aim of facilitating students' independent learning of Data Analytics. The success of the web-based Data Analytics learning module with automated testing validation can be measured by the feedback provided by students at the end of the module.

1.2 Statement of the Research Problem

Based on the background previously outlined, the problem statements for the thesis with the title "Implementation of Data Analytics Learning in the Python Programming Learning Assistance System" are as follows:

1. How can this website assist students in self-learning Python programming for Data Analytics?
2. How to measure the level of understanding of students in the Python Assistance System for Data Analytics Learning in Python Programming Learning Assistance System?

1.3 Scoup of the Research

The problem constraints of conducting a thesis with the title " Implementation of Data Analytics Learning in Python Programming Assistance System" are as follows:

1. Learning platform based on website
2. Users are students of Politeknik Negeri Malang
3. Utilizing the Python programming languages
4. Learning materials related to Data Analytics include load data, data cleaning, data manipulation, data data visualization.

1.4 Objectives of the Research

The purpose of conducting the thesis with the title "Implementation of Data Analytics Learning in Python Programming Assistance System" is as follows:

1. Implementing Data Analytics Learning in Python Programming Assistance System on the website.

2. Providing code answer validation for Data Analytics Learning in Python Programming Assistance System.

1.5. Benefits of the Research

Assisting students in independently learning Data Analytics with automatic programming code verification is expected to enhance their self-directed learning experience.

1.6. Sistematika Penulisan

This thesis consists of several chapters, with the following descriptions for each chapter:

CHAPTER I: INTRODUCTION

This chapter explains the background, problem statement, limitations, objectives, benefits, and systematics of writing that serve as the basis for writing the thesis research.

CHAPTER II: THEORETICAL BASIS

This chapter contains previous research using programming language learning models and testing that has been used. It also contains theories that are used as references to facilitate understanding and solving problems related to the implementation of testing using Codewars_test in the data analytics learning process.

CHAPTER III: RESEARCH METHODOLOGY

This chapter discusses the research methodology, including the time and place of the research, data collection methods, data processing methods, system design, and the design of system tests used in this research.

CHAPTER IV: SYSTEM ANALYSIS AND DESIGN

This chapter contains explanations of the system and system requirements, including functional and non-functional requirements. It also contains system designs, including system model designs and system architecture.

CHAPTER V: IMPLEMENTATION AND TESTING

This chapter discusses the implementation of the design and analysis carried out in the previous section. This chapter also contains testing carried out according to the predetermined test plan.

CHAPTER VI: RESULTS AND DISCUSSION

This chapter contains the results of testing the python programming learning assistance system using codewars_test with the topic of Data Analytics Learning.

CHAPTER VII: CONCLUSION AND SUGGESTIONS

This chapter discusses the conclusions obtained from the results of the implementation and testing of the research and contains suggestions for further system development.

CHAPTER II. THEORETICAL BASIS

2.1 Literature Study

The previous research, taken from the research thesis (Yulvarisma, 2022), is about the Implementation of Basic Programming Learning in Python. This research is based on a website for easy access. The system is designed to facilitate students in independently learning the basics of Python programming. User-inputted code will be evaluated using Codewars. The framework and database utilized are Laravel and MySql. The results were obtained after testing with 40 students from the Information Technology Department at the State Polytechnic of Malang. The testing results indicate that the learning system with the topic of basic Python learning provided has been helpful for students.

Python Programming Learning Assistance (PYPLAS) is developed to assist students in learning Python programming and reduce the workload for university instructors. It provides fill-in-the-blank exercises ranging from beginner to advanced levels, generated using a blank element selection algorithm. The framework and database used are Laravel and SQLite. The research was conducted with 5 students from 2 universities, involving 27 offline Python programming problems. The results of the study indicate an improvement in users' proficiency in Python programming. (Hnin & Zaw, 2021).

According to a publication titled "Studying the Impact of Auto-Graders Giving Immediate Feedback in Programming Assignments" compiled by (Mitra, 2023), while students use immediate input from an auto-grader, it does not discourage them from developing independent testing skills. Furthermore, feedback enables students, particularly those from underrepresented groups (such as women), to learn more successfully and confidently.

Based on the literature study from previous research, the upcoming study will discuss Analytic Data learning in the Python Programming Learning Assistance System. This research is web-based for easy accessibility. The framework for this study uses Laravel, with MySql as the database. It employs Python programming languages for creating the validation file code. Testing for user input code verification will be conducted using the Codewars Test Framework.

2.2 Theoretical Basis

2.2.1 Data Analytics

Data can appear as something simple yet vast and unstructured. In its raw state, data lacks meaning, structure, or relationships, making it unsuitable for decision-making. Data appears in various formats such as database tables, spreadsheets, text files, and others. Information, on the other hand, is the result of processed data; data transforms into information when relationships are formed. Knowledge emerges from the combination of data and information, along with the expertise, experience, and opinions of experts, ultimately providing benefits to the organization (Patnaik, 2019).

Data Analytics is the application of computer systems in analyzing large datasets to support decision-making (Runkler, 2020). Data Analytics is a highly interdisciplinary field, adopting aspects from various other disciplines such as statistics, machine learning, pattern recognition, systems theory, operations research, or artificial intelligence.

2.2.2 Python

Python is a compelling choice for several reasons. Its extensive collection of versatile libraries makes it suitable for a wide range of applications, including statistical programming, deep learning, network applications, web crawling, and embedded systems. The language utilizes high-performance libraries, such as LAPACK for linear algebra in SciPy and CUDA for parallel GPU processing in TensorFlow. Python's simplicity, resembling natural language syntax, makes it easy to learn and use, offering a coding experience that provides freedom without constraints on variables like constants or public/private distinctions. Overall, Python's adaptability, performance, and simplicity make it a preferred language for diverse programming tasks (Mukhopadhyay, 2018).

2.2.3 Learning Assistance System

A Learning Assistance System (LAS) is a comprehensive and integrated platform designed to support and enhance the learning experience for students. It encompasses various tools, resources, and strategies aimed at providing academic support, developing skills, and facilitating personalized learning paths. Typically used in educational institutions, Learning Assistance Systems can include features such as online tutorials, interactive learning modules, virtual classes, and assessments. Often, these systems

incorporate adaptive learning technology to meet the individual learning needs and styles of students. The focus extends beyond academic content to the development of essential skills such as critical thinking, problem-solving, and collaboration. Learning analytics can be integrated to track student progress and identify areas that require additional attention. Overall, Learning Assistance Systems are designed to create a dynamic and inclusive learning environment, promoting student success and academic achievement.

2.2.4 Unit Test

Unit testing is the basic level of testing. Unit testing focuses on testing smaller building blocks of a program or system (William E. Lewis, 2009). Unit testing are practices for testing small, individual and isolated units of entire source code, such as methods and classes in Java (Syaifudin et al., 2020). These tests are often used by developers to find errors and defects that are invisible when a program is executed (Hasibuan, 2021).

2.2.5 Codewars Test Framework

The Codewars Test Framework provides a simple yet effective method for building Python unit tests. This framework is the foundation of the Codewars online platform, ensuring the quality of user-submitted code. But the benefits go beyond that! You can use it to test your own Python code as well. The framework allows you to design individual test cases, each of which serves as a function that accepts inputs and performs assertions (Codewars, n.d.). These assertions are essentially checks that determine whether a particular condition is true or untrue. Furthermore, the Codewars Test Framework has a comprehensive collection of assertion functions that allow you to validate a variety of characteristics of your code, such as equality, error handling, and truthiness. In summary, this framework provides a complete and accessible toolkit for assuring quality.

The choice of testing framework depends on specific needs. Codewars_test is simpler and more focused on education and ease of use. Its syntax is clear and understandable, even for beginners. Assertions like test.assert_equals and test.assert_not_equals directly convey the test's purpose. When a test fails, informative error messages guide users towards corrections. This framework's educational focus helps users grasp fundamental unit testing concepts.

The following is a simple example of unit testing using the Codewars Test Framework for user answer validation:

```
@test.describe('Fixed Tests')
def fixed_tests():

    # Basic Tests: Test the basic behavior (basic understanding
of the task).
    @test.it('Basic Test Cases')
    def basic_tests():
        test.assert_equals(two_oldest_ages([2,  4,  6,  9,  12,
14]), [12, 14])

    # Edge Cases: Test the edge cases, which are not common but
hard to correctly solve.
    # These are needed because "rare but hard cases" are not
well-covered by random tests only.
    @test.it('Edge Cases')
    def edge_case_tests():
        test.assert_equals(two_oldest_ages([0, 0, 0, 0, 0, 0]),
[0, 0])

# Random tests: Test the behavior against your reference
solution.
# This is mainly to prevent the warrior passing the tests by
hardcoding the fixed cases.
# The functions `random`, `randint`, `choice`, `shuffle`, and
`sample` inside Python's `random` module will be helpful.
@test.describe('Random Tests')
def random_tests():

    # The reference solution should be placed here, in order to
prevent the warrior from abusing your reference solution
    def _reference(ages):
        ...

    # When running a random test, you need to make sure that the
expected value is computed first.
```

```
    # If the warrior's solution is run first, it may mutate the
input list and thus easily bypass them.
    # Also, take extra care on your own reference solution to
not mutate the input :)
    def _do_one_test():
        ages = generate_random_case()
        expected = _reference(ages)
        test.assert_equals(two_oldest_ages(ages), expected)
        #             test.assert_equals(two_oldest_ages(ages),
_reference(ages))

    # The number of random tests must be enough to test every
possible aspects of the input.
    # The rule of thumb is 100 tests, but you have to think
carefully according to the requirements of your Kata.
    @test.it('Random Test Cases')
    def random_test_cases():
        for _ in range(100):
            _do_one_test()
```

Figure 2. 1 The example of Codewarse_test implementation

(*Source: codewars.com*)

This code defines two sets of tests for a function called two_oldest_ages. The first set, fixed_tests, focuses on verifying the function's basic behavior with simple test cases. The basic_tests function checks if the function correctly identifies the two oldest ages in a list of numbers.

The second set, random_tests, aims to ensure the function works under various scenarios beyond the basic cases. It includes edge_case_tests that verify its behavior on lists containing only the same age (e.g., all zeros). More importantly, it uses random_tests to test the function against a hidden reference solution (_reference). This prevents the solution from simply memorizing fixed cases to pass the tests. The _do_one_test function generates random age lists, calculates the expected result with the reference solution, and then compares it with the output of the two_oldest_ages function. By running this test 100 times, the code aims to provide a more comprehensive evaluation of the function's ability to handle diverse age distributions.

2.2.6 Learning Material for Data Analytics with Python

The learning material presented includes load data, data manipulation, data visualization. This material is sourced from the W3Schools website and realpython.com. The researcher is responsible for creating a practical guide based on the material or topics to be presented. Testing will be conducted on the results section of the source code submitted by students. Table 2.1 outlines the practical tasks that will be presented.

Table 2. 1 Practical Assignment

No	Practical Assignment	Materials
1	Students gain knowledge about loading data from local files, URLs, and dictionaries and reading it as a CSV file. Learn how to show the data on the console after that.	Load Data
2	Acquiring knowledge about cleaning datasets, including removing superfluous columns, handling missing values, and saving the resultant clean dataframe as a new CSV file.	Data Cleaning
3	Learning the skills necessary to manipulate data in order to measure the entire quantity of a particular record, choose a particular data category, and analyze the data using algorithms comes in useful.	Data Manipulation
4	Studying libraries such as Matplotlib and Seaborn to create informative data visualizations, such as charts, bar diagrams, scatter plots, and other visualizations.	Data Visualization

2.2.7 iCLOP

iCLOP (Intelligent Computer-Assisted Programming Learning Platform) is a comprehensive platform that will include modules for self-learning, automatic assistance, and auto-grading across a wide range of programming languages and platforms. The project intends to address difficulties in the Industry 4.0 age, where automation and software development are crucial. The platform incorporates a number of cutting-edge learning paradigms, such as project-based learning, assignment-based learning, gamification, and blended learning. This rudimentary Python learning system will eventually be included into the iCLOP. This study will include the automated assistance (AA) function, which is used while students are learning a subject.

2.2.8 Test-driven Development

Unit testing serves as the foundation for Test-Driven Development, or TDD, an Agile software development methodology. TDD adopts a test-first approach, requiring the creation of tests for a program component before the actual code implementation. This section guides the reader in effectively implementing unit tests and integrating them into Agile development practices (Parsa, 2023).

TDD (Test-Driven Development) is closely related to development practices that are agile and iterative in nature. It encourages developers to focus on small, testable code units and promotes continuous feedback throughout the development process. By writing tests first, developers gain a clear understanding of the desired functionality and can iteratively build and improve the codebase while maintaining an automated testing safety net. TDD is a repeated cycle process until all tests pass, a cycle as illustrated in Figure 2.1

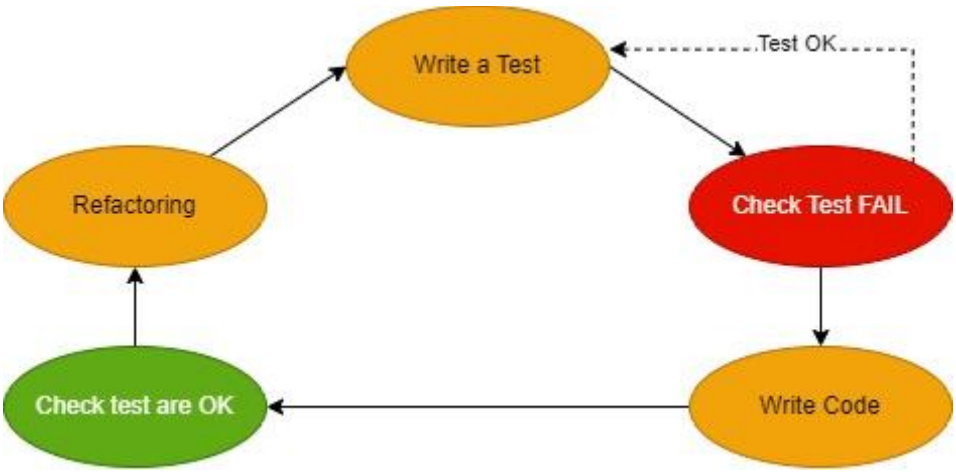


Figure 2. 1 Test-Driven Development Cycle

Implementing new features in software development follows a structured process that begins with writing tests. These tests are brief and clear, serving as a guide for developers to consider system requirements and design before actual coding. Running initial tests is expected to fail because there is no existing application code. This failure is intentional and highlights the development target. If the initial tests succeed, they should be revised to fail. Next, developers write production code only enough to satisfy the tests, being careful not to exceed the scope of the tests. Then, all tests, including new ones, are executed. If all tests pass, it indicates that the new code does not break existing

features and meets the new requirements. In case of failure, developers modify the code until all tests pass. The last step involves refactoring the code to improve cleanliness, eliminate duplication, and enhance readability and maintainability. Throughout this process, test cases are regularly executed to ensure that refactoring does not unintentionally affect unrelated features. This systematic approach, known as Test-Driven Development (TDD), encourages careful consideration of requirements, systematic coding, and continuous improvement of the code (Bruel, et al., 2020).

CHAPTER III. RESEARCH METHODOLOGY

3.1. Time and Place of Research

This research will be conducted at the State Polytechnic of Malang for 3 months, from April 2024 to July 2024.

3.2. Data Collection Methods

Data collection will be conducted at the State Polytechnic of Malang using an experimental method. The subjects of this research are students of the State Polytechnic of Malang. In this experimental process, students will solve problems related to Data Analytics. The learning materials or topic and experiments in the form of document guides, test files, student answer outcomes are the data in this study that the system needs to process. The data that students have uploaded serves as the basis for the compilation of student response results. The uploaded data consists of response codes and student comments from each experiment that they completed. This information will be utilized by researchers to gauge how challenging the experiment is given the Data Analytics Learning content, as well as the success rate of students in completing all job sheets to determine the challenges they encounter.

3.3 System Design

This research focuses on the application of Unit Testing using Codewars_test in the context of Data Analytics learning materials. Before writing the system's program code, it is necessary to prepare a system design, as shown in the figure.

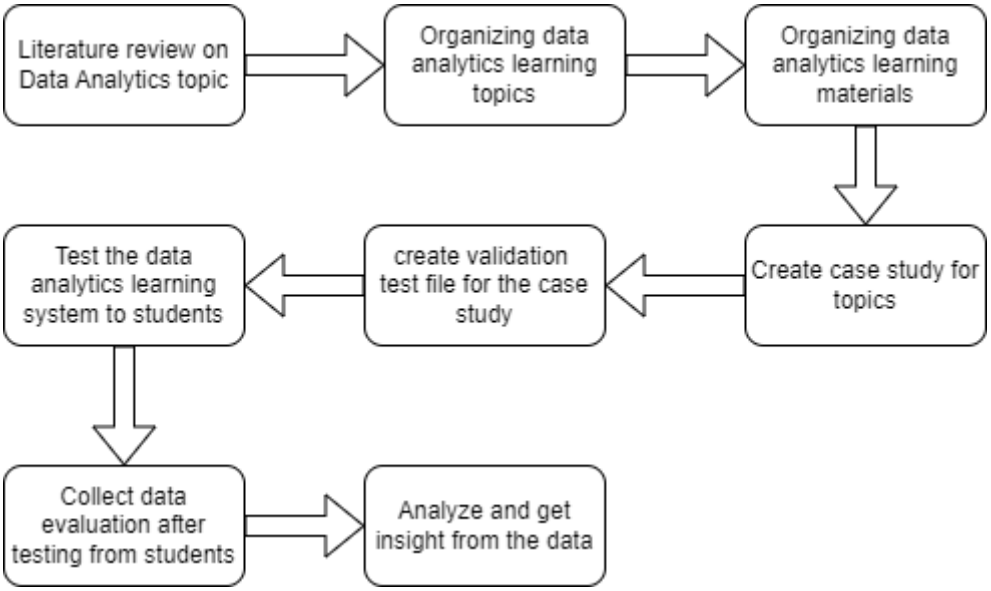


Figure 3. 1 System Design

The graphic displays a flowchart that lists the procedures needed to complete a research project on data analytics learning. To lay the groundwork for the research, a survey of the literature on data analytics-related subjects is the first step in the process. The next step is to arrange the subjects for learning data analytics, and then the learning resources. Student testing of the data analytics learning system is the next step. A validation test file is made for a case study in order to make this easier. Students provide data evaluation once the system has been tested. After then, this data is examined to obtain knowledge. In the context of learning data analytics, the flowchart depicts a cyclical process of research, development, testing, and evaluation.

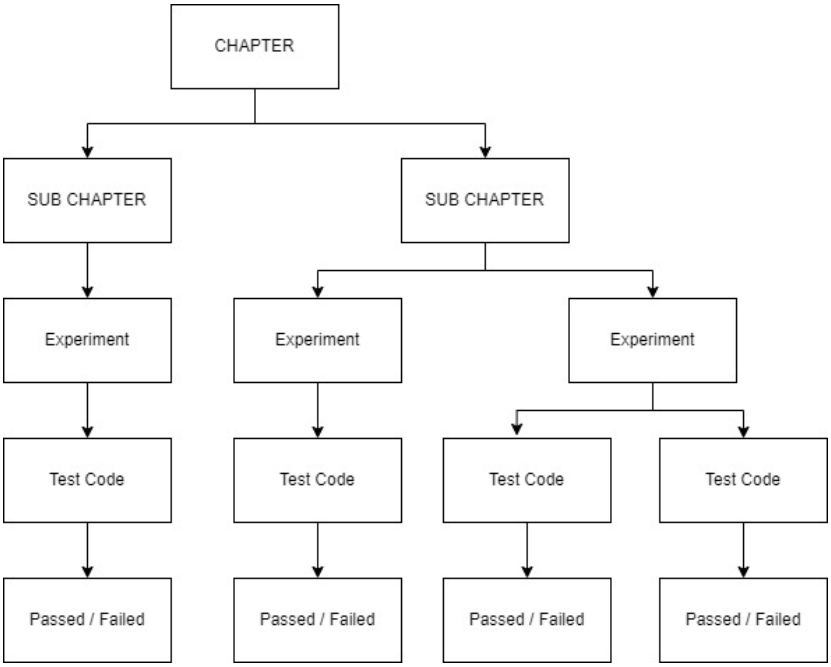


Figure 3. 2 Learning Material Design

In the design of the data analytics learning system in this study, there are main topics or chapters consisting of one to several sub-chapters. Each sub-chapter may contain one to several experiments. Each experiment can have one test code and one passed/failed outcome, and a single experiment may also include multiple test codes and corresponding passed/failed results, depending on the experiment's design.

3.4 System Testing

User testing is a testing process conducted by users with the aim of producing test result documents that serve as evidence that the system meets user requirements. This testing involves requesting volunteers to use the system and provide feedback on the experiments conducted. User testing will be directed towards students in the Information Texchnology program at Politeknik Negeri Malang. A trial will be conducted with 40 students over a period of three days.

To evaluate the success of this research, students were asked to provide comments or feedback on the modules or source code provided during the testing phase via Google Forms. The aim of this was to evaluate the effectiveness of implementing tests using Codewarse_test as previously proposed.

CHAPTER IV. SYSTEM ANALYSIS AND DESIGN

This chapter contains an explanation of the system and system requirements, including functional and non-functional requirements. It also includes a system design that covers system architecture, use cases, and activity diagrams.

8.1 System Description

In this research, a web-based learning system on Data Analytics using codewars_test testing is developed. The learning model consists of three stages: Material Creation, Material Completion, and Evaluation of Completion Results. The material creation activity involves creating materials from the specified topic, which is Data Analytics. Completing the learning materials involves working on practical exercise questions, while evaluating the completion results involves running the test code from the exercise questions to determine the correctness of the work.

4.2 System Requirement Analysis

Explaining the needs for the system that will be constructed during the design phase is the goal of the system requirements analysis phase. A list of needs and user identification are part of the necessary system requirements analysis.

8.1.1 Actor Analysis

The system's user identification phase seeks to identify the users who will subsequently communicate with it.

Tabel 4. 1 User’s Identification

User	Description
Admin	Admin is an actor who manages files consisting of test codes, modules files.
Lecturer	Lecturers are the users responsible for examining and checking the work result of each student.
Students	Students are the users responsible for completing the module practicum assignment.

4.1.2 Functional Requirement

The requirements that outline the functions of the system are known as functional requirements.

The learning module to be prepared will consist of:

- A module containing material about projects and modules in project work.
- A Test file containing code in Python to test whether the program code being worked on is correct.

4.1.3 Non-functional Requirements

Non-functional requirements analysis is a set of specifications that a system must meet. These requirements include both software and hardware needs. The non-functional prerequisites for developing a data analytics learning system are as follows:

- Google Colaboratory.
- Processor Inte Core i5
- RAM 8,00 GB
- SSD 239GB

4.2 System Architecture Design

The process of putting a software system's components into a visual representation is called system architecture design. This research focuses on applying Unit Testing using codewars_test to Python learning materials on Data Analytics. Before writing the system's program code, we will design the system comprehensively. This design includes the overall system structure (system architecture), diagrams illustrating user interactions with the system (use case diagrams), and diagrams showing the system's workflow (activity diagrams).

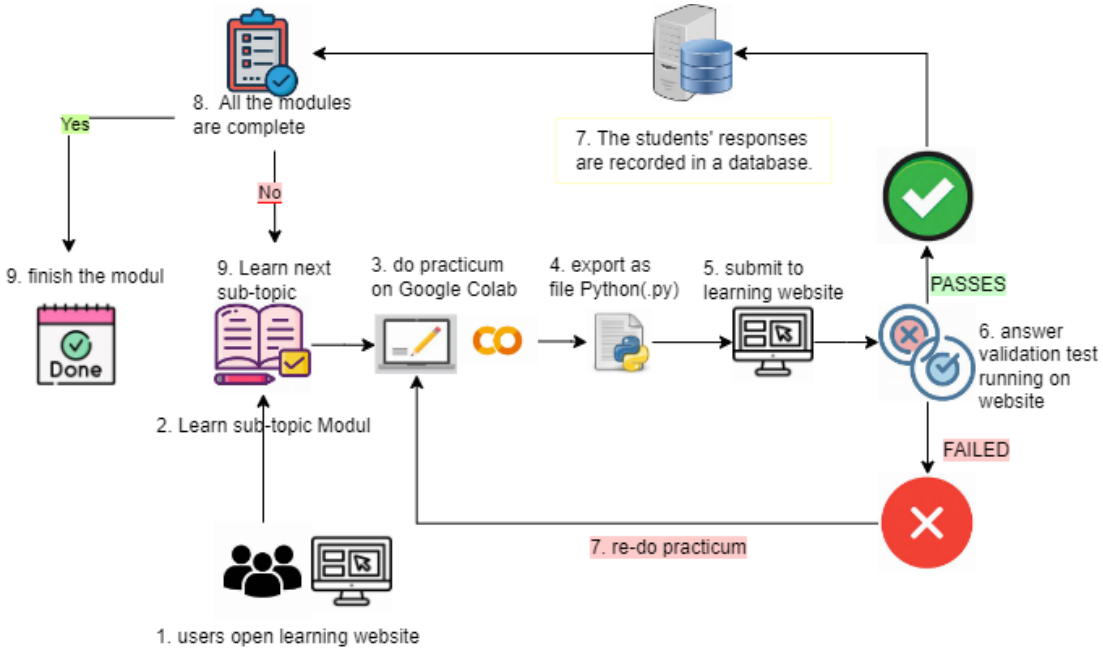


Figure 4. 1 System Architecture Design for Data Analytics Learning

Figure 4.1 presents a sequential overview of the learning process for Data Analytics Learning. The process commences with students preparing their workstations, followed by the distribution of a comprehensive learning module by the instructor. Subsequently, students engage in self-directed study, complete assigned tasks, and conduct self-assessment using provided test codes. Upon successful completion of tasks, the answer will be storen in the database.

4.2.1 Use Case Diagram

A use case diagram is a diagram that represents the interactions between actors and a system. Use case diagrams are commonly used to capture system requirements and to help design the system architecture. Figure 4.2 presents a use case diagram for this research. However, in Figure 4.2, the focus is limited to the development of use cases such as **creating a topic, creating learning materials, and create file test**.

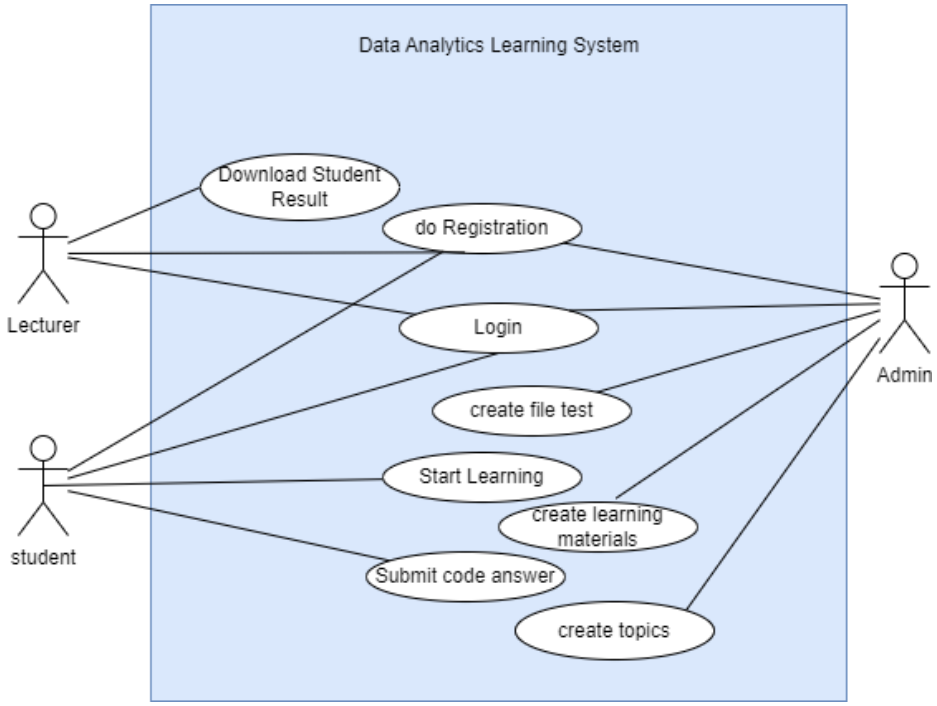


Figure 4. 2 Use Case

Tabel 4. 2 Use case Descriptions

Use Case	**Description**
creating a topic	Determining and creating topics to learn Data Analytics.
create file test	Developing a set of unit tests using codewars_test in Python.
creating learning materials	Creating Data Analytics learning materials divided into sub-chapter, with each section having corresponding assignments.
Registration	Registering within the system to gain access to iCLOP according to the user's level.
Login	Authenticating a user by selecting a user level and entering a username and password to gain access to the system.
Start Learning	The learning process involves several steps, starting from reading the materials, comprehending the concepts, following the step-by-step instructions, and completing all given assignments.

Submit code answer	Submitting the completed work according to the instructions provided in the learning materials.

4.2.4 Activity Diagram

This research will focus on the topic of Data Analytics, which will later be applied to iCLOP. The following is the workflow of the iCLOP system.

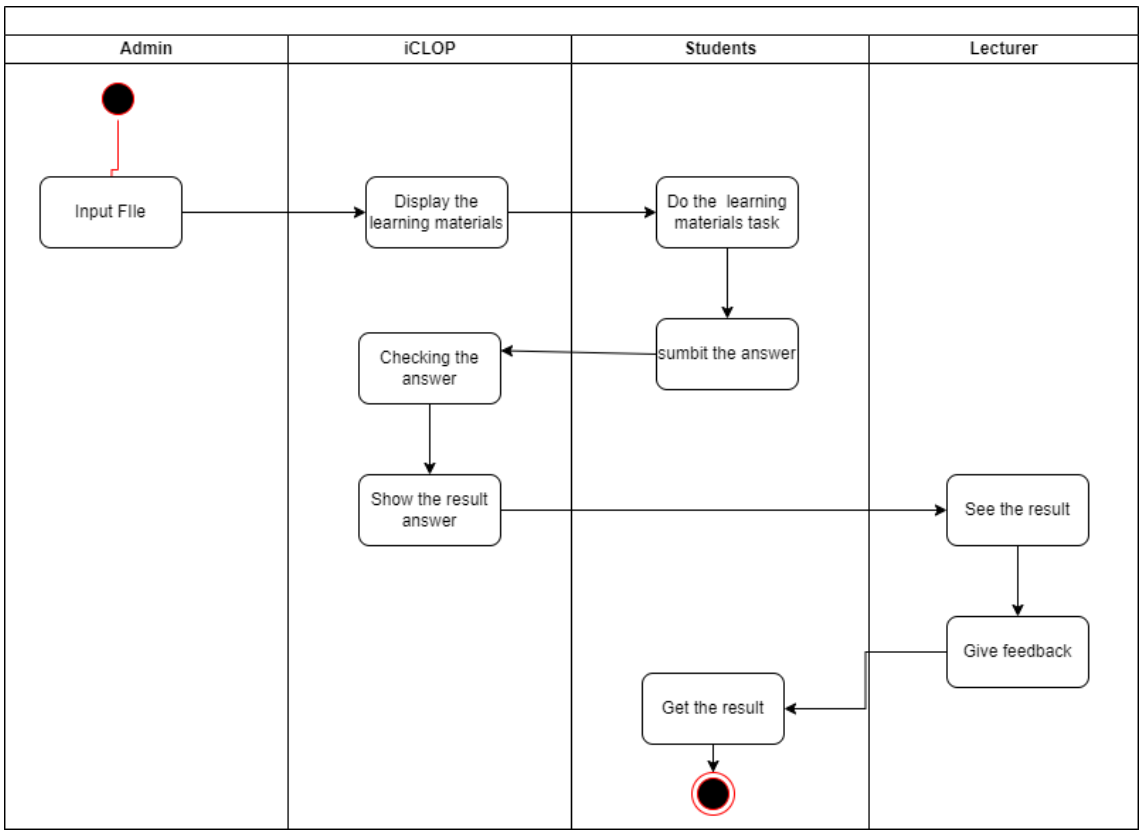


Figure 4. 3 Activity Diagram of iCLOP

Based on the activity diagram of the iCLOP platform, this research focuses on three main stages in the learning process, namely the creation of learning materials, the implementation of learning, and the evaluation of learning outcomes.

4.2.4.1 Creating Learning Materials Activity Diagram

The activity diagram in Figure 4.6 outlines the steps in creating learning materials on Data Analytics. The idea for the materials in the Data Analytics topic is from websites such as W3school.com and realpython.com. First, the admin/author creates a learning topic, in this case, the steps of data analyzing such as loading data, cleaning data, data

manipulation, and data visualization. Next, the admin creates test code to validate students' work, ensuring it covers key aspects of the topic. The test code is then validated to ensure it functions correctly. Additional files needed for the lesson, such as images or documents, are extracted. These files, along with the learning content, are compiled into a module, which serves as a structured guide for students. Finally, once all steps are completed, the learning topic is ready for distribution.

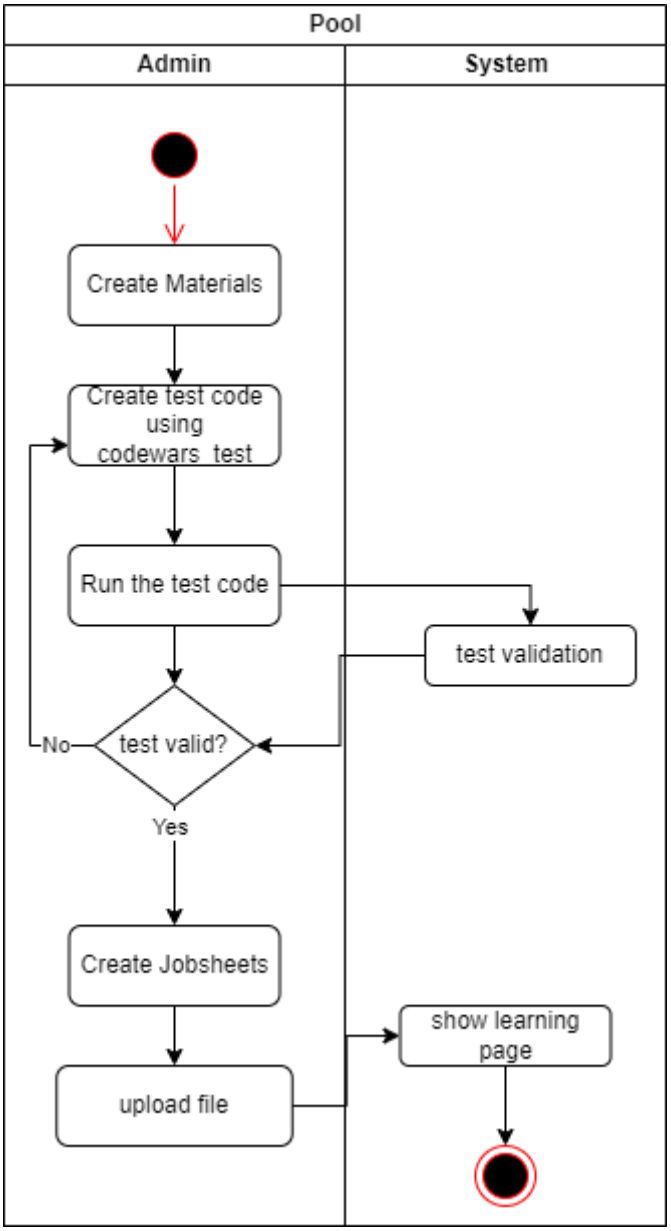


Figure 4. 4 Creating Learning Materials Activity Diagram

4.2.4.2 The implementation of learning Activity Diagram

The activity diagram in Figure 4.7 outlines the steps for completing the learning material on PHP Forms. First, students download two types of files: the Module File and

Test Code. The Module File contains instructions on the learning topic, the Test Code file is used to validate their work, and Additional Files may include supplementary materials. Students then follow the Module File instructions to work on the learning topic. After completing the topic, they use the Test Code to check the correctness and functionality of their implementation. If all tests pass, the final step is to submit their work according to the provided guidelines or additional instructions.

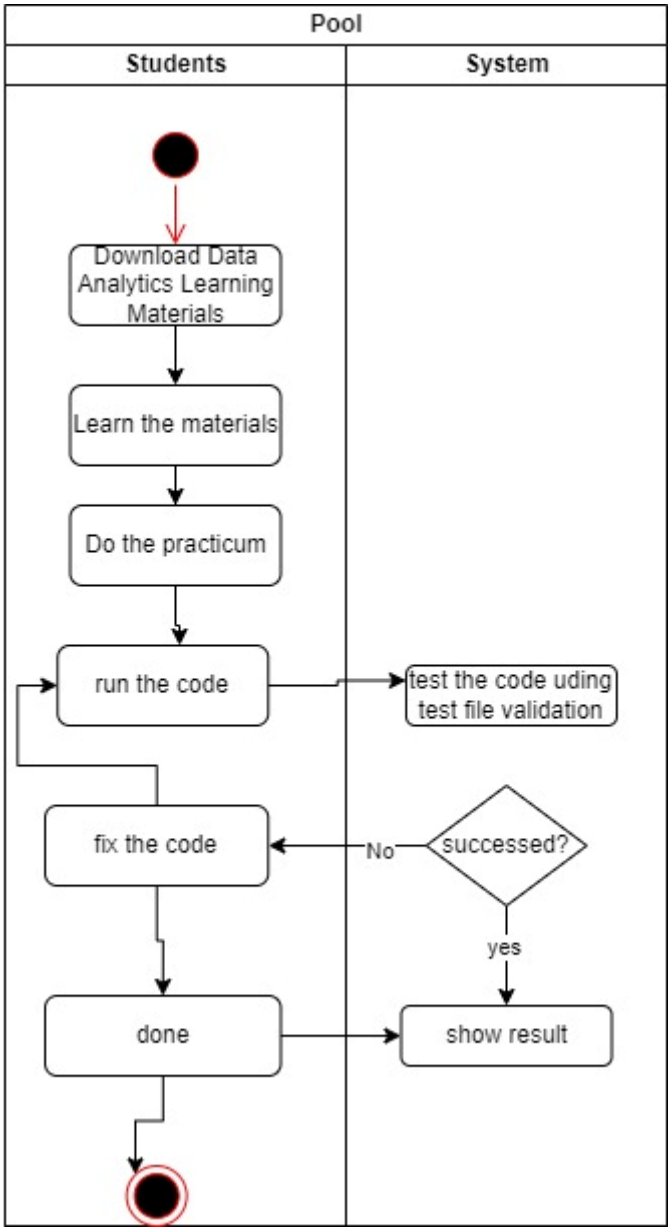


Figure 4. 5 the learning implementation Activity Diagram

4.2.4.3 The evaluation of learning outcomes

The activity diagram in Figure 4.8 outlines the steps in the evaluation process of learning materials on Data Analytics. First, the instructor opens the completed tasks

submitted by the students, which include file answer in Python (.py) format. Next, the instructor reviews these results to provide feedback and assessment. The evaluation may consider factors such as the number of code attempts, the number of successful tests, code clarity, and other criteria set by the instructor.

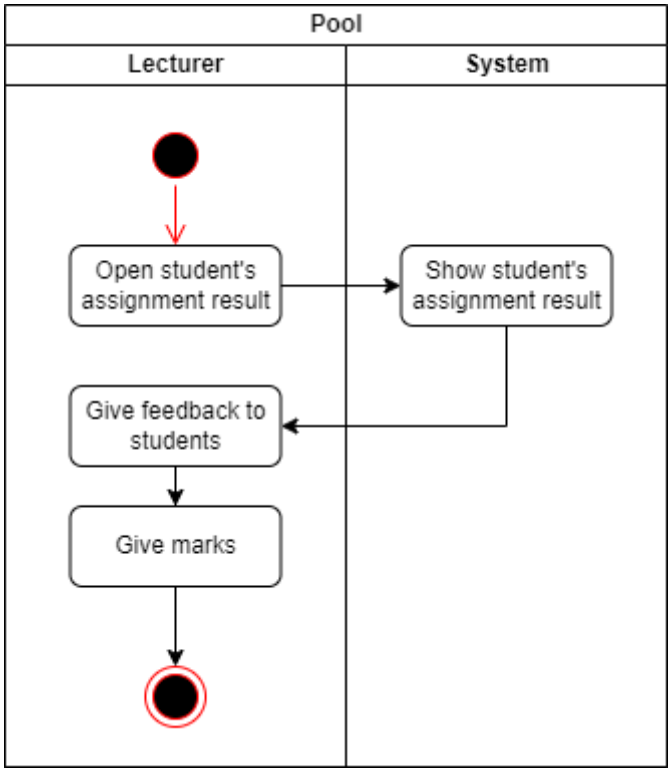


Figure 4. 6 The evaluation of learning outcomes activity diagram

4.3 Experimental Planning of Learning Materials

Material experimental design is a project of experimental creation of learning topics that will be provided on learning data analytics in the system. Here's a list of 9 experimental learning topics:

Tabel 4. 3 Experimental Learning Topics List

No	Experiment Name	Topic	Description
1.	Get the dataset from url and read as CSV file.	Data Loading	Students were given an experiment to access data sets from external sources into the system. Using this function *pd.read_csv(url)*
2.	Display the Produk Online Store Data.	Data Loading	Students are given an experiment to delete data columns that are not needed in the data analysis process. Display 5 first rows in the dataset using *head_rows().*
3.	Handling missing values for Calories data.	Cleaning Dataset	Students are given a practicum to handle missing values in a column and fill them with the column's mean value using the fillna() function.
4.	Cutting a NumPy 2D array on a specific coverage.	Data Manipulations	Taking product-specific sales data and applying slicing to extract specific data on the Numpy 3D array.
5.	Calculate Total Product Revenue.	Data Manipulations	Calculating Total Product Revenue using the sum() function.
6.	Selecting Specific Movies from Movie dataset.	Data Manipulations	Analyses film datasets to find 10 films using argsort() and iloc() functions.
7.	Analyze the Measures of Variability in customer's Age.	Data Manipulations	Analyze the Measures of Variability in customer's Age using mean(), median(), std(), skew(), quantile(), and corr() function.
8.	Electronic Sales Analysis: Understanding Sales Trends	Data Manipulations	Electronic Sales Analysis: Understanding Sales Trends Through Measures of Central

	Through Measures of Central Tendency.		Tendency using mean(), median(), and mode() function.
9.	Visualizing the age distribution of the population based on age range using a Pie Chart.	Data Visualization	analyze the age distribution of the population of a city by visualizing it using pie charts.

CHAPTER V. IMPLEMENTATION AND TESTING

In this section describes the implementation of the system with a detailed display according to the design and components that have been done previously. On the website Python Programming Learning Assistance System – Data Analytics Learning Topic implements Python programming language and testing using Codewars_test.

5.1 Limitations of Implementation

As for the limitations on the implementation of the project on Learning Data Analytics with Python:

1. Development of tasks using Google Colaboratory.
2. Application testing is done using Codewars_test.

5.2 Learning Implementation

5.2.1 Data Analytics Learning Modul

The Data Analytics Learning Module consists of three parts: material review, case studies, and practical exercises. The Data Analytics Learning Module is divided into several sections: Data Analytics material, preparation of Google Colab tools, Data Analytics case study examples, and Data Analytics practical exercises. Table 5.1 provides a description of each practical exercise for each topic.

Table 5. 1 Modul Descriptions

No.	Topics	Descriptions
1	Chapter 1 Practicum 1	Students were given an experiment to access data sets from external sources into the system. Using this function pd.read_csv(url)
2	Chapter 1 Practicum 2	Students are given an experiment to delete data columns that are not needed in the data analysis process. Display 5 first rows in the dataset using head_rows().
3	Chapter 2 Practikum 2	Students were given an experiment to tidy up the column name arrangement to make it uniform. Delete the column using df.rename(columns=clean_names).

4	Chapter 2 Practicum 3	Students are given an experiment to handle an empty value on a column and fill it with the mean value of that column using fillna().
5	Chapter 3 Practicum 1	Taking product-specific sales data and applying slicing to extract specific data on the Numpy 3D array.
6	Chapter 3 Practicum 2	Calculating Total Product Revenue using the sum() function.
7	Chapter 3 Practicum 3	Analyze film datasets to find 10 films using argsort() and iloc() functions.
8	Chapter 3 Practicum 4	Analyze Measures of Variability on customer's Age using mean(), median(), std(), skew(), quantile(), and corr() function.
9	Chapter 3 Practicum 5	Analyzing electronic sales using measures of central tendency (mean, median, and mode).
10	Chapter 4 Practicum 1	Analyze the age distribution of the population of a city by visualizing it using pie charts.

5.2.1.1 Chapter 1 Practicum 1

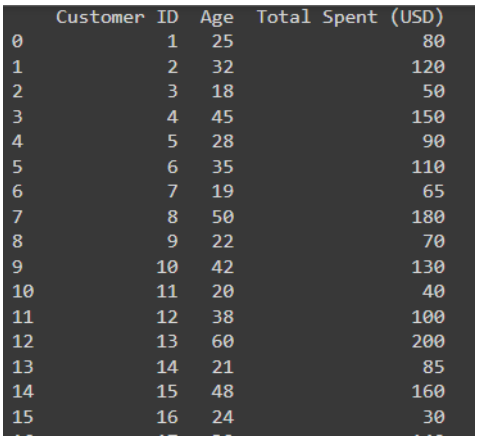
In this module, students will learn how to load data. The given data will be in the form of a URL, and students will learn how to load it into a dataframe. Then, they will display the data in the dataset using the print() function.

```
import pandas as pd

url="https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main/purchases.csv"

def data_load():
  df =pd.read_csv(url)
  return df

print(data_load())
```



	Customer ID	Age	Total Spent (USD)
0	1	25	80
1	2	32	120
2	3	18	50
3	4	45	150
4	5	28	90
5	6	35	110
6	7	19	65
7	8	50	180
8	9	22	70
9	10	42	130
10	11	20	40
11	12	38	100
12	13	60	200
13	14	21	85
14	15	48	160
15	16	24	30

Figure 5. 1 Chapter 1 Practicum 1 Output Result

5.2.1.2 Chapter 1 Practicum 2

In this module, students will learn how to load data. The data provided will be in the form of a URL, and students will learn to load it into a dataframe. Then, they will display the data in the dataset using the print() function. After that, students will be asked to display a sample of the data, specifically a random row from the dataset, using the sample() function.

```
import pandas as pd

url="https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main/online_store_data.csv"

def data_load():
  df =pd.read_csv(url)
  return df

def sample_rows():
  sample = data_load().sample()
  return sample

sample_rows()
```

```
    Order ID  Customer ID Product ID          Product Name   Price  \
39     10039          1.0     Q4MPM5  Gaming Shirt (Black)  146.07

    Order Date  Quantity
39  2024-04-25         3
```

Figure 5. 2 Chapter 1 Practicum 2 Output Result

5.2.1.3 Chapter 2 Practicum 1

In this module, students will learn how to clean data. First, they will remove unnecessary columns in the dataset for analysis using the drop() function. Then, they will tidy up the column names in the dataset using the rename() function. After that, students will display the names of the cleaned columns in the dataset using the print() function.

```
import numpy as np
import pandas as pd

url                                                    =
"https://raw.githubusercontent.com/noora20FH/skripsi_noora2
023/main/nyc_perumahan.csv"  # Replace with your actual URL

def load_data():
  df = pd.read_csv(url)
  return df

def clean_columns():
   unnecessary_columns  =  ['BLOCK',  'LOT','EASE-MENT','TAX
CLASS AT PRESENT','TAX CLASS AT TIME OF SALE']
  df = load_data().drop(unnecessary_columns, axis=1)
  return df

def clean_columns_name():
  clean_names = {
      "BUILDING CLASS CATEGORY": "BUILDING_CLASS_CATEGORY",
```

```
                              "BUILDING        CLASS        AT
PRESENT":"BUILDING_CLASS_AT_PRESENT",
      "APARTMENT NUMBER": "APARTMENT_NUMBER",
      "ZIP CODE": "ZIP_CODE",
      "RESIDENTIAL UNITS": "RESIDENTIAL_UNITS",
      "COMMERCIAL UNITS": "COMMERCIAL_UNITS",
      "TOTAL UNITS": "TOTAL_UNITS",
      "LAND SQUARE FEET": "LAND_SQUARE_FEET",
      "GROSS SQUARE FEET": "GROSS_SQUARE_FEET",
      "Box Office (Millions USD)": "Box_Office",
      "YEAR BUILT": "YEAR_BUILT",
              "BUILDING   CLASS   AT   TIME   OF   SALE":
"BUILDING_CLASS_AT_TIME_OF_SALE",
      "SALE PRICE": "SALE_PRICE",
      "SALE DATE": "SALE_DATE"
  }
  data = clean_columns().rename(columns=clean_names)
  return data
print(clean_columns_name().columns)
```

```
Index(['BOROUGH', 'NEIGHBORHOOD', 'BUILDING_CLASS_CATEGORY',
       'BUILDING_CLASS_AT_PRESENT', 'ADDRESS', 'APARTMENT_NUMBER', 'ZIP_CODE',
       'RESIDENTIAL_UNITS', 'COMMERCIAL_UNITS', 'TOTAL_UNITS',
       'LAND_SQUARE_FEET', 'GROSS_SQUARE_FEET', 'YEAR_BUILT',
       'BUILDING_CLASS_AT_TIME_OF_SALE', 'SALE_PRICE', 'SALE_DATE'],
      dtype='object')
```

Figure 5. 3 Chapter 1 Practicum 2 Output Result

5.2.1.4 Chapter 2 Practicum 3

In this module, students will learn how to clean a dataset by removing null values. First, students will extract rows of data from the dataset that contain null values using the isnul() function. Then, they will fill in the null values with the average value of that column using the fillna() and mean() functions. Finally, they will display all the columns that have null function using the print() function. The output must has 0 value for all columns that

indicate there is no null values anymore because it's already replaced by the average value.

```
import pandas as pd

def load_data():
                                url                                =
"https://raw.githubusercontent.com/noora20FH/skripsi_noora2
023/main/data.csv"
  df = pd.read_csv(url)
  return df
print(load_data().isnull().sum())

def updated_data():
                            updated_df                             =
load_data().fillna(load_data()['Calories'].mean())
  return updated_df

print(updated_data()[["Calories"]].iloc[17:142])
```

```
Missing value:
Duration    0
Pulse       0
Maxpulse    0
Calories    0
dtype: int64
```

Figure 5. 4 Result Chapter 2 Practicum 3

5.2.1.5 Chapter 3 Practicum 1

In this module, students will learn to extract specific data from a 3-dimensional array using array slicing. Then, they will display the specific data using the print() function.

```
import numpy as np

data_penjualan = np.array([
```

```
    [[10, 20, 30],
     [40, 50, 60],
     [70, 80, 90]],

    [[100, 110, 120],
     [130, 140, 150],
     [160, 170, 180]],

    [[190, 200, 210],
     [220, 230, 240],
     [250, 260, 270]]
])
data_spesifik = data_penjualan[0, 1, 1:3]
print(data_spesifik)
```

```
[50 60]
```

Figure 5. 5 Result Chapter 3 Practicum 1

5.2.1.6 Chapter 3 Practicum 2

In this module, students are taught to manipulate datasets in the data analysis process. In this topic, students will update the dataset by adding a new column. Then, students will calculate the total value of that column using the sum() function.

```
import pandas as pd

url                                                    =
"https://raw.githubusercontent.com/noora20FH/skripsi_noora2
023/master/data_toko.csv"  # Replace with your actual URL

def data_load():
  data_toko = pd.read_csv(url)
  return data_toko
def head_rows():
  return data_load().head()
```

```
def updated_data():
  df = data_load().copy()
  df["Total "] = df["Unit price"] * df["Quantity"]
  return df

def total_pendapatan():
  total_pendapatan = updated_data()['Total Revenue'].sum()
  return total_pendapatan

print(total_pendapatan())
```

```
307587.38
```

Figure 5. 6 Result Chapter 3 Practicum 2

5.2.1.7 Chapter 3 Practicum 3

In this module, Data Manipulation, students will learn how to effectively transform and analyze data. To utilize the argsort() function from the NumPy library, students will first convert the data_load() output from a Pandas Series to a NumPy array. Subsequently, they will apply the argsort() function to sort the data based on the 'Critic scores' column. Finally, the top 10 movies with the highest critic scores will be displayed using the print() function..

```
import pandas as pd

url                                                    =
"https://raw.githubusercontent.com/noora20FH/skripsi_noora2
023/main/clean_movie_data.csv"  # Replace with your actual
URL

def data_load():
  data_toko = pd.read_csv(url)
  return data_toko
```

```
def critic_scores():
  critic_scores = data_load()["Critic_Score"].to_numpy()
  return critic_scores

# Sort the movies by Critic Score in descending order
(highest to lowest)
def sorted_indices():
  sorted_indices = critic_scores().argsort()[::-1]
  return sorted_indices

def top_10_movies():
  top_10_movies = data_load().iloc[sorted_indices()[:10]]
  return top_10_movies

print(top_10_movies())
```

```
Top 10 Movies by Critic Score:
                                         Movie_Title  Release_Year    Genre  \
17  The Lord of the Rings: The Return of the King ...        2003.0  Fantasy
10                                          Inception        2010.0   Action
2                                     The Dark Knight        2008.0   Action
4                                       Pulp Fiction        1994.0    Crime
6       The Lord of the Rings: The Return of the King        2003.0  Fantasy
7                                       Forrest Gump        1994.0    Drama
9                                          Whiplash         2014.0    Drama
8                                        Fight Club         1999.0    Drama
11                                         The Good            NaN     1966
12                                       The Matrix         1999.0   Action
```

Figure 5. 7 Result Chapter 3 Practicum 3

5.2.1.8 Chapter 3 Practicum 4

In this module, students will learn about Sales Trends through measures of central tendency using the functions mean(), median(), and mode(). Additionally, they will learn about the Correlation Coefficient to understand whether there is a significant relationship between two variables using the function corr().

```
import pandas as pd
```

```
url                                                                =
'https://raw.githubusercontent.com/noora20FH/skripsi_noora2
023/main/purchases.csv'

def data_load():
  df = pd.read_csv(url)
  return df

def head_rows():
  return data_load().head()

print(data_load().describe())

print("Customer Age:")
print(f"  Mean: {data_load()['Age'].mean()}")
print(f"  Median: {data_load()['Age'].median()}")
print(f"  Standard Deviation: {data_load()['Age'].std()}")
print(f"  Skewness: {data_load()['Age'].skew()}")
print(f"   Quartiles:  {data_load()['Age'].quantile([0.25,
0.5, 0.75])}")

print("\nTotal Spent:")
print(f"  Mean: {data_load()['Total Spent (USD)'].mean()}")
print(f"       Median:      {data_load()['Total      Spent
(USD)'].median()}")
print(f"   Standard  Deviation:  {data_load()['Total  Spent
(USD)'].std()}")
print(f"       Skewness:      {data_load()['Total      Spent
(USD)'].skew()}")
print(f"      Quartiles:      {data_load()['Total      Spent
(USD)'].quantile([0.25, 0.5, 0.75])}")
```

```
correlation = data_load()["Age"].corr(data_load()["Total
Spent (USD)"])
print(f"\nCorrelation Coefficient: {correlation}")
```

```
       Customer ID    Age  Total Spent (USD)
count     50.00000  50.00          50.000000
mean      25.50000  34.54         114.000000
std       14.57738  13.07          52.459897
min        1.00000  15.00          30.000000
25%       13.25000  22.25          71.250000
50%       25.50000  33.50         110.000000
75%       37.75000  45.00         153.750000
max       50.00000  60.00         220.000000
Customer Age:
  Mean: 34.54
  Median: 33.5
  Standard Deviation: 13.069999921927455
  Skewness: 0.2325824223215271
  Quartiles: 0.25    22.25
0.50    33.50
0.75    45.00
Name: Age, dtype: float64
```

```
Total Spent:
  Mean: 114.0
  Median: 110.0
  Standard Deviation: 52.45989721993868
  Skewness: 0.19622312582543433
  Quartiles: 0.25     71.25
0.50    110.00
0.75    153.75
Name: Total Spent (USD), dtype: float64

Correlation Coefficient: 0.9489568324558989
```

Figure 5. 8 Result Chapter 3 Practicum 4

5.2.1.9 Chapter 3 Practicum 5

In this module, students will learn about the use of the mean(), median(), and mode() functions in the electronic sales case study.

```
import pandas as pd

url                                                    =
"https://raw.githubusercontent.com/noora20FH/skripsi_noora2
023/main/penjualan_elektronik.csv"

def load_data():

  data = pd.read_csv(url)
  return data
def mean_penjualan():
  mean_penjualan = load_data()['Jumlah Terjual'].mean()
  return mean_penjualan
```

```
def median_penjualan():
  median_penjlan = load_data()['Jumlah Terjual'].median()
  return median_penjualan

def mode_penjualan():
  mode_penjualan            =            load_data()['Jumlah
Terjual'].mode().iloc[0]
  return mode_penjualan

print(f"Rata-rata penjualan: {mean_penjualan()}")
print(f"Median penjualan: {median_penjualan()}")
print(f"Mode penjualan: {mode_penjualan()}")
```

```
Rata-rata penjualan: 16.09375
Median penjualan: 15.0
Mode penjualan: 8.0
```

Figure 5. 9 Result Chapter 3 Practicum 5

5.2.1.10 Chapter 4 Practicum 1

In this module, students will delve into the art of data visualization by creating insightful pie charts. By effectively utilizing the Matplotlib or Seaborn libraries, students will learn to represent data in a visually engaging and informative manner. They will explore how to customize pie charts with labels, colors, and titles to enhance clarity and interpretation. Through hands-on exercises, students will gain practical experience in creating pie charts that effectively convey the distribution of categorical data.

```
import pandas as pd
import matplotlib.pyplot as plt

def load_data():
  url = ""
  df = pd.read_csv(url)
  return df

fig_size= plt.figure()
```

```
get_population_col = load_data()['']

get_age_col = load_data()['']

plt.pie(get_population_col,                labels=get_age_col,
autopct='%1.1f%%', startangle=90)
plt.legend(title="Age Group")
plt.title('Age Distribution in the City')
plt.axis('equal')
plt.show()
```

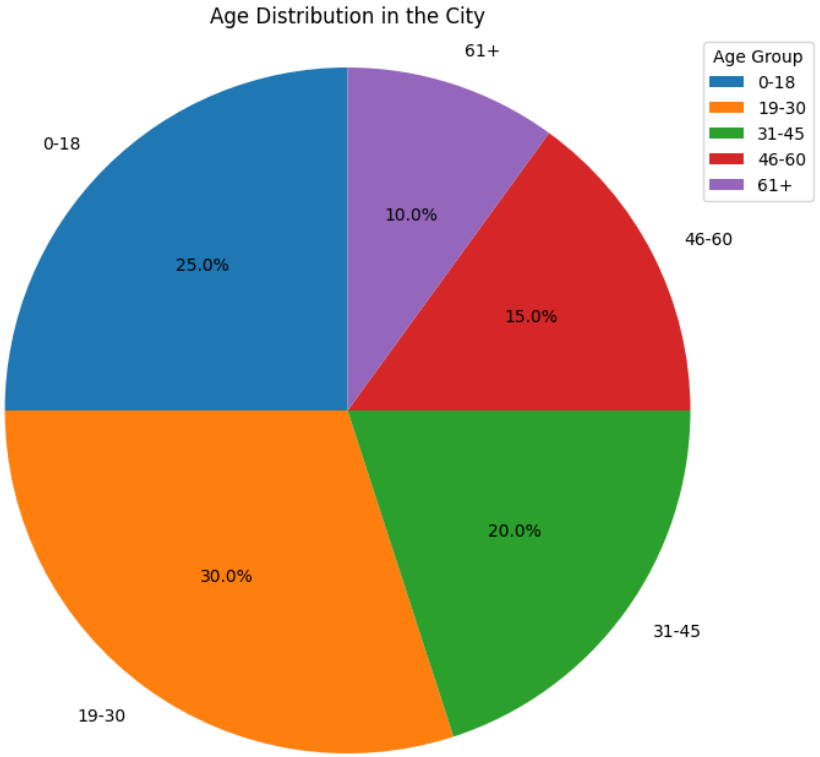
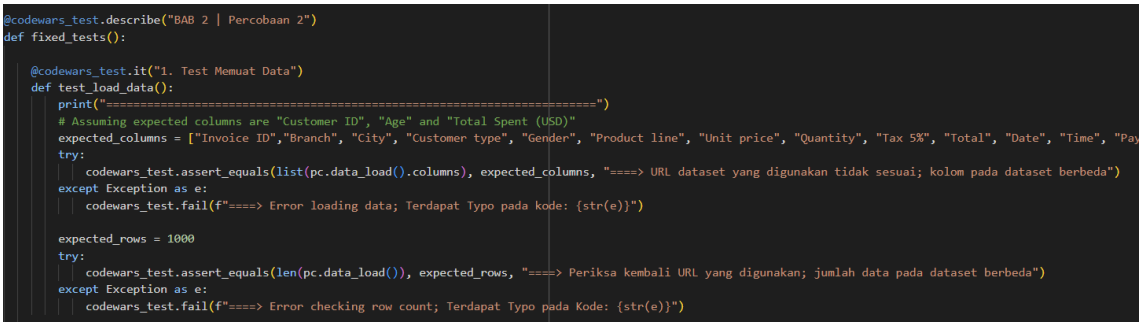


Figure 5. 10 Result Chapter 4 Practicum 1

5.2.2 *Test File* Python

A test file is a set of instructions designed to evaluate student responses to a specific document guide. Each test file focuses on a single Jobsheet. This file is saved as a .**py** file on Google Drive for students to download and upload into their Google Colab projects.

Student code submissions are tested using Codewars_test. Students are required to run these tests after completing each task. Students will test their own code using provided test files after completing each task. The goal is to ensure that the code they produce is correct and meets the specified criteria. The number of test codes depends on the test cases created for each experiment. Testing code typically consists of three parts: loading data test code, function test code, and output test code. The following is an example of a test file from the practical session in Chapter 2, Experiment 2:



```
@codewars_test.describe("BAB 2 | Percobaan 2")
def fixed_tests():

    @codewars_test.it("1. Test Memuat Data")
    def test_load_data():
        print("======================================================================")
        # Assuming expected columns are "Customer ID", "Age" and "Total Spent (USD)"
        expected_columns = ["Invoice ID","Branch", "City", "Customer type", "Gender", "Product line", "Unit price", "Quantity", "Tax 5%", "Total", "Date", "Time", "Pay
        try:
            codewars_test.assert_equals(list(pc.data_load().columns), expected_columns, "====> URL dataset yang digunakan tidak sesuai; kolom pada dataset berbeda")
        except Exception as e:
            codewars_test.fail(f"====> Error loading data; Terdapat Typo pada kode: {str(e)}")

        expected_rows = 1000
        try:
            codewars_test.assert_equals(len(pc.data_load()), expected_rows, "====> Periksa kembali URL yang digunakan; jumlah data pada dataset berbeda")
        except Exception as e:
            codewars_test.fail(f"====> Error checking row count; Terdapat Typo pada Kode: {str(e)}")
```

Figure 5. 11 Loading Data Test Case

In figure 5. 11, the test code runs a test to test whether the dataset has been loaded successfully by checking the line and column equality with the expected answer. Checking the column names in the actual loaded dataset against the expected column names stored in expected_columns. If the column names in the actual dataset differ, a message will appear saying "The URL of the dataset used is incorrect; the columns in the dataset differ". However, if there is an error in the code writing, codewars_test.fail will display an error message "Error Loading data; There is a Typo in the code: {location of the typo}.

"Meanwhile, to check the amount of data in a dataset, we use len(). If the length of the dataset is incorrect, a message will appear saying "Check the URL used, the number of datasets is different." However, if there is an error in the code, codewars_test.fail will display an error message "Error Loading data; There is a Typo in the code:

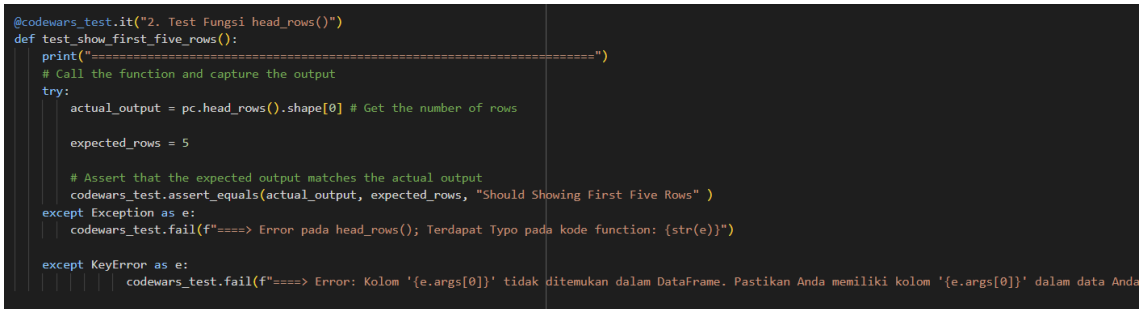


Figure 5. 12 Function Test Case

Figure 5. 12, the testcode runs the test to check whether the requested function is running properly, in this case the head_rows().head(). Try-except is used to handle errors that may occur during code execution. It allows a program to continue running even if an error occurs, instead of immediately stopping with a generic error message. The code that has the potential to cause an error is placed inside the try block. If no error occurs, the code within the try block will execute normally. If an error occurs within the try block, execution will immediately jump to the except block. Inside the except block, code is written to handle the error, such as displaying a more informative error message.

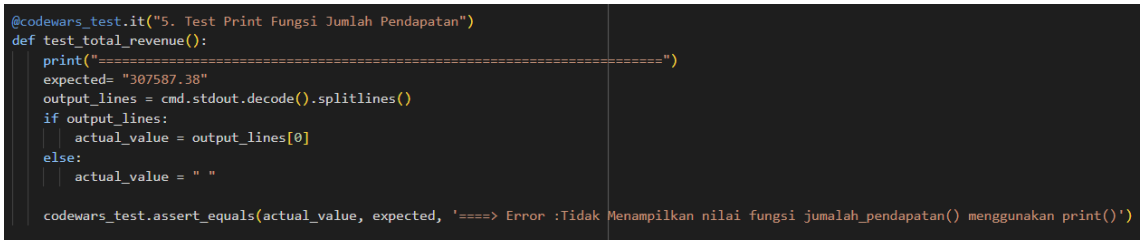


Figure 5. 13 Output Test Case

In Figure 5. 13, which represents the output test case, the output from the command prompt (cmd) will be captured as a string. This string will then be compared against the expected output. If there is no output from the cmd, an empty string will be assigned as the value, and a message will appear stating 'Does not display the function value using print(). The test codes check whether the output of the print() function already matches the answer keys.

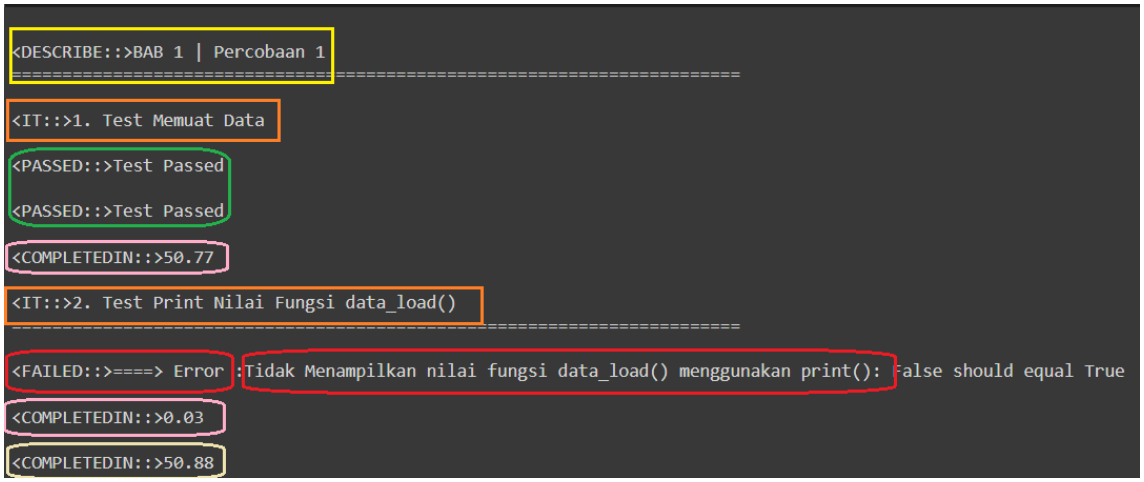


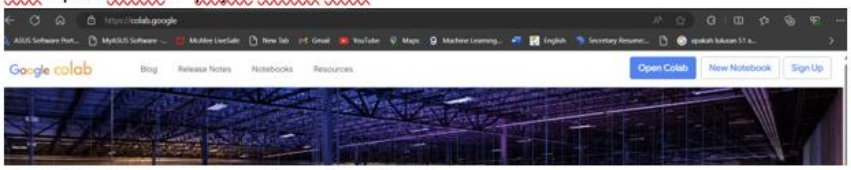
Figure 5. 14 Validation Result Output

The results of the test files that have been run can be seen in figure 5. 14. The output is structured with colored squares to denote different segments of the test execution. A yellow square marks the beginning of a test or experiment, labeled "Chapter 1 | Experiment 1." The subsequent orange square highlights the first test case, which involves loading data. A green square indicates the successful completion of this data loading test. The pink square likely represents the time elapsed for the data loading test in seconds. Following this, a red square signals a failed test case related to printing the value of the data_load() function. The error message specifies that the expected output of True was not achieved. Finally, a white square might denote the total time elapsed for both test cases, encompassing both successful and failed attempts.

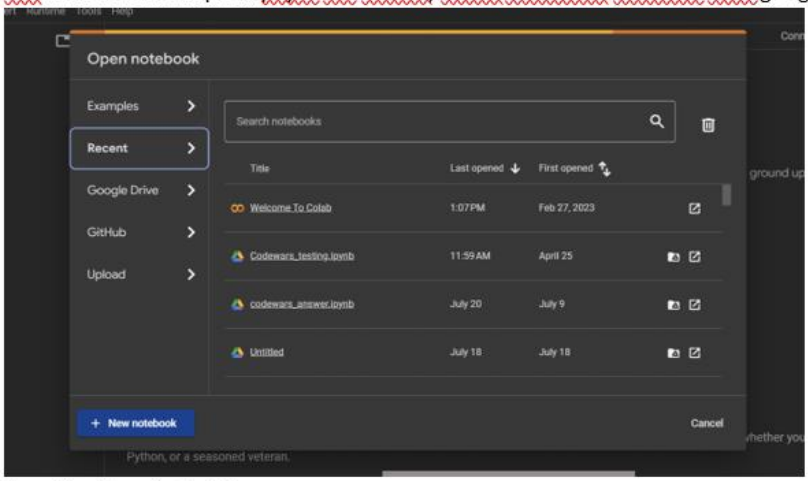
Here is the specific instructions for students to run validation can be found in the module:

1.3 Langkah Persiapan

1. Membuka Google Colab
   a. Buka Google Colaboratory dengan link berikut https://colab.research.google.com/ .
   b. Klik Open Colab di pojok kanan atas

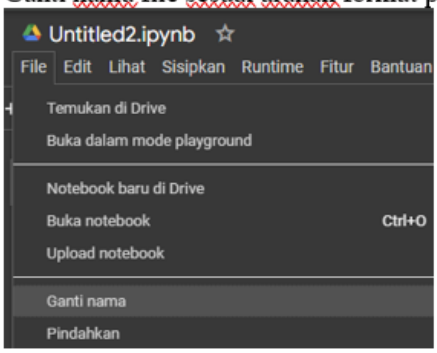


   c. Anda bisa login menggunakan akun Google.
   d. Klik New Notebook pada pojok kiri bawah, untuk membuka halaman baru google colab.



   e. Tampilan Google Colab.

   f. Ganti nama file sesuai arahan format pada praktikum



g. Setelah selesai mengerjakan praktikum, download file dengan format (.py)

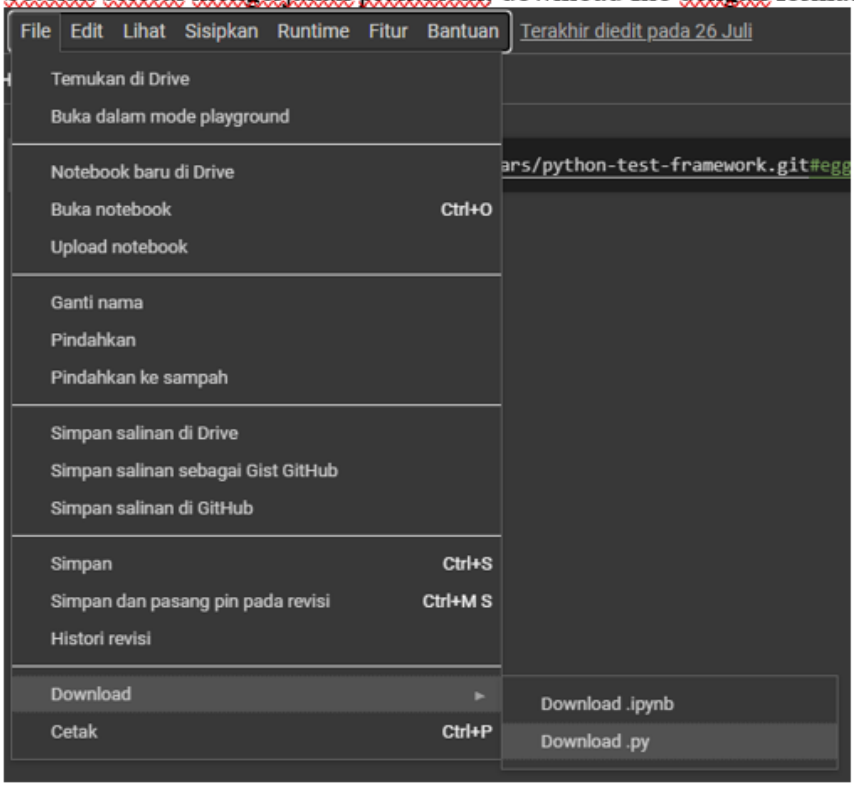


Figure 5. 15 Google Colab Preparation steps on the modul

5.3 Learning Website Implementation

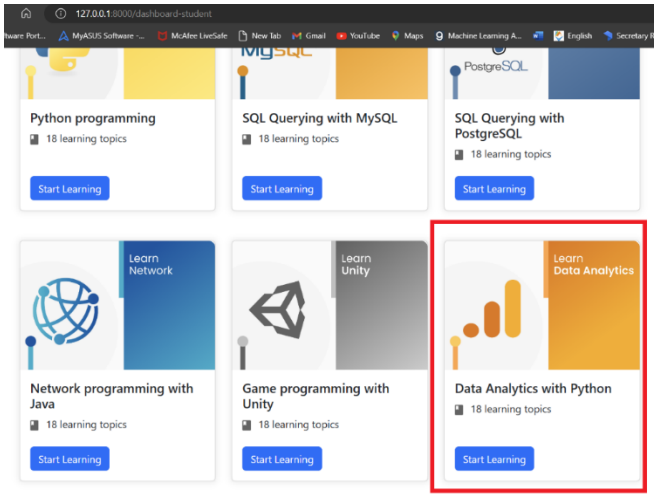


Figure 5. 16 Topic Data Analytics with Python on the website

On the figure 5. 16 that is the learning topic for Data Analytics with Python. Students can click the Start Learning button to open the learning page.

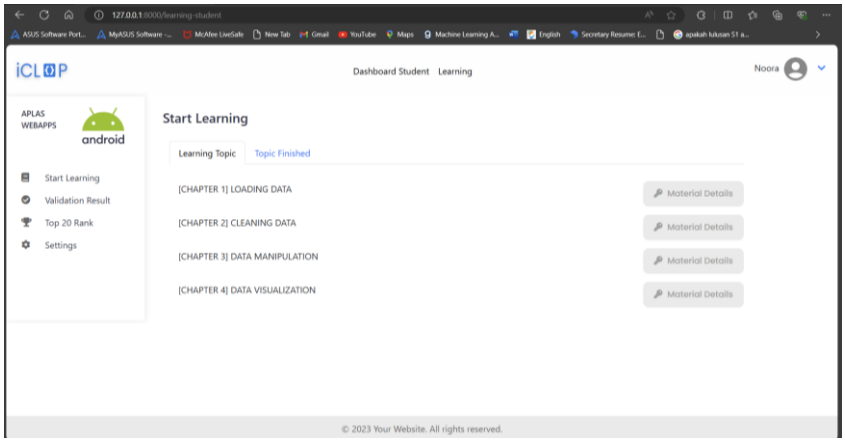


Figure 5. 17 Dashboard student

On the figure 5.17, after the student login and go to the Data Analytics with Python topic. Then the students will redirect to the student's dashboard page. On this page displayed the chapters provided in this Data Analytics topic. For each chapter, there is Materials details that contains some material and tasks related to the learning chapter.

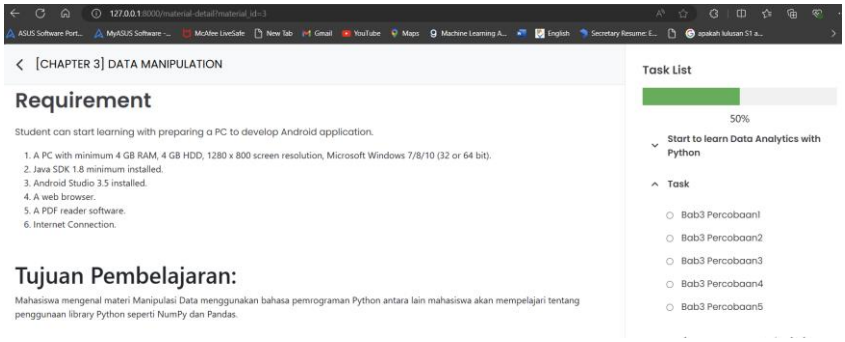


Figure 5. 18 Tasks page

On the figure 5. 18, The website provides some tasks related to each chapter. Students need to complete all the tasks.

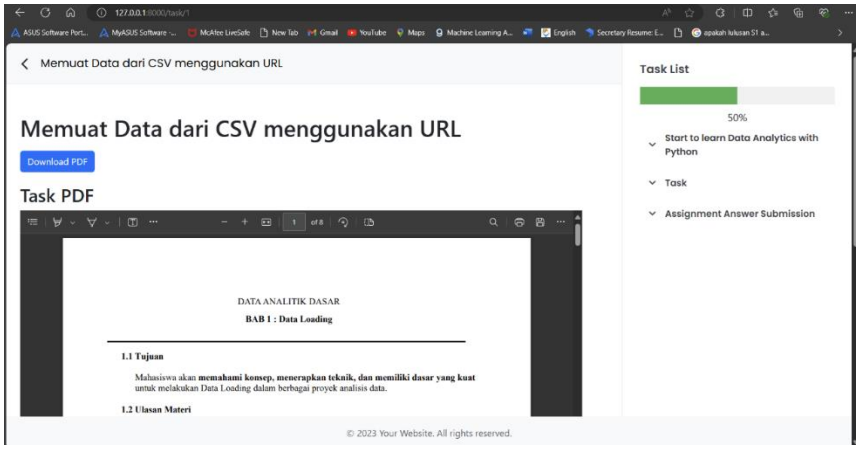


Figure 5. 19 Show PDF Modul

This page provides the pdf module for Data Analytics learning. The students can also download the pdf.

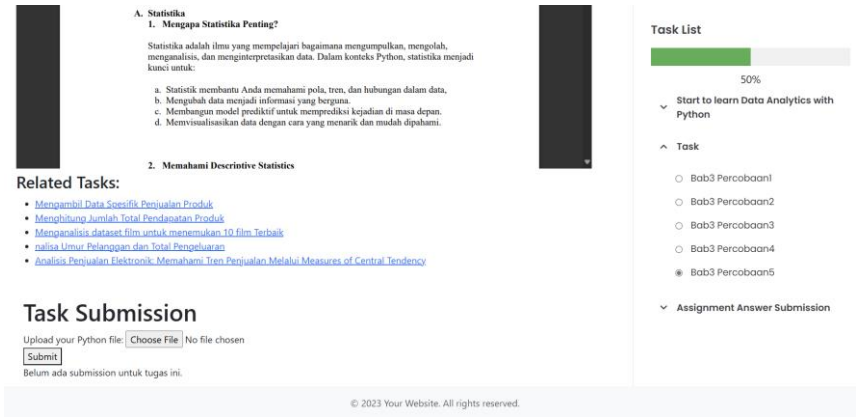


Figure 5. 20 Task Submission Form

On the figure 5. 20, there is task submission form for student to upload their code answer. The file uploaded by the students must be in a Python file. There is also radio button that ease student to go to another task in the same chapter, same with the Related Tasks that link to another task in the same chapter.

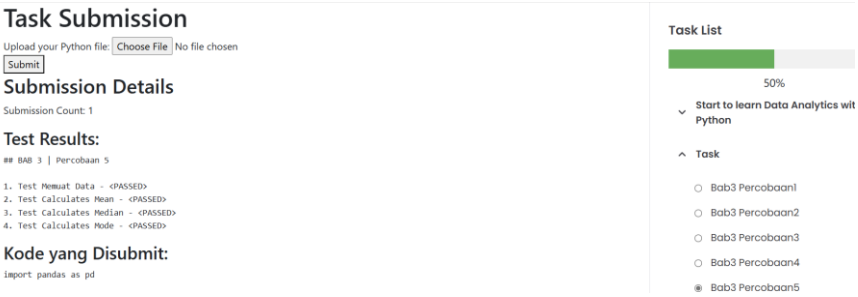


Figure 5. 21 Test Result

On the Figure 5. 21, the result from the validation showing the correctness of the submitted code.

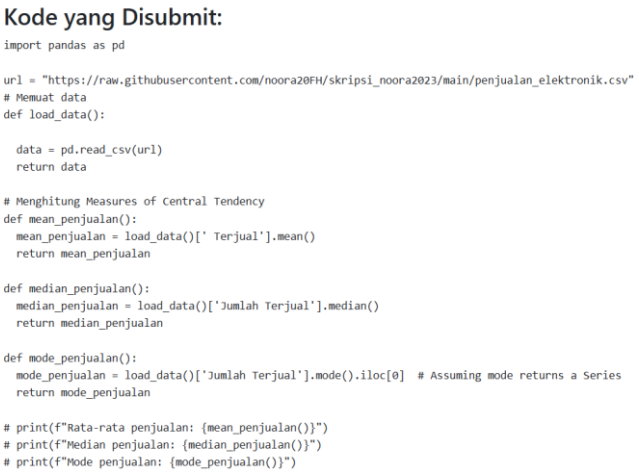


Figure 5. 22 Submitted Code Content

On the figure 5. 22 there is a section on this page that showing the content on the submitted code.

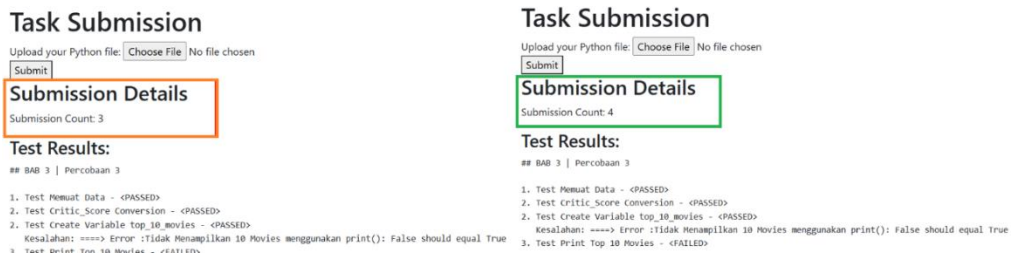


Figure 5. 23 Submission Count

The Submission Count section on the figure 5. 23 below, showing that for each submission for the same task will be counted. So, if the students submit their answer for the first time then it will be counted one, after that if the result test validation still showing

FAILED which mean the students need to fix and re-submit their answer, then the second submission will be counted two and so on.

5.4 Testing

User Testing

Tests were conducted on 40 students of the Faculty of Information Technology at Politeknik Negeri Malang. The task was completed within 3 days. Each student is given 10 experimental learning topics to complete. The students then began working on the tasks according to the instructions and conducted the tests independently. After finish all the jobsheet for 10 topics, the students are provided with Google Forms to record the completion of experimental work along with feedback. Students test on laptop devices and use Google Colaboratory to test the answer code with the test file. Test results from all students are analyzed to determine the difficulty level of the learning materials and to identify challenges faced by students. This information is then used to improve or refine the existing learning materials. In addition to improving the materials, information about student challenges can also be used to modify the way the material is delivered or to change the teaching methods.

The stages of conducting tests in Python learning with the topic of Data Analytics will be explained as follows.

1. If the automatic test fails, students need to fix their code based on the error message that appears and then re-run the test. Finally, if the test is successful, students can continue to the next module.
2. The first step students will take after completing their coding task is to download the test file provided. They can download it through the link provided in the learning module. The module will also explain how to place the test file in Google Colab.
3. After that, students can run the test file to test and verify whether the code they have written meets the requirements.
4. If the test they run fails, they need to fix their code, and if there are still errors or if they have completed all modules, they can submit their work.

CHAPTER VI. RESULTS AND DISCUSSION

Chapter ini menjelaskan tentang hasil pengujian dan pembahasan dari pengerjaan Pembelajaran Python dengan Topik Data Analytics menggunakan Codewars_test.

6.1 Testing Result

The test was conducted on 40 students of the Faculty of Information Technology of the State Polytechnic of Malang. Students tested the entire learning subject within three days. In Table 6.1 below is a list of names of students who have participated in the experiment:

Table 6. 1 List of 40 Student Names

No	Name	NIM	Major
1	Abdulilah Ali Qaid Al-shabany	2041720203	Jurusan Teknologi Informasi
2	Ahmad Farrel Sirajudin Zaidan	2041720238	Jurusan Teknologi Informasi
3	Akhmadheta Hafid Prasetyawan	2041720221	Jurusan Teknologi Informasi
4	Alifiyul Akyun	2041720036	Jurusan Teknologi Informasi
5	Amalia Nuraini	2041720160	Jurusan Teknologi Informasi
6	Andre Maulana Mustofa	2041720211	Jurusan Teknologi Informasi
7	Annisa Aulia Nadhila	2041720023	Jurusan Teknologi Informasi
8	Annisa Fitri Yuliandra	2041720123	Jurusan Teknologi Informasi
9	Atmayanti	2041720016	Jurusan Teknologi Informasi
10	Chan Paul Amol	2041720202	Jurusan Teknologi Informasi
11	Deatrisya Mirela Harahap	2041720013	Jurusan Teknologi Informasi
12	Della Jannata Febiana	204120034	Jurusan Teknologi Informasi
13	Dherisma Hanindita Utami	2041720018	Jurusan Teknologi Informasi
14	Elvira Sania Mufida	2041720080	Jurusan Teknologi Informasi
15	Eva Monika Septiana	2141764017	Jurusan Teknologi Informasi
16	FAHREZA PRIMA HAKIM	2041720210	Jurusan Teknologi Informasi
17	farah zulfa hamidah	2041720069	Jurusan Teknologi Informasi
18	Hafidz Irwan M	2141764079	Jurusan Teknologi Informasi
19	Hanif Widyantoro	2141764039	Jurusan Teknologi Informasi

20	Hilda Khoirotul Hidayah	2041720161	Jurusan Teknologi Informasi
21	Ibnu Khalis Rabbani	2041720159	Jurusan Teknologi Informasi
22	Iftitah Hidayati	2041720006	Jurusan Teknologi Informasi
23	Izzatun Nauly	2041720166	Jurusan Teknologi Informasi
24	Khofifah Amanda	2041720119	Jurusan Teknologi Informasi
25	Lelyta salsabila	1941720026	Jurusan Teknologi Informasi
26	M. Thosin Yuhaililul Hilmi	2141764032	Jurusan Teknologi Informasi
27	Maulana Bintang I.	2041720132	Jurusan Teknologi Informasi
28	Mochammad Hairullah	2041720074	Jurusan Teknologi Informasi
29	Mohammad Izamul Fikri Fahmi	2141720171	Jurusan Teknologi Informasi
30	Muhammad Ilham El Hakim	2041720162	Jurusan Teknologi Informasi
31	Nanda Shabrina Putri Kurnia	2141762064	Jurusan Teknologi Informasi
32	Neha Viranica Nauly	2141764127	Jurusan Teknologi Informasi
33	Nurlaily Asrobika	2041720172	Jurusan Teknologi Informasi
34	Rabiatul Fitra Aulia	2041720154	Jurusan Teknologi Informasi
35	Rofika Nur 'Aini	2041720099	Jurusan Teknologi Informasi
36	Rossa Akmalia	2041720219	Jurusan Teknologi Informasi
37	Salwa Zhafira Pratiwi Wahyudi	2041720138	Jurusan Teknologi Informasi
38	Shine Devi Oktaviana Ronix Syah Putri	2041720065	Jurusan Teknologi Informasi
39	Yaldika Putra Sinaga	2041720056	Jurusan Teknologi Informasi
40	Yoby Ryaian Pratama	2041720039	Jurusan Teknologi Informasi

There are 9 topics they need to complete. They do the practicum on Google Colaboratory and save the answer file as Python File (.py) for each topic. After they complete all of the topics practicum, then they need to upload directly on task submission section on the web page. As shown on figure 5.20.

Students' activities were timed using a timer to determine how long it took to finish each practicum topic. Following the completion of each assignment provided in the

Data Analytics Jobsheet, students filled out a specific form. The purpose of this form is to document the amount of time needed to finish each practicum topic.

6.2 Testing Result Discussions

The results of a trial conducted with 40 Information Technology students who completed a Data Analytics learning module showed a wide range of outcomes.

6.2.1 Time Completion for each Topic

On Figure 6.1 is a graph of the results of the test based on the student's work time of each learning topic. The bottom green line indicates the fastest time a student has spent completing each learning topic, the red line shows the longest time the student has had spent finishing each learning subject, whereas the blue line between the green and red lines represents the average length of time it takes a student to complete each learning theme.

Table 6. 2 Time Completion for each Topic

	b1_p1	b1_p2	b2_p3	b3_p1	b3_p2	b3_p3	b3_p4	b3_p5	b4_p1
max	313.2	420	600	450	660	738.6	752.4	462	900
average	114.78	178.08	241.515	237.09	310.71	508.365	480.555	336.15	416.415
min	60	60	90	80.4	60	60	120	120	180

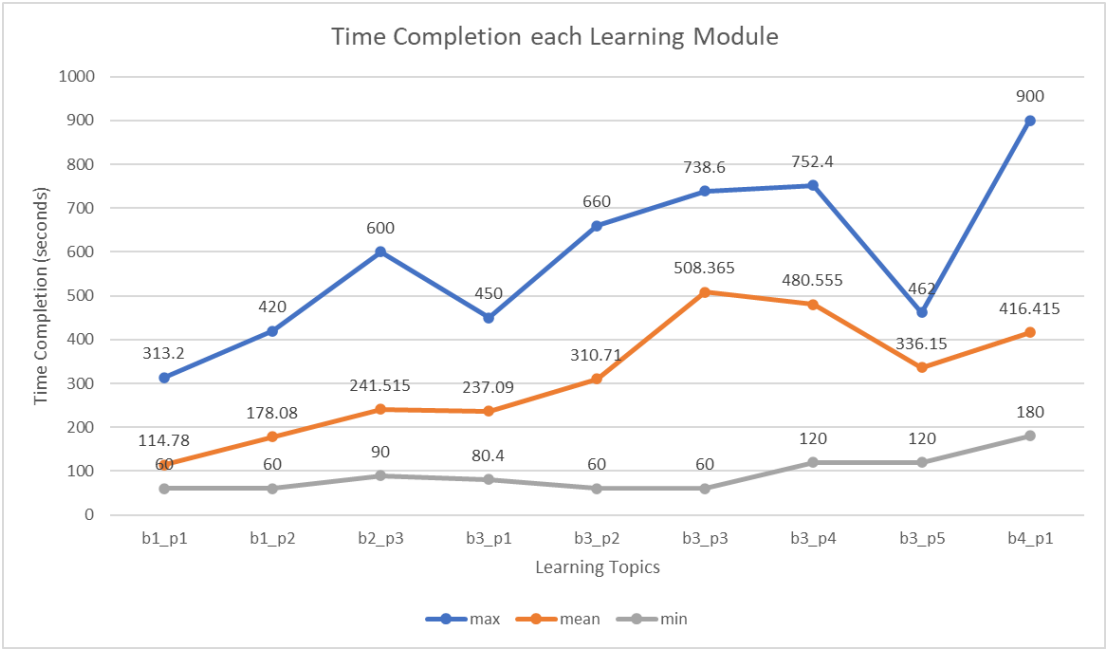


Figure 6. 1 Graphic of Student's Completion Time

From Figure 6. 1 we can assigning categories it can be concluded that the difficult experiments are chapter 3 experiments 3, chapter 3 experiments 4, and chapter 4 experiments 1. Chapter 3 experiment 3 is selecting specific movies from movies dataset. Chapter 3 experiments 4 is the analysis of the Measures of Variability in customer's Age. Chapter 4 experiment 1 is an visualizing the age distribution of population based on age range using a Pie Chart. The reason is that students are unable to understand the guidelines given on the practicum jobsheet.

While the less difficult/medium experiments are chapter 2 experiments 3, chapter 3 experiments 1, chapter 3, experiments 2, and chapter three experiments 5. Chapter 2 experiments 3 is handling missing values for calories data. Chapter 3 experiment 1 is cutting the Numpy array 3 dimensions. Chapter 3 experiment 2 is counting the total revenue. Chapter 3 of Experiment 5 is Electronic Trends Sales Analysis. The given material is more complicated than the previous material.

The last category is the experiment with the fastest completion time that is relatively easy to complete: chapter 1 practicum 1 and chapter 2 practicum 2. Chapter 1 Experiment 1 is to load a.csv dataset using URLs. Chapter 1 Practicum 2 is display the product of Online Store Data. The reason is that students can understand basic matters easily and can be followed by students well.

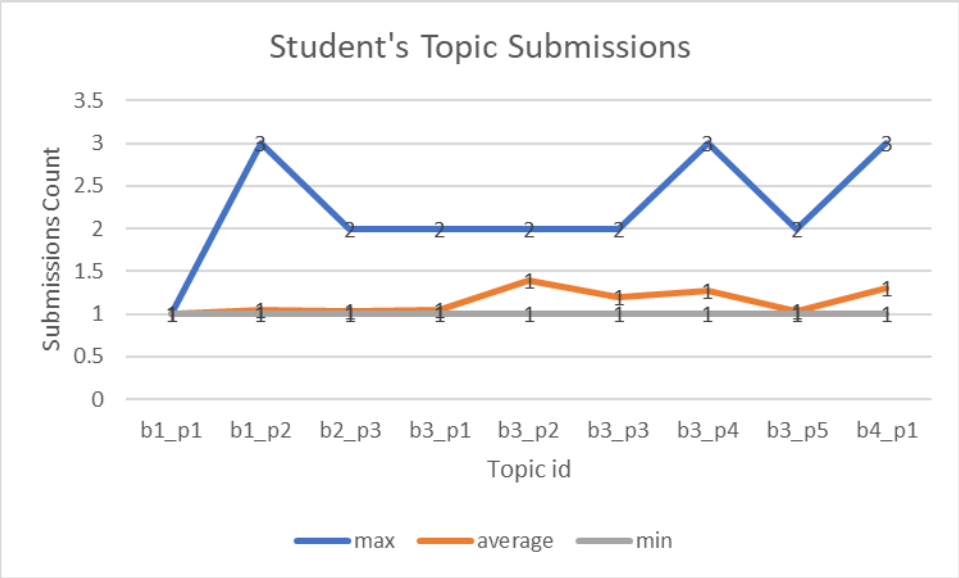


Figure 6. 2 Student's topic submission

From figure 6. 2, the submission counts vary across the topics. Some topics have received a high number of submissions, while others have received relatively few. maximum number of submissions for a single topic is 3. This indicates that some students are submitting multiple times for certain topics, which could be due to various reasons such as seeking feedback, clarification, or making revisions. The average number of submissions per topic falls between 1 and 2. This suggests that, on average, students are submitting 1-2 times per topic. The minimum number of submissions for a topic is 1. This means that there are no topics with zero submissions, and each topic has received at least one submission.

4.1.1 Students Feedback

The feedback from the students has been collected. There are positive comments and suggestions given. Here's a collection of feedback from students:

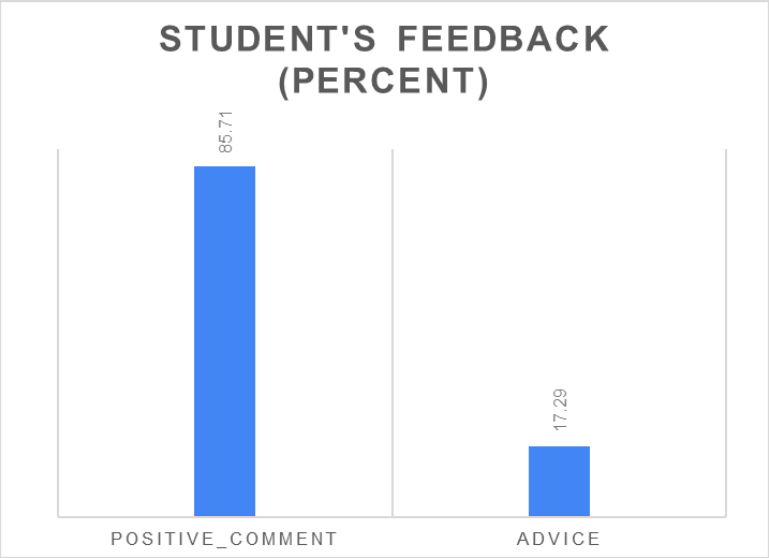


Figure 6. 3 Student's Feedback

The majority of students provided positive comments, constituting 85.71% of the total feedback. This indicates a high level of satisfaction with the practicum. While the overall sentiment is positive, 17.29% of students offered advice or suggestions for improvement. This suggests that there are areas where the practicum could be enhanced.

1. Positive comments:
   - It's good, because it's possible to know if the answers are right or wrong through testing.
   - the explanations and guides on the jobsheet make it very easy for the user to understand the material.
   - I hope it can be applied as a learning module.
   - The jobsheet allowed me to learn independently about data analytics.
2. Advice:
   - The consistency of the language used in the module to use one language or make two versions.
   - The module should be more detailed. It doesn't matter if the sentences are long as long as they are detailed and easy to understand.
   - Perhaps the module presentation could be made more creative to make it a bit more colorful.
   - the guidance in the jobsheet needs to be clarified for better understanding.

CHAPTER VII. CONCLUSION AND ADVICE

7.1 Conclusion

1. The implementation of test-driven development (TDD) using Codewars_test in a Python programming learning context, as demonstrated with the 40 student participants from the Department of Informatics Engineering at Politeknik Negeri Malang, has yielded positive results. The provided module materials were found to be effective in facilitating students' understanding of Data Analytics concepts and guiding them through practical tasks. The positive feedback and suggestions provided by the students further validate the effectiveness of the learning system. Students provided 85.71% positive feedback. Their comments indicate that the module materials were clear, easy to follow, and helpful in facilitating independent learning.

2. The findings of this study suggest that the integration of TDD with Codewars_test in a Data Analytics learning assistance system can be a valuable approach for enhancing students' learning experiences and outcomes. By providing immediate feedback, encouraging a structured approach to problem-solving, and fostering independent learning, TDD can help students develop a deeper understanding of the underlying concepts and improve their coding skills.

7.2 Advice

A suggestion that can help develop Python's basic learning system independently is as follows:

1. For further development of learning materials, it is expected that the document guides are available in two languages Indonesian and English.

2. Validation tests for outputs in the form of graphs can improve the accuracy of the tests.

References

Amarnath, R. (2023, January 11). *www.forbes.com*. Retrieved from Five Data Analytics Trends On Tap For 2023: https://www.forbes.com/sites/forbestechcouncil/2023/01/11/five-data-analytics-trends-on-tap-for-2023/?sh=3e026a716cfd

Bruel, J., Capozucca, A., Mazzara, M., Meyer, B., Naumchev, A., & Sadovykh, A. (2020). Applying Test-Driven Development for Improved Feedback and Automation of Grading in Academic Courses on Software Development. *FISEE 2019*, 310–323.

Codewars. (n.d.). *Pyhon Codewars Test Framework | The Codewars Docs*. Retrieved from docs.codewars.com: https://docs.codewars.com/languages/python/codewars-test/

Forcier, J., Bissex, P., & Chun, W. J. (2008). *Python Web Development with Django.* Addison-Wesley Professional.

Gartner. (2022, May 3). *www.gartner.com*. Retrieved from Gartner Predicts 65% of B2B Sales Organizations Will Transition from Intuition-Based to Data-Driven Decision Making by 2026: https://www.gartner.com/en/newsroom/press-releases/gartner-predicts-65--of-b2b-sales-organizations-will-transition-

Goasduff, L. (2022, April 5). *www.gartner.com*. Retrieved from 12 Data and Analytics Trends to Keep on Your Radar: https://www.gartner.com/en/articles/12-data-and-analytics-trends-to-keep-on-your-radar

Hasibuan, A. N. (2021). Pengujian dengan Unit Testing dan Test case pada Proyek Pengembangan Modul Manajemen Pengguna. *Automata*, 2(1).

Hnin, H. W., & Zaw, K. K. (2021). Element Fill-in-Blank Problems in Python Programming Learning Assistant . *IEEE*.

Mitra, J. (2023). Studying the Impact of Auto-Graders Giving Immediate Feedback in Programming Assignments. *SIGCSE 2023*, 388 - 394.

Mukhopadhyay, S. (2018). *Advance Data Analytics using Python.* Kolkata: APRESS.

Parsa, S. (2023). *Software Testing Automation: Testability Evaluation, Refactoring, Test Data Generation and Fault Localization.* Tehran: Springer.

Patnaik, R. (2019). Data Analytics and Visualization in Libraries. *12th International CALIBER-2019*.

Runkler, T. A. (2020). *Data Analitics Models and Algorithms for Intelligent Data Analysis.* München: Springer Vieweg.

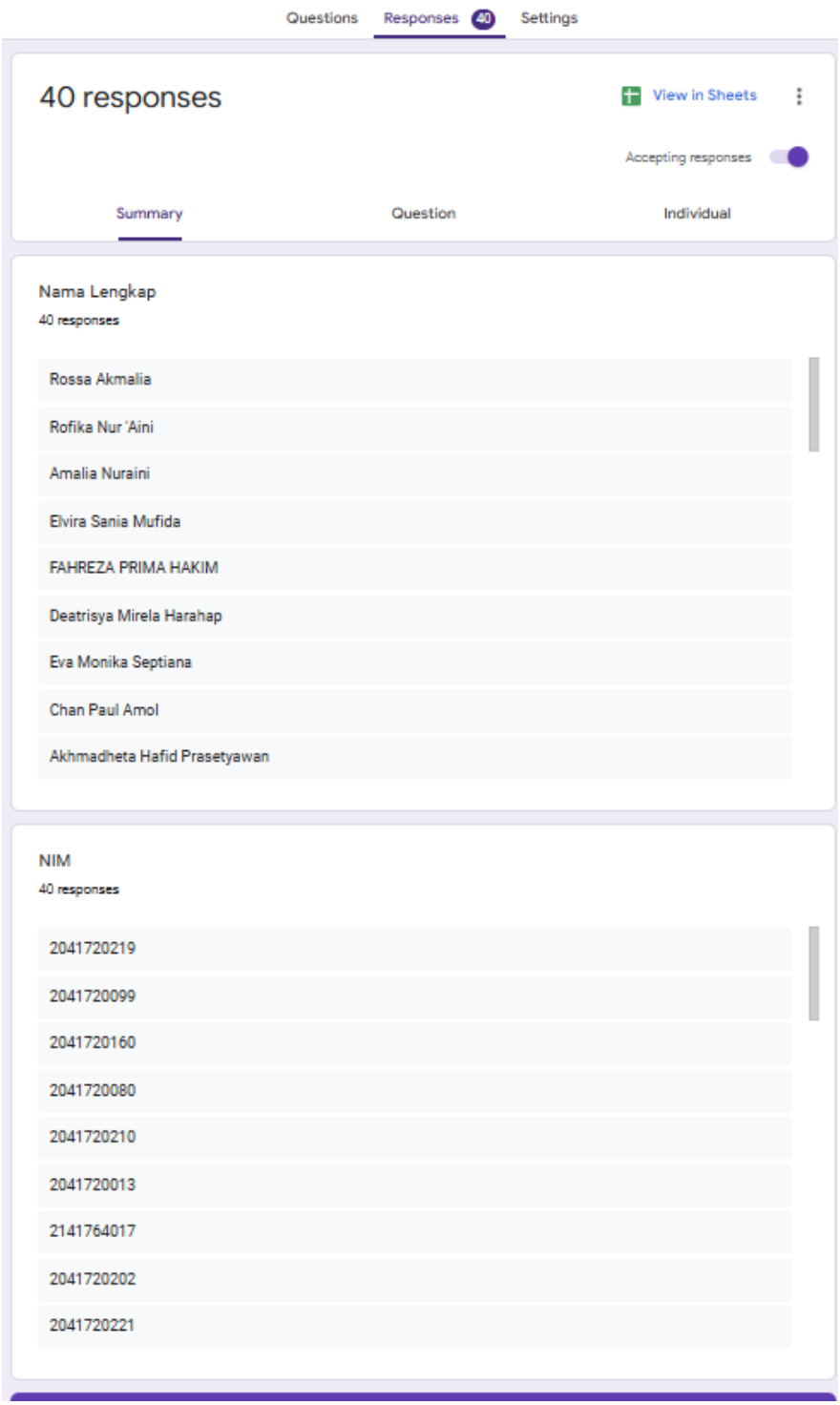
Watequlis Syaifudin, Y. F.-C. (2021). A Web-based Online Platform of Distribution, Collection, and Validation for assignment in Android Programming Learning Assistance System. *Engineering Letters*, 29(3), 1178–1193.

Yosifova, A. (2023, Agustus 10). *365datascience.com*. Retrieved from The Data Analyst Job Outlook in 2023: Research on 1,000+ LinkedIn Job Postings: https://365datascience.com/career-advice/data-analyst-job-outlook/

Yulvarisma, Q. (2022). *IMPLEMENTASI PEMBELAJARAN DASAR PEMROGRAMAN PADA PYTHON PROGRAMMING LEARNING ASSISTANCE SYSTEM.* Diploma thesis, Jurusan Teknologi Informasi.

Apendices

4. Questionnaire

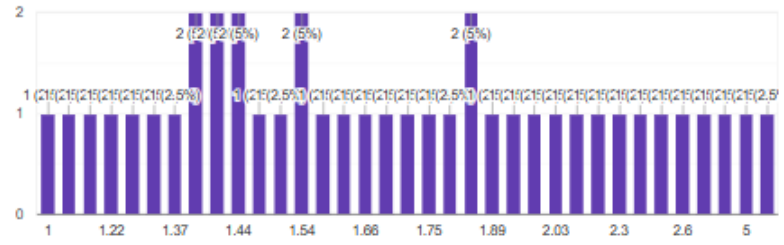


Waktu Penyelesaian

b1p1

Copy

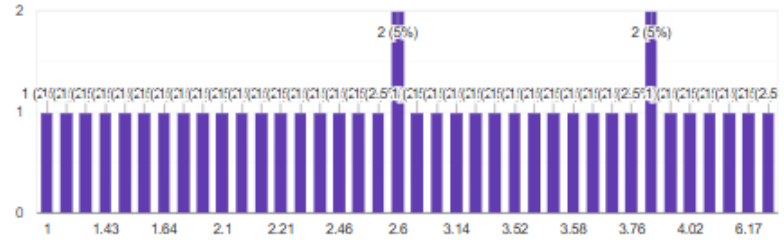
40 responses



b1p2

Copy

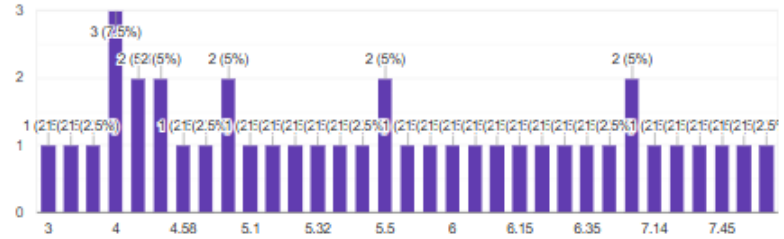
40 responses



b2p2

Copy

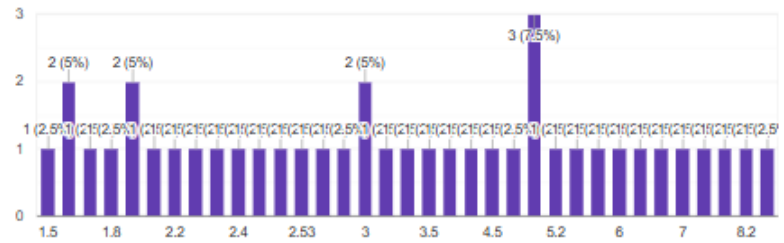
40 responses



b2p3

Copy

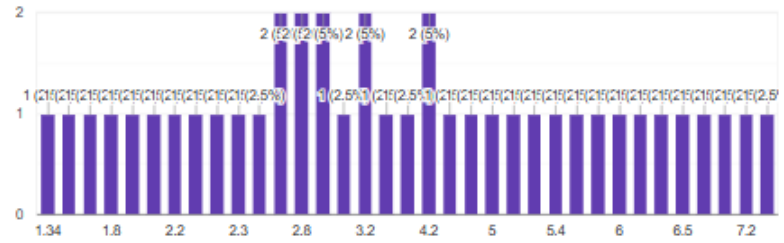
40 responses



b3p1

Copy

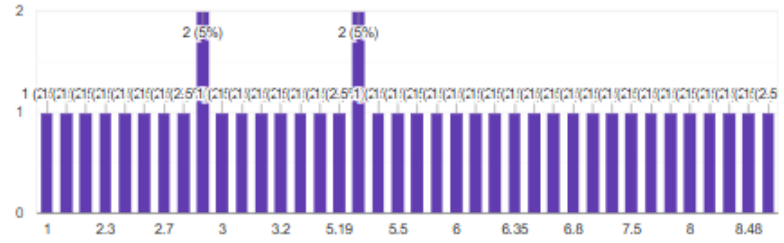
40 responses



b3p2

Copy

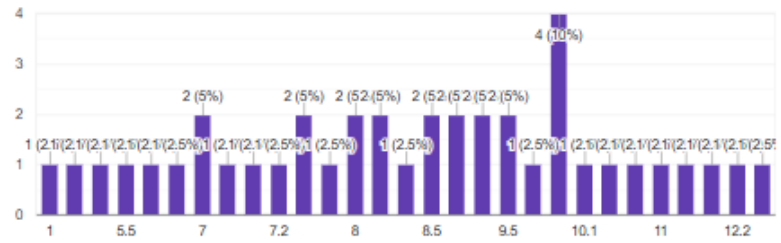
40 responses



b3p3

Copy

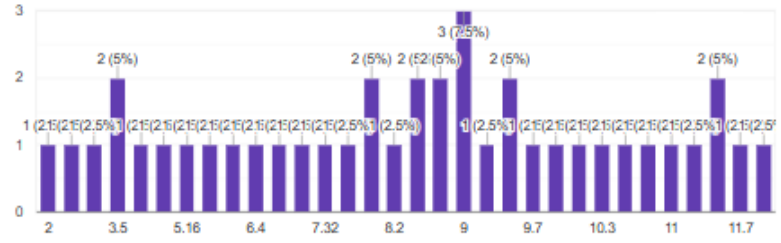
40 responses



b3p4

Copy

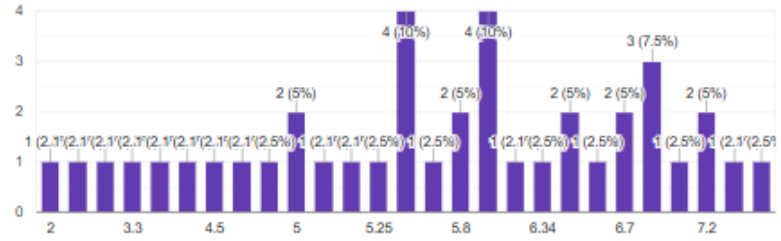
40 responses

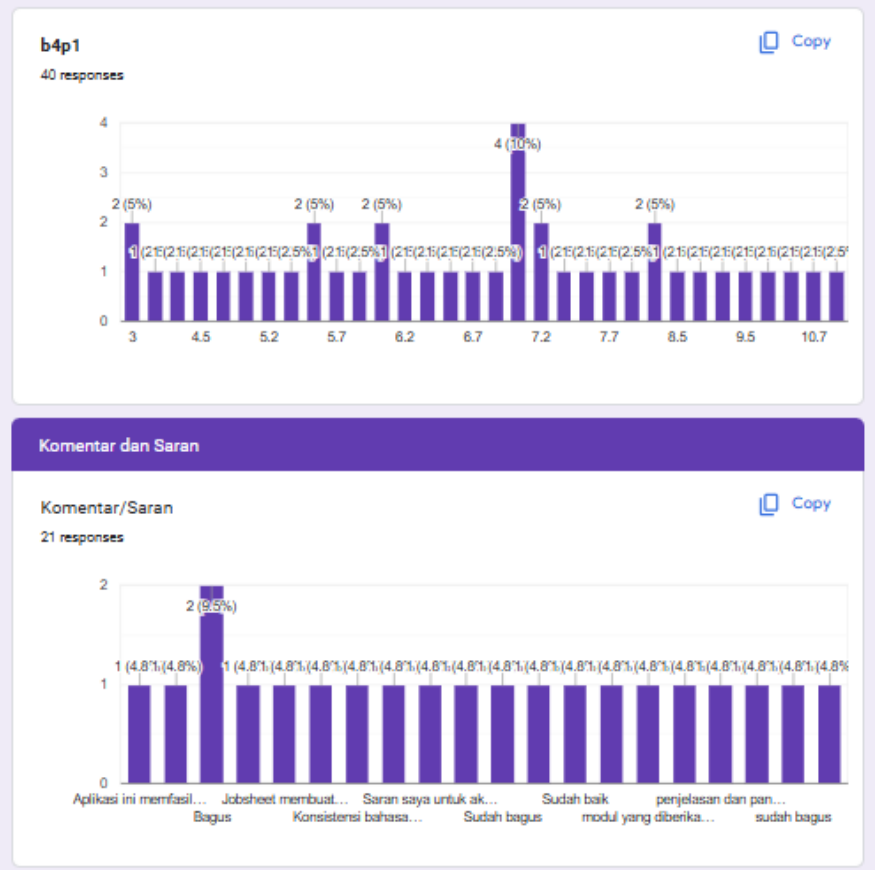


b3p5

Copy

40 responses





2. Test Code Validation

2.1 Chapter 1 Practicum 1 Test file

test_bab1_percobaan1.py

```python
import random
import sys
from pathlib import Path
import numpy as np
import subprocess
import importlib
import codewars_test
import pandas as pd

# Update paths and filename
path_answer = "/content/"  # changed
filename = "answer_bab1_percobaan1.py"  # changed


# Import the answer module
pc = importlib.import_module(filename[:-3], ".")
cmd = subprocess.run([sys.executable, f"{path_answer}{filename}"], capture_output=True)  # changed

@codewars_test.describe("BAB 1 | Percobaan 1")
def fixed_tests():
    print("\n=================================================================")
    @codewars_test.it("1. Test Memuat Data")
    def test_load_data():
        # Assuming expected columns are "Customer ID", "Age" and "Total Spent (USD)"
        expected_columns = ['Customer ID', 'Age', 'Total Spent (USD)']
        codewars_test.assert_equals(list(pc.data_load().columns), expected_columns, "====> URL dataset yang digunakan tidak sesuai; kolom pada dataset berbeda")

        expected_rows = 50
        codewars_test.assert_equals(len(pc.data_load()), expected_rows, "====> Periksa kembali URL yang digunakan; jumlah data pada dataset berbeda")

    @codewars_test.it("2. Test Print Nilai Fungsi data_load()")
    def test_total_revenue():
        print("\n=================================================================")
        output_lines = cmd.stdout.decode().splitlines()
        codewars_test.assert_equals(bool(output_lines), True, '====> Error :Tidak Menampilkan nilai fungsi data_load() menggunakan print()')
```

2.2 Chapter 1 Practicum 2 Test file

```python
import sys
from pathlib import Path
import numpy as np
import subprocess
import importlib
import codewars_test
import pandas as pd

# Update paths and filename
path_answer = "/content/"  # changed
filename = "answer_bab1_percobaan2.py"  # changed


# Import the answer module
pc = importlib.import_module(filename[:-3], ".")
cmd = subprocess.run([sys.executable, f"{path_answer}{filename}"], capture_output=True)  # changed

@codewars_test.describe("BAB 1 | Percobaan 2")
def fixed_tests():

    @codewars_test.it("1. Test Memuat Data ")
    def test_load_data():
        print("=====================================================================")
        try:
            expected_columns = ['Order ID', 'Customer ID', 'Product ID', 'Product Name', 'Price',
                                'Order Date', 'Quantity']
            data = pc.data_load()
            codewars_test.assert_equals(list(data.columns), expected_columns, "====> URL dataset yang digunakan tidak sesuai; kolom pada dataset berbeda")
            expected_rows = 100
            codewars_test.assert_equals(len(data), expected_rows, "====> Periksa kembali URL yang digunakan; jumlah data pada dataset berbeda")
        except AttributeError as e:
```

```python
        except AttributeError as e:
            if "module 'pandas' has no attribute 'read_csv'" in str(e):
                codewars_test.fail("====> Error: Fungsi data_load() tidak menggunakan fungsi read_csv yang benar. Pastikan Anda menggunakan fungsi read_csv untuk mem
            else:
                codewars_test.fail(f"====> Error: Fungsi data_load() tidak ditemukan. Pastikan Anda memiliki fungsi data_load() dalam modul Anda. {str(e)}")
        except pd.errors.EmptyDataError as e:
            codewars_test.fail("====> Error: Data kosong. Pastikan Anda memiliki data yang valid.")
        except pd.errors.ParserError as e:
            codewars_test.fail("====> Error: Kesalahan parsing data. Pastikan Anda memiliki format data yang benar.")
        except Exception as e:
            codewars_test.fail(f"====> Error: An unexpected error occurred in data_load(). {str(e)}")

    @codewars_test.it("2. Test Fungsi sample_rows()")
    def test_sample_rows():
        print("=====================================================================")
        try:
            sample = pc.sample_rows()
            expected_len = 1
            codewars_test.assert_equals(len(sample), expected_len, "Error :Tidak menggunakan fungsi .sample(); Sehingga memiliki lebih dari 1 data; data yang anda mi
            assert isinstance(sample, pd.DataFrame), "Sample harus memiliki tipe data DataFrame"
        except AttributeError as e:
            codewars_test.fail(f"====> Error: Fungsi sample_rows() tidak ditemukan. Pastikan Anda memiliki fungsi sample_rows() dalam modul Anda. {str(e)}")
        except Exception as e:
            codewars_test.fail(f"====> Error: An unexpected error occurred in sample_rows(). {str(e)}")

    @codewars_test.it("3. Test Print Nilai Fungsi sample_rows()")
    def test_total_revenue():
        print("=====================================================================")
        output_lines = cmd.stdout.decode().splitlines()
        codewars_test.assert_equals(bool(output_lines), True, '====> Error :Tidak Menampilkan nilai fungsi sample_rows() menggunakan print()')
```

2.3 Chapter 2 Practicum 2 Test file

```python
import sys
from pathlib import Path
import numpy as np
import subprocess
import importlib
import codewars_test
import pandas as pd

# Update paths and filename
path_answer = "/content/"  # changed
filename = "answer_bab2_percobaan2.py"  # changed

# Import the answer module
pc = importlib.import_module(filename[:-3], ".")
cmd = subprocess.run([sys.executable, f"{path_answer}{filename}"], capture_output=True)  # changed

@codewars_test.describe("Bab 2 | Percobaan 2")
def fixed_tests():

    @codewars_test.it("1. Test Memuat Data")
    def test_load_data():
        print("=====================================================================")

            # Assuming expected columns are "Customer ID", "Age" and "Total Spent (USD)"
        expected_columns = ['BOROUGH', 'NEIGHBORHOOD', 'BUILDING CLASS CATEGORY',
                            'TAX CLASS AT PRESENT', 'BLOCK', 'LOT', 'EASE-MENT',
                            'BUILDING CLASS AT PRESENT', 'ADDRESS', 'APARTMENT NUMBER', 'ZIP CODE',
                            'RESIDENTIAL UNITS', 'COMMERCIAL UNITS', 'TOTAL UNITS',
                            'LAND SQUARE FEET', 'GROSS SQUARE FEET', 'YEAR BUILT',
                            'TAX CLASS AT TIME OF SALE', 'BUILDING CLASS AT TIME OF SALE',
```

```python
        try:
            codewars_test.assert_equals(list(pc.load_data().columns), expected_columns, "====> URL dataset yang digunakan tidak sesuai; kolom pada dataset berbeda")
        except Exception as e:
            codewars_test.fail(f"====> Error loading data; Terdapat Typo pada kode: {str(e)}")

        expected_rows = 84548
        try:
            codewars_test.assert_equals(len(pc.load_data()), expected_rows, "Dataframe should have expected rows")
        except Exception as e:
            codewars_test.fail(f"====> Error checking row count; Terdapat Typo pada Kode: {str(e)}")
        print("")

    # Test data cleaning or processing (replace with your specific logic)
    @codewars_test.it('2. Test Drop Columns')
    def test_data_cleaning():
        print("=====================================================================")
        try:

            expected_columns= 16
            codewars_test.assert_equals(len(pc.clean_columns().columns), expected_columns, "Dataframe should have expected rows")
        except Exception as e:
            codewars_test.fail(f"====> Error: {str(e)}")
        print(" ")

    @codewars_test.it('3. Test Clean Columns Name')
    def test_clean_columns_name():
        print("=====================================================================")

        expected_columns = ['BOROUGH', 'NEIGHBORHOOD', 'BUILDING_CLASS_CATEGORY',
                            'BUILDING_CLASS_AT_PRESENT', 'ADDRESS', 'APARTMENT_NUMBER', 'ZIP_CODE',
                            'RESIDENTIAL_UNITS', 'COMMERCIAL_UNITS', 'TOTAL_UNITS',
```

2.4 Chapter 2 Practicum 3 Test file

```python
import sys
from pathlib import Path
import numpy as np
import subprocess
import importlib
import codewars_test
import pandas as pd

# Update paths and filename
path_answer = "/content/"  # changed
filename = "answer_bab2_percobaan3.py"  # changed

# Import the answer module
pc = importlib.import_module(filename[:-3], ".")
cmd = subprocess.run([sys.executable, f"{path_answer}{filename}"], capture_output=True)  # changed

@codewars_test.describe("BAB 2 | Percobaan 3")
def fixed_tests():

    @codewars_test.it("1. Test Memuat Data")
    def test_load_data():
        print("======================================================================")
        # Assuming expected columns are "Customer ID", "Age" and "Total Spent (USD)"
        expected_columns = ['Duration', 'Pulse', 'Maxpulse', 'Calories']
        try:
            codewars_test.assert_equals(list(pc.load_data().columns), expected_columns, "====> URL dataset yang digunakan tidak sesuai; kolom pada dataset berbeda")
        except Exception as e:
            codewars_test.fail(f"====> Error loading data; Terdapat Typo pada kode: {str(e)}")
```

```python
        try:
            codewars_test.assert_equals(len(pc.load_data()), expected_rows, "====> Periksa kembali URL yang digunakan; jumlah data pada dataset berbeda")
        except Exception as e:
            codewars_test.fail(f"====> Error checking row count; Terdapat Typo pada Kode: {str(e)}")

        print("")

    @codewars_test.it("2. Test Print Total Nilai yang Hilang")
    def test_total_revenue():
        print("======================================================================")
        output_lines = cmd.stdout.decode().splitlines()
        codewars_test.assert_equals(bool(output_lines), True, '====> Error :Tidak Menampilkan total nilai hilang print()')


    @codewars_test.it("2. Test Mengisi Nilai yang Hilang ")
    def test_no_missing_values():
        print("======================================================================")

        # Replace 'Duration', 'Pulse', 'Maxpulse', 'Calories' with your desired columns
        expected_columns_no_null = ['Duration', 'Pulse', 'Maxpulse', 'Calories']
        try:
            for col in expected_columns_no_null:
                codewars_test.assert_equals(pc.updated_data()[col].isnull().sum(), 0, f"Column '{col}' should not have missing values")
        except Exception as e:
            codewars_test.fail(f"====> Error pada update_data(); Terdapat Typo pada kode function: {str(e)}")
        except NameError as e:
            codewars_test.fail(f"====> Error pada update_data(); Terdapat Typo pada kode function: {str(e)}")
        except SyntaxError as e:
            codewars_test.fail(f"====> Error pada update_data(); Terdapat Typo pada kode function: {str(e)}")
```

2.5 Chapter 3 Practicum 1 Test file

```python
import sys
from pathlib import Path
import numpy as np
import subprocess
import importlib
import codewars_test

path_answer = "/content/" #changed
filename = "answer_bab3_percobaan1.py" #changed
pc = importlib.import_module(filename[:-3], ".") #changed

cmd = subprocess.run([sys.executable, f"{path_answer}{filename}"], capture_output=True)#changed

# Test Suite: BAB 2
@codewars_test.describe('BAB 2 | Percobaan 1')
def percobaan1():

    # Test Array Dimension
    @codewars_test.it('1. Test Dimensi Array-')
    def test_array_dimension():
        print("\n=================================================================")
        try:
            # Check if array is a NumPy array
            if not isinstance(pc.data_penjualan, np.ndarray):
                raise AssertionError("Variabel array haruslah berupa NumPy array")

            # Check array dimension (should be 3) and sizes (3, 3, 3)
            expected_shape = (3, 3, 3)
            codewars_test.assert_equals( pc.data_penjualan.shape, expected_shape,'Error : Ukuran array tidak sesuai dengan ketentuan 3x3x3')
        except Exception as e:
            codewars_test.fail(f"====> Error pada updated_data(); Terdapat Kolom dengan penulisan nama yang salah, yaitu; function: {str(e)}")
```

```python
        except KeyError as e:
            codewars_test.fail(f"====> Error: Kolom '{e.args[0]}' tidak ditemukan dalam DataFrame. Pastikan Anda memiliki kolom '{e.args[0]}' dalam data Anda.")
        except NameError as e:
            codewars_test.fail(f"====> Error: Kolom '{e.args[0]}' tidak ditemukan dalam DataFrame. Pastikan Anda memiliki kolom '{e.args[0]}' dalam data Anda.")

    # Test Array Slicing
    @codewars_test.it('2. Test Pemotongan Array-')
    def test_array_slicing():
        print("\n=================================================================")
        expected_array_slicing = np.array([50, 60])
        try:
            actual_array_slicing = pc.data_spesifik
            # Check element-wise equality using np.array_equal
            codewars_test.assert_equals(True, np.array_equal(expected_array_slicing, actual_array_slicing), 'Error : Hasil pemotongan array tidak sesuai; ')
        except Exception as e:
            codewars_test.fail(f"====> Error Terdapat Variabel dengan penulisan nama yang salah, yaitu; function: {str(e)}")
        except KeyError as e:
            codewars_test.fail(f"====> Error: Kolom '{e.args[0]}' tidak ditemukan dalam DataFrame. Pastikan Anda memiliki kolom '{e.args[0]}' dalam data Anda.")

        print("""NOTE:
        Potong pada array kotak index-0,
        ambil baris index-1,
        kolom index-1 hingga index-2""")

    # Test Output Type of Sliced Array
    @codewars_test.it('3. Test Tipe Hasil Pemotongan Array-')
    def test_output_type():
        print("\n=================================================================")
        expected_type = np.ndarray  # Expected type
```

```python
        try:
            # Assuming pc.array_potongan is the sliced array from answer.py
            codewars_test.assert_equals(expected_type, type(pc.data_spesifik), 'Error : Tipe hasil pemotongan array harus berupa NumPy ndarray')
        except Exception as e:
            codewars_test.fail(f"====> Error pada penamaan variabel; function: {str(e)}")
        except KeyError as e:
            codewars_test.fail(f"====> Error: Kolom '{e.args[0]}' tidak ditemukan dalam DataFrame. Pastikan Anda memiliki kolom '{e.args[0]}' dalam data Anda.")


    @codewars_test.it('4. Test Print Hasil Pemotongan Array-')
    def test_output_type():
        print("\n=================================================================")
        expected = "[50 60]"
        output_lines = cmd.stdout.decode().splitlines()
        if output_lines:
            actual_value = output_lines[0]
        else:
            actual_value = " "

        codewars_test.assert_equals(actual_value, expected, 'Error :Tidak Menampilkan nilai data_spesifik menggunakan print()')
```

2.6 Chapter 3 Practicum 2 Test file

```python
import sys
from pathlib import Path
import numpy as np
import subprocess
import importlib
import codewars_test
import pandas as pd

# Update paths and filename
path_answer = "/content/"  # changed
filename = "answer_bab3_percobaan2.py"  # changed

# Import the answer module
pc = importlib.import_module(filename[:-3], ".")
cmd = subprocess.run([sys.executable, f"{path_answer}{filename}"], capture_output=True)  # changed

@codewars_test.describe("BAB 2 | Percobaan 2")
def fixed_tests():

    @codewars_test.it("1. Test Memuat Data")
    def test_load_data():
        print("======================================================================")
        # Assuming expected columns are "Customer ID", "Age" and "Total Spent (USD)"
        expected_columns = ["Invoice ID","Branch", "City", "Customer type", "Gender", "Product line", "Unit price", "Quantity", "Tax 5%", "Total", "Date", "Time", "Pa
        try:
            codewars_test.assert_equals(list(pc.data_load().columns), expected_columns, "====> URL dataset yang digunakan tidak sesuai; kolom pada dataset berbeda")
        except Exception as e:
            codewars_test.fail(f"====> Error loading data; Terdapat Typo pada kode: {str(e)}")

        expected_rows = 1000
```

```python
            codewars_test.assert_equals(len(pc.data_load()), expected_rows, "====> Periksa kembali URL yang digunakan; jumlah data pada dataset berbeda")
        except Exception as e:
            codewars_test.fail(f"====> Error checking row count; Terdapat Typo pada Kode: {str(e)}")

    @codewars_test.it("2. Test Fungsi head_rows()")
    def test_show_first_five_rows():
        print("======================================================================")
        # Call the function and capture the output
        try:
            actual_output = pc.head_rows().shape[0] # Get the number of rows

            expected_rows = 5

            # Assert that the expected output matches the actual output
            codewars_test.assert_equals(actual_output, expected_rows, "Should Showing First Five Rows" )
        except Exception as e:
            codewars_test.fail(f"====> Error pada head_rows(); Terdapat Typo pada kode function: {str(e)}")

        except KeyError as e:
                    codewars_test.fail(f"====> Error: Kolom '{e.args[0]}' tidak ditemukan dalam DataFrame. Pastikan Anda memiliki kolom '{e.args[0]}' dalam data


    @codewars_test.it("3. Test Membuat Kolom Revenue ")
    def test_add_revenue_column():
        print("======================================================================")
```

```python
        try:
            df = pc.updated_data()

            codewars_test.assert_equals("Total Revenue" in df.columns, True,
                                        "====> Error: Kolom 'Total Revenue' tidak ditemukan; Definisikan nama kolom dengan benar")
        except Exception as e:
            codewars_test.fail(f"====> Error pada updated_data(); Terdapat Kolom dengan penulisan nama yang salah, yaitu; function: {str(e)}")
        except KeyError as e:
            codewars_test.fail(f"====> Error: Kolom '{e.args[0]}' tidak ditemukan dalam DataFrame. Pastikan Anda memiliki kolom '{e.args[0]}' dalam data Anda.")

    @codewars_test.it("4. Test Nilai Variabel total_pendapatan")
    def test_total_revenue():
        print("======================================================================")
        # Assert few basic statistics on total spent
        expected_total_revenue = 307587.38

        try:
            df = pc.updated_data()

            if 'Unit price' in df.columns and 'Quantity' in df.columns:
                actual = pc.total_pendapatan()
                codewars_test.assert_equals(actual, expected_total_revenue, "====> Error: Total Revenue should be have the correct value")
            else:
                codewars_test.fail("====> Error: Kolom 'Unit price' atau 'Quantity' tidak ditemukan dalam DataFrame. Pastikan Anda memiliki kolom 'Unit price' da
        except Exception as e:
            codewars_test.fail(f"====> Error: fungsi total_pendapatan. Cek nama dan atribut fungsi. {str(e)}")
```

```python
    @codewars_test.it("5. Test Print Fungsi Jumlah Pendapatan")
    def test_total_revenue():
        print("======================================================================")
        expected= "307587.38"
        output_lines = cmd.stdout.decode().splitlines()
        if output_lines:
            actual_value = output_lines[0]
        else:
            actual_value = " "

        codewars_test.assert_equals(actual_value, expected, '====> Error :Tidak Menampilkan nilai fungsi jumalah_pendapatan() menggunakan print()')

```

2.7 Chapter 3 Practicum 3 Test file

```python
import sys
from pathlib import Path
import numpy as np
import subprocess
import importlib
import codewars_test
import pandas as pd

# Update paths and filename
path_answer = "/content/"  # changed
filename = "answer_bab3_percobaan3.py"  # changed


# Import the answer module
pc = importlib.import_module(filename[:-3], ".")
cmd = subprocess.run([sys.executable, f"{path_answer}{filename}"], capture_output=True)  # changed

@codewars_test.describe("BAB 3 | Percobaan 3")
def fixed_tests():

    @codewars_test.it("1. Test Memuat Data")
    def test_load_data():
        print("======================================================================")
        # Assuming expected columns are "Customer ID", "Age" and "Total Spent (USD)"
        expected_columns = ['Movie_Title', 'Release_Year', 'Genre', 'Director', 'Critic_Score',
       'User_Rating', 'Studio', 'Running_Time', 'Budget', 'Box_Office',
       'Unnamed: 10']
        try:
          codewars_test.assert_equals(list(pc.data_load().columns), expected_columns, "====> URL dataset yang digunakan tidak sesuai; kolom pada dataset berbeda")
        except  Exception as e:
          codewars_test.fail(f"====> Error loading data; Terdapat Typo pada kode: {str(e)}")
```

```python
        expected_rows = 18
        try:
          codewars_test.assert_equals(len(pc.data_load()), expected_rows, "====> URL dataset yang digunakan tidak sesuai; kolom pada dataset berbeda")
        except Exception as e:
            codewars_test.fail(f"====> Error checking row count; Terdapat Typo pada Kode: {str(e)}")
        print("")

    @codewars_test.it("2. Test Critic_Score Conversion")
    def test_critic_scores():
        print("======================================================================")

        try:

          codewars_test.assert_equals(isinstance(pc.critic_scores(), np.ndarray), True,">>>> ERROR pada fungsi critic_scores(); Cek nama variabel")

        except  Exception as e:
          codewars_test.fail(f"====> Error Terdapat Typo pada fungsi critic_scores() pada kode: {str(e)}")
        except  KeyError as e:
          codewars_test.fail(f"====> Error loading data; Terdapat Typo pada kode: {str(e)}")

    @codewars_test.it("2. Test Create Variable top_10_movies")
    def test_top_10_movies():
        print("======================================================================")

        try:

            codewars_test.assert_equals(isinstance(pc.top_10_movies(), pd.DataFrame), True, "top_10_movies should be a pandas DataFrame")
        except Exception as e:
            codewars_test.fail(f"====> Error Terdapat Typo pada blok fungsi sorted_indices() kode: {str(e)}")
    print("")
```

```python
    @codewars_test.it("3. Test Print Top 10 Movies")
    def test_critic_scores():
        print("======================================================================")
        output_lines = cmd.stdout.decode().splitlines()
        codewars_test.assert_equals(bool(output_lines), True, '====> Error :Tidak Menampilkan 10 Movies menggunakan print()')
```

2.8 Chapter 3 Practicum 4 Test file

```python
import sys
from pathlib import Path
import numpy as np
import subprocess
import importlib
import codewars_test
import pandas as pd

# Update paths and filename
path_answer = "/content/"  # changed
filename = "answer_bab3_percobaan4.py"  # changed

# Import the answer module
pc = importlib.import_module(filename[:-3], ".")
cmd = subprocess.run([sys.executable, f"{path_answer}{filename}"], capture_output=True)  # changed


@codewars_test.describe("BAB 4 | Percobaan 4")
def fixed_tests():

    @codewars_test.it("1. Test Memuat Data")
    def test_load_data():
        print("======================================================================")
        # Assuming expected columns are "Customer ID", "Age" and "Total Spent (USD)"
        expected_columns = ["Customer ID","Age", "Total Spent (USD)"]
        try:
          codewars_test.assert_equals(list(pc.data_load().columns), expected_columns, "====> URL dataset yang digunakan tidak sesuai; kolom pada dataset berbeda")
        except  Exception as e:
          codewars_test.fail(f"====> Error loading data; Terdapat Typo pada kode: {str(e)}")

```

```python
        expected_rows = 50
        try:
          codewars_test.assert_equals(len(pc.data_load()), expected_rows, "====> URL dataset yang digunakan tidak sesuai; kolom pada dataset berbeda")
        except Exception as e:
            codewars_test.fail(f"====> Error checking row count; Terdapat Typo pada Kode: {str(e)}")
        print("")

    @codewars_test.it("2. TestShow First Five Rows")
    def test_show_first_five_rows():
        # Call the function and capture the output
        print("======================================================================")
        try:
          actual_output = pc.head_rows().shape[0] # Get the number of rows
          expected_rows = 5

          # Assert that the expected output matches the actual output
          codewars_test.assert_equals(actual_output, expected_rows, "pada head_rows(); Output Tidak Sesuai; Seharusnya menampilkan 5 baris data; Output anda" )
        except Exception as e:
          codewars_test.fail(f"====> Error pada head_rows(); Terdapat Typo pada kode function: {str(e)}")

        print("")

    @codewars_test.it("3. Test calculates descriptive statistics for Customer Age")
    def test_age_descriptive_stats():
        print("======================================================================")
        # Assert few basic statistics on age
        expected_mean =34.54
        expected_std =13.06999992192745 5
        expected_median = 33.5
        expected_skew = 0.2325824223215271
        try:
```

```
        codewars_test.assert_equals(pc.data_load()["Age"].mean(), expected_mean, "Age mean should be positive")
        codewars_test.assert_equals(pc.data_load()["Age"].std(), expected_std, "Age standard deviation should be positive")
        codewars_test.assert_equals(pc.data_load()["Age"].median(), expected_median, "Age mean should be positive")
        codewars_test.assert_equals(pc.data_load()["Age"].skew(), expected_skew, "Age standard deviation should be positive")

        quartile = pc.data_load()["Age"].quantile([0.25, 0.5, 0.75])

        codewars_test.assert_equals(type(quartile), pd.Series, "Quartiles should be a pandas Series")
      except Exception as e:
          codewars_test.fail(f"====> Error pada head_rows(); Terdapat Typo pada kode function: {str(e)}")
      print("")


@codewars_test.it("4. Test quartiles's value | Age")
def test_quartile_values():
    print("======================================================================")
    # Assuming quartiles are for "Age" (adjust expected values as needed)
    try:
      expected_quartiles = np.array([22.25, 33.50, 45.00])  # Adjust these based on your data
      tolerance = 5  # Adjust tolerance for approximate value comparison
      calculated_quartiles = pc.data_load()["Age"].quantile([0.25, 0.5, 0.75])

      codewars_test.assert_equals(np.allclose(calculated_quartiles, expected_quartiles, atol=tolerance), True, "Quartiles should be close to expected values")
    except Exception as e:
        codewars_test.fail(f"====> Error pada head_rows(); Terdapat Typo pada kode function: {str(e)}")

    print("")
```

```
@codewars_test.it("5. Test calculates descriptive statistics for Total Spent")
def test_total_spent_descriptive_stats():
    print("======================================================================")
    # Assert few basic statistics on total spent
    expected_mean =114.0
    expected_std =52.45989721993868
    expected_median = 110.0
    expected_skew = 0.19622312582543433
    try:

      codewars_test.assert_equals(pc.data_load()["Total Spent (USD)"].mean(), expected_mean, "Total Spent mean should be positive")
      codewars_test.assert_equals(pc.data_load()["Total Spent (USD)"].std(), expected_std, "Total Spent standard deviation should be positive")
      codewars_test.assert_equals(pc.data_load()["Total Spent (USD)"].median(), expected_median, "Age mean should be positive")
      codewars_test.assert_equals(pc.data_load()["Total Spent (USD)"].skew(), expected_skew, "Age standard deviation should be positive")

      quartile = pc.data_load()["Total Spent (USD)"].quantile([0.25, 0.5, 0.75])

      codewars_test.assert_equals(type(quartile), pd.Series, "Quartiles should be a pandas Series")
    except Exception as e:
        codewars_test.fail(f"====> Error pada head_rows(); Terdapat Typo pada kode function: {str(e)}")
    print("")

@codewars_test.it("6. Test quartiles's value | Total Spent (USD)")
def test_quartile_values():
    print("======================================================================")
    # Assuming quartiles are for "Age" (adjust expected values as needed)

    expected_quartiles = np.array([71.25, 110.00, 153.75])  # Adjust these based on your data
    tolerance = 5
     # Adjust tolerance for approximate value comparison
    try:
```

```
    try:
      calculated_quartiles = pc.data_load()["Total Spent (USD)"].quantile([0.25, 0.5, 0.75])

      codewars_test.assert_equals(np.allclose(calculated_quartiles, expected_quartiles, atol=tolerance), True, "Quartiles should be close to expected values")
    except Exception as e:
        codewars_test.fail(f"====> Error pada head_rows(); Terdapat Typo pada kode function: {str(e)}")
    print("")

@codewars_test.it("7. calculates correlation coefficient")
def test_correlation():
    print("======================================================================")
    # Assert correlation coefficient is a number
    try:
      corr_= pc.correlation()

      codewars_test.assert_equals(isinstance(corr_, float), True, ">>>> Error pada fungsi correlation(); Variabel 'correlation' Tidak Terdefinisi; Cek nama varia
    except Exception as e:
      codewars_test.fail(f"====> Error pada correlation(); Terdapat Typo pada kode function: {str(e)}")
    except NameError as e:
      codewars_test.fail(f"====> Error pada correlation(); Terdapat Typo pada kode function: {str(e)}")
    print("")
```

2.9 Chapter 3 Practicum 5 Test file

```python
import sys
from pathlib import Path
import numpy as np
import subprocess
import importlib
import codewars_test
import pandas as pd

# Update paths and filename
path_answer = "/content/"  # changed
filename = "answer_bab3_percobaan5.py"  # changed

# Import the answer module
pc = importlib.import_module(filename[:-3], ".")
cmd = subprocess.run([sys.executable, f"{path_answer}{filename}"], capture_output=True)  # changed

@codewars_test.describe("BAB 3 | Percobaan 5")
def fixed_tests():

    @codewars_test.it("1. Test Memuat Data")
    def test_load_data():
        print("=====================================================================")
        # Assuming expected columns are "Customer ID", "Age" and "Total Spent (USD)"
        expected_columns = ['No', 'Nama Produk', 'Kategori', 'Harga (Rp)', 'Jumlah Terjual']
        try:
            codewars_test.assert_equals(list(pc.load_data().columns), expected_columns, "====> URL dataset yang digunakan tidak sesuai; kolom pada dataset berbeda")
        except  Exception as e:
            codewars_test.fail(f"====> Error loading data; Terdapat Typo pada kode: {str(e)}")

        expected_rows = 33
        try:
            codewars_test.assert_equals(len(pc.load_data()), expected_rows, "====> URL dataset yang digunakan tidak sesuai; kolom pada dataset berbeda")
        except Exception as e:
            codewars_test.fail(f"====> Error checking row count; Terdapat Typo pada Kode: {str(e)}")
        print("")

    @codewars_test.it("2. Test Calculates Mean")
    # Assuming your function calculates mean, median, and standard deviation
    def test_descriptive_statistics():
        print("=====================================================================")

        # Test mean of "Jumlah Terjual"
        expected_mean = 16.09375  # Replace with expected mean value

        try:
            codewars_test.assert_equals(pc.load_data()["Jumlah Terjual"].mean(), expected_mean, "Total Spent mean should be positive")
        except Exception as e:
            codewars_test.fail(f"====> Error checking row count; Terdapat Typo pada Kode: {str(e)}")

        print(" ")

    @codewars_test.it("3. Test Calculates Median")
    def test_descriptive_statistics():
        print("=====================================================================")
        # Test median of "Jumlah Terjual"
        expected_median = 15.0  # Replace with expected median value
        data = pc.load_data()["Jumlah Terjual"].median()
        try:
            codewars_test.assert_equals(pc.load_data()["Jumlah Terjual"].median(), expected_median, "Age mean should be positive")
        except Exception as e:
            codewars_test.fail(f"====> Error checking row count; Terdapat Typo pada Kode: {str(e)}")

        print(" ")
        # Test mode (assuming mode function returns a Series)
    @codewars_test.it("4. Test Calculates Mode")
    def test_descriptive_statistics():
        print("=====================================================================")
        data = pc.load_data()
        expected_mode = 8
        mode_jumlah_terjual = data["Jumlah Terjual"].mode().iloc[0]

        try:

            codewars_test.assert_equals(mode_jumlah_terjual, expected_mode)
        except Exception as e:
            codewars_test.fail(f"====> Error checking row count; Terdapat Typo pada Kode: {str(e)}")

        print(" ")
```

2.10 Chapter 4 Practicum 1 Test file

```
import sys
from pathlib import Path
import numpy as np
import subprocess
import importlib
import codewars_test
import pandas as pd
import matplotlib.pyplot as plt

# Update paths and filename
path_answer = "/content/"  # changed
filename = "answer_bab4_percobaan1.py"  # changed

# Import the answer module
pc = importlib.import_module(filename[:-3], ".")
cmd = subprocess.run([sys.executable, f"{path_answer}{filename}"], capture_output=True)  # changed

@codewars_test.describe("BAB 4 | Percobaan 1")
def fixed_tests():
    data = {
        'Age Group': ['0-18', '19-30', '31-45', '46-60', '61+'],
        'Population': [25000, 30000, 20000, 15000, 10000]
        }
    df = pd.DataFrame(data)


    @codewars_test.it("1. Test Memuat Data")
    def test_load_data():
        print("========================================================================")
        # Assuming expected columns are "Customer ID", "Age" and "Total Spent (USD)"
        expected_columns = ['Age Group', 'Population']
        try:
            codewars_test.assert_equals(list(pc.load_data().columns), expected_columns, "====> URL dataset yang digunakan tidak sesuai; kolom pada dataset berbeda")
        except Exception as e:
            codewars_test.fail(f"====> Error loading data; Terdapat Typo pada kode: {str(e)}")

        expected_rows = 5
        try:
            codewars_test.assert_equals(len(pc.load_data()), expected_rows, "Dataframe should have expected rows")
        except Exception as e:
            codewars_test.fail(f"====> Error checking row count; Terdapat Typo pada Kode: {str(e)}")
        print("")

    @codewars_test.it("Get Population Column")
    def test_pie_chart_logic():
        print("========================================================================")
        expected_total = list(df['Population'])
        codewars_test.assert_equals(list(pc.load_data()['Population']), expected_total, "Dataframe should have expected total")
        print("")

    @codewars_test.it("Get Age Group Column")
    def test_pie_chart_logic():
        print("========================================================================")
        expected_total = list(df['Age Group'])
        codewars_test.assert_equals(list(pc.load_data()['Age Group']), expected_total, "Dataframe should have expected total")
        print("")

    @codewars_test.it("Check Pie Chart")
    def test_pie_chart_logic():
        print("========================================================================")
        actual = cmd.stdout.decode().splitlines()[0]
        expected = "Figure(1000x800)"

        try:
            codewars_test.assert_equals(actual, expected, 'Error : Jawaban yang benar adalah "Indonesia"')
        except AttributeError as e:
            print(e)
```

3. Data Analytics Learning Module

Study Case and Practicum

3.1 Chapter 1 Practicum 1

1.5 Contoh Studi Kasus

Memuat Data dari CSV File Lokal

Scenario: Dataset diambil dari direktori github. Muat dataset dan baca data dari CSV format menjadi Pandas DataFrame objek.

Objective:

- Gunakan library Pandas untuk memuat dataset
- Memuat data csv dari file lokal

Langkah - Langkah:

1. **Import Pandas libraries:**

```
import pandas as pd
```

2. **Load Data:**

2.1 Membuat fungsi data_load(). Dalam fungsi data_load(), tentukan variabel df untuk menyimpan pd.read_csv(nama file.csv) untuk membaca data dari file lokal CSV.

```
def data_load():
  df = pd.read_csv('data.csv')
  return df
```

2.2 Panggil fungsi **data_load()** untuk menjalankan program

```
data_load()
```

Tampilan Keseluruhan kode:

```
import pandas as pd

def data_load():
  df = pd.read_csv('data.csv')
  return df

data_load()
```

1.6 Praktikum

Memuat Data dari CSV menggunakan URL

Skenario: Dataset diambil dari direktori github. Muat dataset dan baca data dari CSV format menjadi Pandas DataFrame objek.

Objektif:

- Gunakan library Pandas untuk memuat dataset
- Memuat data csv dari url

Langkah-Langkah:

1. **Import Library:**
   - **import pandas as pd.**
2. **Load Data:**
   2.1 Definisikan variabel **url** ,
   2.2 Gunakan link di bawah ini sebagai nilai dari **url** variabel
   https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main/purchases.csv
   2.3 Buatlah fungsi **data_load().**
   2.4 Definisikan variabel **df** di dalam fungsi **data_load()**. Gunakan **pd.read_csv(url)** membaca data CSV.
   2.5 print fungsi **data_load()** untuk menggunakan **print().**
3. **Submit**
   Submit file dengan nama **answer_bab1_percobaan1.py** pastikan file disimpan dengan format Python file **(.py)**

3.2 Chapter 1 Practicum 2

1.4 Contoh Studi Kasus

Menampilkan Data Produk Toko Online

Scenario: Dataset diambil dari direktori github. Tampilkan lima konten teratas dari Data Produk Toko Online

Objective: Gunakan .head() untuk menampilkan lima konten teratas

Steps:

1. **Import Pandas libraries:**

```
import pandas as pd
```

2. **Load Data:**

2.1 Membuat fungsi data_load(). Dalam fungsi data_load(), tentukan variabel df untuk menyimpan pd.read_csv(nama file.csv) untuk membaca data dari file lokal CSV

```
def data_load():
  df = pd.read_csv('data.csv')
  return df
```

2.2 Bagaimana lima baris pertama dari dataset. Buat fungsi **head_rows()**. Kembalikan nilai pemanggilan fungsi **data_load()** dan penggunaan **head()**

2.3 Panggil fungsi **head_rows()** untuk menampilkan hasil

```
#show first five rows
def head_rows():
  return data_load().head()

head_rows()
```

Tampilan keseluruhan kode

```
import pandas as pd

def data load():
  df = pd.read csv('data.csv')
  return df

#show first five rows
def head_rows():
  return data_load().head()

head_rows()
```

1.5 Praktikum

Menampilkan Data Produk Online Store

Skenario: Dataset diambil dari direktori github. Muat dataset dan baca data dari CSV format menjadi Pandas DataFrame objek.
Objektif: Gunakan library Pandas untuk memuat dataset

Langkah-Langkah:

1. **Import Pandas Library**
   import pandas as pd
2. **Load Data:**
   2.1 Gunakan **url** berikut untuk memuat dataset (https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main/online_store_data.csv).
   2.2 Buatlah fungsi **data_load().**
   2.2.1 Definisikan variabel **df** di dalam fungsi **data_load().**
   2.2.2 Gunakan **pd.read_csv(url)** membaca file csv dataset.
   2.2.3 **return** variabel **df**

   2.3 Buatlah fungsi **sample_rows().**
   2.3.1 Definisikan variabel **sampel** di dalam fungsi **sample_rows ().**
   2.3.2 Pada variabel **sample**, simpan nilai dari pemanggilan **data_load().sample()** yang akan memanggilkan 1 baris data secara acak
   2.3.3 **return** variabel **sample**

   2.4 Tampilkan nilai dari fungsi **sample_rows()** menggunakan **print()**
   2.5 **Submit**
   Simpan file dengan nama **answer_bab1_percobaan2.py** pastikan menyimpan file dengan format file Python **(.py)**

3.3 Chapter 2 Practicum 3

1.4 Contoh Studi Kasus

Preprocessing Data untuk Analisis Statistik

Langkah - Langkah

1. Memuat Pustaka Pandas `import pandas as pd`.

```
import pandas as pd
```

2. Mendefinisikan Fungsi `load_data()`:`def load_data()`. Perintah ini mendefinisikan sebuah fungsi bernama `load_data()`. Fungsi ini akan digunakan untuk memuat data dari file CSV ke dalam DataFrame pandas.
3. Menetapkan URL Data, `url = "https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main/data.csv"`. Perintah ini menetapkan URL file CSV yang berisi data yang ingin dimuat. Pastikan untuk mengganti URL ini dengan URL file CSV Anda yang sebenarnya.
4. Membaca Data CSV. `df = pd.read_csv(url)`. Perintah ini menggunakan fungsi `pd.read_csv()` dari pustaka pandas untuk membaca data dari file CSV yang ditentukan oleh variabel `url`. Hasilnya disimpan dalam objek DataFrame dengan nama `df`.

```
def load_data():
  url =
"https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main
/data.csv"
  df = pd.read_csv(url)
  return df
```

5. Menampilkan Jumlah Nilai Hilang: `print(load_data().isnull().sum())`. Perintah ini memanggil fungsi `load_data()` untuk memuat data dan kemudian menghitung jumlah nilai yang hilang di setiap kolom DataFrame. Hasilnya dicetak ke konsol, menunjukkan berapa banyak nilai yang hilang untuk setiap variabel dalam dataset.

```
print(load_data().isnull().sum())
```

Output:

```
Duration    0
Pulse       0
Maxpulse    0
Calories    5
dtype: int64
```

6. Memuat Data ke dalam DataFrame Baru: `new_df = load_data()`. Perintah ini membuat DataFrame baru bernama `new_df` dan memuat data dari fungsi `load_data()`.

7. Mengisi Nilai Hilang pada Kolom Kalori:
`new_df['Calories'].fillna(new_df['Calories'].mean())`. menggunakan metode `fillna()` pada kolom `Calories` dari DataFrame `new_df`. Metode ini mengisi nilai yang hilang dengan nilai rata-rata dari kolom `Calories`.

```
def updated_data():
  updated_df =
load_data().fillna(load_data()['Calories'].mean())
  return updated_df
```

8. Menampilkan DataFrame Baru `print(new_df)`. Perintah ini mencetak DataFrame `new_df` ke konsol.
9. Menampilkan Jumlah Nilai Hilang Setelah Pengisian Perhatikan bahwa pada kolom `Calories` sekarang tidak memiliki null

```
print(updated_data().isnull().sum())
```

```
Duration    0
Pulse       0
Maxpulse    0
Calories    0
dtype: int64
```

Tampilan keseluruhan kode

```
import pandas as pd

def load_data():
  url =
"https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main/da
ta.csv"
  df = pd.read_csv(url)
  return df
print(load_data().isnull().sum())

def updated_data():
  updated_df = load_data().fillna(load_data()['Calories'].mean())
  return updated_df

print(updated_data().isnull().sum())
```

1.6 Praktikum

Mengisi Nilai Hilang pada Kolom Kalori

Skenario:

Bayangkan Anda adalah seorang ahli gizi yang ingin menganalisis data makanan untuk membantu klien Anda mencapai tujuan kesehatan mereka. Anda memiliki kumpulan data CSV yang berisi informasi tentang berbagai makanan, termasuk nama makanan, kalori, lemak, protein, dan karbohidrat. Namun, Anda menemukan bahwa beberapa nilai kalori hilang dalam dataset. Hal ini dapat membuat analisis data menjadi tidak akurat dan tidak dapat diandalkan.

Objektif:

- Isi nilai kalori hilang untuk tingkatkan kualitas data dan analisis.
- Bantu klien buat pilihan makanan dan rekomendasi diet lebih baik.

Langkah – Langkah:

1. Import library untuk manipulasi data.
2. Define fungsi `load_data()`:
    a. Menetapkan URL file CSV ke variabel `url`.

       https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main/data.csv

    b. Membaca data CSV menggunakan `pd.read_csv(url)`, menyimpannya dalam DataFrame `df`.
    c. Mengembalikan DataFrame `df`.

3. Mengisi Nilai yang hilang dengan nilai rata-rata:
    a. Define fungsi `updated_data()`:
    b. Buat variabel `updated_data`
    c. Memuat data dengan `load_data()`.
    d. Mengisi nilai yang hilang pada kolom 'Calories' dengan rata-rata nilai 'Calories' dalam DataFrame menggunakan `.fillna(load_data['Calories'].mean())`
    e. Mengembalikan nilai `updated_data`.
4. Submit

Simpan nama file dengan **answer_bab2_percobaan3.py** pastikan file tersimpan dalam format Python file **(.py)**

3.4 Chapter 3 Practicum 1

1.5 Contoh Studi Kasus

Memotong Array NumPy 3 Dimensi

Buatlah kode Python yang menggunakan slicing untuk memotong array NumPy 3 dimensi untuk mengambil kotak ke-1, baris ke-2, dan kolom ke-1 hingga ke-2:

Langkah-langkah:

1. **Import pustaka NumPy:** Import pustaka NumPy dengan alias np.

```
import numpy as np
```

2. **Buat array NumPy 3 dimensi:**
   - Define '**array_3d**' variable
   - Use np.array(), membuat array 3D ukuran 2x3x3

```
# Contoh array NumPy 3 dimensi
array_3d = np.array([
  [ [1, 2, 3], [4, 5, 6], [7, 8, 9] ],
  [ [10, 11, 12], [13, 14, 15], [16, 17, 18] ]
])
```

3. Gunakan slicing untuk memotong array. Sintaksnya adalah nama_array[kotak, baris, kolom]. Dalam kasus ini, ambil data pada kotak ke-1, baris ke-2, dan kolom ke-1 hingga ke-2. **array_3d[0, 1, 1:3]**

4. Simpan hasil potongan array dalam variabel **array_potongan**.

```
# Memotong array
array_potongan = array_3d [0, 1, 1:3] # Kotak ke-1, baris ke-2, kolom ke-1 hingga ke-2
```

5. Cetak hasil potongan array ke konsol untuk melihat isinya.

```
print("Array yang dipotong:")
print(array_potongan)
```

Output:

```
Array yang dipotong:
[5 6]
```

Tampilan Keseluruhan Kode

```
import numpy as np
# Contoh array NumPy 3 dimensi
array 3d = np.array([
  [ [1, 2, 3], [4, 5, 6], [7, 8, 9] ],
  [ [10, 11, 12], [13, 14, 15], [16, 17, 18] ]
])
array potongan = array 3d [0, 1, 1:3]
print("Array yang dipotong:")
print(array_potongan)
```

1.6 Praktikum

Mengambil Data Spesifik Penjualan Produk

Skenario: Sebuah toko online memiliki data penjualan produk yang disimpan dalam array NumPy 3 dimensi. Array ini berisi data penjualan produk untuk tiga kategori (elektronik, pakaian, dan mainan) selama tiga bulan (Januari, Februari, dan Maret). Setiap elemen dalam array adalah jumlah produk yang terjual untuk kategori dan bulan tertentu.

Objektif: Menerapkan slicing untuk mengambil data yang spesifik pada Numpy Array 3D.

Langkah – Langkah:

1. Import pustaka NumPy: Import pustaka NumPy dengan alias np.
2. Buat array NumPy 3 dimensi:
    - Buat variable bernama '**data_penjualan**'
    - Buat array NumPy 3D ukuran 3x3x3 menggunakan **np.array()** untuk membuatnya.
    - Gunakan List berikut

```
data penjualan = np.array([
    # Kategori 1 (Elektronik)
    [[10, 20, 30],  # Bulan 1
     [40, 50, 60],  # Bulan 2
     [70, 80, 90]], # Bulan 3

    # Kategori 2 (Pakaian)
    [[100, 110, 120], # Bulan 1
     [130, 140, 150], # Bulan 2
     [160, 170, 180]], # Bulan 3

    # Kategori 3 (Mainan)
    [[190, 200, 210], # Bulan 1
     [220, 230, 240], # Bulan 2
     [250, 260, 270]]  # Bulan 3
])
```

3. Gunakan Slicing untuk memotong array. Sintaksnya adalah nama_array[kotak, baris, kolom]. Ambil data pada Potong kotak ke-1, baris ke-2, dan kolom ke-1 hingga ke-2
4. Simpan hasil potongan array dalam variabel **data_spesifik**.
5. Cetak hasil potongan array ke konsol untuk melihat isinya. Gunakan **print()**

Output:

Array yang dipotong:

```
[50 60]
```

6. Submit file dengan nama **answer_bab3_percobaan1.py** pastikan file disimpan dalam format Python file (**.py**)

3.5 Chapter 3 Practicum 2

1.4 Contoh Studi Kasus

Produk dengan Harga Rata-Rata Tertinggi

Pada contoh ini kita akan belajar menganalisa data dengan menemukan produk dengan garga rata-rata tertinggi. Berikut ini adalah langkah-langkahnya :

1. Buat fungsi **load_data()** untuk membaca file CSV. Simpan pada variable **data_toko** Akses dataset pada url berikut: https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/master/data_toko.csv

```
import pandas as pd

url =
"https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/ma
ster/data_toko.csv"  # Replace with your actual URL
```

Mengembalikan nilai **data_toko** menggunakan **return**

```
# Read the CSV directly from the URL
def data_load():
  data_toko = pd.read_csv(url)
  return data_toko
```

2. **Memeriksa Data:**
   - Buatlah fungsi bernama **head_rows()** untuk menampilkan 10 baris pertama dari DataFrame.

```
# Tampilkan 10 baris pertama DataFrame
def head_rows():
  return data_load().head(10)
```

   - Tampilkan informasi tentang DataFrame, termasuk jumlah baris, kolom, dan tipe data. Menggunakan **info()**

```
data_load().info()
```

3. **Analisis Data:**
   - Hitung pendapatan total dari penjualan semua produk. Buat fungsi **find_highest_average_price()**

```
def find_highest_average_price():
  df = data_load()

  # Kelompokkan data berdasarkan lini produk
  produk_terkelompokan = df.groupby('Product line')

  # Hitung harga rata-rata untuk setiap kelompok produk
  produk_terkelompokan = produk_terkelompokan[['Unit
price']].mean()
```

```
  # Temukan produk dengan harga rata-rata tertinggi
  produk_harga_rata_tinggi =
produk_terkelompokan.idxmax()
  harga_rata_tinggi = produk_terkelompokan.max()

  # Kembalikan produk dengan harga rata-rata tertinggi
dan harga rata-ratanya
  return produk_harga_rata_tinggi, harga_rata_tinggi
```

4. Print produk dengan harga rata-rata tertinggi print()

```
def print_highest_average_price():
  produk_harga_rata_tinggi, harga_rata_tinggi =
find_highest_average_price()
  print(f"Produk dengan harga rata-rata tertinggi:
{produk_harga_rata_tinggi} dengan harga rata-rata
{harga_rata_tinggi}")
```

```
print_highest_average_price()
```

Tampilan Keseluruhan Kode

```
import pandas as pd

url =
"https://raw.githubusercontent.com/noora20FH/skripsi_noor
a2023/master/data_toko.csv"  # Replace with your actual
URL

def data_load():
  data_toko = pd.read_csv(url)
  return data_toko

def head_rows():
  return data_load().head(10)

def find_highest_average_price():
  df = data_load()

  # Kelompokkan data berdasarkan lini produk
  produk_terkelompokan = df.groupby('Product line')

  # Hitung harga rata-rata untuk setiap kelompok produk
  produk_terkelompokan = produk_terkelompokan[['Unit
price']].mean()
```

```
  # Temukan produk dengan harga rata-rata tertinggi
  produk_harga_rata_tinggi =
produk_terkelompokan.idxmax()
  harga_rata_tinggi = produk_terkelompokan.max()

  # Kembalikan produk dengan harga rata-rata tertinggi
dan harga rata-ratanya
  return produk harga rata tinggi, harga rata tinggi

def print_highest_average_price():
  produk_harga_rata_tinggi, harga_rata_tinggi =
find_highest_average_price()
  print(f"Produk dengan harga rata-rata tertinggi:
{produk_harga_rata_tinggi} dengan harga rata-rata
{harga_rata_tinggi}")

# Gunakan fungsi untuk menemukan produk dengan harga
rata-rata tertinggi
print_highest_average_price()
```

1.5 Praktikum

Menganalisis dataset film untuk menemukan 10 film Terbaik

Skenario:

Bayangkan Anda bekerja di platform streaming film dan ingin mempromosikan 10 film terbaik kepada pengguna berdasarkan skor kritikus. Anda memiliki akses ke dataset film yang berisi berbagai informasi, termasuk skor kritikus.

Objektif:

1. Menganalisis dataset film untuk menemukan 10 film dengan skor kritikus tertinggi.
2. Menyajikan daftar 10 film terbaik.

Langkah – Langkah :

1. Import library yang dibutuhkan
2. Buat fungsi **data_load()** untuk membaca file CSV. Simpan pada variable **data_toko**
   Akses dataset pada url berikut:
   https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main/clean_movie_data.csv
   Mengembalikan nilai **data_toko** menggunakan **return**

3. Buatlah fungsi bernama **critic_scores()**

4. Buatlah variable bernama **critic_scores** di dalam fungsi **critic_scores()** untuk merubah tipe data kolom **Critic_Score** pada dataset pada fungsi data_load() menjadi Numpy array menggunakan **to.numpy()**

   nama_variabel = nama_fungsi()["nama_kolom"].to_numpy()

5. Return variabel **critic_scores**

6. Buatlah fungsi bernama **sorted_indices()**
7. Buatlah variable bernama **sorted_indices** di dalam fungsi **sorted_indices()** untuk mengurutkan index dari tinggi ke rendah.
   - Panggil variabel fungsi **critic_scores()** dan gunakan **argsort()** beserta python slicing **[::]** untuk mengurutkan terbalik (tinggi ke rendah).

8. **Return** variabel **sorted_indices**
9. Buatlah fungsi **top_10_movies()**
10. Buatlah variable bernama **top_10_movies** di dalam fungsi **top_10_movies()**
    - Panggil fungsi **data_load()** untuk menggunakan DataFrame yang berisi data movie.
    - Gunakan metode **iloc()** pada **data_load()** untuk mengakses elemen dalam DatFrame
    - Potong array pada **sorted_indices()** menggunakan slicing python **[:]** untuk skor 10 teratas

      nama_variabel = nama_fungsi().iloc[nama_variabel[:10]]

11. Print total 10 movie terbaik menggunakan metode **print()**
12. **Submit**
    Beri file dengan nama **answer_bab3_percobaan3.py** pastikan file dalam format **Python (.py)**

3.6 Chapter 3 Practicum 4

1.4 Contoh Studi Kasus

Analyzing Exam Scores and Study Hours

Skenario: Anda adalah seorang guru dan ingin menyelidiki hubungan antara jumlah jam belajar siswa dan nilai ujiannya. Anda telah mengumpulkan data untuk 10 siswa, termasuk jam belajar dan nilai ujian mereka.

Tujuan: Menggunakan Python untuk menghitung statistik deskriptif dan koefisien korelasi untuk menganalisis data ini dan memahami apakah ada hubungan antara jam belajar dan nilai ujian.

Steps:

1. **Import Pandas libraries:**

```
import pandas as pd
```

2. **Load Data:**

2.1 Definisikan varibel **url** (https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main/score.csv).

```
data =
'https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/ma
in/purchases.csv'
```

2.2 Buat fungsi **data_load()**. Di dalam fungsi **data_load()**, definisikan variabel **df** untuk menyimpan pd.read_csv(url) digunakan untuk membaca data file CSV kedalam Pandas DataFrame objek yang diberi nama data.

```
def data_load():
  df = pd.DataFrame(data)
  return df
```

2.3 Bagaimana lima baris pertama dari kumpulan data. Buat fungsi **head_rows()**. Kembalikan nilai pemanggilan fungsi **data_load()** dan penggunaan **head()**.

```
#show first five rows
def head_rows():
  return data_load().head()
```

3 **Descriptive Statistics:**

Central Tendency:

- Hitung mean, median, dan modus untuk jam belajar dan nilai ujian menggunakan metode deskripsikan() pada DataFrame. Ini akan memberikan ringkasan keseluruhan.

```
print(df.describe())
```

4 **Analyze Study Hours:**

Ini secara khusus berfokus pada kolom "Age ":

- Menghitung usia rata-rata(mean) menggunakan df['Study Hours'].mean().
- Menghitung median usia menggunakan df['Study Hours'].median().
- Menghitung simpangan baku usia menggunakan df['Study Hours'].std().
- Menghitung kemiringan distribusi umur menggunakan df['Study Hours'].skew().
  Skewness menunjukkan seberapa simetris data di sekitar mean.
- Menghitung kuartil (persentil ke-25, persentil ke-50 (median), persentil ke-75) usia menggunakan df['Study Hours'].quantile([0.25, 0.5, 0.75]).

```
print("Study Hours:")
print(f"  Mean: {df['Study Hours'].mean()}")
print(f"  Median: {df['Study Hours'].median()}")
print(f"  Standard Deviation: {df['Study Hours'].std()}")
print(f"  Skewness: {df['Study Hours'].skew()}")
print(f"  Quartiles: {df['Study Hours'].quantile([0.25, 0.5,
0.75])}")  # 25th, 50th, 75th percentiles
```

5 **Analyze Total Spent:**

Mirip dengan menganalisis usia pelanggan, kode ini mengulangi langkah yang sama untuk kolom "Exam Score".

```
print("\nExam Scores:")
print(f"  Mean: {df['Exam Score'].mean()}")
print(f"  Median: {df['Exam Score'].median()}")
print(f"  Standard Deviation: {df['Exam Score'].std()}")
print(f"  Skewness: {df['Exam Score'].skew()}")
print(f"  Quartiles: {df['Exam Score'].quantile([0.25, 0.5,
0.75])}")
```

6 **Correlation Analysis:**

- Mendefinisikan korelasi dalam variabel **correlation**
- Hitung koefisien korelasi antara jam belajar dan nilai ujian menggunakan metode **corr()** pada DataFrame. Ini menghitung koefisien korelasi, yang menunjukkan kekuatan dan arah hubungan linier antara jam belajar dan nilai ujian.
- Cetak variabel korelasinya

```
correlation = df["Study Hours"].corr(df["Exam Score"])
```

```
#A positive value indicates a positive correlation (more study hours
lead to higher scores).
#A value close to zero suggests weak or no correlation.
print(f"Correlation Coefficient: {correlation}")
```

Output:

```
        Student  Study Hours  Exam Score
count  10.00000     10.00000   10.000000
mean    5.50000      4.50000   70.500000
std     3.02765      3.02765   11.965227
min     1.00000      0.00000   50.000000
25%     3.25000      2.25000   62.000000
50%     5.50000      4.50000   73.500000
75%     7.75000      6.75000   79.500000
max    10.00000      9.00000   85.000000
Study Hours:
  Mean: 4.5
  Median: 4.5
  Standard Deviation: 3.0276503540974917
  Skewness: 0.0
  Quartiles: 0.25    2.25
0.50    4.50
0.75    6.75
Name: Study Hours, dtype: float64

Exam Scores:
  Mean: 70.5
  Median: 73.5
  Standard Deviation: 11.96522739719837 8
  Skewness: -0.6136816517120954
  Quartiles: 0.25    62.0
0.50    73.5
0.75    79.5
Name: Exam Score, dtype: float64
Correlation Coefficient: 0.9768778242494333
```

1.5 Praktikum 4

1. Import Library:

- Import *library* yang dibutuhkan. **import pandas as pd.**

2. Load Data:

2.1 Define **url** variable (https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main/purchases.csv).

2.2 Buatlah fungsi **data_load().**

- Buat variabel **df** di dalam fungsi **data_load().** Gunakan **pd.read_csv(url)** untuk membaca data dari CSV file kedalam objek dataframe.

2.3 Tampilkan 5 baris pertama dataset.

- Buat fungsi **head_rows()** Dengan memanggil fungsi **data_load().head()**

3. Descriptive Statistics:

- Buat fungsi **descriptive_statistics()**
- Definisikan variabel **desc_stat**
- Pada variabeli ini, panggil fungsi **data_load()** untuk mendapatkan data. Ambil ringkasan keseluruhan data (mean, median, standard deviation, quartiles) menggunakan .describe().
- Return **desc_stat**

4. Buat fungsi **print_statistics():**

5. Analisa Umur Pelanggan:

- Analisa menggunakan kolom "**Age**":
- Cetak hasil menggunakan **print()**
    - Hitunglah nilai mean menggunakan metode **mean()** pada **data_load()** fungsi.
    - Hitunglah nilai median menggunakan metode **median()** pada **data_load()** fungsi.
    - Hitunglah nilai *Standard Deviation* metode std() pada **data_load()** fungsi.
    - Hitunglah nilai *skewness* menggunakan metode skew() pada **data_load()** fungsi. Skewness menunjukkan seberapa simetris data di sekitar rata-rata.
    - Hitunglah nilai *quartiles* (persentil 25, persentil 50 (median), persentil 75) menggunakan metode **quantile([0.25, 0.5, 0.75])** pada **data_load()** fungsi. **print(nama_fungsi()['nama_kolom'].metode())**

6. Analisa Total Pengeluaran:

- Sama seperti analisa umur pelanggan, ulangi langkah – langkah di atas untuk Analisa pada kolom "**Total Spent (USD)**".

7. Analize the Correlations:

- Buatlah fungsi dengan nama **correlation()**
- Definisikan variabel korelasi bernama "**correlation**"
- Ini menghitung koefisien korelasi antara "**Age**" dan " **Total Spent (USD)**" menggunakan **data_load()["Age"].corr(data_load() ["Total Spent (USD)"])**. Korelasi mengukur kekuatan dan arah hubungan linier antara dua variabel.
- Return nilai dari correlation
- Korelasi variabel cetak menggunakan **print()**.

8. **Submit**

Beri file dengan nama **answer_bab3_percobaan4.py** pastikan format file adalah Python file **(.py)**

3.7 Chapter 3 Practicum 5

1.4 Contoh Studi Kasus

Analyzing Exam Scores and Study Hours

Skenario: Anda adalah seorang guru dan ingin menyelidiki hubungan antara jumlah jam belajar siswa dan nilai ujiannya. Anda telah mengumpulkan data untuk 10 siswa, termasuk jam belajar dan nilai ujian mereka.

Tujuan: Menggunakan Python untuk menghitung statistik deskriptif dan koefisien korelasi untuk menganalisis data ini dan memahami apakah ada hubungan antara jam belajar dan nilai ujian.

Steps:

1. **Import Pandas libraries:**

```
import pandas as pd
```

2. **Load Data:**

2.1 Definisikan varibel **url** (https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main/purchases.csv).

```
url =
'https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/ma
in/purchases.csv'
```

2.2 Buat fungsi **data_load()**. Di dalam fungsi **data_load()**, definisikan variabel **df** untuk menyimpan pd.read_csv(url) digunakan untuk membaca data file CSV kedalam Pandas DataFrame objek yang diberi nama data.

```
def data_load():
  df = pd.DataFrame(data)
  return df
```

2.3 Bagaimana lima baris pertama dari kumpulan data. Buat fungsi **head_rows()**. Kembalikan nilai pemanggilan fungsi **data_load()** dan penggunaan **head()**.

```
#show first five rows
def head_rows():
  return data_load().head()
```

3 **Descriptive Statistics:**

Central Tendency:

- Hitung mean, median, dan modus untuk jam belajar dan nilai ujian menggunakan metode **describe()** pada DataFrame. Ini akan memberikan ringkasan keseluruhan.

```
def describe_data():
```

```
    describe_data = load_data().describe()
    return describe_data
```

4 **Analyze Study Hours:**

Hitunglah Measures of Central Tendency ini secara khusus berfokus pada kolom "Age ":

- Buatlah variable **mean_study_hours**, **median_study_hours**, **std_study_hours**, **skew_study_hours**, dan **quartiles_study_hours**.
- Panggil fungsi **data_load()** kolom 'Study Hours', dan gunakan metode **mean()**, **median()**, **std()**, **skew()**, dan **quantile()** pada tiap variable.

```
# Calculate mean of Study Hours
def mean_study_hours():
  mean_study_hours = load_data()['Study Hours'].mean()
  return mean_study_hours

# Calculate median of Study Hours
def median_study_hours():
  median_study_hours = load_data()['Study Hours'].median()
  return median_study_hours

# Calculate standard deviation of Study Hours
def std_study_hours():
  std_study_hours = load_data()['Study Hours'].std()
  return std_study_hours

# Calculate skewness of Study Hours
def skew_study_hours():
  skew_study_hours = load_data()['Study Hours'].skew()
  return skew_study_hours

# Calculate quartiles of Study Hours
def quartiles_study_hours():
    quartiles_study_hours = load_data()['Study
Hours'].quantile([0.25, 0.5, 0.75])
    return quartiles_study_hours
```

5 **Analyze Total Spent:**

Mirip dengan menganalisis usia pelanggan, kode ini mengulangi langkah yang sama untuk kolom "Exam Score".

```
# Calculate mean of Exam Scores
def mean_exam_scores():
  mean_exam_scores = load_data()['Exam Score'].mean()
  return mean_exam_scores

# Calculate median of Exam Scores
def median_exam_scores():
  median_exam_scores = load_data()['Exam Score'].median()
  return median_exam_scores

# Calculate standard deviation of Exam Scores
def std_exam_scores():
  std_exam_scores = load_data()['Exam Score'].std()
  return std_exam_scores

# Calculate skewness of Exam Scores
def skew_exam_scores():
  skew_exam_scores = load_data()['Exam Score'].skew()
  return skew_exam_scores

# Calculate quartiles of Exam Scores
def quartiles_exam_scores():
  quartiles_exam_scores = load_data()['Exam
Score'].quantile([0.25, 0.5, 0.75])
  return quartiles_exam_scores
```

6 **Correlation Analysis:**

- Mendefinisikan korelasi dalam variabel **correlation**
- Hitung koefisien korelasi antara jam belajar dan nilai ujian menggunakan metode **corr()** pada DataFrame. Ini menghitung koefisien korelasi, yang menunjukkan kekuatan dan arah hubungan linier antara jam belajar dan nilai ujian.
- Cetak variabel korelasinya

```
# Calculate correlation coefficient
def correlation_coefficient():
  correlation_coefficient = load_data()['Study
Hours'].corr(load_data()['Exam Score'])
  return correlation_coefficient
```

7. Cetak hasil menggunakan **print()** untuk tiap perhitungan mean, median, dan mode.

```
print(f"Descriptive statistics: \n{describe_data()}")
print(f"Mean of Study Hours: {mean_study_hours()}")
print(f"Median of Study Hours: {median_study_hours()}")
```

```
print(f"Standard deviation of Study Hours: {std_study_hours()}")
print(f"Skewness of Study Hours: {skew_study_hours()}")
print(f"Quartiles of Study Hours: \n{quartiles_study_hours()}")
print(f"Mean of Exam Scores: {mean_exam_scores()}")
print(f"Median of Exam Scores: {median_exam_scores()}")
print(f"Standard deviation of Exam Scores: {std_exam_scores()}")
print(f"Skewness of Exam Scores: {skew_exam_scores()}")
print(f"Quartiles of Exam Scores: \n{quartiles_exam_scores()}")
print(f"Correlation coefficient: {correlation_coefficient()}")
```

1.5 Praktikum 5

Analisis Penjualan Elektronik: Memahami Tren Penjualan Melalui Measures of Central Tendency

Skenario:

Sebuah perusahaan elektronik ingin memahami pola penjualan produk mereka untuk membuat strategi pemasaran yang lebih efektif. Data penjualan selama periode tertentu tersedia dalam format CSV.

Objektif:

Menghitung Measures of Central Tendency (mean, median, mode): mengetahui nilai penjualan rata-rata, nilai penjualan Tengah, dan nilai penjualan yang paling sering terjadi

Langkah Langkah:

1. Import Library:
   - Import library pandas menggunakan **import pandas as pd**.
2. Load Data:
   2.1 Definisikan variabel **url** (https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main/penjualan_elektronik.csv).
   2.2 Buatlah fungsi **load_data()** untuk membaca data csv langsung dari url. Simpan pada variabel **data**
   2.3 Kembalikan nilai variabel **data** menggunakan **return**

3. Hitunglah *Measures of Central Tendency*:
   - Buatlah fungsi **mean_penjualan()**, **median_penjualan()**, dan **mode_penjualan()**
   - Panggil fungsi **load_data()** kolom '**Jumlah Terjual**', dan gunakan metode **mean()**, **median()**, atau **mode()** pada tiap variable

     nama_variabel = nama_fungsi()[‘nama_kolom’].metode()

   - Return nama variabel
4. Cetak hasil menggunakan **print()** untuk tiap perhitungan mean, median, dan mode
5. Submit
   Simpan file dengan nama **answer_bab3_percobaan5.py** pastikan file disimpan dalam

format Python file (.py).

3.8 Chapter 4 Practicum 1

1.4 Contoh Studi Kasus
Membuat Label, Legenda, dan Sudut Awal

Skenario: Sebuah perusahaan menganalisis data penjualan untuk berbagai kategori produk. Mereka ingin membuat diagram lingkaran untuk memvisualisasikan distribusi penjualan di antara kategori-kategori tersebut.

Objektif:

1. Memberi label persentase pada setiap irisan diagram lingkaran dengan kategori produk yang sesuai.
2. Menambahkan legenda yang dengan jelas mengidentifikasi setiap kategori.
3. Memutar diagram lingkaran untuk memulai pada sudut tertentu, sehingga menekankan kategori yang paling signifikan.

Langkah-langkah:

1. Import Pustaka:

1.1 import pandas as pd: Mengimpor pustaka Pandas untuk manipulasi data.

1.2 import matplotlib.pyplot as plt: Mengimpor pustaka Matplotlib untuk visualisasi data.

```
import pandas as pd
import matplotlib.pyplot as plt
```

2. Fungsi Pemuatan Data:

2.1 def load_data(): Mendefinisikan fungsi load_data untuk memuat data dari file CSV.

2.2 Membaca file CSV dengan URL yang diberikan dan menyimpannya dalam DataFrame df.

https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main/product_category.csv

2.3 Mengembalikan DataFrame df.

```
def load data():
  url =
"https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main
/product_category.csv "
  df = pd.read_csv(url)
  return df
```

3. Fungsi Penentuan Ukuran Gambar:

3.1 definisikan variabel **fig_size**

3.2 Membuat objek gambar dengan ukuran **8x6** inci menggunakan plt.subplots().

```
fig_size = plt.subplots(figsize=(8, 6))
```

4. Membuat Pie Chart:

4.2 Menggunakan plt.pie() untuk membuat pie chart dengan:

- Data penjualan (load_data()['Sales']) sebagai ukuran potongan pie.
- Label kategori produk (load_data()['Product Category']) untuk setiap potongan.
- Persentase pada setiap potongan (autopct='%1.1f%%').
- Sudut awal (startangle=90) untuk memulai pie chart dari posisi jam 12.

```
plt.pie(load data()['Sales'], labels=load data()['Product
Category'], autopct='%1.1f%%', startangle=90)
```

4.3 Menambahkan legenda menggunakan plt.legend(title='Product Category').

```
plt.legend(title='Product Category')
```

4.4 Memberikan judul pada grafik menggunakan plt.title('Sales Distribution by Product Category').

```
plt.title('Sales Distribution by Product Category')
```

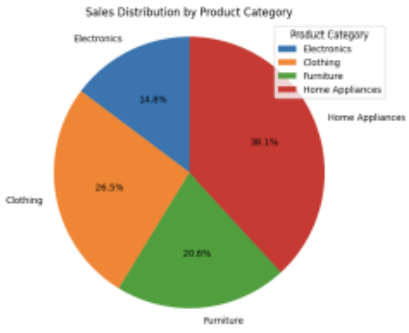
4.5 Menyesuaikan tampilan grafik menggunakan plt.axis('equal').

```
plt.axis('equal')  # Memastikan diagram lingkaran berbentuk bulat
```

4.6 Menampilkan grafik menggunakan plt.show().

```
plt.show()
```

Output:



Tampilan Keseluruhan Kode

```
import pandas as pd
import matplotlib.pyplot as plt

def load_data():
  url =
"https://raw.githubusercontent.com/noora20FH/skripsi noora2023/main
/product category.csv "
  df = pd.read csv(url)
  return df
def create pie chart():

  fig size = plt.subplots(figsize=(8, 6))

  plt.pie(load_data()['Sales'], labels=load_data()['Product
Category'], autopct='%1.1f%%', startangle=90)
  plt.legend(title="Product Category")
  plt.title('Sales Distribution by Product Category')
  plt.axis('equal')
  plt.show()
```

1.4 Praktikum

Skenario:

Departemen perencanaan kota ingin menganalisis distribusi usia penduduknya untuk memahami komposisi demografi dan merencanakan layanan masa depan.

Tujuan:

- Persiapan data: Membuat dataset sampel dengan kategori dan nilai yang sesuai.
- Visualisasi data: Membuat diagram lingkaran untuk merepresentasikan data secara visual.
- Kustomisasi grafik: Menambahkan label, legenda, dan judul untuk meningkatkan keterbacaan dan informativitas grafik.

Langkah – langkah:

1. Import Pustaka:

   1.1 *import pandas as pd*: Mengimpor pustaka Pandas untuk manipulasi data.

   1.2 *import matplotlib.pyplot as plt*: Mengimpor pustaka Matplotlib untuk visualisasi data.

2. Fungsi Pemuatan Data:

2.1 *def load_data()* : definisikan fungsi load_data untuk memuat data dari file CSV.

2.2 buat variabel **df** untuk menyimpan dataframe berupa file csv yang di dapat dari link berikut:

https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main/age_group.csv

2.3 Mengembalikan DataFrame *df* menggunakan return.

3. Membuat Pie Chart:

3.1 Buatlah fungsi bernama **create_pie_chart()**

3.2 Definisikan variabel **df** yang akan menyimpan *data frame* dari load_data()

3.3 Definisikan variabel **get_age_col** untuk mengambil kolom **'Age Group'** dari *data frame.* **df['Age Group']**

3.4 Definisikan variabel **get_population_col** untuk mengambil kolom '**Population** ' dari *data frame.* **df['Population']**

3.5 Definisikan variabel **fig_size** untuk mengatur ukuran gambar.

3.6 Buat objek gambar dengan ukuran *10x8* inci menggunakan *plt.subplots().*

3.7 Menggunakan *plt.pie()* untuk membuat pie chart dengan data populasi, label kelompok umur, persentase, dan sudut awal.

- Ambil data kolom populasi, gunakan variabel **get_population_col** yang telah didefinisikan sebelumnya sebagai ukuran potongan pie.
- Ambil label kategori grup umur, gunakan variabel **get_age_col** yabg telah didefinisikan sebelumnya untuk setiap potongan.
- Persentase pada setiap potongan (**autopct='%1.1f%%'**).
- Sudut awal (**startangle=90**) untuk memulai pie chart dari posisi jam 12.

3.8 Menambahkan legenda menggunakan *plt.legend()* beri nama **"Rentang usia".**

3.9 Memberikan judul pada grafik menggunakan *plt.title()* beri **judul "Distribusi Usia Peduduk pada Kota A".**

3.10 Menyesuaikan tampilan grafik menggunakan *plt.axis('equal').*

3.11 Menampilkan grafik menggunakan *plt.show().*

4. Panggil fungsi **create_pie_chart()** untuk menampilkan pie chart.

5. Submit

Simpan file dengan nama **answer_bab4_percobaan1.py** pastikan file disimpan dalam format Python file (**.py**).