DATA ANALYTICS

BAB 3 : DATA MANIPULATIONS

Praktikum 4

1.1 Tujuan

Mahasiswa mengenal konsep statistik fundamental pada Python

1.2 Ulasan Materi

A. Statistika

1. Mengapa Statistika Penting?

Statistika adalah ilmu yang mempelajari bagaimana mengumpulkan, mengolah, menganalisis, dan menginterpretasikan data. Dalam konteks Python, statistika menjadi kunci untuk:

a. **Statistik membantu Anda memahami pola, tren, dan hubungan dalam data**,
b. **Mengubah data menjadi informasi yang berguna**.
c. **Membangun model prediktif untuk memprediksi kejadian di masa depan**.
d. **Memvisualisasikan data dengan cara yang menarik dan mudah dipahami.**

2. Memahami Descriptive Statistics

Statistik deskriptif adalah tentang menggambarkan dan merangkum data. Ini menggunakan dua pendekatan utama:

1. Pendekatan kuantitatif mendeskripsikan dan merangkum data secara numerik.
2. Pendekatan visual mengilustrasikan data dengan diagram, plot, histogram, dan grafik lainnya.

2.1 Importing libraries

```
import math
import statistics
import numpy as np
import scipy.stats
import pandas as pd
```

2.2 Membuat Data

```
x = [8.0, 1, 2.5, 4, 28.0]
x_with_nan = [8.0, 1, 2.5, math.nan, 4, 28.0]
```

```
y, y_with_nan = np.array(x), np.array(x_with_nan)
z, z_with_nan = pd.Series(x), pd.Series(x_with_nan)
```

2.3 Measures of Central Tendency

- **Mean**
    - **Pure Python:**

```
mean_ = sum(x) / len(x)
```

    - **statistics library:**

```
mean_ = statistics.mean(x)
mean_ = statistics.fmean(x)  # Faster alternative in Python 3.8
```

    - **NmPy:**

```
mean_ = np.mean(y)
mean_ = y.mean()
```

    - **Pandas:**

```
mean_ = z.mean()
```

- **Weighted Mean**
    - **Pure Python:**

```
x = [8.0, 1, 2.5, 4, 28.0]
w = [0.1, 0.2, 0.3, 0.25, 0.15]
wmean = sum(w[i] * x[i] for i in range(len(x))) / sum(w)
```

    - **NumPy:**

```
wmean = np.average(y, weights=w)
wmean = np.average(z, weights=w)
```

- **Harmonic Mean**
    - **Pure Python:**

```
hmean = len(x) / sum(1 / item for item in x)
```

    - **statistics library:**

```
hmean = statistics.harmonic_mean(x)
```

    - **scipy.stats:**

```
hmean = scipy.stats.hmean(y)
hmean = scipy.stats.hmean(z)
```

- **Geometric Mean**
    - **Pure Python:**

```
gmean = 1
for item in x:
    gmean *= item
gmean **= 1 / len(x)
```

- **statistics library (Python 3.8+):**

```
gmean = statistics.geometric_mean(x)
```

- **scipy.stats:**

```
gmean = scipy.stats.gmean(y)
gmean = scipy.stats.gmean(z)
```

- **Median**
  - **Pure Python:**

```
n = len(x)
if n % 2:
    median_ = sorted(x)[round(0.5*(n-1))]
else:
    x_ord, index = sorted(x), round(0.5 * n)
    median_ = 0.5 * (x_ord[index-1] + x_ord[index])
```

  - **statistics library:**

```
median_ = statistics.median(x)
```

  - **NumPy:**

```
median_ = np.median(y)
```

  - **Pandas:**

```
median_ = z.median()
```

- **Mode**
  - **Pure Python:**

```
u = [2, 3, 2, 8, 12]
mode_ = max((u.count(item), item) for item in set(u))[1]
```

  - **statistics library:**

```
mode_ = statistics.mode(u)
mode_ = statistics.multimode(u)  # Returns a list for multimodal
data (Python 3.8+)
```

  - **scipy.stats:**

```
mode_ = scipy.stats.mode(u)
```

  - **Pandas:**

```
u, v, w = pd.Series(u), pd.Series(v), pd.Series([2, 2, math.nan])
u.mode()
```

2.4 Measures of Variability

- **Variance:** Menghitung seberapa tersebar titik data dari mean.
    - Sample variance formula (for n elements): $\Sigma_i(x_i - \text{mean}(x))^2 / (n - 1)$
    - Python (pure):

```
n = len(x)
mean_ = sum(x) / n
var_ = sum((item - mean_)**2 for item in x) / (n - 1)
```

    - Python (statistics):

```
var_ = statistics.variance(x)
```

    - NumPy/pandas:

```
var_ = np.var(y, ddof=1)  # ddof=1 for sample variance
var_ = y.var(ddof=1)
```

- **Standard Deviation:** Akar kuadrat positif dari varians; mempunyai satuan yang sama dengan datanya.
    - Python:

```
std_ = var_ ** 0.5
std_ = statistics.stdev(x)
```

    - NumPy/pandas (same as variance):

```
std_ = np.std(y, ddof=1)
std_ = y.std(ddof=1)
```

- **Skewness:** Mengukur asimetri sampel data.
    - Kemiringan positif: Ekor lebih panjang di sebelah kanan.
    - Kemiringan negatif: Ekor lebih panjang di sebelah kiri.
    - Python (pure):

```
skew_ = (sum((item - mean_)**3 for item in x) * n / ((n - 1) * (n - 2) * std_**3))
```

    - Python (scipy):

```
skew_ = scipy.stats.skew(y, bias=False)  # bias=False for correction
```

    - pandas:

```
skew_ = z.skew()
```

- **Percentiles:** Persentil ke-p adalah nilai sedemikian rupa sehingga p% elemen lebih kecil atau sama dengan itu.

o Kuartil: ke-1 (persentil ke-25), ke-2 (persentil atau median ke-50), ke-3 (persentil ke-75).

- Python (statistics - introduced in Python 3.8):

```
quantiles = statistics.quantiles(x, n=2)  # n: number of percentiles
```

- Python (NumPy):

```
percentile_5 = np.percentile(y, 5)
percentiles = np.percentile(y, [25, 50, 75])
```

- pandas:

```
percentile_5 = z.quantile(0.05)
quartiles = z.quantile([0.25, 0.5, 0.75])
```

- **Ranges:**

o Range: Perbedaan antara elemen maksimum dan minimum.

o Rentang Interkuartil (IQR): Selisih antara kuartil ke-1 dan ke-3.

- Python (NumPy):

```
range_ = np.ptp(y)
IQR = quartiles[1] - quartiles[0]  # from quartile calculation
```

- Built-in functions/methods (min, max):

```
range_ = y.max() - y.min()
```

2.5 Summary of Descriptive Statistics

- SciPy:

```
result = scipy.stats.describe(y, ddof=1, bias=False)
result
```

- Mengembalikan objek dengan berbagai statistik (hitungan, min/maks, mean, varians, skewness, kurtosis).

```
DescribeResult(nobs=9, minmax=(-5.0, 41.0), mean=11.622222222222222,
variance=228.75194444444446, skewness=0.9249043136685094,
kurtosis=0.14770623629658886)
```

describe() returns an object that holds the following descriptive statistics:

- **nobs**: jumlah observasi atau elemen dalam kumpulan data Anda
- **minmax**: tupel dengan nilai minimum dan maksimum kumpulan data Anda
- **mean**: rata-rata kumpulan data Anda
- **variance**: varians kumpulan data Anda

- **skewness**: kemiringan kumpulan data Anda
- **kurtosis**: kurtosis kumpulan data Anda

- pandas:

```
result = z.describe()
result
```

  - Returns a Series object with various statistics (count, mean, std, min/max, quartiles).

```
count    9.000000
mean    11.622222
std     15.124548
min     -5.000000
25%      0.100000
50%      8.000000
75%     21.000000
max     41.000000
dtype: float64
```

Ini mengembalikan Seri baru yang berisi hal-hal berikut:

• **count**: jumlah elemen dalam kumpulan data Anda

• **mean**: rata-rata kumpulan data Anda

• **std**: deviasi standar kumpulan data Anda

• **min and max**: nilai minimum dan maksimum kumpulan data Anda

• **25%, 50%, dan 75%:** kuartil kumpulan data Anda

2.6 Correlation Between Pairs of Data

Korelasi menggambarkan hubungan antara dua variabel.

- Korelasi positif: Nilai yang lebih besar dari satu variabel berhubungan dengan nilai yang lebih besar dari variabel lainnya.
- Korelasi negatif: Nilai yang lebih besar dari satu variabel berhubungan dengan nilai yang lebih kecil dari variabel lainnya.
- Korelasi lemah: Tidak ada hubungan yang jelas antar variabel.

2.7 Covariance

• Kovariansi (kovarians sampel) mengukur arah dan kekuatan hubungan linear.

  o Kovarian positif menunjukkan korelasi positif, hubungan yang lebih kuat dengan nilai yang lebih tinggi.

  o Kovarian negatif menunjukkan korelasi negatif.

  o Kovarian yang mendekati nol menunjukkan korelasi yang lemah.

Formula for Covariance

$s\_xy = \Sigma_i (x\_i - mean(x)) (y\_i - mean(y)) / (n - 1)$

where:

  • s_xy adalah kovariansnya

  • Σ_i mewakili penjumlahan i dari 1 sampai n

  • x_i dan y_i merupakan elemen yang bersesuaian dari variabel x dan y

  • mean(x) dan mean(y) adalah mean sampel dari x dan y

  • n adalah jumlah elemen

Calculating Covariance in Python

- Pure Python:

```
n = len(x)
mean_x, mean_y = sum(x) / n, sum(y) / n
cov_xy = (sum((x[k] - mean_x) * (y[k] - mean_y) for k in
range(n)) / (n - 1))
```

- NumPy:

```
cov_matrix = np.cov(x_, y_)  # cov() calculates covariance matrix
cov_xy = cov_matrix[0, 1]  # element (0, 1) is covariance between
x and y
```

- pandas:

```
cov_xy = x__.cov(y__)
```

2.8 Correlation Coefficient

• Koefisien korelasi (koefisien korelasi Pearson product-moment), dilambangkan dengan r, adalah ukuran korelasi lainnya, yang merupakan versi standar kovarians.

• Berkisar antara -1 (korelasi negatif sempurna) hingga 1 (korelasi positif sempurna), dengan 0 menunjukkan korelasi lemah.

Formula for Correlation Coefficient

```
r = s_xy / (s_x * s_y)
```

where:

- r is the correlation coefficient
- s_xy is the covariance
- s_x and s_y are the standard deviations of x and y

Calculating Correlation Coefficient in Python

- Pure Python (setelah menghitung kovarians dan deviasi standar):

```
r = cov_xy / (std_x * std_y)
```

- SciPy:

```
r, p = scipy.stats.pearsonr(x_, y_)  # pearsonr() returns r and
p-value
```

- NumPy:

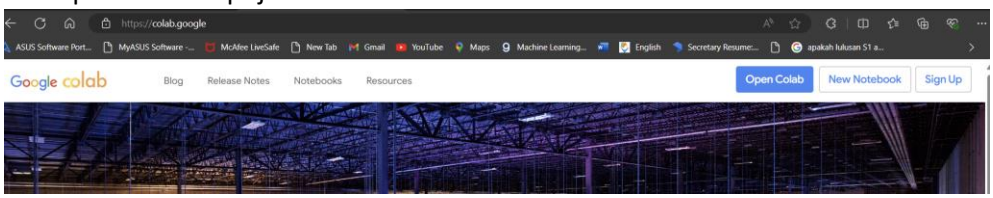
```
corr_matrix = np.corrcoef(x_, y_)  # corrcoef() calculates
correlation coefficient matrix
r = corr_matrix[0, 1]  # element (0, 1) is correlation
coefficient between x and y
```

- pandas:

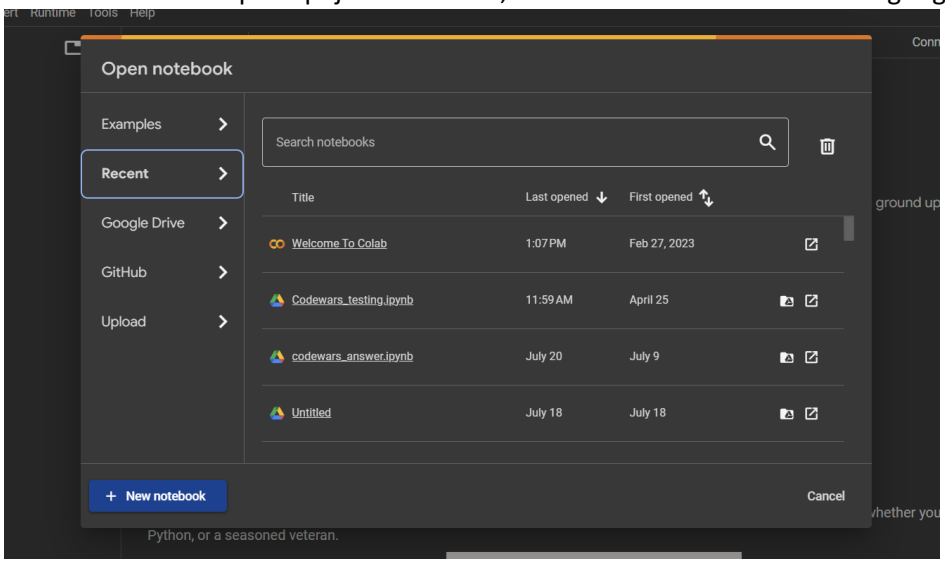
```
r = x__.corr(y__)
```

1.3 Langkah Persiapan

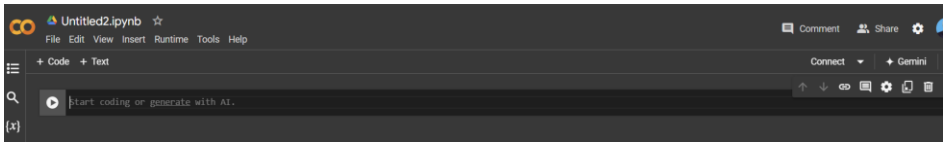
1. Membuka Google Colab
    a. Buka Google Colaboratory dengan link berikut https://colab.research.google.com/ .
    b. Klik Open Colab di pojok kanan atas



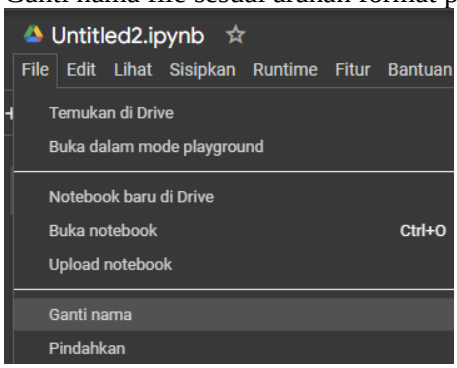
    c. Anda bisa login menggunakan akun Google.
    d. Klik New Notebook pada pojok kiri bawah, untuk membuka halaman baru google colab.



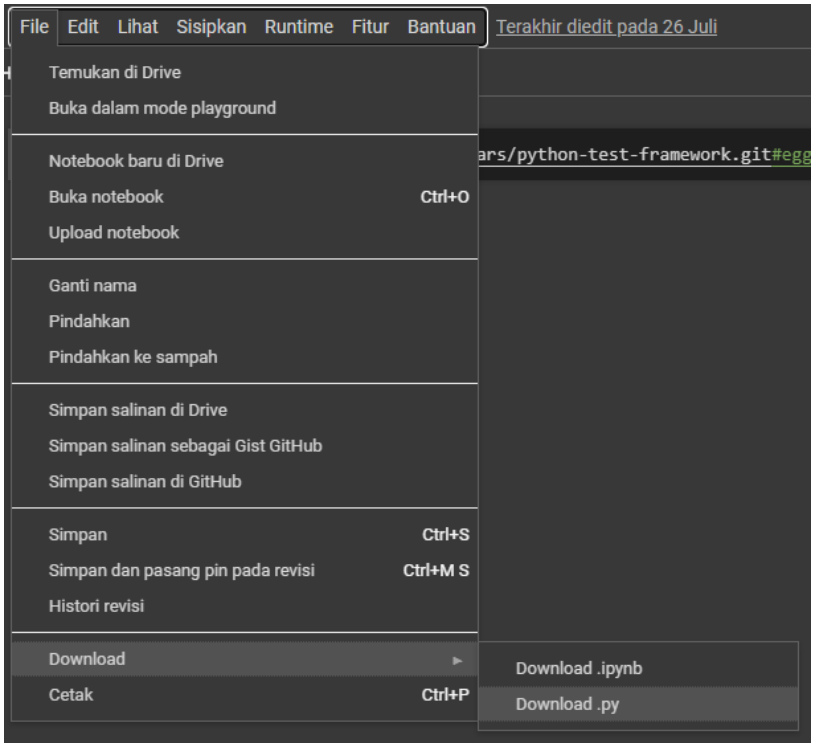
    e. Tampilan Google Colab.



    f. Ganti nama file sesuai arahan format pada praktikum



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


File
Edit
Lihat
Sisipkan
Runtime
Fitur
Bantuan
Terakhir diedit pada 26 Juli
Temukan di Drive
Buka dalam mode playground
Notebook baru di Drive
Buka notebook Ctrl+O
Upload notebook
Ganti nama
Pindahkan
Pindahkan ke sampah
Simpan salinan di Drive
Simpan salinan sebagai Gist GitHub
Simpan salinan di GitHub
Simpan Ctrl+S
Simpan dan pasang pin pada revisi Ctrl+M S
Histori revisi
Download
Cetak Ctrl+P
Download .ipynb
Download .py
ars/python-test-framework.git#egg

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: 11.965227397198378
  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
```

Tampilan Keseluruhan Kode

```
import pandas as pd

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

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

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

# Descriptive Statistics
```

```
def descriptive_statistics():
  print(data_load().describe())

# Print additional statistics for each variable (study hours and
exam scores)
def print_statistics():
  print("Study Hours:")
  print(f"  Mean: {data_load()['Study Hours'].mean()}")
  print(f"  Median: {data_load()['Study Hours'].median()}")
  print(f"  Standard Deviation: {data_load()['Study Hours'].std()}")
  print(f"  Skewness: {data_load()['Study Hours'].skew()}")
  print(f"  Quartiles: {data_load()['Study Hours'].quantile([0.25,
0.5, 0.75])}")

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

# Calculate the correlation coefficient between study hours and exam
scores
def correlation():
  correlation_coefficient = data_load()["Study
Hours"].corr(data_load()["Exam Score"])
  return correlation_coefficient

# Print the results
descriptive_statistics()
print_statistics()
print(f"Correlation Coefficient: {correlation()}")
```

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)**