DATA ANALYTICS

BAB 3 : DATA MANIPULATIONS

Praktikum 5

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

$s_{xy} = \Sigma_i (x_i - \text{mean}(x)) (y_i - \text{mean}(y)) / (n - 1)$

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:

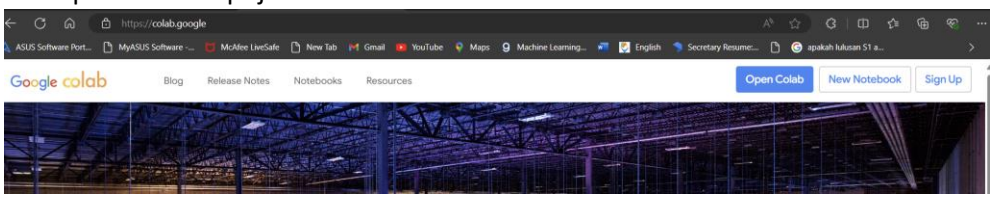
```
corr_matrix = np.corrcoef(x_, y_)  # corrcoef() calculates
correlation coefficient matrix
r = corr_matrix[0, 1]  # element (0, 1) is correlation
coefficient between x and y
```

- pandas:

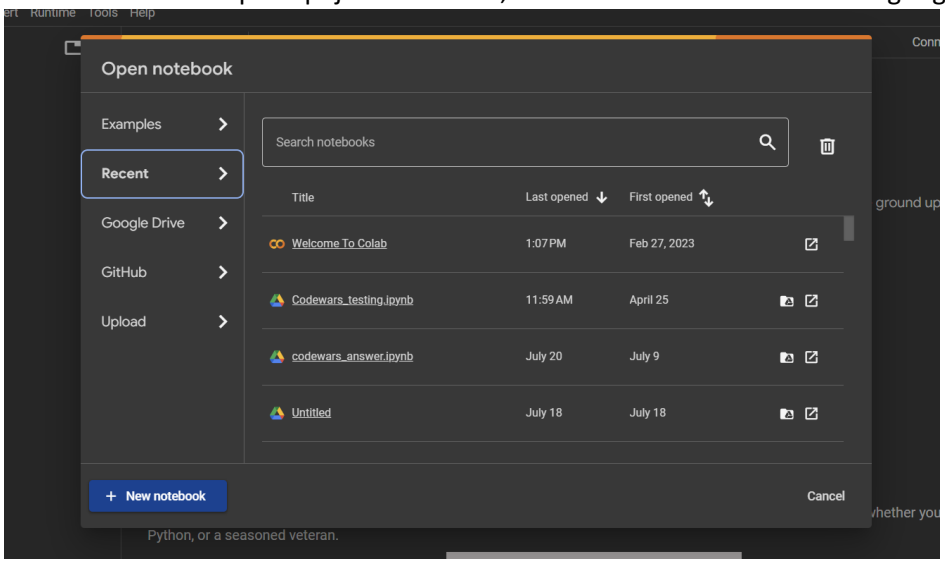
```
r = x__.corr(y__)
```

1.3 Langkah Persiapan

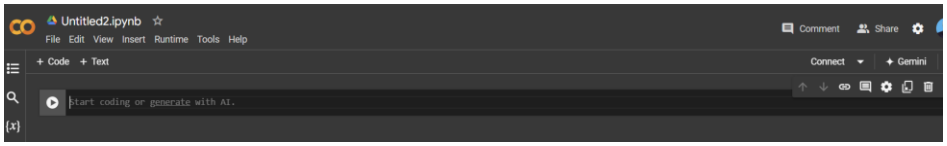
1. Membuka Google Colab
    a. Buka Google Colaboratory dengan link berikut https://colab.research.google.com/ .
    b. Klik Open Colab di pojok kanan atas



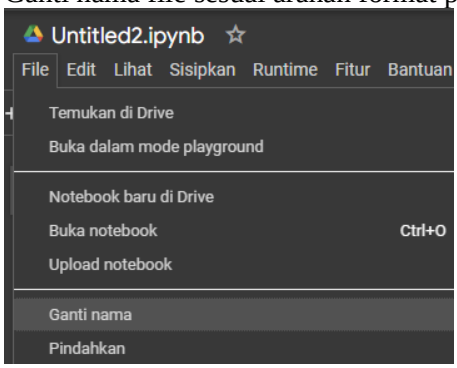
    c. Anda bisa login menggunakan akun Google.
    d. Klik New Notebook pada pojok kiri bawah, untuk membuka halaman baru google colab.



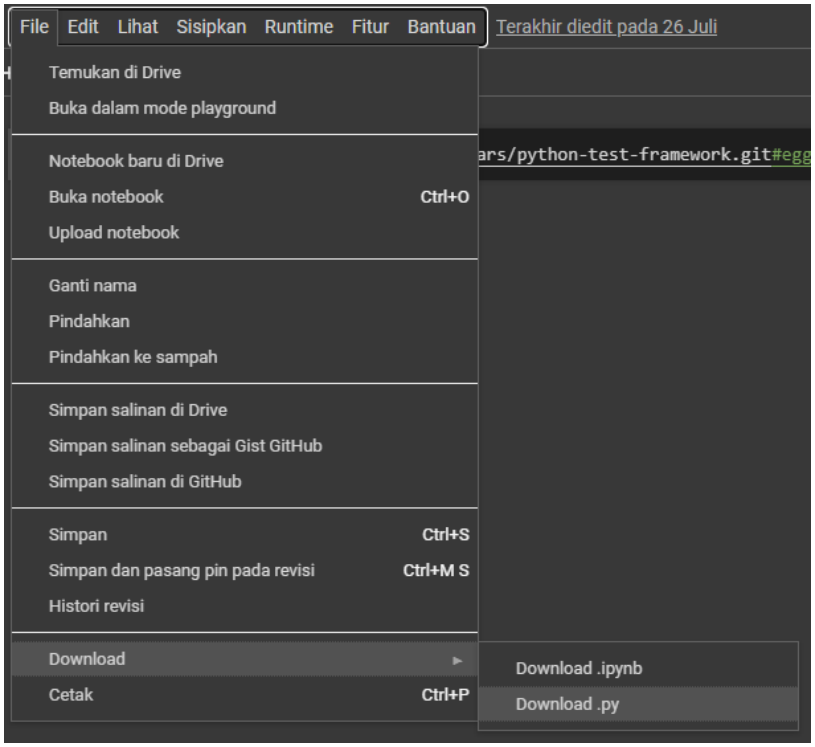
    e. Tampilan Google Colab.



    f. Ganti nama file sesuai arahan format pada praktikum



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



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/main/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()}")
```

Tampilan Keseluruhan Kode

```
import pandas as pd

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

# Load data
```

```
def load_data():
  data = pd.read_csv(url)
  return data

# Show first five rows
def head_rows():
  head_rows = load_data().head()
  return head_rows

# Calculate descriptive statistics
def describe_data():
  describe_data = load_data().describe()
  return describe_data

# 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




# 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

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

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 the pandas library using **import pandas as pd**.
2. Load Data:
   2.1 Define **url** variable (https://raw.githubusercontent.com/noora20FH/skripsi_noora2023/main/penjualan_elektronik.csv).
   2.2 Buatlah fungsi **data_load()** 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 variable **mean_penjualan, median_penjualan,** dan **mode_penjualan**
   - Panggil fungsi **load_data()** kolom ‘penjualan’, dan gunakan metode **mean(), median(),** atau **mode()** pada tiap variable

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

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**).