36. Analysis of Stock Price Variability

Program:-

```
import pandas as pd
import csv
# Create a dummy stock_data.csv file for demonstration
with open('stock_data.csv', 'w', newline='') as f:
   writer = csv.writer(f)
   writer.writerow(['Date', 'Close'])
   writer.writerow(['2023-01-01', 100.0])
   writer.writerow(['2023-01-02', 102.5])
   writer.writerow(['2023-01-03', 101.8])
   writer.writerow(['2023-01-04', 103.2])
   writer.writerow(['2023-01-05', 104.0])
# Read stock data from CSV
# CSV columns: Date, Close
df = pd.read_csv("stock_data.csv")
# Ensure Date is parsed as datetime (optional)
df["Date"] = pd.to_datetime(df["Date"])
# Basic statistics
mean_price = df["Close"].mean()
std_price = df["Close"].std()
min_price = df["Close"].min()
max_price = df["Close"].max()
price_range = max_price - min_price
print("Average closing price:", mean_price)
print("Standard deviation (variability):", std_price)
print("Minimum closing price:", min_price)
print("Maximum closing price:", max_price)
print("Price range:", price_range)
```

Output:-

```
Average closing price: 102.3

Standard deviation (variability): 1.5231546211727824

Minimum closing price: 100.0

Maximum closing price: 104.0

Price range: 4.0
```

37. Correlation Between Study Time and Exam Scores

Program:-

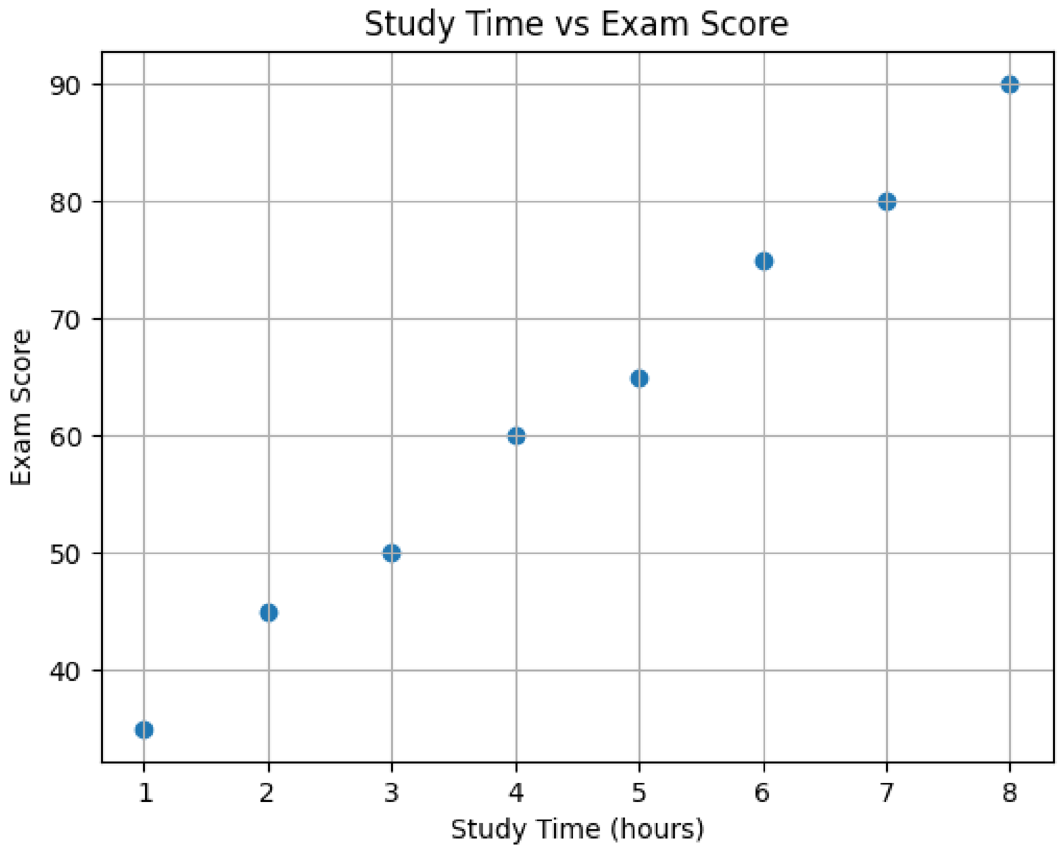
```
import pandas as pd
import matplotlib.pyplot as plt


# Sample data
data = {
   "study_time": [1, 2, 3, 4, 5, 6, 7, 8],
   "score":      [35, 45, 50, 60, 65, 75, 80, 90]
}
df = pd.DataFrame(data)


# Correlation
corr = df["study_time"].corr(df["score"])
print("Correlation between study time and score:", corr)
# Scatter plot
plt.scatter(df["study_time"], df["score"])
plt.xlabel("Study Time (hours)")
plt.ylabel("Exam Score")
plt.title("Study Time vs Exam Score")
plt.grid(True)
plt.show()
```

Output:-

```
Correlation between study time and score: 0.9975674982216793
```



38. Temperature Variability Analysis for Different Cities

Program:-

```
import pandas as pd
# Example: read from CSV (city, date, temperature)
# df = pd.read_csv("temperature_data.csv")
# Sample data for demo
data = {
   "City": ["CityA","CityA","CityA","CityB","CityB","CityB","CityC","CityC","CityC"],
   "Temperature": [30, 32, 31, 25, 27, 29, 35, 36, 34]
}
df = pd.DataFrame(data)
# Group by city
grouped = df.groupby("City")["Temperature"]
mean_temp = grouped.mean()
std_temp = grouped.std()
max_temp = grouped.max()
min_temp = grouped.min()
temp_range = max_temp - min_temp
print("Mean temperature:\n", mean_temp)
print("\nStandard deviation:\n", std_temp)
print("\nTemperature range:\n", temp_range)
# City with highest temperature range
city_highest_range = temp_range.idxmax()
# City with most consistent temperature (lowest std)
city_most_consistent = std_temp.idxmin()
print("\nCity with highest temperature range:", city_highest_range)
print("City with most consistent temperature:", city_most_consistent)
```

Output:-

```
Mean temperature:
 City
CityA    31.0
CityB    27.0
CityC    35.0
Name: Temperature, dtype: float64
Standard deviation:
 City
CityA    1.0
CityB    2.0
CityC    1.0
Name: Temperature, dtype: float64
Temperature range:
 City
CityA    2
CityB    4
CityC    2
Name: Temperature, dtype: int64
City with highest temperature range: CityB
City with most consistent temperature: CityA
```

39. Customer Clustering using K-Means (Transaction Data)

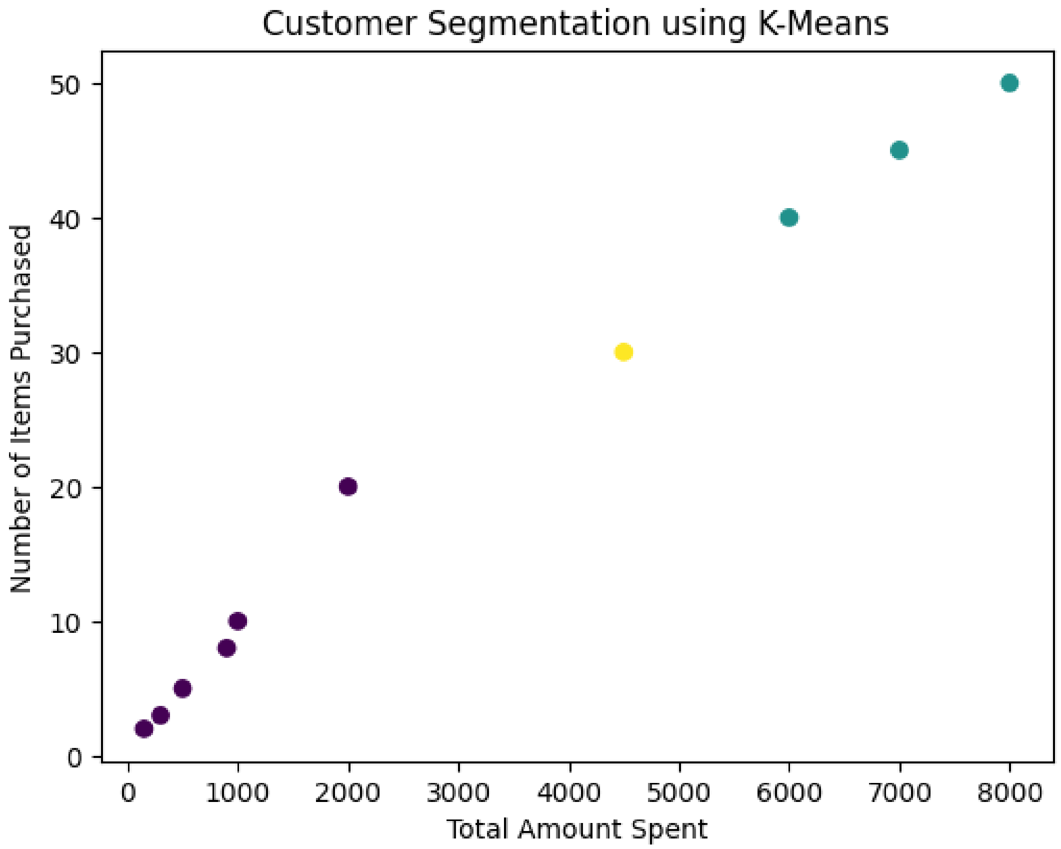
Program:-

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
# Sample transaction data
data = {
    "customer_id": [1,2,3,4,5,6,7,8,9,10],
    "total_amount_spent": [500, 2000, 150, 8000, 7000, 900, 300, 4500, 6000, 1000],
    "num_items": [5, 20, 2, 50, 45, 8, 3, 30, 40, 10]
}

df = pd.DataFrame(data)
X = df[["total_amount_spent", "num_items"]]
# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=0)
df["cluster"] = kmeans.fit_predict(X_scaled)
print("Customer clusters:")
print(df[["customer_id","total_amount_spent","num_items","cluster"]])
# Visualization
plt.scatter(df["total_amount_spent"], df["num_items"], c=df["cluster"])
plt.xlabel("Total Amount Spent")
plt.ylabel("Number of Items Purchased")
plt.title("Customer Segmentation using K-Means")
plt.show()
```

Output:-

```
Customer clusters:
   customer_id  total_amount_spent  num_items  cluster
0           1              500          5       0
1           2             2000         20         0
2           3              150          2       0
3           4             8000         50         1
4           5             7000         45         1
5           6              900          8       0
6           7              300          3       0
7           8             4500         30         2
8           9             6000         40         1
9          10              1000          10        0
```



40. Soccer Players Data Analysis from CSV

Program:-

```
import pandas as pd
import matplotlib.pyplot as plt
import csv
# Create a dummy players.csv file for demonstration
with open('players.csv', 'w', newline=") as f:
    writer = csv.writer(f)
    writer.writerow(['name', 'goals', 'salary', 'age', 'position'])
    writer.writerow(['Player A', 20, 1000000, 25, 'Forward'])
    writer.writerow(['Player B', 15, 800000, 28, 'Midfielder'])
    writer.writerow(['Player C', 22, 1200000, 24, 'Forward'])
    writer.writerow(['Player D', 10, 700000, 30, 'Defender'])
    writer.writerow(['Player E', 5, 500000, 32, 'Defender'])
    writer.writerow(['Player F', 18, 900000, 26, 'Midfielder'])
    writer.writerow(['Player G', 3, 400000, 35, 'Goalkeeper'])
    writer.writerow(['Player H', 25, 1500000, 23, 'Forward'])
# Read data from CSV file
df = pd.read_csv("players.csv")
# Top 5 players by goals
top_goals = df.sort_values(by="goals", ascending=False).head(5)
print("Top 5 players by goals:\n", top_goals)
# Top 5 players by salary
top_salary = df.sort_values(by="salary", ascending=False).head(5)
print("\nTop 5 players by salary:\n", top_salary)
# Average age
avg_age = df["age"].mean()
print("\nAverage age:", avg_age)
```

Output:-

```
Top 5 players by goals:
       name  goals   salary  age    position
7  Player H     25  1500000   23     Forward
2  Player C     22  1200000   24     Forward
0  Player A     20  1000000   25     Forward
5  Player F     18   900000   26  Midfielder
1  Player B     15   800000   28  Midfielder
Top 5 players by salary:
       name  goals   salary  age    position
7  Player H     25  1500000   23     Forward
2  Player C     22  1200000   24     Forward
0  Player A     20  1000000   25     Forward
5  Player F     18   900000   26  Midfielder
1  Player B     15   800000   28  Midfielder
Average age: 27.875
Players above average age:
       name  goals  salary  age    position
1  Player B     15  800000   28  Midfielder
3  Player D     10  700000   30    Defender
4  Player E      5  500000   32    Defender
6  Player G      3  400000   35  Goalkeeper
Distribution of players by position:
 position
Forward       3
Midfielder    2
Defender      2
Goalkeeper    1
Name: count, dtype: int64
```