Sushant University

soaring high

Name – Aman Mirza

CLASS- BCA CYBER SECURITY(SEM-5)

ROLL NO.- 230BCA208

SUBJECT- EDA (Exploratory Data Analysis)

Exploratory Data Analysis on Students Performance Dataset

Common Setup Code

CODE:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

plt.style.use("default")
sns.set(font_scale=1.1)

df = pd.read_csv("StudentsPerformance.csv")
num_cols = ["math score", "reading score", "writing score"]
```

Q1. Import & View Data

QUESTION:

1. Import the dataset and show the first few rows.

CODE:

```
df.head()
```

OUTPUT:

```
  gender race/ethnicity parental level of education         lunch  \
0  female        group B           bachelor's degree      standard
1  female        group C                some college      standard
2  female        group B             master's degree      standard
3    male        group A          associate's degree  free/reduced
4    male        group C                some college      standard

  test preparation course  math score  reading score  writing score
0                    none          72             72             74
1               completed          69             90             88
2                    none          90             95             93
3                    none          47             57             44
4                    none          76             78             75
```

Q2. Missing Values

QUESTION:

2. Identify missing values and handle them.

CODE:

```
# check missing values
df.isnull().sum()

# (in this dataset there are no missing values, but generic handling code)
for col in num_cols:
    if df[col].isnull().sum() > 0:
        df[col].fillna(df[col].median(), inplace=True)

for col in df.columns:
    if df[col].dtype == "object" and df[col].isnull().sum() > 0:
        df[col].fillna(df[col].mode()[0], inplace=True)
```

OUTPUT:

```
gender                         0
race/ethnicity                 0
parental level of education    0
lunch                          0
test preparation course        0
math score                     0
reading score                  0
writing score                  0
dtype: int64
```

Q3. Dataset Description

QUESTION:

3. Describe the dataset — data types, shape, summary statistics.

CODE:

```
df.shape
df.dtypes
df[num_cols].describe()
```

OUTPUT:

Shape: (1000, 8)

```
Dtypes:
gender                         object
race/ethnicity                 object
parental level of education    object
lunch                          object
test preparation course        object
math score                      int64
reading score                   int64
writing score                   int64
dtype: object
```

Summary statistics:

	math score	reading score	writing score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000
max	100.00000	100.000000	100.000000

Q4. Correlation Between Numerical Variables

QUESTION:

4. Find correlations between numerical variables (math, reading, writing scores).

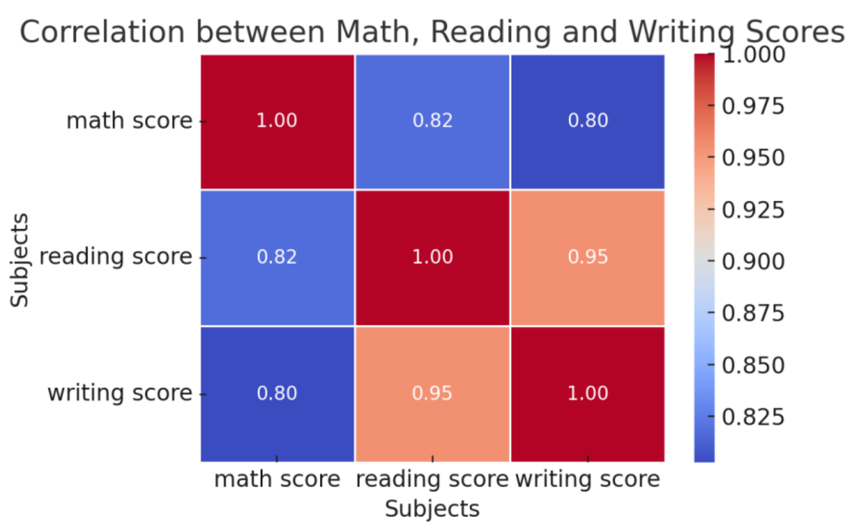
CODE:

```
corr_scores = df[num_cols].corr()
print(corr_scores)

import seaborn as sns
plt.figure(figsize=(6,4))
sns.heatmap(corr_scores, annot=True, cmap="coolwarm", fmt=".2f",
linewidths=0.5)
plt.title("Correlation between Math, Reading and Writing Scores")
plt.xlabel("Subjects")
plt.ylabel("Subjects")
plt.tight_layout()
plt.show()
```

OUTPUT:

	math score	reading score	writing score
math score	1.000000	0.817580	0.802642
reading score	0.817580	1.000000	0.954598
writing score	0.802642	0.954598	1.000000

GRAPH:



Q5(a). Distribution of Scores

QUESTION:

5(a). Use visualizations to explore the distribution of scores in each subject.

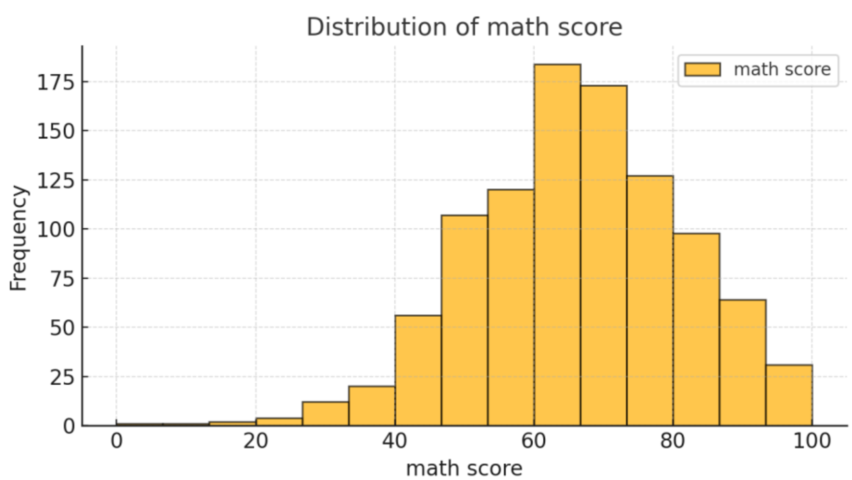
CODE:

```
for col in num_cols:
    plt.figure(figsize=(7,4))
    plt.hist(df[col], bins=15, edgecolor="black", alpha=0.7, label=col)
    plt.title(f"Distribution of {col}", fontsize=14)
    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.grid(True, linestyle="--", alpha=0.5)
    plt.legend()
    plt.tight_layout()
    plt.show()
```

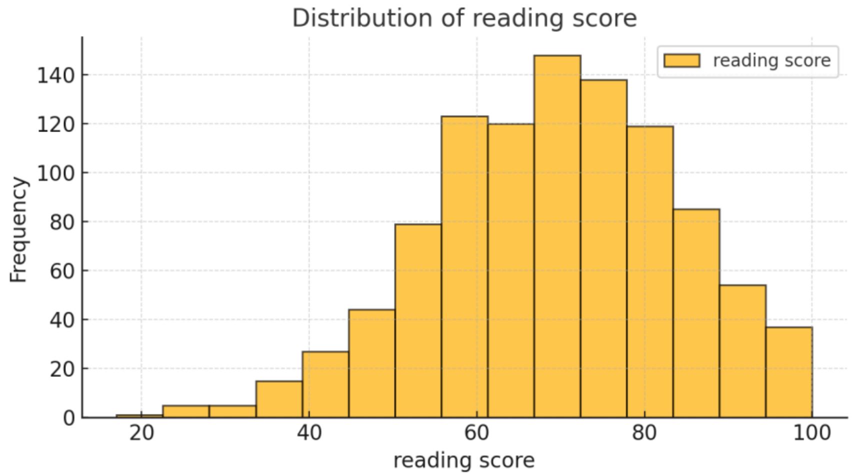
OUTPUT:

Three histograms for math score, reading score and writing score.

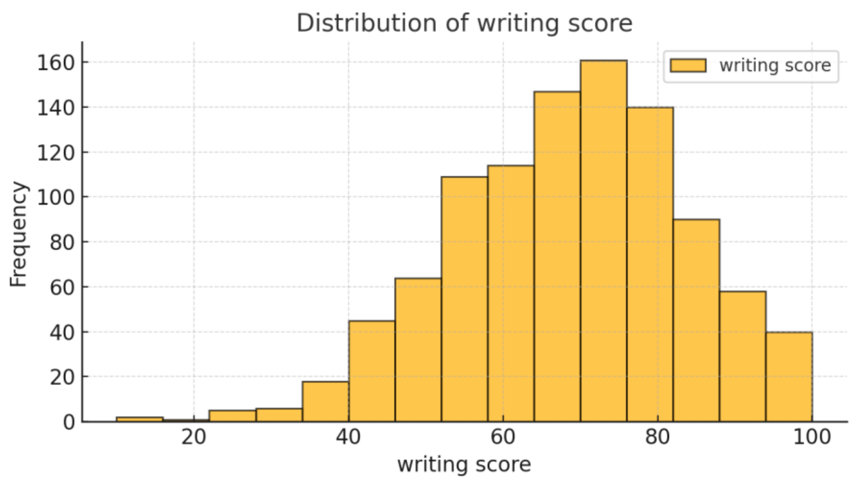
GRAPH: Histogram of math score



GRAPH: Histogram of reading score



GRAPH: Histogram of writing score



Additional: Boxplot of Scores

QUESTION:

Boxplot visualization of three subjects.

CODE:

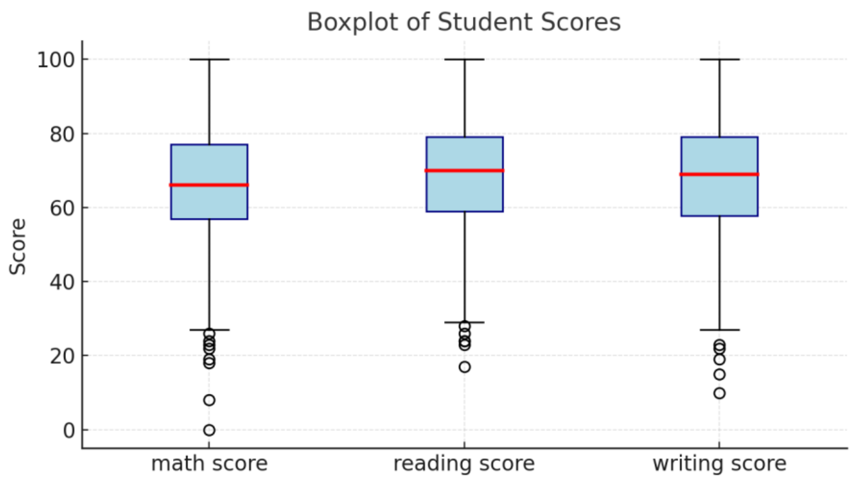
```
plt.figure(figsize=(7,4))
plt.boxplot(
    [df["math score"], df["reading score"], df["writing score"]],
    labels=num_cols,
    patch_artist=True,
    boxprops=dict(facecolor="lightblue", color="navy"),
    medianprops=dict(color="red", linewidth=2)
)
plt.title("Boxplot of Student Scores", fontsize=14)
```

```python
plt.ylabel("Score")
plt.grid(True, linestyle="--", alpha=0.4)
plt.tight_layout()
plt.show()
```

OUTPUT:

Single boxplot comparing math, reading and writing scores with styling.

GRAPH: Boxplot of Scores



Q5(b). Gender and Performance

QUESTION:

5(b). Relationship between gender and performance.

CODE:

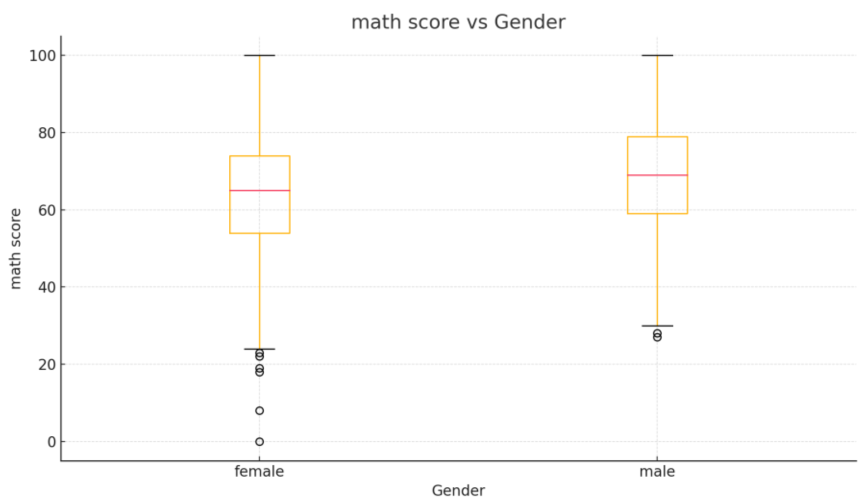
```python
# Average scores by gender
df.groupby("gender")[num_cols].mean()

# Boxplots of scores vs gender
for col in num_cols:
  plt.figure(figsize=(6,4))
  df.boxplot(column=col, by="gender")
  plt.title(f"{col} vs Gender")
  plt.suptitle("")
  plt.xlabel("Gender")
  plt.ylabel(col)
  plt.grid(True, linestyle="--", alpha=0.5)
```

```
plt.tight_layout()
plt.show()
```

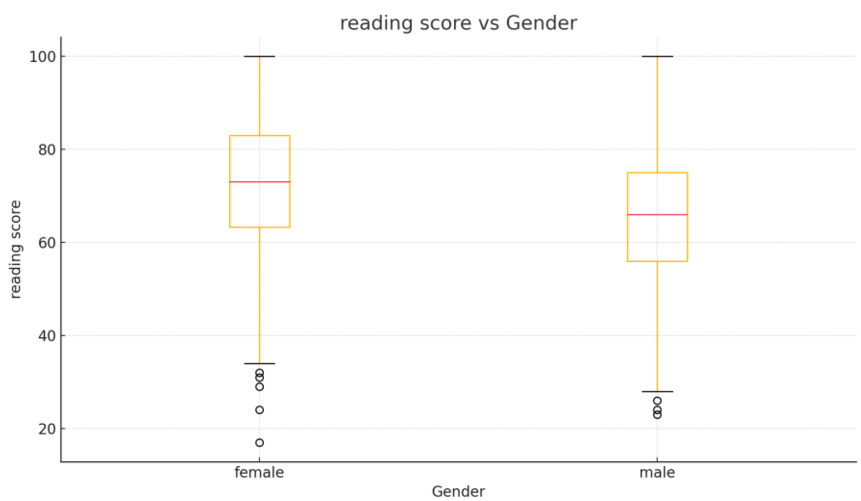
OUTPUT:

	math score	reading score	writing score
gender			
female	63.63	72.61	72.47
male	68.73	65.47	63.31

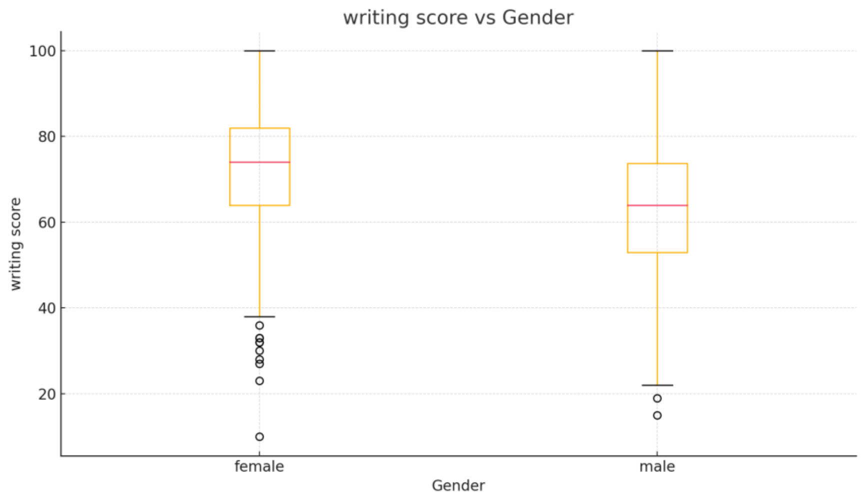
GRAPH: math score vs Gender



GRAPH: reading score vs Gender



GRAPH: writing score vs Gender



Q5(c). Parental Education Effect

QUESTION:

5(c). Effect of parental education on student scores.

CODE:

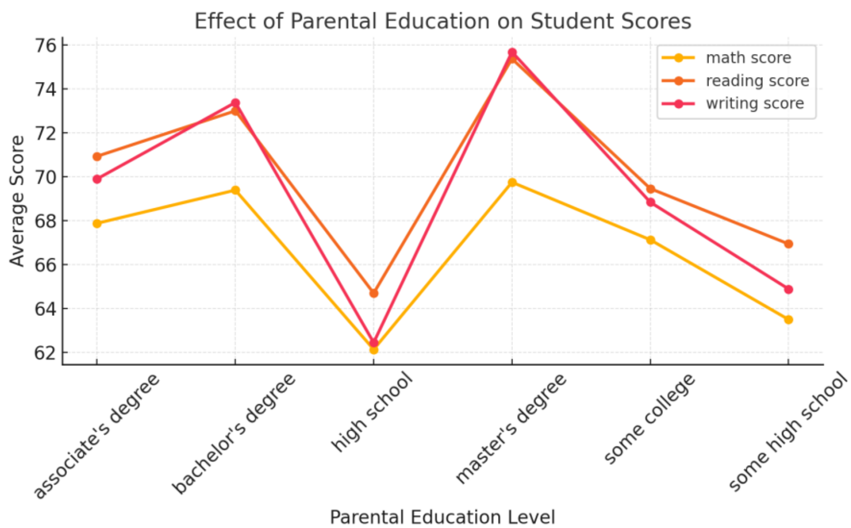
```
avg_by_parent = df.groupby("parental level of
education")[num_cols].mean()

plt.figure(figsize=(8,5))
for col in num_cols:
   plt.plot(avg_by_parent.index, avg_by_parent[col], marker="o",
linewidth=2, label=col)

plt.title("Effect of Parental Education on Student Scores",
fontsize=14)
plt.xlabel("Parental Education Level")
plt.ylabel("Average Score")
plt.grid(True, linestyle="--", alpha=0.4)
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```

OUTPUT:

	math score	reading score	writing score
parental level of education			
associate's degree	67.88	70.93	69.90
bachelor's degree	69.39	73.00	73.38

high school	62.14	64.70	62.45
master's degree	69.75	75.37	75.68
some college	67.13	69.46	68.84
some high school	63.50	66.94	64.89

GRAPH: Effect of Parental Education on Scores



Q6. Outlier Detection

QUESTION:

6. Detect any outliers and explain their potential effect (IQR and Z-score).

CODE:

```
from scipy import stats

def detect_outliers_iqr(series):
  Q1 = series.quantile(0.25)
  Q3 = series.quantile(0.75)
  IQR = Q3 - Q1
  lower = Q1 - 1.5 * IQR
  upper = Q3 + 1.5 * IQR
  mask = (series < lower) | (series > upper)
  return mask.sum(), lower, upper

for col in num_cols:
  count, lb, ub = detect_outliers_iqr(df[col])
  print(col, "IQR outliers:", count, "lower:", lb, "upper:", ub)

for col in num_cols:
  z_scores = np.abs(stats.zscore(df[col]))
  print(col, "Z-score outliers:", (z_scores > 3).sum())
```

OUTPUT:

IQR Method:
math score: 8 outliers (lower=27.00, upper=107.00)
reading score: 6 outliers (lower=29.00, upper=109.00)
writing score: 5 outliers (lower=25.88, upper=110.88)

Z-Score Method:
math score: 4 outliers with |z| > 3
reading score: 4 outliers with |z| > 3
writing score: 4 outliers with |z| > 3

Positive Correlation Example

QUESTION:

Positive correlation – considers height vs. weight of 10 students and maps the positive correlation using heatmap.

CODE:

```python
height_weight_data = {
   "height_cm":[150,155,160,162,165,168,170,173,175,178],
   "weight_kg":[45,48,52,54,57,60,62,65,67,70]
}
hw_df = pd.DataFrame(height_weight_data)
print(hw_df)
print(hw_df.corr())

plt.figure(figsize=(5,4))
sns.heatmap(hw_df.corr(), annot=True, cmap="coolwarm",
fmt=".2f", linewidths=0.5)
plt.title("Positive Correlation: Height vs Weight")
plt.xlabel("Variables")
plt.ylabel("Variables")
plt.tight_layout()
plt.show()
```

OUTPUT:

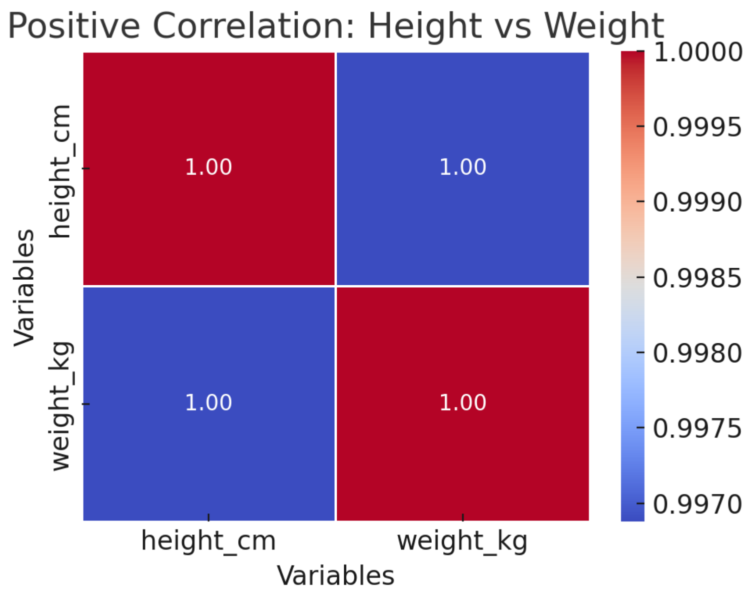
Height vs Weight Data:

	height_cm	weight_kg
0	150	45
1	155	48
2	160	52
3	162	54
4	165	57
5	168	60
6	170	62
7	173	65
8	175	67
9	178	70

Correlation Matrix:

	height_cm	weight_kg
height_cm	1.000000	0.996879
weight_kg	0.996879	1.000000

GRAPH: Positive Correlation Heatmap (Height vs Weight)



Negative Correlation Example

QUESTION:

Negative correlation – TV watching hours vs. study score, mapped using heatmap.

CODE:

```
tv_study_data = {
  "tv_hours_per_day":[1,2,3,4,5,6,7,2.5,4.5,5.5],
  "study_score":[90,85,78,70,65,60,55,82,68,62]
}
ts_df = pd.DataFrame(tv_study_data)
print(ts_df)
print(ts_df.corr())

plt.figure(figsize=(5,4))
sns.heatmap(ts_df.corr(), annot=True, cmap="coolwarm", fmt=".2f",
linewidths=0.5)
plt.title("Negative Correlation: TV Hours vs Study Score")
plt.xlabel("Variables")
plt.ylabel("Variables")
plt.tight_layout()
plt.show()
```

OUTPUT:

TV Hours vs Study Score Data:
  tv_hours_per_day study_score

0	1.0	90
1	2.0	85
2	3.0	78
3	4.0	70
4	5.0	65
5	6.0	60
6	7.0	55
7	2.5	82
8	4.5	68
9	5.5	62

Correlation Matrix:

	tv_hours_per_day	study_score
tv_hours_per_day	1.000000	-0.996229
study_score	-0.996229	1.000000

GRAPH: Negative Correlation Heatmap (TV Hours vs Study Score)