Artificial Intelligence - ISC Class 11
Practical Assignment - Final Term

Question 1: School Marks - Data Cleaning & Outlier Detection

Step 1 & 2: Create Database, Table and Insert Data

```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sqlalchemy import create_engine

# Create database and table using mysql.connector
conn = mysql.connector.connect(host="localhost", user="root", password="root")
cursor = conn.cursor()
cursor.execute("CREATE DATABASE IF NOT EXISTS school")
cursor.execute("USE school")
cursor.execute("DROP TABLE IF EXISTS schooldata")
cursor.execute("""
    CREATE TABLE schooldata (
        Name VARCHAR(10),
        Marks INT
    )
""")
data = [("A", 85), ("B", None), ("C", 95), ("D", -10), ("E", 300), ("F", 88), ("G", 92)]
for name, marks in data:
    cursor.execute("INSERT INTO schooldata VALUES (%s, %s)", (name, marks))
conn.commit()
cursor.close()
conn.close()
print("Database and table created successfully.")
```

```
Database and table created successfully.
```

Step 3 & 4: Import Database in Python and Display DataFrame

```
engine = create_engine("mysql+mysqlconnector://root:root@localhost/school")
df = pd.read_sql("SELECT * FROM schooldata", engine)
print("DataFrame:")
print(df)
```

```
DataFrame:
  Name  Marks
0    A   85.0
1    B    NaN
2    C   95.0
3    D  -10.0
4    E  300.0
5    F   88.0
6    G   92.0
```

Step 5: Replace Missing Values with Mean

```
mean_marks = df["Marks"].mean()
df["Marks"] = df["Marks"].fillna(mean_marks)
print(f"Mean of Marks (excluding null): {mean_marks:.2f}")
print("\nAfter replacing missing values:")
print(df)
```

```
Mean of Marks (excluding null): 108.33

After replacing missing values:
  Name       Marks
0    A   85.000000
1    B  108.333333
2    C   95.000000
3    D  -10.000000
4    E  300.000000
5    F   88.000000
6    G   92.000000
```

Step 6: Replace Inconsistent Marks

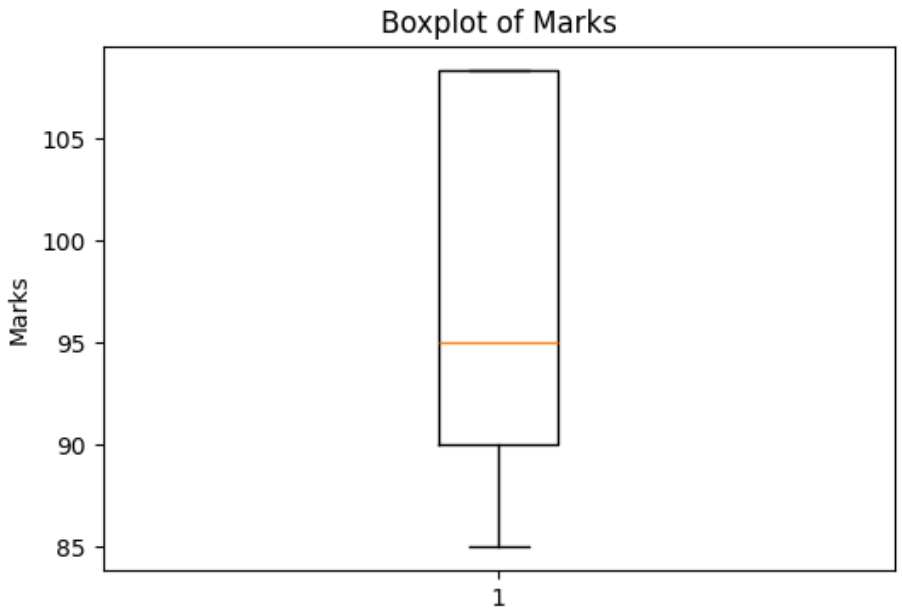
Marks should be between 0 and 100.

```
df.loc[df["Marks"] < 0, "Marks"] = mean_marks
df.loc[df["Marks"] > 100, "Marks"] = mean_marks
print("After replacing inconsistent marks:")
print(df)
```

```
After replacing inconsistent marks:
  Name       Marks
0    A   85.000000
1    B  108.333333
2    C   95.000000
3    D  108.333333
4    E  108.333333
5    F   88.000000
6    G   92.000000
```

Step 7: Detect Outliers (Visual & Statistical)

```
# Visual Detection - Boxplot
plt.figure(figsize=(6, 4))
plt.boxplot(df["Marks"])
plt.title("Boxplot of Marks")
plt.ylabel("Marks")
plt.show()

# Statistical Detection - IQR Method
Q1 = df["Marks"].quantile(0.25)
Q3 = df["Marks"].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = df[(df["Marks"] < lower_bound) | (df["Marks"] > upper_bound)]
print(f"Q1 = {Q1}, Q3 = {Q3}, IQR = {IQR}")
print(f"Lower Bound = {lower_bound}, Upper Bound = {upper_bound}")
```




```
Q1 = 90.0, Q3 = 108.33333333333333, IQR = 18.33333333333333
Lower Bound = 62.50000000000001, Upper Bound = 135.83333333333331
```

Step 8: Print Cleaned Data and Outliers

```
print("Cleaned Data:")
print(df)
print("\nOutliers:")
print(outliers if not outliers.empty else "No outliers detected.")
```

```
Cleaned Data:
  Name       Marks
0    A   85.000000
1    B  108.333333
2    C   95.000000
3    D  108.333333
4    E  108.333333
5    F   88.000000
6    G   92.000000

Outliers:
No outliers detected.
```

Question 2: Temperature - Data Cleaning & Outlier Detection

Step 1 & 2: Create Database, Table and Insert Data

```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```
conn = mysql.connector.connect(host="localhost", user="root", password="root")
cursor = conn.cursor()
cursor.execute("CREATE DATABASE IF NOT EXISTS temperature")
cursor.execute("USE temperature")
cursor.execute("DROP TABLE IF EXISTS tempdata")
cursor.execute("""
    CREATE TABLE tempdata (
        Day VARCHAR(10),
        Temperature INT
    )
""")
data = [("Mon", 30), ("Tue", None), ("Wed", 45), ("Thu", -50), ("Fri", 100), ("Sat", 32), ("Sun", 35)]
for day, temp in data:
    cursor.execute("INSERT INTO tempdata VALUES (%s, %s)", (day, temp))
conn.commit()
cursor.close()
conn.close()
print("Database and table created successfully.")
```

```
Database and table created successfully.
```

Step 3 & 4: Import Database in Python and Display DataFrame

```
engine = create_engine("mysql+mysqlconnector://root:root@localhost/temperature")
df = pd.read_sql("SELECT * FROM tempdata", engine)
print("DataFrame:")
print(df)
```

```
DataFrame:
   Day  Temperature
0  Mon         30.0
1  Tue          NaN
2  Wed         45.0
3  Thu        -50.0
4  Fri        100.0
5  Sat         32.0
6  Sun         35.0
```

Step 5: Replace Missing Values with Mean

```
mean_temp = df["Temperature"].mean()
df["Temperature"] = df["Temperature"].fillna(mean_temp)
print(f"Mean of Temperature (excluding null): {mean_temp:.2f}")
print("\nAfter replacing missing values:")
print(df)
```

```
Mean of Temperature (excluding null): 32.00

After replacing missing values:
   Day  Temperature
0  Mon         30.0
1  Tue         32.0
2  Wed         45.0
3  Thu        -50.0
4  Fri        100.0
5  Sat         32.0
6  Sun         35.0
```

Step 6: Replace Inconsistent Temperatures

Temperature should be between -10 and 55 degrees C.

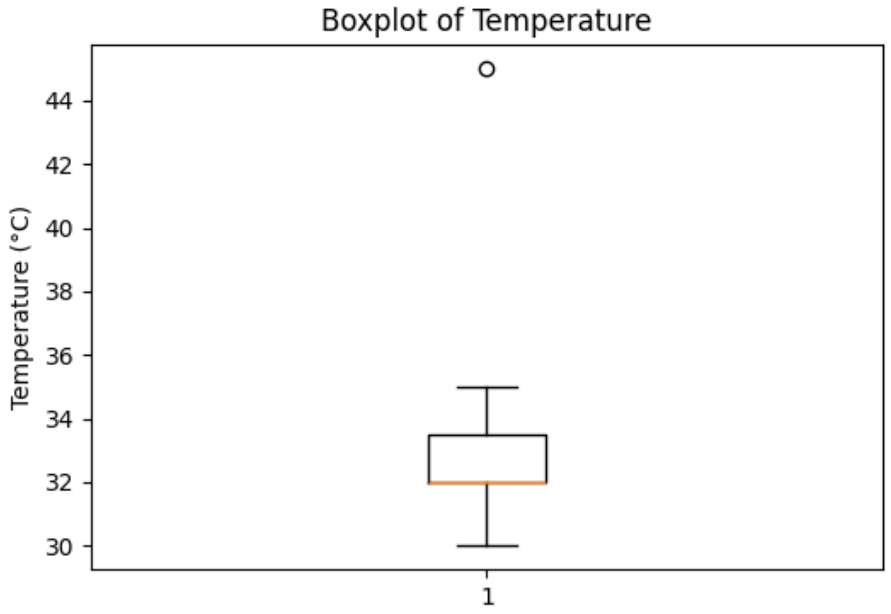
```
df.loc[df["Temperature"] < -10, "Temperature"] = mean_temp
df.loc[df["Temperature"] > 55, "Temperature"] = mean_temp
print("After replacing inconsistent temperatures:")
print(df)
```

```
After replacing inconsistent temperatures:
   Day  Temperature
0  Mon         30.0
1  Tue         32.0
2  Wed         45.0
3  Thu         32.0
4  Fri         32.0
5  Sat         32.0
6  Sun         35.0
```

Step 7: Detect Outliers (Visual & Statistical)

```
# Visual Detection - Boxplot
plt.figure(figsize=(6, 4))
plt.boxplot(df["Temperature"])
plt.title("Boxplot of Temperature")
plt.ylabel("Temperature (°C)")
plt.show()

# Statistical Detection - IQR Method
Q1 = df["Temperature"].quantile(0.25)
```

```
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = df[(df["Temperature"] < lower_bound) | (df["Temperature"] > upper_bound)]
print(f"Q1 = {Q1}, Q3 = {Q3}, IQR = {IQR}")
print(f"Lower Bound = {lower_bound}, Upper Bound = {upper_bound}")
```




```
Q1 = 32.0, Q3 = 33.5, IQR = 1.5
Lower Bound = 29.75, Upper Bound = 35.75
```

Step 8: Print Cleaned Data and Outliers

```
print("Cleaned Data:")
print(df)
print("\nOutliers:")
print(outliers if not outliers.empty else "No outliers detected.")
```

```
Cleaned Data:
   Day  Temperature
0  Mon         30.0
1  Tue         32.0
2  Wed         45.0
3  Thu         32.0
4  Fri         32.0
5  Sat         32.0
6  Sun         35.0

Outliers:
   Day  Temperature
2  Wed         45.0
```

Question 3: Store Details - Data Cleaning & Outlier Detection

Step 1 & 2: Create Database, Table and Insert Data

```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sqlalchemy import create_engine

conn = mysql.connector.connect(host="localhost", user="root", password="root")
cursor = conn.cursor()
cursor.execute("CREATE DATABASE IF NOT EXISTS storedb")
cursor.execute("USE storedb")
cursor.execute("DROP TABLE IF EXISTS sales_data")
cursor.execute("""
    CREATE TABLE sales_data (
        orderid INT,
        product VARCHAR(20),
        category VARCHAR(20),
        price INT,
        quantity INT,
        city VARCHAR(20)
    )
""")
data = [
    (101, "Shirt",   "Casual", 1200, 2,   "Prayagraj"),
    (102, "t-shirt", "Casual", None, 1,   "Prayagraj"),
    (103, "Shirt",   "Casual", 1200, 1,   "Lucknow"),
```

```
    (104, "Blazer",  "Formal", 8000, 1,   "Mumbai"),
    (105, "Shirt",   "Formal", 1200, 2,   "Mumbay"),
    (106, "t-shirt", "Casual", 700,  120, "Delhi"),
    (107, "Jeans",   "Denim",  None, 1,   "Jaipur"),
    (108, "Shirt",   "Casual", 950,  2,   "Prayagraj"),
    (109, "Blazer",  "Formal", 8500, 1,   "Mumbai"),
    (110, "t-shirt", "casual", 750,  2,   "Delhi"),
]
for row in data:
    cursor.execute("INSERT INTO sales_data VALUES (%s, %s, %s, %s, %s, %s)", row)
conn.commit()
cursor.close()
conn.close()
print("Database and table created successfully.")
```

```
Database and table created successfully.
```

Step 3 & 4: Import Database in Python and Display DataFrame

```
engine = create_engine("mysql+mysqlconnector://root:root@localhost/storedb")
df = pd.read_sql("SELECT * FROM sales_data", engine)
print("DataFrame:")
print(df)
```

```
DataFrame:
   orderid  product category   price  quantity       city
0      101    Shirt   Casual  1200.0         2  Prayagraj
1      102  t-shirt   Casual     NaN         1  Prayagraj
2      103    Jeans    Denim  1500.0         0      Delhi
3      104   Blazer   Formal  8000.0         1     Mumbai
4      105    Shirt   Formal  1200.0         2     Mumbay
5      106  t-shirt   Casual   700.0       120      Delhi
6      107    Jeans    Denim     NaN         1     Jaipur
7      108    Shirt   Casual   950.0         2  Prayagraj
8      109   Blazer   Formal  8500.0         1     Mumbai
9      110  t-shirt   casual   750.0         2      Delhi
```

Step 5: Replace Missing Values with Mean

```
mean_price = df["price"].mean()
df["price"] = df["price"].fillna(mean_price)
print(f"Mean of Price (excluding null): {mean_price:.2f}")
print("\nAfter replacing missing values:")
print(df)
```

```
Mean of Price (excluding null): 2850.00

After replacing missing values:
   orderid  product category   price  quantity       city
0      101    Shirt   Casual  1200.0         2  Prayagraj
1      102  t-shirt   Casual  2850.0         1  Prayagraj
2      103    Jeans    Denim  1500.0         0      Delhi
3      104   Blazer   Formal  8000.0         1     Mumbai
4      105    Shirt   Formal  1200.0         2     Mumbay
5      106  t-shirt   Casual   700.0       120      Delhi
6      107    Jeans    Denim  2850.0         1     Jaipur
7      108    Shirt   Casual   950.0         2  Prayagraj
8      109   Blazer   Formal  8500.0         1     Mumbai
9      110  t-shirt   casual   750.0         2      Delhi
```

Step 6: Replace Inconsistent Values

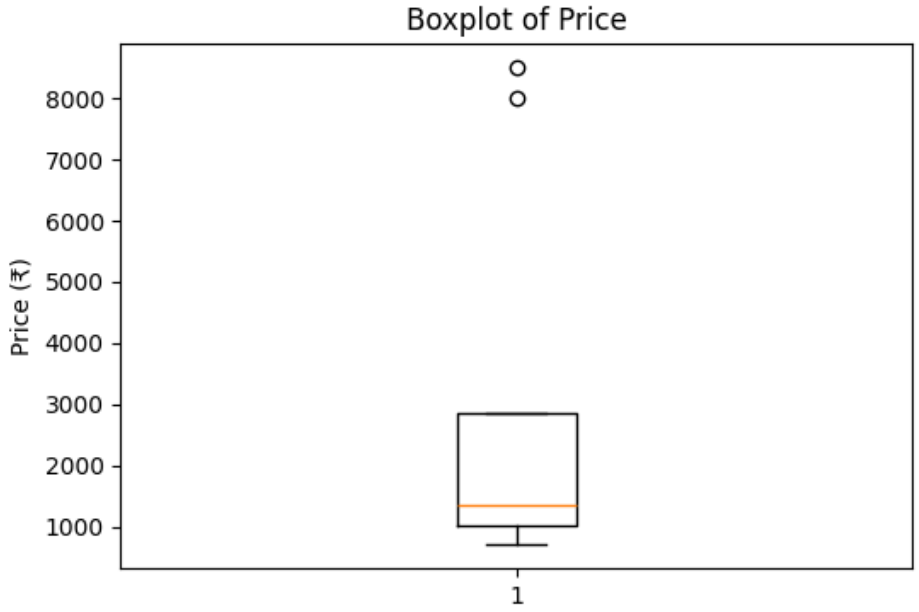
```
# Fix category: 'casual' -> 'Casual'
df["category"] = df["category"].replace({"casual": "Casual"})
# Fix city: 'Mumbay' -> 'Mumbai'
df["city"] = df["city"].replace({"Mumbay": "Mumbai"})
# Fix quantity: 0 and 120 are unrealistic
mean_qty = df["quantity"].mean()
df.loc[df["quantity"] == 0, "quantity"] = round(mean_qty)
df.loc[df["quantity"] > 10, "quantity"] = round(mean_qty)
print("After replacing inconsistent values:")
print(df)
```

```
After replacing inconsistent values:
   orderid  product category   price  quantity       city
0      101    Shirt   Casual  1200.0         2  Prayagraj
1      102  t-shirt   Casual  2850.0         1  Prayagraj
2      103    Jeans    Denim  1500.0        13      Delhi
3      104   Blazer   Formal  8000.0         1     Mumbai
4      105    Shirt   Formal  1200.0         2     Mumbai
5      106  t-shirt   Casual   700.0        13      Delhi
6      107    Jeans    Denim  2850.0         1     Jaipur
7      108    Shirt   Casual   950.0         2  Prayagraj
8      109   Blazer   Formal  8500.0         1     Mumbai
9      110  t-shirt   Casual   750.0         2      Delhi
```

Step 7: Detect Outliers (Visual & Statistical)

```
# Visual Detection - Boxplot for Price
```

```
plt.boxplot(df["price"])
plt.title("Boxplot of Price")
plt.ylabel("Price (₹)")
plt.show()

# Statistical Detection - IQR Method for Price
Q1 = df["price"].quantile(0.25)
Q3 = df["price"].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = df[(df["price"] < lower_bound) | (df["price"] > upper_bound)]
print(f"Q1 = {Q1}, Q3 = {Q3}, IQR = {IQR}")
print(f"Lower Bound = {lower_bound}, Upper Bound = {upper_bound}")
```




```
Q1 = 1012.5, Q3 = 2850.0, IQR = 1837.5
Lower Bound = -1743.75, Upper Bound = 5606.25
```

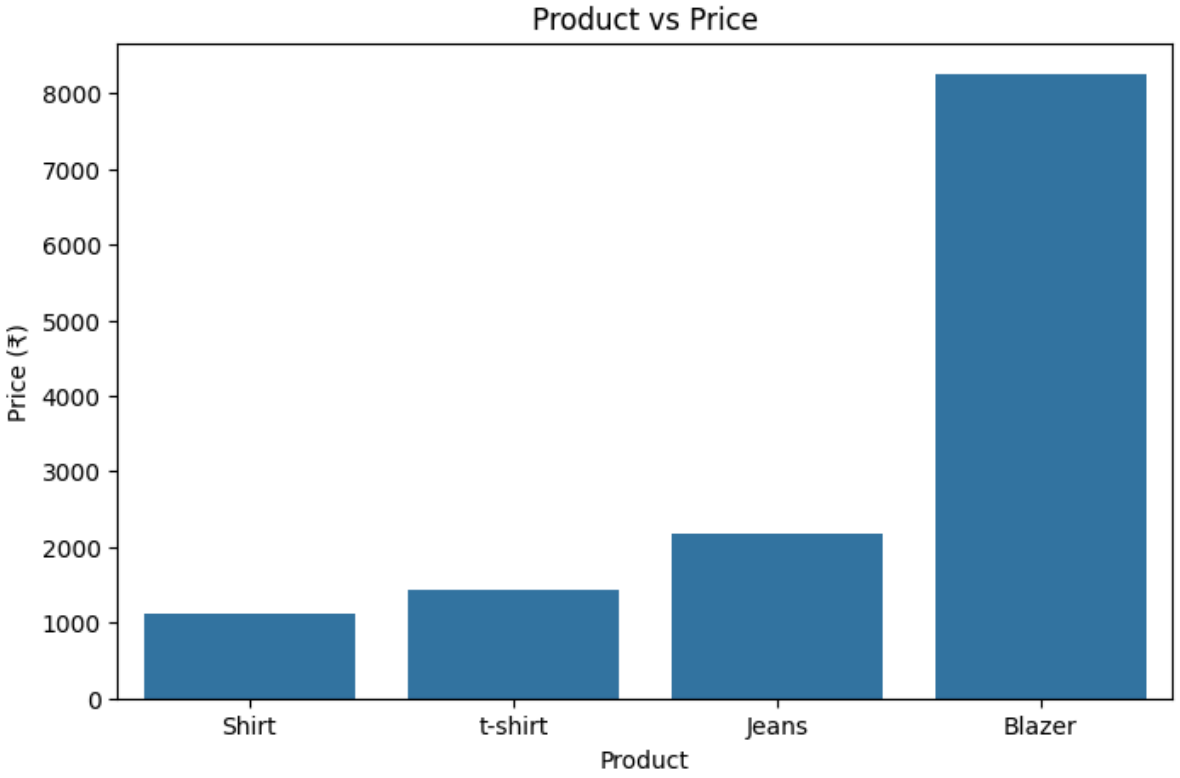
Step 8: Print Cleaned Data and Outliers

```
print("Cleaned Data:")
print(df)
print("\nOutliers (by Price):")
print(outliers if not outliers.empty else "No outliers detected.")
```

```
Cleaned Data:
   orderid  product category   price  quantity       city
0      101     Shirt   Casual  1200.0         2  Prayagraj
1      102   t-shirt   Casual  2850.0         1  Prayagraj
2      103     Jeans    Denim  1500.0        13      Delhi
3      104    Blazer   Formal  8000.0         1     Mumbai
4      105     Shirt   Formal  1200.0         2     Mumbai
5      106   t-shirt   Casual   700.0        13      Delhi
6      107     Jeans    Denim  2850.0         1     Jaipur
7      108     Shirt   Casual   950.0         2  Prayagraj
8      109    Blazer   Formal  8500.0         1     Mumbai
9      110   t-shirt   Casual   750.0         2      Delhi

Outliers (by Price):
   orderid product category   price  quantity    city
3      104  Blazer   Formal  8000.0         1  Mumbai
8      109  Blazer   Formal  8500.0         1  Mumbai
```

Step 9: Product vs Price - Seaborn Barchart

```
plt.figure(figsize=(8, 5))
sns.barplot(x="product", y="price", data=df, estimator="mean", errorbar=None)
plt.title("Product vs Price")
plt.xlabel("Product")
plt.ylabel("Price (₹)")
plt.show()
```



Question 4: Kaggle Dataset - Data Cleaning, Transformation & Standardization

Step 1: Load the Dataset

```
import pandas as pd
import numpy as np

df = pd.read_csv("BL-Student-Data.csv")
print("Dataset loaded successfully.")
```

```
Dataset loaded successfully.
```

Step 2: View Dataset - First 5 Rows

```
print(df.head())
```

```
   StudentID     Full_Name  Gender   Age  Math_Score  Reading_Score  \
0          1  Rahul Sharma    Male  18.0        72.0             72
1          2   Priya Singh  Female   NaN        69.0             90
2          3    Amit Kumar    male  20.0        90.0             95
3          4   Sneha Patel  Female  17.0        47.0             57
4          5    Ravi Verma    Male  19.0        76.0             78

   Writing_Score         DOB     City
0             74  15-03-2007    Delhi
1             88  22-07-2006   Mumbai
2             93  10-01-2005    Delhi
3             44  05-09-2008     Pune
4             75  18-11-2006  Chennai
```

Step 3: View Dataset - Last 5 Rows

```
print(df.tail())
```

```
    StudentID    Full_Name  Gender    Age  Math_Score  Reading_Score  \
10         11    Meera Das  Female   -5.0        81.0             82
11         12  Karan Mehta    Male   19.0         NaN             65
12         13   Divya Iyer  Female  200.0        55.0             60
13         14    Arjun Rao    MALE   18.0        92.0             98
14         15  Neha Kapoor  Female   17.0        68.0             75

    Writing_Score         DOB       City
10             80  14-12-2006    Kolkata
11             70  03-10-2006       Pune
12             58  27-09-2007    Chennai
13             97  20-05-2007  Hyderabad
14             72  16-08-2008      Dilli
```

Step 4: Get Basic Information

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   StudentID      15 non-null     int64
 1   Full_Name      15 non-null     object
 2   Gender         15 non-null     object
 3   Age            13 non-null     float64
 4   Math_Score     14 non-null     float64
 5   Reading_Score  15 non-null     int64
 6   Writing_Score  15 non-null     int64
 7   DOB            15 non-null     object
 8   City           15 non-null     object
dtypes: float64(2), int64(3), object(4)
memory usage: 1.2+ KB
```

Step 5: Get Summary of Data

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

```
       StudentID         Age  Math_Score  Reading_Score  Writing_Score
count  15.000000   13.000000   14.000000      15.000000      15.000000
mean    8.000000   30.461538   70.428571      75.933333      73.466667
std     4.472136   51.351104   15.330785      15.210272      16.560998
min     1.000000   -5.000000   40.000000      45.000000      40.000000
25%     4.500000   18.000000   65.750000      68.500000      68.500000
50%     8.000000   18.000000   71.500000      75.000000      74.000000
75%    11.500000   19.000000   79.750000      86.500000      84.000000
max    15.000000  200.000000   92.000000      98.000000      97.000000
```

Step 6: Check Missing Values

```
print("Missing Values Count:")
print(df.isnull().sum())
```

```
Missing Values Count:
StudentID        0
Full_Name        0
Gender           0
Age              2
Math_Score       1
Reading_Score    0
Writing_Score    0
DOB              0
City             0
dtype: int64
```

Step 7: Fill Missing Values

```
df["Age"] = df["Age"].fillna(df["Age"].mean())
df["Math_Score"] = df["Math_Score"].fillna(df["Math_Score"].mean())
print("After filling missing values:")
print(df.isnull().sum())
```

```
After filling missing values:
StudentID        0
Full_Name        0
Gender           0
Age              0
Math_Score       0
Reading_Score    0
Writing_Score    0
DOB              0
City             0
dtype: int64
```

Step 8: Find and Remove Duplicates

```
print("Duplicate Rows:")
print(df[df.duplicated()])
df.drop_duplicates(inplace=True)
print("\nAfter removing duplicates, total rows:", len(df))
```

```
Duplicate Rows:
Empty DataFrame
Columns: [StudentID, Full_Name, Gender, Age, Math_Score, Reading_Score, Writing_Score, DOB, City]
Index: []

After removing duplicates, total rows: 15
```

Step 9: Convert Gender to Lower Case

```
df["Gender"] = df["Gender"].str.lower()
print(df["Gender"])
```

```
0       male
1     female
2       male
3     female
4       male
5     female
6       male
7     female
8       male
9       male
10    female
11      male
12    female
13      male
14    female
Name: Gender, dtype: object
```

Step 10: Replace Inconsistent Values

```
df["City"] = df["City"].replace({"Dilli": "Delhi"})
mean_age = df["Age"].mean()
df.loc[df["Age"] < 0, "Age"] = mean_age
df.loc[df["Age"] > 25, "Age"] = mean_age
print(df)
```

```
    StudentID     Full_Name  Gender        Age  Math_Score  Reading_Score  \
0           1  Rahul Sharma    male  18.000000   72.000000             72
1           2   Priya Singh  female  30.461538   69.000000             90
2           3    Amit Kumar    male  20.000000   90.000000             95
3           4   Sneha Patel  female  17.000000   47.000000             57
4           5    Ravi Verma    male  19.000000   76.000000             78
5           6   Anita Gupta  female  18.000000   71.000000             83
6           7  Vikram Joshi    male  30.461538   65.000000             72
7           8   Pooja Reddy  female  19.000000   88.000000             95
8           9   Suresh Nair    male  18.000000   40.000000             45
9          10  Rahul Sharma    male  18.000000   72.000000             72
10         11     Meera Das  female  30.461538   81.000000             82
11         12   Karan Mehta    male  19.000000   70.428571             65
12         13    Divya Iyer  female  30.461538   55.000000             60
13         14     Arjun Rao    male  18.000000   92.000000             98
14         15   Neha Kapoor  female  17.000000   68.000000             75

    Writing_Score         DOB       City
0              74  15-03-2007      Delhi
1              88  22-07-2006     Mumbai
2              93  10-01-2005      Delhi
3              44  05-09-2008       Pune
4              75  18-11-2006    Chennai
5              78  30-04-2007     Mumbai
6              67  12-08-2007      Delhi
7              92  25-06-2006  Hyderabad
8              40  08-02-2007    Chennai
9              74  15-03-2007      Delhi
10             80  14-12-2006    Kolkata
11             70  03-10-2006       Pune
12             58  27-09-2007    Chennai
13             97  20-05-2007  Hyderabad
14             72  16-08-2008      Delhi
```

Step 11: Convert DOB to DateTime Format

```
df["DOB"] = pd.to_datetime(df["DOB"], format="%d-%m-%Y")
print(df.dtypes)
```

```
StudentID                int64
Full_Name               object
Gender                  object
Age                    float64
Math_Score             float64
Reading_Score            int64
Writing_Score            int64
DOB             datetime64[ns]
City                    object
dtype: object
```

Step 12: Create a New Column (Total Score)

```
df["Total_Score"] = df["Math_Score"] + df["Reading_Score"] + df["Writing_Score"]
print(df[["Full_Name", "Total_Score"]])
```

```
       Full_Name  Total_Score
0   Rahul Sharma   218.000000
1    Priya Singh   247.000000
2     Amit Kumar   278.000000
3    Sneha Patel   148.000000
4     Ravi Verma   229.000000
5    Anita Gupta   232.000000
6   Vikram Joshi   204.000000
7    Pooja Reddy   275.000000
8    Suresh Nair   125.000000
9   Rahul Sharma   218.000000
10     Meera Das   243.000000
11   Karan Mehta   205.428571
12    Divya Iyer   173.000000
13     Arjun Rao   287.000000
14   Neha Kapoor   215.000000
```

Step 13: Filter Data Based on Condition

```
high_scorers = df[df["Math_Score"] > 80]
print("Students with Math Score > 80:")
print(high_scorers)
```

```
Students with Math Score > 80:
    StudentID    Full_Name  Gender        Age  Math_Score  Reading_Score  \
2           3   Amit Kumar    male  20.000000        90.0             95
7           8  Pooja Reddy  female  19.000000        88.0             95
10         11    Meera Das  female  30.461538        81.0             82
13         14    Arjun Rao    male  18.000000        92.0             98

    Writing_Score        DOB       City  Total_Score
2              93 2005-01-10      Delhi        278.0
7              92 2006-06-25  Hyderabad        275.0
10             80 2006-12-14    Kolkata        243.0
13             97 2007-05-20  Hyderabad        287.0
```

Step 14: Select Specific Columns

```
selected = df[["Full_Name", "Math_Score", "City"]]
print(selected)
```

```
       Full_Name  Math_Score       City
0   Rahul Sharma   72.000000      Delhi
1    Priya Singh   69.000000     Mumbai
2     Amit Kumar   90.000000      Delhi
3    Sneha Patel   47.000000       Pune
4     Ravi Verma   76.000000    Chennai
5    Anita Gupta   71.000000     Mumbai
6   Vikram Joshi   65.000000      Delhi
7    Pooja Reddy   88.000000  Hyderabad
8    Suresh Nair   40.000000    Chennai
9   Rahul Sharma   72.000000      Delhi
10     Meera Das   81.000000    Kolkata
11   Karan Mehta   70.428571       Pune
12    Divya Iyer   55.000000    Chennai
13     Arjun Rao   92.000000  Hyderabad
14   Neha Kapoor   68.000000      Delhi
```

Step 15: Sort Values

```
sorted_df = df.sort_values(by="Math_Score", ascending=False)
print(sorted_df[["Full_Name", "Math_Score"]])
```

```
       Full_Name  Math_Score
13     Arjun Rao   92.000000
2     Amit Kumar   90.000000
7    Pooja Reddy   88.000000
10     Meera Das   81.000000
4     Ravi Verma   76.000000
9   Rahul Sharma   72.000000
0   Rahul Sharma   72.000000
5    Anita Gupta   71.000000
11   Karan Mehta   70.428571
1    Priya Singh   69.000000
14   Neha Kapoor   68.000000
6   Vikram Joshi   65.000000
12    Divya Iyer   55.000000
3    Sneha Patel   47.000000
8    Suresh Nair   40.000000
```

Step 16: Grouping and Aggregation

```
print("Average Math Score by Gender:")
print(df.groupby("Gender")["Math_Score"].mean())
```

```
Average Math Score by Gender:
Gender
female    68.428571
male      72.178571
```

Step 17: Change Datatype

```
df["Age"] = df["Age"].astype(int)
print(df.dtypes)
```

```
StudentID                int64
Full_Name               object
Gender                  object
Age                      int64
Math_Score             float64
Reading_Score            int64
Writing_Score            int64
DOB             datetime64[ns]
City                    object
Total_Score            float64
dtype: object
```

Step 18: Rename Column

```
df.rename(columns={"Full_Name": "Student_Name"}, inplace=True)
print(df.columns.tolist())
```

```
['StudentID', 'Student_Name', 'Gender', 'Age', 'Math_Score', 'Reading_Score', 'Writing_Score', 'DOB', 'City', 'Total_Score']
```

Step 19: Split Column into Two

```
df[["First_Name", "Last_Name"]] = df["Student_Name"].str.split(" ", expand=True)
print(df[["First_Name", "Last_Name"]])
```

```
   First_Name Last_Name
0       Rahul    Sharma
1       Priya     Singh
2        Amit     Kumar
3       Sneha     Patel
4        Ravi     Verma
5       Anita     Gupta
6      Vikram     Joshi
7       Pooja     Reddy
8      Suresh      Nair
9       Rahul    Sharma
10      Meera       Das
11      Karan     Mehta
12      Divya      Iyer
13      Arjun       Rao
14       Neha    Kapoor
```

Step 20: Save Modified Dataset to CSV

```
df.to_csv("Cleaned_Student_Data.csv", index=False)
print("Dataset saved as Cleaned_Student_Data.csv")
```

```
Dataset saved as Cleaned_Student_Data.csv
```

Question 5: House Price Prediction - Linear Regression

Step 1 & 2: Create Database, Table and Insert Data

```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sqlalchemy import create_engine
from sklearn.linear_model import LinearRegression

conn = mysql.connector.connect(host="localhost", user="root", password="root")
cursor = conn.cursor()
cursor.execute("CREATE DATABASE IF NOT EXISTS House")
cursor.execute("USE House")
cursor.execute("DROP TABLE IF EXISTS HouseData")
cursor.execute("""
    CREATE TABLE HouseData (
        HouseID INT,
        Area INT,
        Bedrooms INT,
        Age INT,
        Price INT
    )
""")
data = [
    (1, 800,  2, 10, 40),
    (2, 1000, 3, 8,  50),
    (3, 1200, 3, 5,  65),
    (4, 1500, 4, 4,  80),
    (5, 1800, 4, 2,  95),
]
for row in data:
    cursor.execute("INSERT INTO HouseData VALUES (%s, %s, %s, %s, %s)", row)
```

```
cursor.close()
conn.close()
print("Database and table created successfully.")
```

```
Database and table created successfully.
```

Step 3: Import Database and Display DataFrame

```
engine = create_engine("mysql+mysqlconnector://root:root@localhost/House")
df = pd.read_sql("SELECT * FROM HouseData", engine)
print("DataFrame:")
print(df)
```

```
DataFrame:
   HouseID  Area  Bedrooms  Age  Price
0        1   800         2   10     40
1        2  1000         3    8     50
2        3  1200         3    5     65
3        4  1500         4    4     80
4        5  1800         4    2     95
```

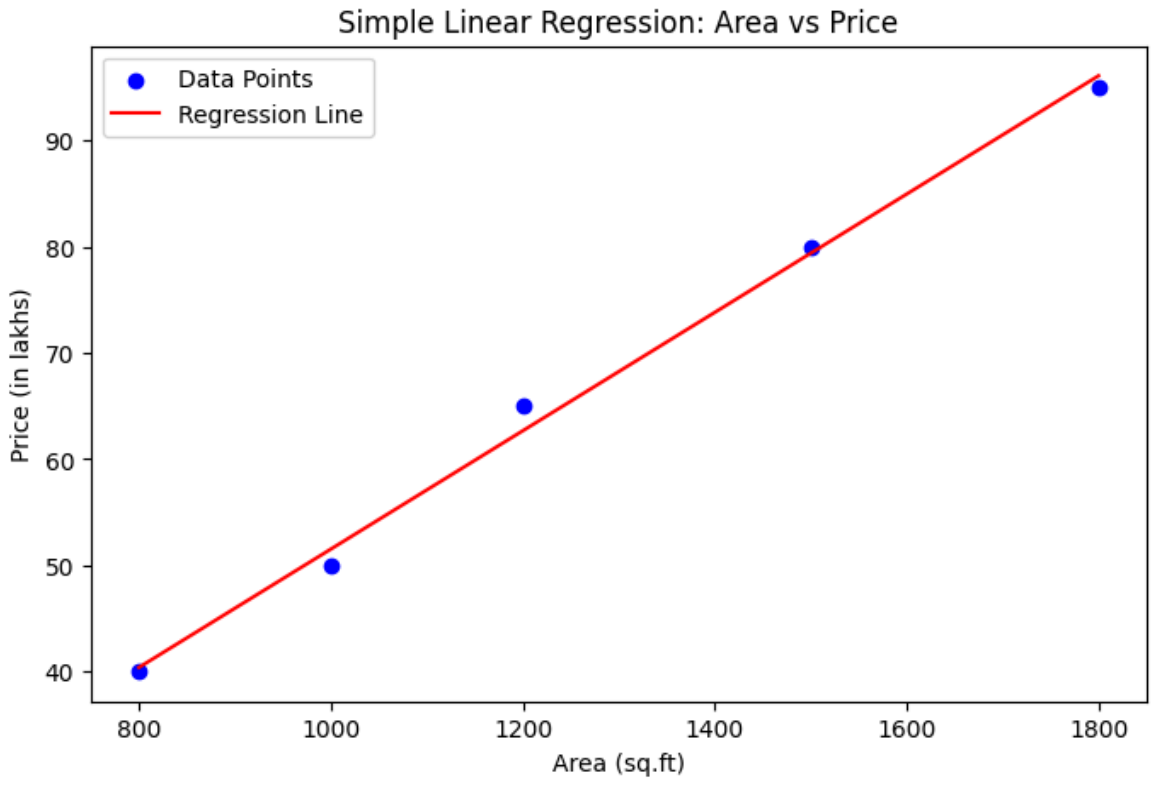
Step 4: Simple Linear Regression - Area vs Price

```
X = df[["Area"]]
Y = df["Price"]

model = LinearRegression()
model.fit(X, Y)

print(f"Coefficient (slope): {model.coef_[0]:.4f}")
print(f"Intercept: {model.intercept_:.4f}")
```

```
Coefficient (slope): 0.0557
Intercept: -4.1772
```

Plot Regression Line with Data Points

```
plt.figure(figsize=(8, 5))
plt.scatter(df["Area"], df["Price"], color="blue", label="Data Points")
plt.plot(df["Area"], model.predict(X), color="red", label="Regression Line")
plt.title("Simple Linear Regression: Area vs Price")
plt.xlabel("Area (sq.ft)")
plt.ylabel("Price (in lakhs)")
plt.legend()
plt.show()
```



Predict Price when Area = 2000 sq.ft

```
predicted = model.predict(pd.DataFrame([[2000]], columns=["Area"]))
print(f"Predicted price for 2000 sq.ft: {predicted[0]:.2f} lakhs")
```

Step 5: Multiple Linear Regression - Area, Bedrooms, Age vs Price

```
X_multi = df[["Area", "Bedrooms", "Age"]]
Y = df["Price"]

model_multi = LinearRegression()
model_multi.fit(X_multi, Y)

print(f"Coefficients: Area={model_multi.coef_[0]:.4f}, Bedrooms={model_multi.coef_[1]:.4f}, Age={model_multi.coef_[2]:.4f}")
print(f"Intercept: {model_multi.intercept_:.4f}")
```

```
Coefficients: Area=0.0430, Bedrooms=-0.7000, Age=-1.8000
Intercept: 24.5000
```

Predict Price for Area=1400, Bedrooms=3, Age=3

```
predicted_multi = model_multi.predict(pd.DataFrame([[1400, 3, 3]], columns=["Area", "Bedrooms", "Age"]))
print(f"Predicted price for (1400 sqft, 3 bedrooms, 3 years): {predicted_multi[0]:.2f} lakhs")
```

```
Predicted price for (1400 sqft, 3 bedrooms, 3 years): 77.20 lakhs
```