

Practical 1 :

Write a Python program that checks if the entered username and password match predefined values.

```
print("Name : Aryan Rana \n Roll No : 1323223")
USERNAME = "admin"
PASSWORD = "12345"
entered_username = input("Enter username: ")
entered_password = input("Enter password: ")
if entered_username.lower() == USERNAME and entered_password.lower() == PASSWORD:
    print(" Login successful! Welcome,", entered_username)
else:
    print(" Invalid username or password. Please try again.")
```

OUTPUT

```
● (base) PS D:\AI> python -u "d:\AI\p1.py"
Name : Aryan Rana
Roll No : 1323223
Enter username: admin
Enter password: 12345
Login successful! Welcome, admin
❖ (base) PS D:\AI>
```

PRACTICAL 2 :

Write a program to print the given inverted right-angled triangle star pattern.

```
*****
 ****
  ***
   **
    *
print("Name : Aryan Rana \n Roll No : 1323223")
rows = 5 # Number of rows
for i in range(rows, 0, -1):
    print('*' * i)
```

OUTPUT

The screenshot shows a terminal window with the following interface elements at the top:

- PROBLEMS (66)
- OUTPUT
- DEBUG CONSOLE
- TERMINAL** (underlined)
- PORTS
- SO

The terminal window displays the following output:

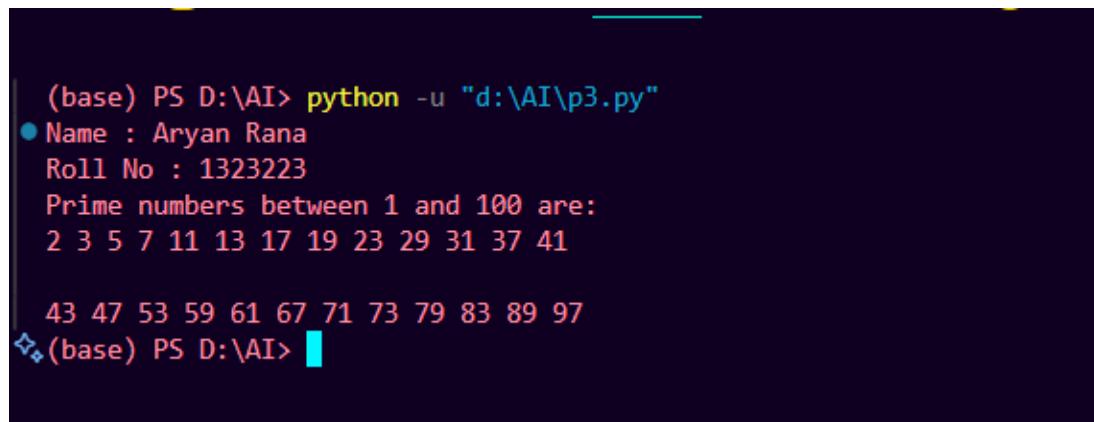
```
(base) PS D:\AI> python -u "d:\AI\p2.py"
● Name : Aryan Rana
Roll No : 1323223
*****
 ****
  ***
   **
    *
(base) PS D:\AI>
```

PRACTICAL 3 :

Write a program to print all prime numbers between 1 and 100.

```
print("Name : Aryan Rana \n Roll No : 1323223")
print("Prime numbers between 1 and 100 are:")
for num in range(2, 101):
    is_prime = True
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            is_prime = False
            break
    if is_prime:
        print(num, end=" ")
    if num == 41 :
        print("\n")
```

OUTPUT



```
(base) PS D:\AI> python -u "d:\AI\p3.py"
● Name : Aryan Rana
Roll No : 1323223
Prime numbers between 1 and 100 are:
2 3 5 7 11 13 17 19 23 29 31 37 41
43 47 53 59 61 67 71 73 79 83 89 97
◆ (base) PS D:\AI>
```

PRACTICAL 4 :

Convert a list of string numbers into integers and sort them in ascending order without using built-in sort functions.

```
print("Name : Aryan Rana \n Roll No : 1323223")
string_numbers = ["10", "3", "45", "2", "8"]
int_numbers = [int(num) for num in string_numbers]
n = len(int_numbers)
for i in range(n):
    for j in range(0, n - i - 1):
        if int_numbers[j] > int_numbers[j + 1]:
            int_numbers[j], int_numbers[j + 1] = int_numbers[j + 1], int_numbers[j]
print("Sorted numbers:", int_numbers)
```

OUTPUT

```
(base) PS D:\AI> python -u "d:\AI\p4.py"
Name : Aryan Rana
Roll No : 1323223
Sorted numbers: [2, 3, 8, 10, 45]
(base) PS D:\AI> 
```

PRACTICAL 5 :

Use a dictionary to count the frequency of elements in a list.

```
print("Name : Aryan Rana \n Roll No : 1323223")  
my_list = ["apple", "banana", "apple", "orange", "banana", "apple"]  
frequency = {}  
for item in my_list:  
    if item in frequency:  
        frequency[item] += 1  
    else:  
        frequency[item] = 1  
print("Frequency of elements:")  
for key, value in frequency.items():  
    print(f"{key}: {value}")
```

OUTPUT

```
(base) PS D:\AI> python -u "d:\AI\p5.py"  
● Name : Aryan Rana  
Roll No : 1323223  
Frequency of elements:  
apple: 3  
banana: 2  
orange: 1  
◆ (base) PS D:\AI>
```

PRACTICAL 6 :

Two sample lists

```
print("Name : Aryan Rana \n Roll No : 1323223")
list1 = [1, 2, 3, 4, 5, 5, 2]
list2 = [4, 5, 6, 7, 5, 4]
common_elements = list(set(list1) & set(list2))
print("Unique common elements:", common_elements)
```

OUTPUT

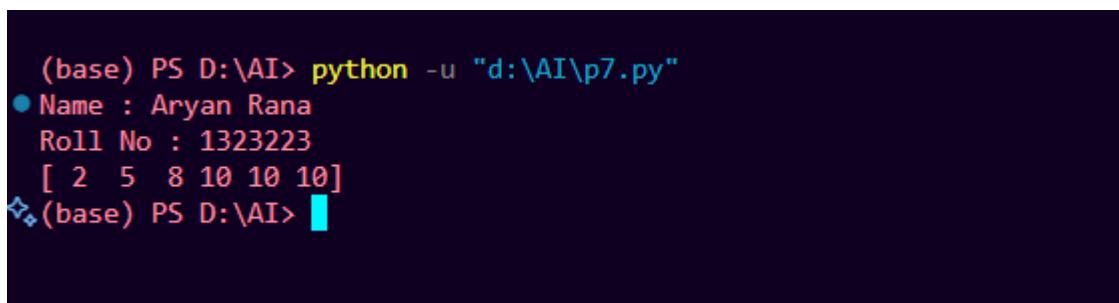
```
(base) PS D:\AI> python -u "d:\AI\p6.py"
Name : Aryan Rana
Roll No : 1323223
Unique common elements: [4, 5]
(base) PS D:\AI> 
```

PRACTICAL 7 :

Replace all elements in a NumPy array greater than a threshold with that threshold

```
import numpy as np  
print("Name : Aryan Rana \n Roll No : 1323223")  
arr = np.array([2, 5, 8, 10, 12, 15])  
threshold = 10  
arr = np.where(arr > threshold, threshold, arr)  
print(arr)
```

OUTPUT



```
(base) PS D:\AI> python -u "d:\AI\p7.py"  
● Name : Aryan Rana  
Roll No : 1323223  
[ 2  5  8 10 10 10]  
◆ (base) PS D:\AI>
```

A screenshot of a terminal window with a black background and white text. It shows the command 'python -u "d:\AI\p7.py"' being run. The output displays the name 'Aryan Rana' and roll number '1323223'. Below this, a list of integers '[2 5 8 10 10 10]' is shown. The prompt '(base) PS D:\AI>' appears at the bottom.

PRACTICAL 8 :

Replace missing values in a Pandas DataFrame with the mean of their columns.

```
import pandas as pd  
import numpy as np  
print("Name : Aryan Rana \n Roll No : 1323223")  
data = {  
    'A': [1, 2, np.nan, 4],  
    'B': [5, np.nan, 7, 8],  
    'C': [np.nan, 11, 12, 13]  
}  
df = pd.DataFrame(data)  
print("Before filling missing values:\n", df)  
df.fillna(df.mean(numeric_only=True), inplace=True)  
print("\nAfter filling missing values:\n", df)
```

OUTPUT

```
(base) PS D:\AI> python -u "d:\AI\p8.py"  
Name : Aryan Rana  
Roll No : 1323223  
Before filling missing values:  
     A      B      C  
0   1.0    5.0    NaN  
1   2.0    NaN    11.0  
2   NaN    7.0    12.0  
3   4.0    8.0    13.0  
  
After filling missing values:  
     A          B          C  
0  1.000000  5.000000  12.0  
1  2.000000  6.666667  11.0  
2  2.333333  7.000000  12.0  
3  4.000000  8.000000  13.0  
*(base) PS D:\AI>
```

PRACTICAL 9 :

Load the Myntra dataset, perform **info**, **shape**, **describe**, and check for null values and calculate Total revenue and discount.

```
import pandas as pd
print("Name : Aryan Rana \n Roll No : 1323223")
df = pd.read_csv("myntra.csv")
print(df.head(1), "\n")
print("==== Dataset Info ====")
print(df.info(), "\n")
print("==== Shape (Rows, Columns) ====")
print(df.shape, "\n")
print("==== Statistical Summary ====")
print(df.describe(), "\n")
print("==== Null Values ====")
print(df.isnull().sum(), "\n")
df["Revenue"] = df["discounted_price"]
df["Discount"] = df["marked_price"] - df["discounted_price"]
total_revenue = df["Revenue"].sum()
total_discount = df["Discount"].sum()
print("==== Totals ====")
print(f" Total Revenue: ₹{total_revenue:.2f}")
print(f" Total Discount Given: ₹{total_discount:.2f}")
```

OUTPUT

```
(base) PS D:\AI> python -u "d:\AI\p9.py"
▶ Name : Aryan Rana
▶ Roll No : 1323223
    product_name ...           product_link
0 Croc Textured Two Fold Wallet ... wallets/lino-perros/lino-perros-women-peach-co...
[1 rows x 7 columns]

== Dataset Info ==
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 168029 entries, 0 to 168028
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   product_name     168029 non-null   object 
 1   brand_name       168029 non-null   object 
 2   rating           168029 non-null   float64
 3   rating_count     168029 non-null   int64  
 4   marked_price     168029 non-null   int64  
 5   discounted_price 168029 non-null   int64  
 6   product_link     168029 non-null   object 
dtypes: float64(1), int64(3), object(3)
memory usage: 9.0+ MB
None

== Shape (Rows, Columns) ==
(168029, 7)

== Statistical Summary ==
      rating  rating_count  marked_price  discounted_price
count  168029.000000  168029.000000  168029.000000  168029.000000
mean    2.264556    114.668658   2509.07457   1515.168757
std     2.101402    800.207065   2402.10918   1800.460291
min     0.000000    0.000000    50.00000    49.000000
25%    0.000000    0.000000   1295.00000   674.000000
50%    3.500000    5.000000   1998.00000   999.000000
75%    4.300000   31.000000   2999.00000  1739.000000
max    5.000000   55900.000000  113999.00000  45900.000000

== Null Values ==
product_name      0
brand_name        0
rating            0
rating_count      0
marked_price      0
discounted_price  0
product_link      0
dtype: int64

== Totals ==
🟡 Total Revenue: ₹254,592,291.00
🟠 Total Discount Given: ₹167,005,000.00
(base) PS D:\AI>
```

PRACTICAL 10 :

From the Myntra dataset, filter products priced above 2000 and belonging to the category “tshirts”.

```
import pandas as pd

df = pd.read_csv("myntra.csv")

df['category'] = df['product_link'].apply(lambda x: x.split('/')[0])

filtered_df = df[(df['discounted_price'] > 2000) & (df['category'] == 'tshirts')]

print(filtered_df)
```

OUTPUT

| | product_name | brand_name | rating | ... | discounted_price | product_link | category |
|------------------------|---------------------------------|------------------|--------|-----|------------------|---|----------|
| 80 | Men Solid T-shirt | MANGO MAN | 0.0 | ... | 2390 | tshirts/mango-man/mango-man-blue-solid-t-shirt... | tshirts |
| 486 | Men Printed Polo Collar T-shirt | Louis Philippe | 0.0 | ... | 2499 | tshirts/louis-philippe/louis-philippe-men-gree... | tshirts |
| 936 | Men Abstract Print Pure Cotton | Levis | 4.1 | ... | 2209 | tshirts/levis/levis-men-white-beige-abtract... | tshirts |
| 1085 | Slim Fit HEAT.ROY T-shirt | ADIDAS | 0.0 | ... | 3059 | tshirts/adidas/adidas-men-white-brand-logo-pri... | tshirts |
| 1199 | Men Striped Polo Collar T-shirt | Van Heusen | 0.0 | ... | 2124 | tshirts/van-heusen/van-heusen-men-navy-blue-st... | tshirts |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 166559 | Men Striped Polo Collar T-shirt | ORIGIN BY ZALORA | 0.0 | ... | 2799 | tshirts/origin-by-zalora/origin-by-zalora-men-... | tshirts |
| 166590 | Men Striped Polo Collar T-shirt | ORIGIN BY ZALORA | 0.0 | ... | 2869 | tshirts/origin-by-zalora/origin-by-zalora-men-... | tshirts |
| 167278 | Men Polo Collar T-shirt | ORIGIN BY ZALORA | 0.0 | ... | 2729 | tshirts/origin-by-zalora/origin-by-zalora-men-... | tshirts |
| 167312 | Men Mandarin Collar T-shirt | ZALORA ACTIVE | 0.0 | ... | 2174 | tshirts/zalora-active/zalora-active-men-black-... | tshirts |
| 167779 | Men Striped Polo Collar T-shirt | ORIGIN BY ZALORA | 0.0 | ... | 2799 | tshirts/origin-by-zalora/origin-by-zalora-men-... | tshirts |
| [774 rows x 8 columns] | | | | | | | |

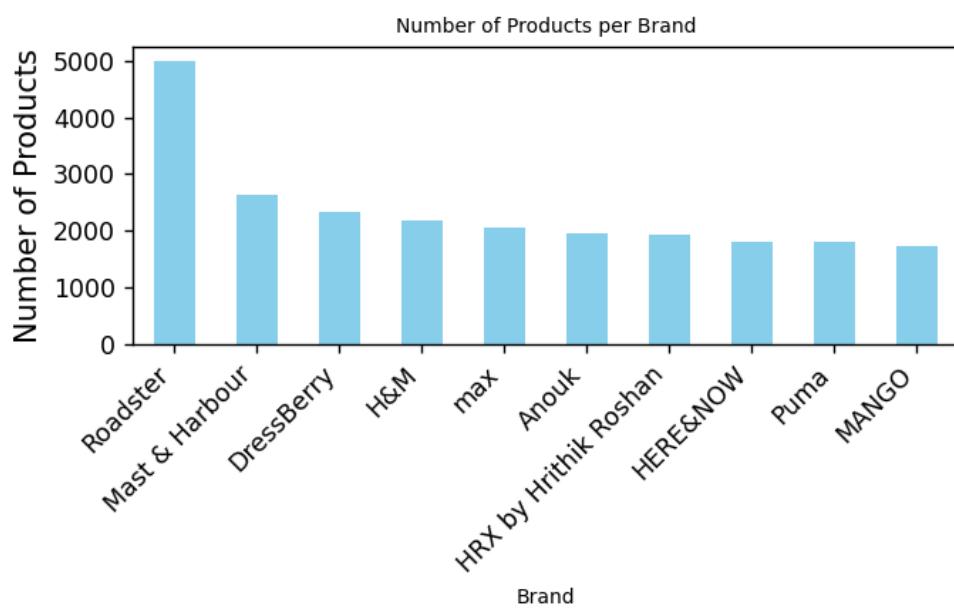
PRACTICAL 11 :

Plot a bar chart showing the number of products per brand

```
import pandas as pd  
import matplotlib.pyplot as plt  
  
df = pd.read_csv("myntra.csv")  
  
brand_counts = df['brand_name'].value_counts()  
  
plt.figure(figsize=(12, 6))  
  
brand_counts.plot(kind='bar', color='skyblue')  
  
plt.title('Number of Products per Brand', fontsize=16)  
  
plt.xlabel('Brand', fontsize=12)  
  
plt.ylabel('Number of Products', fontsize=12)  
  
plt.xticks(rotation=45, ha='right')  
  
plt.tight_layout()  
  
plt.show()
```

OUTPUT

```
(base) PS D:\AI> python -u "d:\AI\p11.py"  
Name : Aryan Rana  
Roll No : 1323223  
>(base) PS D:\AI>
```



PRACTICAL 12 :

Find the average price of products grouped by category.

```
import pandas as pd  
  
df = pd.read_csv("myntra.csv")  
  
df['category'] = df['product_link'].apply(lambda x: x.split('/')[0])  
  
avg_price = df.groupby('category')['discounted_price'].mean()  
  
print("Average price of products per category:\n")  
  
print(avg_price)
```

OUTPUT

```
(base) PS D:\VAI> python -u "d:\AI\p12.py"  
● Name : Aryan Rana  
Roll No : 1323223  
Average price of products per category:  
  
category  
accessory-gift-set      1830.058824  
action-figures-and-play-set 1463.000000  
activity-toys-and-games    879.000000  
air-fryer                  5632.500000  
anklet                      587.500000  
...  
watches                     5488.029851  
water-bottle                 844.540816  
windchimes                   565.333333  
wristbands                   384.000000  
yoga-mats                     1099.275862  
Name: discounted_price, Length: 341, dtype: float64  
❖(base) PS D:\VAI> [ ]
```

PRACTICAL 13:

Load the Housing dataset, clean it, visualize correlation, encode categorical values, split (70:30), train Linear Regression, evaluate with R² and MSE..

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
print("Name : Aryan Rana \nRoll No : 1323223")
housing = fetch_california_housing(as_frame=True)
df = housing.frame
print("Shape:", df.shape)
print(df.head())
print("\nMissing values:\n", df.isnull().sum())
df = df.drop_duplicates()
plt.figure(figsize=(10, 6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()
print("\nData types:\n", df.dtypes)
X = df.drop('MedHouseVal', axis=1)
y = df['MedHouseVal']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
```

```

y_pred = model.predict(X_test)

r2 = r2_score(y_test, y_pred)

mse = mean_squared_error(y_test, y_pred)

print("\nModel Performance:")

print(f"R2 Score: {r2:.4f}")

print(f"Mean Squared Error: {mse:.4f}")

plt.figure(figsize=(6, 6))

sns.scatterplot(x=y_test, y=y_pred)

plt.xlabel("Actual Values")

plt.ylabel("Predicted Values")

plt.title("Actual vs Predicted House Prices")

plt.show()

```

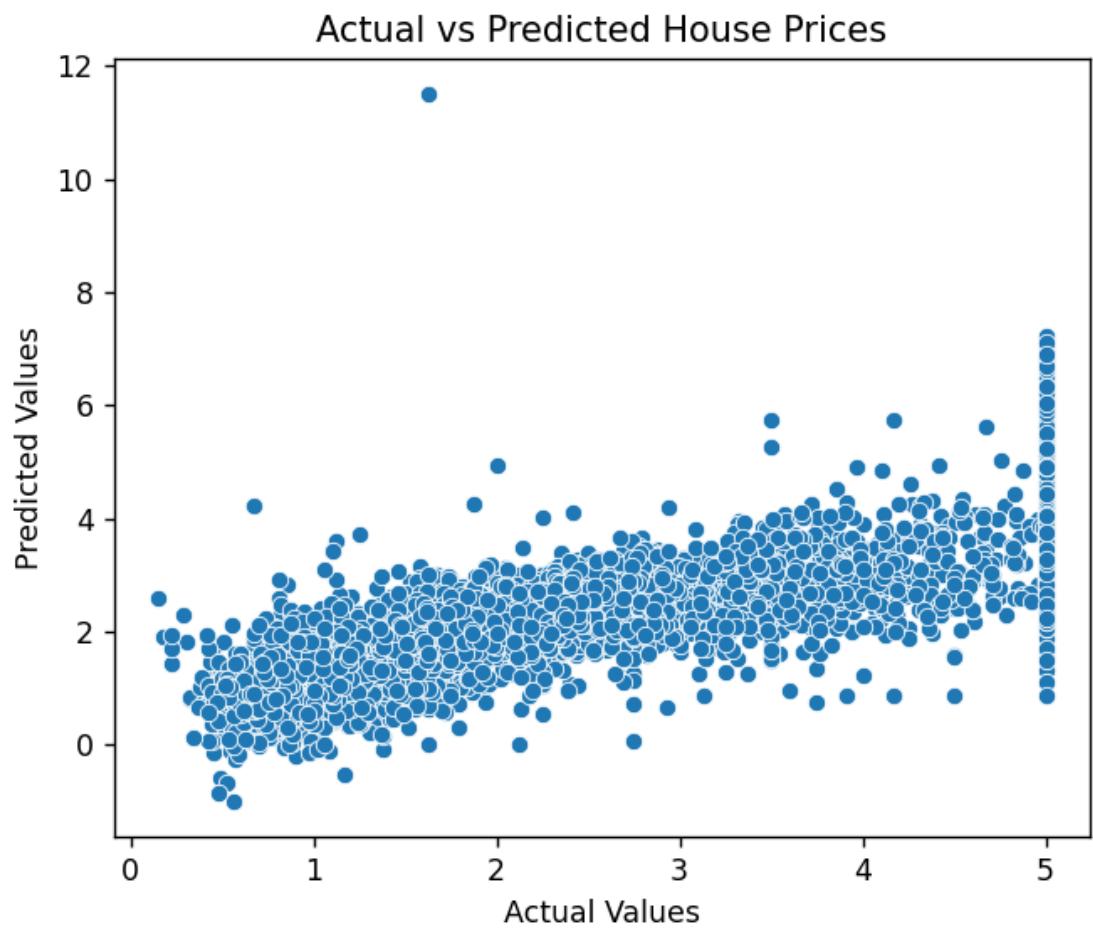
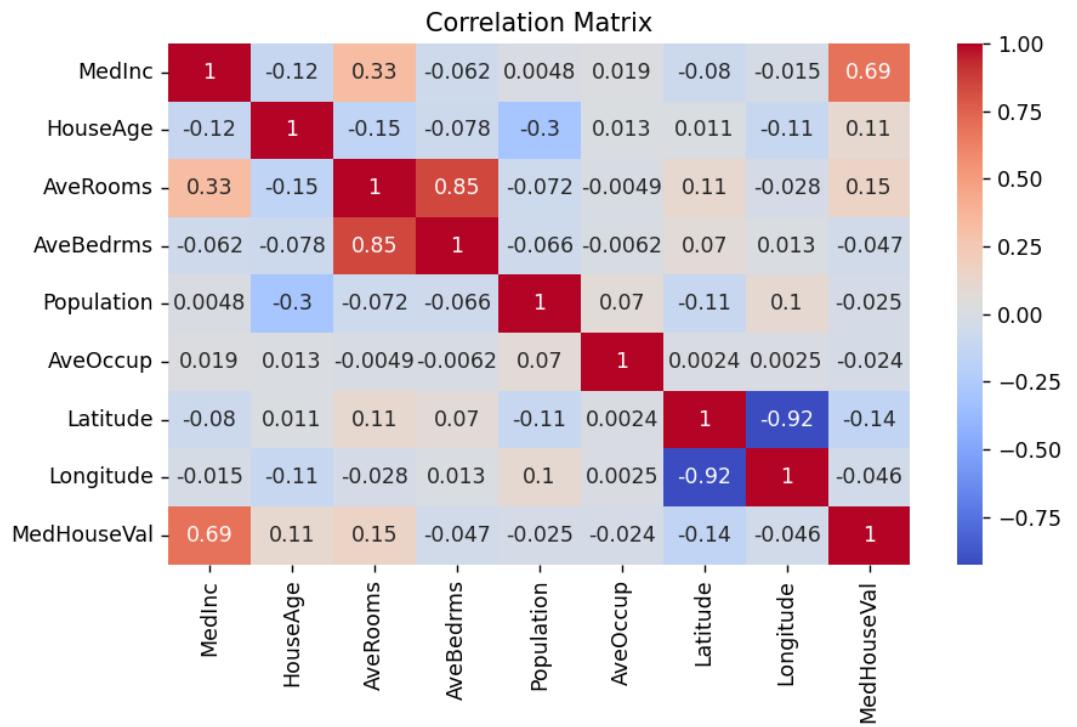
OUTPUT

```

(base) PS D:\AI> python -u "d:\AI\p13.py"
Name : Aryan Rana
Roll No : 1323223
Shape: (20640, 9)
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  Longitude  MedHouseVal
0    8.3252     41.0    6.984127   1.023810      322.0    2.555556     37.88    -122.23      4.526
1    8.3014     21.0    6.238137   0.971880     2401.0    2.109842     37.86    -122.22      3.585
2    7.2574     52.0    8.288136   1.073446      496.0    2.802260     37.85    -122.24      3.521
3    5.6431     52.0    5.817352   1.073059      558.0    2.547945     37.85    -122.25      3.413
4    3.8462     52.0    6.281853   1.081081      565.0    2.181467     37.85    -122.25      3.422

Missing values:
MedInc        0
HouseAge       0
AveRooms       0
AveBedrms      0
Population      0
AveOccup       0
Latitude        0
Longitude       0
MedHouseVal     0
dtype: int64

```



PRACTICAL 14:

Train a Decision Tree classifier on the Iris dataset and evaluate **accuracy**.

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
print("Name : Aryan Rana \nRoll No : 1323223")
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['target'] = iris.target
print("Dataset shape:", df.shape)
print(df.head())
X = df.drop('target', axis=1)
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"\n ✅ Model Accuracy: {accuracy:.4f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=iris.target_names))
cm = confusion_matrix(y_test, y_pred)
```

```

plt.figure(figsize=(5, 4))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=iris.target_names,
            yticklabels=iris.target_names)

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix - Decision Tree on Iris")

plt.show()

```

OUTPUT

```

(base) PS D:\AI> python -u "d:\AI\p14.py"
● Name : Aryan Rana
Roll No : 1323223
Dataset shape: (150, 5)
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  target
0              5.1           3.5            1.4            0.2      0
1              4.9           3.0            1.4            0.2      0
2              4.7           3.2            1.3            0.2      0
3              4.6           3.1            1.5            0.2      0
4              5.0           3.6            1.4            0.2      0

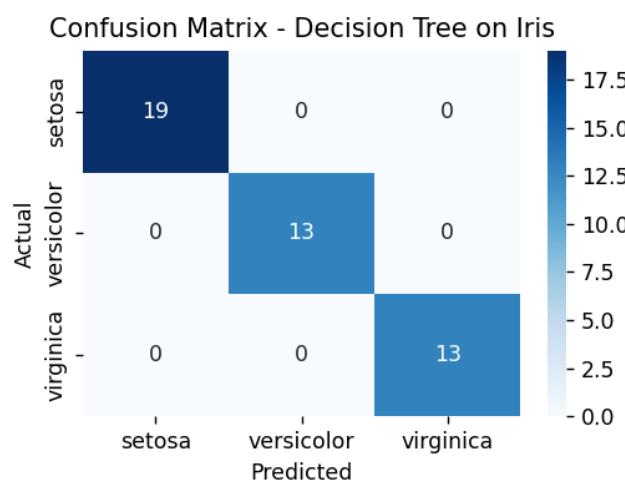
✓ Model Accuracy: 1.0000

Classification Report:
          precision    recall    f1-score   support
setosa       1.00     1.00     1.00      45
versicolor   1.00     1.00     1.00      45
virginica    1.00     1.00     1.00      45

accuracy         -         -         -      45
macro avg       1.00     1.00     1.00      45
weighted avg    1.00     1.00     1.00      45

❖ (base) PS D:\AI> []

```



PRACTICAL 15:

Train a **Random Forest** on Pima Indian Diabetes dataset and compare results with other algorithms using **LazyPredict**.

```
# 1 Import libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from lazypredict.Supervised import LazyClassifier

print("Name : Aryan Rana \nRoll No : 1323223")
df = pd.read_csv("diabetes.csv") # <-- ensure data.csv is in the same folder
print("Dataset shape:", df.shape)
print(df.head())
# Check for missing values
print("\nMissing values before cleaning:\n", df.isnull().sum())

# Replace 0 with NaN in certain medical features (invalid values)
cols_with_zero = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
df[cols_with_zero] = df[cols_with_zero].replace(0, np.nan)

# Fill missing with median values
df.fillna(df.median(), inplace=True)
print("\nMissing values after cleaning:\n", df.isnull().sum())
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
```

```
plt.title("Feature Correlation - Pima Indians Diabetes Dataset")
plt.suptitle("Aryan Rana", fontsize=12, fontweight='bold')
plt.show()

X = df.drop('Outcome', axis=1)
y = df['Outcome']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f"\n ✅ Random Forest Accuracy: {accuracy:.4f}")

print("\nClassification Report:\n", classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted')
plt.ylabel('Actual')

plt.suptitle("Aryan Rana", fontsize=12, fontweight='bold')
plt.title('Confusion Matrix - Random Forest')
plt.show()

clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metric=None)
models, predictions = clf.fit(X_train, X_test, y_train, y_test)

print("\n 📈 Model Comparison Results:\n")
print(models.head(10)) # Show top 10 models
```

OUTPUT

```
(base) PS D:\AI> python -u "d:\AI\p15.py"
Name : Aryan Rana
Roll No : 1323223
Dataset shape: (768, 9)
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI  DiabetesPedigreeFunction  Age  Outcome
0            6     148             72           35      0 33.60          0.63    50       1
1            1      85              66           29      0 26.60          0.35    31       0
2            8     183              64              0      0 23.30          0.67    32       1
3            1      89              66           23      94 28.10          0.17    21       0
4            0     137              40           35      168 43.10          2.29    33       1

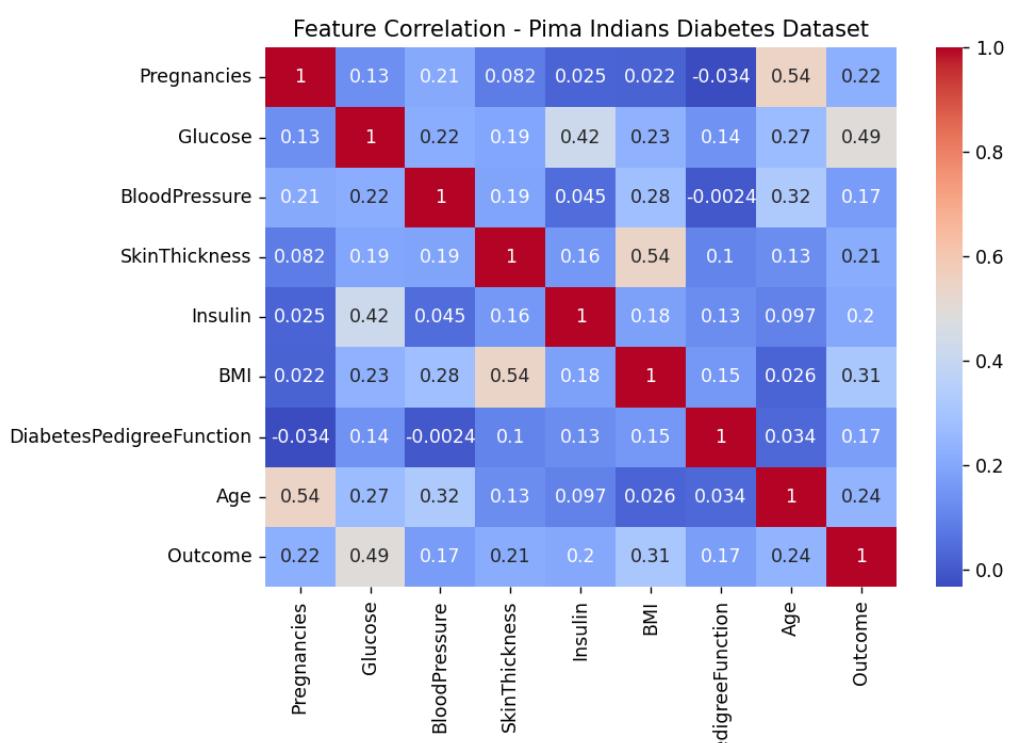
Missing values before cleaning:
Pregnancies      0
Glucose          0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
DiabetesPedigreeFunction  0
Age              0
Outcome          0
dtype: int64

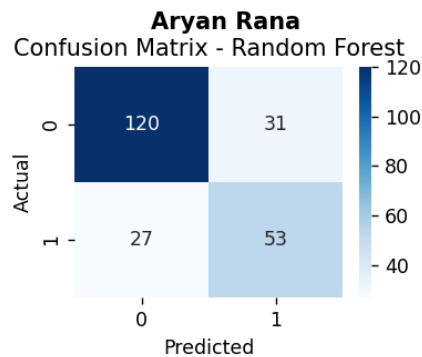
Missing values after cleaning:
Pregnancies      0
Glucose          0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
DiabetesPedigreeFunction  0
Age              0
Outcome          0
dtype: int64

 Random Forest Accuracy: 0.7489

Classification Report:
precision  recall  f1-score  support
0          0.82    0.79    0.81     151
1          0.63    0.66    0.65      80
```

Aryan Rana





```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
100% | 32/32 [00:08<00:00,  3.78it/s]
```

Model Comparison Results:

| Model | Accuracy | Balanced Accuracy | ROC AUC | F1 Score | Time Taken |
|-------------------------------|----------|-------------------|---------|----------|------------|
| RandomForestClassifier | 0.75 | 0.73 | 0.73 | 0.75 | 0.24 |
| AdaBoostClassifier | 0.76 | 0.73 | 0.73 | 0.76 | 0.19 |
| LGBMClassifier | 0.74 | 0.73 | 0.73 | 0.74 | 6.82 |
| BaggingClassifier | 0.76 | 0.73 | 0.73 | 0.76 | 0.09 |
| GaussianNB | 0.74 | 0.72 | 0.72 | 0.74 | 0.02 |
| XGBClassifier | 0.73 | 0.71 | 0.71 | 0.73 | 0.35 |
| QuadraticDiscriminantAnalysis | 0.75 | 0.71 | 0.71 | 0.75 | 0.02 |
| CalibratedClassifierCV | 0.74 | 0.71 | 0.71 | 0.74 | 0.08 |
| RidgeClassifier | 0.74 | 0.71 | 0.71 | 0.74 | 0.02 |
| SVC | 0.74 | 0.70 | 0.70 | 0.74 | 0.04 |

```
(base) PS D:\AI>
```