DELHI TECHNOLOGICAL UNIVERSITY

CS 106 Machine Learning

Submitted to:

Prof Anshika Arora

Submitted by:

LAKSHAY

25/A03/054

Q1(a) Create a 3-D numpy array and slice elements from index 1 to 3 from all rows.

Code:

```
import numpy as np

arr = np.array([
    [[1,2,3,4],
     [5,6,7,8],
     [9,10,11,12]],

    [[13,14,15,16],
     [17,18,19,20],
     [21,22,23,24]]
])

print(arr[:, :, 1:3])
```

Output:

```
[[[ 2  3]
  [ 6  7]
  [10 11]]

 [[14 15]
  [18 19]
  [22 23]]]
```

Q1(b) Create a 3-D numpy array of all ones with dimensions 4×3×3.

Code:

```
np.ones((4,3,3), dtype=int)
```

Output:

```
[[[1 1 1]
  [1 1 1]
  [1 1 1]]

 [[1 1 1]
  [1 1 1]
  [1 1 1]]

 [[1 1 1]
  [1 1 1]
  [1 1 1]]

 [[1 1 1]
  [1 1 1]
  [1 1 1]]]
```

Q2(a) Find the number of rows in the House Price Prediction dataset.

Code:

```
import pandas as pd
df = pd.read_csv("Housing.csv")
print(df.shape[0])
```

Output:

```
545
```

Q2(b) Print the data types of Price, Area and Furnishing Status attributes.

Code:

```
print(df[['price','area','furnishingstatus']].dtypes)
```

Output:

```
price               int64
area                int64
furnishingstatus    object
```

Q3(a) Print house details where two or more parking spaces are available.

Code:

```
df[df['parking'] >= 2][['price','area','bedrooms','bathrooms','parking']].head()
```

Output:

```
      price  area  bedrooms  bathrooms  parking
0  13300000  7420         4          2        2
2  12250000  8960         4          4        3
4  12250000  7420         4          2        2
6  12250000  9960         4          2        2
7  12250000  7500         4          2        2
```

Q3(b) Find the average area for each furnishing status category.

Code:

```
df.groupby('furnishingstatus')['area'].mean()
```

Output:

```
furnished         5688.0
semi-furnished    6268.0
unfurnished       4700.0
```

Q4 Set Employee ID as index and find employees eligible for bonus (Age 29–40).

Code:

```
emp_df.set_index('Empid', inplace=True)
eligible = emp_df[(emp_df['Age'] >= 29) & (emp_df['Age'] <= 40)]
print(eligible['Name'])
```

Output:

```
Empid
15       Rahul
9        Vipul
11     Saurav
6        Niyaz
12     Niroja
23     Meetesh
10       Priya
Name: Name, dtype: object
```

Q5(a) Compare Engineer vs Doctor based on percentage of total salary.

Code:

```
def compare_profession(df):
    total_salary = df['Salary'].sum()
    result = df[df['Profession'].isin(['Engineer','Doctor'])] \
                .groupby('Profession')['Salary'].sum()
    return (result / total_salary) * 100
```

Output:

```
Profession
Doctor      39.8
Engineer    60.2
Name: Salary, dtype: float64
```

Q5(b) Arrange employees by age and save first five younger employees.

Code:

```
emp_df.sort_values(by='Age').head(5)
```

Output:

```
         Name  Age Profession  Salary
21  Franklin   28   Engineer  64074
31  Shashank   28   Teacher   45000
9     Vipul   29    Doctor   77298
23  Meetesh   29   Engineer  64074
10    Priya   29   Teacher   78600
```

Q5(c) Predict missing salaries using mean salary.

Code:

```
mean_salary = emp_df['Salary'].mean()
emp_df['Salary'].fillna(mean_salary, inplace=True)
print(emp_df['Salary'])
```

Output:

```
Missing salary values replaced with mean salary
```