Digital Assignment 4

Name: Kamran Ansari

Reg No: 22MCA0223

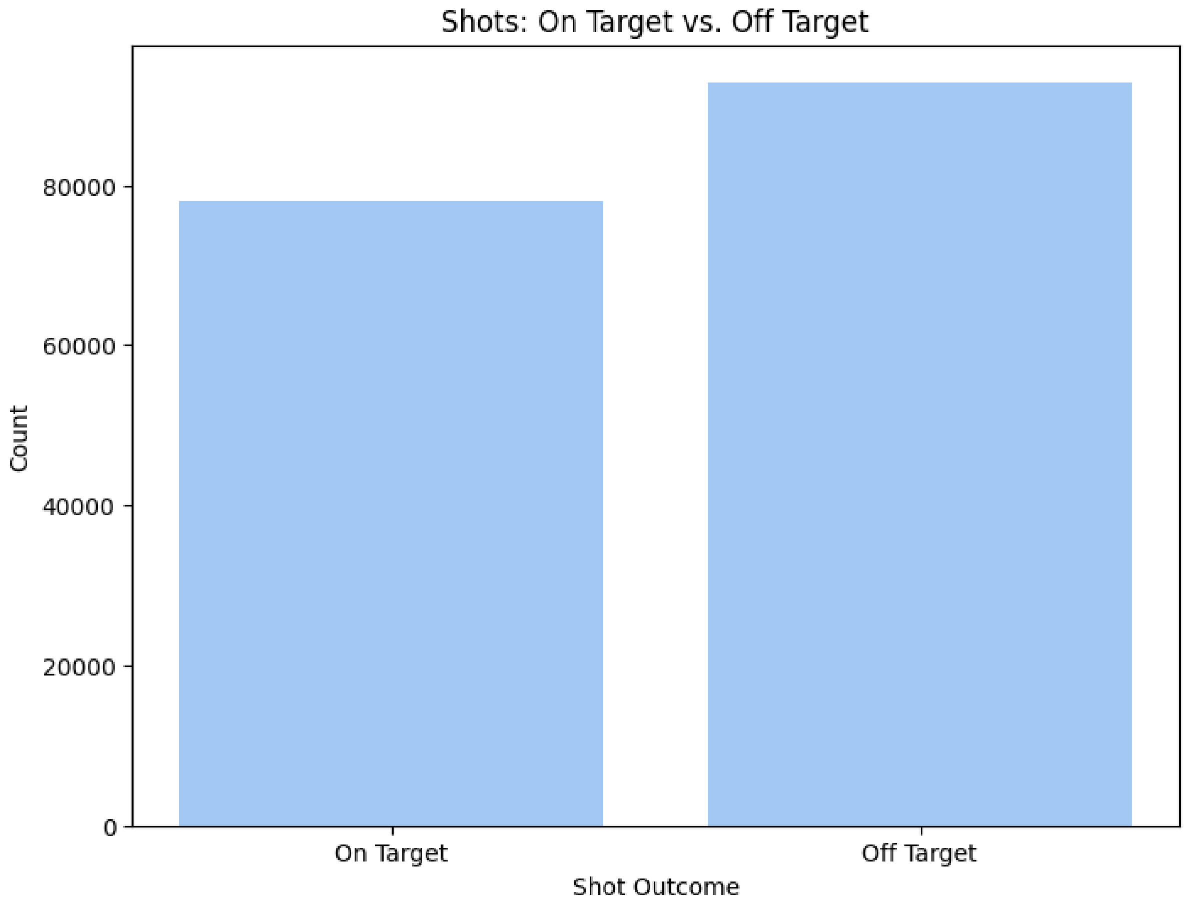
Question 1

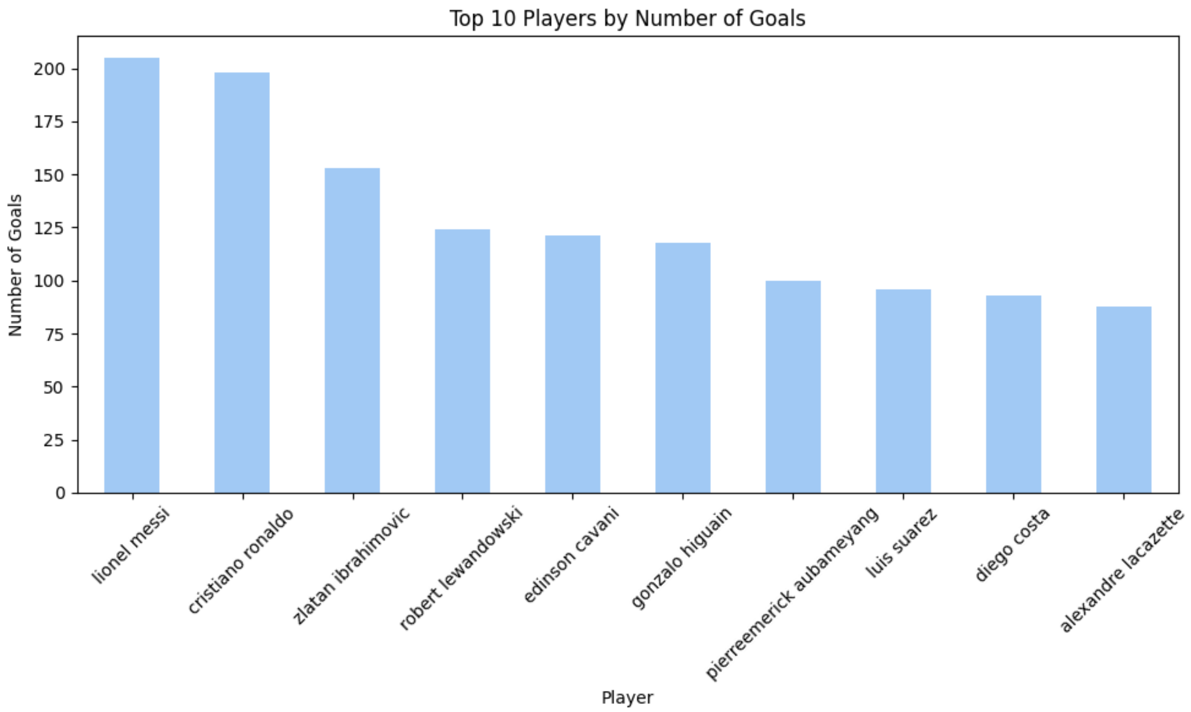
Write a python program for visualising the football statistical data by importing a large football dataset (50MB-1GB) from kaggle. Plot the different forms of graphs using Matplotlib and Seaborn libraries.

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

# Load the dataset
data = pd.read_csv('./archive/events.csv')


sns.set_palette("pastel")


# Filter data for shots
shots_data = data[data['event_type'] == 1]

# Count shots on target and off target
shots_on_target = shots_data[shots_data['shot_outcome'] ==
1]['id_event'].count()
shots_off_target = shots_data[shots_data['shot_outcome'] ==
2]['id_event'].count()

# Create bar graph
plt.figure(figsize=(8, 6))
plt.bar(['On Target', 'Off Target'], [shots_on_target,
shots_off_target])
plt.title('Shots: On Target vs. Off Target')
plt.xlabel('Shot Outcome')
plt.ylabel('Count')
plt.show()



# Extract the relevant attributes
player_goals = data[data['is_goal'] == 1]['player'].value_counts()

# Sort the players based on the number of goals
sorted_players = player_goals.sort_values(ascending=False)

# Select the top 10 players
top_10_players = sorted_players.head(10)

# Create the bar plot
plt.figure(figsize=(10, 6))
top_10_players.plot(kind='bar')
plt.title('Top 10 Players by Number of Goals')
plt.xlabel('Player')
plt.ylabel('Number of Goals')
plt.xticks(rotation=45)
plt.tight_layout()

# Display the plot
plt.show()



# Map bodypart codes to labels
bodypart_labels = {
    1: 'Right Foot',
    2: 'Left Foot',
    3: 'Head'
}

# Count shots by body part
body_parts_counts =
shots_data['bodypart'].map(bodypart_labels).value_counts()

# Create pie chart
plt.figure(figsize=(8, 6))
plt.pie(body_parts_counts, labels=body_parts_counts.index,
autopct='%1.1f%%')
plt.title('Body Parts Used for Shots')
plt.show()

# Mapping situation codes to labels
situation_labels = {
    0: 'Open Play',
    1: 'Set piece',
    2: 'Corner',
    3: 'Free kick'
}

# Create the countplot
sns.countplot(x='situation', data=data)
plt.title('Distribution of Situations')
plt.xlabel('Situation')
plt.ylabel('Count')

# Set x-axis tick labels using the situation labels dictionary
plt.xticks(list(situation_labels.keys()),
list(situation_labels.values()))
```

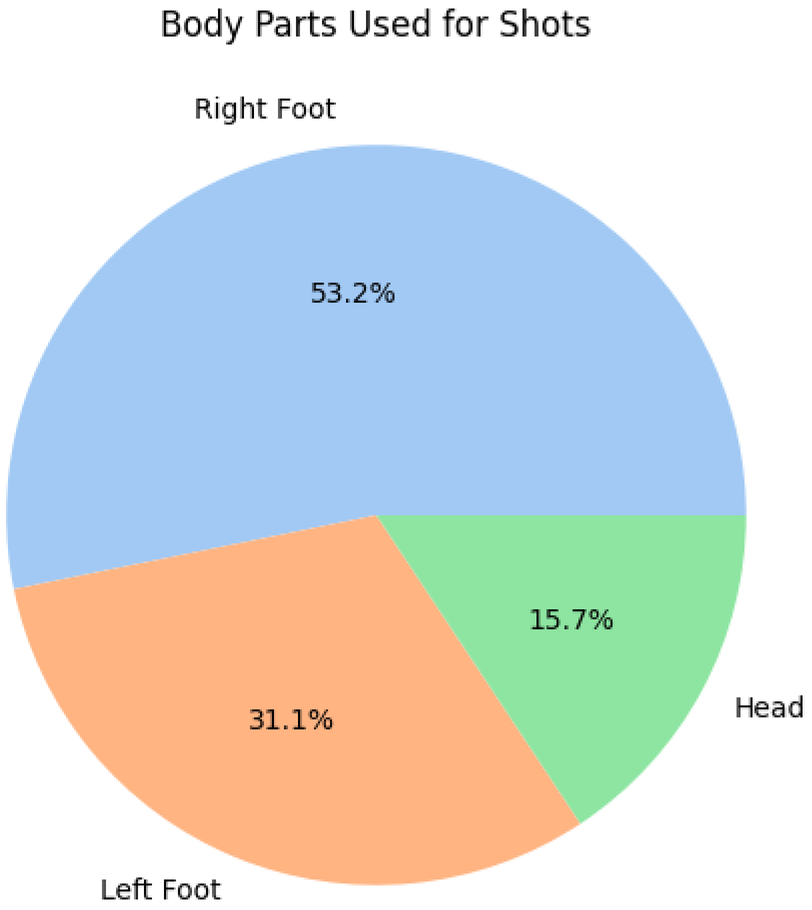
```python
plt.show()


# Select some attributes for correlation analysis
attributes = ['is_goal', 'shot_outcome', 'shot_place', 'fast_break']

# Create a correlation matrix
correlation_matrix = data[attributes].corr()

# Generate the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
linewidths=0.5)
plt.title('Correlation Heatmap of Some Attributes')
plt.show()


data2 = pd.read_csv('./archive/ginf.csv')



# Create the scatter plot
plt.scatter(data2['odd_h'], data2['odd_a'])

# Set labels for the x-axis and y-axis
plt.xlabel('Odds for Home Team')
plt.ylabel('Odds for Away Team')

# Set a title for the plot
plt.title('Scatter Plot: Odds for Home Team vs Odds for Away Team')
```

```
# Display the plot
plt.show()
```

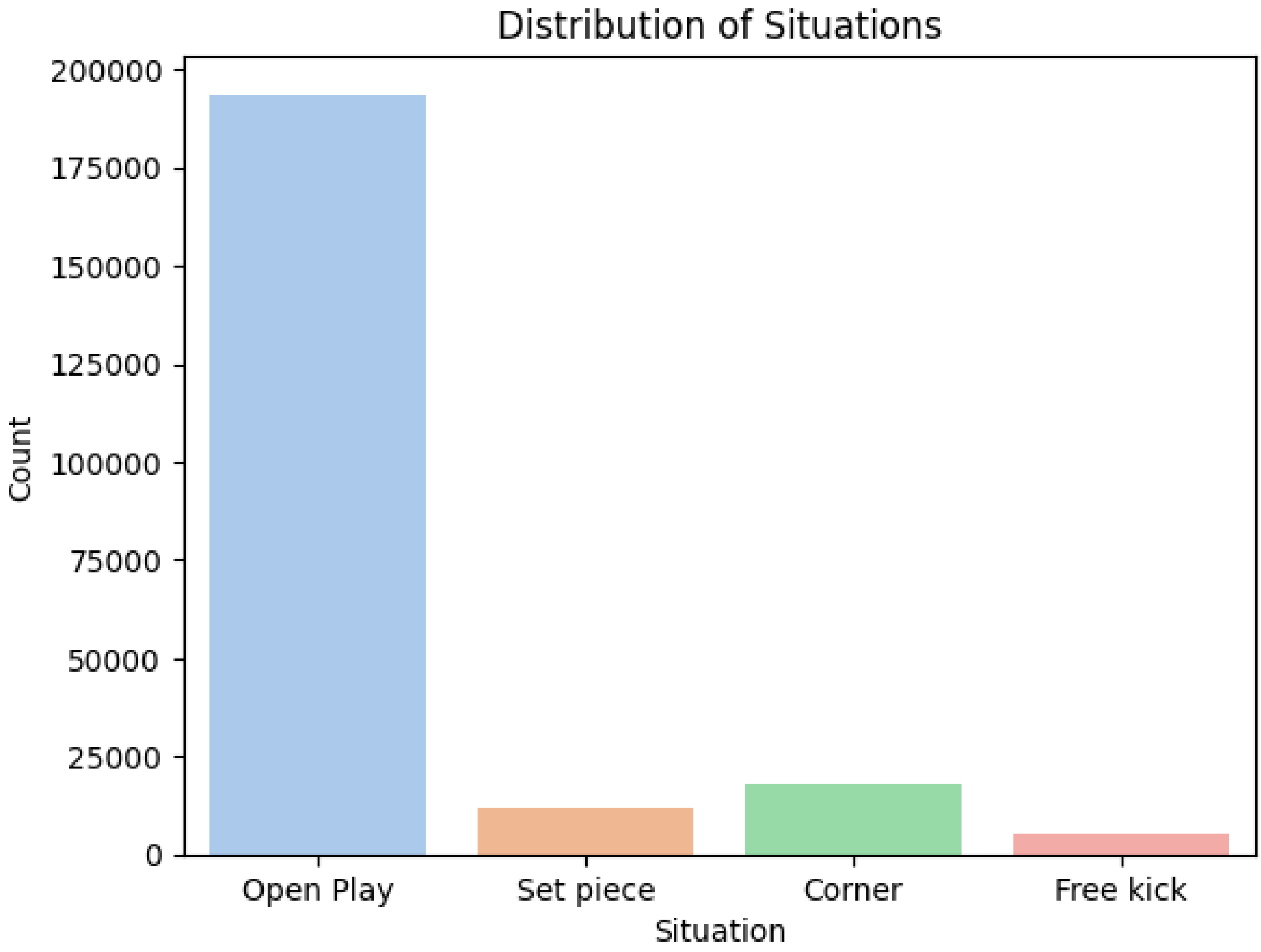


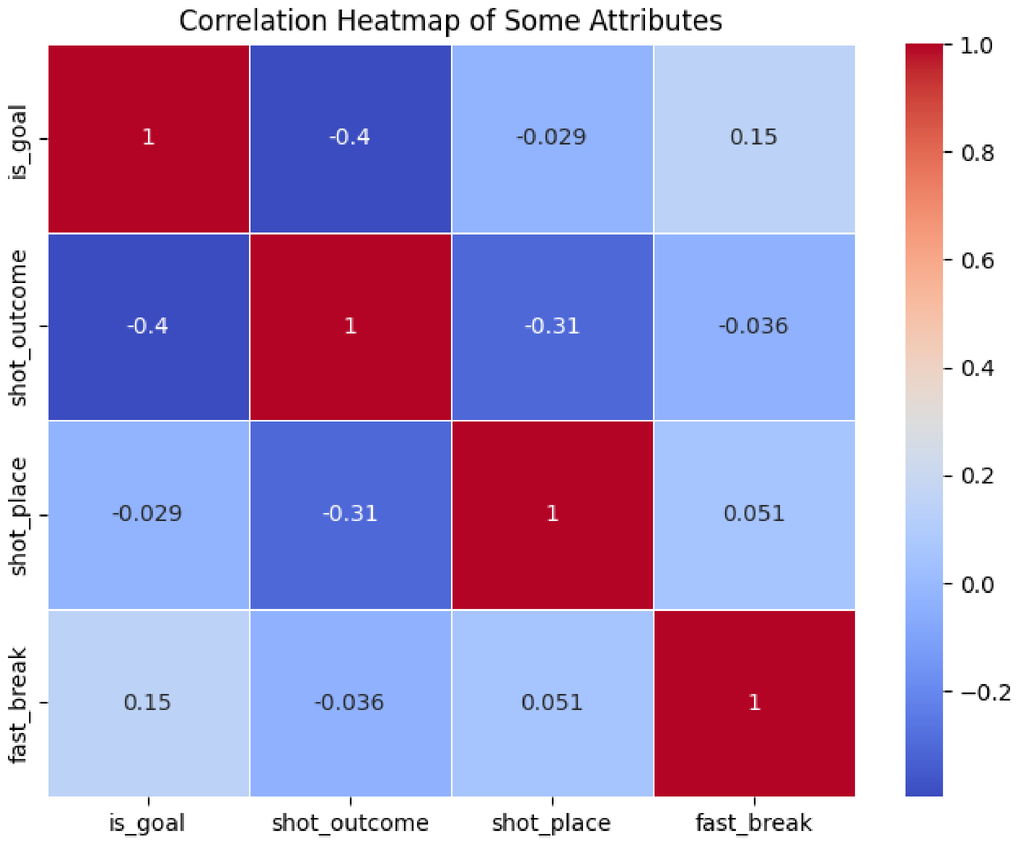


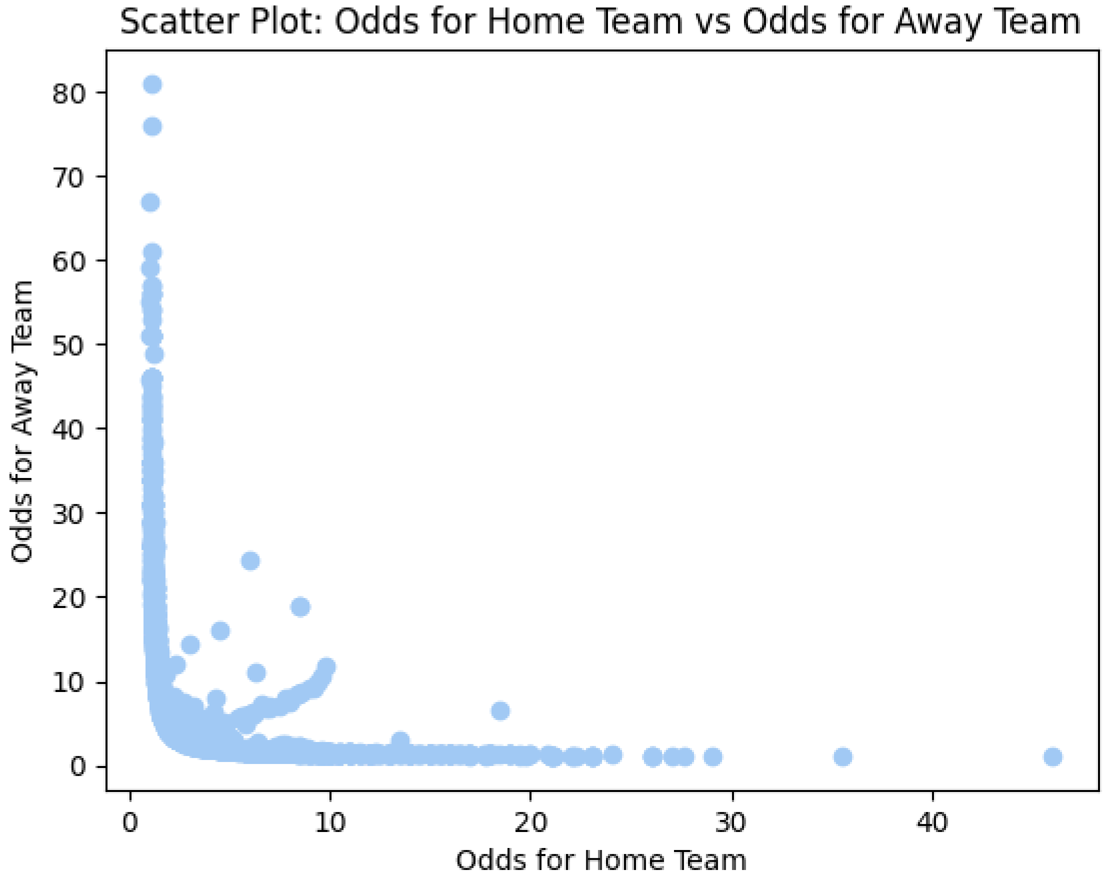
Top 10 Players by Number of Goals



Body Parts Used for Shots


Distribution of Situations
200000
175000
150000
125000
100000
75000
50000
25000
0
Count
Open Play
Set piece
Corner
Free kick
Situation


Correlation Heatmap of Some Attributes
is_goal
shot_outcome
shot_place
fast_break
1
-0.4
-0.029
0.15
-0.4
1
-0.31
-0.036
-0.029
-0.31
1
0.051
0.15
-0.036
0.051
1
is_goal
shot_outcome
shot_place
fast_break
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

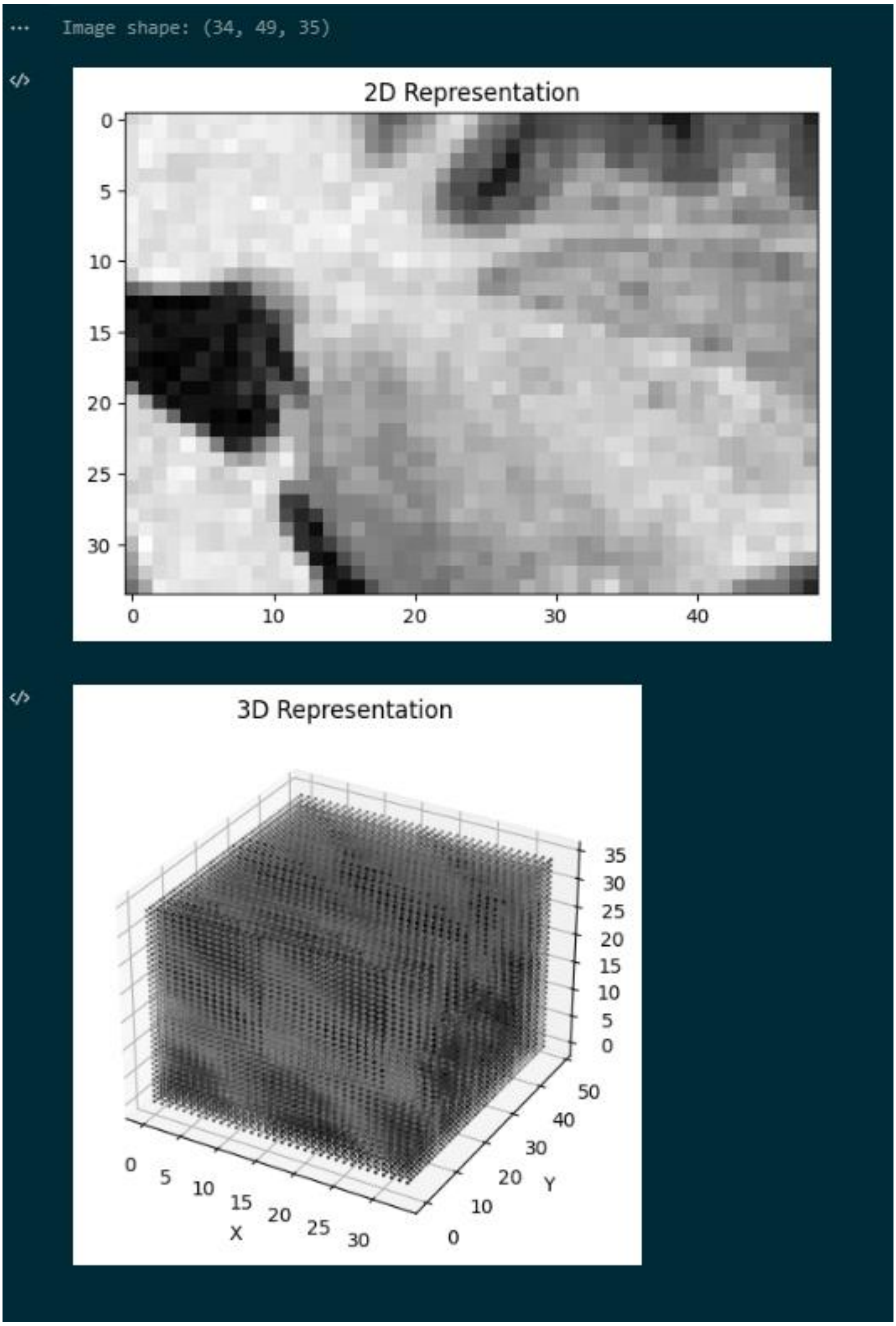

Scatter Plot: Odds for Home Team vs Odds for Away Team
Odds for Away Team
80
70
60
50
40
30
20
10
0
0
10
20
30
40
Odds for Home Team

Question 2

Write a python program for importing Medical image data into NumPy, SciPy, and Pandas arrays. Demonstrate the ways of representing the medical images in two-dimensional (2D) and three-dimensional (3D) format. Note: You can use any form (CSV, JSON, and XLSX) of medical image dataset (50MB-1GB).

```
import numpy as np

import nibabel as nib

import matplotlib.pyplot as plt


# Load the medical image data using nibabel

img_path = 'image.nii'

img = nib.load(img_path)


# Get the raw image data as a NumPy array

img_data = img.get_fdata()


# Print the shape of the image data array

print('Image shape:', img_data.shape)


# Generate a 2D plot of the image data

plt.title('2D Representation')

plt.imshow(img_data[:, :, -1], cmap='gray')

plt.show()


# Generate a 3D plot of the image data

fig = plt.figure()

ax = fig.add_subplot(111, projection='3d')


x_dim, y_dim, z_dim = img_data.shape

x_range = range(x_dim)
```

```python
y_range = range(y_dim)
z_range = range(z_dim)

# Create a meshgrid for the three dimensions
X, Y, Z = np.meshgrid(x_range, y_range, z_range)

# Flatten the meshgrid and image data arrays
X = X.flatten()
Y = Y.flatten()
Z = Z.flatten()
img_flattened = img_data.flatten()

# Plotting the 3D image
ax.scatter(X, Y, Z, c=img_flattened, cmap='gray', s=1)
plt.title('3D Representation')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

plt.show()
```



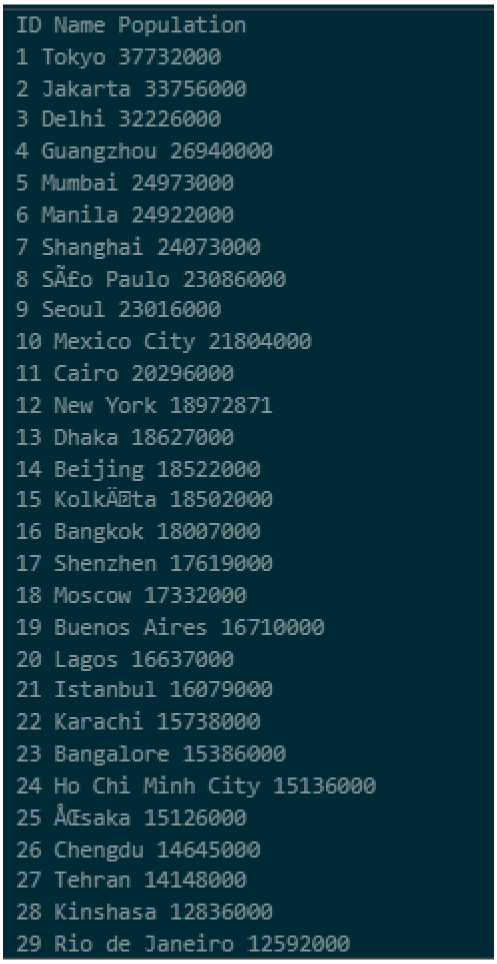
```
Image shape: (34, 49, 35)
```

Question 3

Write a python program for importing data from a database engine SQLite3. For this exercise you can use world.sql example table which holds the world's city names and populations. This table has more than 5,000 entries. Query: SELECT ID, Name, Population FROM City ORDER BY Population DESC LIMIT 1000 ID, Name, and Population are columns (fields) of the table City from which we select data. ORDER BY tells the database engine to sort our data by the Population column, and DESC means descending order. LIMIT allows us to get just the first 1,000 records found.

```python
# Importing sqlite3 module

import sqlite3 as sql


# Connect to the database

conn = sql.connect('world.db')


# Create a cursor

cursor = conn.cursor()


sql_statement = """SELECT id, name, population FROM CITY ORDER BY
population DESC LIMIT 1000"""


cursor.execute(sql_statement)


rows = cursor.fetchall()


print("ID", "Name", "Population")


for row in rows:

    print(row[0], row[1], row[2])
```

```
# Commiting and closing cursor and connection

conn.commit()

cursor.close()

conn.close()
```



```
ID Name Population
1 Tokyo 37732000
2 Jakarta 33756000
3 Delhi 32226000
4 Guangzhou 26940000
5 Mumbai 24973000
6 Manila 24922000
7 Shanghai 24073000
8 SÃ£o Paulo 23086000
9 Seoul 23016000
10 Mexico City 21804000
11 Cairo 20296000
12 New York 18972871
13 Dhaka 18627000
14 Beijing 18522000
15 KolkÄ�ta 18502000
16 Bangkok 18007000
17 Shenzhen 17619000
18 Moscow 17332000
19 Buenos Aires 16710000
20 Lagos 16637000
21 Istanbul 16079000
22 Karachi 15738000
23 Bangalore 15386000
24 Ho Chi Minh City 15136000
25 ÅŒsaka 15126000
26 Chengdu 14645000
27 Tehran 14148000
28 Kinshasa 12836000
29 Rio de Janeiro 12592000
```