ESSENTIALS OF DATA SCIENCE All DIVISIONS

Theory Activity No. 1

NAME : Indrajeet Gire
DIVISION : CS5
ROLL NO : CS5-48
PRN : 202401100059
SUBJECT : EDS

20 Problem Statements for FIFA Dataset:

1. Find the total number of players in the dataset.
2. Find the average age of players.
3. Identify the youngest player.
4. Identify the oldest player.
5. Find the player with the highest overall rating.
6. Find the player with the lowest potential.
7. List the top 5 players based on their value.
8. Find the average wage of all players.
9. How many players have a release clause above €100 million?
10. Find the most common nationality among players.
11. Calculate the average overall rating for players from Brazil.

12. Identify the club with the highest number of players.
13. Find the average height and weight of players.
14. Find players with a skill moves rating of 5.
15. How many players are Goalkeepers (GK)?
16. Find the player with the highest jumping attribute.
17. Find the most valuable goalkeeper.
18. Calculate the total value of players in Manchester United.
19. Find the average strength attribute of all players.
20. Find how many players have a long passing skill above 85.

Code:

```
import pandas as pd


import numpy as np


# Load dataset


df = pd.read_csv('/mnt/data/data set.csv')




# View first few rows
```

```
df.head()
```

# 1. Total number of players -:

```
total_players = df.shape[0]

print("Total Players:", total_players)
```

# 2. Average age of players -:

```
average_age = df['Age'].mean()

print("Average Age:", average_age)
```

# 3. Youngest player -:

```
youngest_player = df[df['Age'] == df['Age'].min()]
print(youngest_player[['Name', 'Age']])
```

# 4. Oldest player -:

```
oldest_player = df[df['Age'] == df['Age'].max()]
print(oldest_player[['Name', 'Age']])
```

# 5. Player with the highest overalrating-:

```
top_overall_player = df[df['Overall'] ==
df['Overall'].max()]

print(top_overall_player[['Name', 'Overall']])
```

6. Player with the lowest potential -:

```
lowest_potential_player = df[df['Potential'] ==
df['Potential'].min()]

print(lowest_potential_player[['Name', 'Potential']])
```

7. Top 5 players based on their value -:

```
top5_value_players = df.sort_values(by='Value',
ascending=False).head(5)

print(top5_value_players[['Name', 'Value']])
```

8. Average wage of all players -:

```
average_wage = df['Wage'].mean()

print("Average Wage:", average_wage)
```

9. Players with release clause above €100 million -:

```
players_100m = df[df['Release Clause'] > 100_000_000]

print(players_100m[['Name', 'Release Clause']])
```

10. Most common nationality -:

```
common_nationality = df['Nationality'].mode()[0]

print("Most Common Nationality:", common_nationality)
```

11. Average overall rating for Brazilian players -:

```
brazilian_avg = df[df['Nationality'] ==
'Brazil']['Overall'].mean()

print("Average Overall of Brazilians:", brazilian_avg)
```

12. Club with the highest number of players -:

```
top_club = df['Club'].value_counts().idxmax()

print("Club with most players:", top_club)
```

13. Average height and weight of players -:

```
average_height = df['Height'].mean()

average_weight = df['Weight'].mean()

print("Average Height:", average_height)
print("Average Weight:", average_weight)
```

14. Players with skill moves rating 5 -:

```
skill5_players = df[df['Skill Moves'] == 5]

print(skill5_players[['Name', 'Skill Moves']])
```

15. Number of Goalkeepers -:

```
goalkeepers = df[df['Position'] == 'GK'].shape[0]

print("Number of Goalkeepers:", goalkeepers)
```

16. Player with highest jumping attribute -:

```
highest_jump = df[df['Jumping'] == df['Jumping'].max()]

print(highest_jump[['Name', 'Jumping']])
```

17. Most valuable goalkeeper -:

```
most_valuable_gk = df[(df['Position'] ==
'GK')].sort_values(by='Value', ascending=False).head(1)

print(most_valuable_gk[['Name', 'Value']])
```

18. Total value of Manchester United players -:

```
manu_value = df[df['Club'] == 'Manchester
United']['Value'].sum()

print("Total Value (Manchester United):", manu_value)
```

19. Average strength attribute of all players -:

```
average_strength = df['Strength'].mean()

print("Average Strength:", average_strength)
```

20. Players with long passing above 85 -:

python

CopyEdit

```
long_passing_85 = df[df['Long Passing'] > 85]

print(long_passing_85[['Name', 'Long Passing']])
```

Output:

1. **Total Sales Revenue:** 10032628.85
2. **Average Price Each:** 83.65854410201914
3. **Quantity Sold per Product Line:**

```
 PRODUCTLINE
Classic Cars         33992
Motorcycles          11663
Planes               10727
Ships                 8127
Trains                2712
Trucks and Buses     10777
Vintage Cars         21069
Name: QUANTITYORDERED, dtype: int64
```

4. **Revenue by Country:**

```
 COUNTRY
Australia       630623.10
Austria         202062.53
Belgium         108412.62
Canada          224078.56
Denmark         245637.15
Finland         329581.91
France         1110916.52
Germany         220472.09
Ireland          57756.43
Italy           374674.31
Japan           188167.81
Norway          307463.70
Philippines      94015.73
Singapore       288488.41
Spain          1215686.92
```

```
Sweden             210014.21
Switzerland        117713.56
UK                 478880.46
USA               3627982.83
Name: SALES, dtype: float64
```

5. Most Popular Product Line:

```
Classic Cars
```

6. Orders per Year:

```
  YEAR_ID
2004      1345
2003      1000
2005       478
Name: count, dtype: int64
```

7. Unique Products

Sold: 109

8. Highest Sale Value:

```
14082.8
```

9. Max Quantity Order:

```
    ORDERNUMBER QUANTITYORDEREDPRODUCTLINE
418           10405                    97 Classic Cars
```

10. Deal Size Counts:

```
  DEALSIZE
Medium     1384
Small      1282
Large       157
Name: count, dtype: int64
```

11. Average Sale per Order Line: 3553.889071909316

12. **Correlation Between Quantity and Sales:**

```
                      QUANTITYORDERED    SALES
QUANTITYORDERED             1.000000 0.551426
SALES                       0.551426 1.000000
```

13. **Earliest Order Date:** 2003-01-06 00:00:00

14. **Orders per Status:**

```
  STATUS
Shipped        2617
Cancelled        60
Resolved         47
On Hold          44
In Process       41
Disputed         14
Name: count, dtype: int64
```

15. **Revenue by Status:**

```
  STATUS
Cancelled       194487.48
Disputed         72212.86
In Process      144729.96
On Hold         178979.19
Resolved        150718.28
Shipped        9291501.08
Name: SALES, dtype: float64
```

16. **Discounted Sales Sample:**

```
     SALES DISCOUNTED_SALE
0 2871.00         2583.900
1 2765.90         2489.310
2 3884.34         3495.906
3 3746.70         3372.030
4 5205.27         4684.743
```

17. **Profit Sample:**

```
 PRICEEACH MSRP QUANTITYORDERED PROFIT
0      95.70   95              30 -21.00
1      81.35   95              34 464.10
2      94.74   95              41   10.66
3      83.26   95              45 528.30
4     100.00   95              49 -245.00
```

18. **Country with Highest Avg Sales:** Denmark

19. **Customers per Country:**

```
 COUNTRY
USA            1004
Spain           342
France          314
Australia       185
UK              144
Italy           113
Finland          92
Norway           85
Singapore        79
Canada           70
Denmark          63
Germany          62
Sweden           57
Austria          55
Japan            52
Belgium          33
Switzerland      31
Philippines      26
Ireland          16
Name: count, dtype: int64
```

20. **Negative Profit Orders:**

```
    ORDERNUMBERPROFIT PRODUCTLINE
```

```
0              10107          -21.00 Motorcycles
4                   10159 -245.00 Motorcycles
5              10168          -59.76 Motorcycles
7                   10188 -240.00 Motorcycles
8              10201          -78.54 Motorcycles
...               ...          ...             ...
2818            10350 -920.00              Ships
2819           10373 -1334.00              Ships
2820           10386 -1978.00              Ships
2821            10397 -280.16              Ships
2822            10414 -541.44              Ships
```