

TENNIS DATA ANALYTICS



Presentation

OVERVIEW



- 01 - Introduction to the business problem and the objectives of the project

02 - Brief description of the dataset and its sources

03 - Statement of the research question and the expected outcomes

INTRODUCTION

1. Which factors have the greatest impact on a player's chances of winning a match?
2. How do these factors vary depending on the surface of the court, the level of the tournament, and other factors?
3. Are there any interactions between factors that have a significant impact on match outcomes?
4. How can players and coaches use this information to improve their performance?



01 DATASET DESCRIPTION

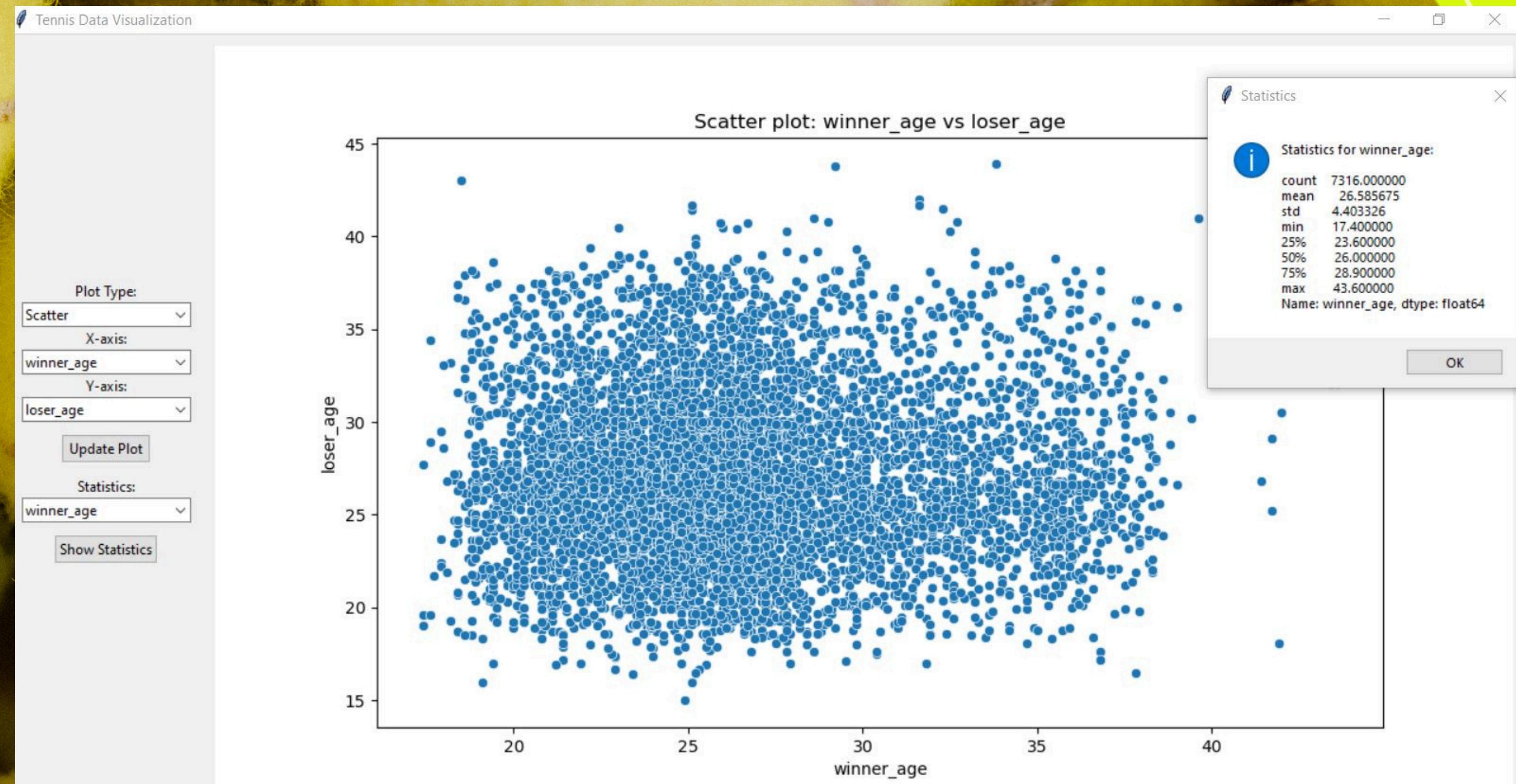
**some columns not relevant for
analyzing the key factors in match
success: 11**

**focus on columns that provide information
about the players' performance during the
match, such as their serve statistics**



HERE IS A COMPLETE AND FUNCTIONAL PYTHON APPLICATION FOR VIEWING TENNIS DATA FROM THE DATASET PROVIDED

02 E.D.A ?



03 KPI

- SELECTION OF RELEVANT KPIS TO MEASURE THE PERFORMANCE OF PLAYERS AND THE SUCCESS OF MATCHES
- CALCULATION OF KPIS FOR EACH PLAYER AND MATCH IN THE DATASET
- ANALYSIS OF THE RELATIONSHIP BETWEEN KPIS AND MATCH SUCCESS

```
# Calculate difference in serve percentage and return percentage between  
winner and loser :
```

```
df['serve_pct_diff'] = df['w_serve_pct'] - df['l_serve_pct']
```

```
df['return_pct_diff'] = df['w_return_pct'] - df['l_return_pct']
```

03- KPI CALCULATION



04 HYPOTHESIS TESTING

- FORMULATION OF HYPOTHESES BASED ON THE EDA AND KPI ANALYSIS
- SELECTION OF APPROPRIATE STATISTICAL TESTS TO TEST THE HYPOTHESES
- INTERPRETATION OF THE RESULTS AND EVALUATION OF THE EVIDENCE IN SUPPORT OF THE HYPOTHESES

04 - HYPOTHESIS TESTING

RACKET
SPORTSWEAR
TENNIS BALL
WRISTBANDS
CLUE



1. **Hypothesis 1: Higher-ranked players have a higher serve percentage.**
 - **Null Hypothesis (H_0): There is no difference in serve percentage between higher-ranked and lower-ranked players.**
 - **Alternative Hypothesis (H_1): Higher-ranked players have a higher serve percentage than lower-ranked players.**

```
# Is more advantageous right player than left hand or the inverse ?
```

```
# calculate the win percentage of right-handed players against left-handed players in a tennis match  
descriptive statistics for EDA.
```

```
# |counts the number of rows in the df where the 'winner_hand' column is 'R' (right-handed)
```

```
# and the 'Loser_hand' column is 'L' (left-handed).
```

```
#giving the number of matches won by right-handed players against left-handed players.
```

```
R = len(atp_combined_df[(atp_combined_df.winner_hand == 'R') & (atp_combined_df.loser_hand == 'L')])
```

```
#Similar to the previous line, but here for left-handed players against right-handed one.
```

```
L = len(atp_combined_df[(atp_combined_df.winner_hand == 'L') & (atp_combined_df.loser_hand == 'R')])
```

```
# The win percentage is calculated as the number of matches won by right-handed players (R)
```

```
# divided by the total number of matches played between right-handed and left-handed players (R + L),
```

```
# multiplied by 100 to get the percentage and finally round the percentage to 2 decimal places.
```

```
if R + L > 0:
```

```
    percentage = round(100 * np.true_divide(R, R + L), 2)
```

```
    print(f'The right hand player has a {percentage}% chance of winning against a left hand player')
```

```
else:
```

```
    print("No data available for right-handed vs left-handed matches")
```

The right hand player has a 53.16% chance of winning against a left hand player

Do players from certain countries (ioc) have a higher win rate on specific surfaces?

```
# Calculate win rates for each country (ioc) on each surface
win_rates = atp_combined_df.groupby(['winner_ioc', 'surface']).size().unstack(fill_value=0)
win_rates = win_rates.div(win_rates.sum(axis=1), axis=0)

# Perform one-way ANOVA for win rates on different surfaces
surfaces = ['Hard', 'Clay', 'Grass']
surface_rates = [win_rates[surface].dropna() for surface in surfaces]

f_statistic, p_value = stats.f_oneway(*surface_rates)

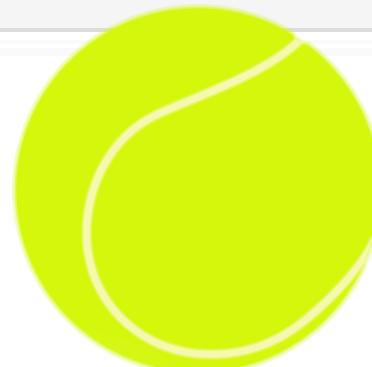
print("One-way ANOVA for IOC win rates on different surfaces:")
print(f"F-statistic: {f_statistic}")
print(f"p-value: {p_value}")
```

One-way ANOVA for IOC win rates on different surfaces:

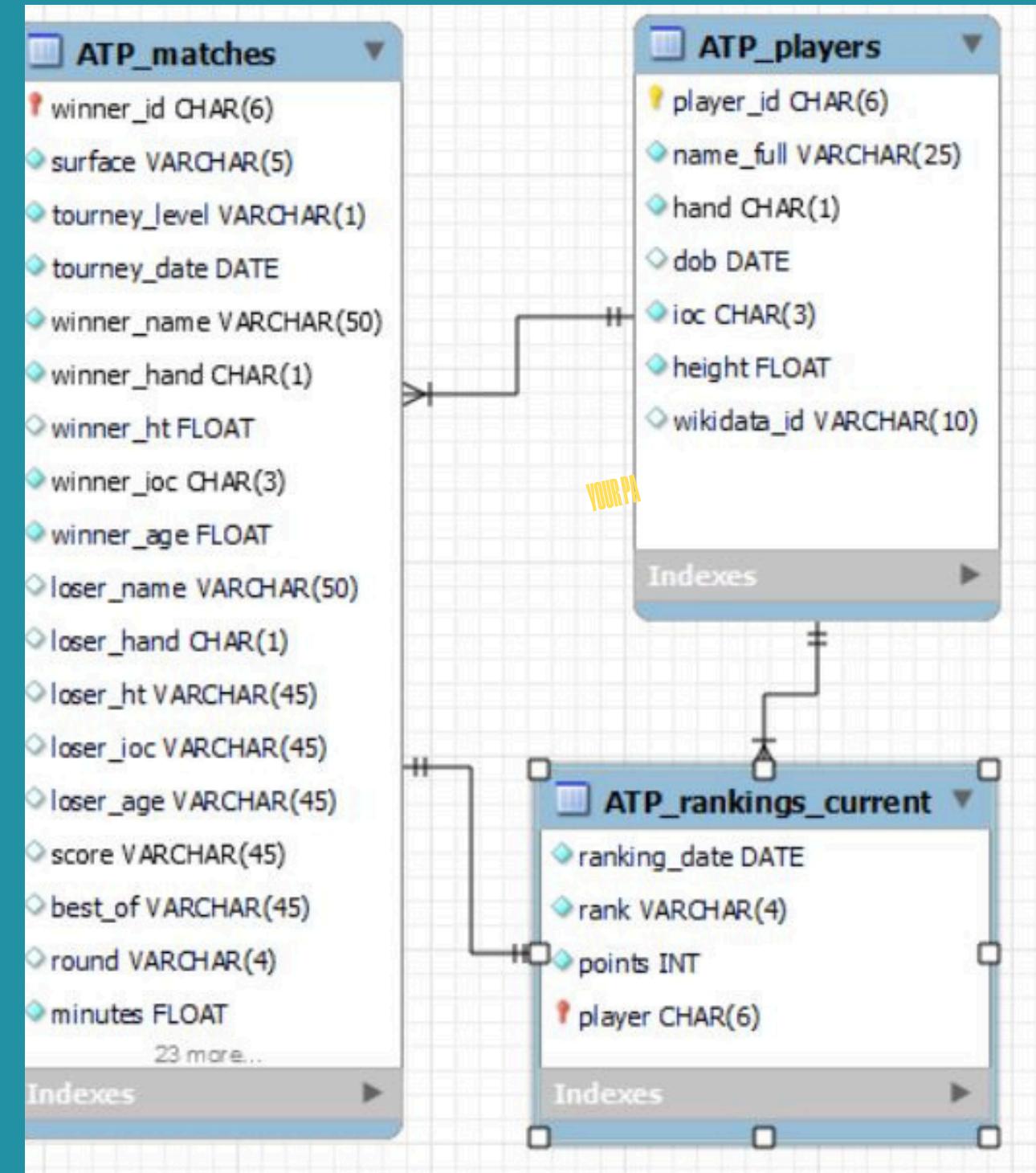
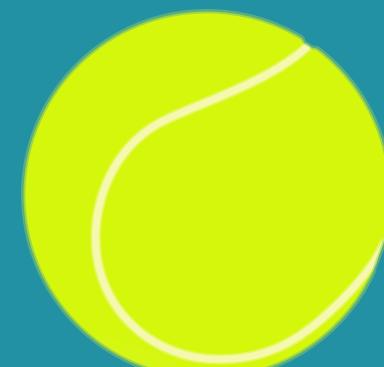
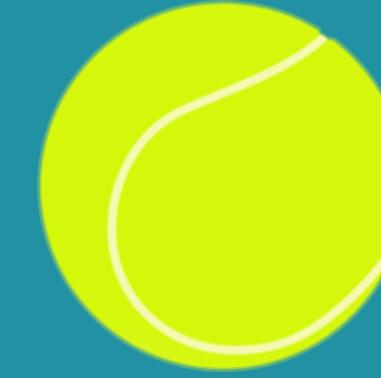
F-statistic: 76.21237917973812

p-value: 3.248305491186151e-26

04 HYPOTHESIS TESTING



5 SQL PART



5 SQL PART

```
SELECT
CASE
    WHEN winner_age < 20 THEN 'Under 20'
    WHEN winner_age BETWEEN 20 AND 25 THEN '20-25'
    WHEN winner_age BETWEEN 26 AND 30 THEN '26-30'
    WHEN winner_age BETWEEN 31 AND 35 THEN '31-35'
    ELSE 'Over 35'
END AS age_group,
COUNT(*) AS winner_count
FROM
atp_matches
GROUP BY
age_group
ORDER BY
CASE age_group
    WHEN 'Under 20' THEN 1
    WHEN '20-25' THEN 2
    WHEN '26-30' THEN 3
    WHEN '31-35' THEN 4
    ELSE 5
END;
```

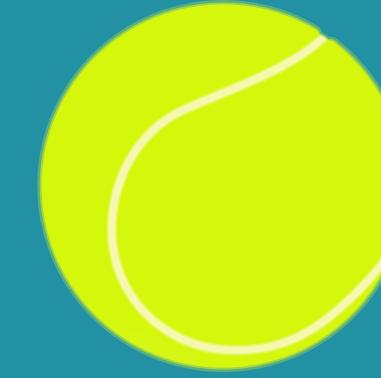
Result Grid | Filter Rows:

	age_group	winner_count
▶	Under 20	42
▶	20-25	522
▶	26-30	261
▶	31-35	174
▶	Over 35	236

Result Grid | Filter Rows:

	age_group	loser_count
▶	Under 20	25
▶	20-25	414
▶	26-30	349
▶	31-35	215
▶	Over 35	232

5 SQL PART



The screenshot shows a database management interface with a central SQL editor window. On the left, the 'SCHEMAS' panel lists databases: sakila, sys, and tennis_db. The tennis_db schema is expanded to show Tables (atp_matches, atp_players, atp_rankings_current) and Views (top_ranked_players_by_country). The 'Information' tab is selected. Below it, the 'Table: atp_rankings_current' section shows its columns: rank (varchar(4)), ranking_date (date), points (int), and player (char(6) PK). The SQL editor window contains the following code:

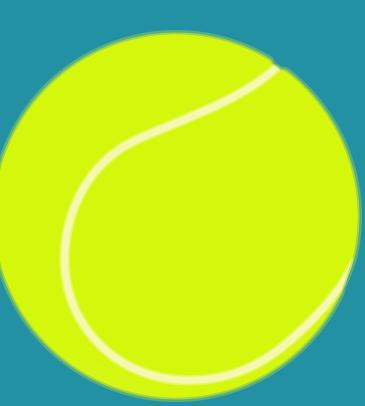
```
1 • use tennis_db;
2 • DROP TABLE IF EXISTS atp_players;
3 • DROP TABLE IF EXISTS atp_rankings_current;
4 • SHOW TABLES;
5 • DESCRIBE atp_players;
6 • DESCRIBE atp_rankings_current;
7 • CREATE VIEW top_ranked_players_by_country AS
8     SELECT
9         p.ioc,
10        p.name_full,
11        r.`rank`
12     FROM
13        atp_players p
14     JOIN
15        atp_rankings_current r ON p.player_id = r.player
16     WHERE
17        p.ioc IS NOT NULL AND r.`rank` IS NOT NULL
18     ORDER BY
19        r.`rank` ASC;
```

Creating a view for
top_ranked_players_by_country

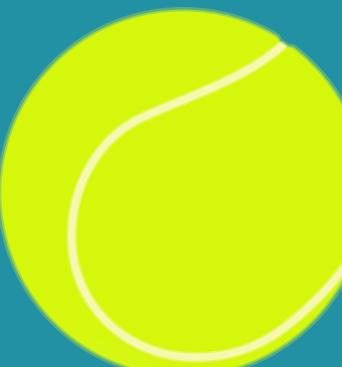
5 SQL PART

```
SELECT
    p.name_full,
    COUNT(*) AS grand_slam_wins
FROM
    atp_players p
JOIN
    atp_matches m ON p.player_id = m.winner_id
WHERE
    m.tourney_level = 'G'
GROUP BY
    p.name_full
ORDER BY
    grand_slam_wins DESC;
```

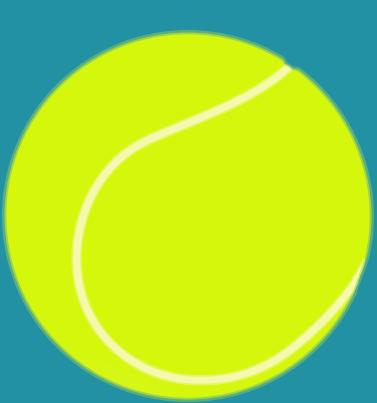
5 SQL PART



```
1      -- Players with the largest improvement in Ranking
2 • CREATE VIEW players_with_largest_ranking_improvement AS
3     SELECT
4         p.name_full,
5             MAX(r1.`rank`) - MIN(r2.`rank`) AS ranking_improvement
6     FROM
7         atp_players p
8     JOIN
9         atp_rankings_current r1 ON p.player_id = r1.player
10    JOIN
11        atp_rankings_current r2 ON p.player_id = r2.player
12    WHERE
13        r1.ranking_date < r2.ranking_date
14    GROUP BY
15        p.name_full
16    ORDER BY
17        ranking_improvement DESC;
```



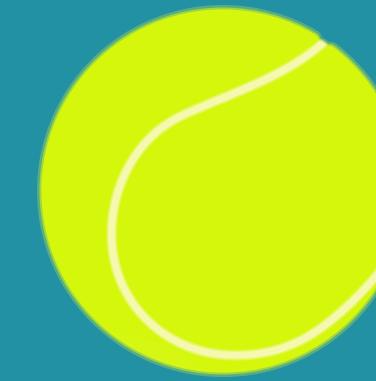
5 SQL PART



```
-- Average ranking of players at different tournaments
SELECT
    p.name_full,
    AVG(CASE WHEN m.tourney_name = 'Australian Open' THEN r.rank END) AS australian_open_rank,
    AVG(CASE WHEN m.tourney_name = 'Roland Garros' THEN r.rank END) AS french_open_rank,
    AVG(CASE WHEN m.tourney_name = 'Wimbledon' THEN r.rank END) AS wimbledon_rank,
    AVG(CASE WHEN m.tourney_name = 'US Open' THEN r.rank END) AS us_open_rank
FROM
    atp_players p
JOIN
    atp_matches m ON p.player_id = m.winner_id
JOIN
    atp_rankings_current r ON p.player_id = r.player
WHERE
    m.tourney_name IN ('Australian Open', 'Roland Garros', 'Wimbledon', 'US Open')
GROUP BY
    p.name_full;
```



CREATE AN API USING A WEB FRAMEWORK : FLASK IN PYTHON



3 API ENDPOINTS CREATION

API USE

1. **/TOP_PLAYERS:** RETURNS THE TOP 10 PLAYERS BASED ON THE NUMBER OF MATCHES WON.

INSTALL PANDAS & FLASK LIBRARIES

1. **/SURFACE_STATS:** PROVIDES AVERAGE STATISTICS FOR EACH PLAYING SURFACE
- 2.
- 3.

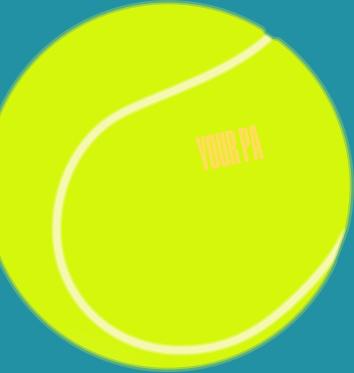
SAVE THE SCRIPT AS TENNIS_API.PY IN THE SAME DIRECTORY AS YOUR CSV AND RUN IT.

1. **/PLAYER_STATS/<PLAYER_NAME>:** GIVES DETAILED STATS FOR A SPECIFIC PLAYER.
- 2.
- 3.

THE API WILL BE ACCESSIBLE AT [HTTP://LOCALHOST:5000](http://localhost:5000).



CREATE AN API USING A WEB FRAMEWORK : FLASK IN PYTHON



3 API ENDPOINTS TEST

1.

TOP PLAYERS:

HTTP://LOCALHOST:5000/TOP_PLAYERS

2.

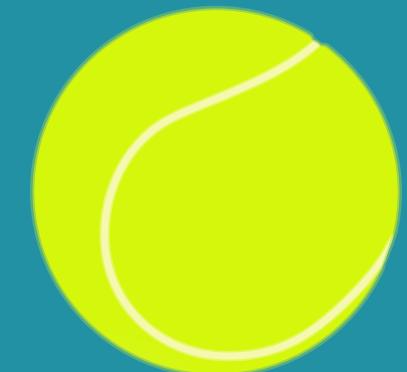
1.

SURFACE STATS:

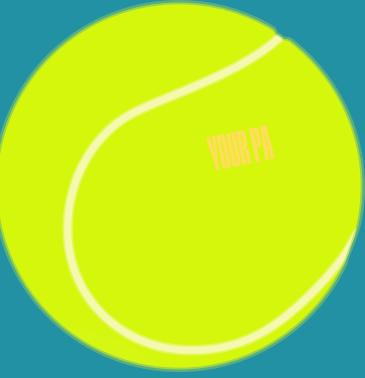
HTTP://LOCALHOST:5000/SURFACE_STATS

2.

PLAYER STATS: HTTP://LOCALHOST:5000/PLAYER_STATS/FELIX AUGER ALIASSIME



CREATE AN API USING A WEB FRAMEWORK: FLASK IN PYTHON



```
from flask import Flask, jsonify
import pandas as pd

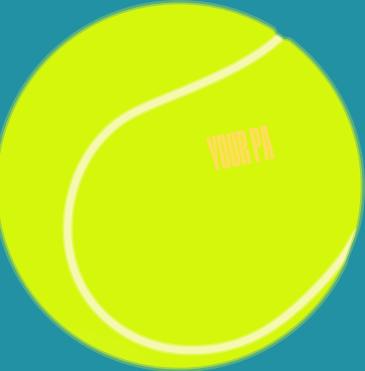
app = Flask(__name__)

# Load the CSV data
df = pd.read_csv('atp_matches_combined_cleaned.csv')

@app.route('/top_players', methods=['GET'])
def top_players():
    top_10 = df.groupby('winner_name').size().sort_values(ascending=False).head(10)
    return jsonify(top_10.to_dict())
```



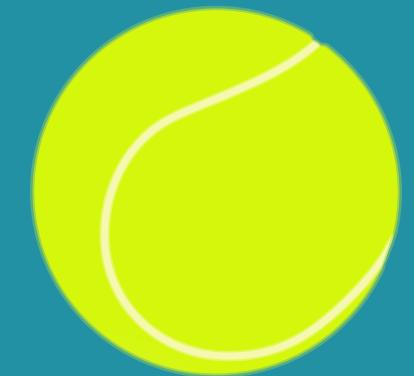
CREATE AN API USING A WEB FRAMEWORK: FLASK IN PYTHON



```
@app.route('/surface_stats', methods=['GET'])
def surface_stats():
    surface_stats = df.groupby('surface').agg({
        'minutes': 'mean',
        'w_ace': 'mean',
        'l_ace': 'mean'
    }).round(2)
    return jsonify(surface_stats.to_dict())

@app.route('/player_stats/<player_name>', methods=['GET'])
def player_stats(player_name):
    player_data = df[df['winner_name'] == player_name].agg({
        'w_ace': 'mean',
        'w_df': 'mean',
        'w_svpt': 'mean',
        'w_1stWon': 'mean',
        'w_2ndWon': 'mean'
    }).round(2)
    return jsonify(player_data.to_dict())

if __name__ == '__main__':
    app.run(debug=True)
```





THANK YOU FOR LISTENING

MANUEL



THANK YOU FOR LISTENING

MANUEL