FIFA WORLD CUP 2022 Analysis using SQL

David Heller

August 15, 2024

The FIFA World Cup, an event cherished by football enthusiasts worldwide, brings together a tapestry of emotions and moments that are etched in the annals of sporting history. While millions tune in to watch the beautiful game from the comfort of their homes, there's an electrifying atmosphere that envelopes the stadiums. This is where dreams are realized, where heroes are born, and where the heartbeat of the sport echoes through the cheers of the fans.

In the realm of international football, talent and teamwork are paramount. Every player who graces the World Cup stage is a testament to years of dedication and unwavering commitment. Their skills are honed to perfection through countless hours of practice, and their ability to adapt and strategize can be the difference between glory and heartbreak.

Yet, in the world of competitive football, it's not just about how well a team executes its own game plan; it's also about understanding and countering the strategies of their opponents. This is where a deep analysis of both the opposition and one's own strengths and weaknesses becomes pivotal.

And this is where you come in.

As an integral part of the team behind the scenes, your role is crucial. You are the strategist, the data guru, the one who can turn numbers and statistics into insights that can give a team the upper hand. Your task is to analyze not only the tactics and tendencies of rival teams but also to assess the performance and potential of your own players.

Using the latest technology and the wealth of data at your disposal, you dive into the world of football analytics. You dissect the playing style of opposing teams, scrutinizing their past performances and identifying key players who pose a threat. You help your team develop strategies to exploit weaknesses and neutralize strengths.

As the tournament progresses, the pressure mounts, and the stakes become even higher. Your work becomes a silent but essential component of the team's success. Coaches, players, and fans may not see your efforts on the field, but they feel the impact in every well-executed play and every victory.

In the world of FIFA, you are the unsung hero, the one who helps turn dreams into reality. Your dedication to the game and your ability to transform data into actionable insights contribute to the magic of the World Cup, making every goal, every save, and every moment on the pitch that much more extraordinary.

Module 1

Task 1: Data Download, Import, and Database Connection.

In [ ]:
```
import pymysql
import pandas as pd
from sqlalchemy import create_engine, text
import getpass  # To get the password without showing the input
password = getpass.getpass()
import warnings
warnings.filterwarnings("ignore") # Ignore the deprecated warning
```

In [ ]:
```
# Define the server connection string (without a specific database)
server_connection_string = f'mysql+pymysql://root:{password}@localhost/'
```

In [ ]:
```
# Create an engine and connect to the MySQL server
server_engine = create_engine(server_connection_string)
```

In [ ]:
```
# Create a new database
database_name = 'fifa_world_cup_2022_project'

with server_engine.connect() as connection:
    connection.execute(text(f"CREATE DATABASE IF NOT EXISTS {database_name};"))
    print(f"Database '{database_name}' created successfully.")
```

```
Database 'fifa_world_cup_2022_project' created successfully.
```

In [ ]:
```
# Connect to the newly created database
db_connection_string = f'mysql+pymysql://root:{password}@localhost/{database_name}'
db_engine = create_engine(db_connection_string)
```

In [ ]:
```
# Reading CSV files into pandas DataFrames
match_data_df = pd.read_csv('match_data.csv')
player_possession_df = pd.read_csv('player_possession.csv')
```

```
player_shooting_df = pd.read_csv('player_shooting.csv')
player_stats_df = pd.read_csv('player_stats.csv')
```

In [ ]:
```
# Loading DataFrames into the MySQL database
match_data_df.to_sql('match_data', con=db_engine, if_exists='replace', index=False)
player_possession_df.to_sql('player_possession', con=db_engine, if_exists='replace', index=False)
player_shootings_df.to_sql('player_shootings', con=db_engine, if_exists='replace', index=False)
player_stats_df.to_sql('player_stats', con=db_engine, if_exists='replace', index=False)

print("Data loaded successfully into the MySQL database.")
```

```
Data loaded successfully into the MySQL database.
```

Module 2

Task 1: Counting Matches.

We are conducting an analysis of the match data in our database to gain insights into the number of matches recorded. This information is crucial for understanding the scale and volume of matches in our records, which can provide valuable insights into various aspects of our business operations and performance.

In [ ]:
```
# SQL query to count the unique matches
query = """
SELECT COUNT(DISTINCT match_no) AS unique_matches_count
FROM match_data;
"""

# Execute the query and fetch the result
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    unique_matches_count = result.fetchone()[0]

# Print the result
print(f"The total number of unique matches recorded is: {unique_matches_count}")
```

```
The total number of unique matches recorded is: 64
```

Task 2: Obtaining a List of Unique Referees.

Continuing our comprehensive analysis of the match data in our database, we are now focusing on understanding the distinct referees who have officiated these matches. This aspect of our project complements our previous efforts to gauge the volume of matches, forming a holistic view of our sports-related operations.

In [ ]:
```python
# SQL query to get the list of unique referees
query = """
SELECT DISTINCT referee
FROM match_data;
"""

# Execute the query and fetch the result
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    referees = result.fetchall()

# Print the list of unique referees
for referee in referees:
    print(referee[0])
```

```
Daniele Orsato
Raphael Claus
Wilton Sampaio
Abdulrahman Ibrahim Al Jassim
Slavko Vincic
Cesar Ramos
Chris Beath
Victor Gomes
Fernando Rapallini
Ivan Barton
Mohammed Abdulla Hassan
Janny Sikazwe
Facundo Tello
Clement Turpin
Ismail Elfath
Alireza Faghani
Mario Escobar
Antonio Mateu Lahoz
Mustapha Ghorbal
Jesus Valenzuela Saez
Daniel Siebert
Szymon Marciniak
Michael Oliver
Andres Matonte
Danny Makkelie
Anthony Taylor
Bakary Papa Gassama
Matt Conger
Stephanie Frappart
```

Task 3: Finding the Hour with the Highest Frequency.

Exploring further into our sports data analysis. We're about to unveil the hour when our matches come alive in their full glory. This query will reveal the peak hour, the epicenter of our sporting excitement, where fans gather, athletes shine, and history is made. It's the hour that defines our sports narrative, and we're on the verge of discovering it.

In [ ]:
```sql
# SQL query to find the hour with the highest frequency of matches
query = """
SELECT hour, COUNT(*) AS match_count
FROM match_data
```

```
GROUP BY hour
ORDER BY match_count DESC
LIMIT 1;
"""

# Execute the query and fetch the result
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    peak_hour = result.fetchone()

# Print the hour with the highest frequency
print(f"The hour with the highest frequency of matches is: {peak_hour[0]} with {peak_hour[1]} matches.")
```

```
The hour with the highest frequency of matches is: 20:00 with 24 matches.
```

Task 4: Fetching Data for Match Number 5.

Continuing our data-driven exploration of the world of sports, we're now embarking on a journey into the heart of a specific match, known simply as "Match Number 5." This match holds within its data the unique story of competition, strategy, and triumph or defeat.

With our sights set on this particular event, we are poised to uncover the intricate details that unfolded during the game – from the players on the field to the actions that shaped its outcome. It's a deep dive into the essence of sports, where every pass, every shot, and every decision becomes a part of the narrative that is Match Number 5.

In [ ]:
```
# SQL query to fetch all data for Match Number 5
query = """
SELECT *
FROM match_data
WHERE match_no = 5;
"""

# Execute the query and fetch the result
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    match_details = result.fetchone()

# Fetch column names
column_names = result.keys()
```

```python
# Print the details of Match Number 5 with column names
print("Details of Match Number 5:")
for column, value in zip(column_names, match_details):
    print(f"{column}: {value}")
```

Details of Match Number 5:
match_no: 5
day_of_week: Tue
date: 2022-11-22
hour: 11:00
venue: Lusail Iconic Stadium
referee: Slavko Vincic
group: Group C
first_team: ARGENTINA
second_team: SAUDI ARABIA
attendance: 88012
1_poss: 69
2_poss: 31
1_goals: 1
2_goals: 2
score: 1,2
1_attempts: 14
2_attempts: 3
1_goal_inside_penalty_area: 1
2_goal_inside_penalty_area: 2
1_goal_outside_penalty_area: 0
2_goal_outside_penalty_area: 0
1_ontarget: 6
2_ontarget: 2
1_offtarget: 5
2_offtarget: 0
1_attempts_inside_penalty_area: 10
2_attempts_inside_penalty_area: 3
1_attempts_outside_penalty_area: 4
2_attempts_outside_penalty_area: 0
1_yellow_cards: 0
2_yellow_cards: 6
1_red_cards: 0
2_red_cards: 0
faul_against_1: 7
faul_against_2: 21
1_offsides: 10
2_offsides: 1
1_passes: 610
2_passes: 267
1_passes_compeletd: 529
2_passes_compeletd: 190

```
1_corners: 9
2_corners: 2
1_free_kicks: 22
2_free_kicks: 16
1_panelties_scored: 1
2_panelties_scored: 0
1_goal_prevented: 4
2_goal_prevented: 14
1_own_goal: 0
2_own_goal: 0
1_forced_turnovers: 65
2_forced_turnovers: 80
1_defensive_pressure_applied: 163
2_defensive_pressure_applied: 361
```

Task 5: Fetching Position Data for Match Number 5.

Our journey through the world of sports data brings us closer to the heart of Match Number 5. We are about to uncover the elemental aspect of possession – who held the reins of control during this match. This query will reveal the ebb and flow of dominance on the field, showcasing the strategic maneuvers, tactical brilliance, and perhaps even moments of contention.

Stay tuned as we delve into the possession statistics of Match Number 5, painting a vivid picture of which team dictated the pace and flow of this captivating contest. It's a window into the dynamics that shape the outcome of sports matches, and we're poised to witness it firsthand.

In [ ]:
```python
# SQL query to fetch possession data for Match Number 5
query = """
SELECT match_no, first_team, second_team, 1_poss AS first_team_possession, 2_poss AS second_team_possession
FROM match_data
WHERE match_no = 5;
"""

# Execute the query and fetch the result
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    possession_data = result.fetchone()

# Fetch column names
column_names = result.keys()
```

```
# Print the possession data of Match Number 5 with column names
print("Possession Data for Match Number 5:")
for column, value in zip(column_names, possession_data):
    print(f"{column}: {value}")
```

```
Possession Data for Match Number 5:
match_no: 5
first_team: ARGENTINA
second_team: SAUDI ARABIA
first_team_possession: 69
second_team_possession: 31
```

Task 6: Retrieving Goal Prevention Data for Match Number 5.

As our exploration of Match Number 5 continues, our attention now shifts to a critical aspect of any sports event – goal prevention.We're about to unveil the defensive prowess displayed by both teams during this match. This query will provide insight into the resilience, strategy, and teamwork exhibited by the players as they worked tirelessly to thwart the opposing team's goal-scoring attempts.

In [ ]:
```
# SQL query to fetch goal prevention data for Match Number 5
query = """
SELECT match_no, first_team, second_team, 1_goal_prevented AS first_team_goals_prevented, 2_goal_prevented AS second_
FROM match_data
WHERE match_no = 5;
"""

# Execute the query and fetch the result
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    goal_prevention_data = result.fetchone()

# Fetch column names
column_names = result.keys()

# Print the goal prevention data of Match Number 5 with column names
print("Goal Prevention Data for Match Number 5:")
for column, value in zip(column_names, goal_prevention_data):
    print(f"{column}: {value}")
```

```
Goal Prevention Data for Match Number 5:
match_no: 5
first_team: ARGENTINA
second_team: SAUDI ARABIA
first_team_goals_prevented: 4
second_team_goals_prevented: 14
```

Task 7: Finding Peak Performance.

Our journey through the data of Match Number 5 takes an intriguing turn as we zero in on a critical aspect of the game – on-target shots. We are delving into the thrilling moments when players aimed for precision and accuracy. This query seeks to identify the instances when Team 1 delivered on-target shots at their highest level.

In [ ]:
```python
# SQL query to fetch peak performance data for Team 1 during Match Number 5
query = """
SELECT match_no, first_team, 1_ontarget AS first_team_on_target_shots
FROM match_data
WHERE match_no = 5
ORDER BY 1_ontarget DESC
LIMIT 1;
"""

# Execute the query and fetch the result
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    peak_performance_data = result.fetchone()

# Fetch column names
column_names = result.keys()

# Print the peak performance data for Team 1 during Match Number 5
print("Peak Performance Data for Team 1 during Match Number 5:")
for column, value in zip(column_names, peak_performance_data):
    print(f"{column}: {value}")
```

```
Peak Performance Data for Team 1 during Match Number 5:
match_no: 5
first_team: ARGENTINA
first_team_on_target_shots: 6
```

Task 8: Identifying Team 2's Top On-Target Performance.

Our data exploration continues to unravel the captivating narratives hidden within our sports dataset. Our latest query dives deep into the performance of the second team, shedding light on their precision and effectiveness in hitting the target.

We are about to uncover pivotal moments when the second team showcased their accuracy, transforming on-target shots into goals. This data provides insight into their ability to seize opportunities and potentially shape the outcome of the match.

In [ ]:
```
# SQL query to fetch the top on-target performance, goals, and the goals/shots_on_target ratio for Team 2 during Matc
query = """
SELECT
    match_no,
    second_team,
    2_goals AS second_team_goals,
    2_ontarget AS second_team_on_target_shots,
    (2_goals / 2_ontarget) AS goals_to_on_target_ratio
FROM match_data
WHERE match_no = 5;
"""

# Execute the query and fetch the result
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    team2_performance_data = result.fetchone()

# Fetch column names
column_names = result.keys()

# Print the detailed performance data for Team 2 during Match Number 5
print("Performance Data for Team 2 during Match Number 5:")
for column, value in zip(column_names, team2_performance_data):
    print(f"{column}: {value}")
```

```
Performance Data for Team 2 during Match Number 5:
match_no: 5
second_team: SAUDI ARABIA
second_team_goals: 2
second_team_on_target_shots: 2
goals_to_on_target_ratio: 1.0000
```

Task 9: Identifying Match with Maximum Attendance.

Our data-driven journey takes an exciting turn as we venture into the realm of match attendance. Within our dataset lies a story waiting to be told—a story of the match that drew the largest crowd, where the energy and enthusiasm of the fans converged to create an unforgettable experience.

We are on the cusp of discovering the match that captured the hearts and imaginations of the most spectators. This data will not only unveil the grandeur of the venue but also reflect the significance of the contest itself.

In [ ]:
```
# SQL query to identify the match with the maximum attendance
query = """
SELECT
    match_no,
    first_team,
    second_team,
    venue,
    attendance
FROM match_data
ORDER BY attendance DESC
LIMIT 1;
"""

# Execute the query and fetch the result
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    max_attendance_data = result.fetchone()

# Fetch column names
column_names = result.keys()

# Print the details of the match with maximum attendance
print("Match with Maximum Attendance:")
for column, value in zip(column_names, max_attendance_data):
    print(f"{column}: {value}")
```

```
Match with Maximum Attendance:
match_no: 61
first_team: ARGENTINA
second_team: CROATIA
venue: Lusail Iconic Stadium
attendance: 88966
```

Task 10: Analyzing Team Performance at Al Janoub Stadium.

Our data exploration now brings us to the illustrious "Al Janoub Stadium", a venue that has witnessed remarkable sportsmanship and unforgettable moments. We are about to dive into the statistics of matches held at this iconic stadium. This data will unveil not just the teams' possession and goals, but also the efficiency with which they converted possession into goals.

Stay tuned as we dissect the performances, revealing which team maximized their possession and turned it into successful goal-scoring opportunities at "Al Janoub Stadium." It's a glimpse into the strategic brilliance and effectiveness of teams in a venue that has become a legendary stage for sports excellence.

In [ ]:
```sql
# SQL query to analyze team performance at Al Janoub Stadium
query = """
SELECT
    match_no,
    first_team,
    second_team,
    1_poss AS first_team_possession,
    2_poss AS second_team_possession,
    1_goals AS first_team_goals,
    2_goals AS second_team_goals,
    (1_goals / 1_poss) * 100 AS first_team_goal_conversion_rate,
    (2_goals / 2_poss) * 100 AS second_team_goal_conversion_rate
FROM match_data
WHERE venue = 'Al Janoub Stadium';
"""

# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    matches_at_al_janoub = result.fetchall()

# Fetch column names
```

```
column_names = result.keys()

# Print the performance data for matches at Al Janoub Stadium
print("Team Performance at Al Janoub Stadium:")
for match in matches_at_al_janoub:
    print("\nMatch Details:")
    for column, value in zip(column_names, match):
        print(f"{column}: {value}")
```

Team Performance at Al Janoub Stadium:

Match Details:
match_no: 8
first_team: FRANCE
second_team: AUSTRALIA
first_team_possession: 62
second_team_possession: 38
first_team_goals: 4
second_team_goals: 1
first_team_goal_conversion_rate: 6.4516
second_team_goal_conversion_rate: 2.6316

Match Details:
match_no: 13
first_team: SWITZERLAND
second_team: CAMEROON
first_team_possession: 51
second_team_possession: 49
first_team_goals: 1
second_team_goals: 0
first_team_goal_conversion_rate: 1.9608
second_team_goal_conversion_rate: 0.0000

Match Details:
match_no: 21
first_team: TUNISIA
second_team: AUSTRALIA
first_team_possession: 59
second_team_possession: 41
first_team_goals: 0
second_team_goals: 1
first_team_goal_conversion_rate: 0.0000
second_team_goal_conversion_rate: 2.4390

Match Details:
match_no: 29
first_team: CAMEROON
second_team: SERBIA
first_team_possession: 41
second_team_possession: 59
first_team_goals: 3

```
second_team_goals: 3
first_team_goal_conversion_rate: 7.3171
second_team_goal_conversion_rate: 5.0847

Match Details:
match_no: 37
first_team: AUSTRALIA
second_team: DENMARK
first_team_possession: 32
second_team_possession: 68
first_team_goals: 1
second_team_goals: 0
first_team_goal_conversion_rate: 3.1250
second_team_goal_conversion_rate: 0.0000

Match Details:
match_no: 45
first_team: GHANA
second_team: URUGUAY
first_team_possession: 51
second_team_possession: 49
first_team_goals: 0
second_team_goals: 2
first_team_goal_conversion_rate: 0.0000
second_team_goal_conversion_rate: 4.0816

Match Details:
match_no: 53
first_team: JAPAN
second_team: CROATIA
first_team_possession: 42
second_team_possession: 58
first_team_goals: 1
second_team_goals: 1
first_team_goal_conversion_rate: 2.3810
second_team_goal_conversion_rate: 1.7241
```

Task 11: Analyzing Penalty Area Goals at Different Venues.

Our data expedition now takes us to the heart of goal-scoring dynamics within various venues, uncovering the nuances of where and how goals are scored. We're poised to unravel the patterns of goal-scoring, distinguishing between goals scored within the

penalty area and those from outside. This data will paint a vivid picture of the venues where precision inside the box or long-range mastery dominates.

In [ ]:
```
# SQL query to get total goals inside and outside penalty area by venue, sorted by total goals
query = """
SELECT
    venue,
    SUM(1_goal_inside_penalty_area + 2_goal_inside_penalty_area) AS total_goals_inside_penalty_area,
    SUM(1_goal_outside_penalty_area + 2_goal_outside_penalty_area) AS total_goals_outside_penalty_area,
    SUM(1_goal_inside_penalty_area + 2_goal_inside_penalty_area + 1_goal_outside_penalty_area + 2_goal_outside_penalt
FROM match_data
GROUP BY venue
ORDER BY total_goals DESC;
"""

# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    goals_by_venue = result.fetchall()

# Fetch column names
column_names = result.keys()

# Print the goals by venue data
print("Total Goals Inside and Outside Penalty Area by Venue:")
for row in goals_by_venue:
    print("")
    for column, value in zip(column_names, row):
        print(f"{column}: {value}")
```

Total Goals Inside and Outside Penalty Area by Venue:

venue: Lusail Iconic Stadium
total_goals_inside_penalty_area: 30
total_goals_outside_penalty_area: 2
total_goals: 32

venue: Khalifa International Stadium
total_goals_inside_penalty_area: 27
total_goals_outside_penalty_area: 3
total_goals: 30

venue: Stadium 974
total_goals_inside_penalty_area: 20
total_goals_outside_penalty_area: 2
total_goals: 22

venue: Al Bayt Stadium
total_goals_inside_penalty_area: 20
total_goals_outside_penalty_area: 1
total_goals: 21

venue: Al Thumama Stadium
total_goals_inside_penalty_area: 21
total_goals_outside_penalty_area: 0
total_goals: 21

venue: Al Janoub Stadium
total_goals_inside_penalty_area: 17
total_goals_outside_penalty_area: 1
total_goals: 18

venue: Ahmed bin Ali Stadium
total_goals_inside_penalty_area: 10
total_goals_outside_penalty_area: 3
total_goals: 13

venue: Education City Stadium
total_goals_inside_penalty_area: 12
total_goals_outside_penalty_area: 1
total_goals: 13

Task 12: Comparing Goal Success Rates.

Our data voyage now leads us to a profound exploration of goal-scoring efficiency across various venues. We are about to uncover the nuanced story of goal-scoring precision within different venues. This data will reveal which venues favor inside-the-box finesse and which embrace long-range accuracy, providing a captivating glimpse into the strategic variations of the game.

In [ ]:
```
# SQL query to compare goal success rates at different venues
query = """
SELECT
    venue,
    SUM(1_goal_inside_penalty_area + 2_goal_inside_penalty_area) AS total_goals_inside_penalty_area,
    SUM(1_goal_outside_penalty_area + 2_goal_outside_penalty_area) AS total_goals_outside_penalty_area,
    (SUM(1_goal_inside_penalty_area + 2_goal_inside_penalty_area) /
     (SUM(1_goal_inside_penalty_area + 2_goal_inside_penalty_area) + SUM(1_goal_outside_penalty_area + 2_goal_outside
    (SUM(1_goal_outside_penalty_area + 2_goal_outside_penalty_area) /
     (SUM(1_goal_inside_penalty_area + 2_goal_inside_penalty_area) + SUM(1_goal_outside_penalty_area + 2_goal_outside
FROM match_data
GROUP BY venue
ORDER BY inside_penalty_area_goal_success_rate DESC, outside_penalty_area_goal_success_rate DESC;
"""

# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    goal_success_rate_data = result.fetchall()

# Fetch column names
column_names = result.keys()

# Print the goal success rates by venue
print("Goal Success Rates at Different Venues:")
for row in goal_success_rate_data:
    print("")
    for column, value in zip(column_names, row):
        print(f"{column}: {value}")
```

Goal Success Rates at Different Venues:

venue: Al Thumama Stadium
total_goals_inside_penalty_area: 21
total_goals_outside_penalty_area: 0
inside_penalty_area_goal_success_rate: 100.0000
outside_penalty_area_goal_success_rate: 0.0000

venue: Al Bayt Stadium
total_goals_inside_penalty_area: 20
total_goals_outside_penalty_area: 1
inside_penalty_area_goal_success_rate: 95.2381
outside_penalty_area_goal_success_rate: 4.7619

venue: Al Janoub Stadium
total_goals_inside_penalty_area: 17
total_goals_outside_penalty_area: 1
inside_penalty_area_goal_success_rate: 94.4444
outside_penalty_area_goal_success_rate: 5.5556

venue: Lusail Iconic Stadium
total_goals_inside_penalty_area: 30
total_goals_outside_penalty_area: 2
inside_penalty_area_goal_success_rate: 93.7500
outside_penalty_area_goal_success_rate: 6.2500

venue: Education City Stadium
total_goals_inside_penalty_area: 12
total_goals_outside_penalty_area: 1
inside_penalty_area_goal_success_rate: 92.3077
outside_penalty_area_goal_success_rate: 7.6923

venue: Stadium 974
total_goals_inside_penalty_area: 20
total_goals_outside_penalty_area: 2
inside_penalty_area_goal_success_rate: 90.9091
outside_penalty_area_goal_success_rate: 9.0909

venue: Khalifa International Stadium
total_goals_inside_penalty_area: 27
total_goals_outside_penalty_area: 3
inside_penalty_area_goal_success_rate: 90.0000

```
outside_penalty_area_goal_success_rate: 10.0000

venue: Ahmed bin Ali Stadium
total_goals_inside_penalty_area: 10
total_goals_outside_penalty_area: 3
inside_penalty_area_goal_success_rate: 76.9231
outside_penalty_area_goal_success_rate: 23.0769
```

Task 13: Stored Procedure for Extracting Match Data for Two Teams.

In our quest for deeper sports insights, we've taken a significant step by creating a custom procedure named getmatchdata2. This procedure, powered by the database, enables us to retrieve specific match data for any given pair of teams. With this procedure in place, we can now uncover the detailed statistics of matches between any two teams in our database. Whether it's venue insights, scoring dynamics, possession statistics, penalty shootout outcomes, or defensive pressures faced, this custom procedure equips us with the tools to dive deep into the intricacies of individual matchups.

Stay tuned as we harness the power of this procedure to unravel the unique stories behind each clash between teams, shedding light on the tactics, strategies, and outcomes that define these sporting encounters.

In [ ]:
```
# SQL to create the stored procedure
create_procedure_sql = """
CREATE PROCEDURE getmatchdata2(IN team1 VARCHAR(100), IN team2 VARCHAR(100))
BEGIN
    SELECT
        match_no,
        day_of_week,
        date,
        hour,
        venue,
        referee,
        `group`,
        first_team,
        second_team,
        attendance,
        1_poss AS first_team_possession,
        2_poss AS second_team_possession,
        1_goals AS first_team_goals,
        2_goals AS second_team_goals,
        1_attempts AS first_team_attempts,
```

```
        2_attempts AS second_team_attempts,
        1_goal_inside_penalty_area AS first_team_goals_inside_penalty_area,
        2_goal_inside_penalty_area AS second_team_goals_inside_penalty_area,
        1_goal_outside_penalty_area AS first_team_goals_outside_penalty_area,
        2_goal_outside_penalty_area AS second_team_goals_outside_penalty_area,
        1_ontarget AS first_team_ontarget_shots,
        2_ontarget AS second_team_ontarget_shots,
        1_offtarget AS first_team_offtarget_shots,
        2_offtarget AS second_team_offtarget_shots,
        1_yellow_cards AS first_team_yellow_cards,
        2_yellow_cards AS second_team_yellow_cards,
        1_red_cards AS first_team_red_cards,
        2_red_cards AS second_team_red_cards,
        faul_against_1 AS fouls_against_first_team,
        faul_against_2 AS fouls_against_second_team,
        1_offsides AS first_team_offsides,
        2_offsides AS second_team_offsides,
        1_passes AS first_team_passes,
        2_passes AS second_team_passes,
        1_passes_compeletd AS first_team_passes_completed,
        2_passes_compeletd AS second_team_passes_completed,
        1_corners AS first_team_corners,
        2_corners AS second_team_corners,
        1_free_kicks AS first_team_free_kicks,
        2_free_kicks AS second_team_free_kicks,
        1_panelties_scored AS first_team_penalties_scored,
        2_panelties_scored AS second_team_penalties_scored,
        1_goal_prevented AS first_team_goal_prevention,
        2_goal_prevented AS second_team_goal_prevention,
        1_own_goal AS first_team_own_goal,
        2_own_goal AS second_team_own_goal,
        1_forced_turnovers AS first_team_forced_turnovers,
        2_forced_turnovers AS second_team_forced_turnovers,
        1_defensive_pressure_applied AS first_team_defensive_pressure,
        2_defensive_pressure_applied AS second_team_defensive_pressure
    FROM match_data
    WHERE (first_team = team1 AND second_team = team2) OR (first_team = team2 AND second_team = team1);
END;
"""

# Execute the SQL to create the stored procedure
```

```
with db_engine.connect() as connection:
    connection.execute(text(create_procedure_sql))
```

Task 14: Calling the Created Procedure.

We're about to reveal the intricate details of a match that pitted these two giants against each other. This data will illuminate the venue where this electrifying showdown took place, the score that kept fans on the edge of their seats, possession dynamics, penalty shootout outcomes, and the defensive pressures faced by both teams.

In [ ]:
```
# SQL command to call the stored procedure
procedure_call_sql = """
CALL getmatchdata2('ARGENTINA', 'FRANCE');
"""

# Execute the procedure and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(procedure_call_sql))
    match_data = result.fetchall()

# Fetch column names
column_names = result.keys()

# Print the match data for the specified teams
print("Match Data for ARGENTINA vs FRANCE:")
for row in match_data:
    print("")
    for column, value in zip(column_names, row):
        print(f"{column}: {value}")
```

Match Data for ARGENTINA vs FRANCE:

match_no: 64
day_of_week: Sun
date: 2022-12-18
hour: 16:00
venue: Lusail Iconic Stadium
referee: Szymon Marciniak
group: Final
first_team: ARGENTINA
second_team: FRANCE
attendance: 88966
first_team_possession: 54
second_team_possession: 46
first_team_goals: 3
second_team_goals: 3
first_team_attempts: 21
second_team_attempts: 10
first_team_goals_inside_penalty_area: 3
second_team_goals_inside_penalty_area: 3
first_team_goals_outside_penalty_area: 0
second_team_goals_outside_penalty_area: 0
first_team_ontarget_shots: 9
second_team_ontarget_shots: 5
first_team_offtarget_shots: 9
second_team_offtarget_shots: 3
first_team_yellow_cards: 4
second_team_yellow_cards: 3
first_team_red_cards: 0
second_team_red_cards: 0
fouls_against_first_team: 26
fouls_against_second_team: 19
first_team_offsides: 4
second_team_offsides: 4
first_team_passes: 648
second_team_passes: 516
first_team_passes_completed: 544
second_team_passes_completed: 419
first_team_corners: 6
second_team_corners: 5
first_team_free_kicks: 22
second_team_free_kicks: 28

```
first_team_penalties_scored: 1
second_team_penalties_scored: 2
first_team_goal_prevention: 11
second_team_goal_prevention: 21
first_team_own_goal: 0
second_team_own_goal: 0
first_team_forced_turnovers: 87
second_team_forced_turnovers: 104
first_team_defensive_pressure: 280
second_team_defensive_pressure: 409
```

Module 3

Task 1: Analyzing Player Performance Criteria for England.

Our journey through the world of football data now zooms in on the dynamic players representing the Portuguese team. We're about to explore the profiles of these exceptional talents. Their limited starts belie their impact on the field, as evidenced by their goal-scoring ability and their involvement in different thirds of the pitch.

In [ ]:
```sql
# SQL query to analyze player performance criteria for England
query = """
SELECT
    ps.player,
    ps.team,
    ps.games_starts,
    ps.goals,
    pp.touches_def_3rd,
    pp.touches_mid_3rd,
    pp.touches_att_3rd
FROM
    player_stats ps
JOIN
    player_possession pp ON ps.player = pp.player
WHERE
    ps.team = 'England'
ORDER BY
    ps.goals DESC, ps.games_starts ASC;
"""
```

```python
# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    portugal_players = result.fetchall()

# Fetch column names
column_names = result.keys()

# Print the player data
print("English Players Performance:")
for row in portugal_players:
    print("")
    for column, value in zip(column_names, row):
        print(f"{column}: {value}")
```

English Players Performance:

player: Marcus Rashford
team: England
games_starts: 1
goals: 3
touches_def_3rd: 6.0
touches_mid_3rd: 43.0
touches_att_3rd: 44.0

player: Bukayo Saka
team: England
games_starts: 4
goals: 3
touches_def_3rd: 17.0
touches_mid_3rd: 48.0
touches_att_3rd: 83.0

player: Harry Kane
team: England
games_starts: 5
goals: 2
touches_def_3rd: 8.0
touches_mid_3rd: 61.0
touches_att_3rd: 71.0

player: Jack Grealish
team: England
games_starts: 0
goals: 1
touches_def_3rd: 12.0
touches_mid_3rd: 34.0
touches_att_3rd: 26.0

player: Raheem Sterling
team: England
games_starts: 2
goals: 1
touches_def_3rd: 5.0
touches_mid_3rd: 37.0
touches_att_3rd: 28.0

```
player: Jordan Henderson
team: England
games_starts: 3
goals: 1
touches_def_3rd: 19.0
touches_mid_3rd: 100.0
touches_att_3rd: 61.0

player: Phil Foden
team: England
games_starts: 3
goals: 1
touches_def_3rd: 8.0
touches_mid_3rd: 55.0
touches_att_3rd: 81.0

player: Jude Bellingham
team: England
games_starts: 5
goals: 1
touches_def_3rd: 44.0
touches_mid_3rd: 203.0
touches_att_3rd: 83.0

player: Callum Wilson
team: England
games_starts: 0
goals: 0
touches_def_3rd: 2.0
touches_mid_3rd: 8.0
touches_att_3rd: 7.0

player: Eric Dier
team: England
games_starts: 0
goals: 0
touches_def_3rd: 22.0
touches_mid_3rd: 23.0
touches_att_3rd: 0.0

player: Kalvin Phillips
team: England
```

games_starts: 0
goals: 0
touches_def_3rd: 8.0
touches_mid_3rd: 12.0
touches_att_3rd: 0.0

player: Trent Alexander-Arnold
team: England
games_starts: 0
goals: 0
touches_def_3rd: 10.0
touches_mid_3rd: 12.0
touches_att_3rd: 2.0

player: Kieran Trippier
team: England
games_starts: 2
goals: 0
touches_def_3rd: 53.0
touches_mid_3rd: 111.0
touches_att_3rd: 45.0

player: Mason Mount
team: England
games_starts: 2
goals: 0
touches_def_3rd: 5.0
touches_mid_3rd: 48.0
touches_att_3rd: 39.0

player: Kyle Walker
team: England
games_starts: 3
goals: 0
touches_def_3rd: 54.0
touches_mid_3rd: 120.0
touches_att_3rd: 27.0

player: Declan Rice
team: England
games_starts: 5
goals: 0

```
touches_def_3rd: 84.0
touches_mid_3rd: 247.0
touches_att_3rd: 34.0

player: Harry Maguire
team: England
games_starts: 5
goals: 0
touches_def_3rd: 204.0
touches_mid_3rd: 224.0
touches_att_3rd: 20.0

player: John Stones
team: England
games_starts: 5
goals: 0
touches_def_3rd: 209.0
touches_mid_3rd: 257.0
touches_att_3rd: 12.0

player: Jordan Pickford
team: England
games_starts: 5
goals: 0
touches_def_3rd: 169.0
touches_mid_3rd: 0.0
touches_att_3rd: 0.0

player: Luke Shaw
team: England
games_starts: 5
goals: 0
touches_def_3rd: 67.0
touches_mid_3rd: 248.0
touches_att_3rd: 107.0
```

Task 2: Analyzing Top 10 Club Dribbling Performance.

Our data journey now unveils an exciting competition of dribbling excellence among football clubs. We're about to embark on a thrilling journey into the world of dribbling supremacy. This data showcases the clubs whose players have displayed remarkable

dribbling skills, both in terms of the total number of successful dribbles and the highest individual dribbling achievement.

In [ ]:
```sql
# SQL query to analyze top 10 clubs by dribbling performance
query = """
SELECT
    ps.club,
    SUM(pp.dribbles_completed) AS total_dribbles_completed,
    MAX(pp.dribbles_completed) AS highest_individual_dribbles_completed
FROM
    player_stats ps
JOIN
    player_possession pp ON ps.player = pp.player
WHERE
    ps.club IS NOT NULL
    AND ps.club NOT IN ('1998', '1988')  -- Exclude incorrect entries
GROUP BY
    ps.club
ORDER BY
    total_dribbles_completed DESC, highest_individual_dribbles_completed DESC
LIMIT 10;
"""

# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    top_clubs = result.fetchall()

# Fetch column names
column_names = result.keys()

# Print the top 10 clubs data
print("Top 10 Clubs by Dribbling Performance:")
for row in top_clubs:
    print("")
    for column, value in zip(column_names, row):
        print(f"{column}: {value}")
```

Top 10 Clubs by Dribbling Performance:

club: Paris S-G
total_dribbles_completed: 53.0
highest_individual_dribbles_completed: 25.0

club: Bayern Munich
total_dribbles_completed: 44.0
highest_individual_dribbles_completed: 19.0

club: Barcelona
total_dribbles_completed: 36.0
highest_individual_dribbles_completed: 9.0

club: Chelsea
total_dribbles_completed: 31.0
highest_individual_dribbles_completed: 12.0

club: Angers
total_dribbles_completed: 24.0
highest_individual_dribbles_completed: 14.0

club: Real Madrid
total_dribbles_completed: 24.0
highest_individual_dribbles_completed: 6.0

club: Ajax
total_dribbles_completed: 22.0
highest_individual_dribbles_completed: 10.0

club: Manchester City
total_dribbles_completed: 21.0
highest_individual_dribbles_completed: 4.0

club: Tottenham
total_dribbles_completed: 20.0
highest_individual_dribbles_completed: 6.0

club: Manchester Utd
total_dribbles_completed: 19.0
highest_individual_dribbles_completed: 5.0

Task 3: Analyzing Goal Scoring Performance of Players Under 25.

Our data exploration now shifts to the young talents in the world of football, those players who have already left their mark on the game despite being under 25 years of age. We're about to uncover the remarkable achievements of these young stars. This data will not only reveal their goal-scoring prowess but also the percentage of goals they contribute compared to their more experienced counterparts.

In [ ]:
```
# SQL query to analyze goal-scoring performance of players under 25 compared to players 25 and older
query = """
WITH under_25_goals AS (
    SELECT
        SUM(goals) AS total_goals_under_25
    FROM
        player_stats
    WHERE
        age < 25
        AND goals > 0
),
over_25_goals AS (
    SELECT
        SUM(goals) AS total_goals_over_25
    FROM
        player_stats
    WHERE
        age >= 25
        AND goals > 0
)
SELECT
    under_25.total_goals_under_25,
    over_25.total_goals_over_25,
    (under_25.total_goals_under_25 / (under_25.total_goals_under_25 + over_25.total_goals_over_25) * 100) AS under_25
    (over_25.total_goals_over_25 / (under_25.total_goals_under_25 + over_25.total_goals_over_25) * 100) AS over_25_go
FROM
    under_25_goals under_25,
    over_25_goals over_25;
"""

# Execute the query and fetch the results
with db_engine.connect() as connection:
```

```
    result = connection.execute(text(query))
    goal_comparison = result.fetchall()

# Fetch column names
column_names = result.keys()

# Print the goal-scoring comparison
print("Goal-Scoring Performance Comparison:")
for row in goal_comparison:
    for column, value in zip(column_names, row):
        print(f"{column}: {value}")
```

```
Goal-Scoring Performance Comparison:
total_goals_under_25: 61
total_goals_over_25: 103
under_25_goal_percentage: 37.1951
over_25_goal_percentage: 62.8049
```

Task 4: Top 5 Clubs with Most Players Under 25.

Our data exploration now zooms in on the clubs that have become hotbeds of young talent, nurturing and developing the next generation of football stars. We are about to discover the football clubs that excel in fostering and harnessing the potential of players under 25. This data will unveil the clubs with the highest number of young talents, highlighting their commitment to youth development.

In [ ]:
```
# SQL query to find the top 5 clubs with the most players under 25
query = """
SELECT
    ps.club,
    COUNT(ps.player) AS number_of_under_25_players
FROM
    player_stats ps
WHERE
    ps.age < 25
    AND ps.club NOT IN ('1997')  -- Exclude incorrect entries
GROUP BY
    ps.club
ORDER BY
    number_of_under_25_players DESC
LIMIT 5;
```

```
"""

# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    top_young_talent_clubs = result.fetchall()

# Fetch column names
column_names = result.keys()

# Print the top 5 clubs with the most players under 25
print("Top 5 Clubs with Most Players Under 25:")
for row in top_young_talent_clubs:
    print("")
    for column, value in zip(column_names, row):
        print(f"{column}: {value}")
```

```
Top 5 Clubs with Most Players Under 25:

club: Barcelona
number_of_under_25_players: 5

club: Bayern Munich
number_of_under_25_players: 5

club: Real Madrid
number_of_under_25_players: 5

club: Rennes
number_of_under_25_players: 4

club: Dortmund
number_of_under_25_players: 4
```

Module 4

Task 1: Top 10 Players with Longest Average Shot Distance.

Our data journey now leads us to the realm of precision and goal-scoring prowess on the football field. We're about to reveal the names of the top players whose shot accuracy and goal-scoring ability shine brightly. This data will highlight the players with the

longest average shot distance, showcasing their ability to find the target from a distance.

Stay tuned as we explore the stories of these remarkable athletes, where every shot becomes an artful attempt to make the net ripple. It's a journey into the world of precision, technique, and the thrill of witnessing long-range strikes that leave fans in awe and goalkeepers helpless.

In [ ]:
```
# SQL query to find the top 10 players with the longest average shot distance
query = """
SELECT
    ps.player,
    ps.team,
    ps.goals,
    ps.shots,
    ps.average_shot_distance
FROM
    player_shootings ps
WHERE
    ps.shots > 0
ORDER BY
    ps.average_shot_distance DESC
LIMIT 10;
"""

# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    top_long_distance_shooters = result.fetchall()

# Fetch column names
column_names = result.keys()

# Print the top 10 players with the longest average shot distance
print("Top 10 Players with Longest Average Shot Distance:")
for row in top_long_distance_shooters:
    print("")
    for column, value in zip(column_names, row):
        print(f"{column}: {value}")
```

Top 10 Players with Longest Average Shot Distance:

player: Gareth Bale
team: Wales
goals: 1
shots: 1
average_shot_distance: 46.0

player: Rodri
team: Spain
goals: 0
shots: 1
average_shot_distance: 36.0

player: Saud Abdulhamid
team: Saudi Arabia
goals: 0
shots: 1
average_shot_distance: 34.0

player: Abdelhamid Sabiri
team: Morocco
goals: 0
shots: 3
average_shot_distance: 34.0

player: Kwon Kyung-won
team: Korea Republic
goals: 0
shots: 1
average_shot_distance: 33.0

player: Edson Alvarez
team: Mexico
goals: 0
shots: 1
average_shot_distance: 33.0

player: Joao Palhinha
team: Portugal
goals: 0
shots: 1

```
average_shot_distance: 33.0

player: Alex Sandro
team: Brazil
goals: 0
shots: 1
average_shot_distance: 33.0

player: Federico Valverde
team: Uruguay
goals: 0
shots: 7
average_shot_distance: 33.0

player: Ferjani Sassi
team: Tunisia
goals: 0
shots: 1
average_shot_distance: 33.0
```

Task 2: Identifying High-Performing Players with Precision Shooting.

Our data exploration now takes a closer look at the players who not only have a keen eye for the goal but also a remarkable shot accuracy. We are about to uncover the names of players who are both prolific goal-scorers and incredibly precise in their shooting. This data showcases the individuals who combine their goal-scoring instincts with an exceptional ability to find the target.

In [ ]:
```sql
# SQL query to identify high-performing players with precision shooting
query = """
SELECT
    ps.player,
    ps.team,
    ps.goals,
    ps.shots,
    ps.shots_on_target,
    (ps.shots_on_target / ps.shots) * 100 AS shot_accuracy_percentage
FROM
    player_shootings ps
WHERE
    ps.shots > 0
    AND ps.goals > 0
```

```
    AND ps.shots_on_target <= ps.shots
ORDER BY
    ps.goals DESC, shot_accuracy_percentage DESC
LIMIT 10;
"""

# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    high_performers = result.fetchall()

# Fetch column names
column_names = result.keys()

# Print the high-performing players with precision shooting
print("High-Performing Players with Precision Shooting:")
for row in high_performers:
    print("")
    for column, value in zip(column_names, row):
        print(f"{column}: {value}")
```

High-Performing Players with Precision Shooting:

player: Kylian Mbappe
team: France
goals: 8
shots: 29
shots_on_target: 11.0
shot_accuracy_percentage: 37.93103448275862

player: Lionel Messi
team: Argentina
goals: 7
shots: 27
shots_on_target: 13.0
shot_accuracy_percentage: 48.148148148148145

player: Julian Alvarez
team: Argentina
goals: 4
shots: 11
shots_on_target: 8.0
shot_accuracy_percentage: 72.72727272727273

player: Olivier Giroud
team: France
goals: 4
shots: 16
shots_on_target: 6.0
shot_accuracy_percentage: 37.5

player: Bukayo Saka
team: England
goals: 3
shots: 7
shots_on_target: 5.0
shot_accuracy_percentage: 71.42857142857143

player: Goncalo Ramos
team: Portugal
goals: 3
shots: 7
shots_on_target: 5.0

```
shot_accuracy_percentage: 71.42857142857143

player: Alvaro Morata
team: Spain
goals: 3
shots: 8
shots_on_target: 5.0
shot_accuracy_percentage: 62.5

player: Cody Gakpo
team: Netherlands
goals: 3
shots: 5
shots_on_target: 3.0
shot_accuracy_percentage: 60.0

player: Marcus Rashford
team: England
goals: 3
shots: 11
shots_on_target: 6.0
shot_accuracy_percentage: 54.54545454545454

player: Richarlison
team: Brazil
goals: 3
shots: 8
shots_on_target: 4.0
shot_accuracy_percentage: 50.0
```

Task 3: Top 10 Clubs with Young and High-Performing Players.

Our data journey now delves into the dynamic landscape of young talents making a significant impact on the football pitch. We're about to uncover the clubs that are home to a remarkable assembly of young talents who meet specific criteria for their age, goal-scoring, and shooting accuracy. This data will highlight the clubs that nurture and showcase the potential of these emerging stars.

In [ ]:
```sql
# SQL query to find the top 10 clubs with young and high-performing players
query = """
SELECT
    ps.club,
```

```
    COUNT(ps.player) AS number_of_young_high_performers,
    AVG(ps.goals) AS average_goals,
    AVG((ps_shooting.shots_on_target / ps_shooting.shots) * 100) AS average_shot_accuracy
FROM
    player_stats ps
JOIN
    player_shootings ps_shooting ON ps.player = ps_shooting.player
WHERE
    ps.age < 25
    AND ps_shooting.goals > 0
    AND ps_shooting.shots > 0
    AND ps_shooting.shots_on_target <= ps_shooting.shots
GROUP BY
    ps.club
ORDER BY
    number_of_young_high_performers DESC, average_goals DESC, average_shot_accuracy DESC
LIMIT 10;
"""

# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    top_clubs_young_talents = result.fetchall()

# Fetch column names
column_names = result.keys()

# Print the top 10 clubs with young and high-performing players
print("Top 10 Clubs with Young and High-Performing Players:")
for row in top_clubs_young_talents:
    print("")
    for column, value in zip(column_names, row):
        print(f"{column}: {value}")
```

Top 10 Clubs with Young and High-Performing Players:

club: Manchester City
number_of_young_high_performers: 2
average_goals: 2.5000
average_shot_accuracy: 44.6969696969697

club: Benfica
number_of_young_high_performers: 2
average_goals: 2.0000
average_shot_accuracy: 54.464285714285715

club: Barcelona
number_of_young_high_performers: 2
average_goals: 1.5000
average_shot_accuracy: 66.66666666666666

club: Chelsea
number_of_young_high_performers: 2
average_goals: 1.5000
average_shot_accuracy: 57.77777777777778

club: RB Leipzig
number_of_young_high_performers: 2
average_goals: 1.0000
average_shot_accuracy: 50.0

club: Brighton
number_of_young_high_performers: 2
average_goals: 1.0000
average_shot_accuracy: 45.238095238095234

club: Real Madrid
number_of_young_high_performers: 2
average_goals: 1.0000
average_shot_accuracy: 38.888888888888886

club: Atl?tico Madrid
number_of_young_high_performers: 2
average_goals: 1.0000
average_shot_accuracy: 38.63636363636363

```
club: Arsenal
number_of_young_high_performers: 1
average_goals: 3.0000
average_shot_accuracy: 71.42857142857143

club: PSV Eindhoven
number_of_young_high_performers: 1
average_goals: 3.0000
average_shot_accuracy: 60.0
```

Task 4: Top 10 Teams with Young and High-Performing Players.

Our data exploration now takes us to the heart of football teams that are becoming incubators for young talent, nurturing the future stars of the sport. We're about to unveil the football teams that have embraced youth with open arms, where young players meeting specific criteria for their age, goal-scoring, and shooting accuracy are thriving. This data will highlight the teams that are building a future powered by emerging talents.

In [ ]:
```sql
# SQL query to find the top 10 teams with young and high-performing players
query = """
SELECT
    ps.team,
    COUNT(ps.player) AS number_of_young_high_performers,
    AVG(ps.goals) AS average_goals,
    AVG((ps_shooting.shots_on_target / ps_shooting.shots) * 100) AS average_shot_accuracy
FROM
    player_stats ps
JOIN
    player_shootings ps_shooting ON ps.player = ps_shooting.player
WHERE
    ps.age < 25
    AND ps_shooting.goals > 0
    AND ps_shooting.shots > 0
    AND ps_shooting.shots_on_target <= ps_shooting.shots
GROUP BY
    ps.team
ORDER BY
    number_of_young_high_performers DESC, average_goals DESC, average_shot_accuracy DESC
LIMIT 10;
"""
```

```python
# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    top_teams_young_talents = result.fetchall()

# Fetch column names
column_names = result.keys()

# Print the top 10 teams with young and high-performing players
print("Top 10 Teams with Young and High-Performing Players:")
for row in top_teams_young_talents:
    print("")
    for column, value in zip(column_names, row):
        print(f"{column}: {value}")
```

Top 10 Teams with Young and High-Performing Players:

team: Argentina
number_of_young_high_performers: 4
average_goals: 1.7500
average_shot_accuracy: 54.342532467532465

team: France
number_of_young_high_performers: 3
average_goals: 3.3333
average_shot_accuracy: 40.421455938697314

team: England
number_of_young_high_performers: 3
average_goals: 1.6667
average_shot_accuracy: 49.36507936507937

team: Spain
number_of_young_high_performers: 3
average_goals: 1.3333
average_shot_accuracy: 61.11111111111111

team: Ghana
number_of_young_high_performers: 3
average_goals: 1.3333
average_shot_accuracy: 44.444444444444436

team: United States
number_of_young_high_performers: 3
average_goals: 1.0000
average_shot_accuracy: 49.629629629629626

team: Portugal
number_of_young_high_performers: 2
average_goals: 2.0000
average_shot_accuracy: 49.35064935064935

team: Japan
number_of_young_high_performers: 2
average_goals: 1.5000
average_shot_accuracy: 87.5

```
team: Morocco
number_of_young_high_performers: 2
average_goals: 1.0000
average_shot_accuracy: 83.33333333333333

team: Serbia
number_of_young_high_performers: 2
average_goals: 1.0000
average_shot_accuracy: 75.0
```

Task 5: Analyzing Player Performance in Terms of Goals, Shots, and Assists.

Our journey into the realm of football statistics now unveils the players who have truly made their mark on the pitch. We're about to showcase the profiles of these remarkable athletes, where goals, assists, and shot accuracy converge to tell the story of their impact on the field. This data will reveal the top goal-scorers and playmakers, offering insights into their contributions to their respective teams.

In [ ]:
```sql
# SQL query to analyze player performance in terms of goals, shots, and assists
query = """
SELECT
    ps.player,
    ps.team,
    ps.goals,
    ps_shooting.shots,
    ps_shooting.shots_on_target,
    (ps_shooting.shots_on_target / ps_shooting.shots) * 100 AS shot_accuracy_percentage,
    ps.assists
FROM
    player_stats ps
JOIN
    player_shootings ps_shooting ON ps.player = ps_shooting.player
WHERE
    ps.goals > 0
    AND ps_shooting.shots > 0
ORDER BY
    ps.goals DESC, ps.assists DESC, shot_accuracy_percentage DESC
LIMIT 10;
"""

# Execute the query and fetch the results
```

```
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    top_performers = result.fetchall()

# Fetch column names
column_names = result.keys()

# Print the top players in terms of goals, shots, and assists
print("Top Players in Terms of Goals, Shots, and Assists:")
for row in top_performers:
    print("")
    for column, value in zip(column_names, row):
        print(f"{column}: {value}")
```

Top Players in Terms of Goals, Shots, and Assists:

player: Kylian Mbappe
team: France
goals: 8
shots: 29
shots_on_target: 11.0
shot_accuracy_percentage: 37.93103448275862
assists: 2

player: Lionel Messi
team: Argentina
goals: 7
shots: 27
shots_on_target: 13.0
shot_accuracy_percentage: 48.148148148148145
assists: 3

player: Julian Alvarez
team: Argentina
goals: 4
shots: 11
shots_on_target: 8.0
shot_accuracy_percentage: 72.72727272727273
assists: 0

player: Olivier Giroud
team: France
goals: 4
shots: 16
shots_on_target: 6.0
shot_accuracy_percentage: 37.5
assists: 0

player: Goncalo Ramos
team: Portugal
goals: 3
shots: 7
shots_on_target: 5.0
shot_accuracy_percentage: 71.42857142857143
assists: 1

```
player: Alvaro Morata
team: Spain
goals: 3
shots: 8
shots_on_target: 5.0
shot_accuracy_percentage: 62.5
assists: 1

player: Bukayo Saka
team: England
goals: 3
shots: 7
shots_on_target: 5.0
shot_accuracy_percentage: 71.42857142857143
assists: 0

player: Cody Gakpo
team: Netherlands
goals: 3
shots: 5
shots_on_target: 3.0
shot_accuracy_percentage: 60.0
assists: 0

player: Marcus Rashford
team: England
goals: 3
shots: 11
shots_on_target: 6.0
shot_accuracy_percentage: 54.54545454545454
assists: 0

player: Richarlison
team: Brazil
goals: 3
shots: 8
shots_on_target: 4.0
shot_accuracy_percentage: 50.0
assists: 0
```

Module 5

Task 1: Analyzing Player Performance in the Opponent's Penalty Area.

Our data exploration now brings us to the heart of the action on the football pitch, where players showcase their ability to make an impact in the opponent's penalty area. We're about to unveil the players who have a significant presence in the opposition's penalty area. This data will showcase their ability to create opportunities, apply pressure, and potentially find the back of the net in crucial moments.

In [ ]:
```
# SQL query to analyze player performance in the opponent's penalty area
query = """
SELECT
    ps.player,
    ps.team,
    pp.touches_att_pen_area AS touches_in_penalty_area,
    ps.goals,
    ps_shooting.shots,
    ps.assists
FROM
    player_stats ps
JOIN
    player_possession pp ON ps.player = pp.player
JOIN
    player_shootings ps_shooting ON ps.player = ps_shooting.player
WHERE
    pp.touches_att_pen_area > 0
ORDER BY
    pp.touches_att_pen_area DESC, ps.goals DESC, ps.assists DESC
LIMIT 10;
"""

# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    top_penalty_area_players = result.fetchall()

# Fetch column names
column_names = result.keys()

# Print the top players in the opponent's penalty area
print("Top Players in the Opponent's Penalty Area:")
```

```
for row in top_penalty_area_players:
    print("")
    for column, value in zip(column_names, row):
        print(f"{column}: {value}")
```

Top Players in the Opponent's Penalty Area:

player: Kylian Mbappe
team: France
touches_in_penalty_area: 61.0
goals: 8
shots: 29
assists: 2

player: Lionel Messi
team: Argentina
touches_in_penalty_area: 38.0
goals: 7
shots: 27
assists: 3

player: Ivan Perisic
team: Croatia
touches_in_penalty_area: 34.0
goals: 1
shots: 16
assists: 3

player: Jamal Musiala
team: Germany
touches_in_penalty_area: 28.0
goals: 0
shots: 12
assists: 1

player: Christian Pulisic
team: United States
touches_in_penalty_area: 23.0
goals: 1
shots: 9
assists: 2

player: Olivier Giroud
team: France
touches_in_penalty_area: 21.0
goals: 4
shots: 16

```
assists: 0

player: Serge Gnabry
team: Germany
touches_in_penalty_area: 21.0
goals: 1
shots: 12
assists: 1

player: Julian Alvarez
team: Argentina
touches_in_penalty_area: 20.0
goals: 4
shots: 11
assists: 0

player: Ismaila Sarr
team: Senegal
touches_in_penalty_area: 20.0
goals: 1
shots: 10
assists: 0

player: Raphinha
team: Brazil
touches_in_penalty_area: 20.0
goals: 0
shots: 8
assists: 1
```

Task 2: Count of Players with Touches in Opponent's Penalty Area by Club.

Our journey into the realm of football statistics now turns to the clubs that excel in creating opportunities and applying pressure in the opponent's penalty area. We're about to unveil the clubs that have a wealth of players capable of making their presence felt in the opposition's penalty area. This data will highlight the teams with a strategic advantage when it comes to attacking and creating goal-scoring opportunities.

In [ ]:
```
# SQL query to count players with touches in the opponent's penalty area by club
query = """
SELECT
```

```
    ps.club,
    COUNT(DISTINCT ps.player) AS number_of_players_in_penalty_area
FROM
    player_stats ps
JOIN
    player_possession pp ON ps.player = pp.player
WHERE
    pp.touches_att_pen_area > 0
GROUP BY
    ps.club
ORDER BY
    number_of_players_in_penalty_area DESC
LIMIT 10;
"""

# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    players_in_penalty_area_by_club = result.fetchall()

# Fetch column names
column_names = result.keys()

# Print the count of players with touches in the opponent's penalty area by club
print("Count of Players with Touches in Opponent's Penalty Area by Club:")
for row in players_in_penalty_area_by_club:
    print("")
    for column, value in zip(column_names, row):
        print(f"{column}: {value}")
```

```
Count of Players with Touches in Opponent's Penalty Area by Club:

club: Manchester Utd
number_of_players_in_penalty_area: 13

club: Barcelona
number_of_players_in_penalty_area: 13

club: Manchester City
number_of_players_in_penalty_area: 12

club: Bayern Munich
number_of_players_in_penalty_area: 12

club: Atl?tico Madrid
number_of_players_in_penalty_area: 9

club: Real Madrid
number_of_players_in_penalty_area: 9

club: Chelsea
number_of_players_in_penalty_area: 8

club: Juventus
number_of_players_in_penalty_area: 7

club: Tottenham
number_of_players_in_penalty_area: 7

club: Ajax
number_of_players_in_penalty_area: 7
```

Task 3: Average Player Touches in Different Field Areas.

Our journey into the intricate world of football statistics now leads us to the art of ball distribution and field presence. We're about to uncover the average touch patterns across different areas of the field. This data will paint a vivid picture of how players distribute the ball and position themselves in various zones, whether it's in their defensive penalty area, the midfield, or the attacking third.

In [ ]:
```
# SQL query to calculate average player touches and percentages in different field areas
query = """
```

```
SELECT
    AVG(pp.touches_def_pen_area) AS avg_touches_defensive_penalty_area,
    (AVG(pp.touches_def_pen_area) / AVG(pp.touches_def_pen_area + pp.touches_def_3rd + pp.touches_mid_3rd + pp.touche

    AVG(pp.touches_def_3rd) AS avg_touches_defensive_third,
    (AVG(pp.touches_def_3rd) / AVG(pp.touches_def_pen_area + pp.touches_def_3rd + pp.touches_mid_3rd + pp.touches_att

    AVG(pp.touches_mid_3rd) AS avg_touches_midfield_third,
    (AVG(pp.touches_mid_3rd) / AVG(pp.touches_def_pen_area + pp.touches_def_3rd + pp.touches_mid_3rd + pp.touches_att

    AVG(pp.touches_att_3rd) AS avg_touches_attacking_third,
    (AVG(pp.touches_att_3rd) / AVG(pp.touches_def_pen_area + pp.touches_def_3rd + pp.touches_mid_3rd + pp.touches_att

    AVG(pp.touches_att_pen_area) AS avg_touches_attacking_penalty_area,
    (AVG(pp.touches_att_pen_area) / AVG(pp.touches_def_pen_area + pp.touches_def_3rd + pp.touches_mid_3rd + pp.touche
FROM
    player_possession pp;
"""

# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    average_touches = result.fetchone()

# Fetch column names
column_names = result.keys()

# Print the average player touches and percentages in different field areas
print("Average Player Touches and Percentages in Different Field Areas:")
for column, value in zip(column_names, average_touches):
    print("")
    print(f"{column}: {value}")
```

```
Average Player Touches and Percentages in Different Field Areas:

avg_touches_defensive_penalty_area: 12.936484490398819

percent_touches_defensive_penalty_area: 9.418115731629943

avg_touches_defensive_third: 38.45790251107829

percent_touches_defensive_third: 27.998408448129386

avg_touches_midfield_third: 57.08862629246676

percent_touches_midfield_third: 41.56208665354711

avg_touches_attacking_third: 25.438700147710488

percent_touches_attacking_third: 18.520071834908755

avg_touches_attacking_penalty_area: 3.4357459379615953

percent_touches_attacking_penalty_area: 2.5013173317847963
```

Task 4: Count of Players with Goals by Position.

Our data journey now shifts its focus to the positions on the football field where players have displayed their goal-scoring prowess. We're about to uncover the goal-scoring tendencies of players across different positions. This data will highlight which positions are more likely to contribute to the team's goal tally and provide valuable insights into the versatility and attacking capabilities of players in each role.

In [ ]:
```sql
# SQL query to count players with goals by position
query = """
SELECT
    ps.position,
    COUNT(DISTINCT ps.player) AS number_of_goal_scorers
FROM
    player_stats ps
WHERE
    ps.goals > 0
    AND LENGTH(ps.position) < 3
GROUP BY
```

```
    ps.position
ORDER BY
    number_of_goal_scorers DESC;
"""

# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    goal_scorers_by_position = result.fetchall()

# Fetch column names
column_names = result.keys()

# Print the count of players with goals by position
print("Count of Players with Goals by Position:")
for row in goal_scorers_by_position:
    print("")
    for column, value in zip(column_names, row):
        print(f"{column}: {value}")
```

```
Count of Players with Goals by Position:

position: FW
number_of_goal_scorers: 61

position: MF
number_of_goal_scorers: 30

position: DF
number_of_goal_scorers: 20
```

Task 5: Top-Scoring Defender in the Dataset.

Our data exploration now zeroes in on the defenders who have made a significant impact in the goal-scoring department. We're about to unveil the top goal-scoring defender, the one who has shown that defenders can be a potent force in the attack as well. This data will showcase the exceptional ability of a defender to contribute to their team's goal tally.

In [ ]:
```
# SQL query to find the top-scoring defender
query = """
SELECT
```

```
    ps.player,
    ps.team,
    ps.goals
FROM
    player_stats ps
WHERE
    ps.position = 'DF'
ORDER BY
    ps.goals DESC
LIMIT 1;
"""

# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    top_scoring_defender = result.fetchone()

# Fetch column names
column_names = result.keys()

# Print the top-scoring defender
print("Top-Scoring Defender in the Dataset:")
for column, value in zip(column_names, top_scoring_defender):
    print(f"{column}: {value}")
```

```
Top-Scoring Defender in the Dataset:
player: Daley Blind
team: Netherlands
goals: 1
```

Task 6: Top-Scoring Midfielder.

Our data exploration now shifts to the midfield maestro who has excelled in the art of goal-scoring. We're about to unveil the top goal-scoring midfielder, the player who has proven that the midfield is not just about creating opportunities but also finishing them with finesse. This data will showcase the unique blend of creativity and goal-scoring ability possessed by this exceptional midfielder.

In [ ]:
```
# SQL query to find the top-scoring midfielder
query = """
SELECT
    ps.player,
```

```
    ps.team,
    ps.goals
FROM
    player_stats ps
WHERE
    ps.position = 'MF'
ORDER BY
    ps.goals DESC
LIMIT 1;
"""

# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    top_scoring_midfielder = result.fetchone()

# Fetch column names
column_names = result.keys()

# Print the top-scoring midfielder
print("Top-Scoring Midfielder in the Dataset:")
for column, value in zip(column_names, top_scoring_midfielder):
    print(f"{column}: {value}")
```

```
Top-Scoring Midfielder in the Dataset:
player: Giorgian De Arrascaeta
team: Uruguay
goals: 2
```

Task 7: Top-Scoring Forward.

Our data exploration now takes us to the frontlines of football, where goal-scoring prowess is paramount. We're about to unveil the top goal-scoring forward, the player who embodies the essence of attacking excellence. This data will showcase the striking abilities of this exceptional forward, who has the knack for finding the back of the net when it matters most.

In [ ]:
```
# SQL query to find the top-scoring forward
query = """
SELECT
    ps.player,
    ps.team,
```

```
    ps.goals
FROM
    player_stats ps
WHERE
    ps.position = 'FW'
ORDER BY
    ps.goals DESC
LIMIT 1;
"""

# Execute the query and fetch the results
with db_engine.connect() as connection:
    result = connection.execute(text(query))
    top_scoring_forward = result.fetchone()

# Fetch column names
column_names = result.keys()

# Print the top-scoring forward
print("Top-Scoring Forward in the Dataset:")
for column, value in zip(column_names, top_scoring_forward):
    print(f"{column}: {value}")
```

```
Top-Scoring Forward in the Dataset:
player: Kylian Mbappe
team: France
goals: 8
```