The Godfather of Talent | Futurense

Futurense Technologies

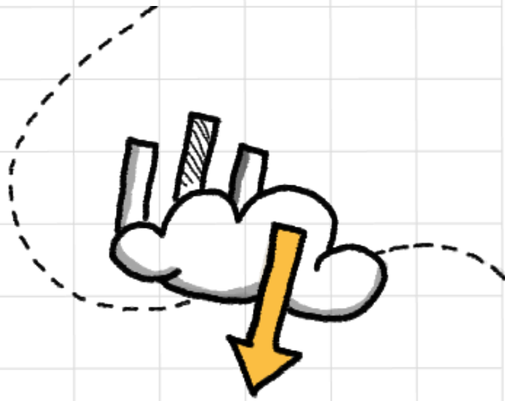
Project Report

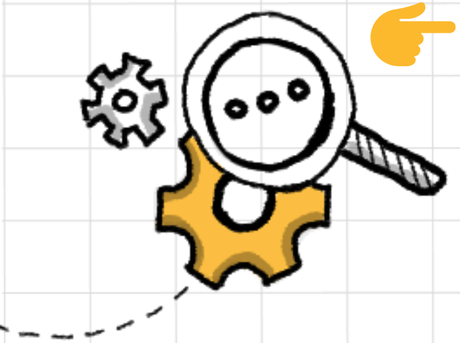
Group Name: Growth Gurus

1. Akansha Shetty 22BTRAD002
2. Chimirala Kowstubha 22BTRAD012
3. Kaparotu Venkata Surya Tharani 22BTRAD018

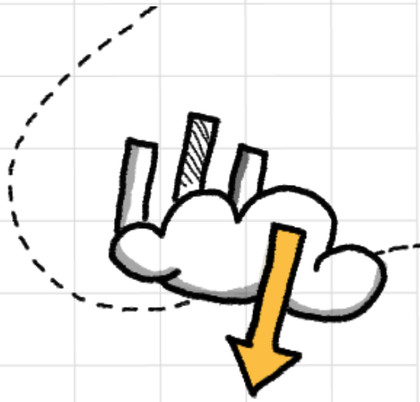
The Godfather of Talent | Futurense

Index





Introduction

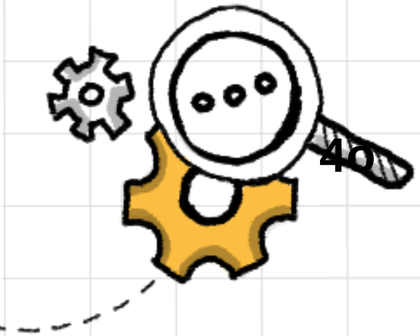


This project involves :

- Analysis of sports data for player performance insights.
- Identify factors influencing player performance.
- Support decision-making for sports franchises.

Nature of dataset:

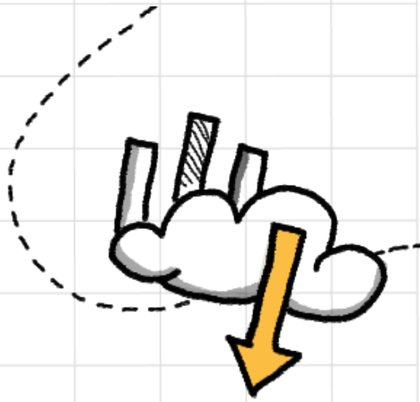
1. **Synthetic sports dataset.**
2. **Includes demographics, stats, injuries, training histories.**
3. **Contains performance metrics (goals, assists, tackles) over several seasons.**



The Godfather of Talent | Futurense

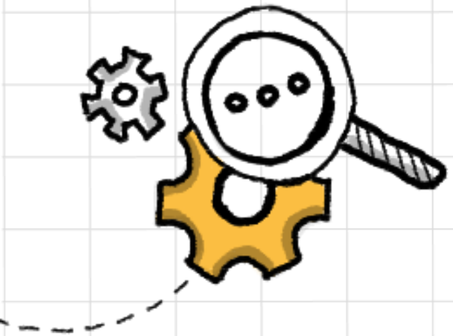
Problem Statement - 1

Data Cleaning and Augmentation



Task:

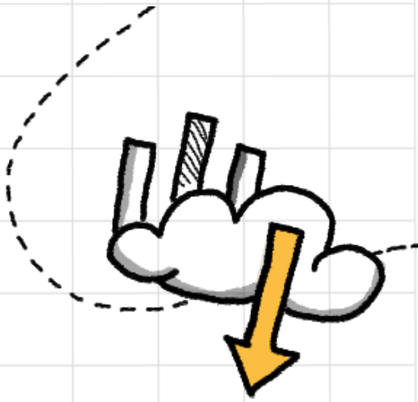
- Handle missing values with advanced imputation.
- Correct anomalies by outlier detection.
- Dealing with duplicates



Findings:

- More than 3500 null values are identified
- Nearly 3000 duplicates are present
- 4000 outliers are identified

Handling of Missing Values



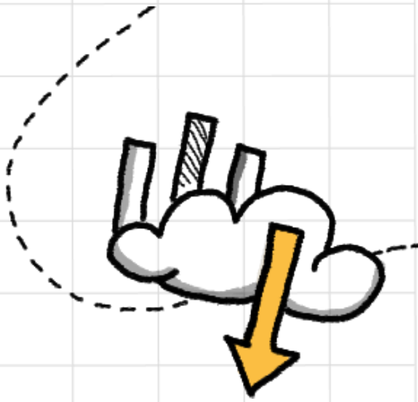
Using knnimputer

```
# imputing null values in height column using knn
feature_cols = df[['Age', 'Height']]
imputer = KNNImputer(n_neighbors=5)
imputed_values = imputer.fit_transform(feature_cols)
imputed_df = pd.DataFrame(imputed_values, columns=['Age', 'Height'])
df['Height'] = imputed_df['Height']
```

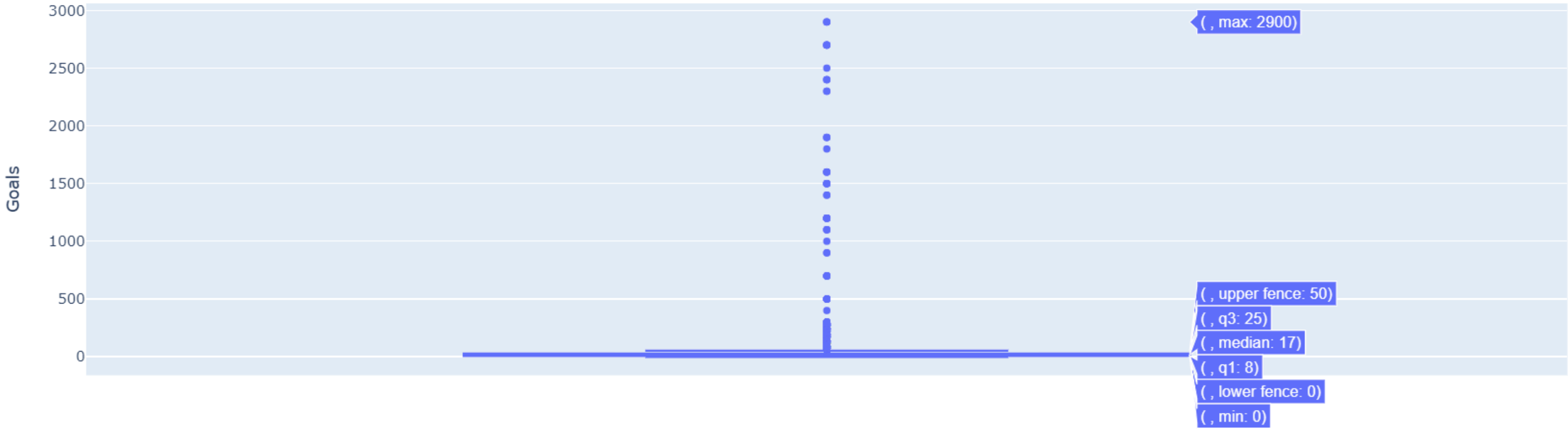
Using fillna

```
df['Goals'].fillna(0,inplace=True)
df['Assists'].fillna(0,inplace=True)
```

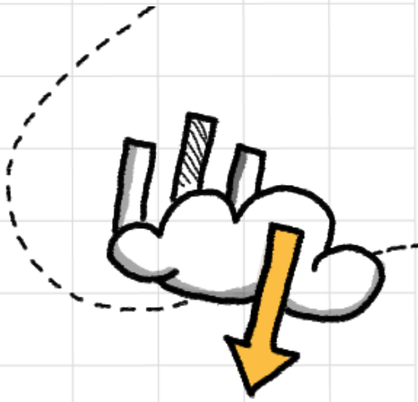
Visualization of Outliers



```
import plotly.express as px
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
df[numeric_cols] = df[numeric_cols].apply(pd.to_numeric, errors='coerce')
non_numeric_cols = df.columns.difference(numeric_cols)
numeric_df = df.drop(columns=non_numeric_cols)
for i in numeric_df.columns:
    fig = px.box(df, y=i)
    fig.show()
```



Handling of Outliers



Using Capping

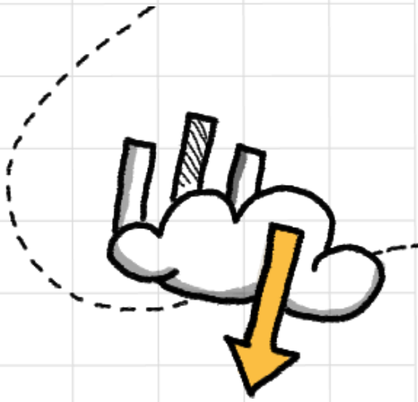
```
# Habdling outliers using capping for goals column
q1 = df["Goals"].quantile(0.25)
q3 = df["Goals"].quantile(0.75)
IQR=q3-q1
upper_limit = q3 + (IQR*1.5)
lower_limit = q1 - (IQR*1.5)
df["Goals"] = np.where(df["Goals"]> upper_limit, upper_limit,np.where(df["Goals"] < lower_limit,lower_limit,df["Goals"]))
```

Using Mean Value

```
# Handling outliers using mean for height and weight columns
def impute_outliers_IQR_mean(df):
    q1=df.quantile(0.25)
    q3=df.quantile(0.75)
    IQR=q3-q1
    upper = df[~(df>(q3+1.5*IQR))].max()
    lower = df[~(df<(q1-1.5*IQR))].min()
    df = np.where(df > upper, df.mean(), np.where(df < lower,df.mean(),df))
    return df

df['Height'] = impute_outliers_IQR_mean(df['Height'])
df['Weight'] = impute_outliers_IQR_mean(df['Weight'])
```

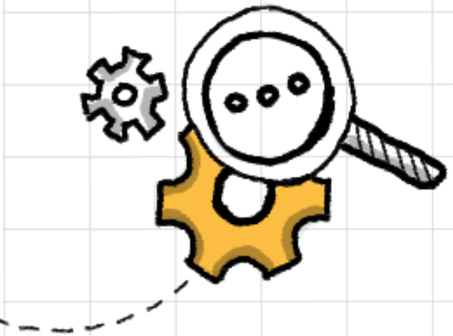
Problem Statement - 3

Data Ingestion Strategies



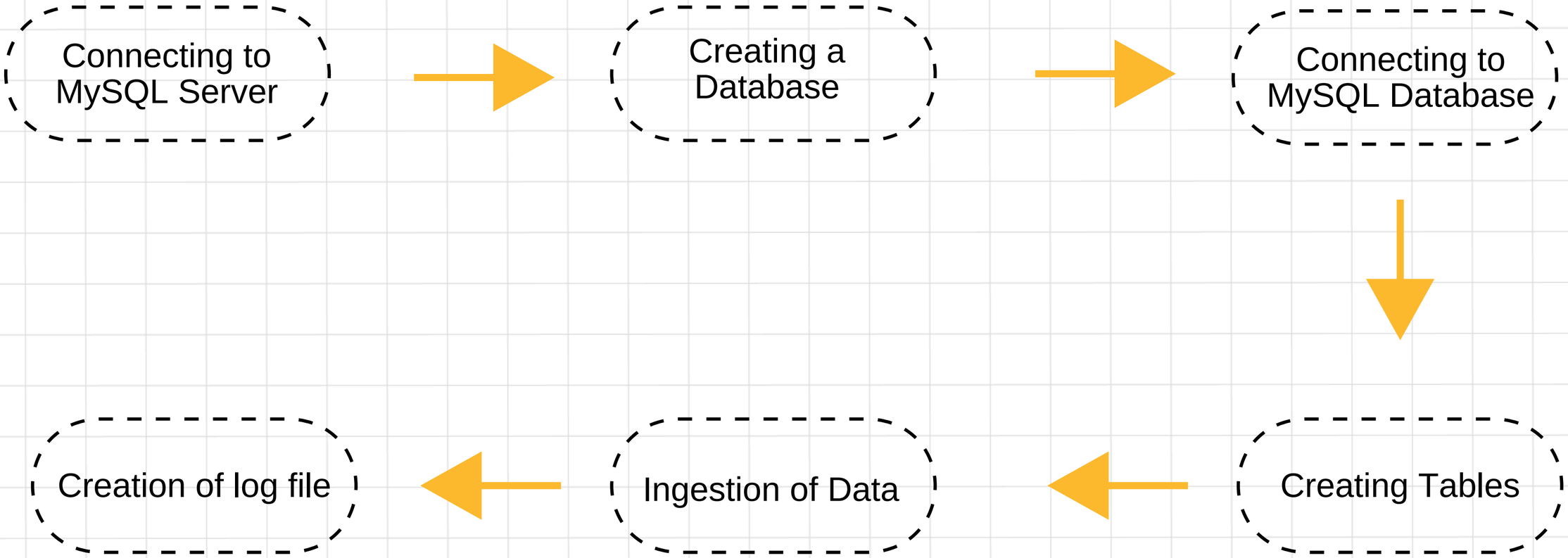
- Create an incremental data loading pipeline.
- Implement logging and monitoring for performance.
- Utilize Python, pandas, and SQL for implementation.

Findings:



1. Successful MySQL connection, database creation, and table setup enable data ingestion for sports analytics.
2. Ingestion process initiated, inserting player data into multiple tables for comprehensive analysis, with logging to track process status and errors.

Data Ingestion Strategies Outcome

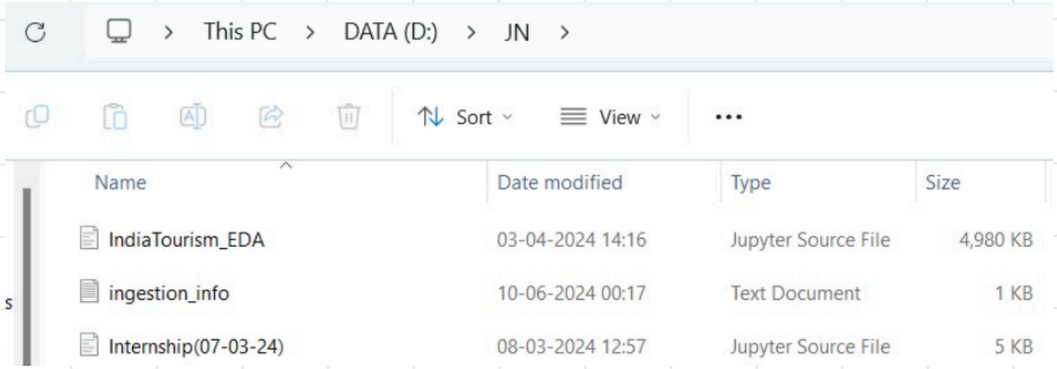


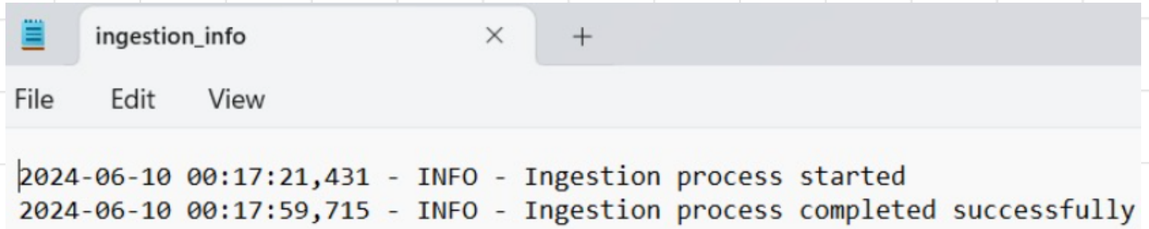
https://github.com/Kowstubha9/futurense-internship-phsae-2-capstone-project/blob/main/problem-statement_code-files/PS-3/PS-3.ipynb

Data Ingestion Strategies Outcome

Creation of log file

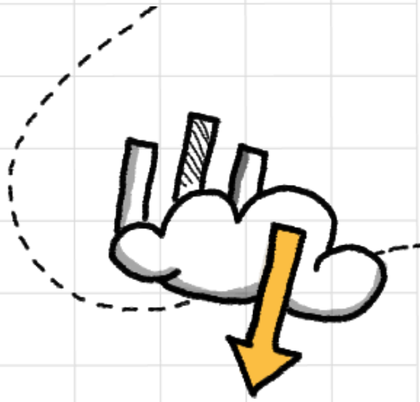
```
logging.basicConfig(filename='ingestion_info.log', level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
def perform_ingestion():
    try:
        # Inserting values into the database
        logging.info("Ingestion process started")
        insert_values(connection, "Playerinfo", data[['SerialID', 'Player', 'Team', 'Position', 'Season']])
        insert_values(connection, "Playerdemographics", data[['SerialID', 'Age', 'Height', 'Weight']])
        insert_values(connection, "Playerstats", data[['SerialID', 'Goals', 'Assists', 'YellowCards', 'RedCards', 'PassComp
        insert_values(connection, "Playertraininginfo", data[['SerialID', 'TrainingHours', 'EffectiveTraining']])
        insert_values(connection, "Playerperformance", data[['SerialID', 'PressurePerformanceImpact', 'MatchPressure']])
        insert_values(connection, "Playerhealth", data[['SerialID', 'InjuryHistory', 'PlayerFatigue', 'FatigueInjuryCorrela
        logging.info("Ingestion process completed successfully")
    except Exception as e:
        logging.error(f"Error during ingestion process: {str(e)}")

perform_ingestion()
```





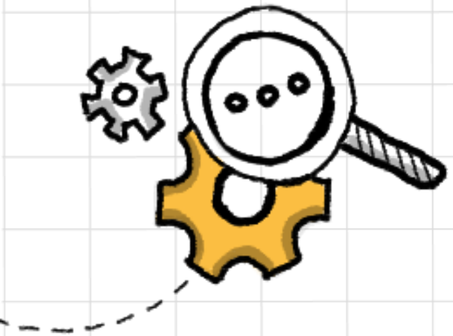
Problem Statement - 6

Data Warehousing



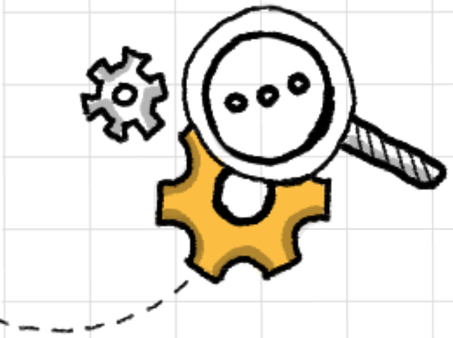
Tasks:

- Design and implement an efficient data warehouse schema.
- Ensure support for complex analytical queries.
- Implement robust data security and access controls.



Steps:

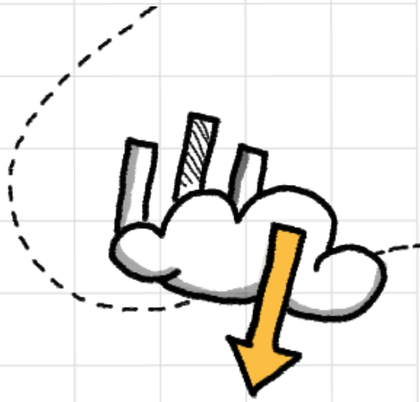
- Designing database schema
- Creation of database and tables
- Implementation of security and acceess mechanisms
- Creation database schema diagram by establishing cardinality relations



The Godfather of Talent | Futurense

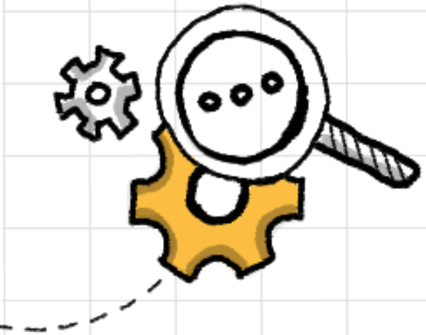
Problem Statement - 4

Pass Completion Rate vs. Assists



- Analyze pass completion rate vs. assists relationship.
- Create scatter plot, identify outliers using advanced methods.
- Plot line of best fit, perform regression analysis.
- Evaluate model using appropriate metrics.

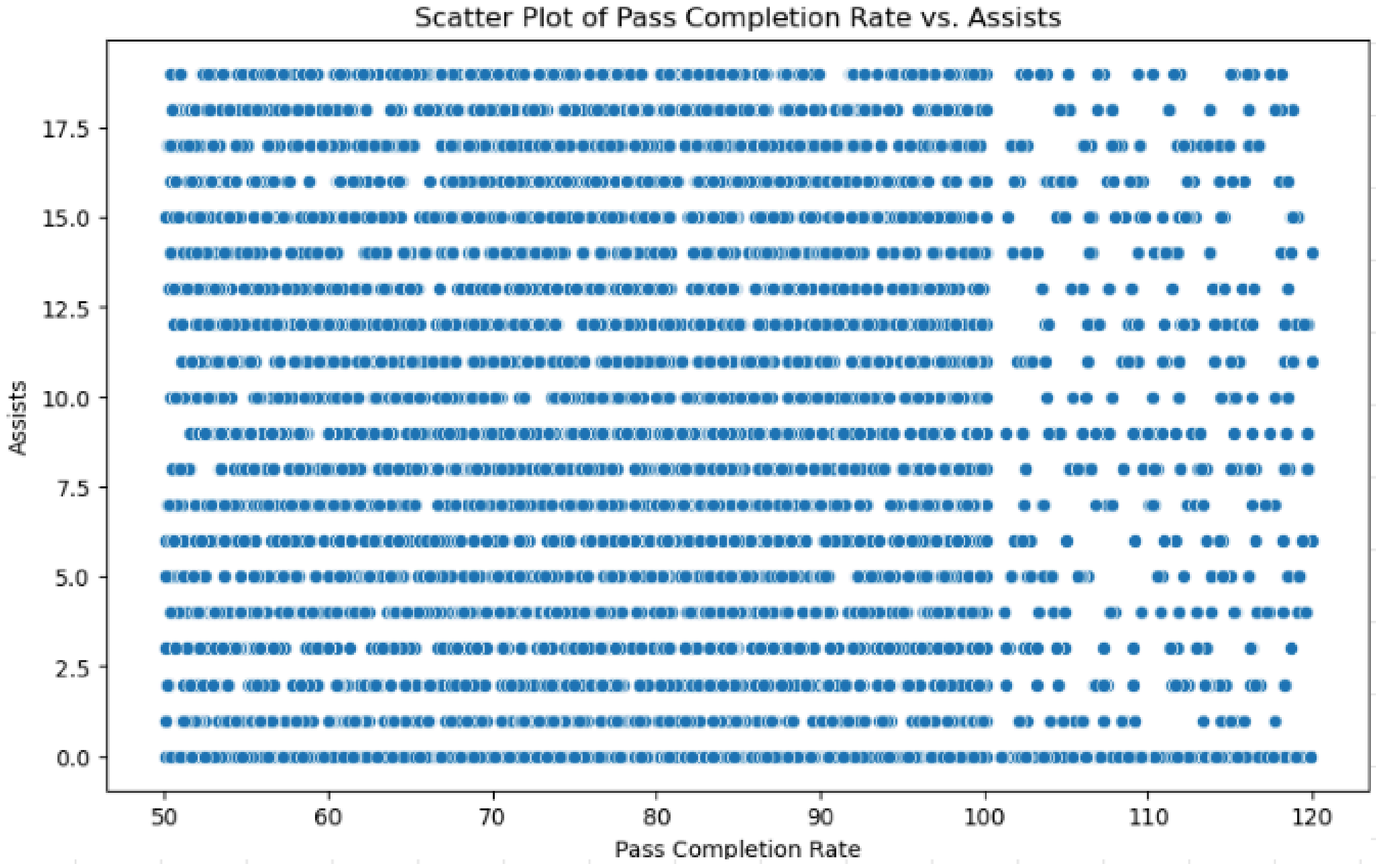
Findings:



1. Higher pass completion rates tend to coincide with more assists → accurate passing facilitates goal-scoring opportunities.
2. Non-Linear Relationship. **Pass completion doesn't guarantee assists**; varied factors like player position and opponent defense impact goals.

Pass Completion Rate vs. Assists Outcome

Scatter Plot

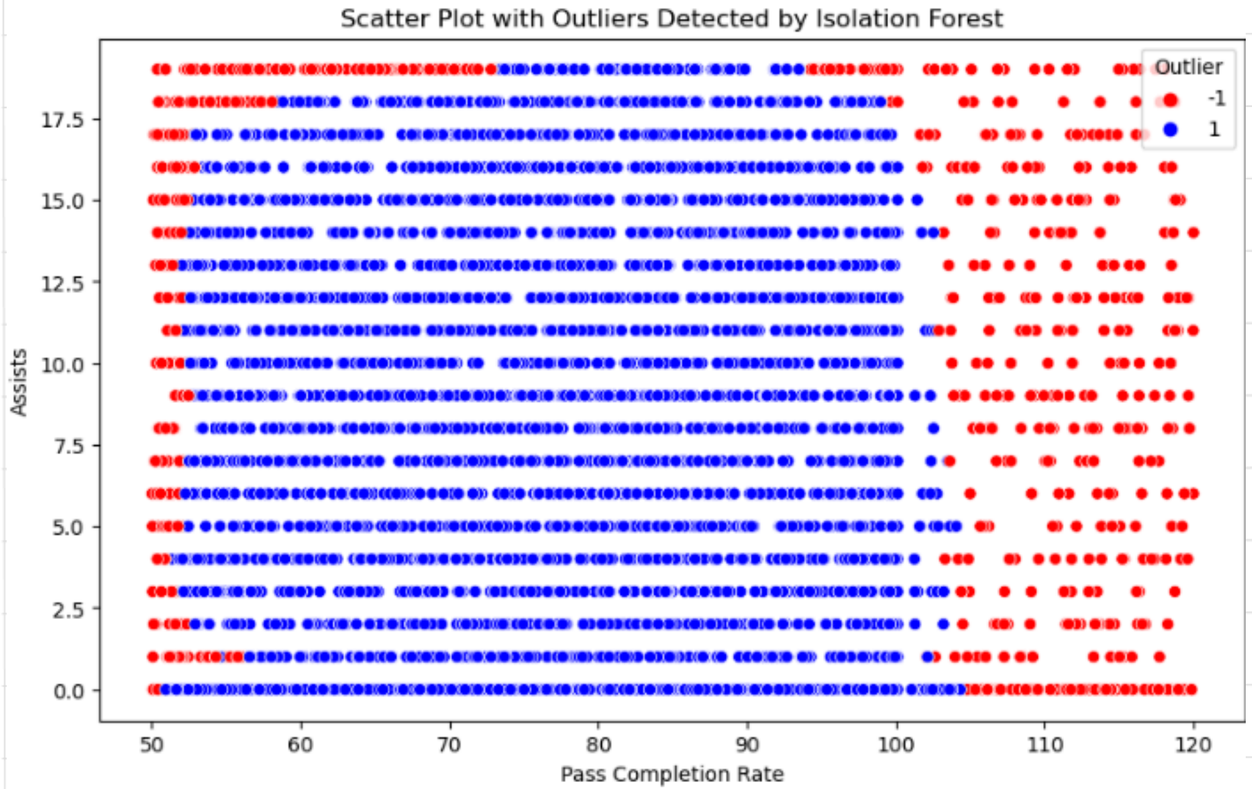



```python
from sklearn.linear_model import LinearRegression

# Set up the data for the regression
X = df[['PassCompletionRate']] # Our feature (independent variable)
y = df['Assists'] # Our target (dependent variable)

# Creating and fitting the linear regression model
model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)

# Plot the scatter plot with the regression line
plt.figure(figsize=(10, 6))
sns.scatterplot(x='PassCompletionRate', y='Assists', data=df) # Plot the original data
plt.plot(df['PassCompletionRate'], y_pred, color='red', linewidth=2) # Plot the regression line
plt.title('Regression Line: Pass Completion Rate vs. Assists')
plt.xlabel('Pass Completion Rate')
plt.ylabel('Assists')
plt.show()
```

Pass Completion Rate vs. Assists Outcome

Outlier Detection using Isolation Forest

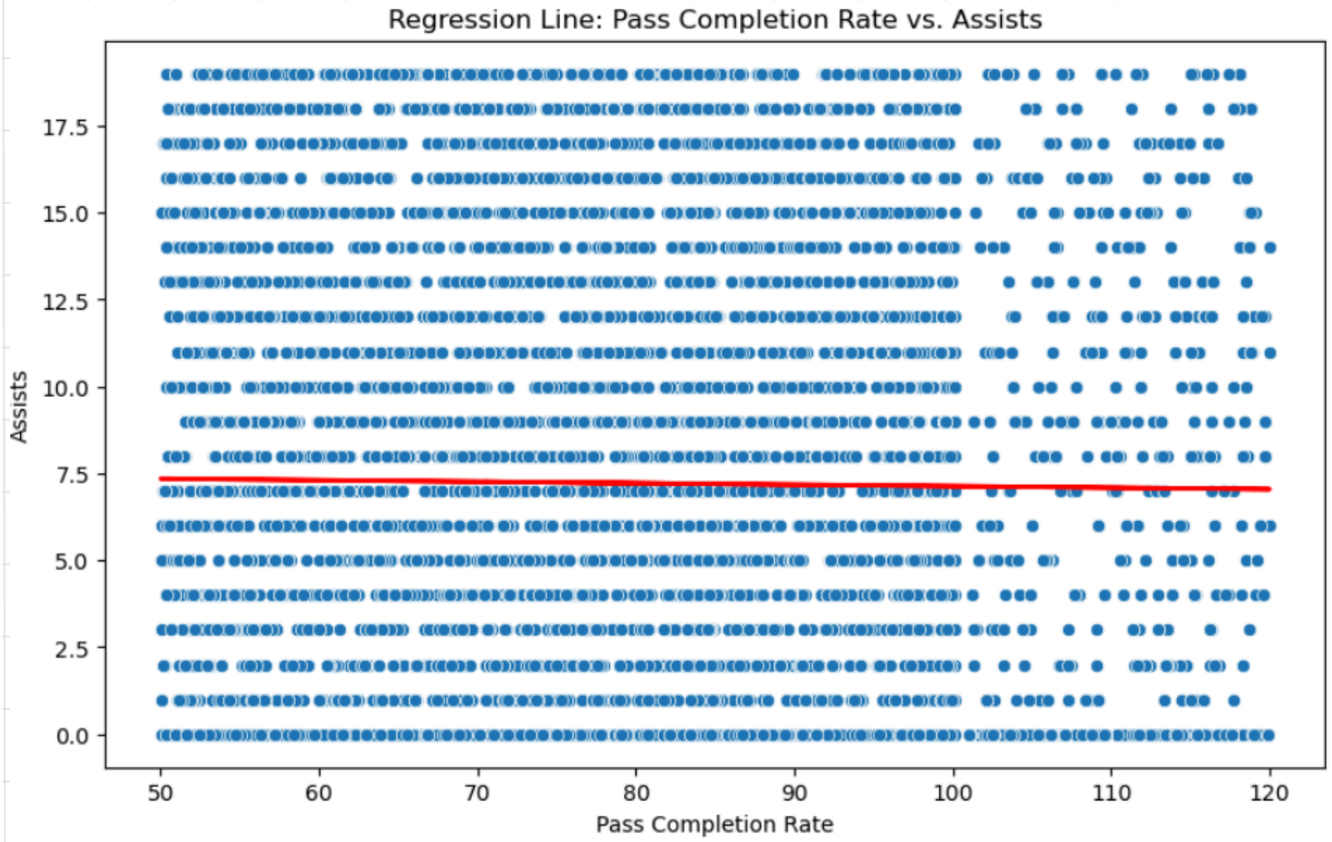
```
from sklearn.ensemble import IsolationForest
# Use Isolation Forest to find outliers in our data
iso_forest = IsolationForest(contamination=0.1)  # Set how many outliers we expect
outliers = iso_forest.fit_predict(df)  # Fit the model and predict outliers
df['Outlier'] = outliers  # Add the outlier info to our dataframe
```




```
# Plot the data, showing which points are outliers
plt.figure(figsize=(10, 6))  # Set the size of the plot
# Plot the data with outliers highlighted
sns.scatterplot(x='PassCompletionRate', y='Assists', hue='Outlier', data=df, palette={1: 'blue', -1: 'red'})
plt.title('Scatter Plot with Outliers Detected by Isolation Forest')
plt.xlabel('Pass Completion Rate')
plt.ylabel('Assists')
plt.show()  # Show the plot
```

Pass Completion Rate vs. Assists Outcome

Regression Analysis

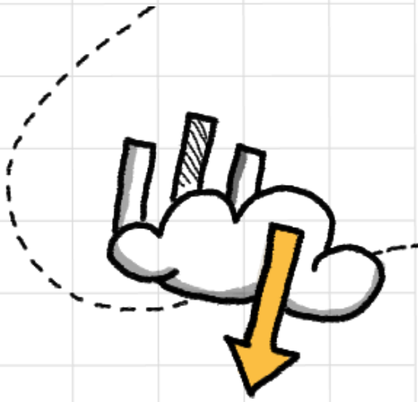



```python
from sklearn.linear_model import LinearRegression

# Set up the data for the regression
X = df[['PassCompletionRate']] # Our feature (independent variable)
y = df['Assists'] # Our target (dependent variable)

# Creating and fitting the linear regression model
model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)

# Plot the scatter plot with the regression line
plt.figure(figsize=(10, 6))
sns.scatterplot(x='PassCompletionRate', y='Assists', data=df) # Plot the original data
plt.plot(df['PassCompletionRate'], y_pred, color='red', linewidth=2) # Plot the regression line
plt.title('Regression Line: Pass Completion Rate vs. Assists')
plt.xlabel('Pass Completion Rate')
plt.ylabel('Assists')
```

The Godfather of Talent | Futurense

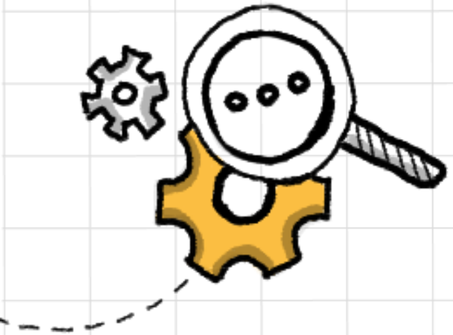
Problem Statement - 5

Advanced Data Transformations



- Perform complex dataset transformations.
- Conduct feature engineering for meaningful new features.
- Implement data optimization: normalization, dimensionality reduction.

Findings:



1. Explained Variance. First two principal components explain around **12.47%** of dataset variance indicating limited capture of dataset variation.
2. Soccer Analysis. Pass completion rates range from **50%** to **110%**, assists vary from **0** to **17.5** per game, highlighting a **potential correlation** between completion rates and assists.

Advanced Data Transformations Outcome

Dimensionality Reduction using PCA

```
# Reduce the number of features in our dataset so we can visualize it more easily
# Imagine squishing our data down into a 2D graph so we can see the big picture
from sklearn.decomposition import PCA  # Import PCA for dimensionality reduction
pca = PCA(n_components=2)  # Initialize PCA object to reduce dimensions to 2
new_sports_pca = pca.fit_transform(new_sports_scaled)  # Perform PCA on the normalized data
```

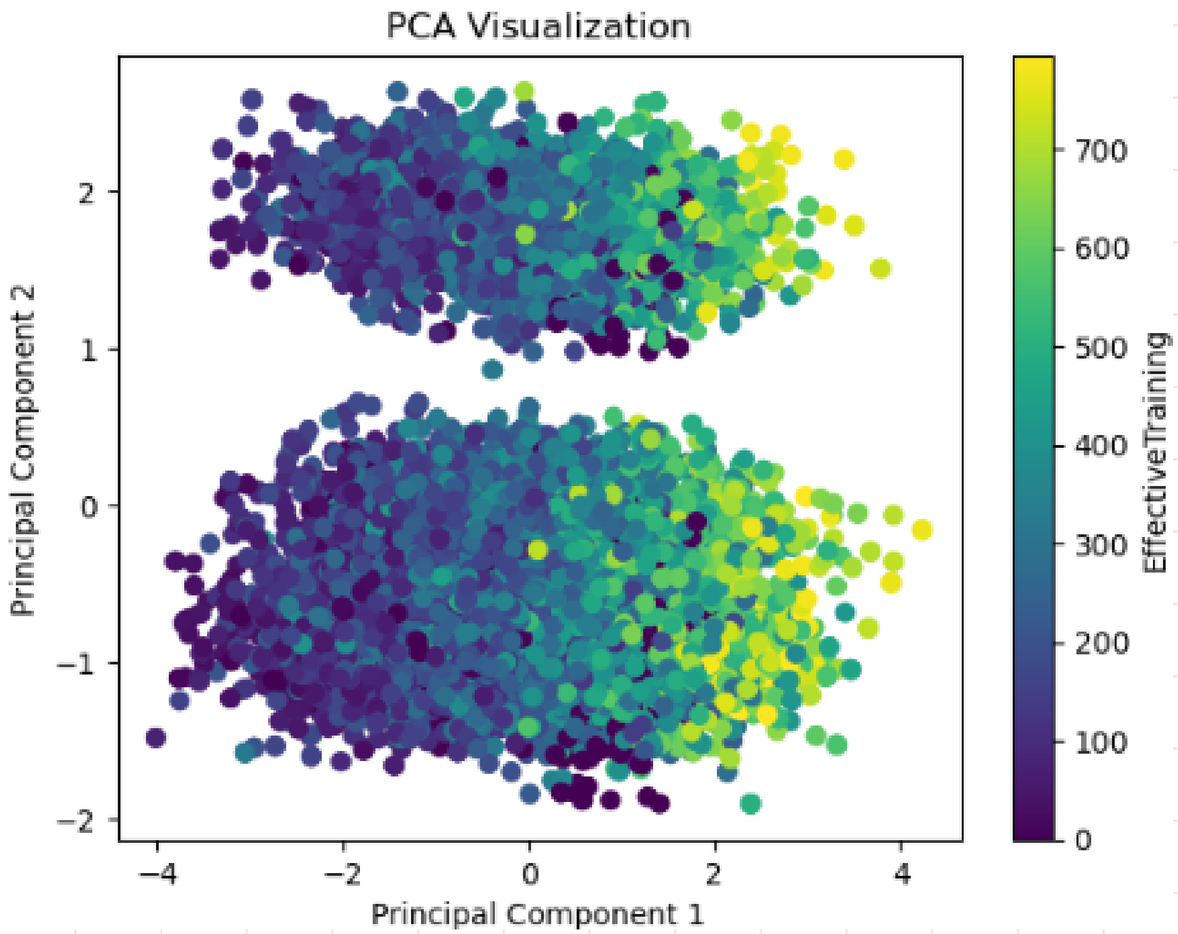
To check if the code has worked:

```
print("Explained Variance Ratio:", pca.explained_variance_ratio_)
```

```
Explained Variance Ratio: [0.06764174 0.05712927]
```

- **0.06764174**, indicates - **6.76%** of the total variance in the dataset.
- **0.05712927**, indicates - a **5.71%** of the total variance in the 2nd dataset.
- Together, these two principal components explain around **12.47% of the total variance** in the dataset.

Advanced Data Transformations Outcome




```python
# Draw a graph to show our data in a simple way
# This helps us understand the patterns and relationships between different parts of the data
import matplotlib.pyplot as plt  # Import matplotlib for visualization
# Scatter plot of PCA components
plt.scatter(new_sports_pca[:, 0], new_sports_pca[:, 1], c=y, cmap='viridis')
plt.xlabel('Principal Component 1')  # X-axis label
plt.ylabel('Principal Component 2')  # Y-axis label
plt.title('PCA Visualization')  # Title of the plot
plt.colorbar(label='EffectiveTraining')  # Color bar legend for target variable
plt.show()  # Display the plot
```

- **Principal Component 1**

Most significant underlying factor in player's performance identified by PCA.

- **Principal Component 2**

Additional important information about player's performance not captured by the first component.

Advanced Data Transformations Outcome

Feature Selection using SelectKBest

```
# Choose the most important features from our dataset
# We want to focus on the ones that have the biggest impact on our target variable
# Encode categorical columns using one-hot encoding
new_sports_encoded = pd.get_dummies(new_sports)

# Feature selection using SelectKBest
X = new_sports_encoded.drop(columns=['EffectiveTraining'])  # Features
y = new_sports_encoded['EffectiveTraining']  # Target variable
selector = SelectKBest(score_func=f_regression, k=5)  # Initialize SelectKBest object to select top 5 features
X_selected = selector.fit_transform(X, y)  # Select top 5 features
selected_feature_names = X.columns[selector.get_support()]  # Get the names of the selected features
```

Display Selected Features

```
# Show the names of the features that we selected as the most important
# These are the ones that will help us make the best predictions
print("Selected features:", selected_feature_names)
```

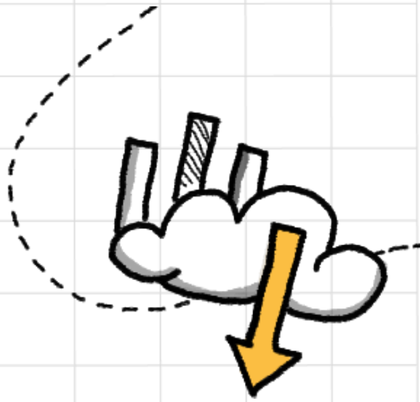
```
Selected features: Index(['Goals', 'Assists', 'DistanceCovered', 'PressurePerformanceImpact',
       'Position_Forward'],
      dtype='object')
```

- **Goals** - Indicator of a player's scoring ability and offensive contribution.
- **Assists** - Reflects playmaking skills and ability to set up scoring opportunities.
- **Distance Covered** - Represents work rate, endurance, and contribution in both attack and defense.
- **Pressure Performance Impact** - Shows effectiveness in handling pressure situations.
- **Position_Forward** - Specifies role in creating scoring chances and contributing to offensive plays.

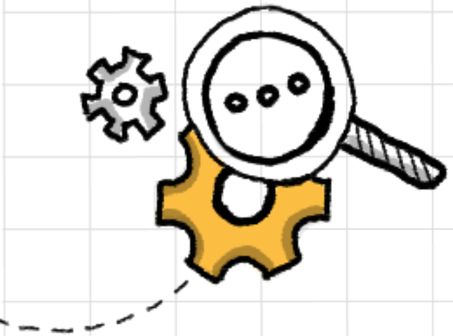
The Godfather of Talent | Futurense

Problem Statement - 2

Position Analysis



1. Analyze player positions for highest and lowest counts.
2. Use statistical analysis to compare distribution to uniform.
3. Plot player counts per position and distribution as pie chart.

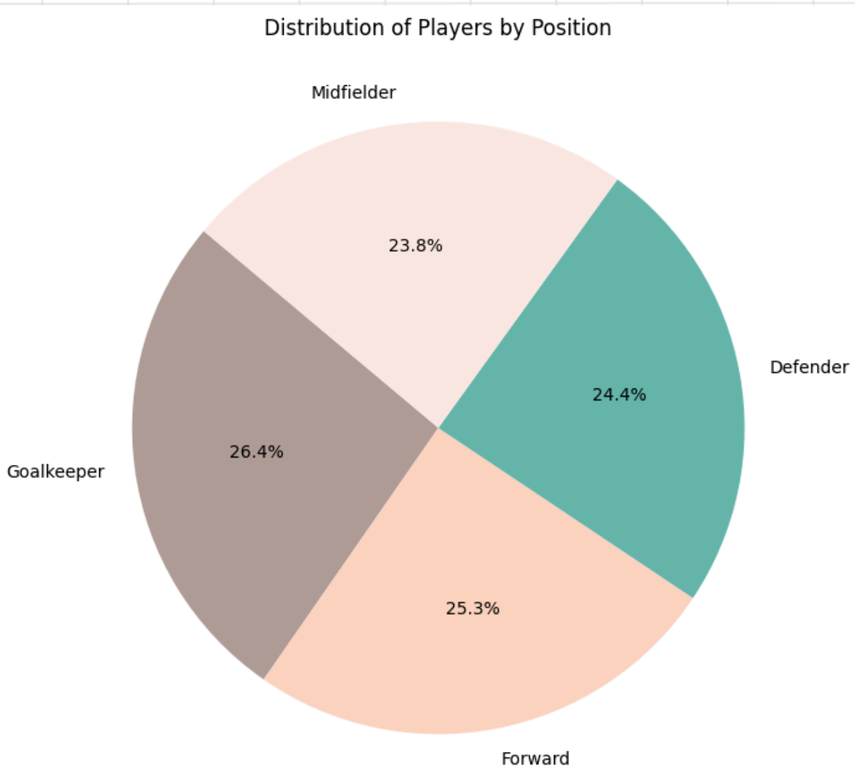


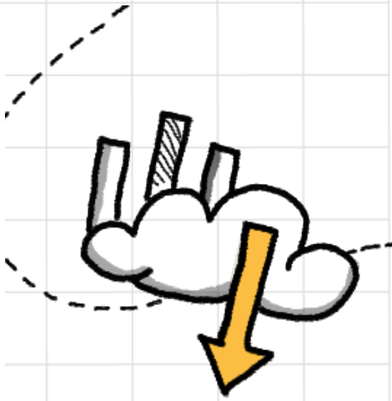
Findings:

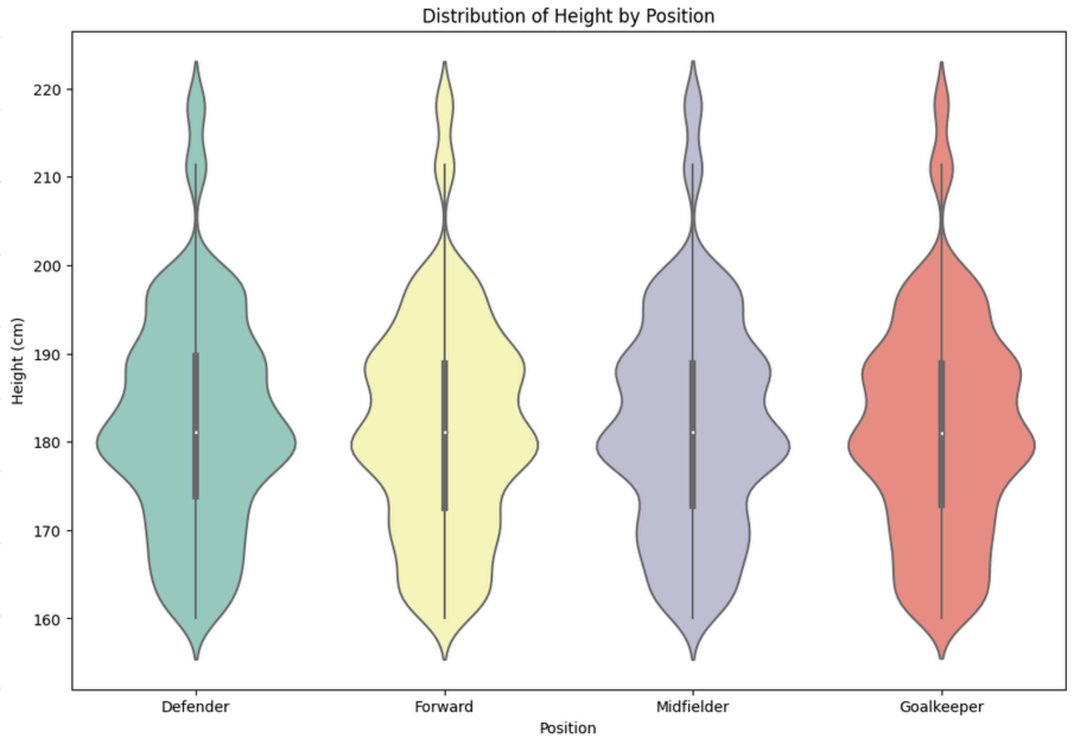
1. Forwards and goalkeepers dominate the dataset, with defenders and midfielders making up smaller proportions.
2. Focusing more on offensive and midfield strategies, potentially at the expense of defensive strength.

Position Analysis Outcome

```
colors = ['#B09E99', '#FAD4C0', '#64B6AC', '#FEE9E1']
plt.figure(figsize=(8, 8))
position_counts.plot(kind='pie', autopct='%1.1f%%', startangle=140, colors=colors)
plt.title('Distribution of Players by Position')
plt.ylabel('')
plt.show()
```



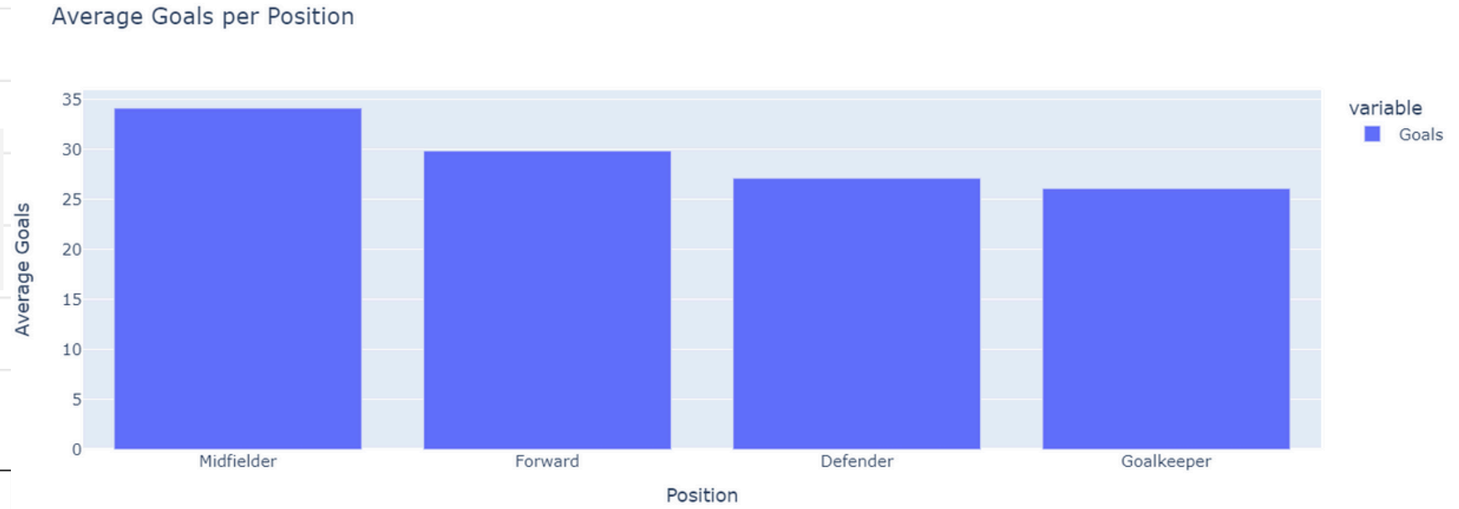


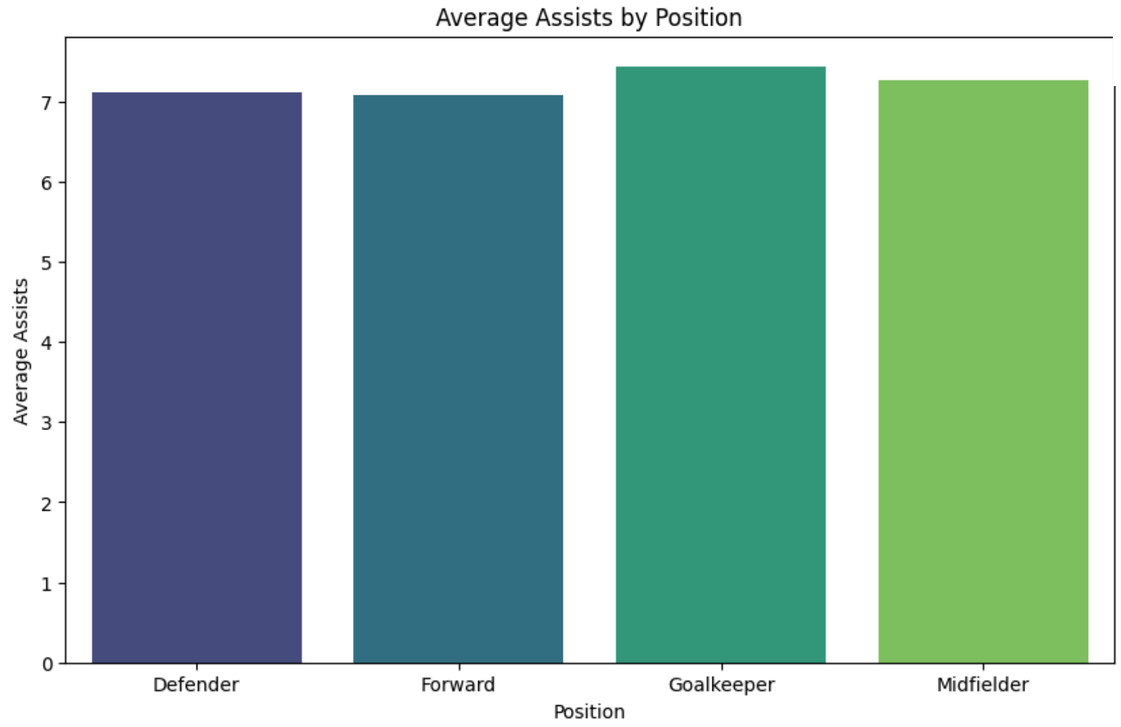



```
plt.figure(figsize=(12, 8))
sns.violinplot(x='Position', y='Height', data=df, palette='Set3')
plt.title('Distribution of Height by Position')
plt.xlabel('Position')
plt.ylabel('Height (cm)')
plt.show()
```

Position Analysis Outcome

```
avg_goals_per_position = df.groupby('Position')['Goals'].mean().sort_values(ascending=False)

fig3 = px.bar(avg_goals_per_position, title='Average Goals per Position')
fig3.update_layout(xaxis_title='Position', yaxis_title='Average Goals')
fig3.show()
```

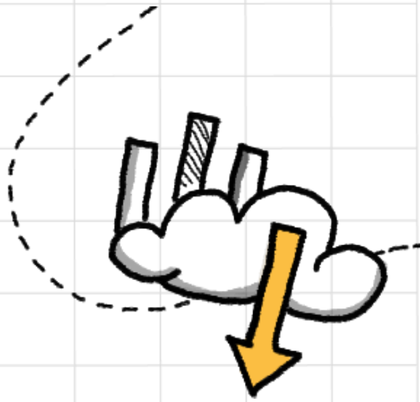





```
performance_by_position = df.groupby('Position')[['Goals', 'Assists']].mean()

plt.figure(figsize=(10, 6))
sns.barplot(x=performance_by_position.index, y=performance_by_position['Assists'], palette='viridis')
plt.title('Average Assists by Position')
plt.xlabel('Position')
plt.ylabel('Average Assists')
plt.show()
```

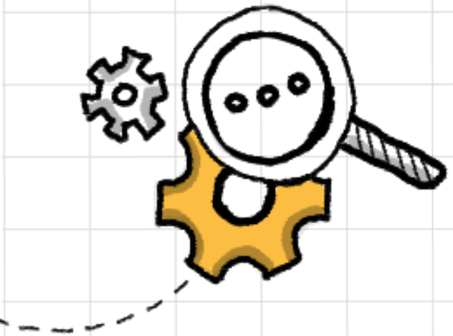
Problem Statement - 7

Team Goal Analysis



1. Identify team with highest goals.
2. Create horizontal bar plot and stacked bar chart.
3. Identify top scorer, analyze performance metrics over time.

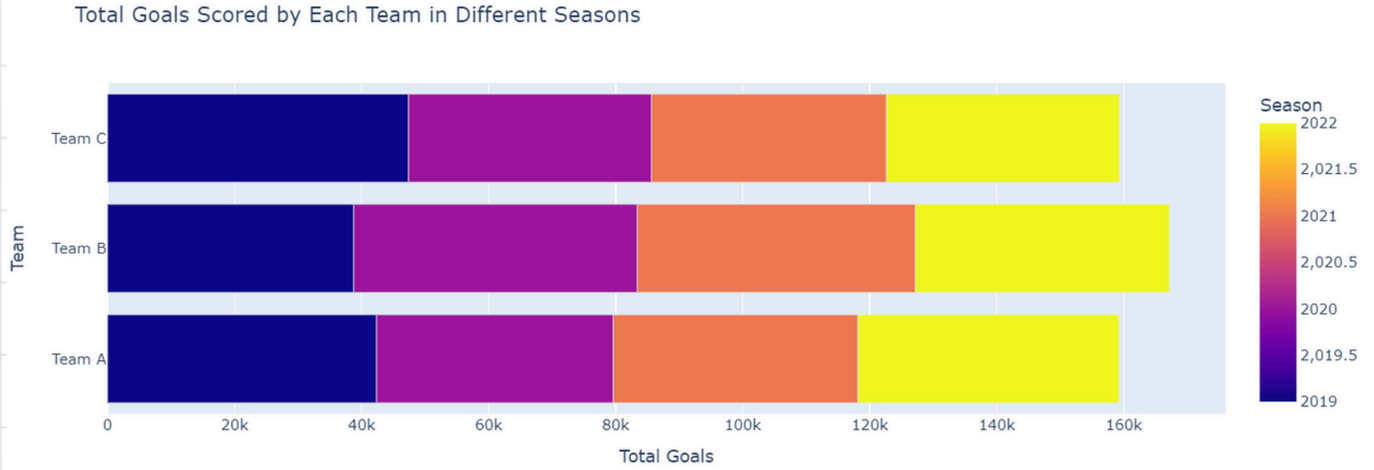
Findings:



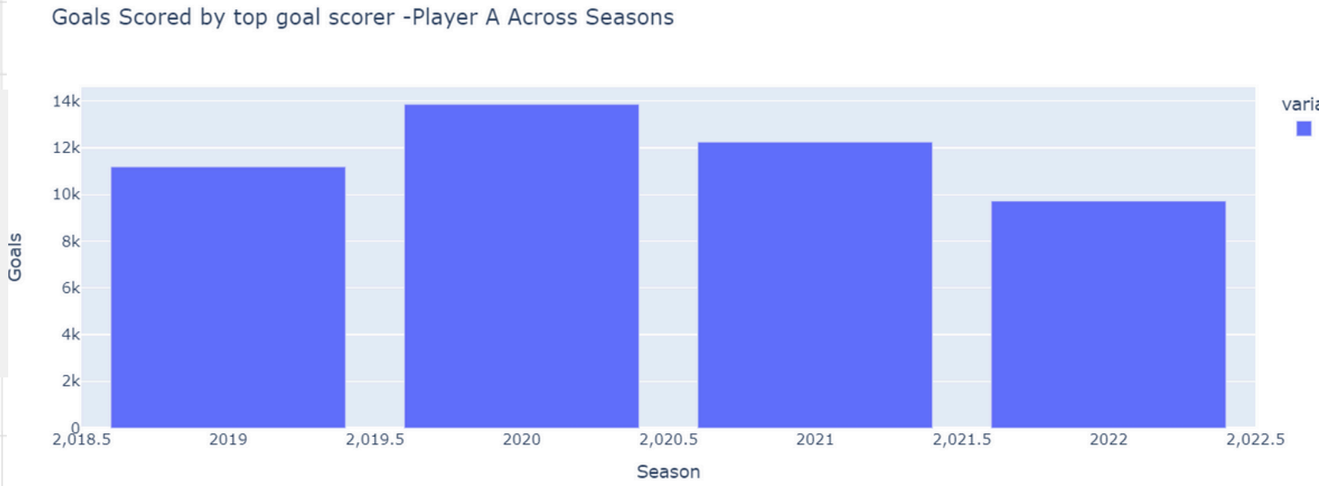
1. Midfielders are the ones who goal the most compared to other positions.
2. Team B is the top scorer which scored the most number of goals and their best performance was in the year 2020.

Team Goals Analysis Outcome

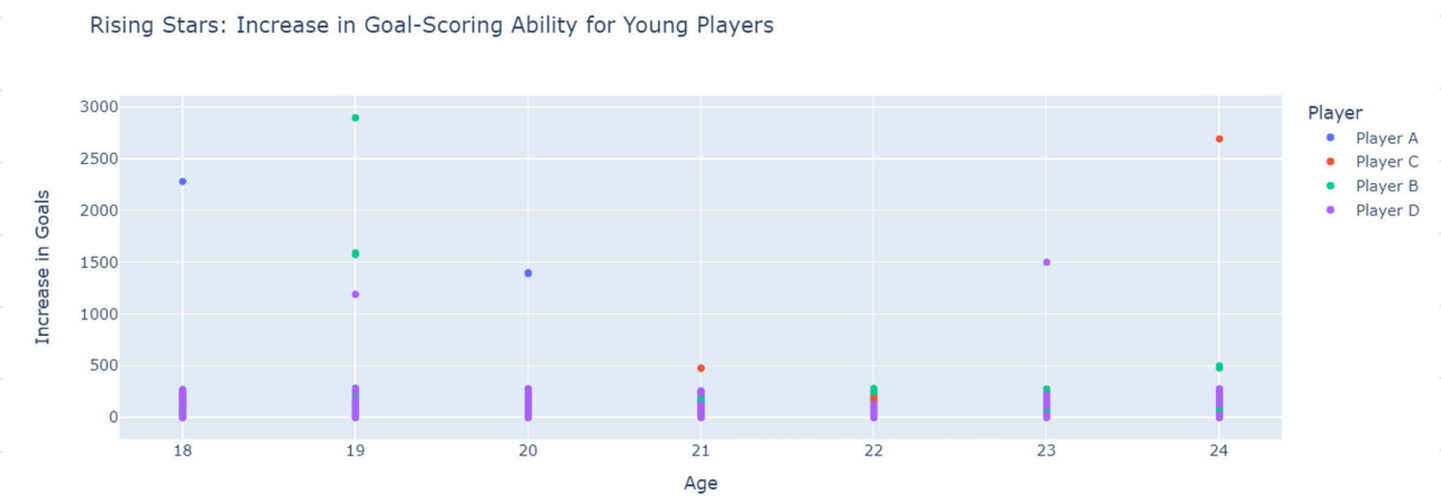
```
team_goals_season = df.groupby(['Team', 'Season'])['Goals'].sum().reset_index()

fig = px.bar(team_goals_season, x='Goals', y='Team', color='Season',
             orientation='h', title='Total Goals Scored by Each Team in Different Seasons',
             labels={'Goals': 'Total Goals', 'Team': 'Team', 'Season': 'Season'})
fig.show()
```




```
top_team_players = df[df['Team'] == top_team]
top_goal_scorer = top_team_players.groupby('Player')['Goals'].sum().idxmax()
top_goal_scorer_goals = top_team_players[top_team_players['Player'] == top_goal_scorer]

print(f"Top goal scorer in {top_team} is : ", top_goal_scorer)
print(top_goal_scorer_goals)

top_goal_scorer_season_goals = top_goal_scorer_goals.groupby('Season')['Goals'].sum().sort_index()

fig3 = px.bar(top_goal_scorer_season_goals, title=f'Goals Scored by top goal scorer -{top_goal_scorer} Across Seasons')
fig3.update_layout(xaxis_title='Season', yaxis_title='Goals')
fig3.show()
```



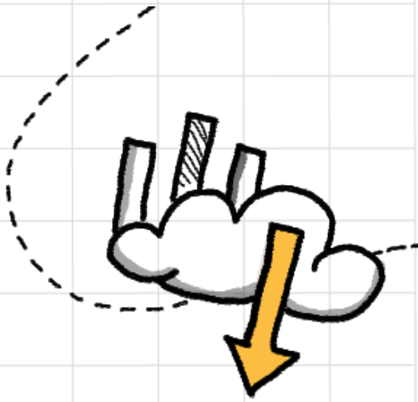
Team Goals Analysis Outcome



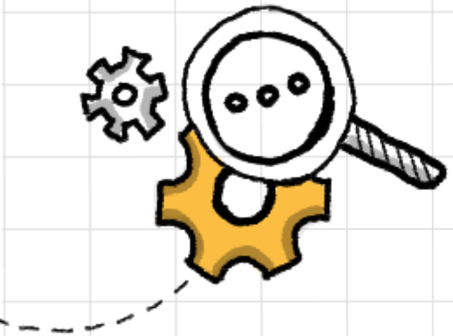

```
age_threshold = 25

df['GoalsDiff'] = df.groupby('Player')['Goals'].diff()

rising_stars = df[(df['Age'] < age_threshold) & (df['GoalsDiff'] > 0)]

fig = px.scatter(rising_stars, x='Age', y='GoalsDiff', color='Player',
                 title='Rising Stars: Increase in Goal-Scoring Ability for Young Players',
                 labels={'Age': 'Age', 'GoalsDiff': 'Increase in Goals', 'Player': 'Player'})
fig.show()
```

Problem Statement - 8

Reporting and Visualization



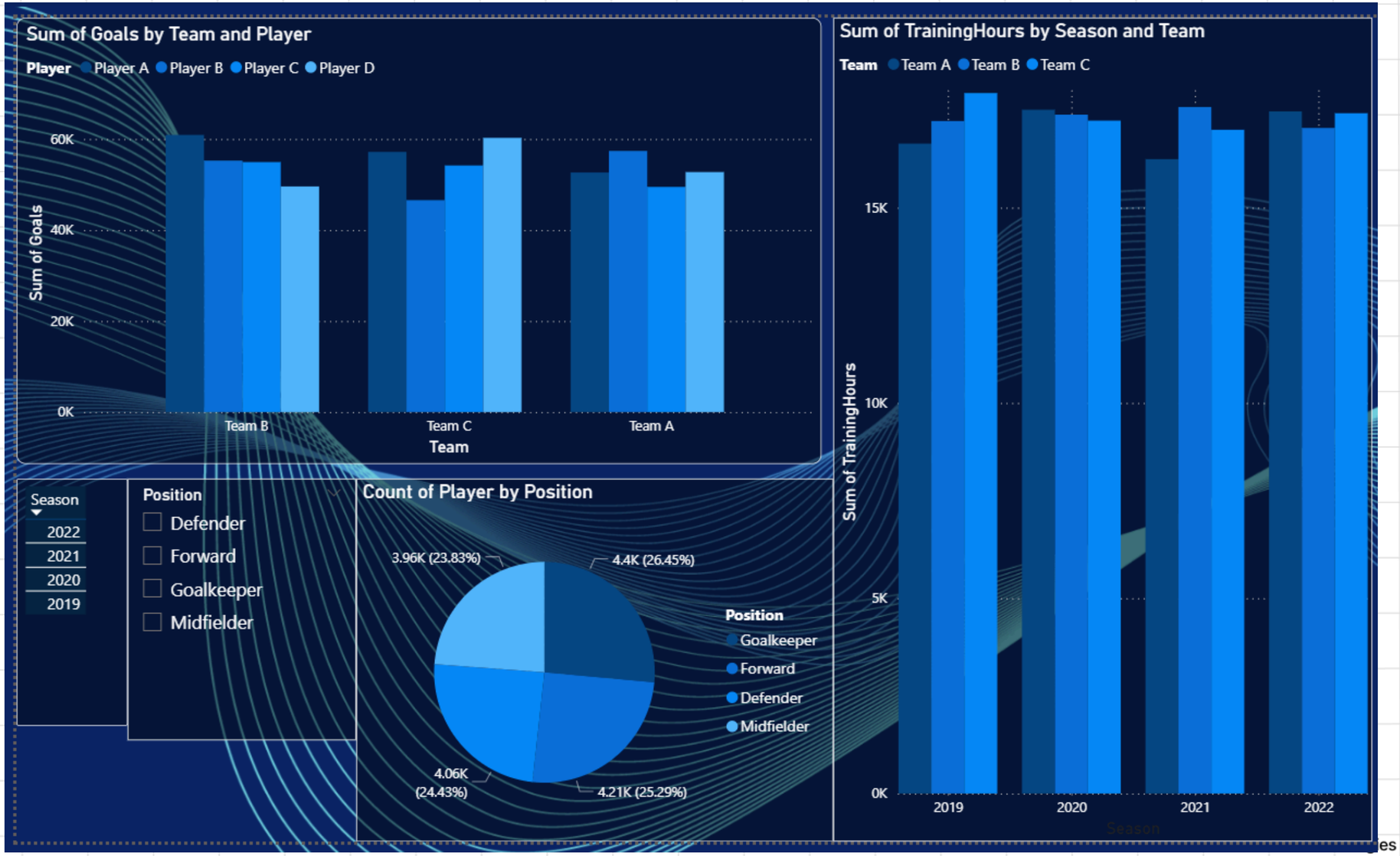
- Develop interactive dashboards and visualizations.
- Use tools like **Power BI**, Tableau, or custom web apps.
- Create reports on player performance, team strategies, and improvement areas.



Findings:

1. Team B and Team C have the highest individual player contributions.
2. Consistent training hours likely boost goal counts for Teams B and C, while Team A's training slightly decreased.
3. Goalkeepers and Forwards dominate the dataset.

Reporting & Visualization



The Godfather of Talent | Futurense

Case Study

BY MAURICIO SAVARESE

Updated 1:47 AM GMT+5:30, November 23, 2023

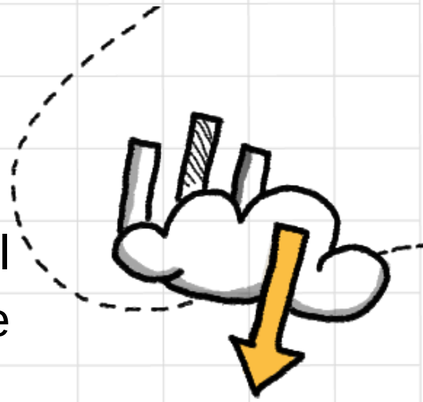
Many soccer analysts in Brazil say the team has struggled this year because Diniz has had little time to implement his tactics. Injuries and dwindling performances have also contributed to the team's poor form.

The problem for Brazil is that formula takes time to succeed, as it did with Fluminense. If opposing teams move the ball fast enough, they will find an unmarked man at some point, something that has happened to Brazil in several World Cup qualifying matches.

"We have a different style. It is different from what we had with Tite," Brazil defender Emerson Royal said Wednesday. "It is not easy to play like that. Few teams in the world can do that. What Diniz is trying to do with Brazil is a very hard thing to do."

The Godfather of Talent | Futurense

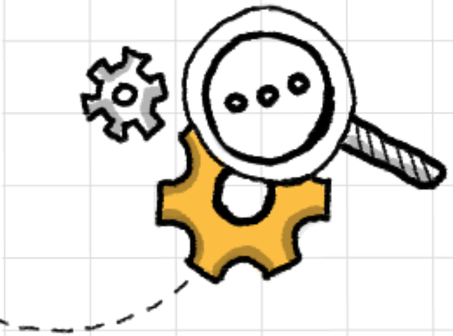
FUTURE SCOPE



1. Implement a **dynamic player development program**, training versatile players to excel in multiple positions, enhancing squad adaptability based on opponents and game situations.

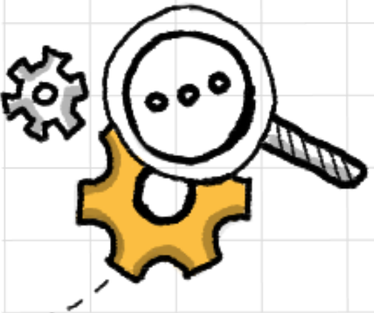
2. Develop **AI-powered performance analysis tools** utilizing advanced analytics to predict opponent strategies and recommend real-time tactical adjustments based on player data.

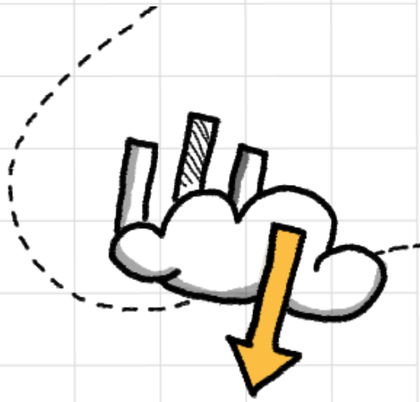
3. **Personalized Training with Wearable Tech**. Track exertion levels, injury risk, and fatigue for informed real-time substitution decisions.



REFERENCES

1. **CaseStudy:** Link
2. **Scikit-learn documentation on PCA:** Link
3. **Towards Data Science article on PCA:** Link
4. **GitHub repository: ml-pca-mysql** Link





Thank you!!!