```
csv_path = "/content/drive/MyDrive/t20_csv_files"
```

```
# 1. Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# 📌 Install dependencies
!pip install openai matplotlib seaborn pandas

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os, json
from openai import OpenAI

# 🔑 Set your OpenAI API key
os.environ["OPENAI_API_KEY"] = "sk-xxxxxx"   # replace with your key
client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

# ------------------------------
# 0. Choose Best Model
# ------------------------------
candidate_models = ["gpt-4o-mini", "gpt-4.1"]

if "gpt-4.1" in candidate_models:
    best_model = "gpt-4.1"
    reason = "Chosen for higher reasoning ability and better detailed explanations, suitable for presentation reports."
else:
    best_model = "gpt-4o-mini"
    reason = "Chosen for faster responses and lower cost, suitable for quick summaries."

print("✅ Best Model Selected for Cricket Analysis:", best_model)
print("ℹ️ Reason:", reason)
print("-"*80)

# ------------------------------
# 1. Mount Google Drive
# ------------------------------
from google.colab import drive
drive.mount('/content/drive')

# ------------------------------
# 2. Load Data
# ------------------------------
csv_path = "/content/drive/MyDrive/t20_csv_files"

dim_match_summary = pd.read_csv(os.path.join(csv_path, "t20_csv_files/dim_match_summary.csv"))
dim_players = pd.read_csv(os.path.join(csv_path, "t20_csv_files/dim_players.csv"))
dim_players_no_image = pd.read_csv(os.path.join(csv_path, "t20_csv_files/dim_players_no_images.csv"))
fact_bating_summary = pd.read_csv(os.path.join(csv_path, "t20_csv_files/fact_bating_summary.csv"))
fact_bowling_summary = pd.read_csv(os.path.join(csv_path, "t20_csv_files/fact_bowling_summary.csv"))

# ------------------------------
# 2a. Remove 'image' column completely
# ------------------------------
dim_players = dim_players.loc[:, ~dim_players.columns.str.contains("image", case=False)]
dim_players_no_image = dim_players_no_image.loc[:, ~dim_players_no_image.columns.str.contains("image", case=False)]

json_path = "/content/drive/MyDrive/t20_json_files/t20_wc_match_results.json"
if os.path.exists(json_path):
    with open(json_path, "r") as f:
        t20_json = json.load(f)
else:
    t20_json = None

# ------------------------------
# 3. Helper → Explain Function
# ------------------------------
def explain_with_models(title, df):
    prompt = f"""
    You are a cricket data analyst.
    Here is the data from analysis: \n{df.to_string(index=False)}
    Write a short, clear explanation for presentation with title: {title}.
    """
    try:
        response = client.chat.completions.create(
            model=best_model,
            messages=[
                {"role": "system", "content": "You explain cricket data clearly for project presentations."},
                {"role": "user", "content": prompt}
            ]
        )
        explanation = response.choices[0].message.content
    except Exception as e:
        explanation = f"Error: {e}"

    print(f"\n📌 {title}")
    print(f"\n=== Using {best_model} ===\n{explanation}\n")
    print("-"*80)

# ------------------------------
# 4. Exploratory Prints
# ------------------------------
pd.set_option("display.max_rows", None)

print("\nMatch Summary:")
display(dim_match_summary.style.background_gradient(cmap="Blues"))

print("\nBatting Summary:")
display(fact_bating_summary.style.background_gradient(cmap="Greens"))

print("\nBowling Summary:")
display(fact_bowling_summary.style.background_gradient(cmap="Reds"))

print("\nPlayers Data (without image columns):")
display(dim_players.style.background_gradient(cmap="Purples"))

# ------------------------------
# 5. Analytics Features + Explanations
# ------------------------------

## 1. Top Run Scorers → Horizontal Bar
top_batsmen = fact_bating_summary.groupby('batsmanName')['runs'].sum().sort_values(ascending=False).head(10)
top_batsmen.plot(kind='barh', figsize=(10,5), color="skyblue", title="Top 10 Run Scorers")
plt.xlabel("Total Runs")
plt.ylabel("Batsman")
plt.show()
display(top_batsmen.reset_index().style.background_gradient(cmap="Blues"))
explain_with_models("Top Run Scorers", top_batsmen.reset_index())

## 2. Top Wicket Takers → Pie Chart
top_bowlers = fact_bowling_summary.groupby('bowlerName')['wickets'].sum().sort_values(ascending=False).head(6)
plt.figure(figsize=(7,7))
top_bowlers.plot.pie(autopct='%1.1f%%', colormap="Set3", ylabel="")
plt.title("Top 6 Wicket Takers Share")
plt.show()
display(top_bowlers.reset_index().style.background_gradient(cmap="Oranges"))
explain_with_models("Top Wicket Takers", top_bowlers.reset_index())

## 3. Most Matches Played by Player → Bar
most_matches = dim_players.groupby('name')['name'].count().sort_values(ascending=False).head(10)
df_most_matches = most_matches.reset_index(name='match_count')
sns.barplot(x='match_count', y='name', data=df_most_matches, palette="Purples_r")
plt.title("Top 10 Players by Matches Played")
plt.xlabel("Matches Played")
plt.ylabel("Player")
plt.show()
display(df_most_matches.style.background_gradient(cmap="Purples"))
explain_with_models("Most Matches Played by Player", df_most_matches)

## 4. Best Strike Rate Batsmen → Scatter
batsman_stats = fact_bating_summary.groupby('batsmanName').agg({'runs':'sum','balls':'sum'})
batsman_stats['strike_rate'] = (batsman_stats['runs'] / batsman_stats['balls']) * 100
if (batsman_stats['runs'] > 200).any():
    best_strikers = batsman_stats[batsman_stats['runs'] > 200].sort_values('strike_rate', ascending=False).head(15)
else:
    best_strikers = batsman_stats.sort_values('strike_rate', ascending=False).head(15)

plt.figure(figsize=(8,6))
```

```
sns.scatterplot(x='runs', y='strike_rate', data=best_strikers.reset_index(), s=100, color="green")
for i, row in best_strikers.reset_index().iterrows():
    plt.text(row['runs'], row['strike_rate'], row['batsmanName'], fontsize=8)
plt.title("Best Strike Rate Batsmen (200+ runs)")
plt.xlabel("Runs Scored")
plt.ylabel("Strike Rate")
plt.show()
display(best_strikers.reset_index().style.background_gradient(cmap="Greens"))
explain_with_models("Best Strike Rate Batsmen", best_strikers.reset_index())

## 5. Best Bowling Economy → Line Chart
fact_bowling_summary['overs'] = pd.to_numeric(fact_bowling_summary['overs'], errors='coerce')
bowler_stats = fact_bowling_summary.groupby('bowlerName').agg({'runs':'sum','overs':'sum'})
bowler_stats['economy'] = bowler_stats['runs'] / bowler_stats['overs']
if (bowler_stats['overs'] > 50).any():
    best_economy = bowler_stats[bowler_stats['overs'] > 50].sort_values('economy').head(10)
else:
    best_economy = bowler_stats.sort_values('economy').head(10)

best_economy_plot = best_economy.reset_index()
plt.figure(figsize=(10,5))
sns.lineplot(x='bowlerName', y='economy', data=best_economy_plot, marker="o", color="red")
plt.title("Best Economy Bowlers (min 50 overs)")
plt.xticks(rotation=45, ha="right")
plt.show()
display(best_economy.reset_index().style.background_gradient(cmap="Reds"))
explain_with_models("Best Economy Bowlers", best_economy.reset_index())

## 6. Team Wins → Donut Chart
team_wins = dim_match_summary['winner'].value_counts().head(6)
plt.figure(figsize=(7,7))
team_wins.plot.pie(autopct='%1.1f%%', colormap="tab20", wedgeprops=dict(width=0.4))
plt.title("Top 6 Winning Teams Share")
plt.ylabel("")
plt.show()
display(team_wins.reset_index().style.background_gradient(cmap="BuGn"))
explain_with_models("Top Winning Teams", team_wins.reset_index())

## 7. Matches per Season → Area Chart
dim_match_summary['year'] = pd.to_datetime(dim_match_summary['matchDate']).dt.year
season_matches = dim_match_summary.groupby('year')['match_id'].count()
plt.figure(figsize=(10,5))
plt.fill_between(season_matches.index, season_matches.values, color="brown", alpha=0.5)
plt.plot(season_matches.index, season_matches.values, marker='o', color="brown")
plt.title("Matches per Year")
plt.xlabel("Year")
plt.ylabel("Number of Matches")
plt.show()
display(season_matches.reset_index().style.background_gradient(cmap="YlOrBr"))
explain_with_models("Matches per Season", season_matches.reset_index())

# ------------------------------
# 6. Final LLM Summary Report
# ------------------------------
summary_prompt = """
You are preparing a clean project presentation summary of cricket data analysis.
Summarize the findings across batting, bowling, players, teams, and matches.
Write it like a report (4-5 short paragraphs) for a student project.
"""

try:
    summary_response = client.chat.completions.create(
        model=best_model,
        messages=[
            {"role": "system", "content": "You summarize cricket insights clearly for presentations."},
            {"role": "user", "content": summary_prompt}
        ]
    )
    print(f"\n📌 Final Project Summary Report (Using {best_model}):\n")
    print(summary_response.choices[0].message.content)
    print("-"*80)
except Exception as e:
    print(f"LLM Summary Error ({best_model}): {e}")
```