Cricket Matches Sheet Analysis

Project Overview

This project focuses on extracting, transforming, and analyzing cricket match data from Cricsheet.org. The goal is to automate data collection, store it in an SQL database, perform deep analysis using SQL and Python, and visualize key insights through both Python plots and Power BI dashboards.

Project Objectives

- Web scrape cricket data using Selenium
- Parse and transform JSON files into structured data
- Store match and ball-level data in SQLite
- Perform SQL-based data analysis (20 queries)
- Visualize trends using Python (matplotlib, seaborn)
- Build an interactive Power BI dashboard

Tools and Technologies Used

- **Languages & Libraries**: Python, Selenium, Pandas, JSON
- **Database**: SQLite (via sqlite3 connector)
- **Querying Language**: SQL
- **Visualization**: Matplotlib, Seaborn, Power BI Desktop

Web Scraping with Selenium

Goal: Navigate to cricsheet.org and download match data ZIPs for Test, ODI, T20, and IPL formats.

Steps:

- Used Selenium to automate browser interaction
- Scraped and filtered `.zip` files containing match JSONs
- Downloaded only the 4 required formats

Code Highlights:

```
from selenium import webdriver
from selenium.webdriver.common.by import By
import time, os, requests

# Browser setup and navigate to site
...

# Find and download only selected formats (test, odi, t20, ipl)
...
```

Output:

- Downloaded 4 ZIPs into `cricsheet_json_files/`

Data Extraction and JSON Parsing

Goal: Extract `.zip` files and convert match JSON into DataFrames

Steps:

- Extract ZIPs using `zipfile`
- Created two major functions:
  - `parse_ball_by_ball()`: Generates `df_test`, `df_odi`, `df_t20`, `df_ipl`
  - `matchSummary()`: Produces match-level summaries

Code Highlights:

```
def parse_ball_by_ball(folder_path, match_type_label):
    # Extract batter, bowler, runs, wickets, etc.

def matchSummary(folder_path, match_type_label):
    # Extract match date, venue, result, toss info, margin
```

Output:

- 4 ball-by-ball DataFrames
- 1 merged match_summary DataFrame with margins and outcomes

SQLite Database Creation

Goal: Store parsed DataFrames in a SQLite database for efficient querying

Steps:

- Used `sqlite3` to create and connect to a `.db` file
- Stored all 6 DataFrames as individual tables

Code:

```
conn = sqlite3.connect("cricsheet_analysis.db")
df_test.to_sql("test_matches", conn, if_exists="replace")
...
df_match_summary.to_sql("match_summary", conn, if_exists="replace")
```

Output:

- Tables Created: `test_matches`, `odi_matches`, `t20_matches`, `ipl_matches`, `test_summary`, `match_summary`

SQL Queries for Insights

Goal: Use SQL to extract insights from cricket data

Total Queries: 20

Key Insights Extracted:

- Top 10 batsmen by runs (ODI)
- Most sixes (T20), fours (ODI)
- Best bowlers by wickets and dot balls (T20, IPL)
- Highest win % in Test cricket
- Toss winner vs match winner
- Match margins, match formats count
- Total boundaries, total centuries, 50+ scores

Code Snippet:

```
SELECT batter, SUM(runs_batter) AS total_runs
FROM odi_matches
GROUP BY batter
ORDER BY total_runs DESC
LIMIT 10;
```

Output:

- Result stored and visualized using pandas

EDA Using Python

Goal: Plot the results from SQL using Seaborn/Matplotlib

Visuals Created (at least 10):

- Top batsmen in ODIs
- Best T20 bowlers
- Win % by Test teams
- Match margin types

- Toss decision impact
- Dot ball bowlers
- Centuries and fifties
- Format-wise match counts

Example Code:

```
sns.barplot(data=df_result, y="batter", x="total_runs")
plt.title("Top 10 Run Scorers in ODIs")
plt.show()
```

Output:

- Graphs saved and displayed

Power BI Dashboard

Goal: Create an interactive dashboard using Power BI Desktop

Setup:

- Connected SQLite DB using **ODBC driver**
- Imported tables: match_summary, odi_matches etc.

Visualizations:

- Player performance (batting & bowling)
- Match outcomes per team
- Format-wise win/loss
- Toss decisions and victory type

Output:

- Dashboard screenshots saved