# 1.Quick overview (what we’ll build)

1. Install required software: Python, pip, MySQL Server (or use XAMPP), Git (optional).
2. Create a MySQL database & user.
3. Prepare CSV sample data.
4. Use Python scripts to load CSVs into MySQL.
5. Use Python (pandas + SQLAlchemy) to analyze data and create plots.
6. Query players and produce reports.

Software:

* Python 3.8+ (includes pip)
* MySQL Server (or XAMPP / WAMP / MAMP)
* Python packages: pandas, sqlalchemy, mysql-connector-python (or pymysql), matplotlib

Install MySQL:

brew install mysql

brew services start mysql

mysql\_secure\_installation

## Ubuntu / Debian (Linux)

sudo apt update

sudo apt install python3 python3-pip python3-venv mysql-server -y

sudo systemctl start mysql

sudo mysql\_secure\_installation

**Folder Structure**

cricket\_tracker/

├── data/

│ ├── players.csv

│ ├── matches.csv

│ └── performances.csv

├── scripts/

│ ├── load\_to\_db.py

│ └── analysis.py

├── plots/

│ ├── total\_runs\_per\_player.png

│ └── runs\_distribution.png

# 2 — Create project folder and Python virtual environment

Open terminal / cmd:

mkdir cricket-tracker

cd cricket-tracker

# Create virtual environment (recommended)

python -m venv venv

# Activate:

# Windows:

venv\Scripts\activate

# macOS / Linux:

source venv/bin/activate

# Install Python packages

pip install pandas sqlalchemy mysql-connector-python matplotlib

If you prefer pymysql instead of mysql-connector-python, install pymysql and use mysql+pymysql in SQLAlchemy connection strings.

# 3 — Create MySQL database and user

Open MySQL client (terminal):

# linux/mac

mysql -u root -p

# windows (use MySQL shell or Workbench)

Inside MySQL prompt:

-- Create database and user (replace 'your\_password' with chosen password)

CREATE DATABASE cricket\_db CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;

CREATE USER 'cricket\_user'@'localhost' IDENTIFIED BY 'cricket\_pass123';

GRANT ALL PRIVILEGES ON cricket\_db.\* TO 'cricket\_user'@'localhost';

FLUSH PRIVILEGES;

EXIT;

# 4 — Prepare CSV files (sample data)

Create folder data/ and 3 CSV files: players.csv, matches.csv, performances.csv.

Example players.csv:

player\_id,full\_name,role,birth\_date,state

P001,Mithali Raj,Batsman,1982-12-03,Andhra Pradesh

P002,Smarak Allrounder,Allrounder,1995-07-10,Maharashtra

P003,Harmanpreet Kaur,Batsman,1990-03-08,Punjab

P004,Jhulan Goswami,Bowler,1982-11-25,West Bengal

Example matches.csv:

match\_id,date,opponent,venue,tournament

M001,2023-03-12,Australia,Adelaide,ODI Series

M002,2023-03-18,England,Chennai,T20 Series

M003,2023-04-05,South Africa,Pune,ODI Series

Example performances.csv (note not\_out & runs\_conceded included):

perf\_id,match\_id,player\_id,runs,balls\_faced,fours,sixes,wickets,overs,maiden,balls\_bowled,runs\_conceded,catches,stumpings,not\_out

PF001,M001,P001,78,95,8,0,0,0.0,0,0,0,1,0,0

PF002,M001,P004,2,3,0,0,2,10.0,0,60,45,0,0,0

PF003,M002,P003,56,40,6,1,0,0.0,0,0,0,0,0,0

PF004,M002,P002,12,15,1,0,1,4.0,0,24,20,1,0,0

PF005,M003,P001,34,50,4,0,0,0.0,0,0,0,0,0,0

PF006,M003,P004,1,2,0,0,3,9.0,0,54,30,0,0,0

Save these CSVs in cricket-tracker/data/.

# 5 — MySQL-compatible schema (fixes VARCHAR)

MySQL cannot index TEXT in keys — use VARCHAR. Save this file sql\_schema\_mysql.sql or run in MySQL:

USE cricket;

CREATE TABLE IF NOT EXISTS players (

player\_id VARCHAR(50) PRIMARY KEY,

full\_name VARCHAR(100) NOT NULL,

role VARCHAR(50),

birth\_date DATE,

state VARCHAR(50)

);

CREATE TABLE IF NOT EXISTS matches (

match\_id VARCHAR(50) PRIMARY KEY,

date DATE,

opponent VARCHAR(100),

venue VARCHAR(100),

tournament VARCHAR(100)

);

CREATE TABLE IF NOT EXISTS performances (

perf\_id VARCHAR(50) PRIMARY KEY,

match\_id VARCHAR(50),

player\_id VARCHAR(50),

runs INT DEFAULT 0,

balls\_faced INT DEFAULT 0,

fours INT DEFAULT 0,

sixes INT DEFAULT 0,

wickets INT DEFAULT 0,

overs DECIMAL(4,1) DEFAULT 0.0,

maiden INT DEFAULT 0,

balls\_bowled INT DEFAULT 0,

runs\_conceded INT DEFAULT 0,

catches INT DEFAULT 0,

stumpings INT DEFAULT 0,

not\_out INT DEFAULT 0,

FOREIGN KEY (match\_id) REFERENCES matches(match\_id),

FOREIGN KEY (player\_id) REFERENCES players(player\_id)

);

CREATE INDEX idx\_perf\_player ON performances (player\_id);

CREATE INDEX idx\_perf\_match ON performances (match\_id);

Run the file in MySQL Workbench or the mysql CLI:

mysql -u root -p cricket < sql\_schema\_mysql.sql

# 6 — Python script to load CSVs into MySQL

Create scripts/load\_to\_mysql.py:

"""

Cricket Tracker - Load CSV Data into MySQL Database

---------------------------------------------------

"""

**import** pandas **as** pd

**from** sqlalchemy **import** create\_engine, text

**import** os

# ---------------------------------------------------------------------

# 1. Database connection details

# ---------------------------------------------------------------------

DB\_USER **=** "root"       # MySQL username

DB\_PASS **=** ""    # MySQL password

DB\_HOST **=** "localhost"          # Host (use 127.0.0.1 if localhost fails)

DB\_NAME **=** "cricket"         # Database name

# ---------------------------------------------------------------------

# 2. CSV file locations (make sure files exist in 'data' folder)

# ---------------------------------------------------------------------

CSV\_PLAYERS **=** os.path.join('data', 'players.csv')

CSV\_MATCHES **=** os.path.join('data', 'matches.csv')

CSV\_PERF **=** os.path.join('data', 'performances.csv')

# ---------------------------------------------------------------------

# 3. Create SQLAlchemy engine for MySQL

# ---------------------------------------------------------------------

# "mysql+mysqlconnector://" is the connection dialect for MySQL

engine **=** create\_engine(

**f**"mysql+mysqlconnector://{DB\_USER}:{DB\_PASS}@{DB\_HOST}/{DB\_NAME}",

    echo**=False**

)

# ---------------------------------------------------------------------

# 4. Helper function to load a CSV into a table

# ---------------------------------------------------------------------

**def** load\_csv\_to\_table(csv\_path, table\_name):

    """Load CSV data into MySQL table using pandas."""

    print(**f**"Loading {csv\_path} → {table\_name} ...")

    df **=** pd.read\_csv(csv\_path)

    df.to\_sql(table\_name, con**=**engine, if\_exists**=**'append', index**=False**)

    print(**f**"✅ Loaded {len(df)} records into {table\_name}")

# ---------------------------------------------------------------------

# 5. Main logic

# ---------------------------------------------------------------------

**def** main():

    print("Starting data load process...")

    # Step 1: Disable foreign key checks so we can truncate safely

**with** engine.begin() **as** conn:

        print("Truncating old data...")

        conn.execute(text("SET FOREIGN\_KEY\_CHECKS=0;"))

        conn.execute(text("TRUNCATE TABLE performances;"))

        conn.execute(text("TRUNCATE TABLE matches;"))

        conn.execute(text("TRUNCATE TABLE players;"))

        conn.execute(text("SET FOREIGN\_KEY\_CHECKS=1;"))

        print("Tables truncated successfully.")

    # Step 2: Load new CSV data into MySQL

    load\_csv\_to\_table(CSV\_PLAYERS, 'players')

    load\_csv\_to\_table(CSV\_MATCHES, 'matches')

    load\_csv\_to\_table(CSV\_PERF, 'performances')

    print("🎉 All CSV files successfully loaded into MySQL database!")

# ---------------------------------------------------------------------

# 6. Run the script

# ---------------------------------------------------------------------

**if** \_\_name\_\_ **==** "\_\_main\_\_":

**try**:

        main()

**except** Exception **as** e:

        print("❌ Error while loading data:", e)

pip install pandas sqlalchemy mysql-connector-python

**Loaded CSVs into MySQL database.**

# Run the Code

python scripts/load\_to\_db.py

**Example output:**

Starting data load process...

Truncating old data...

Tables truncated successfully.

Loading data/players.csv → players ...

✅ Loaded 4 records into players

Loading data/matches.csv → matches ...

✅ Loaded 3 records into matches

Loading data/performances.csv → performances ...

✅ Loaded 6 records into performances

🎉 All CSV files successfully loaded into MySQL database!

# 7 — Python analysis script (reads data from MySQL)

Create scripts/analysis.py:

"""

Cricket Tracker - Performance Analysis using MySQL, Pandas, and Matplotlib

---------------------------------------------------------------------------"""

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**from** sqlalchemy **import** create\_engine

**import** os

# ---------------------------------------------------------------------

# 1. Database connection setup

# ---------------------------------------------------------------------

DB\_USER **=** "root"       # must match your MySQL setup

DB\_PASS **=** ""

DB\_HOST **=** "localhost"

DB\_NAME **=** "cricket"

# Create SQLAlchemy engine

engine **=** create\_engine(

**f**"mysql+mysqlconnector://{DB\_USER}:{DB\_PASS}@{DB\_HOST}/{DB\_NAME}"

)

# ---------------------------------------------------------------------

# 2. Load tables from MySQL into pandas DataFrames

# ---------------------------------------------------------------------

print("📦 Loading data from MySQL...")

players **=** pd.read\_sql("SELECT \* FROM players", engine)

matches **=** pd.read\_sql("SELECT \* FROM matches", engine)

perf **=** pd.read\_sql("SELECT \* FROM performances", engine)

print(**f**"✅ Loaded {len(players)} players, {len(matches)} matches, {len(perf)} performance records.")

# ---------------------------------------------------------------------

# 3. Data analysis using pandas

# ---------------------------------------------------------------------

print("\n⚙️ Analyzing player performance...")

# Total runs and wickets per player

summary **=** perf.groupby('player\_id', as\_index**=False**).agg({

    'runs': 'sum',

    'balls\_faced': 'sum',

    'wickets': 'sum',

    'not\_out': 'sum'

})

# Add player name

summary **=** summary.merge(players[['player\_id', 'full\_name', 'role']], on**=**'player\_id', how**=**'left')

# Calculate batting average and strike rate

summary['innings'] **=** perf.groupby('player\_id')['perf\_id'].count().values

summary['dismissals'] **=** summary['innings'] **-** summary['not\_out']

summary['dismissals'] **=** summary['dismissals'].replace(0, pd.NA)

summary['batting\_average'] **=** summary['runs'] **/** summary['dismissals']

summary['strike\_rate'] **=** summary.apply(

**lambda** r: (r['runs'] **/** r['balls\_faced'] **\*** 100) **if** r['balls\_faced'] **>** 0 **else** 0,

    axis**=**1

)

# Sort players by total runs

summary **=** summary.sort\_values(by**=**'runs', ascending**=False**)

print("📊 Batting Summary:")

print(summary[['player\_id', 'full\_name', 'role', 'runs', 'wickets', 'batting\_average', 'strike\_rate']].to\_string(index**=False**))

# ---------------------------------------------------------------------

# 4. Visualization (Matplotlib)

# ---------------------------------------------------------------------

os.makedirs('plots', exist\_ok**=True**)

# Bar Chart - Total Runs per Player

plt.figure(figsize**=**(8,5))

plt.bar(summary['full\_name'], summary['runs'], color**=**'skyblue', edgecolor**=**'black')

plt.title("Total Runs by Player")

plt.xlabel("Player")

plt.ylabel("Total Runs")

plt.xticks(rotation**=**20)

plt.tight\_layout()

plt.savefig('plots/total\_runs\_per\_player.png')

plt.close()

# Histogram - Distribution of Runs per Innings

plt.figure(figsize**=**(8,5))

plt.hist(perf['runs'], bins**=**range(0, 101, 10), color**=**'lightgreen', edgecolor**=**'black')

plt.title("Distribution of Runs per Innings")

plt.xlabel("Runs Scored")

plt.ylabel("Frequency")

plt.tight\_layout()

plt.savefig('plots/runs\_distribution.png')

plt.close()

print("Charts created in the 'plots' folder:")

print(" - total\_runs\_per\_player.png")

print(" - runs\_distribution.png")

# ---------------------------------------------------------------------

# 5. Individual player performance report

# ---------------------------------------------------------------------

**def** player\_report(player\_id):

    """Return detailed stats for a given player."""

    pinfo **=** players[players['player\_id'] **==** player\_id].iloc[0]

    pperf **=** perf[perf['player\_id'] **==** player\_id].merge(matches, on**=**'match\_id')

    total\_runs **=** pperf['runs'].sum()

    total\_wickets **=** pperf['wickets'].sum()

    total\_balls **=** pperf['balls\_faced'].sum()

    innings **=** len(pperf)

    not\_outs **=** pperf['not\_out'].sum()

    dismissals **=** innings **-** not\_outs **if** innings **>** 0 **else** 0

    avg **=** round(total\_runs **/** dismissals, 2) **if** dismissals **>** 0 **else** total\_runs

    sr **=** round(total\_runs **/** total\_balls **\*** 100, 2) **if** total\_balls **>** 0 **else** 0

    print(" Player Performance Report:")

    print(**f**" Player Name : {pinfo['full\_name']}")

    print(**f**" Role        : {pinfo['role']}")

    print(**f**" Total Runs  : {total\_runs}")

    print(**f**" Total Wkts  : {total\_wickets}")

    print(**f**" Innings     : {innings}")

    print(**f**" Not Outs    : {not\_outs}")

    print(**f**" Average     : {avg}")

    print(**f**" Strike Rate : {sr}\n")

    print(" Match-wise Breakdown:")

    print(pperf[['match\_id', 'date', 'opponent', 'runs', 'balls\_faced', 'wickets']].to\_string(index**=False**))

# Example report (you can test with any player\_id like P001)

player\_report('P001')

print("Analysis completed successfully!")

Run:

python scripts/analysis.py

Outputs:

* Console: batting summary and player report
* plots/ folder: PNGs for charts

**Matplotlib** **for visualization and Reports**

# Run the analysis

python scripts/analysis\_mysql.py

**Example output:**

📦 Loading data from MySQL...

✅ Loaded 4 players, 3 matches, 6 performance records.

⚙️ Analyzing player performance...

📊 Batting Summary:

player\_id full\_name role runs wickets batting\_average strike\_rate

P001 Mithali Raj Batsman 112 0 56.00 74.12

P003 Harmanpreet Kaur Batsman 56 0 56.00 140.00

P004 Jhulan Goswami Bowler 3 5 1.50 25.00

P002 Smarak Allrounder Allrounder 12 1 12.00 80.00

📈 Charts created in the 'plots' folder:

- total\_runs\_per\_player.png

- runs\_distribution.png

📄 Player Performance Report:

Player Name : Mithali Raj

Role : Batsman

Total Runs : 112

Total Wkts : 0

Innings : 2

Not Outs : 0

Average : 56.0

Strike Rate : 74.12

Match-wise Breakdown:

match\_id date opponent runs balls\_faced wickets

M001 2023-03-12 Australia 78 95 0

M003 2023-04-05 South Africa 34 50 0

✅ Analysis completed successfully!