

[Click here to visit the tutorial video.](#)

# PROJECT PHASE-4 REPORT

TEAM 20 : DATA DUDES

December 3, 2023

## 1 Team Members

- Tanishq Agarwal(2022101060)
- Kevin Thakkar(2022101064)
- Sahil Patel(2022101046)
- Gopal Garg(2022101079)

## 2 Introduction

Our Python CLI offers a user-friendly interface for accessing and manipulating the database, with various features to enhance the gaming experience. Whether you're a player tracking progress, a developer seeking insights, or an analyst exploring trends, the PUBG Mobile Database is your go-to resource. Join us as we delve into the world of PUBG Mobile data, empowering users to explore, analyze, and make informed decisions based on their gaming experiences. Let the adventure begin!

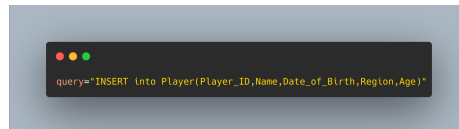
## 3 List Of Queries we have implemented:

1. *Inserting Player Information*
2. *Inserting maps Information*
3. *Inserting weapons Information*
4. *Inserting clans Information*
5. *Inserting team Information*
6. *Inserting extension Information*
7. *Inserting Player to clan*
8. *Inserting match details of the game at it's start*

9. *Deleting a map*
10. *Deleting a player from the clan*
11. *Updating clan-leader of a clan*
12. *Updating match details of a game after it has ended*
13. *Updating player's PLAYER-ID*
14. *To show list of maps in a game*
15. *To show list of weapons in a game*
16. *To show list of players in a game*
17. *To show teams with highest win-rate*
18. *To show players with highest total-wins*
19. *To show players who have teamed-up*
20. *To show the weapon with most kills*
21. *To show the output of list of all game maps*
22. *Retrieving extension for a particular gun*
23. *Retrieving all guns which have damage greater than a particular value*
24. *Retrieving top gun for a particular player*
25. *Retrieving best team members for a particular value*
26. *Retrieving players having KD greater than or equal to a particular value*

## 4 Explanation for each

- **First Query:** The Player ID is unique to each player to maintain data



*Figure 1: Query-1*

integrity. This query is essential for populating the "Player" table with information about individual players, including their personal details such as name, date of birth, region, and age.

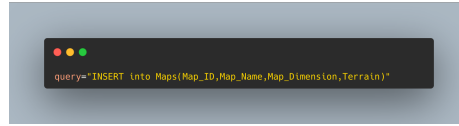


Figure 2: Query-2

- **Second Query:** The Map ID is unique to each map to maintain data integrity. This query is crucial for populating the "Maps" table with information about different maps in the game, including their names, dimensions, and terrain types.

- **Third Query:** The Weapon ID is unique to each weapon to maintain

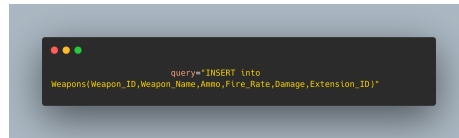


Figure 3: Query-3

data integrity. This query is essential for populating the "Weapons" table with information about different weapons in the game, including their names, ammunition type, fire rate, damage, and any associated extensions.

- **Fourth Query:** The Clan ID is unique to each clan to maintain data

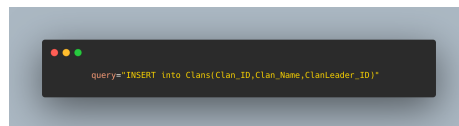
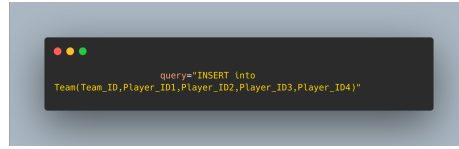


Figure 4: Query-4

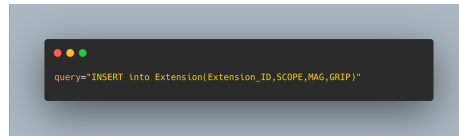
integrity. This query is crucial for populating the "Clans" table with information about different clans in the game, including their names and the unique identifier of the player who serves as the clan leader.

- **Fifth Query:** The Team ID is unique to each team to maintain data integrity. This query is essential for populating the "Team" table with information about different teams in the game, specifying the unique identifier of each player in the team.



*Figure 5: Query-5*

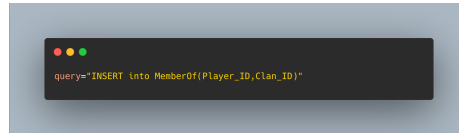
- **Sixth Query:** The Extension ID is unique to each extension to maintain



*Figure 6: Query-6*

data integrity. This query is crucial for populating the "Extension" table with information about different extensions in the game, specifying the types of scope, magazine, and grip associated with each extension.

- **Seventh Query:** The Provided Player-ID corresponds to an existing



*Figure 7: Query-7*

player in the "Player" table. This query is crucial for populating the "Member-Of" table, establishing relationships between players and clans, indicating which players belong to which clans in the game.

- **Eighth Query:** The Provided Map-ID corresponds to an existing map in the "Maps" table. This query is useful for deleting a specific map entry from the "Maps" table based on its unique identifier, allowing for the removal of outdated or unnecessary map information from the database.

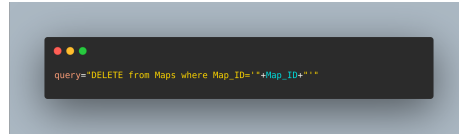


Figure 8: Query-8

- **Ninth Query:** The Provided Player-ID and Clan-ID correspond to



Figure 9: Query-9

an existing membership record in the "Member-Of" table. This query is useful for removing the association between a player and a clan in the "Member-Of" table, effectively ending the player's membership in the specified clan.

- **Tenth Query:** The Provided Clan-ID corresponds to an existing clan in



Figure 10: Query-10

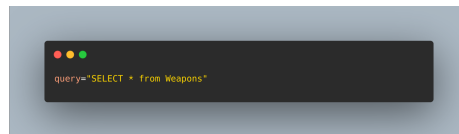
the "Clans" table. This query is useful for updating the leader of a clan in the "Clans" table, allowing for changes in leadership within the gaming community.

- **Eleventh Query:** The asterisk (\*) is a wildcard character that represents all columns in the specified table. This query is useful when you want to view the complete data present in the "Maps" table.



*Figure 11: Query-11*

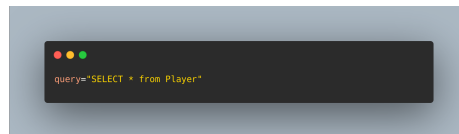
- **Twelfth Query:** The asterisk (\*) is a wildcard character that represents



*Figure 12: Query-12*

all columns in the specified table. This query is useful when you want to view the complete data present in the "Weapons" table.

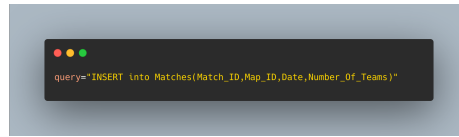
- **Thirteenth Query:** The asterisk (\*) is a wildcard character that rep-



*Figure 13: Query-13*

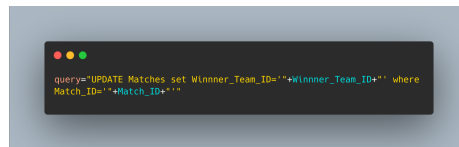
resents all columns in the specified table. This query is useful when you want to view the complete data present in the "Players" table.

- **Fourteenth Query:** Ensure that the provided Match-ID is unique to each match to maintain data integrity. This query is crucial for populating the "Matches" table with information about different matches in the game, including the map associated with the match, date, and the number of participating teams.



*Figure 14: Query-14*

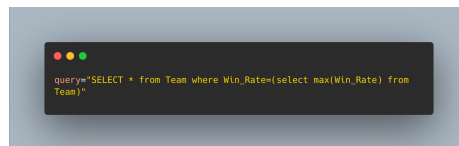
- **Fifteenth Query:** Ensure that the provided Match-ID corresponds to



*Figure 15: Query-15*

an existing match in the "Matches" table. This query is useful for updating the winner of a match in the "Matches" table, reflecting the outcome of the game in the database.

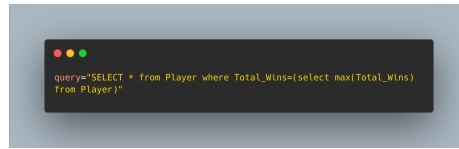
- **Sixteenth Query:** The sub-query (SELECT MAX(Win-Rate) FROM



*Figure 16: Query-16*

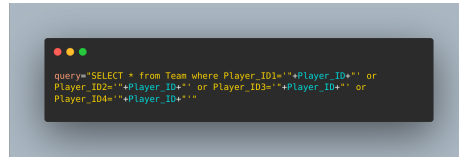
Team) is used to find the maximum win rate in the "Team" table. This query is useful for identifying the team or teams with the highest win rate in the "Team" table.

- **Seventeenth Query:** The sub-query (SELECT MAX(Total-Wins) FROM Player) is used to find the maximum total wins in the "Player" table. This query is useful for identifying the player or players with the highest total wins in the "Player" table.



*Figure 17: Query-17*

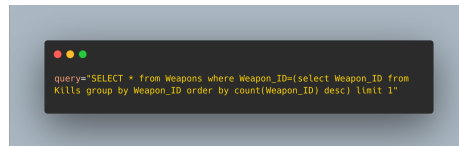
- **Eighteenth Query:** The Provided Player-ID ('player789') corresponds



*Figure 18: Query-18*

to an existing player in the "Player" table. This query is useful for retrieving information about teams that include a specific player in the "Team" table.

- **Nineteenth Query:** The sub-query is used to find the "Weapon-ID"



*Figure 19: Query-19*

with the highest number of kills in the "Kills" table. This query is useful for identifying the weapon with the most kills in the game, based on the data in the "Kills" table.

- **Twentieth Query:** This query is useful for retrieving information about weapons and their associated extensions based on specified search criteria.

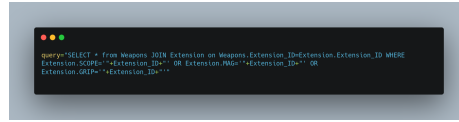


Figure 20: Query-20

- **Twenty-First Query:** This query is useful for retrieving information

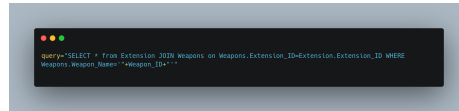


Figure 21: Query-21

about extensions and their associated weapons based on specified search criteria.

- **Twenty-Second Query:** This query is useful for retrieving information

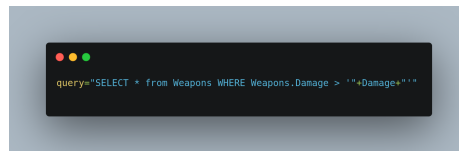
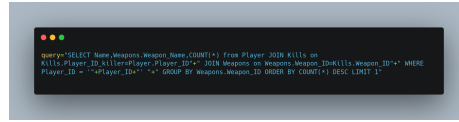


Figure 22: Query-22

about weapons with a damage value exceeding a specified threshold.

- **Twenty-Third Query:** Firstly, it specifies the columns you want to select: Name from the "Player" table, Weapon-Name from the "Weapons" table, and the count of occurrences of Weapon-ID. Then, we used join operations that combine rows from the "Player," "Kills," and "Weapons" tables based on the specified conditions. Then, it filters the results to only include rows where the Player-ID or Name in the "Player" table contains the specified value. Then, it groups the results by Player-ID and Weapon-ID, allowing you to count occurrences of each unique combination. Then order it in descending order and put the limit to 1.



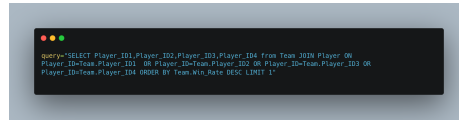
```

query="SELECT Name,Weapons,Weapons_Count(*) From Player JOIN KILLS ON
KILLS.Player_ID=KillsPlayer.Player_ID+" JOIN Weapons ON Weapons.Weapon_ID=" WHERE
Player_ID = "Player_ID" " GROUP BY Weapons.Weapon_ID ORDER BY COUNT(*) DESC LIMIT 1"

```

Figure 23: Query-23

- **Twenty-Fourth Query:** Firstly, it specifies the columns you want to



```

query="SELECT Player_ID1,Player_ID2,Player_ID3,Player_ID4 from Team JOIN Player ON
Player_ID=Team.Player_ID1 OR Player_ID=Team.Player_ID2 OR Player_ID=Team.Player_ID3 OR
Player_ID=Team.Player_ID4 GROUP BY Team.Win_Rate (DESC LIMIT 1)"

```

Figure 24: Query-24

select: the names of the players from the "Player" table (aliased as P1, P2, P3, and P4) in the "Team" table. Then we used join operations that combine rows from the "Team" table with corresponding rows from the "Player" table for each player in the team. Then it filters the results to only include rows where any of the player IDs or names in the team match the specified value.

- **Twenty-Fifth Query:** Firstly, it specifies that you want to retrieve the



```

query="SELECT Name,COUNT(*),Player.Total_Matches_Played as k1 from Player+" JOIN KILLS ON
KILLS.Player_ID=KillsPlayer.Player_ID GROUP BY Player.Player_ID HAVING "k1">=COUNT(*)"

```

Figure 25: Query-25

player names and the count of kills. Then we used join operation that combines rows from the "Player" table with corresponding rows from the "KILLS" table based on the specified condition and group the players by unique player IDs. Then it filters the grouped results to only include rows where the count of kills is greater than or equal to 1.5 times the total matches played by the player.

- **Twenty-Sixth Query:** First it indicates that data will be updated in the "Player" table. Then a player will be updated along with its new value on the condition that new player ID is equal to the old player ID.

A terminal window with a dark background and light gray text. The text is a SQL query: `query='UPDATE Player set Player_ID="'+New_Player_ID+"' where Player_ID='"+Player_ID+"''`. The window has three colored dots (red, yellow, green) in the top left corner.

```
query='UPDATE Player set Player_ID="'+New_Player_ID+"' where
Player_ID='"+Player_ID+"''
```

*Figure 26: Query-26*