Project 3
DOCUMENTATION:

There are several different functions that correspond to the different functionalities:

Note: Exec Query is used by every function except transactions. Transactions has its own executor that begins with the query "BEGIN" being executed and then the user can proceed with their tranaction
Note: welcome_interface is presented to the user when they first open the cli and it presents the list of options then calls the appropriate functionality handler based on the user input.

All outputs to the CLI are printed to the user for ease of use.

Now here are the different functionalites and their corresponding functions with explanations:

1. Insert Data

- insert_data function -
  inputs: isSubquery - Boolean, outputs: void or String
  - This function when called will ask the user what table they would like to insert into using a for loop that runs through the keys in the schema file where the schema of the database is defined. Then from there based on the users input, if it is not a valid input we will ask them for a new input until it is valid, otherwise we will ask them the value that they want to include for every parameter via a for loop and then once we have all the values we will either call the insert_data_executor which will prepare the query and execute it or we will return the query as a stringif the isSubquery boolean is true. The isSubquery boolean is only true when the subquery function is calling the insert function.

- insert_data_executor function -
  inputs: tableName - String, values - array of inputs from user, outputs: void
  - This function simply acts as an intermediary between the insert_data function and the exec_query functio. This function will take in the tablename and the values and then from there is transforms the array of values to a comma separated string and then calls exec_query on the prepared query with format INSERT INTO {s[tableName]} VALUES ({valuesToString})

INSERT EXAMPLE CAN BE SEEN BELOW

```
/Users/zakyoung/Desktop/proj2/venv/bin/python /Users/zakyoung/Desktop/proj2/main.py

        Welcome to the Database CLI Interface!
        Please select an option:
        1. Insert Data
        2. Delete Data
        3. Update Data
        4. Search Data
        5. Aggregate Functions
        6. Sorting
        7. Joins
        8. Grouping
        9. Subqueries
        10. Transactions
        11. Error Handling
        12. Exit

Enter your choice (1-12): 1
Please select the table that you would like to insert into:
1. team
2. combine
3. player
4. front_office
5. social_media
6. arena
7. draft
8. official
9. attributes
10. inactives
11. game
12. game_context
13. line_score
Please input the number of your choice: 1
Please enter a value for team_id: 30
Please enter a value for full_name: TEST FOR DOCUMENTATION
Please enter a value for abbreviation: TFD
Please enter a value for nickname: TEST
Please enter a value for year_founded: 2024
Successful operation
Would you like to do another operation (Y/N):
```

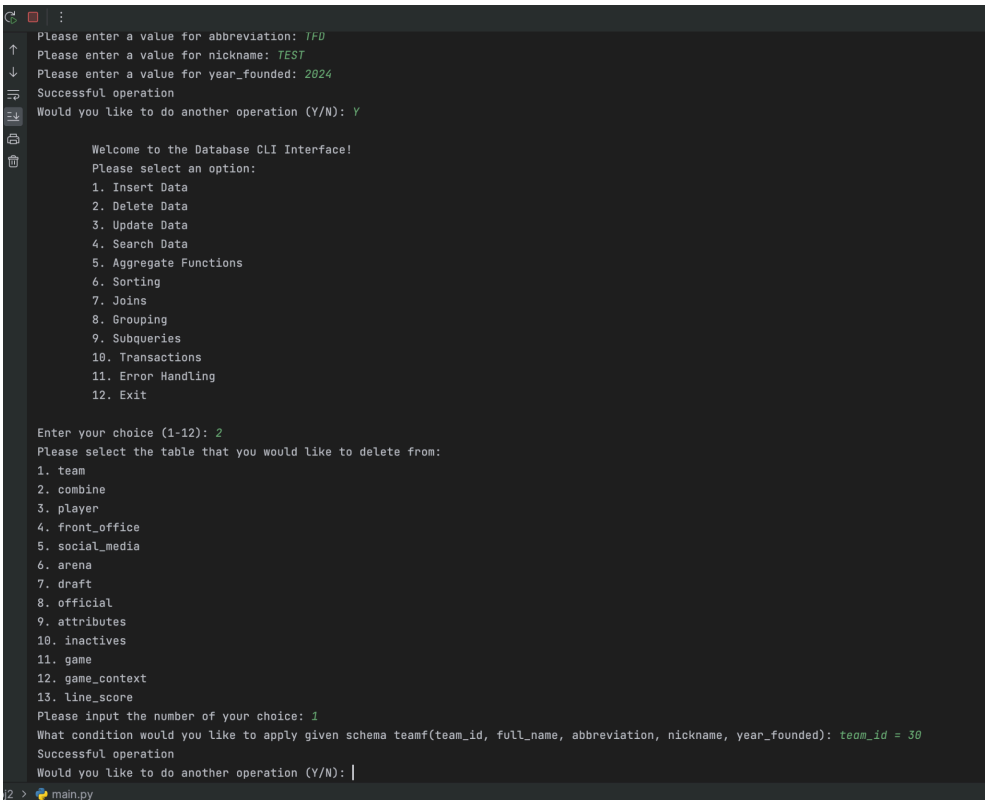
main.py

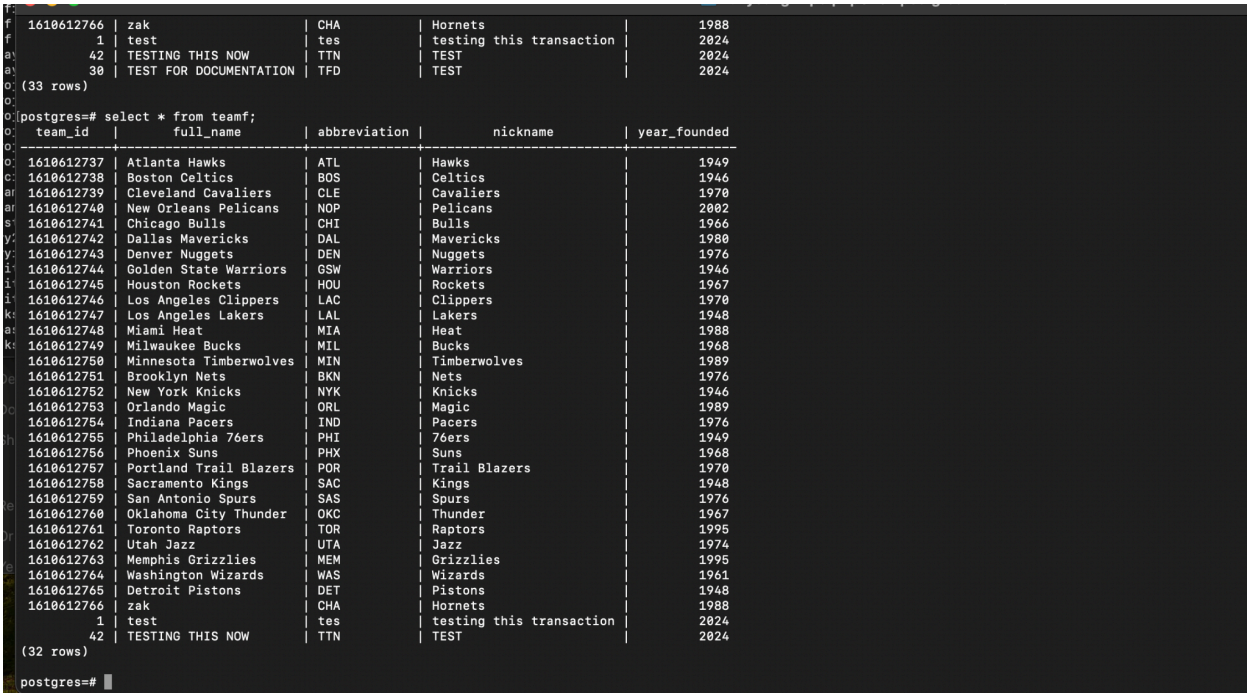
```
postgres=# select * from teamf;
  team_id   |       full_name        | abbreviation |         nickname         | year_founded
------------+------------------------+--------------+--------------------------+--------------
 1610612737 | Atlanta Hawks          | ATL          | Hawks                    |         1949
 1610612738 | Boston Celtics         | BOS          | Celtics                  |         1946
 1610612739 | Cleveland Cavaliers    | CLE          | Cavaliers                |         1970
 1610612740 | New Orleans Pelicans   | NOP          | Pelicans                 |         2002
 1610612741 | Chicago Bulls          | CHI          | Bulls                    |         1966
 1610612742 | Dallas Mavericks       | DAL          | Mavericks                |         1980
 1610612743 | Denver Nuggets         | DEN          | Nuggets                  |         1976
 1610612744 | Golden State Warriors  | GSW          | Warriors                 |         1946
 1610612745 | Houston Rockets        | HOU          | Rockets                  |         1967
 1610612746 | Los Angeles Clippers   | LAC          | Clippers                 |         1970
 1610612747 | Los Angeles Lakers     | LAL          | Lakers                   |         1948
 1610612748 | Miami Heat             | MIA          | Heat                     |         1988
 1610612749 | Milwaukee Bucks        | MIL          | Bucks                    |         1968
 1610612750 | Minnesota Timberwolves | MIN          | Timberwolves             |         1989
 1610612751 | Brooklyn Nets          | BKN          | Nets                     |         1976
 1610612752 | New York Knicks        | NYK          | Knicks                   |         1946
 1610612753 | Orlando Magic          | ORL          | Magic                    |         1989
 1610612754 | Indiana Pacers         | IND          | Pacers                   |         1976
 1610612755 | Philadelphia 76ers     | PHI          | 76ers                    |         1949
 1610612756 | Phoenix Suns           | PHX          | Suns                     |         1968
 1610612757 | Portland Trail Blazers | POR          | Trail Blazers            |         1970
 1610612758 | Sacramento Kings       | SAC          | Kings                    |         1948
 1610612759 | San Antonio Spurs      | SAS          | Spurs                    |         1976
 1610612760 | Oklahoma City Thunder  | OKC          | Thunder                  |         1967
 1610612761 | Toronto Raptors        | TOR          | Raptors                  |         1995
 1610612762 | Utah Jazz              | UTA          | Jazz                     |         1974
 1610612763 | Memphis Grizzlies      | MEM          | Grizzlies                |         1995
 1610612764 | Washington Wizards     | WAS          | Wizards                  |         1961
 1610612765 | Detroit Pistons        | DET          | Pistons                  |         1948
 1610612766 | zak                    | CHA          | Hornets                  |         1988
          1 | test                   | tes          | testing this transaction |         2024
         42 | TESTING THIS NOW       | TTN          | TEST                     |         2024
         30 | TEST FOR DOCUMENTATION | TFD          | TEST                     |         2024
(33 rows)

postgres=#
```

2. Delete Data
    - delete_data function
      inputs: isSubquery - Boolean, outputs: void or String
        - This function first asks what table they want to delete data from and then we use a for loop to present them the options from there we take in their choice until they provide a valid choice and then we ask them on what condition they want to delete entries from the table on. Once we take in this info if isSubquery is false we call the delete_data_executor function otherwise we return the final query as a String.

    - delete_data_executor function
      inputs: tableName: String ,condition: String, outputs: void
      This function simply just inserts tableName and condition into the formatted query and then calls exec_query on this query.

DELETE EXAMPLE CAN BE SEEN BELOW:





```
 1610612766 | zak                    | CHA          | Hornets                 |         1988
          1 | test                   | tes          | testing this transaction |         2024
         42 | TESTING THIS NOW       | TTN          | TEST                    |         2024
         30 | TEST FOR DOCUMENTATION | TFD          | TEST                    |         2024
(33 rows)

postgres=# select * from teamf;
  team_id   |       full_name        | abbreviation |         nickname         | year_founded
------------+------------------------+--------------+--------------------------+--------------
 1610612737 | Atlanta Hawks          | ATL          | Hawks                    |         1949
 1610612738 | Boston Celtics         | BOS          | Celtics                  |         1946
 1610612739 | Cleveland Cavaliers    | CLE          | Cavaliers                |         1970
 1610612740 | New Orleans Pelicans   | NOP          | Pelicans                 |         2002
 1610612741 | Chicago Bulls          | CHI          | Bulls                    |         1966
 1610612742 | Dallas Mavericks       | DAL          | Mavericks                |         1980
 1610612743 | Denver Nuggets         | DEN          | Nuggets                  |         1976
 1610612744 | Golden State Warriors  | GSW          | Warriors                 |         1946
 1610612745 | Houston Rockets        | HOU          | Rockets                  |         1967
 1610612746 | Los Angeles Clippers   | LAC          | Clippers                 |         1970
 1610612747 | Los Angeles Lakers     | LAL          | Lakers                   |         1948
 1610612748 | Miami Heat             | MIA          | Heat                     |         1988
 1610612749 | Milwaukee Bucks        | MIL          | Bucks                    |         1968
 1610612750 | Minnesota Timberwolves | MIN          | Timberwolves             |         1989
 1610612751 | Brooklyn Nets          | BKN          | Nets                     |         1976
 1610612752 | New York Knicks        | NYK          | Knicks                   |         1946
 1610612753 | Orlando Magic          | ORL          | Magic                    |         1989
 1610612754 | Indiana Pacers         | IND          | Pacers                   |         1976
 1610612755 | Philadelphia 76ers     | PHI          | 76ers                    |         1949
 1610612756 | Phoenix Suns           | PHX          | Suns                     |         1968
 1610612757 | Portland Trail Blazers | POR          | Trail Blazers            |         1970
 1610612758 | Sacramento Kings       | SAC          | Kings                    |         1948
 1610612759 | San Antonio Spurs      | SAS          | Spurs                    |         1976
 1610612760 | Oklahoma City Thunder  | OKC          | Thunder                  |         1967
 1610612761 | Toronto Raptors        | TOR          | Raptors                  |         1995
 1610612762 | Utah Jazz              | UTA          | Jazz                     |         1974
 1610612763 | Memphis Grizzlies      | MEM          | Grizzlies                |         1995
 1610612764 | Washington Wizards     | WAS          | Wizards                  |         1961
 1610612765 | Detroit Pistons        | DET          | Pistons                  |         1948
 1610612766 | zak                    | CHA          | Hornets                  |         1988
          1 | test                   | tes          | testing this transaction |         2024
         42 | TESTING THIS NOW       | TTN          | TEST                     |         2024
(32 rows)

postgres=#
```

3. Update Data
    - update_data function
      inputs: isSubquery - Boolean, outputs: void or String
        - This function will ask the user which table they would like to update from, take this input until they give a valid answer, and then from their will request the colum that they want to modify in and then request the new value that they want to be assigned and then finally the condition that they want to update on. Once we all all this information from the user we either return the formatted string if this is a subquery or we call update_data_executor with all the information we just collected
    - update_date_executor function
      inputs: tableName:tableName:String, columnToChange:String ,newValue:String,condition: String outputs: void
      This function simply just inserts tableName, columnToChange, newValue, and condition to the query defined as UPDATE {tableName} SET {columnToChange} = \'{newValue}\' WHERE {condition}" and then from there calls the exec_query function to execute the query.

    UPDATE EXAMPLE CAN BE SEEN BELOW:

```
        1. Insert Data
        2. Delete Data
        3. Update Data
        4. Search Data
        5. Aggregate Functions
        6. Sorting
        7. Joins
        8. Grouping
        9. Subqueries
        10. Transactions
        11. Error Handling
        12. Exit

Enter your choice (1-12): 3
Please select the table that you would like to update from:
1. team
2. combine
3. player
4. front_office
5. social_media
6. arena
7. draft
8. official
9. attributes
10. inactives
11. game
12. game_context
13. line_score
Please input the number of your choice: 1
For Column:
1: team_id
2: full_name
3: abbreviation
4: nickname
5: year_founded
Which column would you like to modify: 2
Enter the new value you want to be assigned: NAME CHANGE
What condition would you like to apply given schema teamf(team_id, full_name, abbreviation, nickname, year_founded): team_id = 42
Successful operation
Would you like to do another operation (Y/N):
```

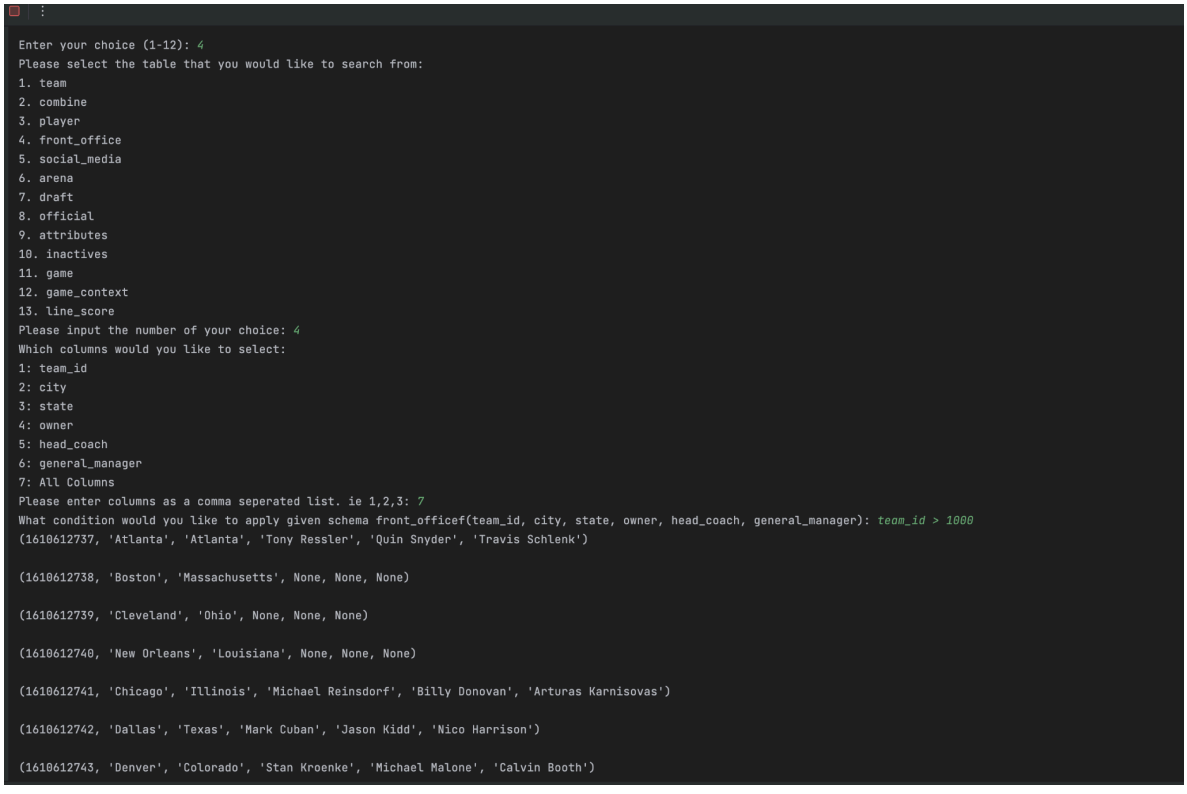
main.py

zakyoung — psql -p5432 postgres — 223×44

```
1610612765 | Detroit Pistons         | DET          | Pistons                 |         1948
1610612766 | zak                     | CHA          | Hornets                 |         1988
         1 | test                    | tes          | testing this transaction |         2024
        42 | TESTING THIS NOW        | TTN          | TEST                    |         2024
32 rows)

ostgres=# select * from teamf;
 team_id   |        full_name         | abbreviation |         nickname         | year_founded
-----------+--------------------------+--------------+--------------------------+--------------
1610612737 | Atlanta Hawks            | ATL          | Hawks                    |         1949
1610612738 | Boston Celtics           | BOS          | Celtics                  |         1946
1610612739 | Cleveland Cavaliers      | CLE          | Cavaliers                |         1970
1610612740 | New Orleans Pelicans     | NOP          | Pelicans                 |         2002
1610612741 | Chicago Bulls            | CHI          | Bulls                    |         1966
1610612742 | Dallas Mavericks         | DAL          | Mavericks                |         1980
1610612743 | Denver Nuggets           | DEN          | Nuggets                  |         1976
1610612744 | Golden State Warriors    | GSW          | Warriors                 |         1946
1610612745 | Houston Rockets          | HOU          | Rockets                  |         1967
1610612746 | Los Angeles Clippers     | LAC          | Clippers                 |         1970
1610612747 | Los Angeles Lakers       | LAL          | Lakers                   |         1948
1610612748 | Miami Heat               | MIA          | Heat                     |         1988
1610612749 | Milwaukee Bucks          | MIL          | Bucks                    |         1968
1610612750 | Minnesota Timberwolves   | MIN          | Timberwolves             |         1989
1610612751 | Brooklyn Nets            | BKN          | Nets                     |         1976
1610612752 | New York Knicks          | NYK          | Knicks                   |         1946
1610612753 | Orlando Magic            | ORL          | Magic                    |         1989
1610612754 | Indiana Pacers           | IND          | Pacers                   |         1976
1610612755 | Philadelphia 76ers       | PHI          | 76ers                    |         1949
1610612756 | Phoenix Suns             | PHX          | Suns                     |         1968
1610612757 | Portland Trail Blazers   | POR          | Trail Blazers            |         1970
1610612758 | Sacramento Kings         | SAC          | Kings                    |         1948
1610612759 | San Antonio Spurs        | SAS          | Spurs                    |         1976
1610612760 | Oklahoma City Thunder    | OKC          | Thunder                  |         1967
1610612761 | Toronto Raptors          | TOR          | Raptors                  |         1995
1610612762 | Utah Jazz                | UTA          | Jazz                     |         1974
1610612763 | Memphis Grizzlies        | MEM          | Grizzlies                |         1995
1610612764 | Washington Wizards       | WAS          | Wizards                  |         1961
1610612765 | Detroit Pistons          | DET          | Pistons                  |         1948
1610612766 | zak                      | CHA          | Hornets                  |         1988
         1 | test                     | tes          | testing this transaction |         2024
        42 | NAME CHANGE              | TTN          | TEST                     |         2024
32 rows)

ostgres=#
```

4. Search Data
    - search_data function
      inputs: isSubquery - Boolean, outputs: void or String
        - This function will first act the user which table they want to search from and then will present a list of tables for the user to decide from. Then from there we will ask the user to select the columns that they want to include in the search. We present this to them telling them to enter the numbers of the columns that they want to include from the numbered list of columns. From there we ask what condition they want to apply and we present the schema to them so that they can easily input the condition they want to select on or they could provide the final option which will allow them to select All Columns. Then finally it will either return a string if isSubquery is true otherwise it will call search_data_executor to do the execution of the query.
    - search_data_executor
      Inputs: tableName: String,condition: String, colsn: String, outputs: void
        - This function simply just inserts tableName, condition, and cols into a string and then passes the query of format SELECT {cols} FROM {tableName} WHERE {condition} to the exec_query function.

SEARCH EXAMPLE CAN BE SEEN BELOW:



5. Aggregate Functions

Aggregate_data function

inputs: isSubquery - Boolean, outputs: void or String

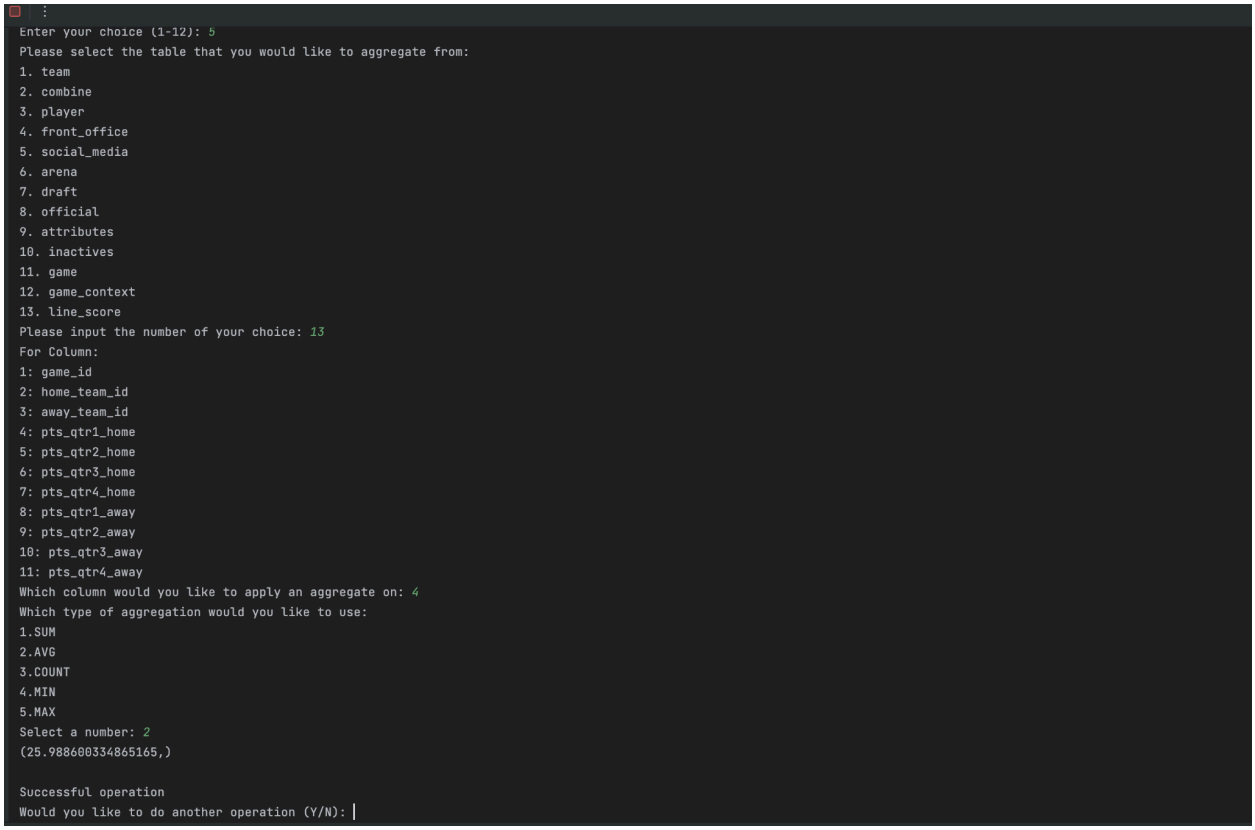
This function first asks the user what table they want to aggregate from and then from there has error handling to make sure the the user enters a valid table, after that we get the column that the user wants to apply an aggregate on. Then finally we ask the user what type of aggregate they want to use from the options of SUM, AVG, COUNT, MIN, and MAX. Then after this we either return the string query is isSubquery is true or we execute teh query using the aggregate_functions_executor function. The query that is produced has the following format SELECT {aggy}({column}) FROM {chosen}

Aggregate_functions_executor function

Inputs: tablesName: String ,column: String ,calculationType:String outputs: void

This function simply just creates the query and then executes it by calling exec_query on the query.

AGGREGATE EXAMPLE CAN BE SEEN BELOW:



In this example we get the average number of points scored by the home team in the first quarter which turns out to be 25.98 given our dataset.

6. Sorting

   Sorting function

   inputs: isSubquery - Boolean, outputs: void or String

   The sorting function first asks the user what table they want to sort from and then from there it will ask them what column that they want to sort on and provide them a list of column names corresponding to the table. Then finally it will ask if they want to sort by ASC or DESC order and then will either execute the query or return the string of the query

   Sorting_executor function

   Inputs: tableName: String,column: String ,sortBy: String, outputs: void

   This function simply builds the query and then calls exec_query on it

   SORTING EXAMPLE CAN BE SEEN BELOW

```
Successful operation
Would you like to do another operation (Y/N): Y

        Welcome to the Database CLI Interface!
        Please select an option:
        1. Insert Data
        2. Delete Data
        3. Update Data
        4. Search Data
        5. Aggregate Functions
        6. Sorting
        7. Joins
        8. Grouping
        9. Subqueries
        10. Transactions
        11. Error Handling
        12. Exit

Enter your choice (1-12): 6
Please select the table that you would like to sort from:
1. team
2. combine
3. player
4. front_office
5. social_media
6. arena
7. draft
8. official
9. attributes
10. inactives
11. game
12. game_context
13. line_score
Please input the number of your choice: 6
For Column:
1: team_id
2: arena_name
3: arena_capacity
Which column would you like to apply sort on: 3
Which type of ordering would you like to use:
```

```
2.DESC
Select a number: 2
(1610612760, 'Paycom Center', None)

(1610612738, None, None)

(1610612739, None, None)

(1610612740, None, None)

(1610612765, 'Little Caesars Arena', None)

(1610612751, 'Barclays Center', None)

(1610612752, None, None)

(1610612753, None, None)

(1610612762, 'Delta Center', None)

(1610612755, 'Wells Fargo Center', None)

(1610612756, 'Footprint Center', None)

(1610612761, 'Scotiabank Arena', None)

(1610612743, 'Ball Arena', None)

(1610612744, 'Chase Center', None)

(1610612741, 'United Center', 21711.0)

(1610612764, 'Capital One Arena', 20647.0)

(1610612757, 'Moda Center', 19980.0)

(1610612748, 'Kaseya Center', 19600.0)

(1610612750, 'Target Center', 19356.0)
```

```
(1610612761, 'Scotiabank Arena', None)

(1610612743, 'Ball Arena', None)

(1610612744, 'Chase Center', None)

(1610612741, 'United Center', 21711.0)

(1610612764, 'Capital One Arena', 20647.0)

(1610612757, 'Moda Center', 19980.0)

(1610612748, 'Kaseya Center', 19600.0)

(1610612750, 'Target Center', 19356.0)

(1610612742, 'American Airlines Center', 19200.0)

(1610612746, 'Crypto.com Arena', 19060.0)

(1610612747, 'Crypto.com Arena', 19060.0)

(1610612766, 'Spectrum Center', 19026.0)

(1610612737, 'State Farm Arena', 18729.0)

(1610612759, 'AT&T Center', 18694.0)

(1610612754, 'Gainbridge Fieldhouse ', 18345.0)

(1610612763, 'FedExForum', 18119.0)

(1610612745, 'Toyota Center', 18104.0)

(1610612749, 'Fiserv Forum', 17500.0)

(1610612758, 'Golden 1 Center', 17500.0)

Successful operation
Would you like to do another operation (Y/N):
```

7. Joins

Joins function

inputs: isSubquery - Boolean, outputs: void or String

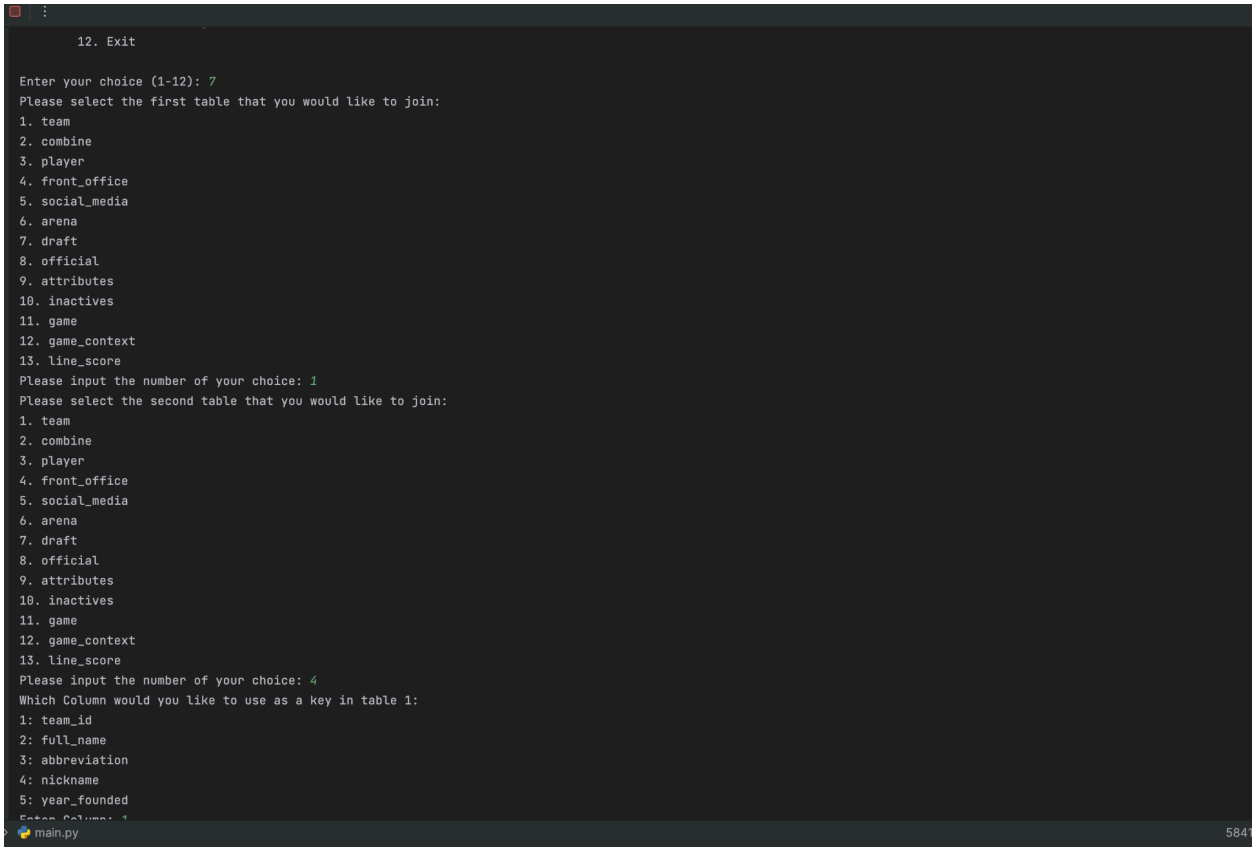
This function will first ask the user the first table that they want to join on, then it will ask them the second table that they want to join on and this information will be stored. Then for both tables we will ask them what the key that they want to join on should be and then finally either the joins_executor will be called and the query will be assembled there or the query will be returned if it is a subquery

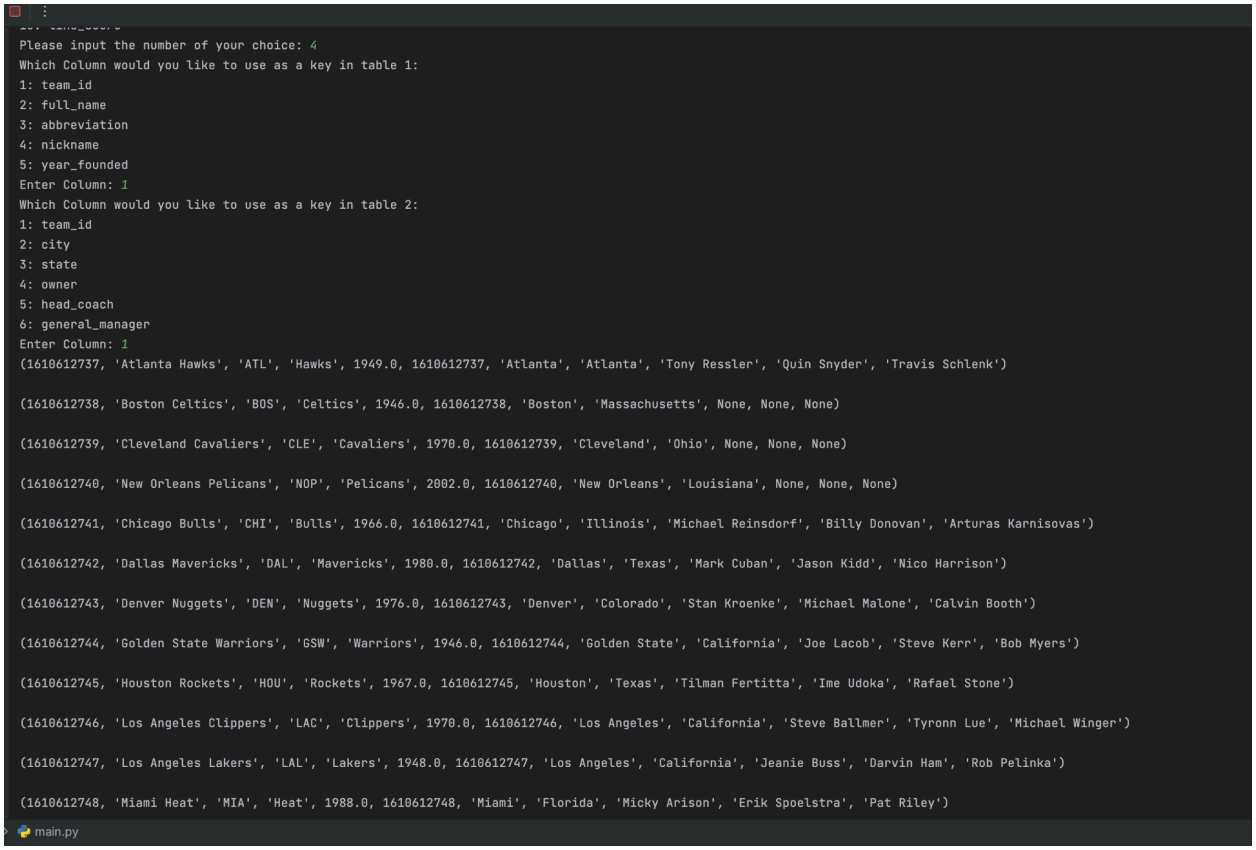
Joins_executor function

Inputs: table1: String,table2: String,key1: String,key2: String): outputs: void

This function is called by the joins function when it is not apart of a subqeuery and will simply call exec_query with the created query.

JOIN EXAMPLE USAGE CAN BE SEEN BELOW:





8. Grouping

Grouping function

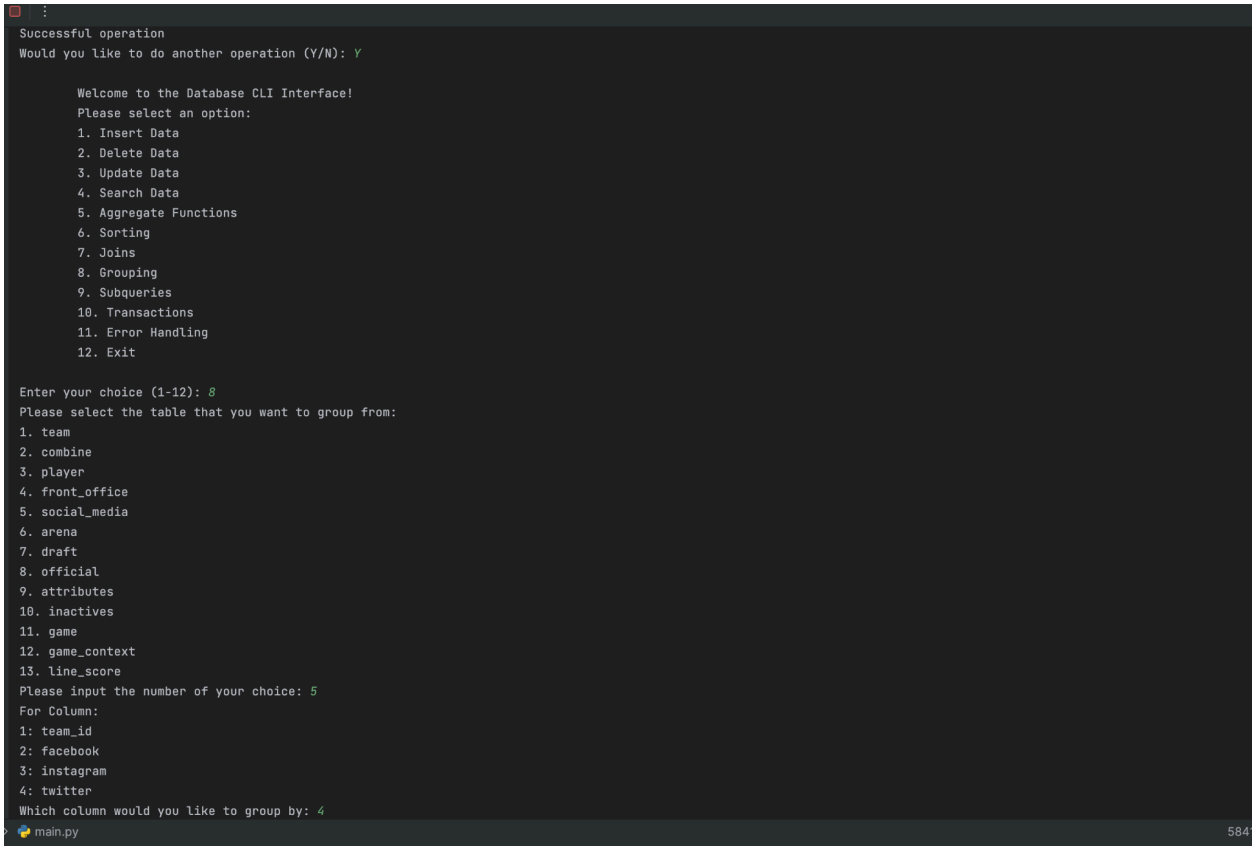
inputs: isSubquery - Boolean, outputs: void or String

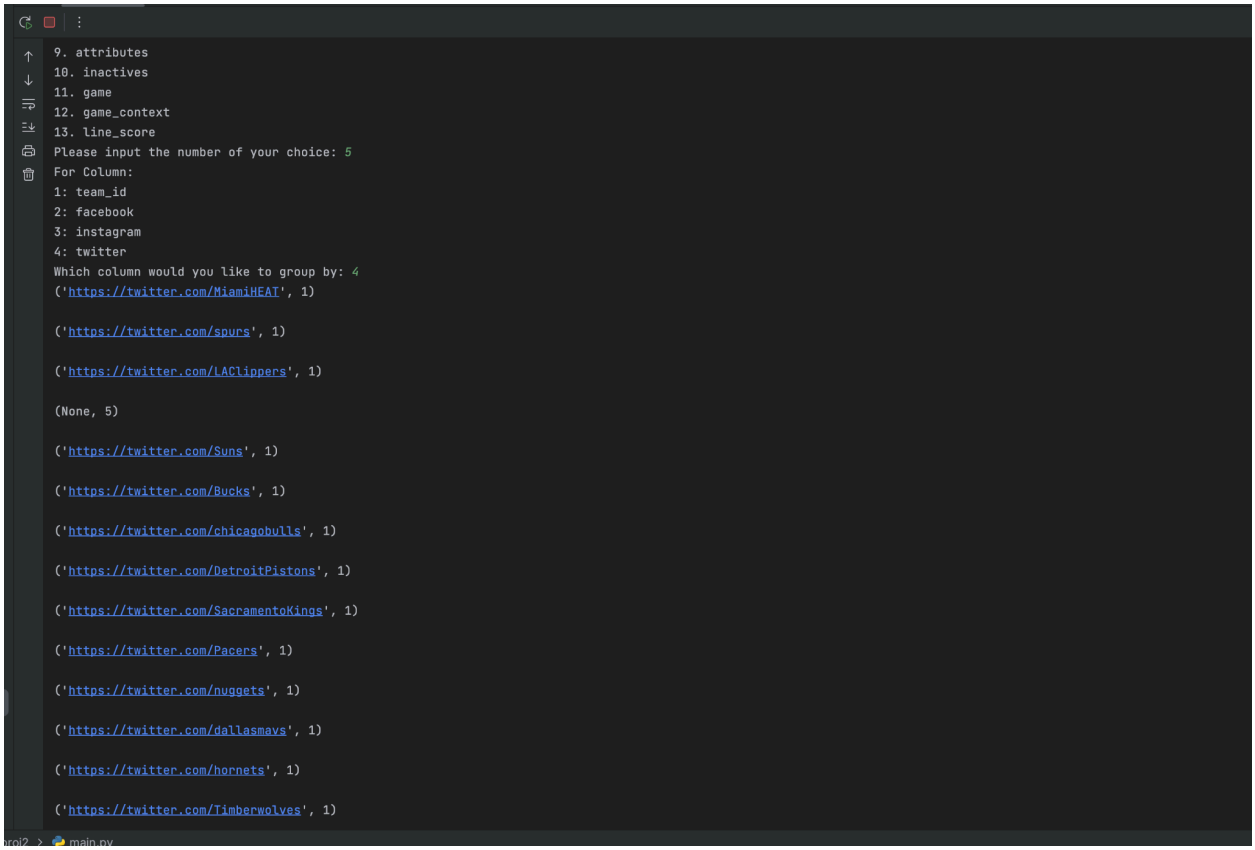
The grouping function first asks the user what table they want to group from and then from there will ask the user what column they want to group by and then it will either return the query or call grouping_executor.

Grouping_executor function

Inputs: column1: String,tableName: String outputs: void

This function simply just builds the query from the inputs and then calls exec_query on the created query.


Successful operation
Would you like to do another operation (Y/N): Y
Welcome to the Database CLI Interface!
Please select an option:
1. Insert Data
2. Delete Data
3. Update Data
4. Search Data
5. Aggregate Functions
6. Sorting
7. Joins
8. Grouping
9. Subqueries
10. Transactions
11. Error Handling
12. Exit
Enter your choice (1-12): 8
Please select the table that you want to group from:
1. team
2. combine
3. player
4. front_office
5. social_media
6. arena
7. draft
8. official
9. attributes
10. inactives
11. game
12. game_context
13. line_score
Please input the number of your choice: 5
For Column:
1: team_id
2: facebook
3: instagram
4: twitter
Which column would you like to group by: 4
main.py



9. Subqueries

Subqueries function

inputs: isSubquery - Boolean, outputs: void or String

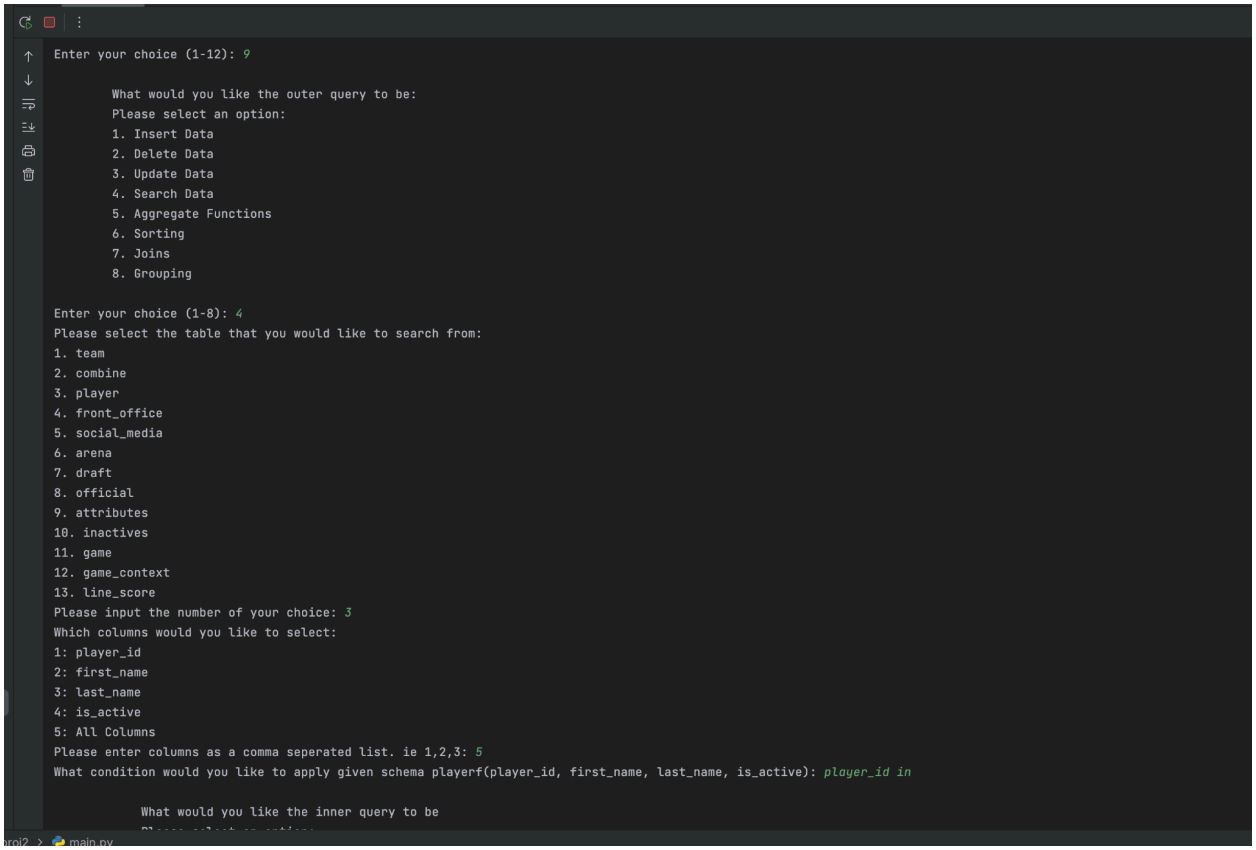
This function is the reason every other function has a isSubquery input, this function will first ask the user what they want the outer query to be made of and then from there it will call one of the above functions and then the cli will take the information in as if it was going to execute the query. Then it will return this information to the subqueries function and it will ask the user for the inner query and it will do the same thing. One of the cool things about this funcion is it can do a subquery in a subquery so it recursively can build any query. After the query is build it either returns the query if it is a subquery or it will call the subqueries executor with all of the arguments.

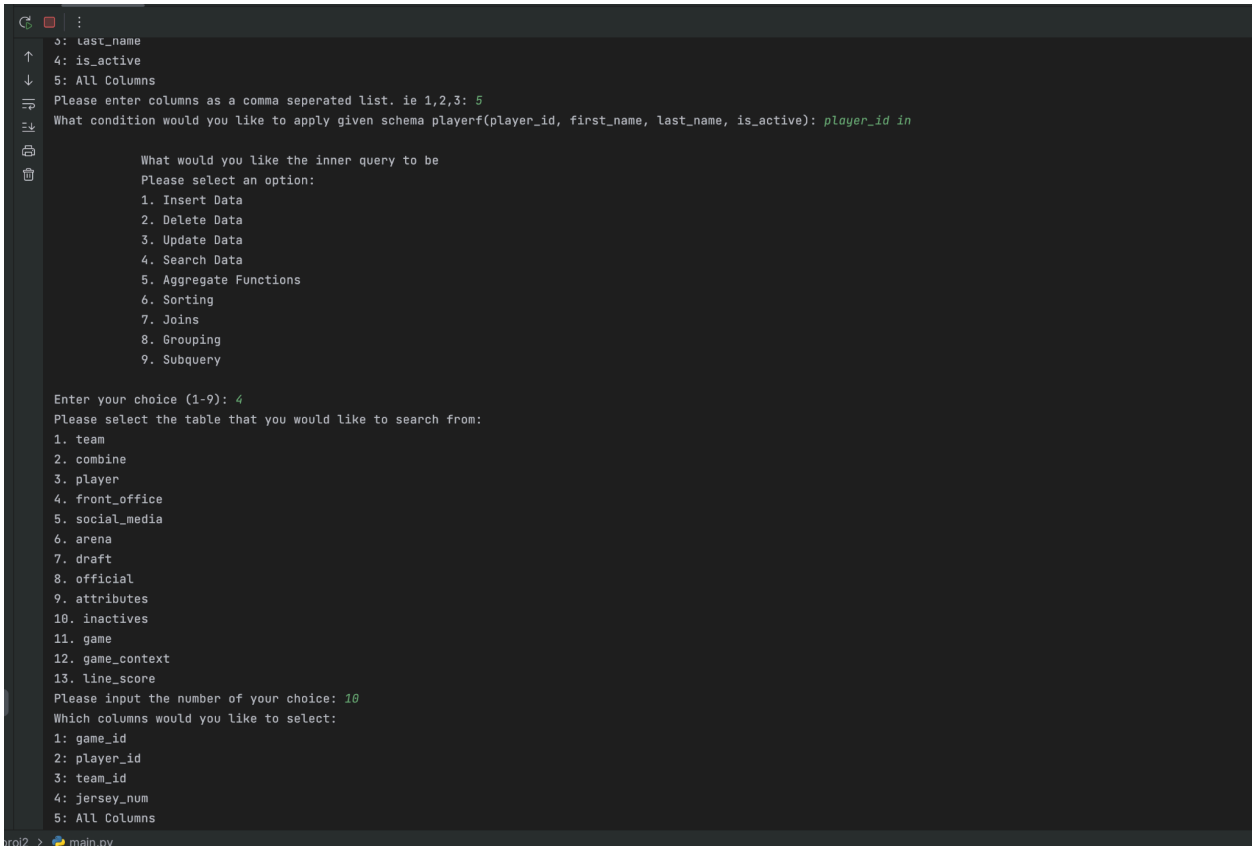
Subqueries_executor

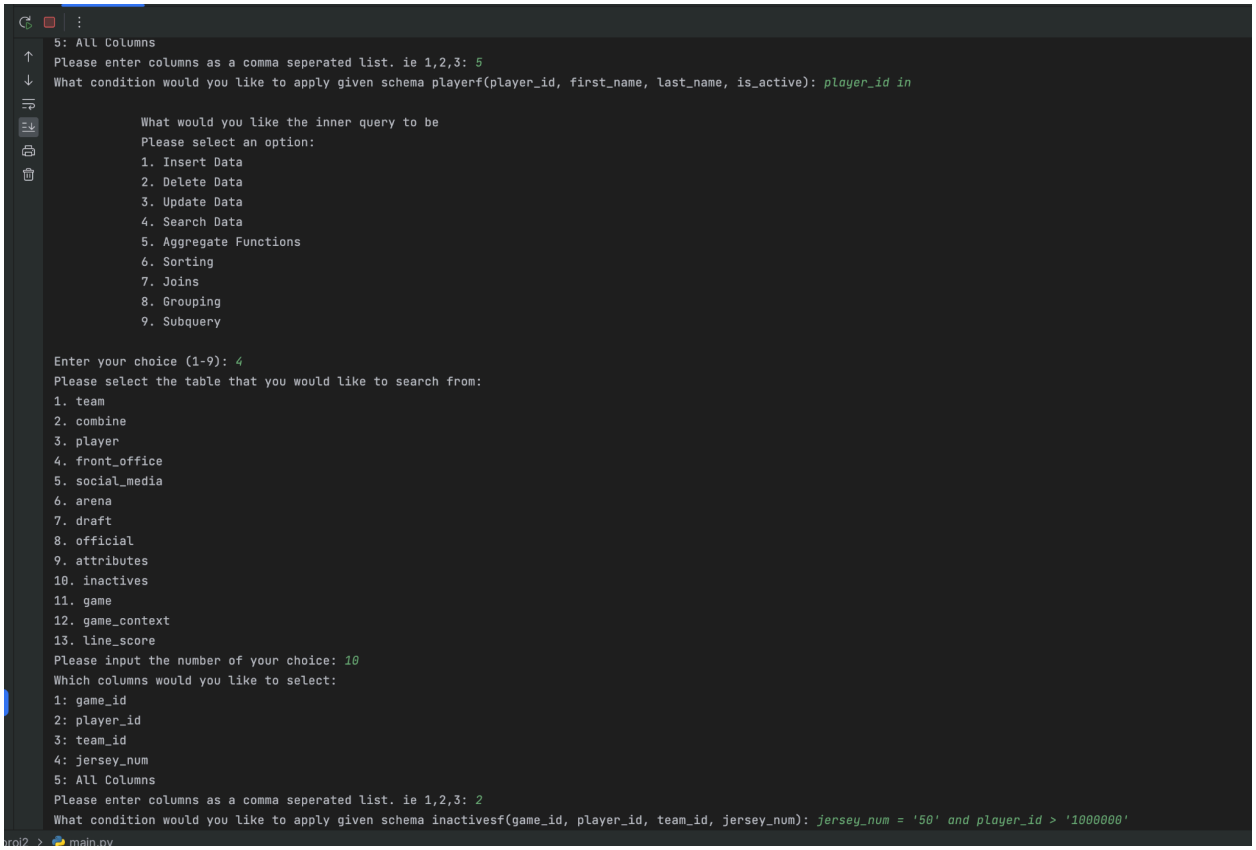
Inputs: outerQuery: String,innerQuery: String, outputs: void

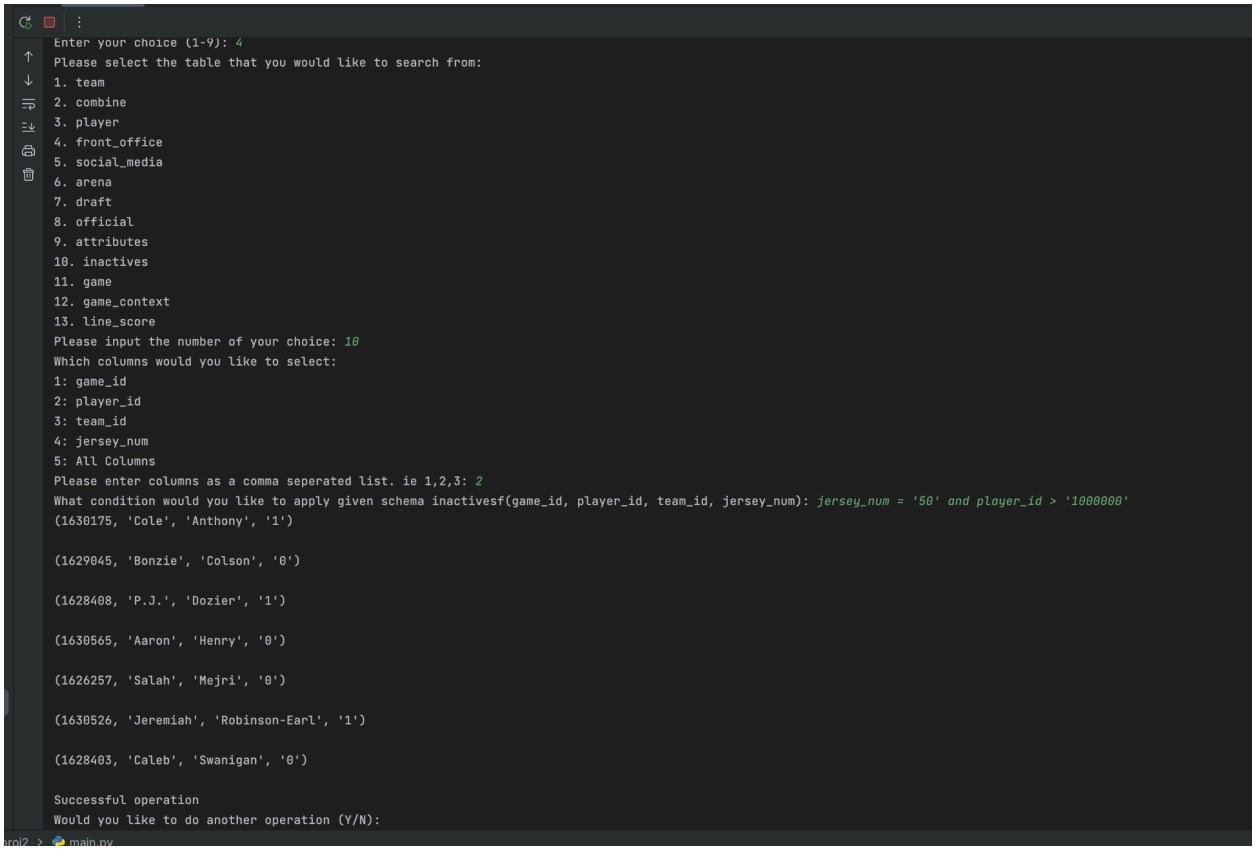
This function simply takes in the two queries and then calls exec_query on each one.

SUBQUERIES EXAMPLE CAN BE SEEN BELOW:


Enter your choice (1-12): 9
What would you like the outer query to be:
Please select an option:
1. Insert Data
2. Delete Data
3. Update Data
4. Search Data
5. Aggregate Functions
6. Sorting
7. Joins
8. Grouping
Enter your choice (1-8): 4
Please select the table that you would like to search from:
1. team
2. combine
3. player
4. front_office
5. social_media
6. arena
7. draft
8. official
9. attributes
10. inactives
11. game
12. game_context
13. line_score
Please input the number of your choice: 3
Which columns would you like to select:
1: player_id
2: first_name
3: last_name
4: is_active
5: All Columns
Please enter columns as a comma seperated list. ie 1,2,3: 5
What condition would you like to apply given schema playerf(player_id, first_name, last_name, is_active): player_id in
What would you like the inner query to be
main.py


3: last_name
4: is_active
5: All Columns
Please enter columns as a comma seperated list. ie 1,2,3: 5
What condition would you like to apply given schema playerf(player_id, first_name, last_name, is_active): player_id in
What would you like the inner query to be
Please select an option:
1. Insert Data
2. Delete Data
3. Update Data
4. Search Data
5. Aggregate Functions
6. Sorting
7. Joins
8. Grouping
9. Subquery
Enter your choice (1-9): 4
Please select the table that you would like to search from:
1. team
2. combine
3. player
4. front_office
5. social_media
6. arena
7. draft
8. official
9. attributes
10. inactives
11. game
12. game_context
13. line_score
Please input the number of your choice: 10
Which columns would you like to select:
1: game_id
2: player_id
3: team_id
4: jersey_num
5: All Columns
proj2 > main.py


5: All Columns
Please enter columns as a comma seperated list. ie 1,2,3: 5
What condition would you like to apply given schema playerf(player_id, first_name, last_name, is_active): player_id in
What would you like the inner query to be
Please select an option:
1. Insert Data
2. Delete Data
3. Update Data
4. Search Data
5. Aggregate Functions
6. Sorting
7. Joins
8. Grouping
9. Subquery
Enter your choice (1-9): 4
Please select the table that you would like to search from:
1. team
2. combine
3. player
4. front_office
5. social_media
6. arena
7. draft
8. official
9. attributes
10. inactives
11. game
12. game_context
13. line_score
Please input the number of your choice: 10
Which columns would you like to select:
1: game_id
2: player_id
3: team_id
4: jersey_num
5: All Columns
Please enter columns as a comma seperated list. ie 1,2,3: 2
What condition would you like to apply given schema inactivesf(game_id, player_id, team_id, jersey_num): jersey_num = '50' and player_id > '1000000'
main.py



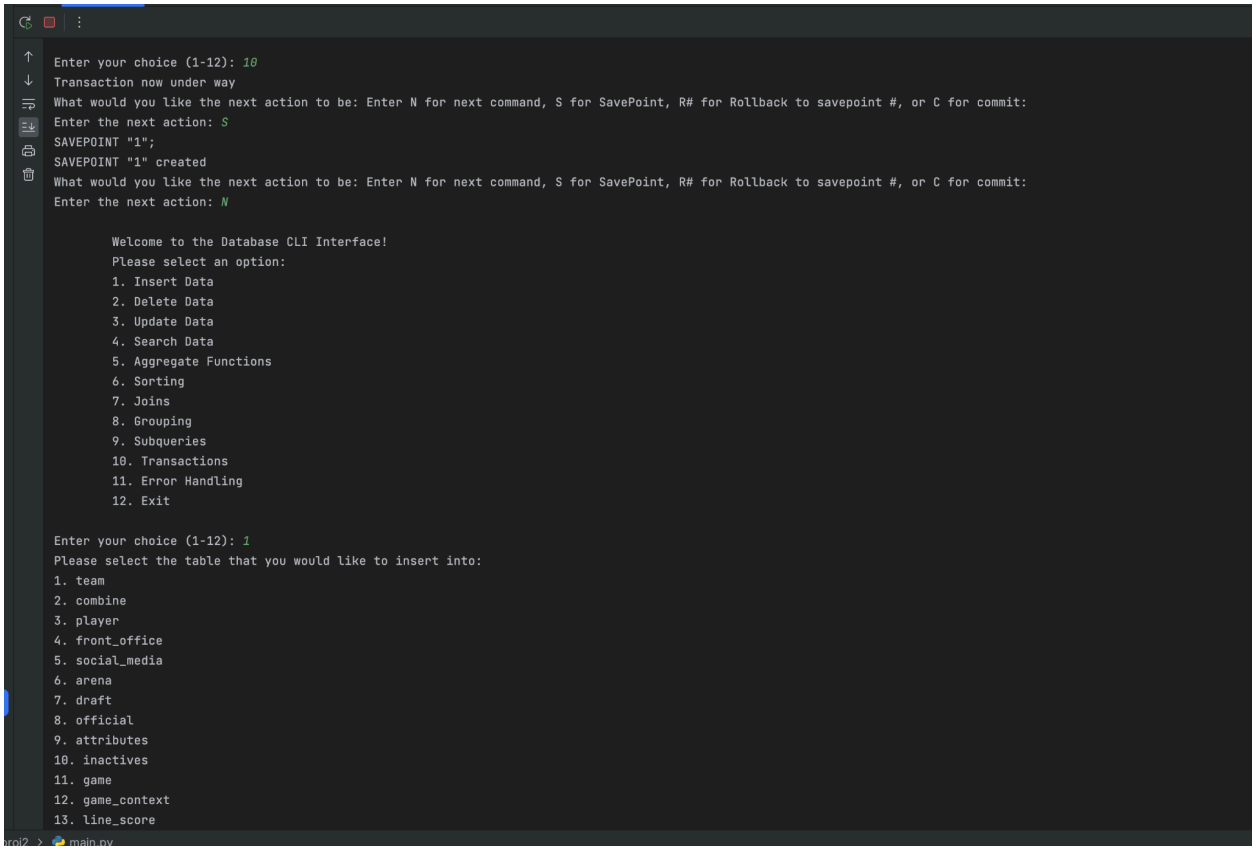
10. Transactions

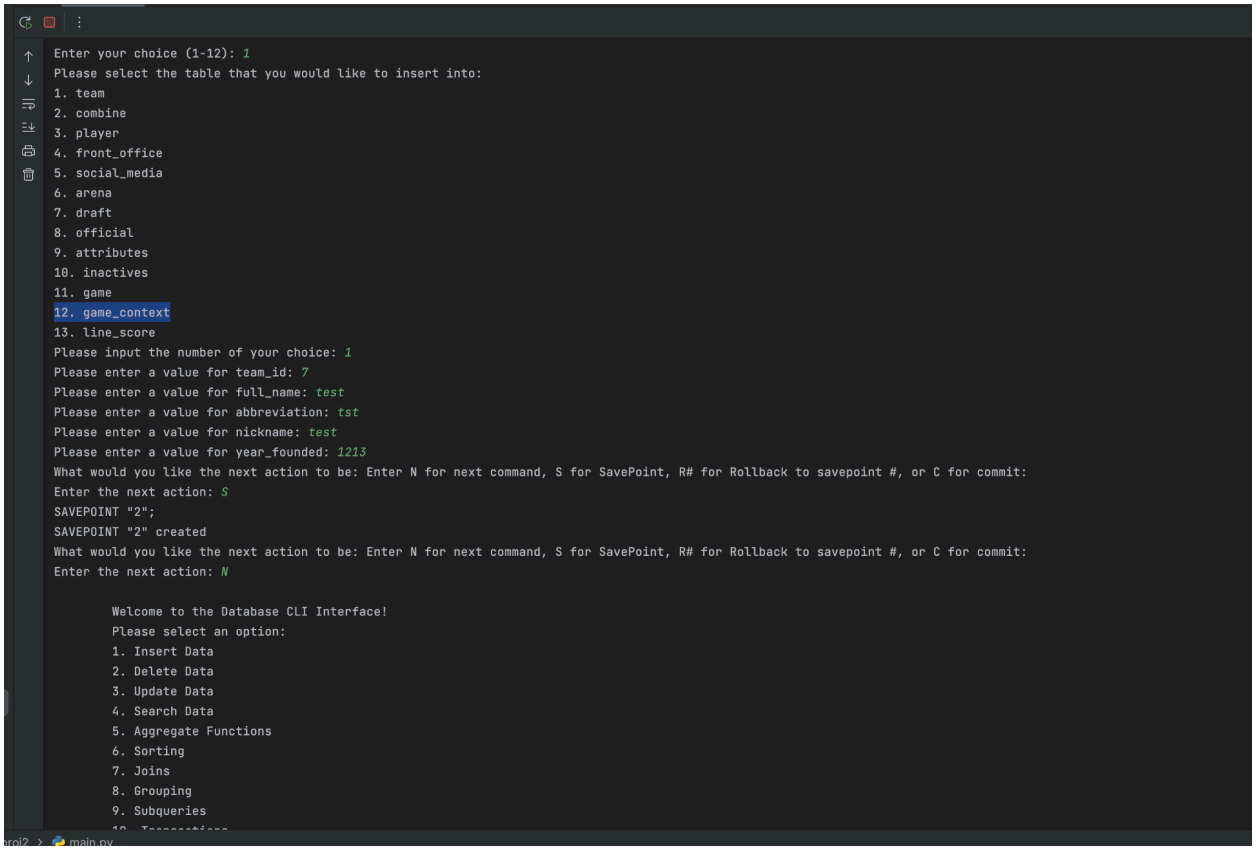
Transactions function

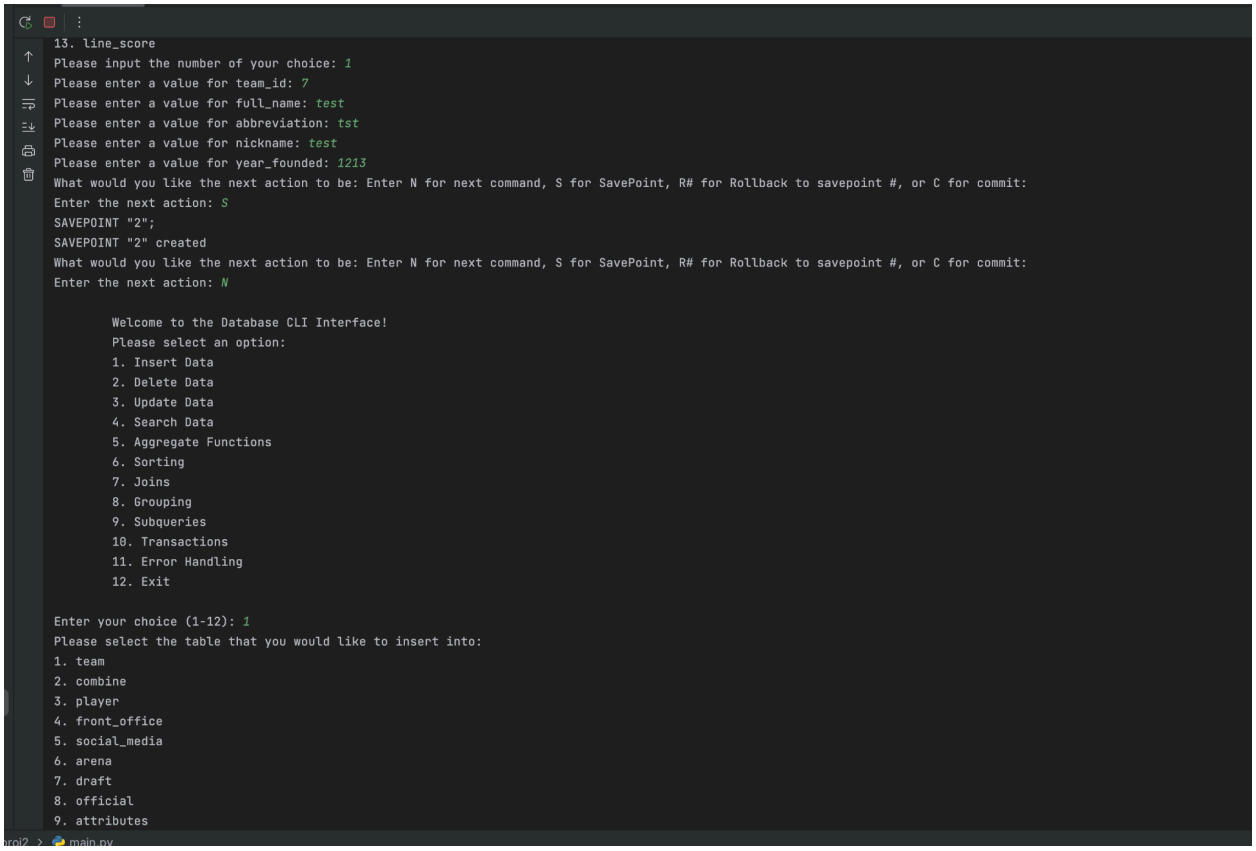
Inputs: Void outputs: void

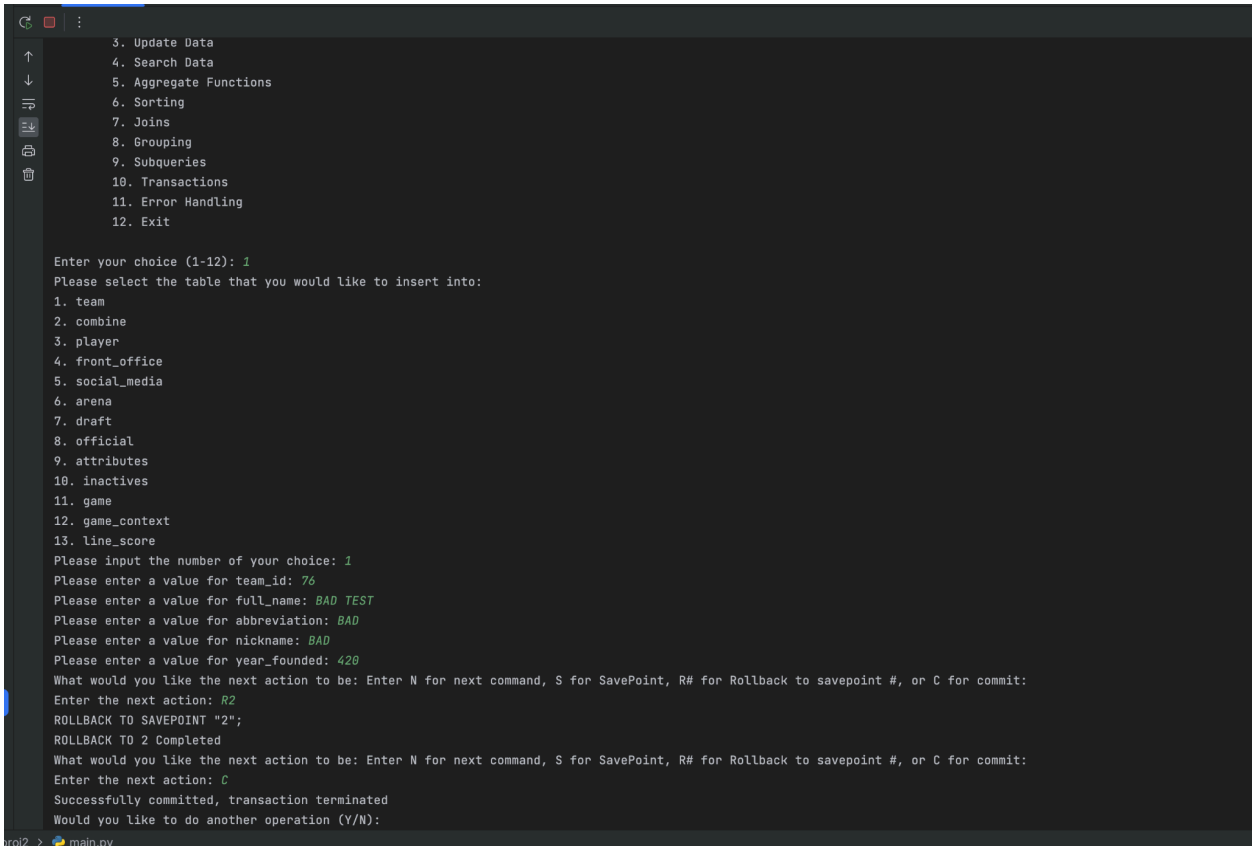
This is the only function that does not call exec_query but rather establishes its own connection which gives it more control over how the transaction is processed. We use a while loop that after executing any of the above function will ask the user if they want to enter another command, add a savepoint, rollback or commit. Then after all of the queries have been executed the connection will be committed and the connection will be closed and a message will be presented to the user if it was successful. Otherwise we employ error handling to catch anty database errors.

TRANSACTIONS EXAMPLE CAN BE SEEN BELOW:


Enter your choice (1-12): 10
Transaction now under way
What would you like the next action to be: Enter N for next command, S for SavePoint, R# for Rollback to savepoint #, or C for commit:
Enter the next action: S
SAVEPOINT "1";
SAVEPOINT "1" created
What would you like the next action to be: Enter N for next command, S for SavePoint, R# for Rollback to savepoint #, or C for commit:
Enter the next action: N
Welcome to the Database CLI Interface!
Please select an option:
1. Insert Data
2. Delete Data
3. Update Data
4. Search Data
5. Aggregate Functions
6. Sorting
7. Joins
8. Grouping
9. Subqueries
10. Transactions
11. Error Handling
12. Exit
Enter your choice (1-12): 1
Please select the table that you would like to insert into:
1. team
2. combine
3. player
4. front_office
5. social_media
6. arena
7. draft
8. official
9. attributes
10. inactives
11. game
12. game_context
13. line_score


Enter your choice (1-12): 1
Please select the table that you would like to insert into:
1. team
2. combine
3. player
4. front_office
5. social_media
6. arena
7. draft
8. official
9. attributes
10. inactives
11. game
12. game_context
13. line_score
Please input the number of your choice: 1
Please enter a value for team_id: 7
Please enter a value for full_name: test
Please enter a value for abbreviation: tst
Please enter a value for nickname: test
Please enter a value for year_founded: 1213
What would you like the next action to be: Enter N for next command, S for SavePoint, R# for Rollback to savepoint #, or C for commit:
Enter the next action: S
SAVEPOINT "2";
SAVEPOINT "2" created
What would you like the next action to be: Enter N for next command, S for SavePoint, R# for Rollback to savepoint #, or C for commit:
Enter the next action: N
Welcome to the Database CLI Interface!
Please select an option:
1. Insert Data
2. Delete Data
3. Update Data
4. Search Data
5. Aggregate Functions
6. Sorting
7. Joins
8. Grouping
9. Subqueries

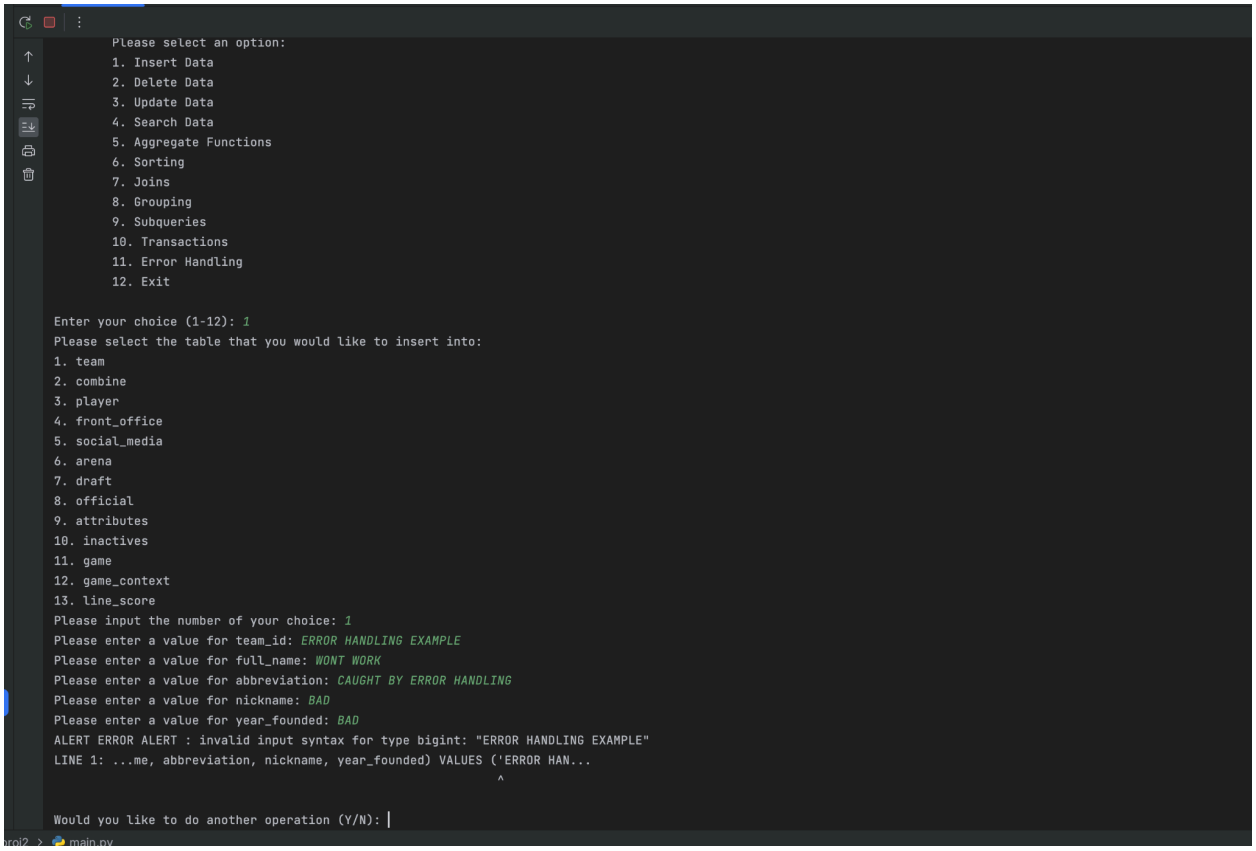

13. line_score
Please input the number of your choice: 1
Please enter a value for team_id: 7
Please enter a value for full_name: test
Please enter a value for abbreviation: tst
Please enter a value for nickname: test
Please enter a value for year_founded: 1213
What would you like the next action to be: Enter N for next command, S for SavePoint, R# for Rollback to savepoint #, or C for commit:
Enter the next action: S
SAVEPOINT "2";
SAVEPOINT "2" created
What would you like the next action to be: Enter N for next command, S for SavePoint, R# for Rollback to savepoint #, or C for commit:
Enter the next action: N
Welcome to the Database CLI Interface!
Please select an option:
1. Insert Data
2. Delete Data
3. Update Data
4. Search Data
5. Aggregate Functions
6. Sorting
7. Joins
8. Grouping
9. Subqueries
10. Transactions
11. Error Handling
12. Exit
Enter your choice (1-12): 1
Please select the table that you would like to insert into:
1. team
2. combine
3. player
4. front_office
5. social_media
6. arena
7. draft
8. official
9. attributes
main.py



11. Error Handling

For error handling we have our exec_query function wrapped in a try catch so that if we have any issues with the database we can catch the exception and print it out to the user. Along with this in many places if the user gives the cli and incorrect input we will ask the user for the correct input until they give it.”

OUTPUT FROM ERROR HANDLING CAN BE SEEN BELOW:
As we see here team_id must be a bigint not a VARCHAR



12. Exit
    - This just thanksthe use for using the cli interface and then terminates the cli