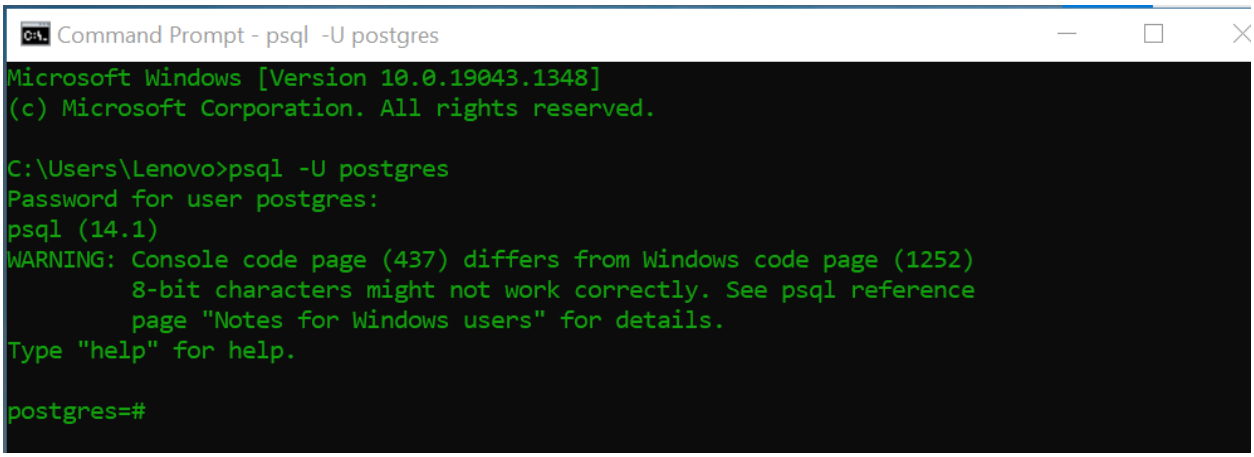
Introduction

Importing a CSV file into psql is an important procedure that allows the user to extract meaningful insights from spreadsheets. Although excel has a Graphical User Interface (GUI) and is more user-friendly, it fails to handle large CSV files and big data. This is where SQL comes in handy with its ability to handle large files.

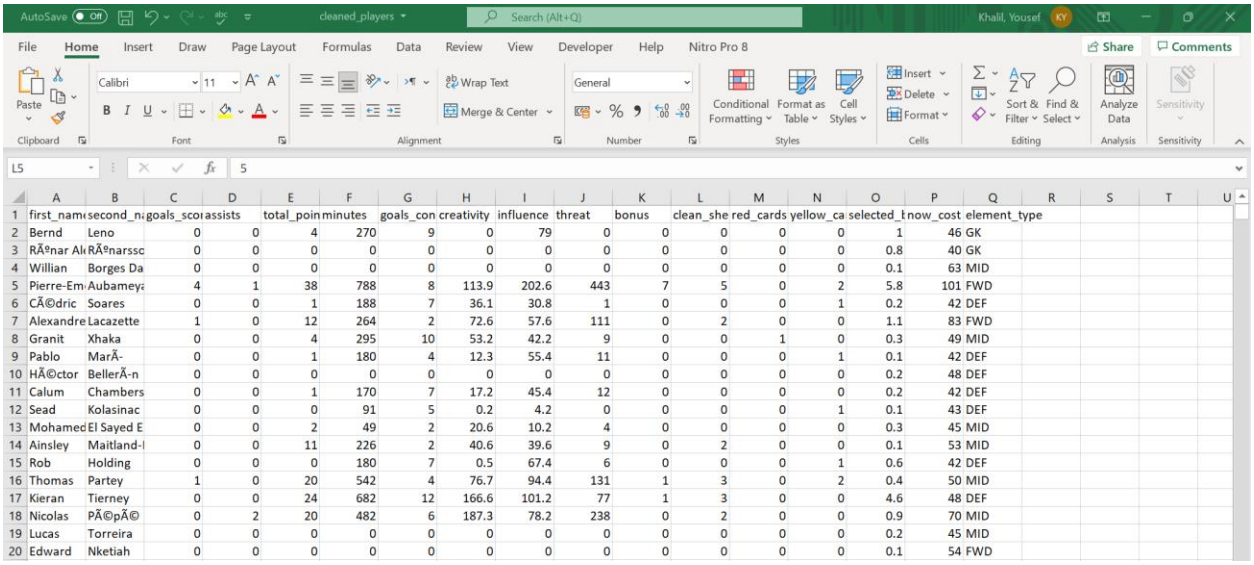
This report will explore how to load a CSV file into psql in command line, to perform queries and subqueries on the data, and to export the results of a query (a table) into a new CSV file. The chosen CSV file is a fantasy football player dataset for the 2021/2022 Barclays Premier League season. It contains player statistics such as goals, assists, creativity, and others. The aim of this report will be to extract meaningful insights from the data using queries and subqueries to advise the user on the best players to bug and bargains.

Question 1



We use **psql -U postgres** and type in the password to log into the local database.

Question 2



first_nam	second_n	goals_sco	assists	total_poin	minutes	goals_con	creativity	influence	threat	bonus	clean_she	red_cards	yellow_ca	selected_t	now_cost	element_type
Bernd	Leno	0	0	4	270	9	0	79	0	0	0	0	0	1	46	GK
RÃªnar Alk	RÃªnarssc	0	0	0	0	0	0	0	0	0	0	0	0	0.8	40	GK
Willian	Borges Da	0	0	0	0	0	0	0	0	0	0	0	0	0.1	63	MID
Pierre-Em	Aubameya	4	1	38	788	8	113.9	202.6	443	7	5	0	2	5.8	101	FWD
CÃ©dric	Soares	0	0	1	188	7	36.1	30.8	1	0	0	0	1	0.2	42	DEF
Alexandre	Lacazette	1	0	12	264	2	72.6	57.6	111	0	2	0	0	1.1	83	FWD
Granit	Xhaka	0	0	4	295	10	53.2	42.2	9	0	0	1	0	0.3	49	MID
Pablo	MarÃ-	0	0	1	180	4	12.3	55.4	11	0	0	0	1	0.1	42	DEF
HÃ©ctor	BellerÃ-n	0	0	0	0	0	0	0	0	0	0	0	0	0.2	48	DEF
Calum	Chambers	0	0	1	170	7	17.2	45.4	12	0	0	0	0	0.2	42	DEF
Sead	Kolasinac	0	0	0	91	5	0.2	4.2	0	0	0	0	1	0.1	43	DEF
Mohamed	El Sayed E	0	0	2	49	2	20.6	10.2	4	0	0	0	0	0.3	45	MID
Ainsley	Maitland-I	0	0	11	226	2	40.6	39.6	9	0	2	0	0	0.1	53	MID
Rob	Holding	0	0	0	180	7	0.5	67.4	6	0	0	0	1	0.6	42	DEF
Thomas	Partey	1	0	20	542	4	76.7	94.4	131	1	3	0	2	0.4	50	MID
Kieran	Tierney	0	0	24	682	12	166.6	101.2	77	1	3	0	0	4.6	48	DEF
Nicolas	PÃ©pÃ©	0	2	20	482	6	187.3	78.2	238	0	2	0	0	0.9	70	MID
Lucas	Torreira	0	0	0	0	0	0	0	0	0	0	0	0	0.2	45	MID
Edward	Nketiah	0	0	0	0	0	0	0	0	0	0	0	0	0.1	54	FWD

Fantasy Premier League Players 21/22

We choose **cleaned_players** (source : github), a database of all players in the 2021/2022 Barclays Premier League season who are available for selection in the Fantasy Premier League game. Fantasy Football is an online game where gamers have a budget to choose a fantasy squad of Barclays Premier League players based on how well they think these players will perform. If the players they pick, perform well the gamers will gain points and move up in the rankings. The csv consists of the following columns:

first_name: first name of player

second_name: family name

goals_scored: goals scored so far

assists: goal assists provided

total_points: number of points gained by player in fantasy across all matches played

minutes: minutes played on the pitch

goals_conceded: number of goals player's team has conceded

creativity: rating of player's ability to create chances

influence: rating of player's ability to individually influence the game and its outcome

threat: rating of player's ability to cause trouble for the opposition in offence

bonus: accumulated bonus points

clean_sheets: number of matches player's team has completed without conceding any goals

red_cards: number of red cards player has received

yellow_cards: number of yellow cards player has received

selected_by_percent: percentage of fantasy gamers who have picked the player in their squad

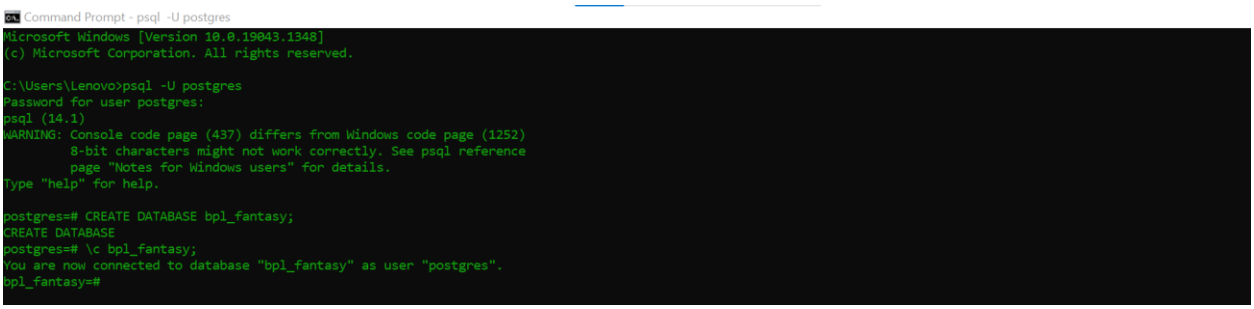
now_cost: current price in fantasy

element_type: position of player on the field

Question 3

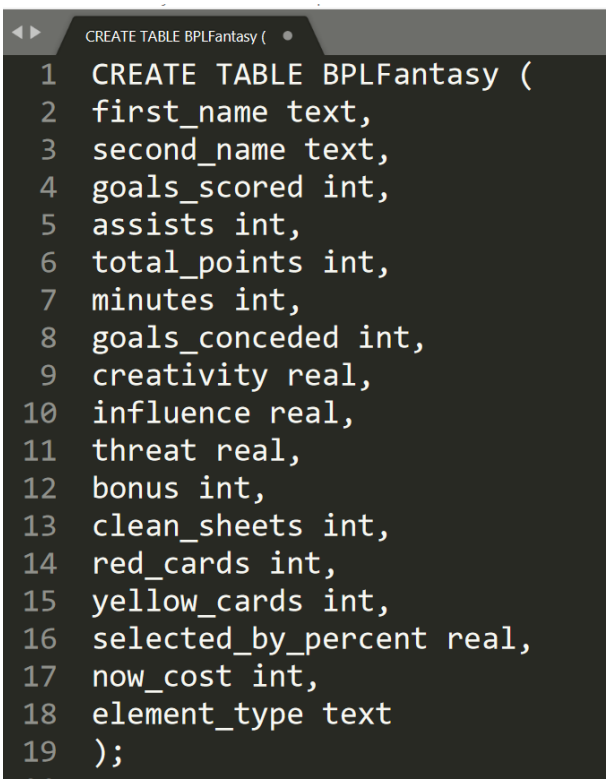
We use the **CREATE DATABASE** command followed by the name of choice for the database. In this case, the chosen name of the database is **bpl_fantasy** for relevance.

Question 4



We use the "**\c**" command followed by **bpl_fantasy** in command line to connect to the database.

Question 5



We use **CREATE TABLE** to create the table **BPLFantasy**

The columns which are of the **integer** datatype are allocated the **int** datatype (an alias for integer). These are take on exact numerical values (no decimals).

The columns which are **floats** are allocated the **real** datatype. Floats may be decimals.

Strings are allocated the **text** datatype.

Question 6

```
bpl_fantasy=#
bpl_fantasy=# CREATE TABLE BPLFantasy (
bpl_fantasy(# first_name text,
bpl_fantasy(# second_name text,
bpl_fantasy(# goals_scored int,
bpl_fantasy(# assists int,
bpl_fantasy(# total_points int,
bpl_fantasy(# minutes int,
bpl_fantasy(# goals_conceded int,
bpl_fantasy(# creativity real,
bpl_fantasy(# influence real,
bpl_fantasy(# threat real,
bpl_fantasy(# bonus int,
bpl_fantasy(# clean_sheets int,
bpl_fantasy(# red_cards int,
bpl_fantasy(# yellow_cards int,
bpl_fantasy(# selected_by_percent real,
bpl_fantasy(# now_cost int,
bpl_fantasy(# element_type text
bpl_fantasy(# );
CREATE TABLE
bpl_fantasy=#
bpl_fantasy=# \dt
           List of relations
 Schema |    Name    | Type  |  Owner
--------+------------+-------+----------
 public | bplfantasy | table | postgres
(1 row)


bpl_fantasy=#
```

We run the **CREATE TABLE** command to create the table and define its columns and their datatypes. Then, we use "**\dt**" to check that the table was created.

Question 7

```
bpl_fantasy=#
bpl_fantasy=# \copy BPLFantasy FROM 'C:\Users\Lenovo\Desktop\New folder\Fantasy-Premier-League\data\2021-22\cleaned_players.csv'DELIMITER','CSV HEADER;
COPY 623
bpl_fantasy=#
```

Only "**\copy**" worked in this case. "**\copy**" should be followed by the name of the table ("**BPLFantasy"**) that we created in the **bpl_fantasy** database. We then include the **FROM** phrase and specify the directory of the csv file after. We finally use **DELIMITER ',' CSV HEADER** to designate that the imported file is a csv file.

Question 8

```
bpl_fantasy=# \copy BPLFantasy FROM 'C:\Users\Lenovo\Desktop\New folder\Fantasy-Premier-League\data\2021-22\cleaned_players.csv' DELIMITER ',' CSV HEADER;
COPY 623
bpl_fantasy=# SELECT * FROM BPLFantasy;
     first_name      |          second_name          | goals_scored | assists | total_points | minutes | goals_conceded | creativity | influence | threat | bonus | clean_sheets | red
_cards | yellow_cards | selected_by_percent | now_cost | element_type
--------------------+-------------------------------+--------------+---------+--------------+---------+----------------+------------+-----------+--------+-------+--------------+----
-------+--------------+---------------------+----------+--------------
 Bernd              | Leno                          |            0 |       0 |            4 |     270 |              9 |          0 |        79 |      0 |     0 |            0 |
     0 |            0 |                   1 |       46 | GK
 R├╢nar Alex        | R├╢narsson                    |            0 |       0 |            0 |       0 |              0 |          0 |         0 |      0 |     0 |            0 |
     0 |            0 |                 0.8 |       40 | GK
 Willian            | Borges Da Silva               |            0 |       0 |            0 |       0 |              0 |          0 |         0 |      0 |     0 |            0 |
     0 |            0 |                 0.1 |       63 | MID Pierre-Emerick      | Aubameyang                    |            4 |       1 |           38 |    788 |             8 |
   113.9 |     202.6 |     443 |       7 |            5 |         0 |         2 |                5.8 |       101 | FWD C├-dric            | Soares                        |
     0 |         0 |               1 |     188 |              7 |      36.1 |      30.8 |       1 |     0 |            0 |         0 |            1 |           0.2 |       42 | DEF Al
exandre          | Lacazette                     |            1 |       0 |           12 |     264 |            2 |       72.6 |      57.6 |    111 |     0 |            2 |
   0 |           0 |                 1.1 |       83 | FWD Granit              | Xhaka                         |            0 |       0 |            4 |     295 |            10 |
  53.2 |      42.2 |       9 |       0 |            0 |         1 |         0 |              0.3 |        49 | MID Pablo              | Mar├╢                         |
 0 |         0 |               1 |     180 |              4 |      12.3 |      55.4 |      11 |     0 |            0 |         0 |            1 |           0.1 |       42 | DEF H├-ct
or           | Beller├╢n                     |            0 |       0 |            0 |       0 |              0 |          0 |         0 |      0 |     0 |            0 |           0
 |            0 |                 0.2 |       48 | DEF Calum               | Chambers                      |            0 |       0 |            1 |     170 |             7 |
17.2 |      45.4 |      12 |       0 |            0 |         0 |         0 |              0.2 |        42 | DEF Sead               | Kolasinac                     |
|         0 | -- More  --
```

We use **SELECT * FROM BPLFantasy** to display the entire BPLFantasy table.

Question 9

Query 1:

The purpose of this query is to return the number of players in each field position and rank each position by its player count (to find the most abundant field positions).

To accomplish this, we select the **element_type** (field position) column and **COUNT(*)** as our aggregate function because the aim is to count the number of players in each element type. We rename the count column as **player_count**. We also use **GROUP BY** to group the records by element_type. Finally, to rank the positions by player count, we use **ORDER BY** and limit the result to 20 rows.

```
31  SELECT element_type, COUNT(*)
32  AS player_count
33  FROM BPLFantasy
34  GROUP BY element_type
35  ORDER BY player_count DESC;
```

Query 1

```
bpl_fantasy=# SELECT element_type, COUNT(*)
bpl_fantasy-# AS player_count
bpl_fantasy-# FROM BPLFantasy
bpl_fantasy-# GROUP BY element_type
bpl_fantasy-# ORDER BY player_count DESC;
 element_type | player_count
--------------+--------------
 MID          |          510
 DEF          |          422
 FWD          |          172
 GK           |          142
(4 rows)
```

Results Query 1

Query 2:

The purpose of this query is to select the top 20 outfield players in the Premier League who have earned the most points so far in the season. If two players have the same total points, their goal record will be compared and the player with the greatest number of goals will be ranked above in the list.

To accomplish this, we use the **SELECT** statement to return the relevant columns. We use **WHERE** to filter the rows to only outfield players. Then, we use **ORDER BY** to rank the rows in descending order by total points and goals scored, and **limit** to 20 records.

```sql
SELECT DISTINCT first_name,
second_name, total_points,
goals_scored
FROM BPLFantasy
WHERE element_type != 'GK'
ORDER BY total_points DESC,
goals_scored DESC
LIMIT 20;
```

Query 2

```
bpl_fantasy=# SELECT DISTINCT first_name, second_name, total_points, goals_scored
bpl_fantasy-# FROM BPLFantasy
bpl_fantasy-# WHERE element_type != 'GK'
bpl_fantasy-# ORDER BY total_points DESC, goals_scored DESC
bpl_fantasy-# LIMIT 20;
     first_name     |   second_name    | total_points | goals_scored
--------------------+------------------+--------------+--------------
 Mohamed            | Salah            |          117 |           10
 Jo├úo Pedro Cavaco | Cancelo          |           67 |            0
 Trent              | Alexander-Arnold |           64 |            1
 Reece              | James            |           63 |            4
 Conor              | Gallagher        |           62 |            4
 Michail            | Antonio          |           61 |            6
 Jamie              | Vardy            |           60 |            7
 Sadio              | Man├⌐            |           59 |            6
 Heung-Min          | Son              |           58 |            4
 Emile              | Smith Rowe       |           57 |            4
 Sa├»d              | Benrahma         |           57 |            3
 Jarrod             | Bowen            |           55 |            2
 Antonio            | R├╝diger         |           53 |            1
 Bruno Miguel       | Borges Fernandes |           52 |            4
 Andros             | Townsend         |           52 |            3
 Pablo              | Fornals          |           51 |            4
 Leandro            | Trossard         |           51 |            3
 Youri              | Tielemans        |           51 |            3
 Raphael            | Dias Belloli     |           50 |            5
 Gabriel Fernando   | de Jesus         |           50 |            2
(20 rows)
```

Results Query 2

Query 3:

The purpose of this query is to return the first name, last name, element type, individual influence score, and average score partitioned by **element_type**, of each player, and rank the players according to their individual influence. This is useful to compare how the individual influence scores of the most influential players compare to the average in their field positions.

To accomplish this, we use the **SELECT** statement to select the relevant columns. At the end of the SELECT statement, we add the **avg(influence)** before the **OVER** window function. **avg(influence)** will be calculated for each **element_type** since we partitioned by **element_type**. Finally, we rank the rows by the individual influence of each player using **ORDER BY**.

```
38  SELECT DISTINCT first_name,
    second_name, element_type,
    influence,
39  avg(influence) OVER(PARTITION
    BY element_type)
40  FROM BPLFantasy
41  ORDER BY influence DESC
42  LIMIT 20;
43
```

Query 3

```
bpl_fantasy=# SELECT DISTINCT first_name, second_name, element_type, influence,
bpl_fantasy-# avg(influence) OVER(PARTITION BY element_type)
bpl_fantasy-# FROM BPLFantasy
bpl_fantasy-# ORDER BY influence DESC
bpl_fantasy-# LIMIT 20;
  first_name  |   second_name    | element_type | influence |        avg
--------------+------------------+--------------+-----------+-------------------
 Mohamed      | Salah            | MID          |     563.2 | 74.87843140620811
 Youri        | Tielemans        | MID          |     335.8 | 74.87843140620811
 Bruno Miguel | Borges Fernandes | MID          |     323.6 | 74.87843140620811
 Tim          | Krul             | GK           |     318.4 | 69.55211278082619
 Michail      | Antonio          | FWD          |       312 | 66.14651135078003
 Declan       | Rice             | MID          |     308.6 | 74.87843140620811
 Illan        | Meslier          | GK           |     307.8 | 69.55211278082619
 Michael      | Keane            | DEF          |     306.4 | 84.91658742768222
 Conor        | Gallagher        | MID          |     300.4 | 74.87843140620811
 Kasper       | Schmeichel       | GK           |     298.4 | 69.55211278082619
 Trent        | Alexander-Arnold | DEF          |     294.4 | 84.91658742768222
 Andros       | Townsend         | MID          |     289.2 | 74.87843140620811
 Raphael      | Dias Belloli     | MID          |     281.8 | 74.87843140620811
 Sadio        | Man├⌐            | MID          |     280.8 | 74.87843140620811
 Jos├⌐        | Malheiro de S├¡  | GK           |       279 | 69.55211278082619
 Jamie        | Vardy            | FWD          |       277 | 66.14651135078003
 James        | Tarkowski        | DEF          |     276.6 | 84.91658742768222
 Emiliano     | Mart├¡nez        | GK           |       268 | 69.55211278082619
 Jarrod       | Bowen            | MID          |       268 | 74.87843140620811
 Pablo        | Fornals          | MID          |     267.2 | 74.87843140620811
(20 rows)
```

Results Query 3

Query 4:

The aim of this query is to find the 20 most lethal forwards in the league (highest threat) and compare their threat scores to the average threat score of all forwards in the league.

To accomplish this, we first creat the CTE **avg_threats** which groups all the records by element_type (position) and computes the average threat of each position. Using this CTE, we creat another CTE **fwd_avg_threats** which selects only the row belonging to the element_type **FWD.** Then for the main query, we select the first name, second name, individual threat level, and average threat level for forwards from the CTE **fwd_avg_threats.** We filter the rows to display only forwards since we only want to compare threat levels between forwards and not other positions. We also filter them to display only records with threat scores above the average for forwards (using the fwd_avg_threat CTE). We finally order the rows by descending threat level and limited the rows to 20. This result could not have been accomplished without a subquery.

```sql
WITH avg_threats AS
(
SELECT element_type, AVG(threat)
AS avg_threat
FROM BPLFantasy
GROUP BY element_type
), fwd_avg_threat AS
(
SELECT avg_threat
FROM avg_threats
WHERE element_type = 'FWD')


SELECT DISTINCT first_name, second_name, threat, (SELECT *
FROM fwd_avg_threat)
FROM BPLFantasy
WHERE element_type = 'FWD'
AND threat > (SELECT *
FROM fwd_avg_threat)
ORDER BY threat DESC
LIMIT 20;
```

Query 4

```
bpl_fantasy=# WITH avg_threats AS
bpl_fantasy-# (
bpl_fantasy(# SELECT element_type, AVG(threat)
bpl_fantasy(# AS avg_threat
bpl_fantasy(# FROM BPLFantasy
bpl_fantasy(# GROUP BY element_type
bpl_fantasy(# ), fwd_avg_threat AS
bpl_fantasy-# (
bpl_fantasy(# SELECT avg_threat
bpl_fantasy(# FROM avg_threats
bpl_fantasy(# WHERE element_type = 'FWD')
bpl_fantasy-#
bpl_fantasy-#
bpl_fantasy-# SELECT DISTINCT first_name, second_name, threat, (SELECT *
bpl_fantasy(# FROM fwd_avg_threat)
bpl_fantasy-# FROM BPLFantasy
bpl_fantasy-# WHERE element_type = 'FWD'
bpl_fantasy-# AND threat > (SELECT *
bpl_fantasy(# FROM fwd_avg_threat)
bpl_fantasy-# ORDER BY threat DESC
bpl_fantasy-# LIMIT 20;
   first_name     |    second_name    | threat |    avg_threat
------------------+-------------------+--------+------------------
 Michail          | Antonio           |    590 | 120.3953488372093
 Pierre-Emerick   | Aubameyang        |    443 | 120.3953488372093
 Cristiano Ronaldo | dos Santos Aveiro |    410 | 120.3953488372093
 Jamie            | Vardy             |    387 | 120.3953488372093
 Gabriel Fernando | de Jesus          |    374 | 120.3953488372093
 Ollie            | Watkins           |    373 | 120.3953488372093
 Ra╠╣l            | Jim├⌐nez          |    356 | 120.3953488372093
 Ivan             | Toney             |    342 | 120.3953488372093
 Emmanuel         | Dennis            |    325 | 120.3953488372093
 Chris            | Wood              |    320 | 120.3953488372093
 Adam             | Armstrong         |    311 | 120.3953488372093
 Romelu           | Lukaku            |    294 | 120.3953488372093
 Harry            | Kane              |    279 | 120.3953488372093
 Teemu            | Pukki             |    269 | 120.3953488372093
 Joshua           | King              |    266 | 120.3953488372093
 Allan            | Saint-Maximin     |    262 | 120.3953488372093
 Christian        | Benteke           |    240 | 120.3953488372093
 Patrick          | Bamford           |    240 | 120.3953488372093
 Rodrigo          | Moreno            |    229 | 120.3953488372093
 Danny            | Ings              |    223 | 120.3953488372093
(20 rows)
```

Results Query 4

Query 5:

This query is very useful for any gamer who is looking to add a great midfielder to their squad for a bargain. It returns all midfielders who cost lower than the average cost of a midfielder, but who have at the same time produced more points than the average for a midfielder. It also ranks these players by decreasing total points and decreasing creativity (an important indicator for a midfielder). This is very good example of how a gamer can leverage this data to gain an advantage in fantasy football.

To accomplish this, (just like Query 4) we first create a CTE for the average of the total points (**average_points**) grouped by **element_type** . Then using this CTE, we create a new CTE **mid_avg_points** to retrieve the average total points for a midfielder. Then, we do the same procedure to find the average price, and the average price for a midfielder. Finally, in the main query, we select the first name, last name, creativity score, price, and total points columns. We filter the rows with the **WHERE** statement to retrieve only midfielders, who cost less than the average (using **mid_avg_price CTE**), and who produced more points than the average (using **mid_avg_points CTE**). Finally, we use **ORDER BY** to order in descending order by the number of points then the creativity score.

```
72  WITH average_points AS
73  (
74  SELECT element_type, AVG(total_points)
75  AS avg_points
76  FROM BPLFantasy
77  GROUP BY element_type
78  HAVING element_type = 'MID'
79  ), mid_avg_points AS
80
81  (
82  SELECT avg_points
83  FROM average_points
84  WHERE element_type = 'MID'),
85  average_price AS
86  (
87
88  SELECT element_type, AVG(now_cost)
89  AS avg_price
90  FROM BPLFantasy
91  GROUP BY element_type
92  HAVING element_type = 'MID'
93  ), mid_avg_price AS
94
95  (
96  SELECT avg_price
97  FROM average_price
98  WHERE element_type = 'MID')
99
100
101 SELECT DISTINCT first_name, second_name, creativity, now_cost, total_points
102 FROM BPLFantasy
103 WHERE element_type = 'MID'
104 AND now_cost < (SELECT * FROM mid_avg_price)
105 AND total_points > (SELECT * FROM mid_avg_points)
106 ORDER BY total_points DESC, creativity DESC
107 LIMIT 20;
```

```
bpl_fantasy=# WITH average_points AS
bpl_fantasy-# (
bpl_fantasy(# SELECT element_type, AVG(total_points)
bpl_fantasy(# AS avg_points
bpl_fantasy(# FROM BPLFantasy
bpl_fantasy(# GROUP BY element_type
bpl_fantasy(# HAVING element_type = 'MID'
bpl_fantasy(# ), mid_avg_points AS
bpl_fantasy-#
bpl_fantasy-# (
bpl_fantasy(# SELECT avg_points
bpl_fantasy(# FROM average_points
bpl_fantasy(# WHERE element_type = 'MID'),
bpl_fantasy-# average_price AS
bpl_fantasy-# (
bpl_fantasy(#
bpl_fantasy(# SELECT element_type, AVG(now_cost)
bpl_fantasy(# AS avg_price
bpl_fantasy(# FROM BPLFantasy
bpl_fantasy(# GROUP BY element_type
bpl_fantasy(# HAVING element_type = 'MID'
bpl_fantasy(# ), mid_avg_price AS
bpl_fantasy-#
bpl_fantasy-# (
bpl_fantasy(# SELECT avg_price
bpl_fantasy(# FROM average_price
bpl_fantasy(# WHERE element_type = 'MID')
bpl_fantasy-#
bpl_fantasy-#
bpl_fantasy-# SELECT DISTINCT first_name, second_name, creativity, now_cost, total_points
bpl_fantasy-# FROM BPLFantasy
bpl_fantasy-# WHERE element_type = 'MID'
bpl_fantasy-# AND now_cost < (SELECT * FROM mid_avg_price)
bpl_fantasy-# AND total_points > (SELECT * FROM mid_avg_points)
bpl_fantasy-# ORDER BY total_points DESC, creativity DESC
bpl_fantasy-# LIMIT 20;
    first_name     |   second_name    | creativity | now_cost | total_points
-------------------+------------------+------------+----------+--------------
 Mateo             | Kovacic          |      189.1 |       52 |           46
 Abdoulaye         | Doucour├─        |      121.1 |       54 |           45
 Declan            | Rice             |      169.1 |       50 |           41
 Jordan            | Henderson        |      242.2 |       50 |           38
```

Results Query 5 P1

```
 Pierre-Emile      | H├╕jbjerg        |      110.4 |       49 |           36
 James             | McArthur         |      156.9 |       45 |           33
 Allan             | Marques Loureiro |      127.3 |       46 |           32
 Christian         | N├╕rgaard        |       79.2 |       50 |           31
 Naby              | Keita            |       72.5 |       50 |           30
 Ashley            | Westwood         |      229.5 |       53 |           28
 Stuart            | Dallas           |      184.7 |       51 |           27
 Alexis            | Mac Allister     |       58.3 |       54 |           27
 R├╟ben Diogo      | da Silva Neves   |      196.7 |       54 |           25
 Douglas Luiz      | Soares de Paulo  |      176.1 |       46 |           25
 Mateusz           | Klich            |      164.2 |       54 |           25
 Vitaly            | Janelt           |      115.2 |       50 |           25
 Jo├║o Filipe Iria | Santos Moutinho  |      238.8 |       50 |           24
 Mathias           | Normann          |       93.5 |       45 |           24
 Isaac             | Hayden           |       32.1 |       45 |           24
 Adam              | Lallana          |       91.8 |       54 |           23
(20 rows)
```

Results Query 5 P2

Question 10

This query returns a table of the top 20 goalkeepers with most clean sheets and goals conceded (descending order), who have played more minutes than the average for the league.

To accomplish this, we first use the **SELECT** statement to filter the columns we want which are first name, last name, goals conceded, and clean sheets. We use the **WHERE** statement to filter the data to only goalkeepers who have played more minutes than the average (used a subquery). Then, we use **ORDER BY** to order the rows by clean sheets and goals conceded and limited the table to 20 rows.

To create the table that stores the result of the above query, we use **CREATE TABLE** followed by the name of the table I chose (**top_gks).** Then, we copy the query into parentheses after **AS.** This means that the result of the query is stored in the new table.

```
112
113    SELECT DISTINCT first_name, second_name,
       goals_conceded, clean_sheets
114    FROM BPLFantasy
115    WHERE element_type = 'GK' AND minutes > (SELECT AVG(
       minutes) FROM BPLFantasy)
116    ORDER BY clean_sheets DESC, goals_conceded DESC
117    LIMIT 20;
118
```

```
bpl_fantasy=# SELECT DISTINCT first_name, second_name, goals_conceded, clean_sheets
bpl_fantasy-# FROM BPLFantasy
bpl_fantasy-# WHERE element_type = 'GK' AND minutes > (SELECT AVG(minutes) FROM BPLFantasy)
bpl_fantasy-# ORDER BY clean_sheets DESC, goals_conceded ASC
bpl_fantasy-# LIMIT 20;
 first_name |    second_name    | goals_conceded | clean_sheets
------------+-------------------+----------------+--------------
 Edouard    | Mendy             |              4 |            6
 Ederson    | Santana de Moraes |              6 |            6
 Aaron      | Ramsdale          |              4 |            5
 Alisson    | Ramses Becker     |             11 |            5
 Alex       | McCarthy          |             12 |            5
 Robert     | S├ínchez          |             12 |            4
 Vicente    | Guaita            |             14 |            4
 Hugo       | Lloris            |             16 |            4
 David      | Raya Martin       |              9 |            3
 Jordan     | Pickford          |             13 |            3
 Lukasz     | Fabianski         |             13 |            3
 Emiliano   | Mart├¡nez         |             17 |            3
 Jos├⌐      | Malheiro de S├í   |             12 |            2
 David      | de Gea            |             17 |            2
 Tim        | Krul              |             26 |            2
 Nick       | Pope              |             17 |            1
 Illan      | Meslier           |             18 |            1
 Kasper     | Schmeichel        |             18 |            1
 Daniel     | Bachmann          |              7 |            0
 Ben        | Foster            |             12 |            0
(20 rows)
```

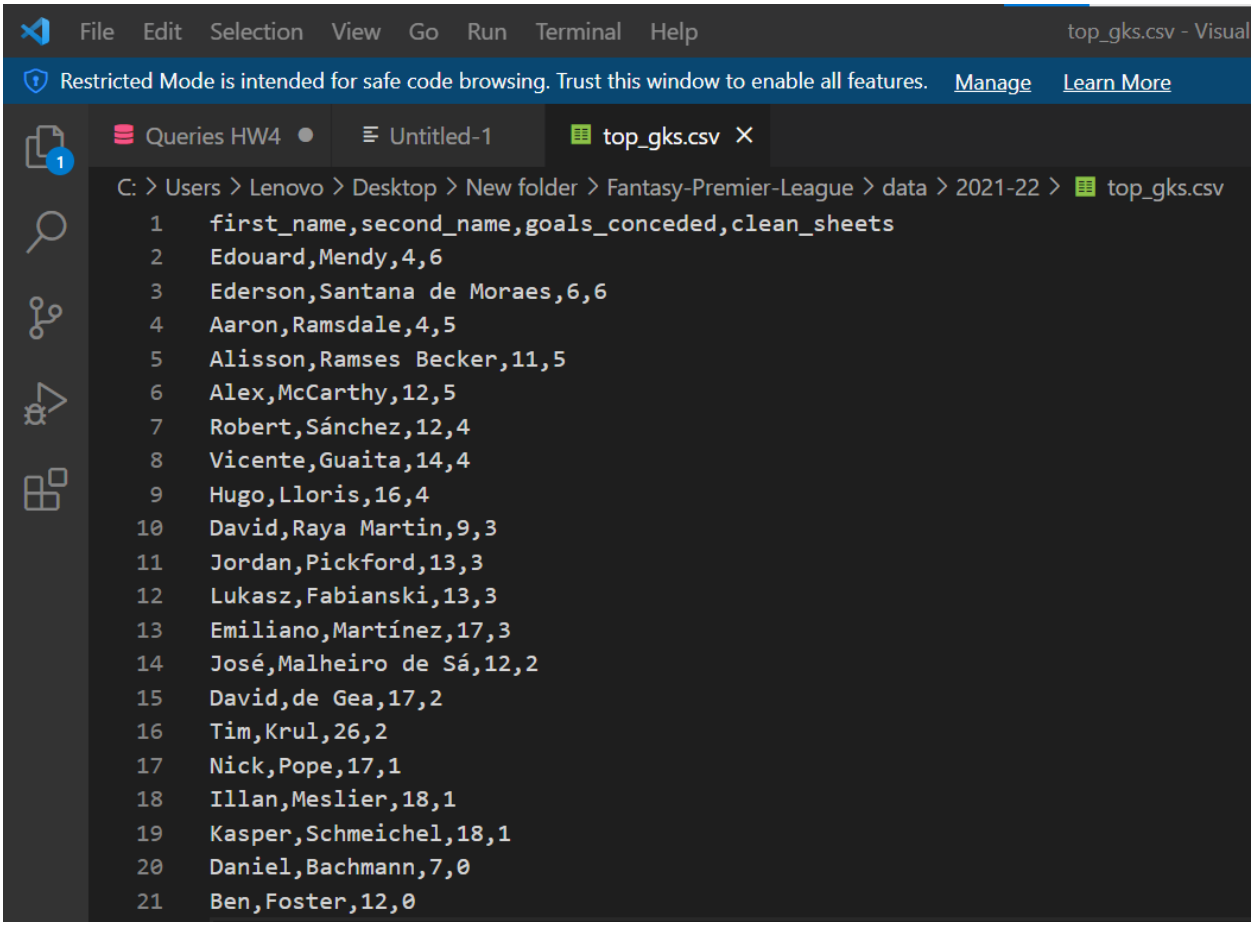
```
bpl_fantasy=# CREATE TABLE top_gks
bpl_fantasy-# AS (SELECT DISTINCT first_name, second_name, goals_conceded, clean_sheets
bpl_fantasy(# FROM BPLFantasy
bpl_fantasy(# WHERE element_type = 'GK' AND minutes > (SELECT AVG(minutes) FROM BPLFantasy)
bpl_fantasy(# ORDER BY clean_sheets DESC, goals_conceded ASC
bpl_fantasy(# LIMIT 20);
SELECT 20
bpl_fantasy=# SELECT * FROM top_gks;
 first_name |    second_name    | goals_conceded | clean_sheets
------------+-------------------+----------------+--------------
 Edouard    | Mendy             |              4 |            6
 Ederson    | Santana de Moraes |              6 |            6
 Aaron      | Ramsdale          |              4 |            5
 Alisson    | Ramses Becker     |             11 |            5
 Alex       | McCarthy          |             12 |            5
 Robert     | S├ínchez          |             12 |            4
 Vicente    | Guaita            |             14 |            4
 Hugo       | Lloris            |             16 |            4
 David      | Raya Martin       |              9 |            3
 Jordan     | Pickford          |             13 |            3
 Lukasz     | Fabianski         |             13 |            3
 Emiliano   | Mart├¡nez         |             17 |            3
 Jos├⌐      | Malheiro de S├í   |             12 |            2
 David      | de Gea            |             17 |            2
 Tim        | Krul              |             26 |            2
 Nick       | Pope              |             17 |            1
 Illan      | Meslier           |             18 |            1
 Kasper     | Schmeichel        |             18 |            1
 Daniel     | Bachmann          |              7 |            0
 Ben        | Foster            |             12 |            0
(20 rows)
```

Question 11

We use **COPY** to copy the table **top_gks** from Question 10 to the new csv file **top_gks.csv**. Then, we specify the **delimiter** as a **comma** and indicated that the first row is a header row.

We then open the CSV in Visual Studio Code.

```
bpl_fantasy=#
bpl_fantasy=# \COPY top_gks TO 'C:\Users\Lenovo\Desktop\New folder\Fantasy-Premier-League\data\2021-22\top_gks.csv' DELIMITER ',' CSV HEADER;
COPY 20
bpl_fantasy=#
```



Conclusion

To sum up this report, we first created a new local database **bpl_fantasy** using the CREATE DATABASE function. To later copy the CSV into the database as a relation, we created a new empty table **BPLFantasy** which contains the columns that have identical names and datatypes to the columns in the CSV file. Then, we used **\copy** to copy the CSV into our new table.

To understand the data better, we wrote five queries of varying complexity. The first query showed us how many players were in the league from each field position. The second query showed us the names and stats of the players who have generated the most fantasy points and scored the most goals. The third query identified the most influential outfield players and compared their influence to the average influence of all players in their position. The fourth query identified the most lethal forwards using CTEs and compared their threat scores to the average threat score of all forwards. The fifth and final query is the most useful one since it identified the midfielders who produce the most fantasy points and are most creative, while having an above average creativity in their respective positions. Finally, we queried a table of top

performing goalkeepers (least goals conceded and most clean sheets) and who have played more minutes than the average. We then exported the results of this table into a CSV file **top_gks** and opened the file in Visual Studio Code.

This assignment was a great learning experience for me since I got to practice queries and subqueries. I feel much more confident now about planning and writing queries