CS 486/586 Introduction to Databases Project - Part II

Name: Soham Nandi

Datasets Finalizing:

- I've finalized total 7 datasets from the database I got from the Kaggle website. The raw datasets contain the data of a football tournament.

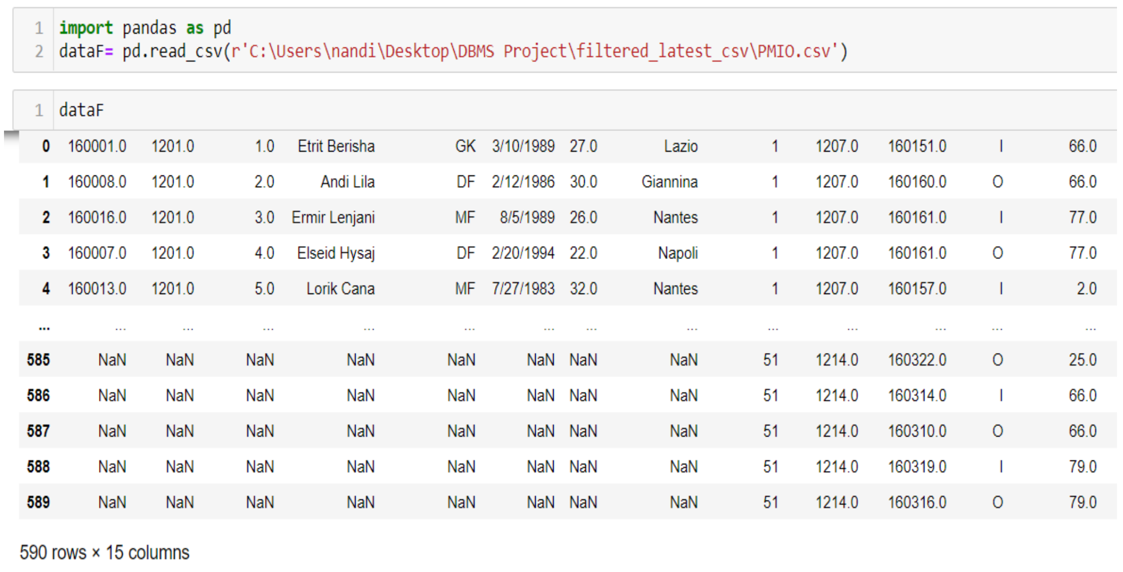
- I have thought about cleaning and normalizing the data using Python's pandas tools, which is a high-level data manipulation tool, before constructing the database and populating the relations, so that it would be simple to create and operate on the football league database effectively.

Describing Normalization of Raw Dataset:

In the raw dataset I found out that there are a lot of rows where there were null values along with redundant data. I separated them into two different tables in order to make the database simple and more readable.

I have implemented pandas library in python to achieve this normalization goal. The workflow is described below.

<u>Step-1:</u> I've imported the csv file as dataframe named 'dataF' , read the csv and printed the dataframe.

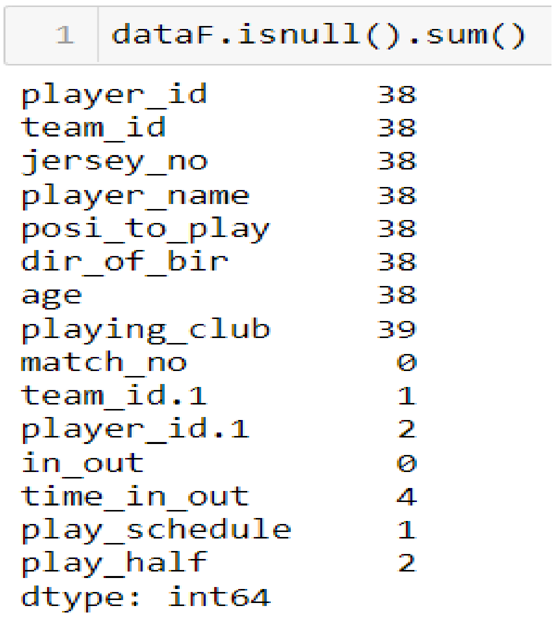


```python
import pandas as pd
dataF= pd.read_csv(r'C:\Users\nandi\Desktop\DBMS Project\filtered_latest_csv\PMIO.csv')
```

```python
dataF
```

```
  0  160001.0  1201.0   1.0  Etrit Berisha   GK  3/10/1989  27.0      Lazio   1  1207.0  160151.0  I  66.0
  1  160008.0  1201.0   2.0      Andi Lila   DF  2/12/1986  30.0   Giannina   1  1207.0  160160.0  O  66.0
  2  160016.0  1201.0   3.0  Ermir Lenjani   MF   8/5/1989  26.0     Nantes   1  1207.0  160161.0  I  77.0
  3  160007.0  1201.0   4.0   Elseid Hysaj   DF  2/20/1994  22.0     Napoli   1  1207.0  160161.0  O  77.0
  4  160013.0  1201.0   5.0     Lorik Cana   MF  7/27/1983  32.0     Nantes   1  1207.0  160157.0  I   2.0
...       ...     ...   ...            ...  ...        ...   ...        ...  ...    ...       ... ...   ...
585       NaN     NaN   NaN            NaN  NaN        NaN   NaN        NaN  51  1214.0  160322.0  O  25.0
586       NaN     NaN   NaN            NaN  NaN        NaN   NaN        NaN  51  1214.0  160314.0  I  66.0
587       NaN     NaN   NaN            NaN  NaN        NaN   NaN        NaN  51  1214.0  160310.0  O  66.0
588       NaN     NaN   NaN            NaN  NaN        NaN   NaN        NaN  51  1214.0  160319.0  I  79.0
589       NaN     NaN   NaN            NaN  NaN        NaN   NaN        NaN  51  1214.0  160316.0  O  79.0

590 rows × 15 columns
```

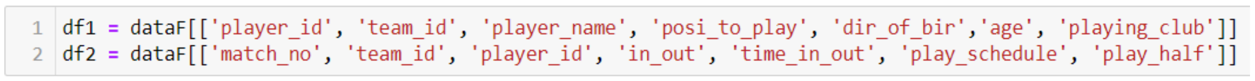
Step-2: I’ve used the command ‘dataF.isnull().sum()’ to get the count of NULL values present in respective columns of that particular csv file.



```
dataF.isnull().sum()
```

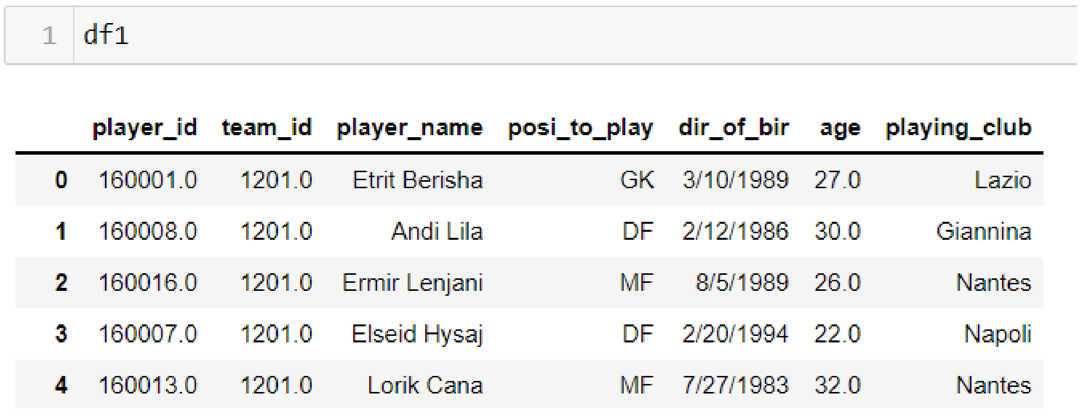
```
player_id        38
team_id          38
jersey_no        38
player_name      38
posi_to_play     38
dir_of_bir       38
age              38
playing_club     39
match_no          0
team_id.1         1
player_id.1       2
in_out            0
time_in_out       4
play_schedule     1
play_half         2
dtype: int64
```

Step-3: To make the dataset more readable, I’ve split the csv in two csv files as respective dataframe ‘df1’ and ‘df2’.



```
df1 = dataF[['player_id', 'team_id', 'player_name', 'posi_to_play', 'dir_of_bir','age', 'playing_club']]
df2 = dataF[['match_no', 'team_id', 'player_id', 'in_out', 'time_in_out', 'play_schedule', 'play_half']]
```

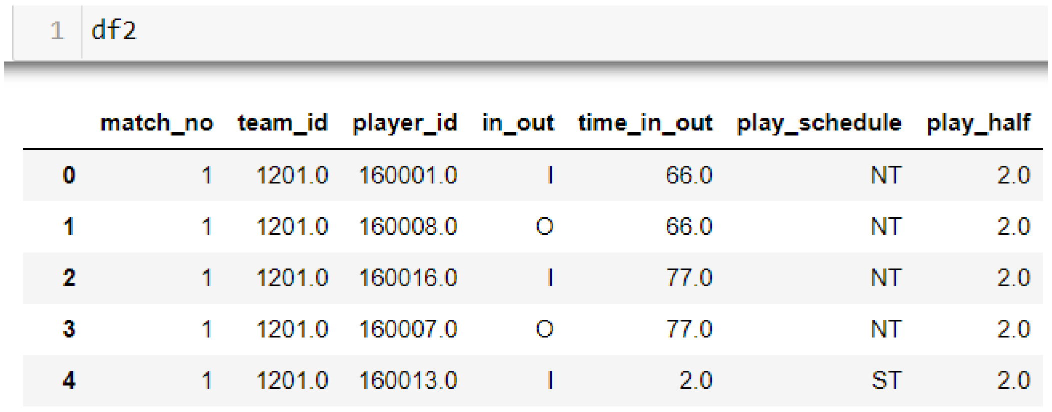
Step-4: After separating I’ve shown the two different datasets I got from raw file.
df1:



```
df1
```

	player_id	team_id	player_name	posi_to_play	dir_of_bir	age	playing_club
0	160001.0	1201.0	Etrit Berisha	GK	3/10/1989	27.0	Lazio
1	160008.0	1201.0	Andi Lila	DF	2/12/1986	30.0	Giannina
2	160016.0	1201.0	Ermir Lenjani	MF	8/5/1989	26.0	Nantes
3	160007.0	1201.0	Elseid Hysaj	DF	2/20/1994	22.0	Napoli
4	160013.0	1201.0	Lorik Cana	MF	7/27/1983	32.0	Nantes

df2:

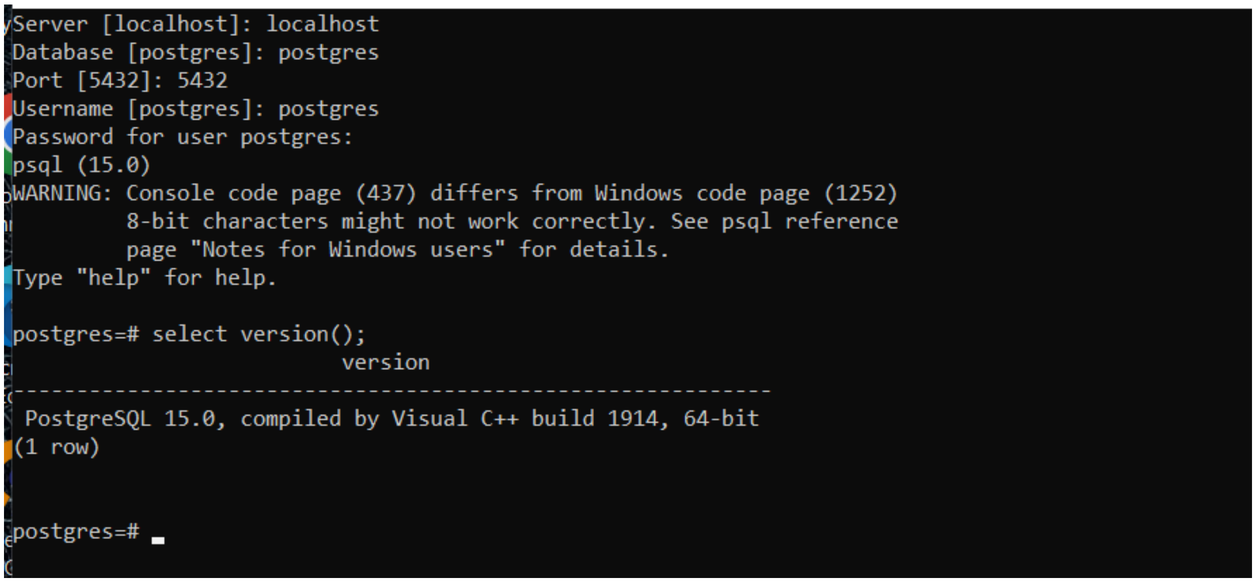


```
df2
```

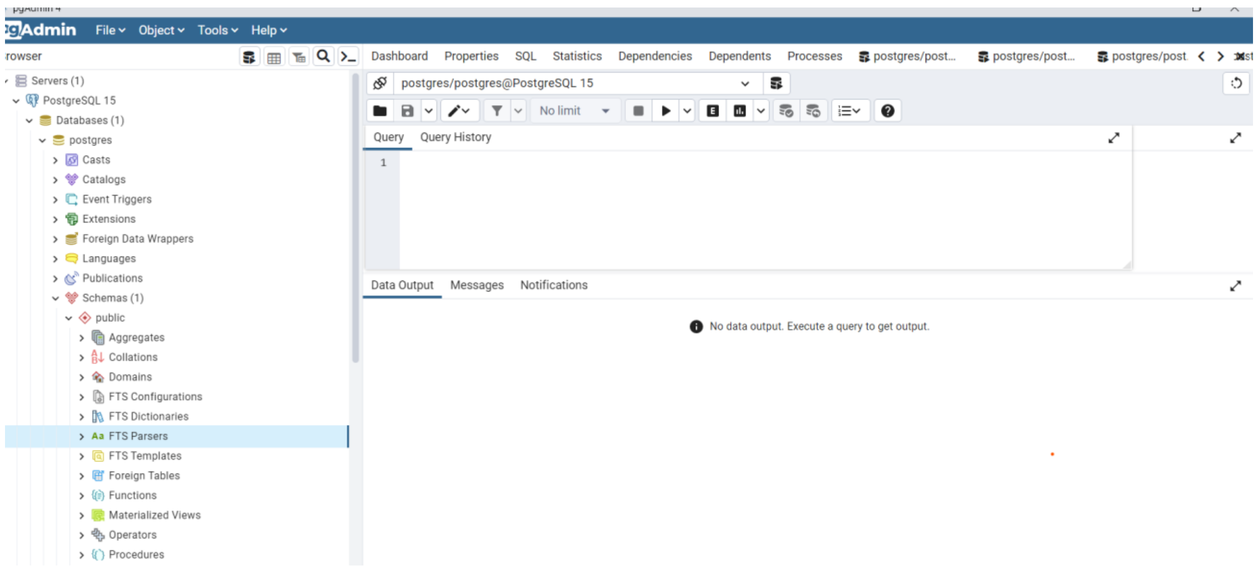
	match_no	team_id	player_id	in_out	time_in_out	play_schedule	play_half
0	1	1201.0	160001.0	I	66.0	NT	2.0
1	1	1201.0	160008.0	O	66.0	NT	2.0
2	1	1201.0	160016.0	I	77.0	NT	2.0
3	1	1201.0	160007.0	O	77.0	NT	2.0
4	1	1201.0	160013.0	I	2.0	ST	2.0

1. **Show that you have successfully installed Postgres and you can use pgAdmin or psql on your local machine or virtual machine (you can provide a screenshot).**

ANS: I have successfully installed Postgres and I can use pgAdmin or psql on my local machine.

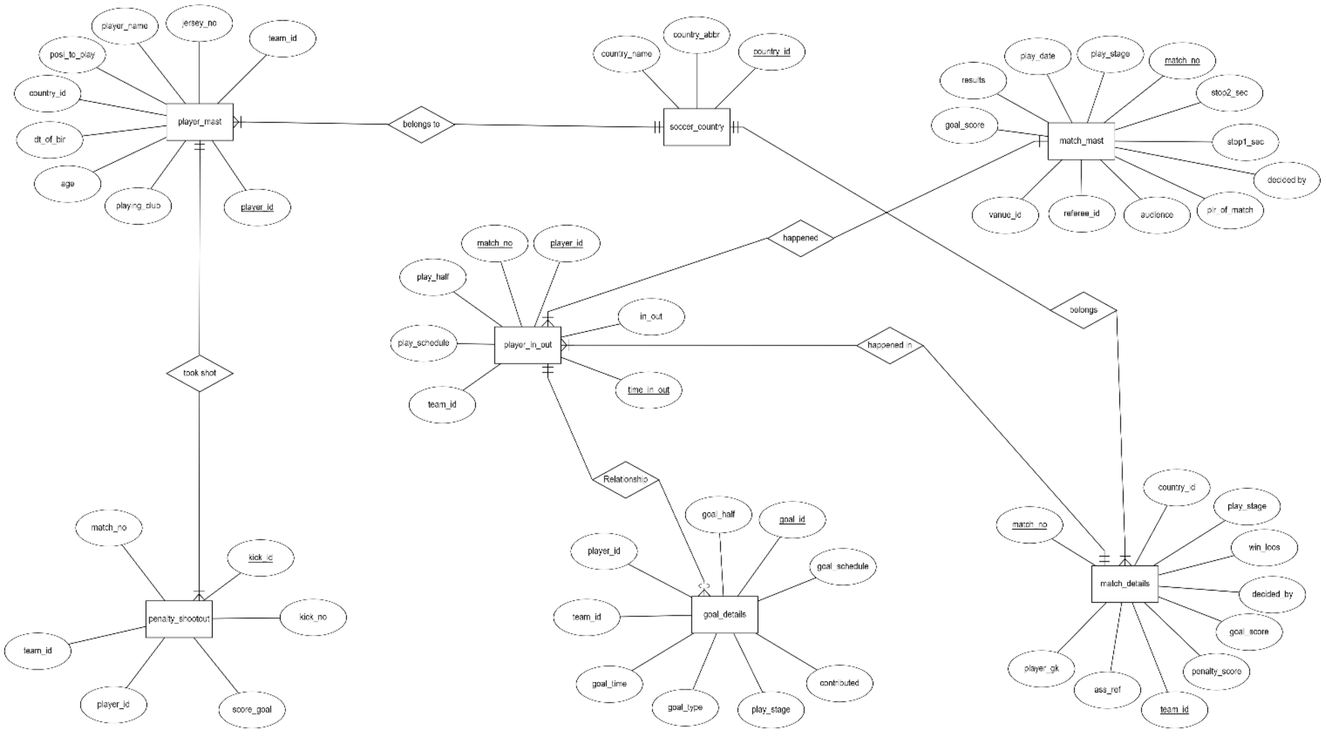


```
Server [localhost]: localhost
Database [postgres]: postgres
Port [5432]: 5432
Username [postgres]: postgres
Password for user postgres:
psql (15.0)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# select version();
                          version
------------------------------------------------------------
 PostgreSQL 15.0, compiled by Visual C++ build 1914, 64-bit
(1 row)


postgres=# _
```



No data output. Execute a query to get output.

2. ER Diagram:



3. Database schema designed with your designed tables, attribute declaration, primary/foreign keys, views, or temporary tables etc. (You should demonstrate a variety of schema and SQL features such as a variety of data types, keys, foreign keys, different cardinalities).

ANS:

- soccer_country(country_id VARCHAR, country_abbr VARCAHAR, country_name VARCHAR)
  country_id is primary key

- player_mast (player_id VARCHAR, team_id VARCHAR, jersey_no INT, player_name VARCHAR, posi_to_play CHAR, dt_of_bir DATE, age INT, playing_club VARCHAR, country_id VARCHAR)
  player_id is primary key
  country_id is foreign key referencing soccer_country(country_id)

- match_mast(match_no VARCHAR, play_stage VARCHAR, play_date DATE, results CHAR(5), decided_by VARCHAR, goal_score VARCHAR, vanue_id VARCHAR, referee_id VARCHAR, audience INT, plr_of_match INT, stop1_sec INT, stop2_sec INT)
  match_no is primary key

- goal_details(goal_id VARCHAR, match_no INT, player_id VARCHAR, team_id VARCHAR, goal_time INT, goal_type CHAR(10), play_stage CHAR(10), goal_schedule CHAR(10), goal_half INT)
  goal_id is primary key
  player_id is foreign key referencing player_in_out(player_id)

- match_details(match_no VARCHAR, team_id VARCHAR, play_stage VARCHAR, win_lose VARCHAR, decided_by VARCHAR, goal_score INT, penalty_score INT, ass_ref INT, player_gk INT)
  match_no, team_id working together as primary key
  country_id is foreign key referencing soccer_country(country_id)

- penalty_shootout(kick_id VARCHAR, match_no INT, team_id VARCHAR, player_id VARCHAR, score_goal VARCHAR, kick_no INT)
  kick_id is primary key
  player_id is foreign key referencing penalty_shootout(player_id)

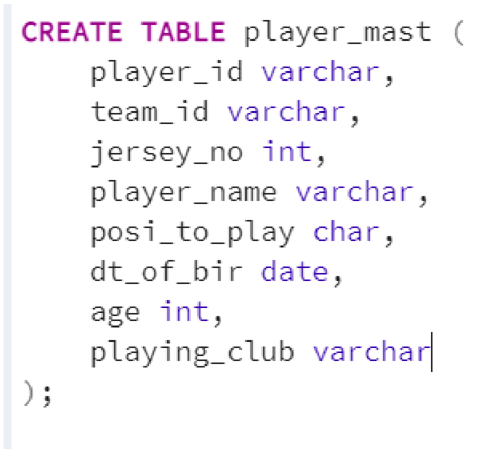
- player_in_out(match_no VARCHAR, player_id VARCHAR, time_in_out INT, in_out CHAR, play_schedule CHAR, team_id VARCHAR, play_half INT)

    match_no, player_id, time_in_out will work together as primary key.
    match_no is foreign key referencing player_in_out(match_no)
    team_id is foreign key referencing match_details(team_id)

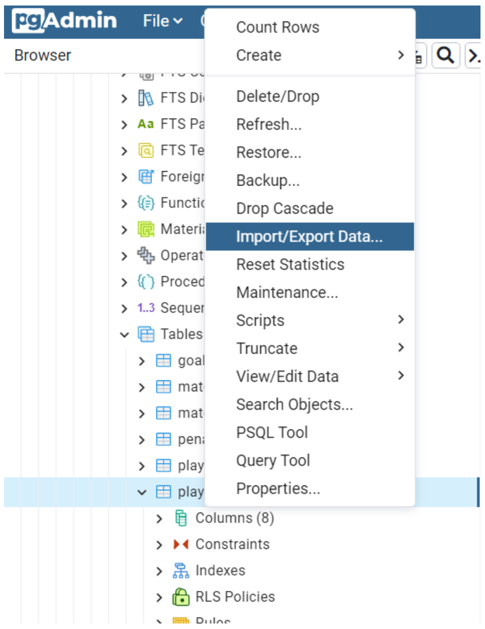
4. **Data loading process and data preprocessing and cleaning. You should have effective data entry and avoid large amounts of manual data entry. Please describe what you have done to clean your data fully, what you have considered in cleaning, and how you have chosen to load and import your data in your tables.**

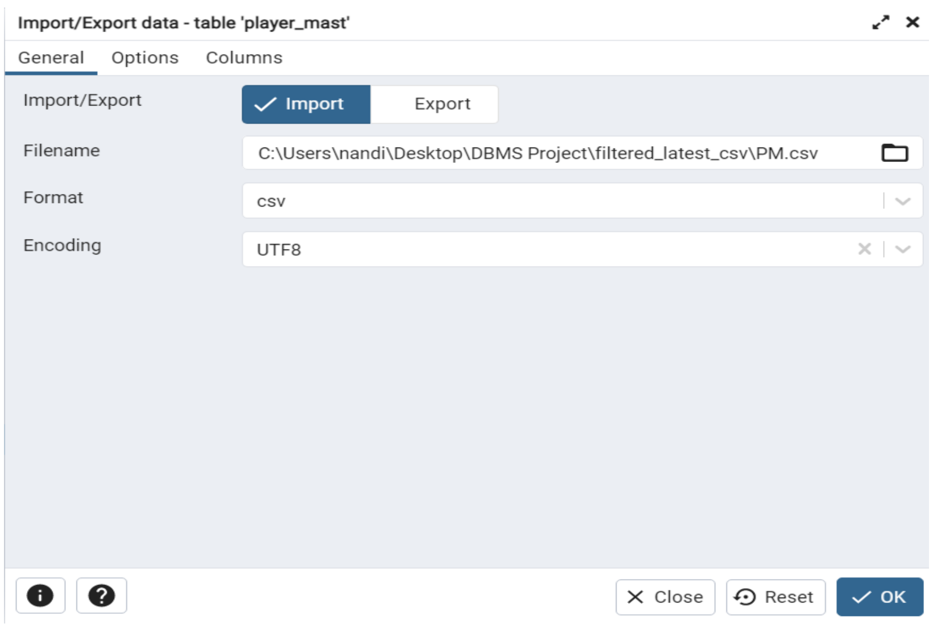
ANS: ***<u>Data Loading</u>***

I've first created the tables and then imported the data from csv file to those respective tables.



```
CREATE TABLE player_mast (
    player_id varchar,
    team_id varchar,
    jersey_no int,
    player_name varchar,
    posi_to_play char,
    dt_of_bir date,
    age int,
    playing_club varchar
);
```




Import/Export data - table 'player_mast'

General | Options | Columns

Import/Export: Import

Filename: C:\Users\nandi\Desktop\DBMS Project\filtered_latest_csv\PM.csv

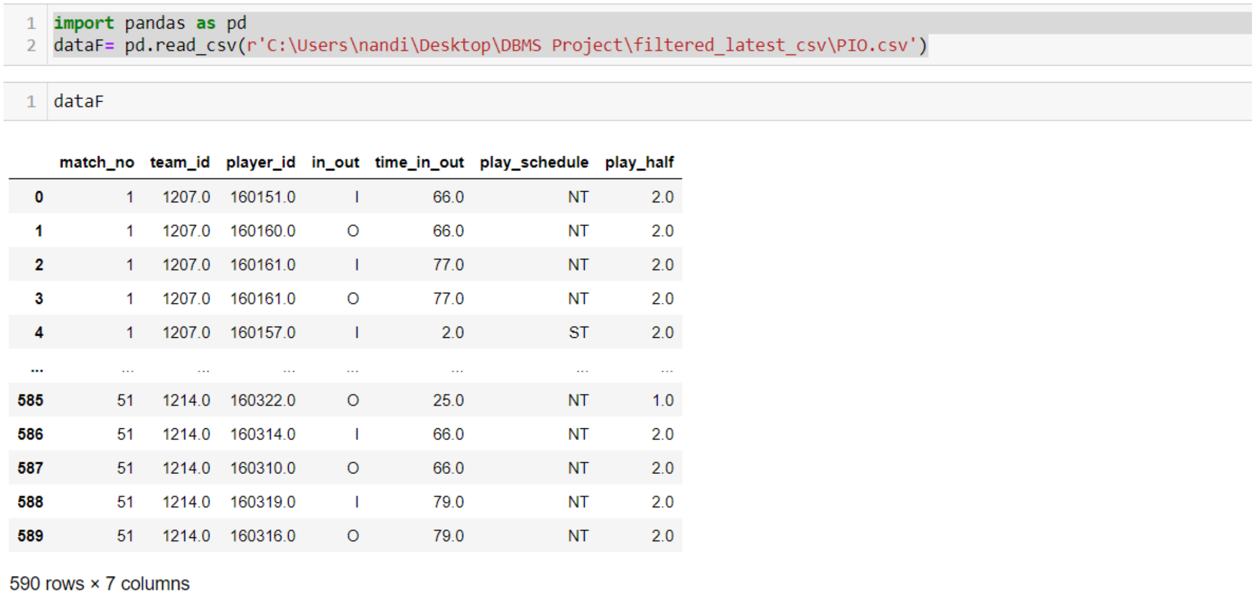
Format: csv

Encoding: UTF8

Data Preprocessing and Data Cleaning:

Method-1: First I've imported the csv file to Jupiter notebook and used pandas library in python to find the null values. I kept the file as data frame named 'dataF'. The function I used is 'dataF.isnull().sum()'. After that I got the count of NULL values in each respective column. Later I've dropped them using 'dataF = dataF.dropna()'.

Uploading csv and read the file -



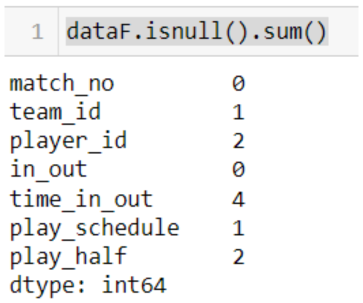
```python
import pandas as pd
dataF= pd.read_csv(r'C:\Users\nandi\Desktop\DBMS Project\filtered_latest_csv\PIO.csv')
```

```python
dataF
```

	match_no	team_id	player_id	in_out	time_in_out	play_schedule	play_half
0	1	1207.0	160151.0	I	66.0	NT	2.0
1	1	1207.0	160160.0	O	66.0	NT	2.0
2	1	1207.0	160161.0	I	77.0	NT	2.0
3	1	1207.0	160161.0	O	77.0	NT	2.0
4	1	1207.0	160157.0	I	2.0	ST	2.0
...	...	...	...	...	...	...	...
585	51	1214.0	160322.0	O	25.0	NT	1.0
586	51	1214.0	160314.0	I	66.0	NT	2.0
587	51	1214.0	160310.0	O	66.0	NT	2.0
588	51	1214.0	160319.0	I	79.0	NT	2.0
589	51	1214.0	160316.0	O	79.0	NT	2.0

590 rows × 7 columns

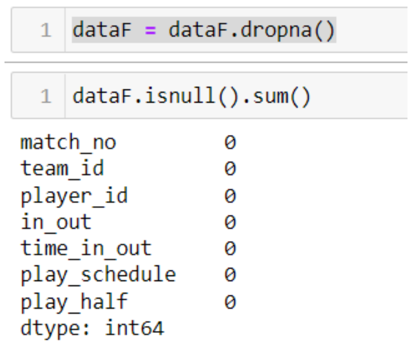
Finding NULL values-



```python
dataF.isnull().sum()
```

```
match_no         0
team_id          1
player_id        2
in_out           0
time_in_out      4
play_schedule    1
play_half        2
dtype: int64
```

Dropping NULL values and checking if NULL is got ridden-



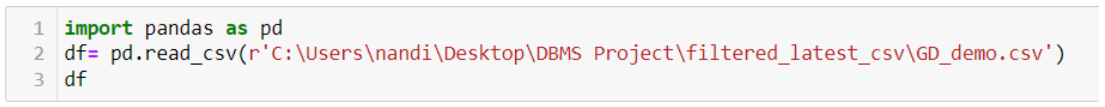
```python
dataF = dataF.dropna()
```

```python
dataF.isnull().sum()
```

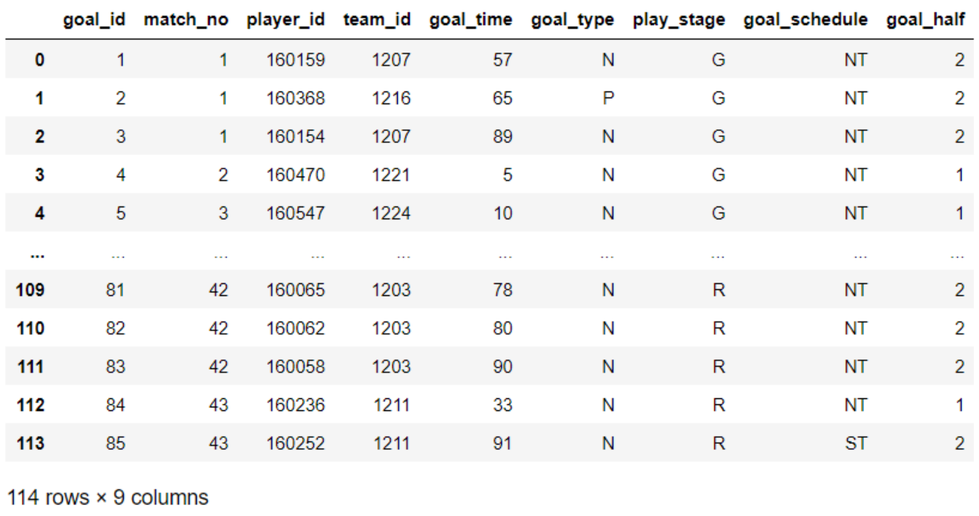
```
match_no         0
team_id          0
player_id        0
in_out           0
time_in_out      0
play_schedule    0
play_half        0
dtype: int64
```

<u>Method-2</u>: After getting rid of NULL values, I've looked for redundant or duplicates entries in the datasets. I got duplicate rows in some datasets. 'duplicate = df[df.duplicated()] function I've used to trace the duplicate or redundant entries. Later I dropped them using ' df=df.drop_duplicates()'.

Uploading csv and read the file -



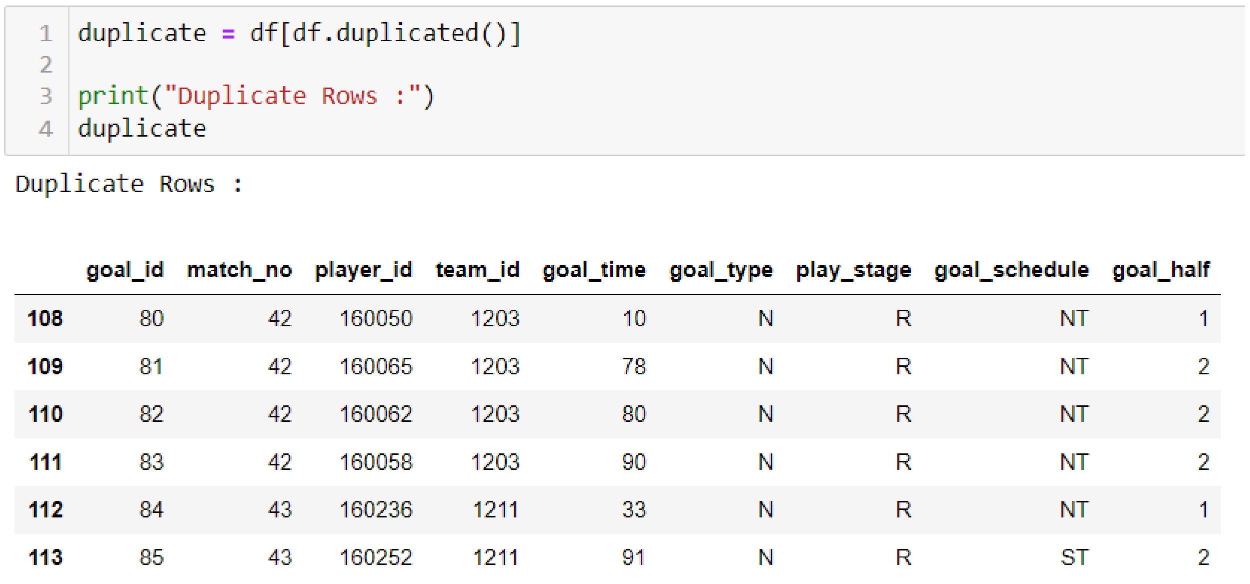
```python
import pandas as pd
df= pd.read_csv(r'C:\Users\nandi\Desktop\DBMS Project\filtered_latest_csv\GD_demo.csv')
df
```



	goal_id	match_no	player_id	team_id	goal_time	goal_type	play_stage	goal_schedule	goal_half
0	1	1	160159	1207	57	N	G	NT	2
1	2	1	160368	1216	65	P	G	NT	2
2	3	1	160154	1207	89	N	G	NT	2
3	4	2	160470	1221	5	N	G	NT	1
4	5	3	160547	1224	10	N	G	NT	1
...	...	...	...	...	...	...	...	...	...
109	81	42	160065	1203	78	N	R	NT	2
110	82	42	160062	1203	80	N	R	NT	2
111	83	42	160058	1203	90	N	R	NT	2
112	84	43	160236	1211	33	N	R	NT	1
113	85	43	160252	1211	91	N	R	ST	2

114 rows × 9 columns

Checking for Redundant entries:

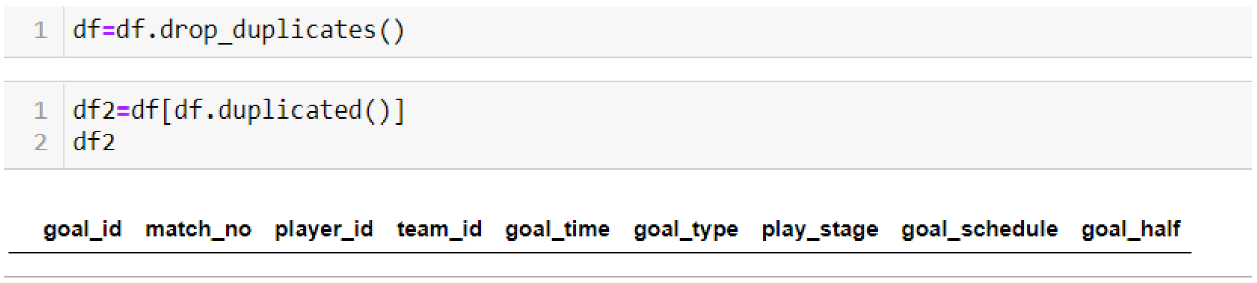


```python
duplicate = df[df.duplicated()]

print("Duplicate Rows :")
duplicate
```

```
Duplicate Rows :
```

	goal_id	match_no	player_id	team_id	goal_time	goal_type	play_stage	goal_schedule	goal_half
108	80	42	160050	1203	10	N	R	NT	1
109	81	42	160065	1203	78	N	R	NT	2
110	82	42	160062	1203	80	N	R	NT	2
111	83	42	160058	1203	90	N	R	NT	2
112	84	43	160236	1211	33	N	R	NT	1
113	85	43	160252	1211	91	N	R	ST	2

Dropping the duplicates and Re-checking:



```python
df=df.drop_duplicates()
```

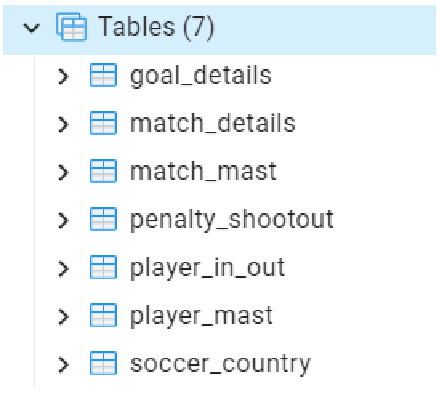
```python
df2=df[df.duplicated()]
df2
```

goal_id	match_no	player_id	team_id	goal_time	goal_type	play_stage	goal_schedule	goal_half

5. Screenshots of the first five rows of all the populated tables. Your screenshots need to show sufficiently show your working environment as well.

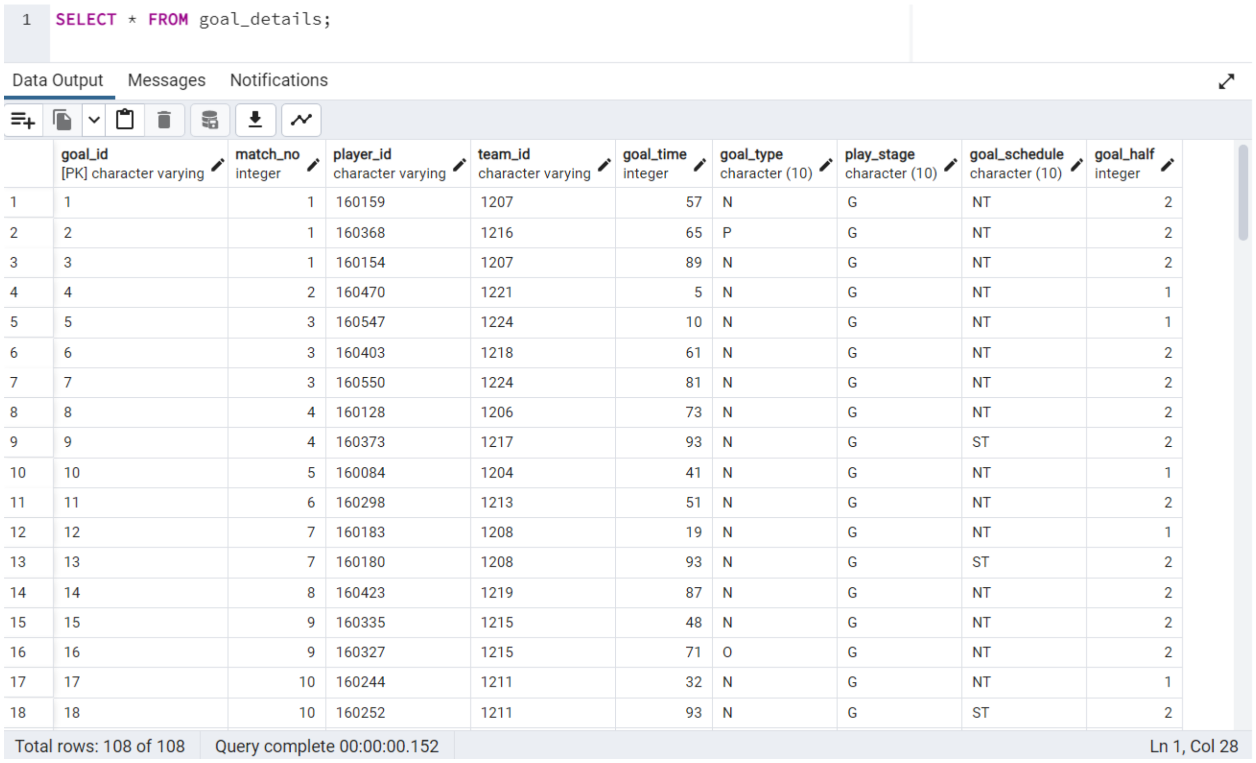
Ans:

❖ *All the relation of my database:*



Tables (7)
- goal_details
- match_details
- match_mast
- penalty_shootout
- player_in_out
- player_mast
- soccer_country

❖ *goal_details:*



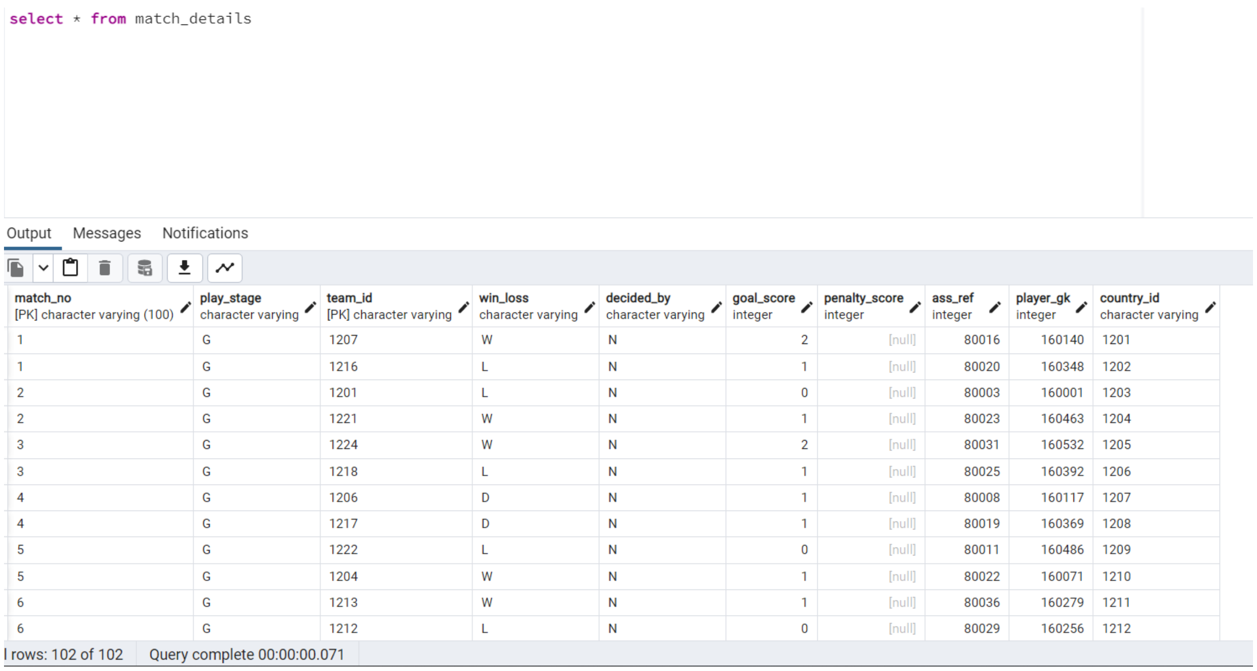
```
SELECT * FROM goal_details;
```

Data Output Messages Notifications

	goal_id [PK] character varying	match_no integer	player_id character varying	team_id character varying	goal_time integer	goal_type character (10)	play_stage character (10)	goal_schedule character (10)	goal_half integer
1	1	1	160159	1207	57	N	G	NT	2
2	2	1	160368	1216	65	P	G	NT	2
3	3	1	160154	1207	89	N	G	NT	2
4	4	2	160470	1221	5	N	G	NT	1
5	5	3	160547	1224	10	N	G	NT	1
6	6	3	160403	1218	61	N	G	NT	2
7	7	3	160550	1224	81	N	G	NT	2
8	8	4	160128	1206	73	N	G	NT	2
9	9	4	160373	1217	93	N	G	ST	2
10	10	5	160084	1204	41	N	G	NT	1
11	11	6	160298	1213	51	N	G	NT	2
12	12	7	160183	1208	19	N	G	NT	1
13	13	7	160180	1208	93	N	G	ST	2
14	14	8	160423	1219	87	N	G	NT	2
15	15	9	160335	1215	48	N	G	NT	2
16	16	9	160327	1215	71	O	G	NT	2
17	17	10	160244	1211	32	N	G	NT	1
18	18	10	160252	1211	93	N	G	ST	2

Total rows: 108 of 108 Query complete 00:00:00.152 Ln 1, Col 28

❖ *match_details:*



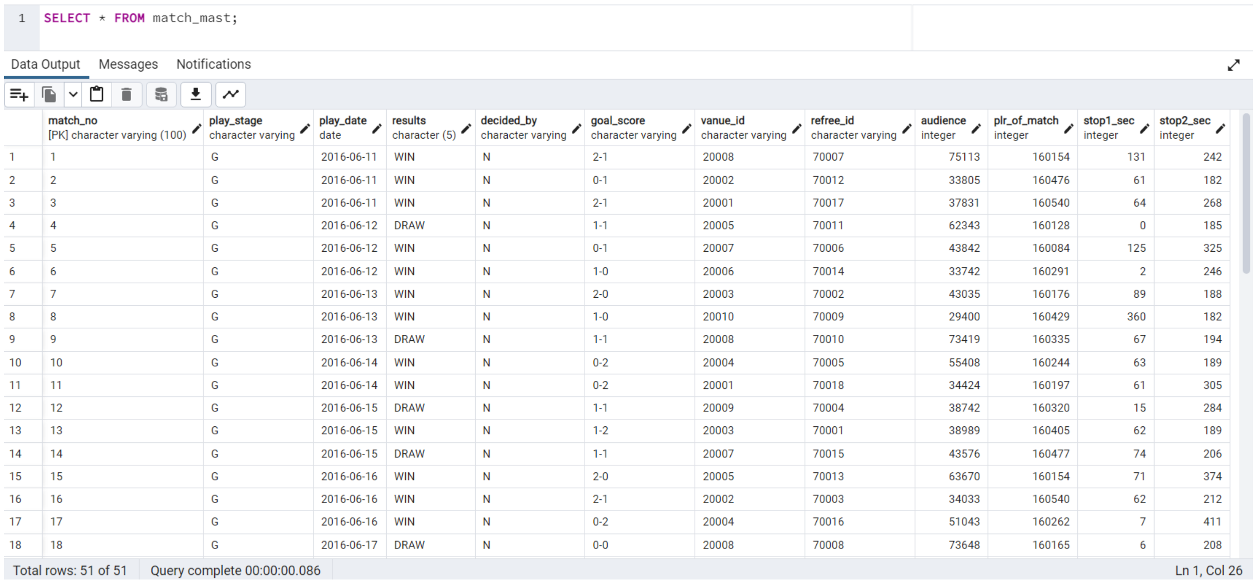
```
select * from match_details
```

Output Messages Notifications

match_no [PK] character varying (100)	play_stage character varying	team_id [PK] character varying	win_loss character varying	decided_by character varying	goal_score integer	penalty_score integer	ass_ref integer	player_gk integer	country_id character varying
1	G	1207	W	N	2	[null]	80016	160140	1201
1	G	1216	L	N	1	[null]	80020	160348	1202
2	G	1201	L	N	0	[null]	80003	160001	1203
2	G	1221	W	N	1	[null]	80023	160463	1204
3	G	1224	W	N	2	[null]	80031	160532	1205
3	G	1218	L	N	1	[null]	80025	160392	1206
4	G	1206	D	N	1	[null]	80008	160117	1207
4	G	1217	D	N	1	[null]	80019	160369	1208
5	G	1222	L	N	0	[null]	80011	160486	1209
5	G	1204	W	N	1	[null]	80022	160071	1210
6	G	1213	W	N	1	[null]	80036	160279	1211
6	G	1212	L	N	0	[null]	80029	160256	1212

l rows: 102 of 102 Query complete 00:00:00.071

- *match_mast:*



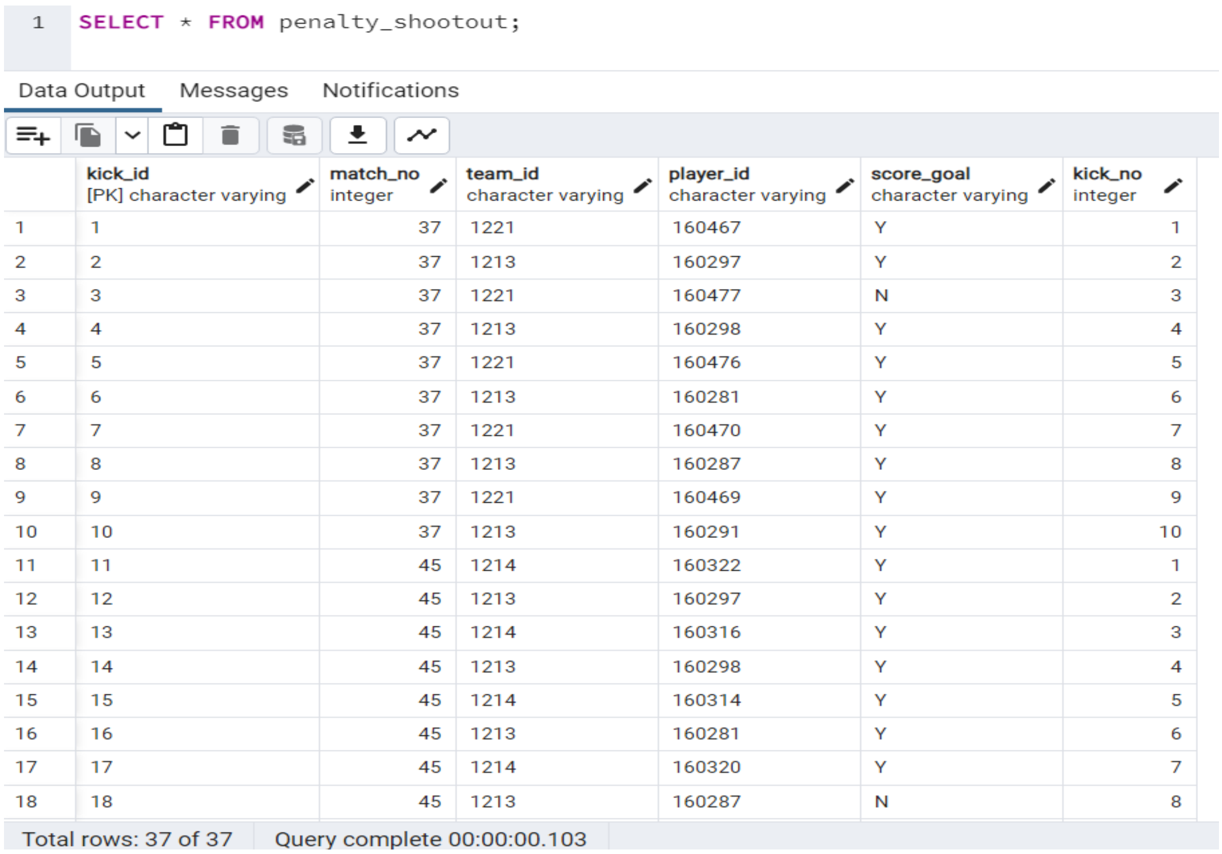
```
SELECT * FROM match_mast;
```

Data Output   Messages   Notifications

	match_no [PK] character varying (100)	play_stage character varying	play_date date	results character (5)	decided_by character varying	goal_score character varying	vanue_id character varying	refree_id character varying	audience integer	plr_of_match integer	stop1_sec integer	stop2_sec integer
1	1	G	2016-06-11	WIN	N	2-1	20008	70007	75113	160154	131	242
2	2	G	2016-06-11	WIN	N	0-1	20002	70012	33805	160476	61	182
3	3	G	2016-06-11	WIN	N	2-1	20001	70017	37831	160540	64	268
4	4	G	2016-06-12	DRAW	N	1-1	20005	70011	62343	160128	0	185
5	5	G	2016-06-12	WIN	N	0-1	20007	70006	43842	160084	125	325
6	6	G	2016-06-12	WIN	N	1-0	20006	70014	33742	160291	2	246
7	7	G	2016-06-13	WIN	N	2-0	20003	70002	43035	160176	89	188
8	8	G	2016-06-13	WIN	N	1-0	20010	70009	29400	160429	360	182
9	9	G	2016-06-13	DRAW	N	1-1	20008	70010	73419	160335	67	194
10	10	G	2016-06-14	WIN	N	0-2	20004	70005	55408	160244	63	189
11	11	G	2016-06-14	WIN	N	0-2	20001	70018	34424	160197	61	305
12	12	G	2016-06-15	DRAW	N	1-1	20009	70004	38742	160320	15	284
13	13	G	2016-06-15	WIN	N	1-2	20003	70001	38989	160405	62	189
14	14	G	2016-06-15	DRAW	N	1-1	20007	70015	43576	160477	74	206
15	15	G	2016-06-16	WIN	N	2-0	20005	70013	63670	160154	71	374
16	16	G	2016-06-16	WIN	N	2-1	20002	70003	34033	160540	62	212
17	17	G	2016-06-16	WIN	N	0-2	20004	70016	51043	160262	7	411
18	18	G	2016-06-17	DRAW	N	0-0	20008	70008	73648	160165	6	208

Total rows: 51 of 51   Query complete 00:00:00.086   Ln 1, Col 26

- *penalty_shootout:*



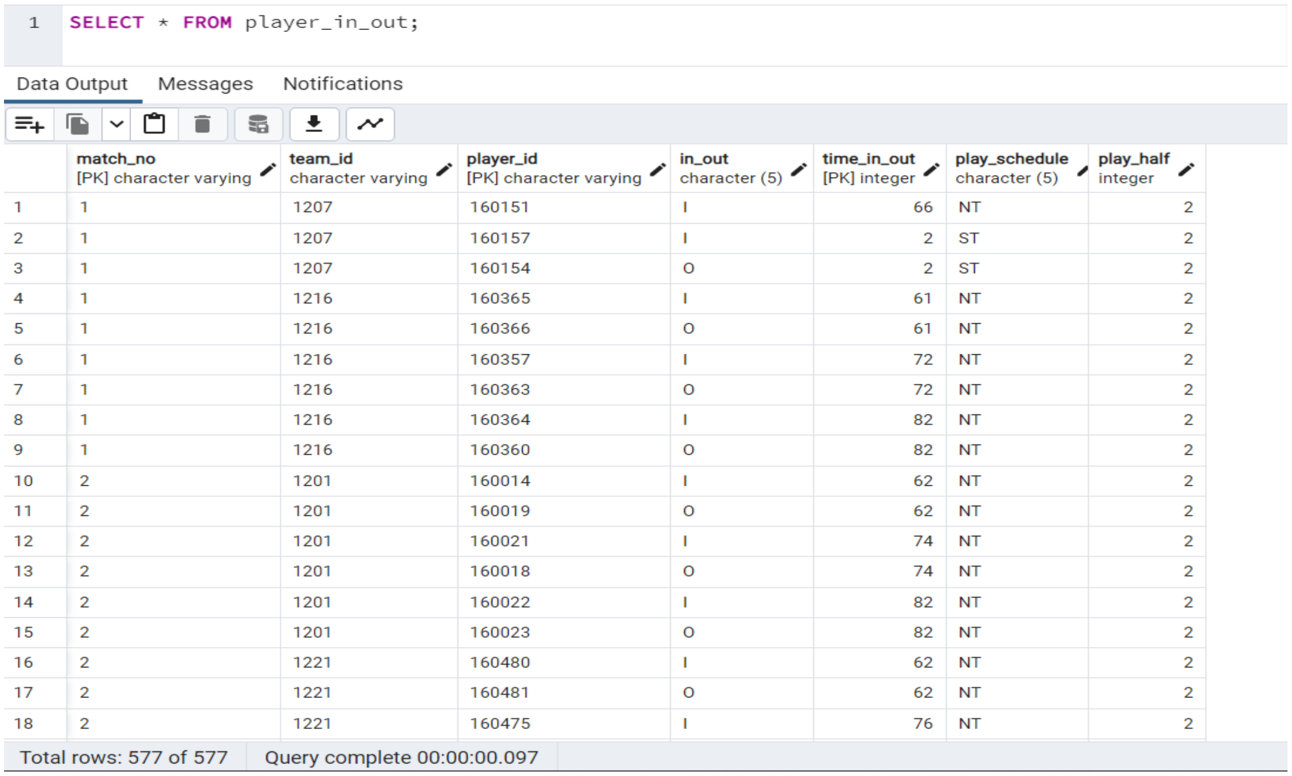
```
SELECT * FROM penalty_shootout;
```

Data Output   Messages   Notifications

	kick_id [PK] character varying	match_no integer	team_id character varying	player_id character varying	score_goal character varying	kick_no integer
1	1	37	1221	160467	Y	1
2	2	37	1213	160297	Y	2
3	3	37	1221	160477	N	3
4	4	37	1213	160298	Y	4
5	5	37	1221	160476	Y	5
6	6	37	1213	160281	Y	6
7	7	37	1221	160470	Y	7
8	8	37	1213	160287	Y	8
9	9	37	1221	160469	Y	9
10	10	37	1213	160291	Y	10
11	11	45	1214	160322	Y	1
12	12	45	1213	160297	Y	2
13	13	45	1214	160316	Y	3
14	14	45	1213	160298	Y	4
15	15	45	1214	160314	Y	5
16	16	45	1213	160281	Y	6
17	17	45	1214	160320	Y	7
18	18	45	1213	160287	N	8

Total rows: 37 of 37   Query complete 00:00:00.103

❖ *player_in_out:*



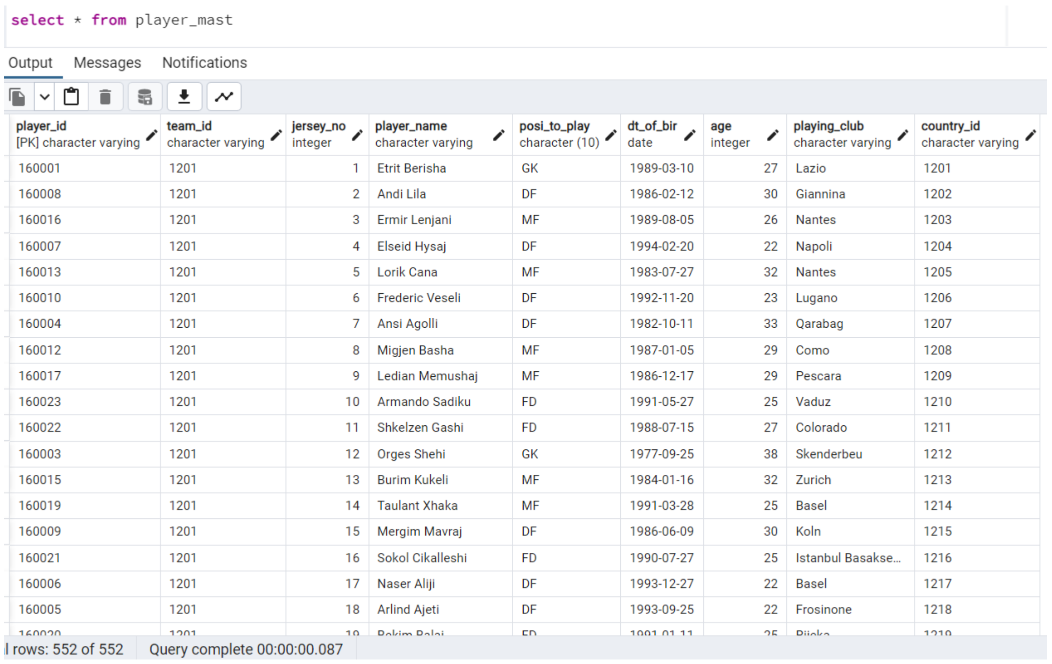
```sql
SELECT * FROM player_in_out;
```

Data Output Messages Notifications

	match_no [PK] character varying	team_id character varying	player_id [PK] character varying	in_out character (5)	time_in_out [PK] integer	play_schedule character (5)	play_half integer
1	1	1207	160151	I	66	NT	2
2	1	1207	160157	I	2	ST	2
3	1	1207	160154	O	2	ST	2
4	1	1216	160365	I	61	NT	2
5	1	1216	160366	O	61	NT	2
6	1	1216	160357	I	72	NT	2
7	1	1216	160363	O	72	NT	2
8	1	1216	160364	I	82	NT	2
9	1	1216	160360	O	82	NT	2
10	2	1201	160014	I	62	NT	2
11	2	1201	160019	O	62	NT	2
12	2	1201	160021	I	74	NT	2
13	2	1201	160018	O	74	NT	2
14	2	1201	160022	I	82	NT	2
15	2	1201	160023	O	82	NT	2
16	2	1221	160480	I	62	NT	2
17	2	1221	160481	O	62	NT	2
18	2	1221	160475	I	76	NT	2

Total rows: 577 of 577 Query complete 00:00:00.097

❖ *player_mast:*



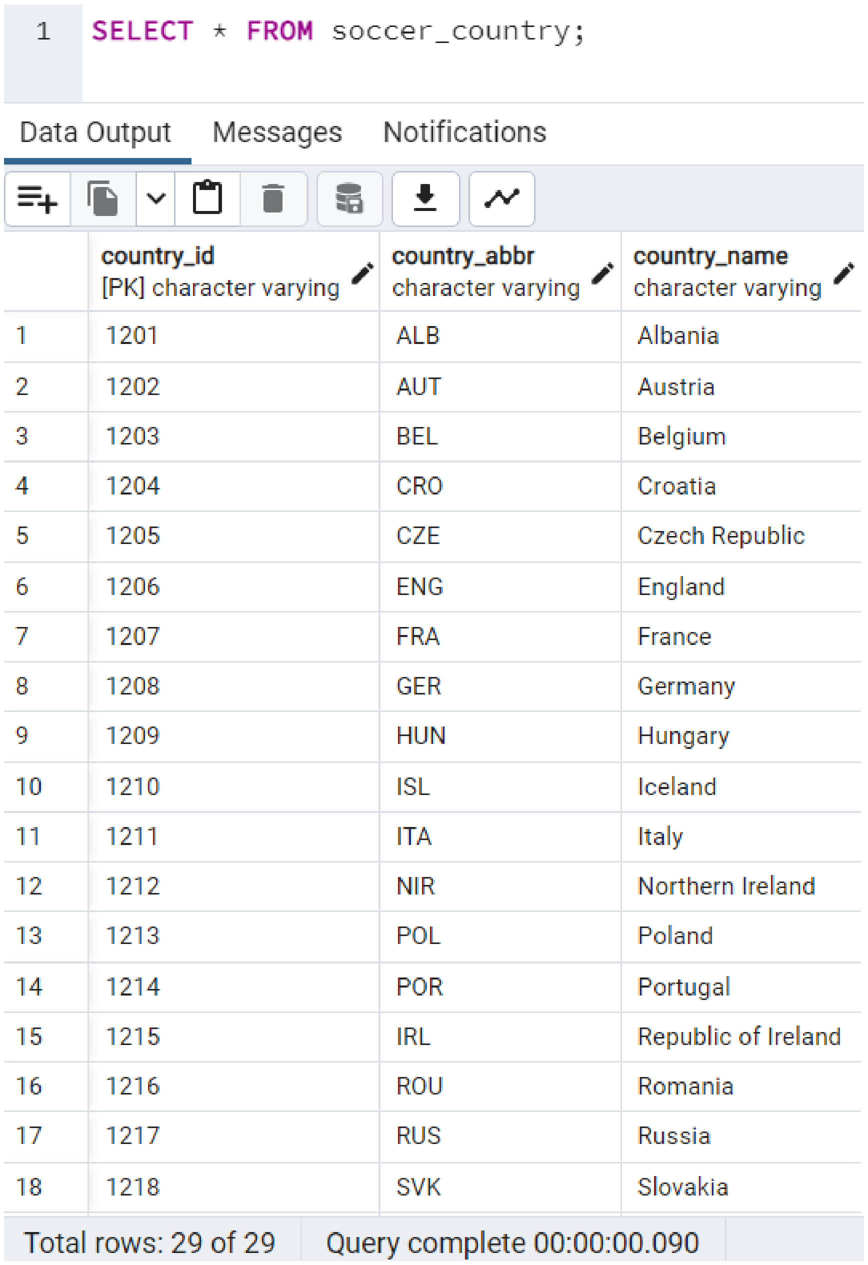
```sql
select * from player_mast
```

Output Messages Notifications

player_id [PK] character varying	team_id character varying	jersey_no integer	player_name character varying	posi_to_play character (10)	dt_of_bir date	age integer	playing_club character varying	country_id character varying
160001	1201	1	Etrit Berisha	GK	1989-03-10	27	Lazio	1201
160008	1201	2	Andi Lila	DF	1986-02-12	30	Giannina	1202
160016	1201	3	Ermir Lenjani	MF	1989-08-05	26	Nantes	1203
160007	1201	4	Elseid Hysaj	DF	1994-02-20	22	Napoli	1204
160013	1201	5	Lorik Cana	MF	1983-07-27	32	Nantes	1205
160010	1201	6	Frederic Veseli	DF	1992-11-20	23	Lugano	1206
160004	1201	7	Ansi Agolli	DF	1982-10-11	33	Qarabag	1207
160012	1201	8	Migjen Basha	MF	1987-01-05	29	Como	1208
160017	1201	9	Ledian Memushaj	MF	1986-12-17	29	Pescara	1209
160023	1201	10	Armando Sadiku	FD	1991-05-27	25	Vaduz	1210
160022	1201	11	Shkelzen Gashi	FD	1988-07-15	27	Colorado	1211
160003	1201	12	Orges Shehi	GK	1977-09-25	38	Skenderbeu	1212
160015	1201	13	Burim Kukeli	MF	1984-01-16	32	Zurich	1213
160019	1201	14	Taulant Xhaka	MF	1991-03-28	25	Basel	1214
160009	1201	15	Mergim Mavraj	DF	1986-06-09	30	Koln	1215
160021	1201	16	Sokol Cikalleshi	FD	1990-07-27	25	Istanbul Basakse...	1216
160006	1201	17	Naser Aliji	DF	1993-12-27	22	Basel	1217
160005	1201	18	Arlind Ajeti	DF	1993-09-25	22	Frosinone	1218

l rows: 552 of 552 Query complete 00:00:00.087

❖ *soccer_country:*


```
SELECT * FROM soccer_country;
```

Data Output | Messages | Notifications

	country_id [PK] character varying	country_abbr character varying	country_name character varying
1	1201	ALB	Albania
2	1202	AUT	Austria
3	1203	BEL	Belgium
4	1204	CRO	Croatia
5	1205	CZE	Czech Republic
6	1206	ENG	England
7	1207	FRA	France
8	1208	GER	Germany
9	1209	HUN	Hungary
10	1210	ISL	Iceland
11	1211	ITA	Italy
12	1212	NIR	Northern Ireland
13	1213	POL	Poland
14	1214	POR	Portugal
15	1215	IRL	Republic of Ireland
16	1216	ROU	Romania
17	1217	RUS	Russia
18	1218	SVK	Slovakia

Total rows: 29 of 29 Query complete 00:00:00.090

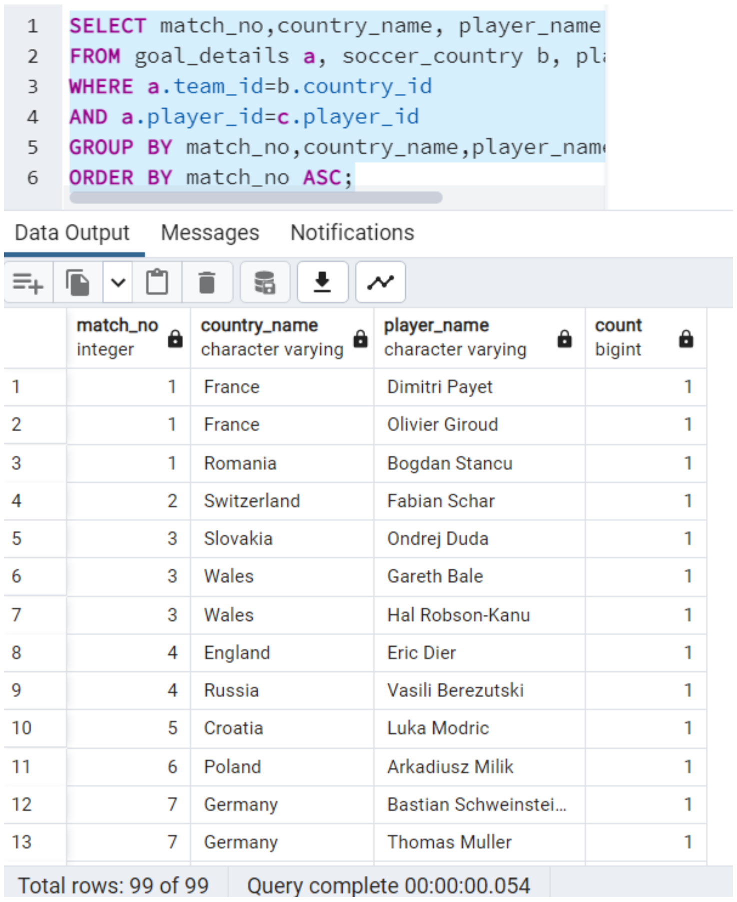
6. Your 10 English questions and 10 query execution of those questions with screenshots of the first five rows of output and the total number of rows in the result. Design a variety of complex questions, as mentioned earlier.

   1. Write a SQL query to find the players who scored the most goals in each match. Group the result set on match number and player name. Sort the result-set in ascending order by match number. Return match number, country name, player name and number of matches.
   *[**Changed question. Reason**: wanted to make a bit complicated query compared to the question I mentioned in part 1]*

ANS:

```
SELECT match_no,country_name, player_name, COUNT(match_no)
FROM goal_details a, soccer_country b, player_mast c
WHERE a.team_id=b.country_id
AND a.player_id=c.player_id
GROUP BY match_no,country_name,player_name
ORDER BY match_no ASC;
```

OUTPUT:



```
SELECT match_no,country_name, player_name
FROM goal_details a, soccer_country b, pl
WHERE a.team_id=b.country_id
AND a.player_id=c.player_id
GROUP BY match_no,country_name,player_nam
ORDER BY match_no ASC;
```

	match_no integer	country_name character varying	player_name character varying	count bigint
1	1	France	Dimitri Payet	1
2	1	France	Olivier Giroud	1
3	1	Romania	Bogdan Stancu	1
4	2	Switzerland	Fabian Schar	1
5	3	Slovakia	Ondrej Duda	1
6	3	Wales	Gareth Bale	1
7	3	Wales	Hal Robson-Kanu	1
8	4	England	Eric Dier	1
9	4	Russia	Vasili Berezutski	1
10	5	Croatia	Luka Modric	1
11	6	Poland	Arkadiusz Milik	1
12	7	Germany	Bastian Schweinstei...	1
13	7	Germany	Thomas Muller	1

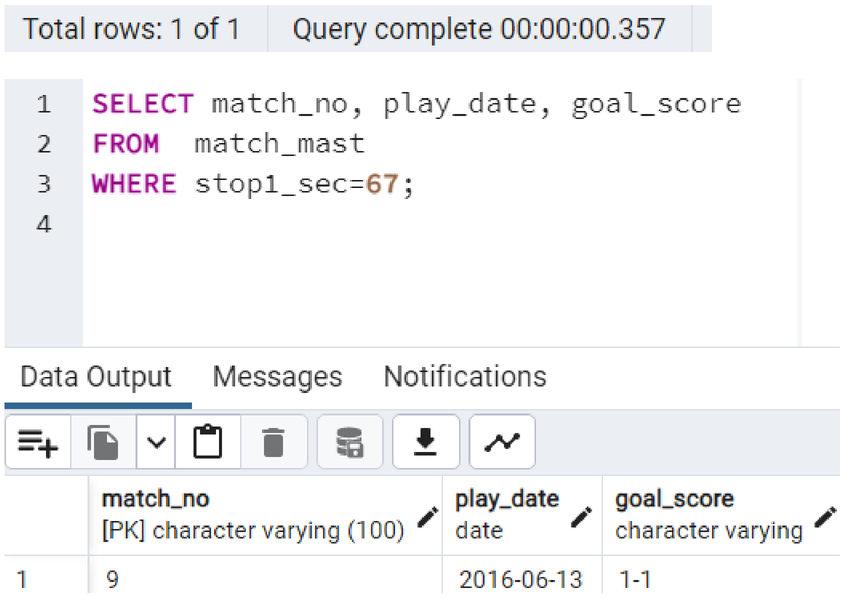
Total rows: 99 of 99 Query complete 00:00:00.054

2. Find the match no, date of play of a match that had 67 sec stoppage time in the first half of play. Return the game number, date of play, and goal scored.

ANS:

```
SELECT match_no, play_date, goal_score
FROM  match_mast
WHERE stop1_sec=67;
```

OUTPUT:



Total rows: 1 of 1 | Query complete 00:00:00.357

```
SELECT match_no, play_date, goal_score
FROM  match_mast
WHERE stop1_sec=67;
```

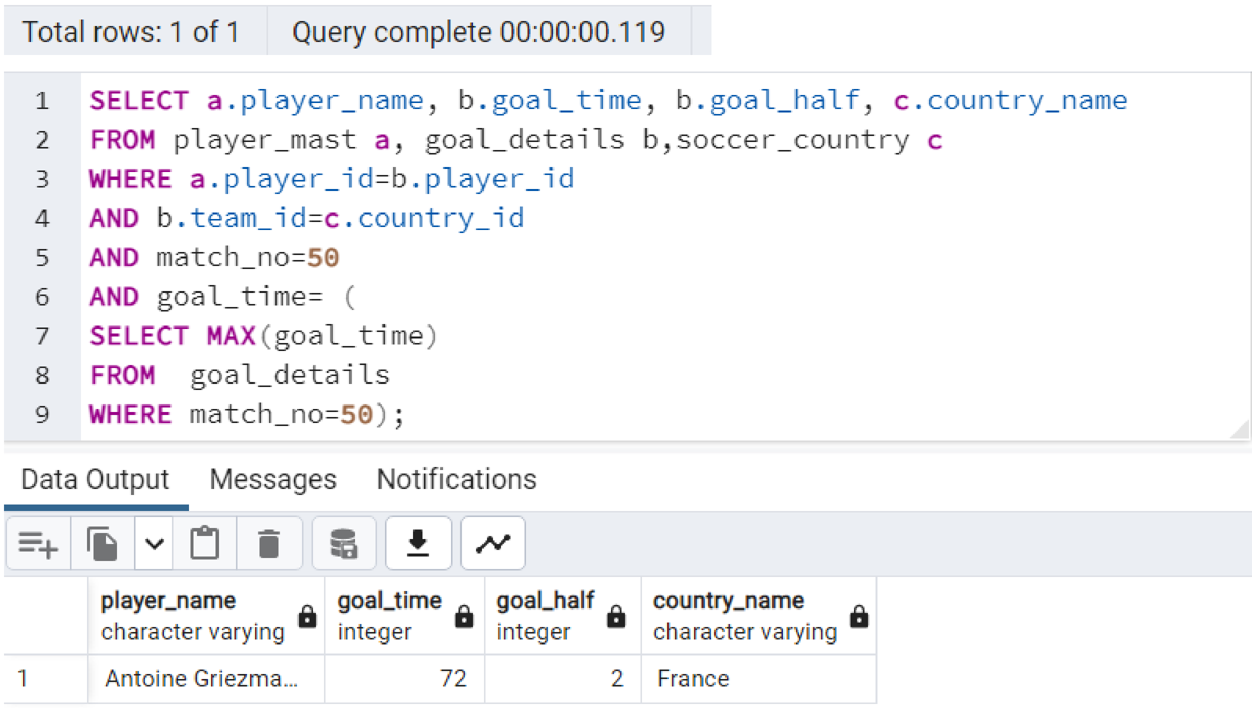
	match_no [PK] character varying (100)	play_date date	goal_score character varying
1	9	2016-06-13	1-1

3. Write a SQL query to find the players who scored the last goal in the second semi-final, i.e., the 50th match of the tournament. Return player name, goal time, goal half, country name.

*[**Changed question. Reason**: I noticed that 3rd and 4th query were kind of similar in part 1, so changed the 3rd query and made a question where I can use aggregate operator]*

ANS:
```
SELECT a.player_name, b.goal_time, b.goal_half, c.country_name
FROM player_mast a, goal_details b,soccer_country c
WHERE a.player_id=b.player_id
AND b.team_id=c.country_id
AND match_no=50
AND goal_time= (
SELECT MAX(goal_time)
FROM  goal_details
WHERE match_no=50);
```

OUTPUT:



Total rows: 1 of 1   Query complete 00:00:00.119

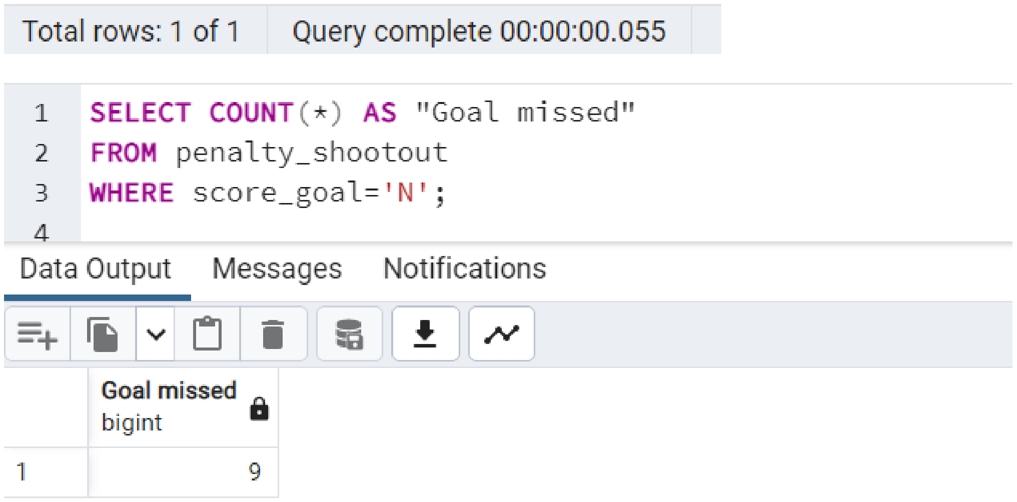
```
SELECT a.player_name, b.goal_time, b.goal_half, c.country_name
FROM player_mast a, goal_details b,soccer_country c
WHERE a.player_id=b.player_id
AND b.team_id=c.country_id
AND match_no=50
AND goal_time= (
SELECT MAX(goal_time)
FROM  goal_details
WHERE match_no=50);
```

	player_name character varying	goal_time integer	goal_half integer	country_name character varying
1	Antoine Griezma...	72	2	France

4. Write a SQL query to count the number of shots missed or saved in penalty shootout matches. Return number of shots missed as "Goal missed".

ANS:
```
SELECT COUNT(*) AS "Goal missed"
FROM penalty_shootout
WHERE score_goal='N';
```

OUTPUT:



Total rows: 1 of 1 Query complete 00:00:00.055

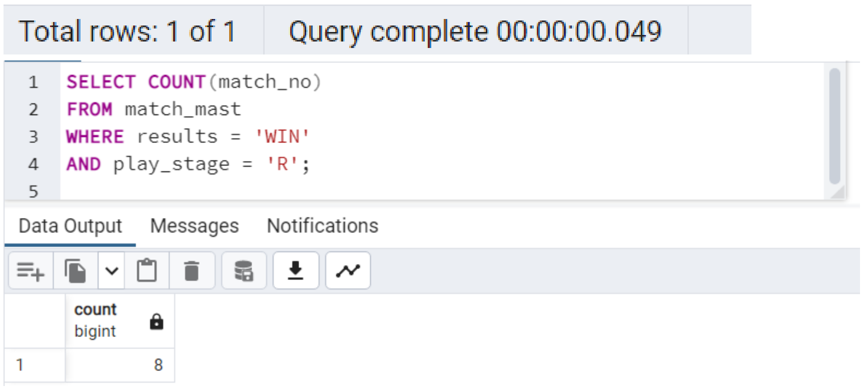
```
SELECT COUNT(*) AS "Goal missed"
FROM penalty_shootout
WHERE score_goal='N';
```

	Goal missed bigint
1	9

5. Create a query to count the number of matches that finished with a win in the 'Round of 16'.

ANS:
```
SELECT COUNT(match_no)
FROM match_mast
WHERE results = 'WIN'
AND play_stage = 'R';
```

OUTPUT:



Total rows: 1 of 1 Query complete 00:00:00.049

```
SELECT COUNT(match_no)
FROM match_mast
WHERE results = 'WIN'
AND play_stage = 'R';
```

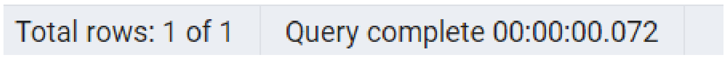
	count bigint
1	8

6. Write a SQL query to find out who scored the most goals in the tournament. Return player name, country name and highest individual scorer.
   *[**Changed question. Reason**: The query is near to similar to which I've mentioned in part-1, but made little changes in order to use 'subquery' and make it a bit complicated]*

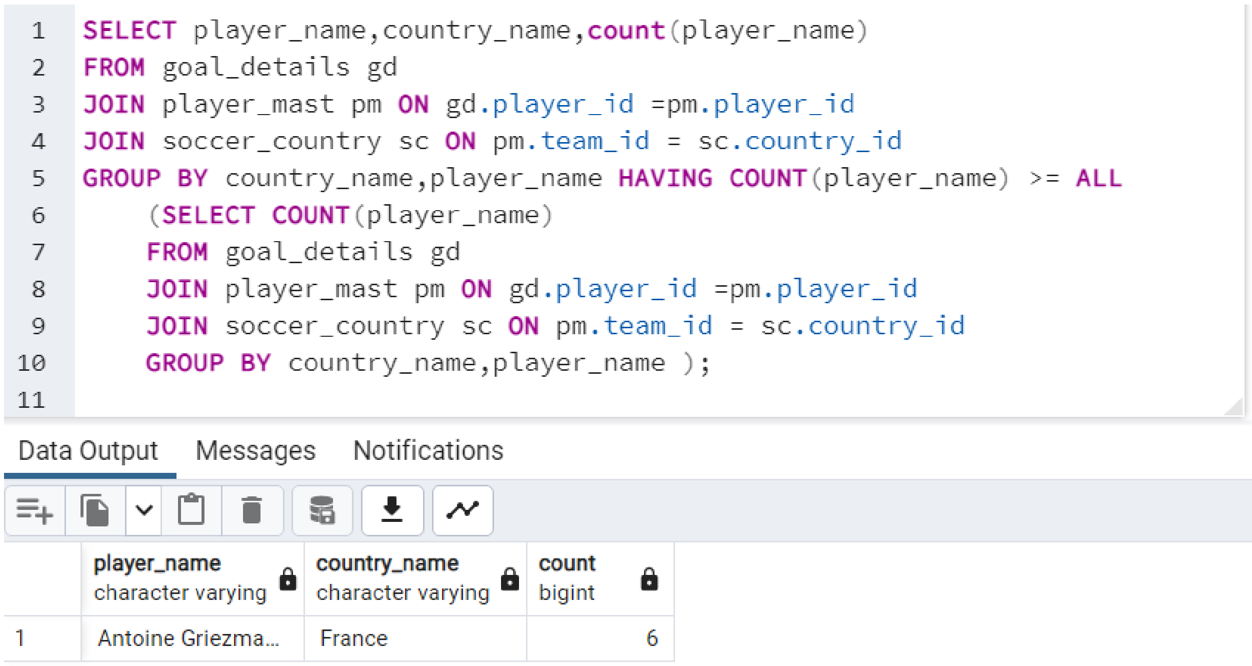
ANS:

```
SELECT player_name,country_name,count(player_name)
FROM goal_details gd
JOIN player_mast pm ON gd.player_id =pm.player_id
JOIN soccer_country sc ON pm.team_id = sc.country_id
GROUP BY country_name,player_name HAVING COUNT(player_name) >= ALL
        (SELECT COUNT(player_name)
        FROM goal_details gd
        JOIN player_mast pm ON gd.player_id =pm.player_id
        JOIN soccer_country sc ON pm.team_id = sc.country_id
        GROUP BY country_name,player_name );
```

OUTPUT:



Total rows: 1 of 1   Query complete 00:00:00.072



```
SELECT player_name,country_name,count(player_name)
FROM goal_details gd
JOIN player_mast pm ON gd.player_id =pm.player_id
JOIN soccer_country sc ON pm.team_id = sc.country_id
GROUP BY country_name,player_name HAVING COUNT(player_name) >= ALL
    (SELECT COUNT(player_name)
    FROM goal_details gd
    JOIN player_mast pm ON gd.player_id =pm.player_id
    JOIN soccer_country sc ON pm.team_id = sc.country_id
    GROUP BY country_name,player_name );
```

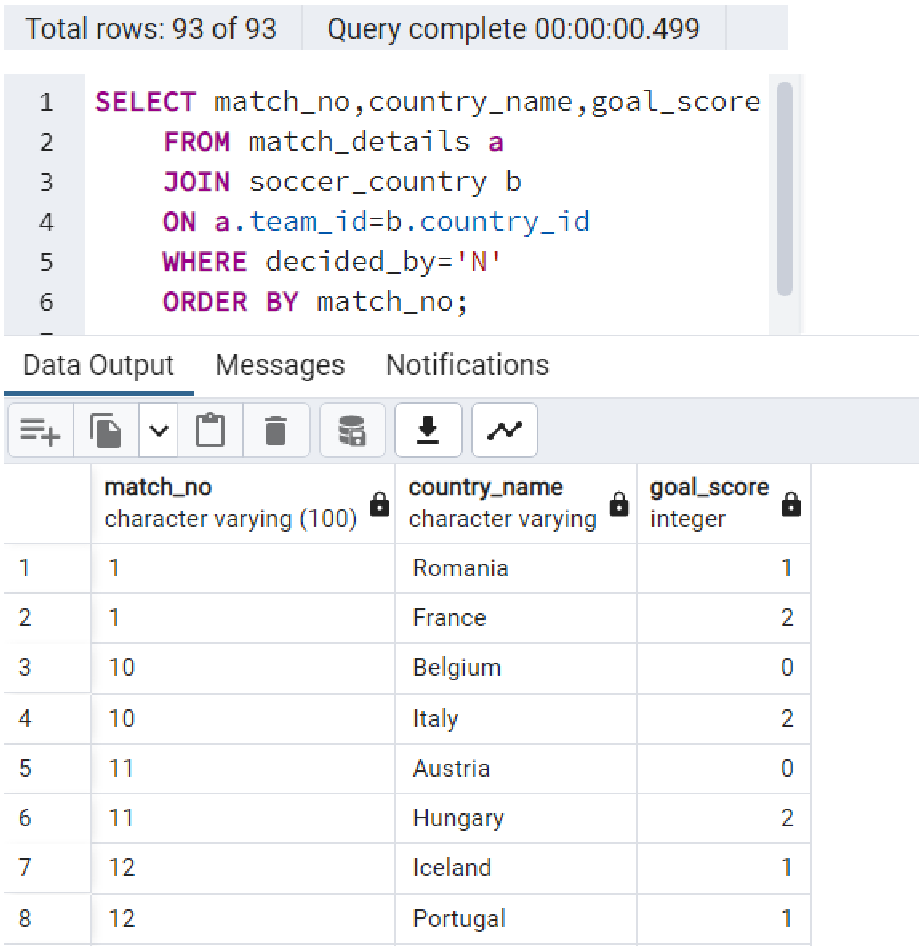
	player_name character varying	country_name character varying	count bigint
1	Antoine Griezma...	France	6

7. Write a SQL query to find the number of goals scored by each team in each match during normal play. Return match number, country name and goal score.
*[**Changed question. Reason**: I wasn't aware about the topics we've to cover in part-2, in order to fulfill the topic requirement, I've changed this question, to make a query using 'order by']*

ANS:

```
SELECT match_no,country_name,goal_score
FROM match_details a
JOIN soccer_country b
ON a.team_id=b.country_id
WHERE decided_by='N'
ORDER BY match_no;
```

OUTPUT:



Total rows: 93 of 93 Query complete 00:00:00.499

```
SELECT match_no,country_name,goal_score
    FROM match_details a
    JOIN soccer_country b
    ON a.team_id=b.country_id
    WHERE decided_by='N'
    ORDER BY match_no;
```

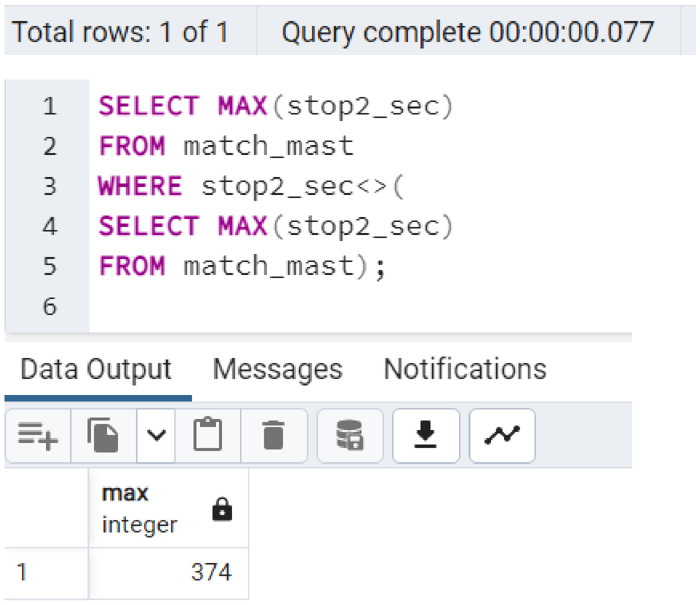
	match_no character varying (100)	country_name character varying	goal_score integer
1	1	Romania	1
2	1	France	2
3	10	Belgium	0
4	10	Italy	2
5	11	Austria	0
6	11	Hungary	2
7	12	Iceland	1
8	12	Portugal	1

8. Write a query to find the 2nd highest stoppage time which have been added in the 2nd half of play

ANS:

```
SELECT MAX(stop2_sec)
FROM match_mast
WHERE stop2_sec<>(
SELECT MAX(stop2_sec)
FROM match_mast);
```

OUTPUT:



Total rows: 1 of 1   Query complete 00:00:00.077

```
SELECT MAX(stop2_sec)
FROM match_mast
WHERE stop2_sec<>(
SELECT MAX(stop2_sec)
FROM match_mast);
```

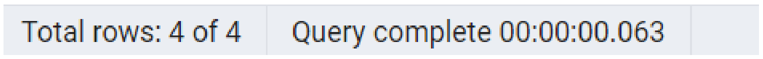
	max integer
1	374

9. Write a query to find the final four teams in the tournament. Return country name.
[***Changed question. Reason****: This question is also similar to the question I've mentioned in part-1 but the database I got, it had 17 tables, as I didn't use that much tables, I changes this question according to the datasets I've chosen]*

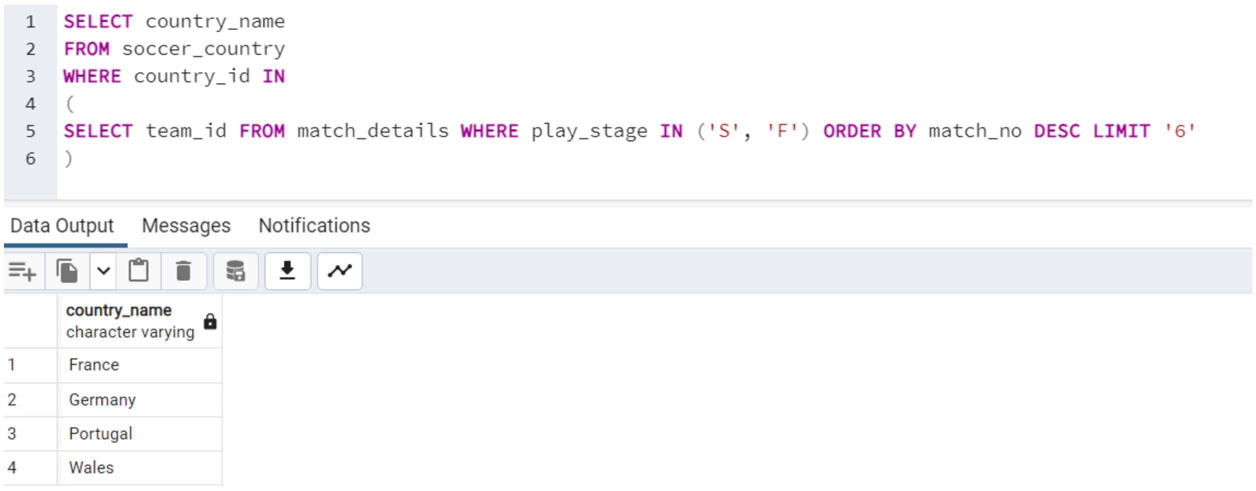
ANS:

```
SELECT country_name
FROM soccer_country
WHERE country_id IN
(
SELECT team_id FROM match_details WHERE play_stage IN ('S', 'F') ORDER BY match_no
DESC LIMIT '6')
```

OUTPUT:



Total rows: 4 of 4 Query complete 00:00:00.063



```
SELECT country_name
FROM soccer_country
WHERE country_id IN
(
SELECT team_id FROM match_details WHERE play_stage IN ('S', 'F') ORDER BY match_no DESC LIMIT '6'
)
```

Data Output Messages Notifications

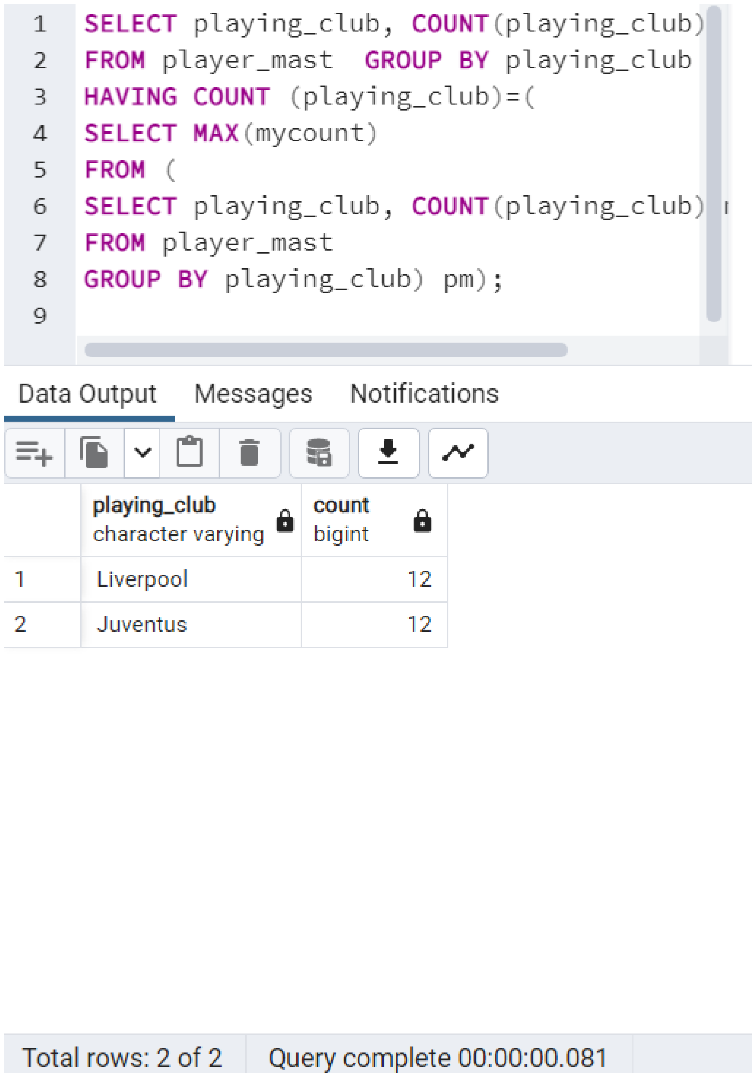
	country_name character varying
1	France
2	Germany
3	Portugal
4	Wales

10. write a SQL query to find the club, which supplied the greatest number of players to the tournament. Return club name, number of players.
[***Changed question. Reason**: The reason behind changing this question is almost same as the previous query and moreover I wanted to use aggregate function and subqueries together to make it a bit complicated.]*

ANS:

```
SELECT playing_club, COUNT(playing_club)
FROM player_mast  GROUP BY playing_club
HAVING COUNT (playing_club)=(
SELECT MAX(mycount)
FROM (
SELECT playing_club, COUNT(playing_club) mycount
FROM player_mast
GROUP BY playing_club) pm);
```

OUTPUT:



```
SELECT playing_club, COUNT(playing_club)
FROM player_mast  GROUP BY playing_club
HAVING COUNT (playing_club)=(
SELECT MAX(mycount)
FROM (
SELECT playing_club, COUNT(playing_club)
FROM player_mast
GROUP BY playing_club) pm);
```

	playing_club character varying	count bigint
1	Liverpool	12
2	Juventus	12

Total rows: 2 of 2   Query complete 00:00:00.081