NBA Statistics

Author: Nathan Metens

The NBA Schema

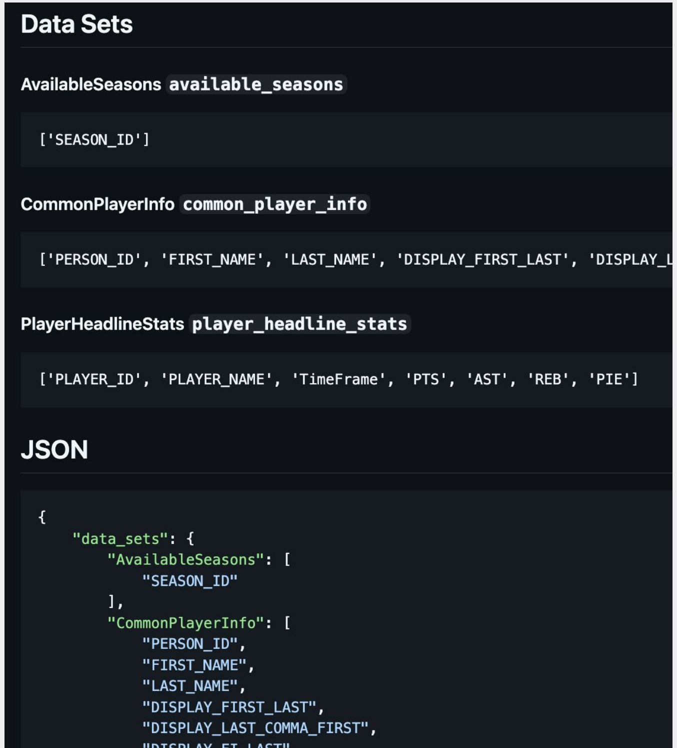
- I love basketball.
- Keep track of player statistical data on nba.com.
  - PPG - Points Per Game
  - 3P% - 3 Point Percentage
- Players of interest:
  - Anthony Edwards
  - Jalen Brunson
- Where I got the data? nba_api endpoints:
  - players
  - Playergamestats
  - Huge amounts of up to date statistics
  - Extensive documentation
  - Python compatible

Plan of Extraction

- Original data: basketball-reference
  - Pros:
    - csv files easily available
    - csv to sql methods available
  - Cons:
    - would take too long to extract 1,000 csv files manually
- Instead, use Python and create a GitHub repository.
- nba_api is a Python module, connecting to nba.com.
- Use API calls to gather rows of data.

Extraction Step 1)

- Create the tables:



Data Sets

AvailableSeasons `available_seasons`

```
['SEASON_ID']
```

CommonPlayerInfo `common_player_info`

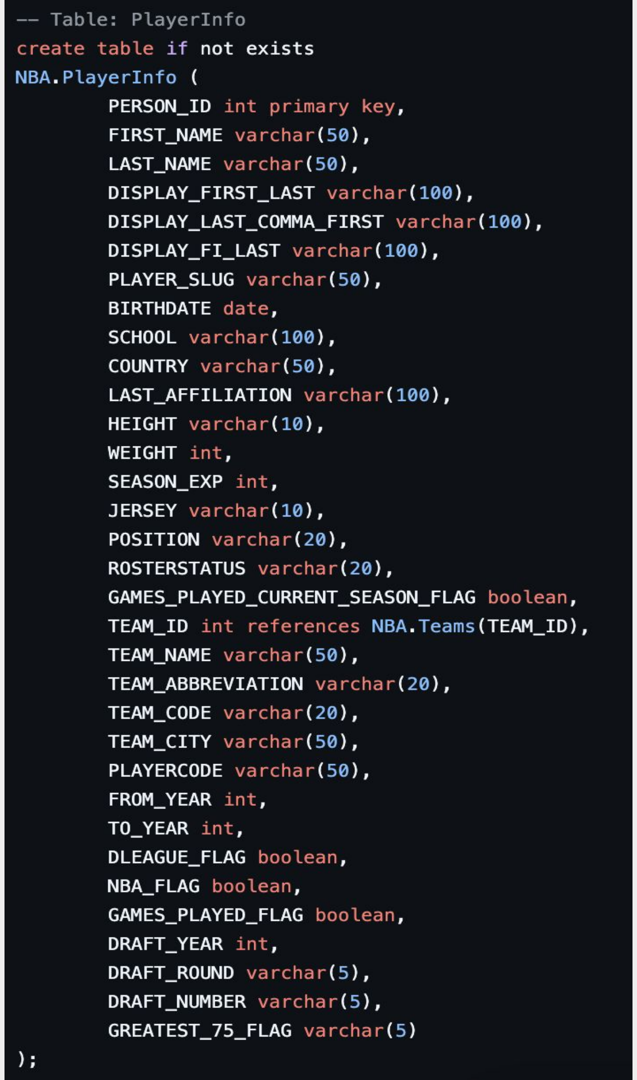
```
['PERSON_ID', 'FIRST_NAME', 'LAST_NAME', 'DISPLAY_FIRST_LAST', 'DISPLAY_L
```

PlayerHeadlineStats `player_headline_stats`

```
['PLAYER_ID', 'PLAYER_NAME', 'TimeFrame', 'PTS', 'AST', 'REB', 'PIE']
```

JSON

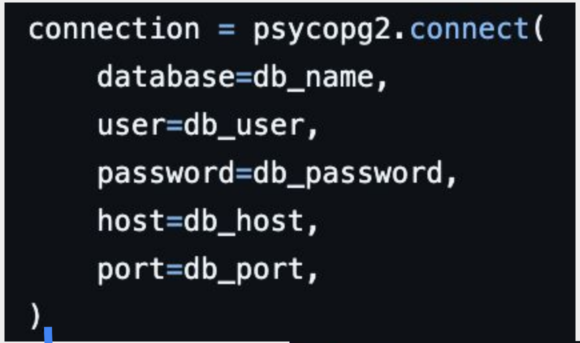
```
{
    "data_sets": {
        "AvailableSeasons": [
            "SEASON_ID"
        ],
        "CommonPlayerInfo": [
            "PERSON_ID",
            "FIRST_NAME",
            "LAST_NAME",
            "DISPLAY_FIRST_LAST",
            "DISPLAY_LAST_COMMA_FIRST",
            "DISPLAY_FI_LAST"
```



```sql
-- Table: PlayerInfo
create table if not exists
NBA.PlayerInfo (
        PERSON_ID int primary key,
        FIRST_NAME varchar(50),
        LAST_NAME varchar(50),
        DISPLAY_FIRST_LAST varchar(100),
        DISPLAY_LAST_COMMA_FIRST varchar(100),
        DISPLAY_FI_LAST varchar(100),
        PLAYER_SLUG varchar(50),
        BIRTHDATE date,
        SCHOOL varchar(100),
        COUNTRY varchar(50),
        LAST_AFFILIATION varchar(100),
        HEIGHT varchar(10),
        WEIGHT int,
        SEASON_EXP int,
        JERSEY varchar(10),
        POSITION varchar(20),
        ROSTERSTATUS varchar(20),
        GAMES_PLAYED_CURRENT_SEASON_FLAG boolean,
        TEAM_ID int references NBA.Teams(TEAM_ID),
        TEAM_NAME varchar(50),
        TEAM_ABBREVIATION varchar(20),
        TEAM_CODE varchar(20),
        TEAM_CITY varchar(50),
        PLAYERCODE varchar(50),
        FROM_YEAR int,
        TO_YEAR int,
        DLEAGUE_FLAG boolean,
        NBA_FLAG boolean,
        GAMES_PLAYED_FLAG boolean,
        DRAFT_YEAR int,
        DRAFT_ROUND varchar(5),
        DRAFT_NUMBER varchar(5),
        GREATEST_75_FLAG varchar(5)
);
```

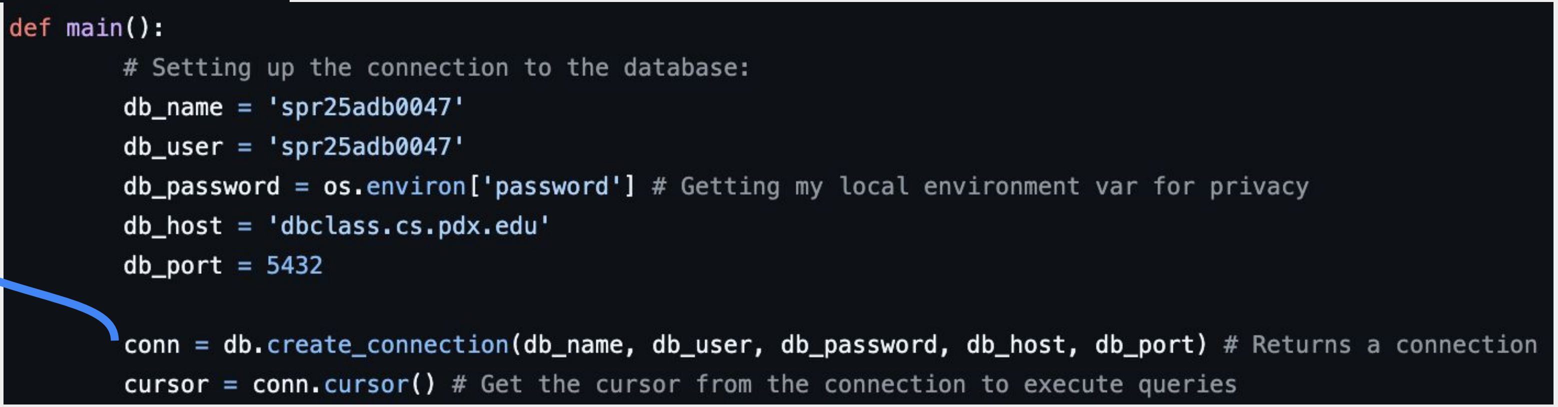
Extraction Step 2)

- Connect to the PostgreSQL database through psycopg2.



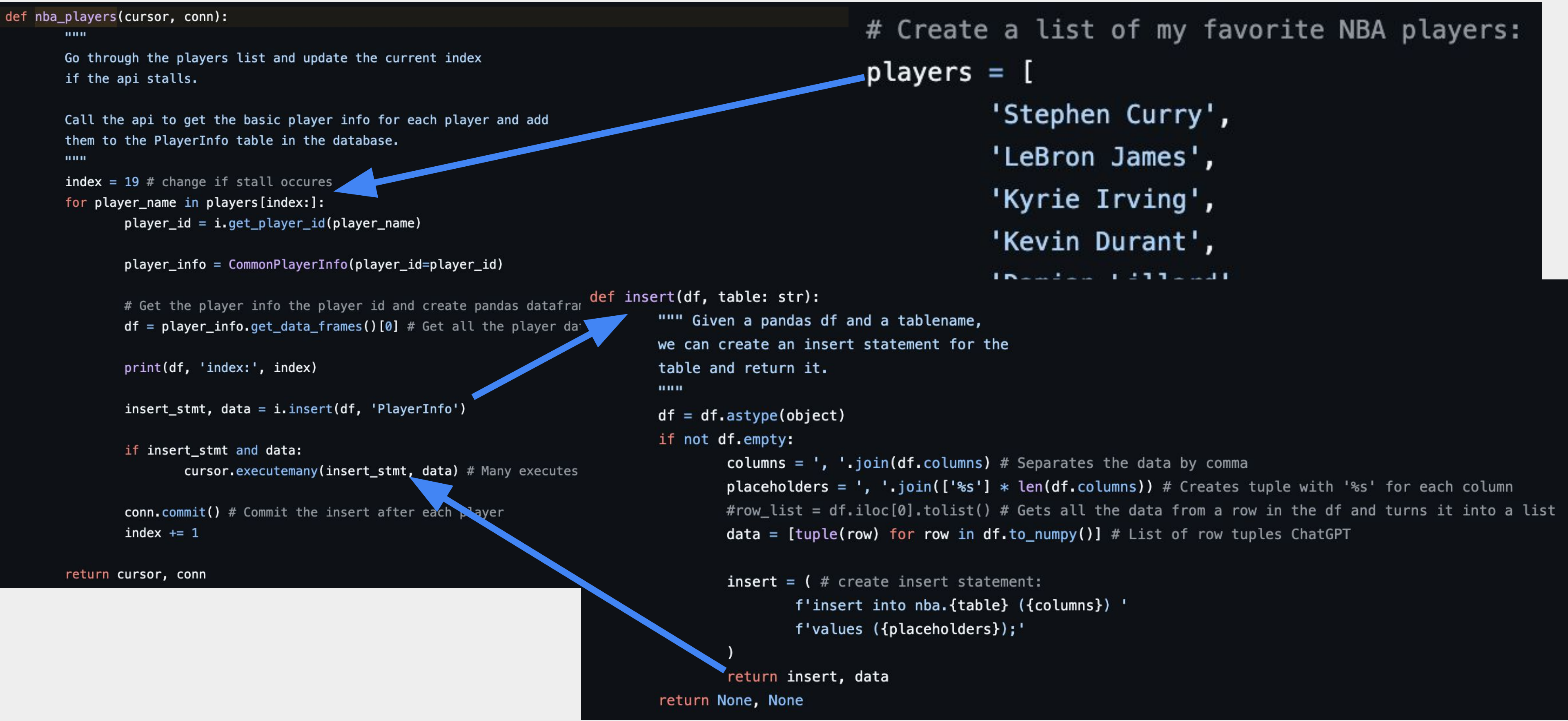
```
connection = psycopg2.connect(
    database=db_name,
    user=db_user,
    password=db_password,
    host=db_host,
    port=db_port,
)
```

- A "popular PostgreSQL database adapter for Python."
- Allows remote connection.



```
def main():
        # Setting up the connection to the database:
        db_name = 'spr25adb0047'
        db_user = 'spr25adb0047'
        db_password = os.environ['password'] # Getting my local environment var for privacy
        db_host = 'dbclass.cs.pdx.edu'
        db_port = 5432

        conn = db.create_connection(db_name, db_user, db_password, db_host, db_port) # Returns a connection
        cursor = conn.cursor() # Get the cursor from the connection to execute queries
```

Extraction Step 3) Call the nba_api endpoint and commit changes:



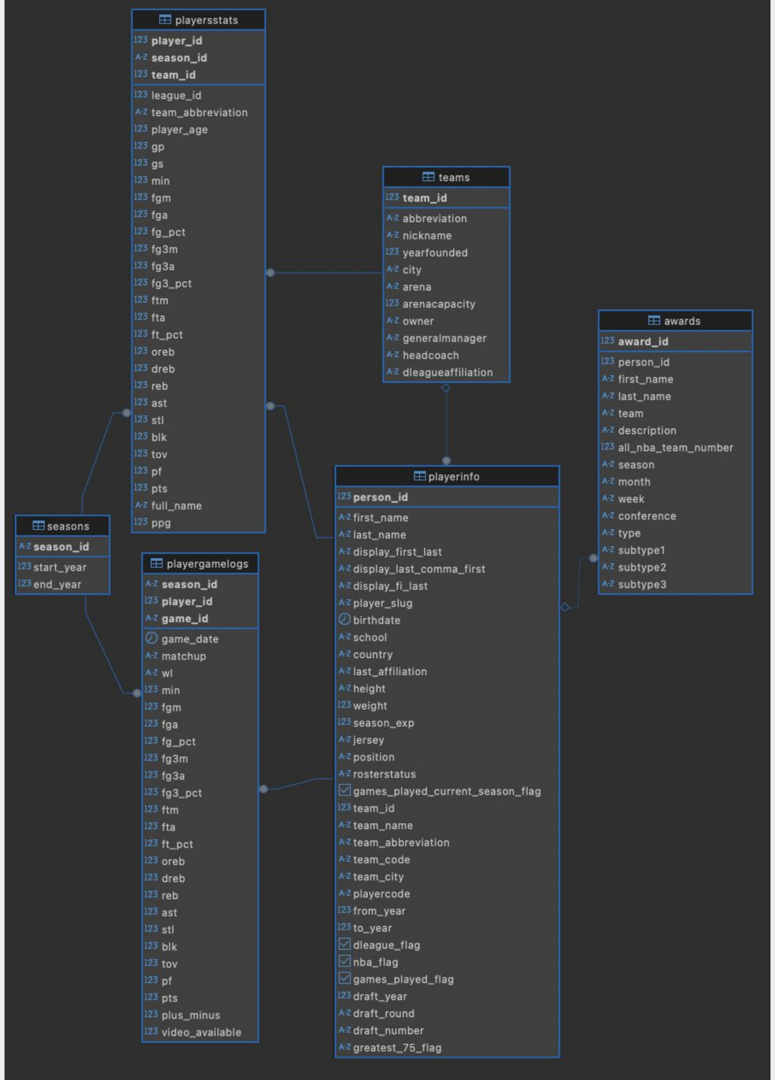
```python
def nba_players(cursor, conn):
    """
    Go through the players list and update the current index
    if the api stalls.

    Call the api to get the basic player info for each player and add
    them to the PlayerInfo table in the database.
    """
    index = 19 # change if stall occures
    for player_name in players[index:]:
        player_id = i.get_player_id(player_name)

        player_info = CommonPlayerInfo(player_id=player_id)

        # Get the player info the player id and create pandas datafra
        df = player_info.get_data_frames()[0] # Get all the player da

        print(df, 'index:', index)

        insert_stmt, data = i.insert(df, 'PlayerInfo')

        if insert_stmt and data:
            cursor.executemany(insert_stmt, data) # Many executes

        conn.commit() # Commit the insert after each player
        index += 1

    return cursor, conn
```

```python
# Create a list of my favorite NBA players:
players = [
            'Stephen Curry',
            'LeBron James',
            'Kyrie Irving',
            'Kevin Durant',
            'Damian Lillard'
```

```python
def insert(df, table: str):
    """ Given a pandas df and a tablename,
    we can create an insert statement for the
    table and return it.
    """
    df = df.astype(object)
    if not df.empty:
        columns = ', '.join(df.columns) # Separates the data by comma
        placeholders = ', '.join(['%s'] * len(df.columns)) # Creates tuple with '%s' for each column
        #row_list = df.iloc[0].tolist() # Gets all the data from a row in the df and turns it into a list
        data = [tuple(row) for row in df.to_numpy()] # List of row tuples ChatGPT

        insert = ( # create insert statement:
            f'insert into nba.{table} ({columns}) '
            f'values ({placeholders});'
        )
        return insert, data
    return None, None
```

Final Diagram

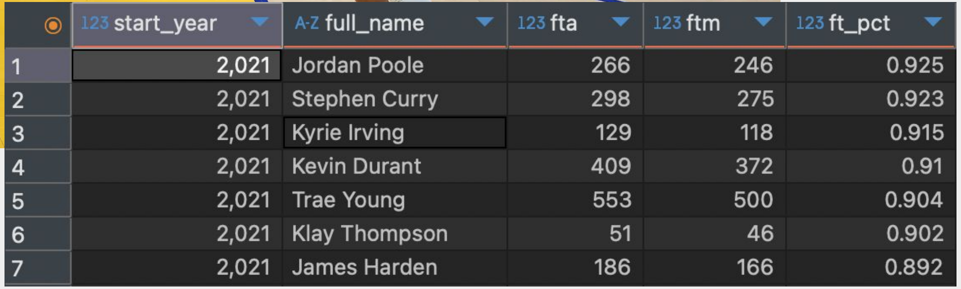
- 6 total tables
- 22,738 total rows extracted.
- Largest tables:
  - playergamelogs: 21,289 rows
  - awards: 1,016 rows
  - playersstats: 341 rows
  - Other tables ~30 rows each.
- Tables are connected via foreign keys:
  - player_id
  - season_id
  - team_id
- 30 NBA teams.
- 82 games per season.



Question 1)

- **What was the average free-throw percentage of each player in the database in 2021?**

```
select
    s.start_year,
    p.full_name,
    p.fta, p.ftm, p.ft_pct
from nba.seasons s
join nba.playersstats p
on s.season_id = p.season_id
where s.season_id = '2021-22'
order by p.ft_pct desc;
```

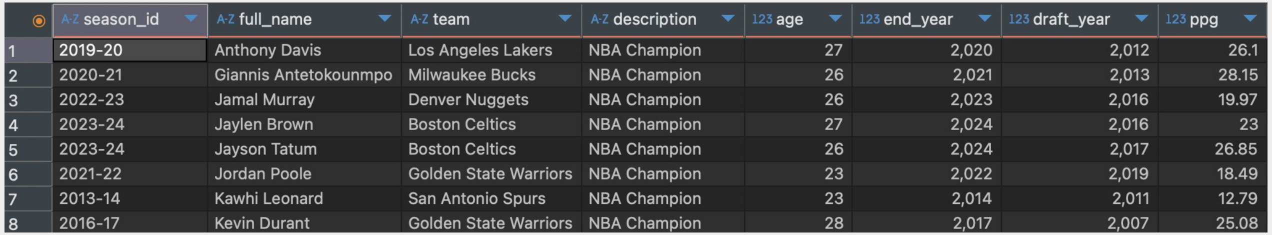


	start_year	full_name	fta	ftm	ft_pct
1	2,021	Jordan Poole	266	246	0.925
2	2,021	Stephen Curry	298	275	0.923
3	2,021	Kyrie Irving	129	118	0.915
4	2,021	Kevin Durant	409	372	0.91
5	2,021	Trae Young	553	500	0.904
6	2,021	Klay Thompson	51	46	0.902
7	2,021	James Harden	186	166	0.892

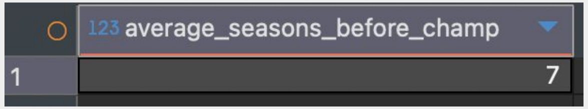
Question 2)

- **On average, how many seasons does it take for a player to win a championship?**

```
create view seasons_to_champ as
select distinct on (full_name)
        s.season_id, full_name,
        team, description,
        min(player_age) age,
        s.end_year, draft_year, ppg
from nba.awards a
join nba.playersstats p
on a.person_id = p.player_id and a.season = p.season_id
join nba.playerinfo p2
on p.player_id = p2.person_id
join nba.seasons s
on p.season_id = s.season_id
where description = 'NBA Champion'
group by full_name, s.season_id, full_name, team, description, end_year, draft_year, ppg;

select round(avg(end_year - draft_year), 0) as average_seasons_before_champ
from seasons_to_champ;
```



	season_id	full_name	team	description	age	end_year	draft_year	ppg
1	2019-20	Anthony Davis	Los Angeles Lakers	NBA Champion	27	2,020	2,012	26.1
2	2020-21	Giannis Antetokounmpo	Milwaukee Bucks	NBA Champion	26	2,021	2,013	28.15
3	2022-23	Jamal Murray	Denver Nuggets	NBA Champion	26	2,023	2,016	19.97
4	2023-24	Jaylen Brown	Boston Celtics	NBA Champion	27	2,024	2,016	23
5	2023-24	Jayson Tatum	Boston Celtics	NBA Champion	26	2,024	2,017	26.85
6	2021-22	Jordan Poole	Golden State Warriors	NBA Champion	23	2,022	2,019	18.49
7	2013-14	Kawhi Leonard	San Antonio Spurs	NBA Champion	23	2,014	2,011	12.79
8	2016-17	Kevin Durant	Golden State Warriors	NBA Champion	28	2,017	2,007	25.08

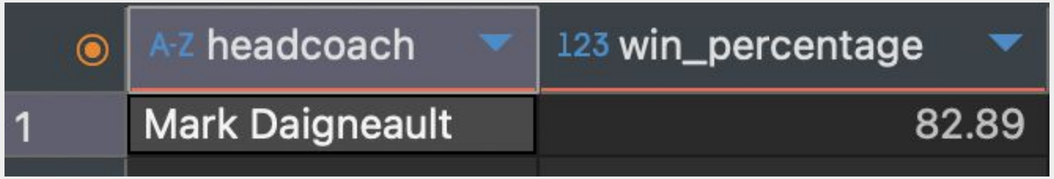


	average_seasons_before_champ
1	7

Question 3)

- **Which coach has the highest winning percentage in 2024-25?**

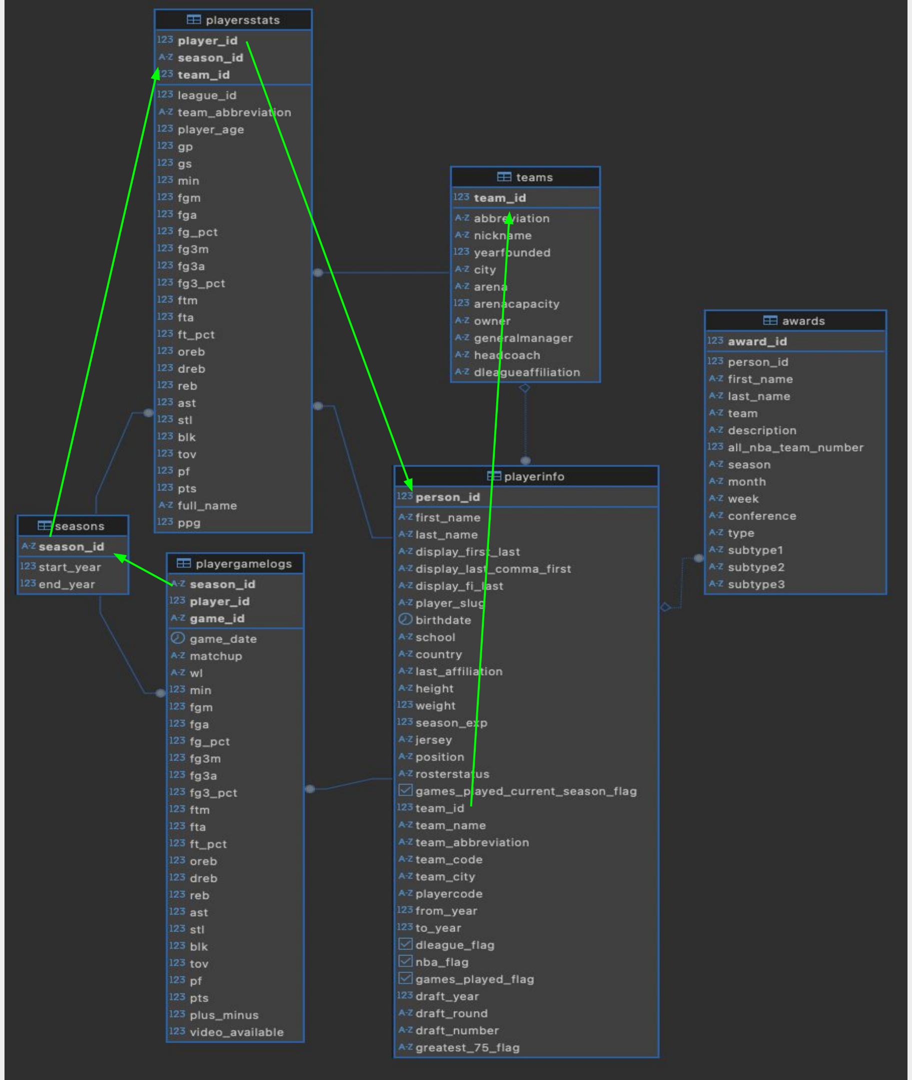
```
create view wins as
select
        headcoach,
        abbreviation,
        sum(case when wl = 'W' then 1 else 0 end) as wins,
        sum(case when wl = 'L' then 1 else 0 end) as losses,
        round(sum(case when wl = 'W' then 1 else 0 end) / count(*) * 100.0 , 2) as win_percentage
from nba.playergamelogs p
join nba.seasons s
on p.season_id = s.season_id
join nba.playersstats p2
on s.season_id = p2.season_id and p.player_id = p2.player_id
join nba.playerinfo p3
on p2.player_id = p3.person_id
join nba.teams t
on p3.team_id = t.team_id
where p.season_id = '2024-25'
group by headcoach, abbreviation
order by win_percentage desc
limit 1;

select headcoach, win_percentage from wins;
```



	headcoach	win_percentage
1	Mark Daigneault	82.89

Map of the Joins

```
from nba.playergamelogs p
join nba.seasons s
on p.season_id = s.season_id
join nba.playersstats p2
on s.season_id = p2.season_id
and p.player_id = p2.player_id
join nba.playerinfo p3
on p2.player_id = p3.person_id
join nba.teams t
on p3.team_id = t.team_id
```



What I Learned

- How to connect to a database in Python with psycopg2 using its documentation.
- Using the **cursor** and **connection** variables created from the database connection to perform SQL commands in Python.
- How to join multiple tables in my schema to answer fun NBA statistical questions that I wanted to know.

What I Found Interesting

This project was a lot of fun because I was able to investigate a topic that means a lot to me. I was able to go from **api -> python -> sql** which was a good experience instead of just doing csv extraction.

Resources

- Pandas Documentation
- nba.com for fact checking the nba_api
- psycopg2 for turning connection and turning Python to SQL.
- Lecture slides on indexes and views.
- My GitHub for all my work.
- Google images for all the pictures.