# \*\*NFL Player Profiles with Arrest records (ETL-Project)\*\*

## The purpose of this project is to take data from different sources otherwise known as extract, then transform the data by dealing with missing values, choosing data to include in the final table, renaming columns, and dropping duplicate data, and finally loading the transformed data into a relational database. The project also identified NFL player arrest records and their demographic player records pulled from player profile records. As a bonus, we are also asking: "Which college did the most players with arrest records during the period 2000 -2017 come from?"

## The Process:

## We analyzed all the available arrest records from Kaggle NFL Arrests 2000 - 2017. A record of reported NFL Arrests with details about Crime, Team and Player.

# Extract

# ![Alt Text](https://www.bing.com/th?id=OIP.Fw4E9H7ZRyhHzWK7xk6jUQHaFB&w=233&h=160&c=7&o=5&dpr=1.5&pid=1.7)

## Original Data Sources include

\* Web Scraping: 'html' from https://www.kaggle.com/patrickmurphy/nfl-arrests

\* pull: 'json' from https://www.kaggle.com/zynicide/nfl-football-player-stats#profiles\_1512362725.022629.json

\* Create dataframe from NFL Arrests csv file

\* Choose columns wanted from original dataframe

\* Rename columns

\* Create dataframe from .json file nfl profiles

## PLease see the code below for the process of extracting the data

###### `csv\_file = "NFL\_arrests.csv"`

###### `arrests\_data\_df = pd.read\_csv(csv\_file)`

###### `arrests\_data\_df.head()`

# Transform: (Data Cleanup & Analysis)

## ![Alt Text](https://www.bing.com/th?id=OIP.l8MinJm6s4scX7EmLp\_hvAAAAA&w=179&h=178&c=7&o=5&dpr=1.5&pid=1.7)

\* Clean nfl arrests by saving only the 1st instance of a player name and removing duplicate entries.

\* Clean nfl profiles. Duplicate players removed from the profile section

\* All na files filled with not applicable

### As part of the transformation of the data, the columns of each new created dataframe was evaluated and columns that were relevant to the project was selected from the dataframes. Next, some columns were renamed to give them more descriptive labeling. After manipulating columns, each dataframe was evaluated to find what type of data was included in the database, the shape of the data meaning the number of rows and columns in the database, missing data, and duplicate data.

### After evaluating all the aspects mentioned above, the nfl\_profiles\_df dataframe was found to have missing data in the "college","draft\_team", and "hometown" columns. The missing data in those columns would be filled with "not available", "not drafted", and "not available" respectively due to the desire to not eliminate any rows of data from the dataframe. Each dataframe was found to have duplicate data in the player column. The duplicates were dropped except for the first instance of the player name. The reason for this choice is due to the fact that we were addressing only one instance of an NFL player's arrests and not taking into account multiple arrests for the same individual.

### A sampling of the code used for transformation can be found below:

###### `nfl\_arrests\_df = arrests\_data\_df[['NAME', 'DATE', 'CASE', 'CATEGORY']].copy()`

###### `nfl\_arrests\_df = nfl\_arrests\_df.rename(columns={"DATE": "offense\_date", "NAME": "player", "CASE": "outcome", "CATEGORY": "charge"})`

###### `nfl\_arrests\_df.dtypes`

###### `nfl\_arrests\_df.shape`

###### `nfl\_arrests\_df.isna().sum()`

###### `nfl\_arrests\_df.drop\_duplicates(subset = 'player', keep = 'first', inplace = True)`

# Load the Data

# ![Alt Text](https://www.bing.com/th?id=OIP.EKlqoGs8WygAu7Nq5-gKFgHaHa&w=208&h=206&c=7&o=5&pid=1.7)

## After the data was cleaned, the next step in the ETL process is loading. For this project a relational database was chosen; specifically PostgreSQL. The reason for choosing a relational database is due to the fact that each table is related to players in the NFL. One table is named "nfl\_arrests" and the other "nfl\_profiles". Each table can be queried seperately; however as will be explaind later, the tables can be joined where the player's names match in both tables. Joining the tables in this way gives the most robust data than either of the tables do seperately.

## Steps in the loading process:

In this project SQL Alchemy was used to connect to PostgreSQL; so in essence SQL Alchemy is a ORM that will be connected to PostgreSQL through the use of python and Jupyter Notebooks.

1. Connect the orm to the local database with the code below:

###### `engine = create\_engine(f"postgresql://postgres:{secret.user\_pass}@localhost:5432/nfl\_etl")`

###### `connection = engine.connect()`

2. Add the 2 panda dataframes to the local database with the code below:

###### `nfl\_arrests\_df.to\_sql(name='nfl\_arrests', con=engine, if\_exists='append', index=True)`

###### `nfl\_profiles\_df.to\_sql(name='nfl\_profiles', con=engine, if\_exists='append', index=True)`

## After Dataframes Loaded into PostgreSQL

After the tables were loaded into PostgreSQL a query was performed using SQL Alchemy to combine the data of each table in a format that would give much more interesting data regarding the NFL players in the dataframe. The code for running the query is below:

###### `query = "select nfl\_arrests.player, nfl\_arrests.offense\_date, nfl\_arrests.outcome, nfl\_arrests.charge, nfl\_profiles.hometown,nfl\_profiles.college, nfl\_profiles.draft\_team From nfl\_arrests Inner Join nfl\_profiles ON nfl\_arrests.player = nfl\_profiles.player"`

## The queried data was brought into a pandas dataframe using the code below:

###### `combined\_nfl\_stats = pd.read\_sql(query, connection)`

\*Please note that you must use your personal password for SQL Alchemy as my password has been redacted for privacy\*