ETL PROJECT REPORT

By Hakob Antonyan, Vito Perez, Gerard Tieng

PROJECT SUMMARY

The mission of our project was to combine as many data sources as we could find online to create a thorough and comprehensive database of information about every NBA player that was active during the 2018-2019 season.

EXTRACT

A total of five different sources were used to create our database:

#	Summary	Format	Source	URL
1	2018-19 player performance statistics	Excel	nbastuffer.com	https://www.nbastuffer.com/2018-2019-nba-player-stats/
2	Biographical and performance statistical data for NBA players born 1902 to 1999	CSV	data.world; @datadavis	https://data.world/datadavis/nba-salaries/workspace/file?filename=players.csv
3	Yearly salary for players in dataset #2	CSV	data.world; @datadavis	https://data.world/datadavis/nba-salaries/workspace/file?filename=salaries_1985to2018.csv
4	Yes/No tattoo status of NBA basketball players, presumably from the 2014-15 season	CSV	data.world; @fivethirtyeight	https://data.world/fivethirtyeight/nba-tattoos/workspace/file?filename=nba-tattoos-data.csv
5	List of NBA basketball players from the 2018-19 season, categorized by their uniform numbers	HTML	basketball-reference.com	https://www.basketball-reference.com/leagues/NBA_2019_numbers.html

TRANSFORM

1. **NBA STUFFER (PLAYER STATS)**
    a. **pd.drop** — eliminate the following non-essential columns:
        i. Offensive Rating
        ii. Defensive Rating
        iii. Usage Rate
        iv. Effective Shooting Percentage
        v. True Shooting Percentage
        vi. Versatility Index
    b. **pd.rename** — renaming columns with long descriptions to simplified abbreviations
    c. **pd.sort_values** — sort dataset by Name, Ascending
    d. **pd.reset_index** — reset index of dataset to sorted names

2. **DATA WORLD (BIO INFO)**
    a. **pd.drop** — drop statistical columns to isolate biographical information.
    b. **pd.to_datetime** — convert CSV date objects to datetime for selection
    c. **data subset** — eliminate most non-active players from dataset by filtering players older than Vince Carter, the oldest player of the 2018 NBA season
    d. **reorder columns** — column reordering to most relevant bio information first
    e. **pd.sort_values** — sort dataset by Name, Ascending
    f. **pd.reset_index** — reset index of dataset to sorted names

3. **DATA WORLD (SALARY INFO)**
    a. **pd.drop** — eliminate non-essental columns
    b. **data subset** — subsetting data to most current salary data
    c. **pd.sort_values** — sort dataset by id, Ascending
    d. **pd.reset_index** — reset index of dataset to sorted id

4. **DATA WORLD (PLAYER TATTOOS)**
    a. **pd.rename** — renaming column names to align with keys

5. **BASKETBALL-REFERENCE (UNIFORM NUMBERS)**
    a. BeautifulSoup — request and parse HTML code of website
    b. soup.find_all — isolate table class holding uniform number and player name
    c. soup.find_all loop & list.append — use loop to separate and categorize uniform numbers, player name, and team names within tables into 3 lists
    d. define dictionary — assemble lists of names, numbers, and teams into one dictionary
    e. pd.DataFrame.from_dict — transform dictionary to Pandas dataframe
    f. **pd.sort_values** — sort dataset by Name, Ascending
    g. **pd.reset_index** — reset index of dataset to sorted names

6. **JOINING THE FIVE DATASETS**
    a. BIO INFO <— SALARIES
        i. Join LEFT on PLAYER_ID key
    b. BIO/SALARIES <— STATISTICS
        i. Join LEFT on PLAYER NAME key
    c. BIO/SALARIES/STATISTICS <— UNIFORM NUMBERS
        i. Join LEFT on the PLAYER NAME key
    d. BIO/SALARIES/STATISTICS/UNIFORM NUMBERS <— TATTOOS
        i. Join LEFT on the PLAYER NAMES
    e. SET INDEX
        i. Use pd.set_index() to assign PLAYER_ID derived from BIO INFO table as the index for the final table

LOAD

MongoDB Vs PostgreSQL.

Since we have tables in CSV files and it is structured, PostgreSQL would be a better choice for our project. We are using SQLAlchemy to convert the Pandas dataframe.

1. **PANDAS TO SQLITE**
    a. **from sqlalchemy import create_engine, inspect**
        i. Create connection to sqlite flavor to produce a flat file of the pandas dataframe.
        ii. Inspect engine connection to ensure row headers were imported properly.
    b. **finaldata.to_sql('nba2018', con=engine)**
        i. Pandas command to convert pandas table to sqlite file

2. **PANDAS TO POSTGRES**
    a. **create_engine('postgresql+psycopg2://postgres:password@localhost:5432/ETL-NBA-Players')**
        i. Use Postgres flavor and psycopg2 driver to connect to Postgres database labelled ‘ETL-NBA-Players’
    b. **finaldata.to_sql(name='nba_players_18_19', con=conn)**
        i. Pandas command to convert pandas dataframe to postgres database table

POST-MORTEM

Final record count: 643.

Challenges: Mid-season trades prove to bloat the data beyond the 550 active players, with some players appearing with one or more records and differing statistics but duplicate biographical data.

For the next interation of
Separating the master datasets in SQL as:

- PLAYER STATISTICS + UNIFORM NUMBER
- BIO + TATTOO
- SALARIES

ended up having two primary keys because they were traded during midseason. This caused the player to be in two different teams having two different uniform numbers.

When joining the tables, in the final table NaN values were created. Most of the players' information in bio and statistics stayed the same.

Extracting the Data
Find the reliable data from different sources and making sure that the players name match.

Transforming the Data
Make sure did not create any duplicates.

Load the Data
Convert pandas dataframe to sqlite file to create a compact file for easy transferability.
Convert pandas dataframe to postgres database for storage and simple reading.