Data Analytics Bootcamp

**ETL Project**

Mens and Womens Shoe Database Creation

horizontal line

# 

# Introduction

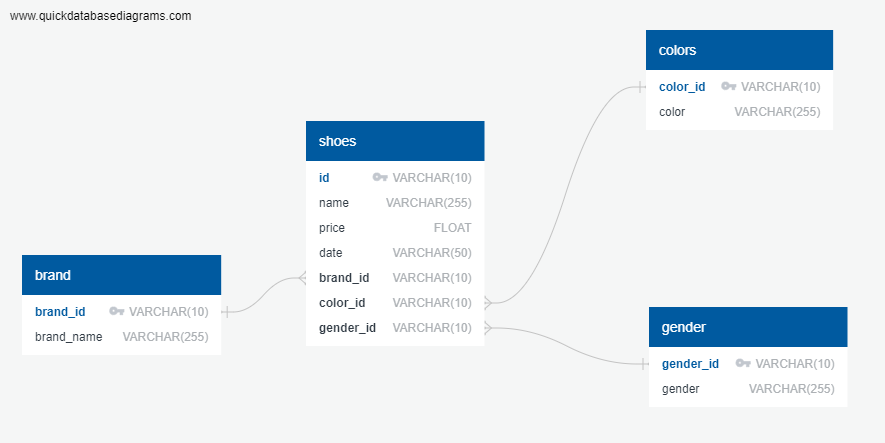
Our team was tasked to Extract, Transform and Load a number of disparate datasets into a single database. As part of the process, we needed to clean the data and get it ready for use in a production database, and be able to query the data.

We found two datasets on Kaggle.com, relating to the sale of mens and womens shoes over a period of time. We thought this would be appropriate data for our project because

* The two datasets could be related to each other (i.e. mens and womens shoes have similar properties - sizes, prices, colours, brands etc)
* There was a lot of data in the datasets - the Mens dataset had over 19,000 rows, and the Women's dataset had approximately 10,000 rows
* The data looked to be relatively dirty and required substantial cleaning prior to loading into the database
* Some of the data in the datasets looked to be redundant, and we thought it would be good practice for us to transform the redundant data into relational data for SQL.

# Design

Our Team went through the available datasets and decided on a resulting database schema that looked as follows:



# Data Cleaning

We downloaded CSV files from Kaggle.com and set to work. We initially had three datasets - 1x Mens Shoes dataset, and 2x Womens Shoes dataset. The first Womens Shoes dataset had a high level of redundant data, once duplicates had been removed only 653 of 10,000 records remained, so we switched to the other Womens dataset.

We loaded the CSV files into Pandas, and our team figured out what cleaning tasks were required, and drafted a basic schema for our database.

Our Data Cleaning tasks were divided between the three of us, and were as follows:

* Search the Name column for items that contain shoe-like entries, and drop rows that do not contain a shoe-like word in their name (e.g. boot, sandal, heel, etc). This removed 6504 rows from the Mens Shoes dataset, and 1921 rows from the Women’s Shoes dataset.
* Remove all rows that had a price not listed in US Dollars. This removed 751 rows from the Men’s Shoes dataset.
* Filter out shoes that are on Sale (where the isSale column = False). At the 11th hour we found a bug in our code that meant this cleaning aspect did not work correctly, however we do not think it substantially affected the quality of our final dataset, as we chose the highest price of a given item anyway.
* Sort or Group the shoes by their Shoe ID, and filter out duplicate entries. To make the decision on which shoes to filter out, we chose to get the MAX price for each Shoe ID. This removed 1179 rows from the Men’s dataset.
* Delimit the Colour column, and keep the first entry only. Many shoes had colour entries like “Brown,Tan,Gold,White,Pearl” for example. We decided it was too complex to retain all this information, and just chose to keep a single colour. Rows without a colour entry were dropped. This removed 2261 rows from the Men’s dataset.
* Date Read-in. The “Date” columns had an unusual syntax, for example “2017-01-09T20:25:29Z”. Some coding was required to clean the date strings up, to make them readable by DateTime. Dates that could not be adequately read were removed. This removed 2153 rows from the Women’s Shoes dataset.

Overall, our datasets were reduced substantially. We were all surprised by the amount of data that got dropped.

* The Men’s Shoes Dataset was reduced from 19,315 rows, down to 3015 rows, an 85% reduction.
* The Women’s Shoes Dataset was reduced from 10,000 rows, down to 5698 rows, a 43% reduction.

One of the outcomes of our Data Cleaning was a decision that we were searching for the highest RRP price of a listed shoe out of the dataset, if there were multiple entries for the same item. Another decision from our cleaning process was to reduce the complexity regarding Shoe Colours, and have only one colour assigned to a particular Shoe.

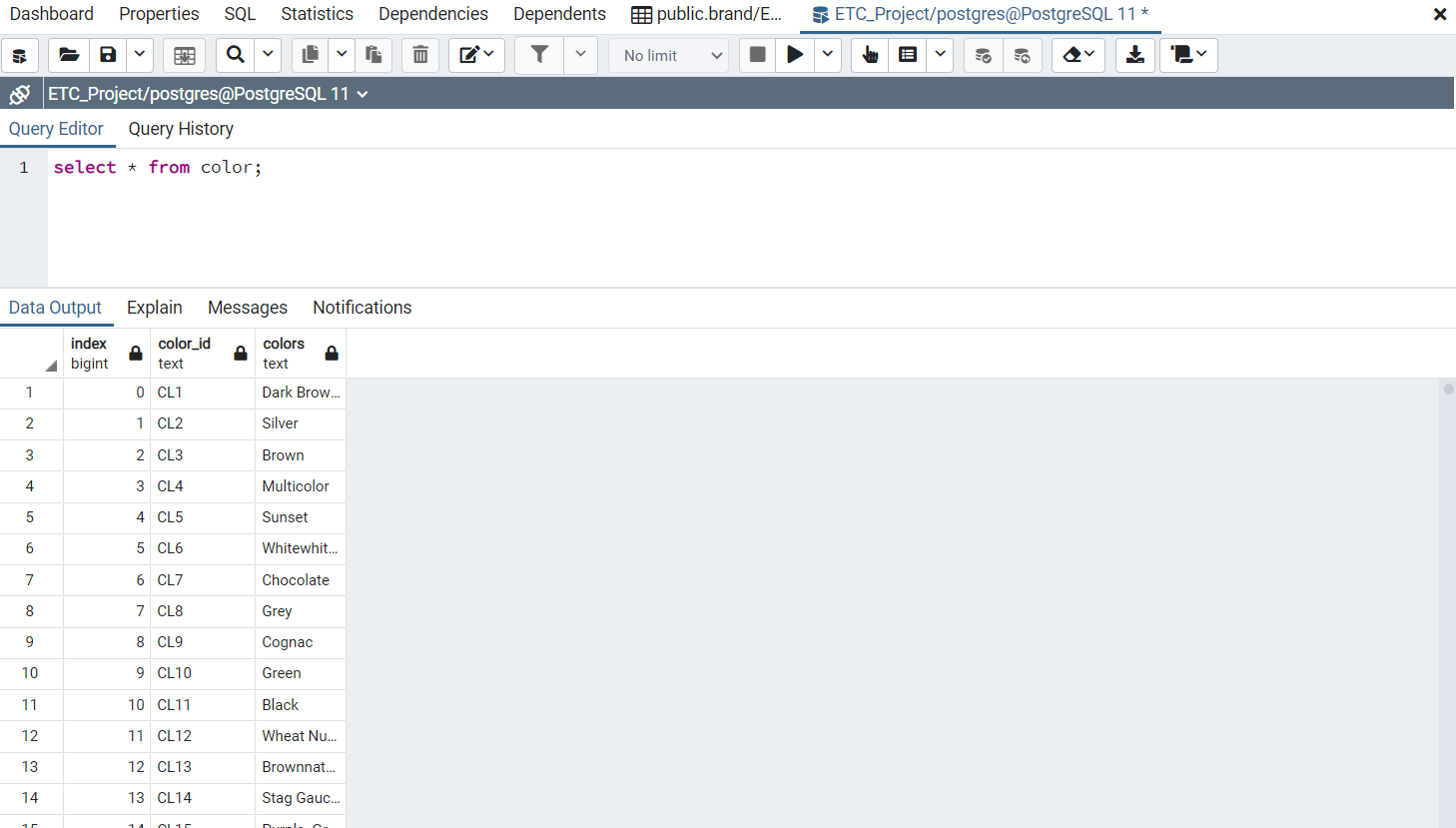
# Final Clean & Load into PostgreSQL

Prior to loading into Postgres, we ran a final cleanup. This did some tasks such as:

* Setting Title Case on all the Brands and Colours. This reduced the overall brand and colour count by about 10%.
* Setting Lower Case on the column names and table names
* Renaming the price column for easier referencing in queries

We used SQLAlchemy to create a connection to the PostgreSQL database, and uploaded our data into the database. There were possibly some bugs in our upload code, as the resulting tables contained an index number column which may not have been intended.

### Color Table



### 

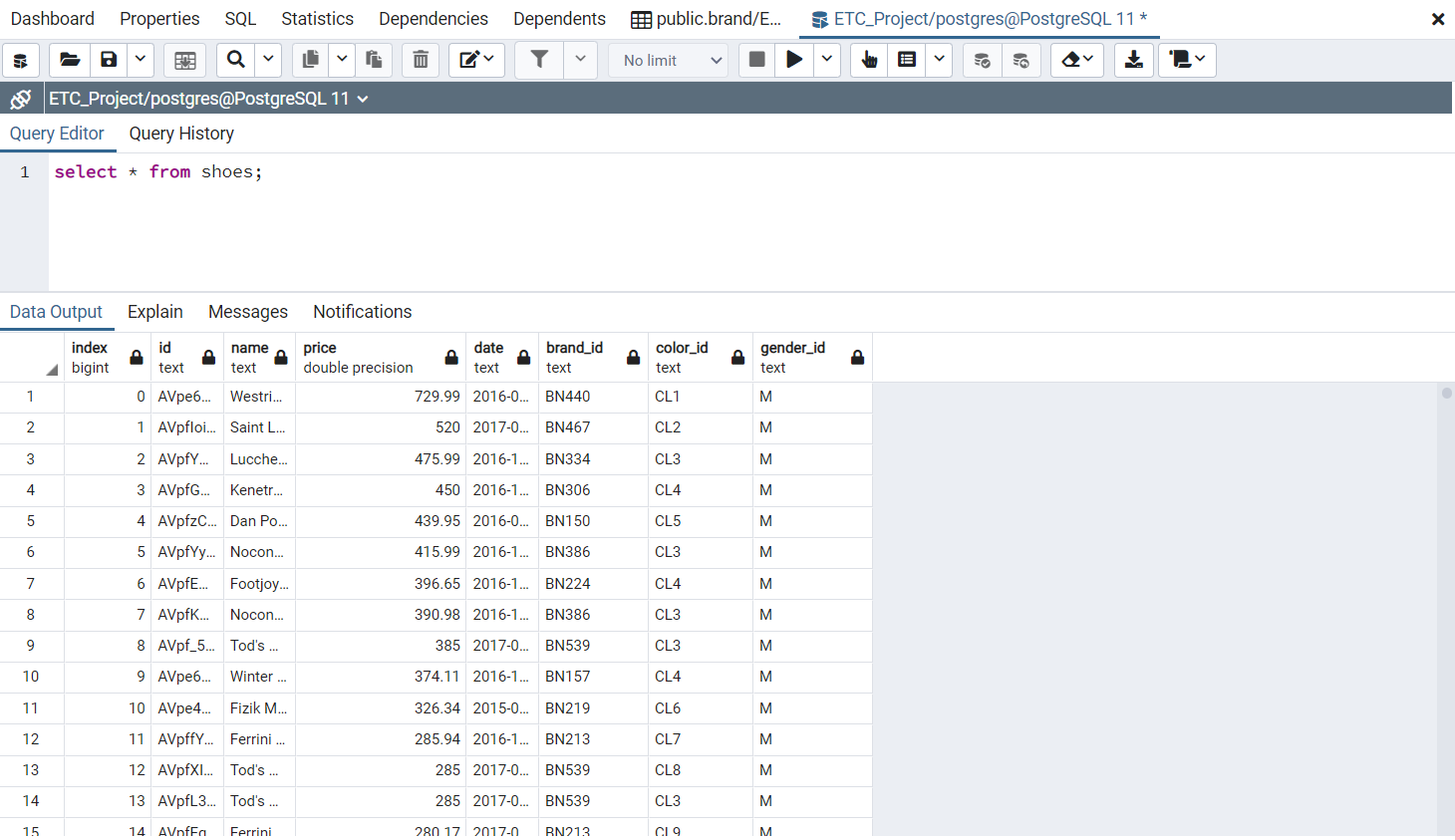
### 

### 

### 

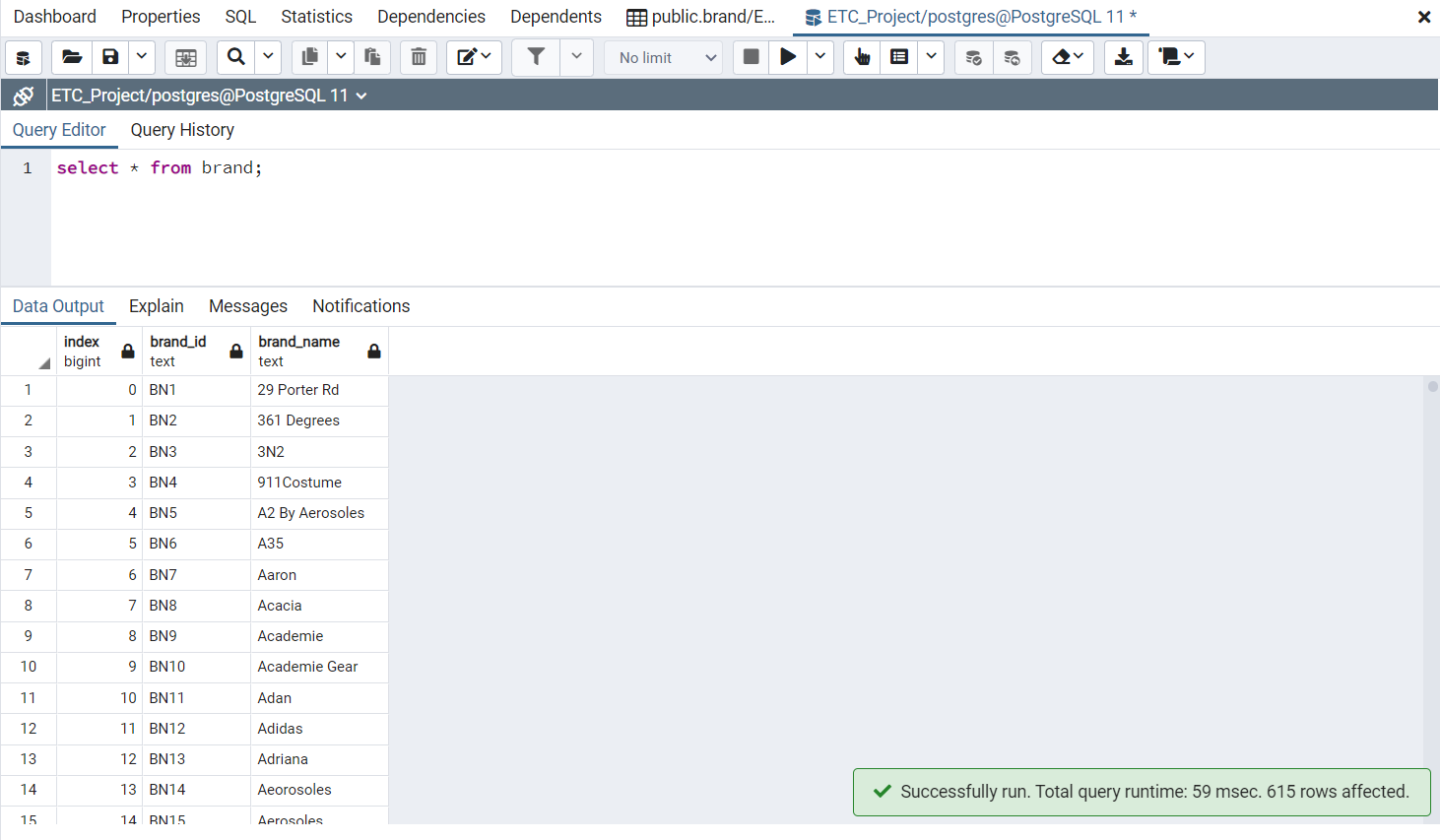
### 

### Shoes Table



### Gender table:

### Brand table



# 

# 

# 

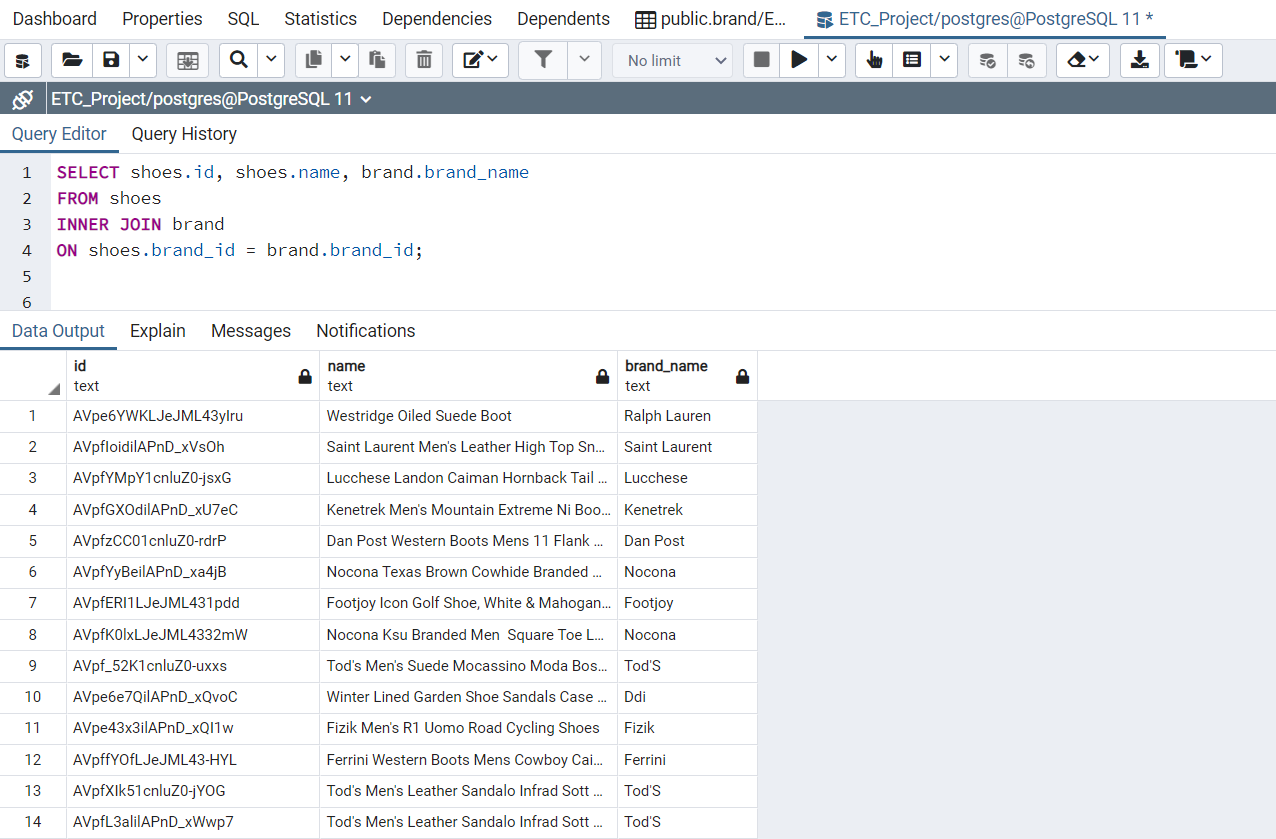
# 

# 

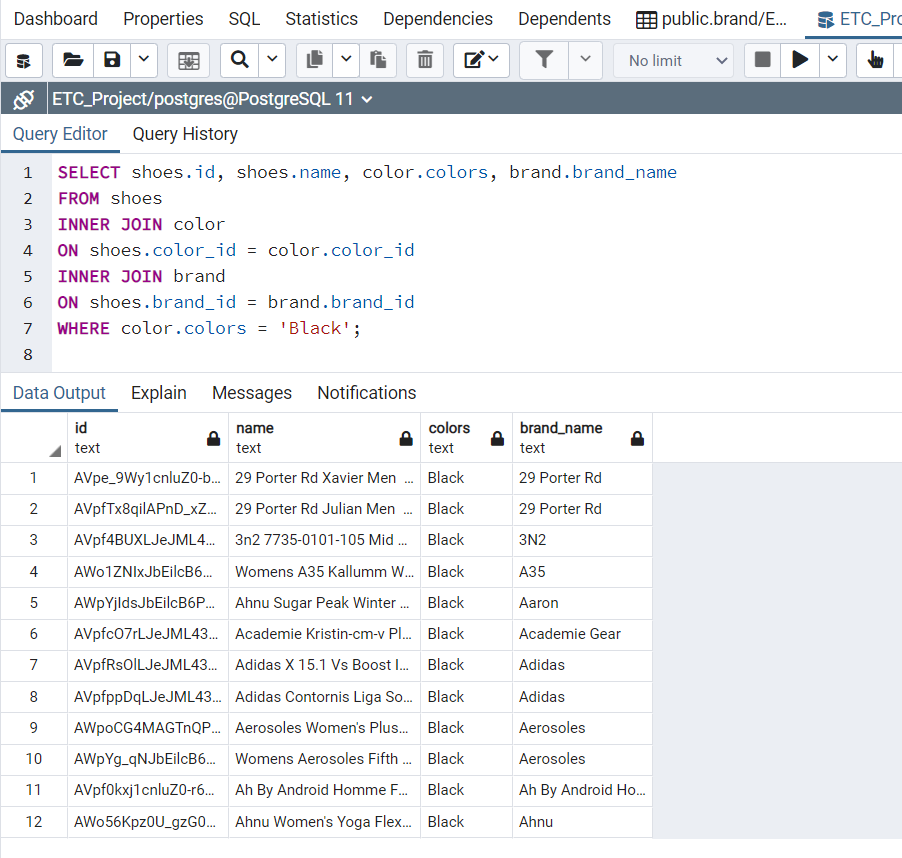
# Queries and Analysis

We ran a few queries on the database to check it, as follows.

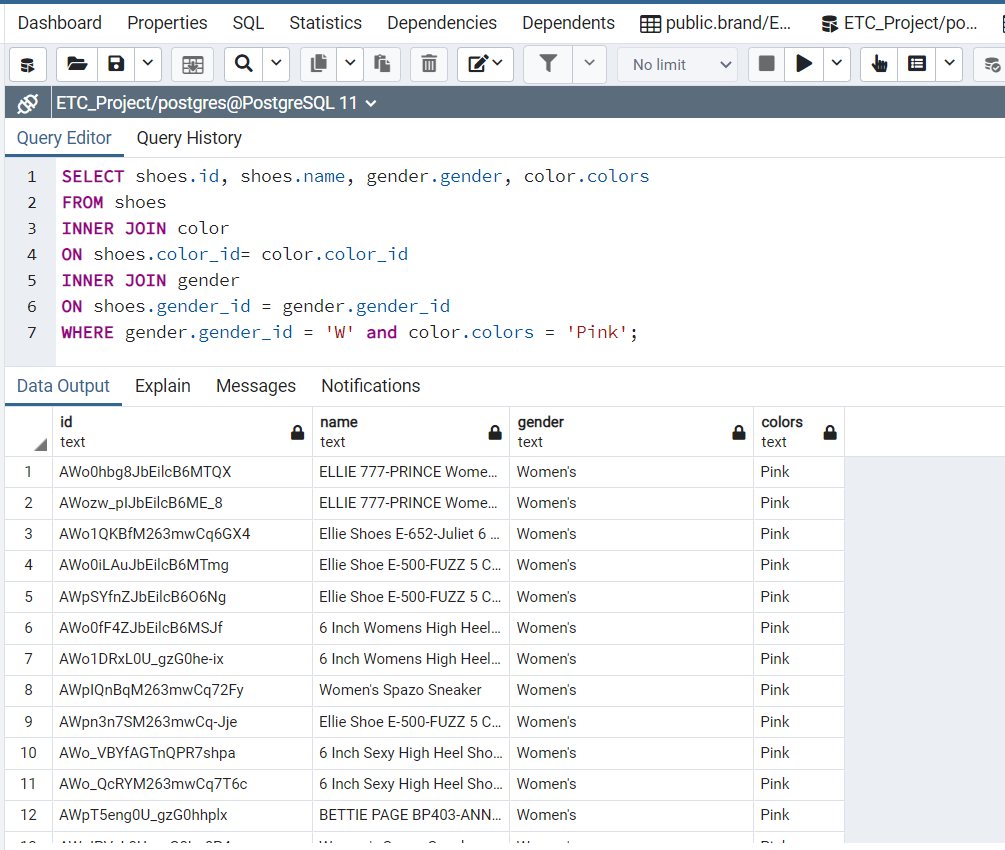
### Join shoes and brand table and display all the shoes id, name and their brand name



### Join shoe, brand and color table, display shoe id, name, color and brand\_name for all shoes with ‘Black’ color.



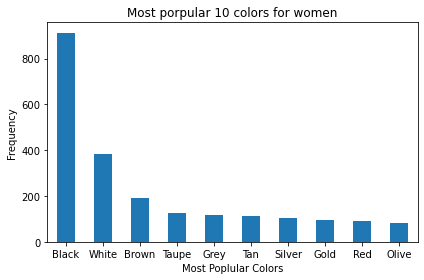
### Display only Women’s shoes and color ‘Pink’.

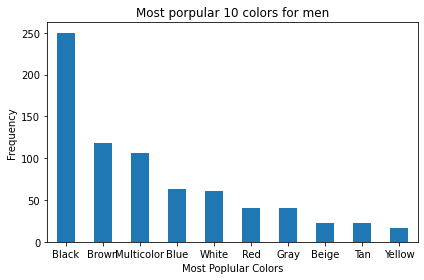


We have also imported the database from SQL back to pandas and completed some analysis.

### Most popular shoe colors for men and women in bar chart

### (SQL\_Plot.ipynb)





# Project documentary

### ETLProject\_Combined\_All.ipynb

Run this file to:

* Import from csv to python
* Clean up men’s and women’s shoe data separately, and save to separate csv file
* Create 3 reference tables: color, gender, brand
* Combine the 2 dataset
* Store in PostgressSQL (There is no need to run a schema before importing)

### SQL\_Plot.ipynb

Run this file to:

* Export tables from PostgresSQL and save as pandas dataframe
* Analyse data using pandas
* Plot bar charts

# REFERENCES

Women’s Shoe Prices

<https://www.kaggle.com/datafiniti/womens-shoes-prices>

Men’s Shoe Prices

<https://www.kaggle.com/datafiniti/mens-shoe-prices>