ETL REPORT:

**Extract**: *The sources of data that you will extract from.***Transform:** *The type of transformation needed for this data (cleaning, joining, filtering, aggregating, etc)***Load**: *The type of final production database to load the data into (relational or non-relational).***Analysis:** The final tables or collections that will be used in the production database.

**Background**:

We used Seattle Airbnb data we found through Kaggle and on the Airbnb website. The data was in CSV form originally and we used all of the 2019 data that was available, which consisted of 8 CSVs (January – August). The CSV's included data for each listing – neighborhood, price, review information, room type, minimum nights, etc. for each month. We also included a CSV with availability data for 2019 to show dates that each listing was available or filled.

**ETL Process:**

**EXTRACT**

1. This process started with downloading each CSV (8) by creating a path in Python using Jupyter Notebook.

**TRANSFORM**

1. Using pandas, we created a dataframe for each of the CSV files.
2. Then for each dataframe we dropped the unnecessary columns to get the dataset we wanted.
3. Next we renamed the columns to make the join of the dataframes easier.
4. Then we joined each dataframe sequentially and dropped any repetitive columns.
5. We then re-arranged the columns of the merged dataframe that contained all of the data for the 8 CSVs.
6. The final dataframe was then saved and exported to a CSV.

**LOAD**

1. We decided to store the data into a SQL database using pgAdmin 4 and created a table with the final 2019 CSV file.
2. We built a table with the listing id as the primary key. We did have to change the names of some of the columns because they contained a “.” and created an error when creating the table.
3. We then created another table in SQL with the 2019 availability listing CSV.
4. With the two tables, we were able to query and join on listing id to pull data based on availability for each listing in 2019.
5. By using a relational database, this allows us to more easily query and extract data that we need.

**ANALYSIS**

**Price vs review-count:**

We wanted to investigate effects of reviews-count on price of locations. We realized that reviews per month in each dataframe contained updated data of the year so we used the last months reviews data for the analysis. Mean price of the year was determine for each Airbnb id and saved in a separate column (mean price). We used scatter-plot to display relationship between reviews-per-month and Airbnb price. The plot revealed that airbnb price decreases as reviews increases which seem unusual. One possible reason for the inverse relationship could be properties receiving high number of negative reviews. High number of negative reviews may lower the price. However, it is unclear at the point because we do not know the kind of reviews (-ve or +ve) all the properties are receiving.

**Price vs location:**

We also wanted to figure out the most expensive locations in Seattle. Therefore, we grouped the data by location(neighbourhood group) and arranged the data in descending order of mean price. We used the top ten data based on price and used bar-graph to show top ten location based on price. The price ranged from $234 -$131. Seattle downtown was the most expensive location of them all.

**Price vs room type:**

We wanted to examine how price changes with room-type. We grouped the data by room-type and used bar-graph to show the price by room-type. As expected, entire-home/apt are expensive than the other room types. Entire-home costs about 2 times as much as private-room and 3 times as much as shared-room.

**Statistical analysis of price and location:**

We wanted to determine if the prices of the top ten expensive locations were statistically different. So we used one-way anova on the prices of the locations which showed that there was a significant difference in prices between expensive locations with p-value 4.36-39.

|  |
| --- |
| **SQL Queries**  CREATE TABLE airbnb\_seattle1 ( |
| id INT PRIMARY KEY, | |
| name VARCHAR, | |
| neighbourhood\_group VARCHAR, | |
| neighbourhood VARCHAR, | |
| room\_type VARCHAR, | |
| p1 NUMERIC, | |
| p2 NUMERIC, | |
| p3 NUMERIC, | |
| p4 NUMERIC, | |
| p5 NUMERIC, | |
| p6 NUMERIC, | |
| p7 NUMERIC, | |
| p8 NUMERIC, | |
| min\_nights1 NUMERIC, | |
| min\_nights2 NUMERIC, | |
| min\_nights3 NUMERIC, | |
| min\_nights4 NUMERIC, | |
| min\_nights5 NUMERIC, | |
| min\_nights6 NUMERIC, | |
| min\_nights7 NUMERIC, | |
| min\_nights8 NUMERIC, | |
| reviews\_no\_1 NUMERIC, | |
| reviews\_no\_2 NUMERIC, | |
| reviews\_no\_3 NUMERIC, | |
| reviews\_no\_4 NUMERIC, | |
| reviews\_no\_5 NUMERIC, | |
| reviews\_no\_6 NUMERIC, | |
| reviews\_no\_7 NUMERIC, | |
| reviews\_no\_8 NUMERIC, | |
| reviews\_month1 NUMERIC, | |
| reviews\_month2 NUMERIC, | |
| reviews\_month3 NUMERIC, | |
| reviews\_month4 NUMERIC, | |
| reviews\_month5 NUMERIC, | |
| reviews\_month6 NUMERIC, | |
| reviews\_month7 NUMERIC, | |
| reviews\_month8 NUMERIC, | |
| listings\_count1 NUMERIC, | |
| listings\_count2 NUMERIC, | |
| listings\_count3 NUMERIC, | |
| listings\_count4 NUMERIC, | |
| listings\_count5 NUMERIC, | |
| listings\_count6 NUMERIC, | |
| listings\_count7 NUMERIC, | |
| listings\_count8 NUMERIC) | |
|  | |
|  | |
|  | |
| CREATE TABLE airbnb\_calendar1 ( | |
| listing\_id INT, | |
| date VARCHAR, | |
| available VARCHAR, | |
| price NUMERIC, | |
| minimum\_nights NUMERIC, | |
| maximum\_nights NUMERIC); | |
|  | |
| **Analysis Queries:** | |
| With both airbnb tables saved into SQL, the user can join tables on listing ID. | |
|  | |
| 1.**Neighborhood has the most listings (Broadway)** | |
| SELECT count(neighbourhood), neighbourhood | |
| FROM airbnb\_seattle1 | |
| GROUP BY neighbourhood | |
| ORDER BY count(neighbourhood) DESC | |
|  | |
| 2**. Which neighborhood has the most availabiilty in 2019 (Wallingford)/ least availabiilty in 2019 (Highland Park)** | |
|  | |
| SELECT | |
| c.available, | |
| count(c.available), | |
| s.neighbourhood | |
| -- AVG(p1) | |
| FROM airbnb\_calendar c | |
| JOIN airbnb\_seattle1 s | |
| ON c.listing\_id = s.id | |
| WHERE available = 't' | |
| GROUP BY s.neighbourhood, c.available | |
| ORDER BY count(c.available) DESC | |
|  | |
| **3. Average price per night/neighborhood.** | |
| SELECT neighbourhood, AVG(price) | |
| FROM airbnb\_seattle1 s | |
| JOIN airbnb\_calendar1 c | |
| ON c.listing\_id = s.id | |
| GROUP BY neighbourhood | |
| ORDER BY AVG(price) DESC | |