ETL Report

Group 3 - Bionic Blobs

Group Members: Miko, Alex, Shirley
ETL Process Date: 5/11/2023-5/26/2023

Introduction

In our data analysis we are focusing on the rise in AirBnB popularity in New York City by examining the changes in prices, number, bookings, and reviews of AirBnBs in New York City with some comparison relative to hotels and rental homes to see the changes over the years and possible trends between one another. Also looking into the features that make some AirBnBs more appealing to book or more expensive than others using machine learning algorithms. We take on some secondary datasets such as rental home pricings and hotel pricings for comparisons. And also incorporate a crime database and tourist locations as well as holiday lists to see some external features impacting the fluctuations of AirBnB price and popularity in New York City.

Questions we aim to answer within our research include but is not limited to:

General Questions:

1. Most expensive and least expensive areas of airbnb in new york?
2. When is the most popular time of the year to book airbnb in New York? (seasonal, bring in holidays)
3. How long is the average stay?
4. Count of Airbnbs over time
5. Count of properties per host

Machine Learning Questions:

1. Is there a correlation between neighborhood and price? Can we use this to predict airbnb prices based on location? regression
2. What amenities affect price most? regression
3. What factors will affect the review ratings of Airbnb? Can we use them to predict the ratings of Airbnb?

Business Questions:

1. Trends of average price of rental homes in new york from zillow data vs trend of average price of airbnb in new york to see any trends by month, year
2. Are hotels more expensive on average compared to AirBnBs?
3. Does the amount of crime in each borough have any impact on the amount of airbnb bookings? How about tourist attractions?
4. Does Airplane fare prices affect AirBnB bookings and when is that?

<u>Why we need to transform our data:</u>

We need to transform our data so that our data sets can be merged upon neighborhood groups and date so that we can aggregate our data by date, month, year and neighborhood group for each unique airbnb, rental home, hotel listing, etc to get approximations on price and count for time series or bar plot data to compare to one another. We also have to make sure overly aggregated data is denormalized so we can aggregate it the way we would like to to make sure all datasets are grouped in similar ways to compare to one another. We need to break out some column labels for features of our dataset such as amenities, room type, reviews, host details in order to record them properly for visualizations. There is also the option to union/append tables with locations to be able to display multiple categories of geographical data for listings, crime occurrences, hotel, and tourist attraction locations.. Lastly, we need to get rid of unnecessary columns and rename data types to match one another for comparisons across different datasets.

Matching Naming Conventions :

Column	DataType	Value convention	Example
Neighborhood Group/ Borough	str	proper	Queens, Manhattan, Staten Island, Bronx, Brooklyn
Date	Date	YYYY-MM-DD	2023-05-10

Data Sources

1. *Get the Data*. Inside Airbnb. (n.d.). http://insideairbnb.com/get-the-data
2. Dgomonov. (2019, August 12). *New York City airbnb open data*. Kaggle. https://www.kaggle.com/datasets/dgomonov/new-york-city-airbnb-open-data
3. Diaz-Bérrio, G. (2017, November 16). *New York Hotels*. Kaggle. https://www.kaggle.com/datasets/gdberrio/new-york-hotels
4. (DOF), D. of F. (2021, June 18). *Hotels Properties Citywide: NYC open data*. Hotels Properties Citywide | NYC Open Data. https://data.cityofnewyork.us/City-Government/Hotels-Properties-Citywide/tjus-cn27
5. Samoshyn, A. (2020, July 12). *New York City police crime data historic*. Kaggle. https://www.kaggle.com/datasets/mrmorj/new-york-city-police-crime-data-historic
6. *Housing Data*. Zillow. (2023, April 25). https://www.zillow.com/research/data/
7. Wong, D. (2022, October 18). *Flight prices*. Kaggle. https://www.kaggle.com/datasets/dilwong/flightprices?resource=download
8. Jain, N. (n.d.). *Top 50 NYC attractions you can't miss*. Attractions of America. https://www.attractionsofamerica.com/attractions/new-york-city-top-10-attractions.php

Other Cited Sources, but not datasets:

Wikimedia Foundation. (2023, April 3). *Timeline of Airbnb*. Wikipedia. https://en.wikipedia.org/wiki/Timeline_of_Airbnb

Herries, J. (n.d.). *New York population density*. ArcGIS Hub. https://hub.arcgis.com/maps/80f9b95a4ce0491091f1477710f6a91d

2023 Major Daily Holidays by Month. Holiday Insights. (2023, January 1). https://www.holidayinsights.com/everyday.htm

http://www.citypopulation.de/en/usa/newyorkcity/

Extraction

Data	Data Source	Obtained How	Data Format	Extraction Steps
1	Inside Airbnb	Download from insideairbnb.com	CSV	1. Download the CSV ● Import the data into Pandas/Pyspark/Power Query for Exploratory Data Analysis (EDA) ● Load the data into Azure Storage Explorer Data Lake/Blob
2	New York City airbnb open data	Download from Kaggle	CSV	1. Download the CSV ● Import the data into Pandas/Pyspark/Power Query for EDA ● Load the data into Azure Storage Explorer Data Lake/Blob
3	New York Hotels	Download from Kaggle	CSV	1. Download the CSV ● Import the data into Pandas/Pyspark/Power Query for EDA ● Load the data into Azure Storage Explorer Data Lake/Blob
4	Hotels Properties Citywide	Download from NYC Open Data Portal	CSV	1. Download the CSV ● Import the data into Pandas/Pyspark/Power Query for EDA ● Load the data into Azure Storage Explorer Data Lake/Blob
5	New York City police	Download from Kaggle	CSV	1. Download the CSV ● Import the data into Pandas/Pyspark/Power Query for

	crime data historic			EDA ● Load the data into Azure Storage Explorer Data Lake/Blob
6	Housing Data. Zillow.	Open source data downloaded from Zillow website for research/data	CSV	1. Download the CSV ● Import the data into Pandas/Pyspark/Power Query for EDA ● Load the data into Azure Storage Explorer Data Lake/Blob
7	Flight prices	Download from Kaggle	CSV	1. Download the CSV ● Import the data into Pandas/Pyspark/Power Query for EDA ● Load the data into Azure Storage Explorer Data Lake/Blob
8	Top 50 NYC attractions you can't miss	Web Scrape from website, and use forward geocoding API to obtain latitude/longitude	Website HTML	1. Write python code to send http request to get the html from the website 2. Parse the html using a web scraper parser like beautiful soup 3. Find specific html tags/classes for each section of data you want to extract 4. Loop through the html and store the data into a dictionary 5. Create dataframe from dictionary and load into kafka producer, csv to go to Data Lake or to sql or all 3

Transformation

<u>AirBnB Data:</u>

1. CSVs downloaded from Inside AirBnB website.
2. CSV uploaded to Azure Data Factory, in a data blob.
3. Transformed with Databricks
    a. Established mount point.
    b. Extracted all .csv files from location, and initialized variables for each with the naming convention, "dataframes['table_name]."
    c. Eliminated unnecessary columns.
    d. Aggregated values for dataframes['calendar'] to show total days available in the span of a year instead of a value for each date of each listing.
4. Prepared 'Amenitites' Table for machine learning:

a. Removed all columns except ‘amenities’ and ‘price.’ The column ‘amenitites’ had a list in brackets of all amenities for each listing.

amenities	price
["50\" TV", "Bathtub", "Microwave", "Free driveway parking on premises", "Laundromat nearby", "Kitchen", "Refrige...	143
["Free parking on premises", "Carbon monoxide alarm", "First aid kit", "Wifi", "TV", "Smoke alarm", "Kitchen"]	30
["BBQ grill", "Security cameras on property", "Fire pit"]	157
["Security cameras on property", "Keypad", "Carbon monoxide alarm", "Wifi", "Self check-in", "Lock on bedroom doo...	89
["Hangers", "Clothing storage: closet", "Security cameras on property", "Bathtub", "Fast wifi \u2013 363 Mbps", "Mic...	125
["Hangers", "Bathtub", "Microwave", "Hot water", "Dishes and silverware", "Kitchen", "Refrigerator", "Hair dryer", "L...	63
["Air conditioning", "Free parking on premises", "Pets allowed", "Wifi", "Lock on bedroom door", "Fire pit", "Kitchen"]	82
["Hangers", "Kitchen", "Shampoo", "Heating", "Carbon monoxide alarm", "Private entrance", "Wifi", "TV", "Smoke al...	118
["Lock on bedroom door", "Air conditioning", "Fire extinguisher", "Kitchen", "Shampoo", "Heating", "Carbon monoxi...	49
["Hangers", "Bathtub", "Microwave", "Dishwasher", "Dishes and silverware", "Kitchen", "Paid parking on premises", "...	115

b. Filtered rows where ‘price’ was equal to 0 or greater than 2000.
c. Parsed JSON to extract amenities.

amenities	price
Ceiling fan	143
Dedicated workspace	143
Gas stove	143
Cooking basics	143
Private backyard	143
Hot water kettle	143
Bed linens	143
Hot water	143
Free parking on premises	30
Carbon monoxide alarm	30

d. Duplicated query and made a ‘Top_amenities’ query that filters out all values that have fewer than 600 instances for each amenity.
e. Performed an inner merge on ‘amenities’ in order to leave only top amenities for main query.
f. Added a new column with ‘1’ in every row
g. Pivoted amenities on column with ‘1’, and replaced each null with 0, resulting in a boolean value for each amenity.

<u>NYC Crime:</u>

1. Load dataset into jupyter notebook using pandas
2. Used .info() and .columns to decide relevant columns to use
3. Set dataframe to columns that will be used:

```
df = df[['CMPLNT_FR_DT','OFNS_DESC','LAW_CAT_CD',
        'BORO_NM', 'SUSP_RACE', 'SUSP_SEX', 'Latitude',
        'Longitude','VIC_AGE_GROUP','VIC_RACE', 'VIC_SEX']]
```

4. Renamed columns for clarity:

```
df.rename(columns={'CMPLNT_FR_DT' : 'Date Occured',
                   'OFNS_DESC':'Crime Description','LAW_CAT_CD':'Level of Offense',
                   'BORO_NM':'Borough',"SUSP_RACE": "Suspect's Race",
                   'SUSP_SEX':"Suspect's Gender", 'VIC_AGE_GROUP': "Victim's Age Group",
                   'VIC_RACE':"Victim's Race",'VIC_SEX':"Victim's Gender"})
```

5. Renamed names in 'Borough' to match naming convention
    a. Ex: QUEENS > Queens
6. Switch date format from MM/DD/YYYY to YYYY-MM-DD to match naming convention
7. Remove Nulls

Zillow:

NYC Neighborhood rental homes average prices:

1. Read in csv into databricks notebook using pyspark with header = True and inferschema=True
2. Changed to a pandas dataframe
    a. Filtered State for NY only
    b. Dropped columns : RegionID, SizeRank, RegionType, StateName, Metro, StateCodeFIPS, MunicipalCodeFIPS
    c. Remove any null values
3. Unpivoted all the date columns
    a. Make an empty data frame for concatenation
    b. Iterate through each row of the filtered out dataset of NY
    c. Unpivot on the RegionName which is the neighborhood/county (5 of these) with the values to unpivot being all of the columns with dates
    d. Break out the year and month from the date into their own columns for easy filter late on
    e. Rename the pivoted values to 'Avg_Rental_Home_Price
    f. Concat the new pivoted row on the empty data frame as a union so they all lay on top of each other as it loops through each row
4. Map the County names into its neighborhood names using a dictionary to make a new column that has the neighborhood names
    a.
    ```
    mydict = {'Bronx County':'Bronx',
    'Kings County':'Brooklyn',
    'New York County': 'Manhattan',
    'Queens County':'Queens',
    'Richmond County':'Staten island'}
    ```
5. Change the datatypes to their respective values such as int for year, int for month, and date for the date column
6. Turn it back into a spark dataframe and write to csv into the datalake as a cleaned data

NYC Zillow home listings time series (No Neighborhood groupings):

1. Read csv into databricks using pyspark with header = True and inferschema = True
2. Turn into a pandas dataframe
3. Filter out RegionName to be for New York, NY only
4. Drop columns : RegionID, SizeRank, RegionType, StateName
5. Remove any null values
6. Unpivot the rest of the columns because they are all date columns
    a. Use RegionName as the id column and the columns to unpivot to be the rest of them since those are dates
7. Break out the months and year from the date into their own column once unpivoted
8. Rename the value column from the unpivot to be 'Count_of_Homes_Listed'
9. Change datatypes of the columns such as year to int, month to int and count of homes listed as int and date as date
10. Turn into a pyspark dataframe and read it back into the datalake

Hotels:

1. Read csv into databricks using pyspark with header = True and inferschema = True
2. Turn zipcodes into an int by removing the -extra4 digit zipcodes
3. Map the zipcodes into the respective neighborhoods -> added a neighborhood column
4. Filter out the rows with nonnull neighborhoods because these are the only ones in the 5 boroughs
5. Clean up datatypes
    a. Make star_rating a double (remove the rating where it is NULL as a string)
    b. Drop unnecessary columns like hotel id
6. Turn into pyspark dataframe and load back into datalake as a clean csv

Tourist Locations (Webscraped):

1. Webscrape a url that has 50 top tourist locations in new york city: `"https://www.attractionsofamerica.com/attractions/new-york-city-top-10-attractions.php"`
2. Find the html div that contains each individual tourist attraction and separate out its name, address, and website. Website or address can sometimes be null/ not found
3. Store all this information in a dataframe
4. Breakout the zipcode from the address
5. Map the zipcode into its respective new york city borough
6. Use a forward geolocation api to convert the address into a latitude and longitude
    a. Append this to the dataframe (probably have to be a copy)
7. Turn the columns into their respective datatypes
8. Turn into a pyspark dataframe and load back into datalake as a clean csv

Flight prices:

1. Load into Power Query (probably powerbi)
2. Filter out to only have data for New York City Airports (JFK, EWR, LGA) as the destination airport
3. Keep only the columns for the destination airport, date, fare price, total distance traveled
4. Divide the fare price / total distance traveled to try to normalize the price across different starting flight locations -> new column for this
5. Break out the months from the date
6. Aggregate the data to group by month and get the average of the fareprice/totaldistancetraveled

NYC Population:

1. Create a table in excel or powerBI and add the values in for the year 2022 manually to easily have a population table within powerBI when making visuals.

Data Mappings from source to SQL database:

NYC AirBnB Data:

SQL DB Table Name	*Table Column Name (Source)*	*SQL DB Field Name (Destination)*

Listings_BB

Source	Listings_BB Destination
id	Listing_ID primary key bigint
host_id	Host_ID bigint not null
name	Listing_Name varchar(1000) null
description	About varchar(8000) null
price	Price_Str varchar(1000) not null
property type	Type varchar(1000) null
accomodates	Number_of_People int null
minimum_nights	Min_n int null
maximum_nights	Max_n int null
first_review	First_Review_Date date null
price	Price float null

Reviews_BB

Source	Reviews_BB Destination
id	ID bigint primary key
listing_id	Listing_ID bigint null
date	Date date null
reviewer_id	ReviewerID int null
comments	Comment varchar(8000) null

Calendars_BB

Source	Calendars_BB Destination
listing_id	ListingID bigint primary key
- aggregated by listing_id and year of date for minimum_nights	Count int not null

Hosts_BB

Source	Hosts_BB Destination
host_id	Host_ID bigint primary key
host_is_superhost	Host_is_superhost varchar(10) null
host_listings_count	Host_listings_count int null
host_since	Host_since date null
host_response_time	Host_response_time varchar(100) null
host_response_rate	Host_response_rate varchar(100) null
host_acceptance_rate	Host_acceptance_rate varchar(100) null

NYCBoroughs_BB

Source	NYCBoroughs_BB Destination
-	ID int primary key identity(1,1)
neighbourhood_group_cleansed	Borough_name varchar(100) not null
-	County_name varchar(100) not null

Locations_BB

Source	Locations_BB Destination
-	ID bigint primary key identity(1,1)
Latitude float not null	Latitude float not null
Longitude float not null	Longitude float not null
(NYCBoroughs_BB - ID)	Bourough_ID int null
-	Zipcode varchar(100) null
-	Address varchar(100) null
-	Dataset varchar(100) not null

NYC Crime:

SQL DB Table Name	*Table Column Name (Source)*	*SQL DB Field Name (Destination)*

Crime_Types_BB

Source	Crime_Types_BB Destination
-	ID int primary key identity(1,1)
OFNS_DESC	Crime_Description varchar(100) not null
LAW_CAT_CD	Level_Of_Offense varchar(100) not null

NYC_Crimes_BB	

Source	NYC_Crimes_BB Destination
-	ID int primary key identity(1,1)
CMPLNT_NUM	Complain_Number bigint not null
CMPLNT_FR_DT	Date_Occured date not null
- (Crime_Types_BB - ID)	Crime_Type_ID int not null
- (Locations_BB - ID)	Location_ID bigint not null

Locations_BB	

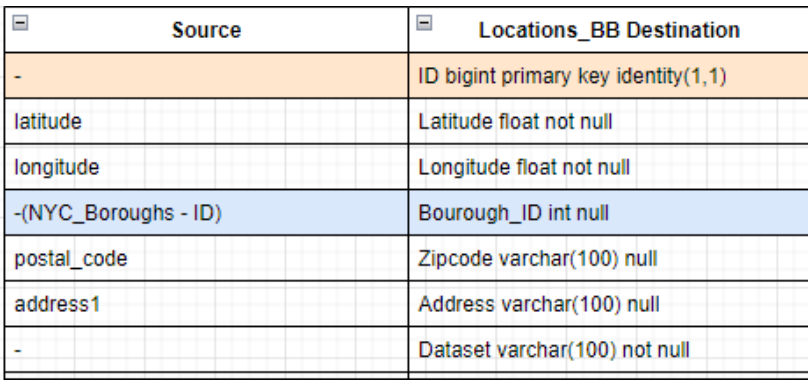
Source	Locations_BB Destination
-	ID bigint primary key identity(1,1)
Latitude	Latitude float not null
Longitude	Longitude float not null
- (NYCBoroughs_BB - ID)	Bourough_ID int null
-	Zipcode varchar(100) null
-	Address varchar(100) null
-	Dataset varchar(100) not null

NYCBoroughs_BB	*BORO_NM - Used for Bourough_ID*

Hotels:

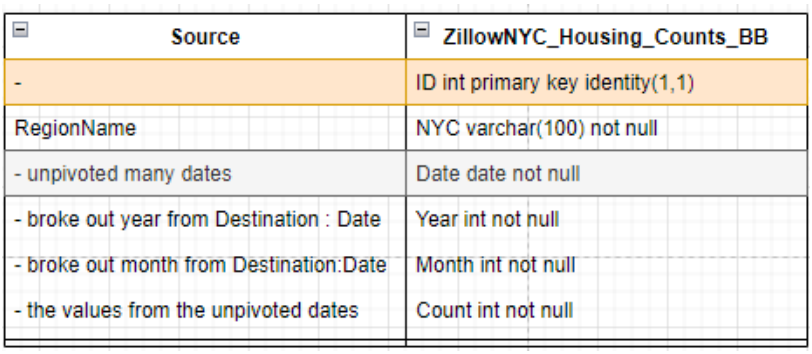
SQL DB Table Name	*Table Column Name (Source)*	*SQL DB Field Name (Destination)*
Hotels_BB		

Source	Hotels_BB Destination
-	ID int primary key identity(1,1)
name	Hotel_name varchar(100) not null
- (Locations_BB - ID)	Location_ID bigint not null
star_rating	Hotel_star_rating float not null
high_rate	Hotel_high_rate float not null
low_rate	Hotel_low_rate not null

Locations_BB	
NYCBoroughs_BB	*Had to map borough using zipcode*

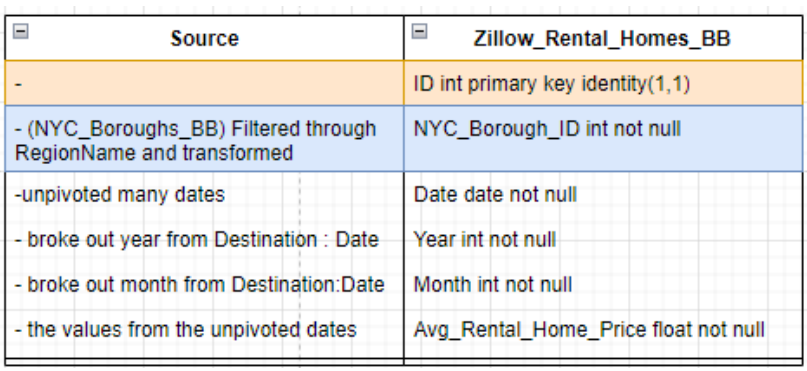
Source	Locations_BB Destination
-	ID bigint primary key identity(1,1)
latitude	Latitude float not null
longitude	Longitude float not null
-(NYC_Boroughs - ID)	Bourough_ID int null
postal_code	Zipcode varchar(100) null
address1	Address varchar(100) null
-	Dataset varchar(100) not null

Zillow Housing datasets:

SQL DB Table Name	*Table Column Name (Source)*	*SQL DB Field Name (Destination)*
ZillowNYC_Housing_Counts_BB		

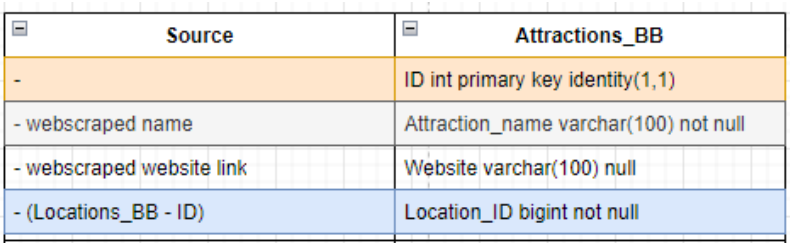
Source	ZillowNYC_Housing_Counts_BB
-	ID int primary key identity(1,1)
RegionName	NYC varchar(100) not null
- unpivoted many dates	Date date not null
- broke out year from Destination : Date	Year int not null
- broke out month from Destination:Date	Month int not null
- the values from the unpivoted dates	Count int not null

SQL DB Table Name	*Table Column Name (Source)*	*SQL DB Field Name (Destination)*

Zillow_Rental_Homes_BB	

Source	Zillow_Rental_Homes_BB
-	ID int primary key identity(1,1)
- (NYC_Boroughs_BB) Filtered through RegionName and transformed	NYC_Borough_ID int not null
-unpivoted many dates	Date date not null
- broke out year from Destination : Date	Year int not null
- broke out month from Destination:Date	Month int not null
- the values from the unpivoted dates	Avg_Rental_Home_Price float not null

Tourist Attractions:

SQL DB Table Name	*Table Column Name (Source)*	*SQL DB Field Name (Destination)*
Attractions_BB		
Locations_BB	*Address and Zipcode Webscraped and used Forward Geocoding to get latitude and longitude*	

Source	Attractions_BB
-	ID int primary key identity(1,1)
- webscraped name	Attraction_name varchar(100) not null
- webscraped website link	Website varchar(100) null
- (Locations_BB - ID)	Location_ID bigint not null

*Columns that do not show up on Data Mapping were dropped from the data and were not included in the SQL database.

*Columns may not be dropped for Machine Learning and column cleaning was decided by the individual model creator

Load

Load our cleaned and transformed data back into our data lake blob storage.

Then:

Machine Learning:

1. Load the cleaned dataset into python/notebook for machine learning processing into a model
2. Turn the categorical columns into numerical values with dummy variables

3. Make sure to get rid of nulls by either applying a mean, mode or removal
4. Standardize all the originally numerical column values
5. Train, validation, test split the dataset
6. Pick a model to train on + Pick features and target values
7. Train the model and check R squared score, and create visualizations

OR:

SQL Database -> Power BI visuals:

1. Created a password-protected SQL database 'Bionic_blobs', from the parent database.
1. Used Databricks to upload files from step 3 into SQL server, "Bionic_blobs."

```
database = 'Bionic_blobs'
user = 'Bionic_blobs'
password = 'SuperSecurepw!'
server = "cohort50sqlserver.database.windows.net"

for table_name in dataframes.keys():
    table = f"dbo.{table_name}"
    dataframes[table_name].write.format("jdbc").option(
        "url", f"jdbc:sqlserver://{server}:1433;databaseName={database};"
        ) \
        .mode("overwrite") \
        .option("dbtable", table) \
        .option("user", user) \
        .option("password", password) \
        .option("driver", "com.microsoft.sqlserver.jdbc.SQLServerDriver") \
        .save()
```

2. Load the cleaned datasets into SQL tables (denormalized)
3. Write DDL to create the foreign key, primary key, and unique key constraints to make relationships between the tables.
    a. See DDL file and ERD diagram for exact table definition and relationships
4. Write the DML to load the denormalized SQL tables to normalized SQL tables
    a. See DML file for the order in which the tables were filled with data and the specific select statements used from the denormalized data to input
5. Load the SQL database into Power BI to create visuals via merges, appending, aggregations, etc

Conclusion

Now we have all our data cleaned and either loaded into a normalized SQL database for visuals or cleaned to be easily changed into numerical values to train a machine learning model on. Our cleaned data is also placed back as csv files onto our data lake blob. We can make data factory pipelines to easily automate cleaning our data and putting it back into our data lake or SQL database or to be read into machine learning python code if the datasets were to ever update.