

BnBTrend Insights - Project Report

Introduction

Project Name : BnBTrend Insights

Problem Statement: Airbnb wants to set the right prices and decide where to grow in New York City. They need to know how prices and customer reviews are related. This helps them understand what customers like and how it affects their booking choices. By looking at these trends over time, Airbnb can make smarter decisions about prices and where to expand in NYC.

The BnBTrend Insights system aids managers in setting competitive prices and enhancing guest satisfaction by analyzing varied neighborhood demands and room types. It integrates structured and unstructured data, utilizing AWS services like S3, Lambda, Cloud Watch, RDS, and Glue for efficient data management.

Goal Statement: Airbnb wants to make more money and keep customers happy in New York City. They're trying to figure out how prices and customer reviews are related. By doing this, they can make better decisions about where to grow and how much to charge.

Our Stakeholders: Airbnb, Customers, Hosts, Investors.

Business/ Use Cases:

- For Marketing and Sales teams: Insights from booking trend analysis can inform targeted marketing campaigns tailored to specific customer segments and preferences.
- For Revenue Managers: By understanding demand fluctuations, stakeholders can dynamically adjust room rates to maximize revenue while ensuring competitiveness.
- For hotel shareholders: Stakeholders can leverage insights from booking trend analysis to provide investors and shareholders with transparent and data-driven performance reports.

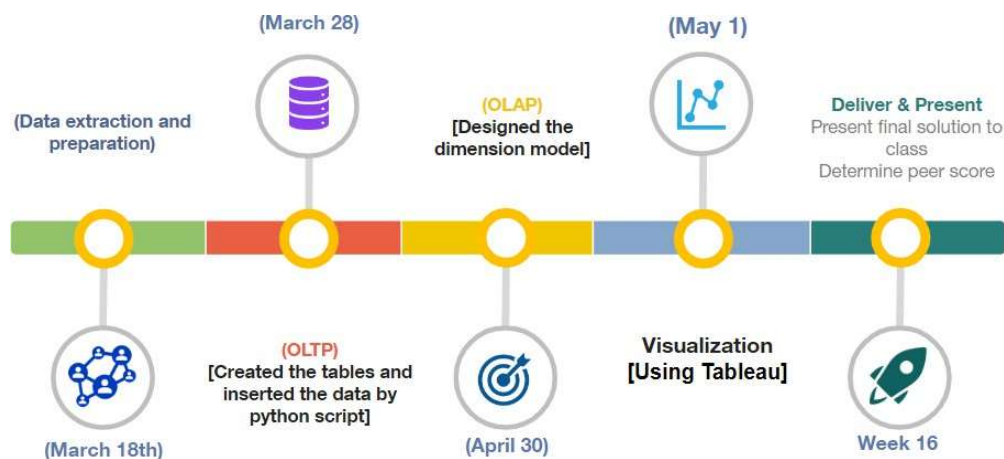
Kimball four-Step Dimensional Design Process:

1. **Select the business process:** In this case, the business process revolves around understanding the relationship between prices, customer reviews, booking trends, and

neighborhood demands in order to make informed decisions about pricing strategy and expansion in New York City.

2. **Declare the grain:** The grain represents the level of detail at which the facts will be stored and analyzed. In this scenario, the grain could be at the level of individual bookings or reservations, capturing details such as booking dates, prices, customer reviews, room types, neighborhood information, etc.
3. **Identify the dimensions:** Dimensions are the different perspectives or attributes by which the facts can be analyzed. In your case, dimensions could include time (booking dates), location (neighborhoods), customer segments, room types, pricing tiers, and customer satisfaction levels (captured through reviews).
4. **Identify the facts:** Facts are the numerical data or metrics that represent the measures of the business process. In this context, facts could include metrics such as booking frequency, average prices, customer review ratings, revenue generated per booking, occupancy rates, etc.

Timeline and Assumption for this initiative:



- Design Phase: 2 weeks
- Data Acquisition: 2 weeks
- Data Cleaning: 2 weeks
- Data Transformation: 3 weeks
- Data Visualization: 1 week
- Testing and Validation: 2 weeks
- Presentation and Launch: 2 weeks

Data Acquisition:

Source 1 : airbnb_listings.csv (Structured Dataset)

- Description: The dataset is a collection of listings from Airbnb for New York City in 2023.
- Method to Download: CSV Download
- Data Dictionary:

ID and Name	Unique identifier and name of the listing.
Host ID and Host Name	Information koi about the host.
ALocation	This includes both the neighborhood group (such as Manhattan or Brooklyn) and the specific neighborhood, along with geographical coordinates (latitude and longitude).
Room Type	The type of room offered (e.g., Entire home/apt, Private room).
Price and Minimum Nights	Number of Reviews, Last Review, and Reviews per Month: Metrics related to customer feedback.
Calculated Host Listings Count and Availability 365	Information on how many listings the host has in total and the number of days the listing is available in a year.
Number of Reviews LTM (Last Twelve Months) and License	Recent review count and licensing information, which might be related to local regulations

Source 2 : API (Unstructured Data)

- Method to Download: API Get request
- Unstructured Dataset
- Link: <https://insideairbnb.com/new-york-city>
- Data Dictionary:

Field	Type	Calculated	Description
id	integer		Airbnb's unique identifier for the listing
listing_url	text	y	
scrape_id	bigint	y	Inside Airbnb "Scrape" this was part of
last_scraped	datetime	y	UTC. The date and time this listing was "scraped".

source	text		One of "neighbourhood search" or "previous scrape". "neighbourhood search" means that the listing was found by searching the city, while "previous scrape" means that the listing was seen in another scrape performed in the last 65 days, and the listing was confirmed to be still available on the Airbnb site.
name	text		Name of the listing
description	text		Detailed description of the listing
neighborhood_overview	text		Host's description of the neighbourhood
picture_url	text		URL to the Airbnb hosted regular sized image for the listing
host_id	integer		Airbnb's unique identifier for the host/user
host_url	text	y	The Airbnb page for the host
host_name	text		Name of the host. Usually just the first name(s).
host_since	date		The date the host/user was created. For hosts that are Airbnb guests this could be the date they registered as a guest.
host_location	text		The host's self reported location
host_about	text		Description about the host
host_response_time			
host_response_rate			
host_acceptance_rate			That rate at which a host accepts booking requests.
host_is_superhost	boolean [t=true; f=false]		
host_thumbnail_url	text		
host_picture_url	text		
host_neighbourhood	text		
host_listings_count	text		The number of listings the host has (per Airbnb unknown calculations)
host_total_listings_count	text		The number of listings the host has (per Airbnb unknown calculations)
host_verifications			
host_has_profile_pic	boolean [t=true; f=false]		
host_identity_verified	boolean [t=true; f=false]		

neighbourhood	text		
neighbourhood_cleansed	text	y	The neighbourhood as geocoded using the latitude and longitude against neighborhoods as defined by open or public digital shapefiles.
neighbourhood_group_cleansed	text	y	The neighbourhood group as geocoded using the latitude and longitude against neighborhoods as defined by open or public digital shapefiles.
latitude	numeric		Uses the World Geodetic System (WGS84) projection for latitude and longitude.
longitude	numeric		Uses the World Geodetic System (WGS84) projection for latitude and longitude.
property_type	text		Self selected property type. Hotels and Bed and Breakfasts are described as such by their hosts in this field
room_type	text		[Entire home/apt Private room Shared room Hotel] All homes are grouped into the following three room types: Entire place Private room Shared room Entire place
accommodates	integer		The maximum capacity of the listing
bathrooms	numeric		The number of bathrooms in the listing
bathrooms_text	string		The number of bathrooms in the listing. On the Airbnb web-site, the bathrooms field has evolved from a number to a textual description. For older scrapes, bathrooms is used.
bedrooms	integer		The number of bedrooms
beds	integer		The number of bed(s)
amenities	json		
price	currency		daily price in local currency
minimum_nights	integer		minimum number of night stay for the listing (calendar rules may be different)
maximum_nights	integer		maximum number of night stay for the listing (calendar rules may be different)
minimum_minimum_nights	integer	y	the smallest minimum_night value from the calendar (looking 365 nights in the future)
maximum_minimum_nights	integer	y	the largest minimum_night value from the calendar (looking 365 nights in the future)

minimum_maximum_nights	integer	y	the smallest maximum_night value from the calender (looking 365 nights in the future)
maximum_maximum_nights	integer	y	the largest maximum_night value from the calender (looking 365 nights in the future)
minimum_nights_avg_ntm	numeric	y	the average minimum_night value from the calender (looking 365 nights in the future)
maximum_nights_avg_ntm	numeric	y	the average maximum_night value from the calender (looking 365 nights in the future)
calendar_updated	date		
has_availability	boolean		[t=true; f=false]
availability_30	integer	y	availability_x. The availability of the listing x days in the future as determined by the calendar. Note a listing may not be available because it has been booked by a guest or blocked by the host.
availability_60	integer	y	availability_x. The availability of the listing x days in the future as determined by the calendar. Note a listing may not be available because it has been booked by a guest or blocked by the host.
availability_90	integer	y	availability_x. The availability of the listing x days in the future as determined by the calendar. Note a listing may not be available because it has been booked by a guest or blocked by the host.
availability_365	integer	y	availability_x. The availability of the listing x days in the future as determined by the calendar. Note a listing may not be available because it has been booked by a guest or blocked by the host.
calendar_last_scraped	date		
number_of_reviews	integer		The number of reviews the listing has
number_of_reviews_ltm	integer	y	The number of reviews the listing has (in the last 12 months)
number_of_reviews_l30d	integer	y	The number of reviews the listing has (in the last 30 days)
first_review	date	y	The date of the first/oldest review
last_review	date	y	The date of the last/newest review
review_scores_rating			
review_scores_accuracy			
review_scores_cleanliness			
review_scores_checkin			
review_scores_communication			

review_scores_location			
review_scores_value			
license	text		The licence/permit/registration number
instant_bookable	boolean		[t=true; f=false]. Whether the guest can automatically book the listing without the host requiring to accept their booking request. An indicator of a commercial listing.
calculated_host_listings_count	integer	y	The number of listings the host has in the current scrape, in the city/region geography.
calculated_host_listings_count_entire_homes	integer	y	The number of Entire home/apt listings the host has in the current scrape, in the city/region geography
calculated_host_listings_count_private_rooms	integer	y	The number of Private room listings the host has in the current scrape, in the city/region geography
calculated_host_listings_count_shared_rooms	integer	y	The number of Shared room listings the host has in the current scrape, in the city/region geography
reviews_per_month	numeric	y	<p>The average number of reviews per month the listing has over the lifetime of the listing.</p> <p>Psuedocoe/~SQL:</p> <p>IF scrape_date - first_review <= 30 THEN number_of_reviews ELSE number_of_reviews / ((scrape_date - first_review + 1) / (365/12))</p>

Data Models

Fact_Listings (*Fact table*)

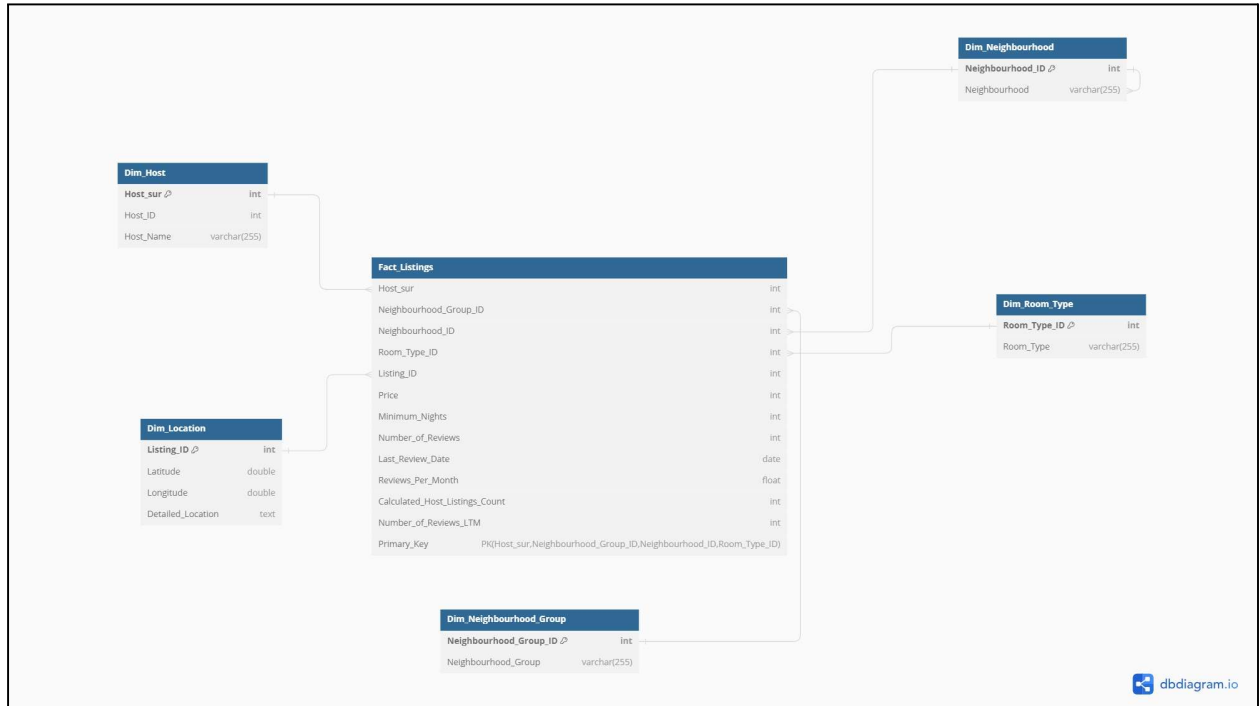
Dim_Host (*Dimension table*)

Dim_Location (*Dimension table*)

Dim_Neighbourhood_Group (*Dimension table*)

Dim_Room_type (*Dimension table*)

Dim_Neighbourhood (*Dimension table*)



Data Profiling

Data profiling is done to the dataset to understand its structure, quality, and characteristics. This helps us to gain insights into the data before performing any analysis. The screenshot below shows the overview of the dataset that we have.

Link to the htm page of the data profiling : [Link to our github](#)

Overview		
<div> <div>Overview</div> <div>Alerts 16</div> <div>Reproduction</div> </div>		
Dataset statistics		
Number of variables	17	
Number of observations	42931	
Missing cells	63506	
Missing cells (%)	8.7%	
Duplicate rows	0	
Duplicate rows (%)	0.0%	
Total size in memory	5.6 MIB	
Average record size in memory	136.0 B	
Variable types		
Numeric	10	
Text	4	
Categorical	2	
Date Time	1	

Platform and services to use:

A. Amazon Web Services (AWS)

B. Services to be used:

AWS RDS (Relational Database Service) - We will utilize AWS RDS to store both OLTP (Online Transaction Processing) and OLAP (Online Analytical Processing) data related to hotel bookings.

AWS VPC (Virtual Private Cloud) - AWS VPC will be employed to create a virtual network environment for our project, enabling us to isolate our resources and securely manage communication between them.

AWS S3 (Simple Storage Service) - Raw data pertaining to hotel bookings will be stored in AWS S3 buckets.

AWS Glue - AWS Glue will be used to automate the Extract, Transform, Load (ETL) process for our data.

AWS Lambda - To obtain data to and from S3 file.

AWS CloudWatch - AWS CloudWatch will be utilized for scheduling Lambda functions and monitoring the health and performance of our AWS resources.

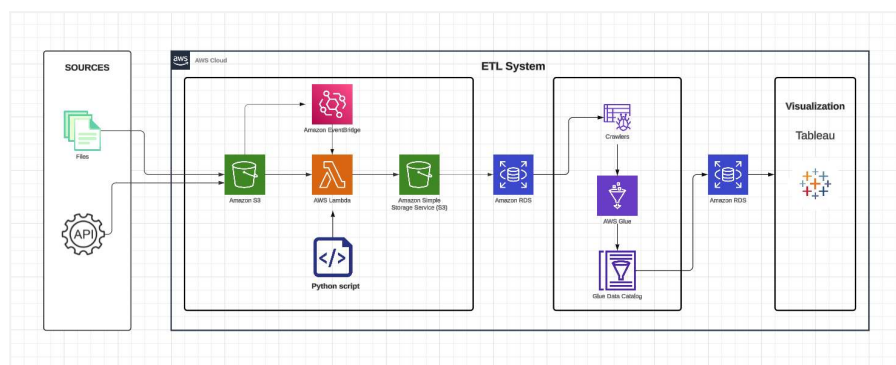
C. Tableau for Visualization

AWS Architecture:

AWS architecture refers to the design and structure of applications, systems, and infrastructure deployed on Amazon Web Services (AWS) cloud platform.

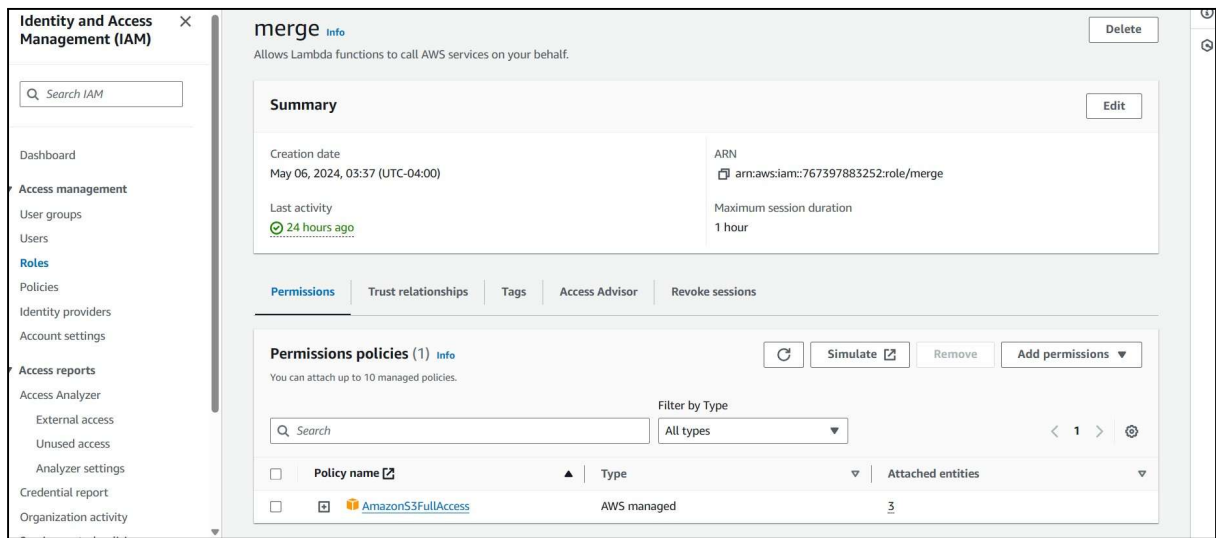
Here, the data is passing from the data sources to AWS cloud.

There are various functions happening in the ETL System, and finally the visualization of the data based on the problem statement is on the tableau dashboard.

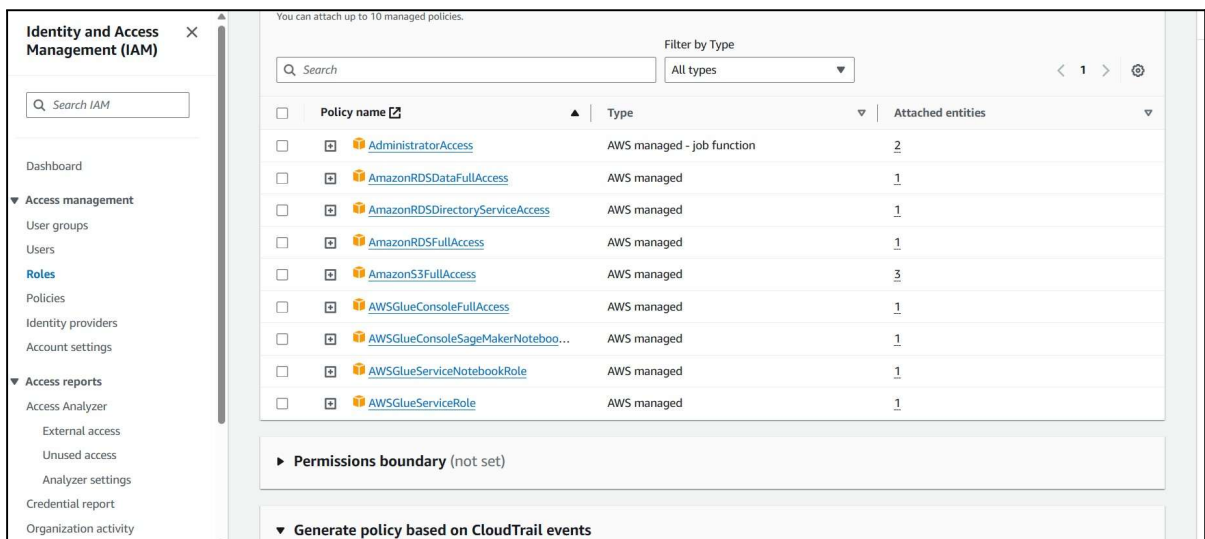


IAM Roles and Responsibilities:

This screenshot shows the IAM policies of the Lambda function.

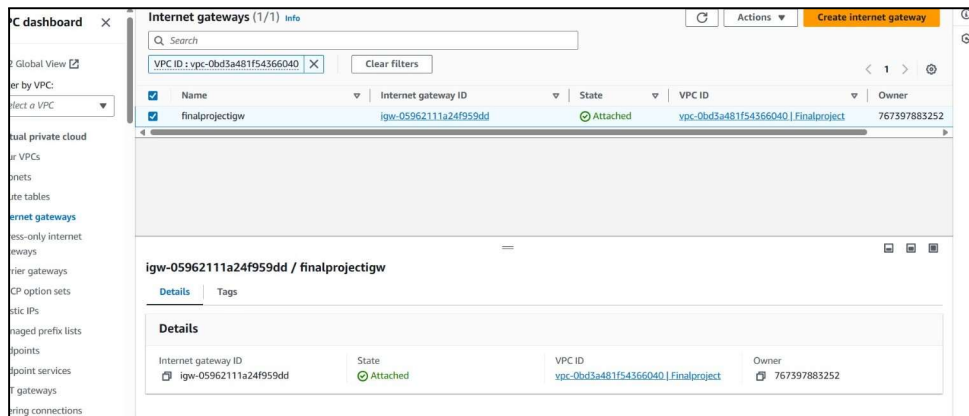


This screenshot shows the IAM policies for AWS Glue.

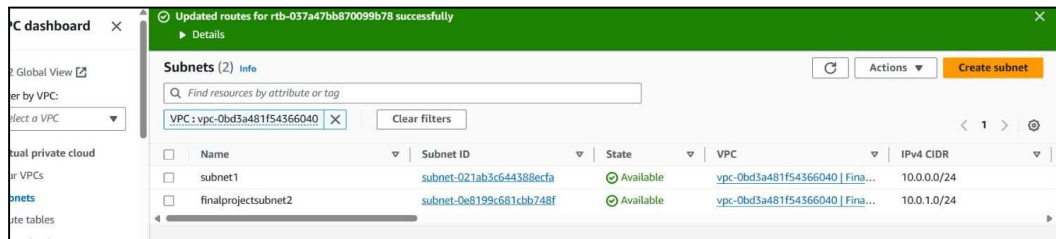


AWS Services

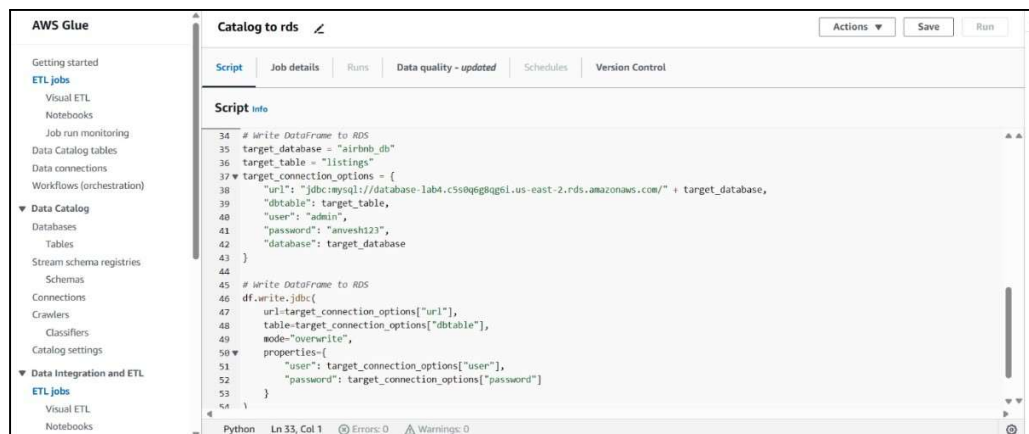
VPC Information (Virtual Private Network):



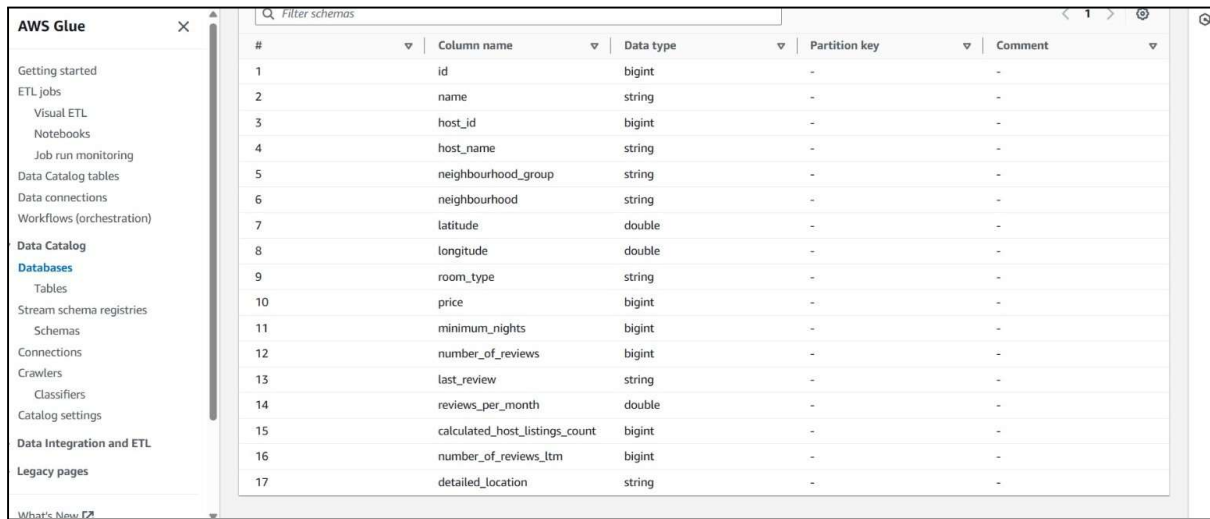
Subnet Information



ETL Jobs



Tables into the crawler



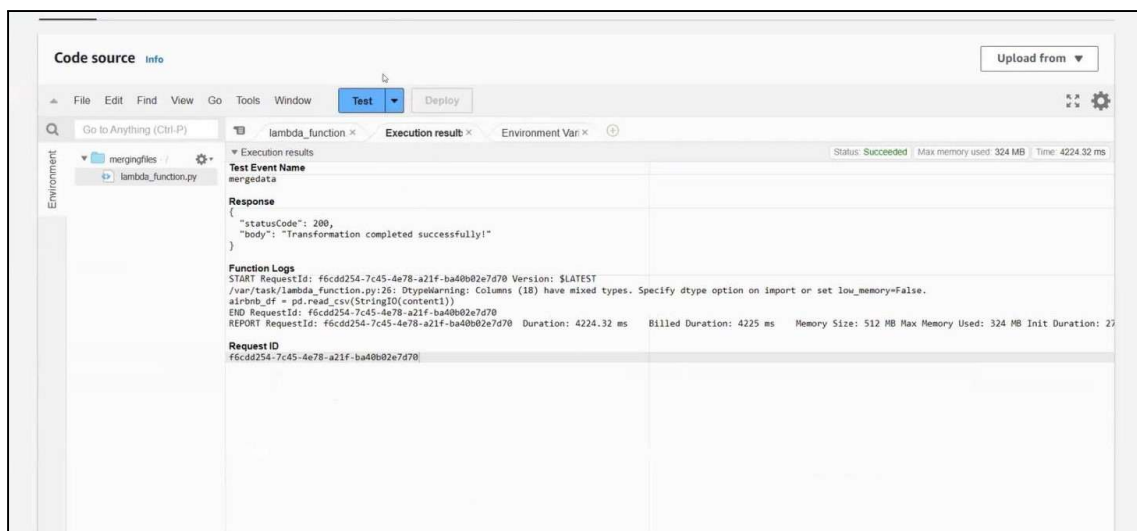
The screenshot shows the AWS Glue console interface. On the left is a navigation menu with options like 'Getting started', 'ETL jobs', 'Data Catalog', 'Databases', 'Tables', 'Stream schema registries', 'Schemas', 'Connections', 'Crawlers', 'Classifiers', 'Catalog settings', 'Data Integration and ETL', and 'Legacy pages'. The 'Tables' option is selected. The main area displays a table schema with columns: #, Column name, Data type, Partition key, and Comment. The table contains 17 rows of data.

#	Column name	Data type	Partition key	Comment
1	id	bigint	-	-
2	name	string	-	-
3	host_id	bigint	-	-
4	host_name	string	-	-
5	neighbourhood_group	string	-	-
6	neighbourhood	string	-	-
7	latitude	double	-	-
8	longitude	double	-	-
9	room_type	string	-	-
10	price	bigint	-	-
11	minimum_nights	bigint	-	-
12	number_of_reviews	bigint	-	-
13	last_review	string	-	-
14	reviews_per_month	double	-	-
15	calculated_host_listings_count	bigint	-	-
16	number_of_reviews_ltm	bigint	-	-
17	detailed_location	string	-	-

Data Extraction

Data extraction is the process of retrieving or pulling data from various sources, such as databases, files, websites, or other repositories, to be used for analysis, reporting, or any other purpose. It's a fundamental step in data processing pipelines, especially in scenarios where data needs to be aggregated, transformed, and loaded into another system or format.

Data extraction is done with the help of AWS Lambda. The screenshot below shows the execution of the Lambda function.



Data Transformation

Data transformation takes data from one place, Amazon S3, and puts it into another place called the RDS database. Basically, we made it so the system can talk to both S3 and the RDS database. It grabs the data we need from S3, makes sure it's all good, and then pops it into the RDS database so we can use it. It is even set up to do this automatically and regularly, so the database is always full of the latest info from the data coming in with AWS Lambda.

```
print("Inserting data into the database...")
df.to_sql('listings', con=engine, index=False, if_exists='append', chunksize=500) # Adjust chunksize based on your needs
print("Data successfully inserted into the database.")

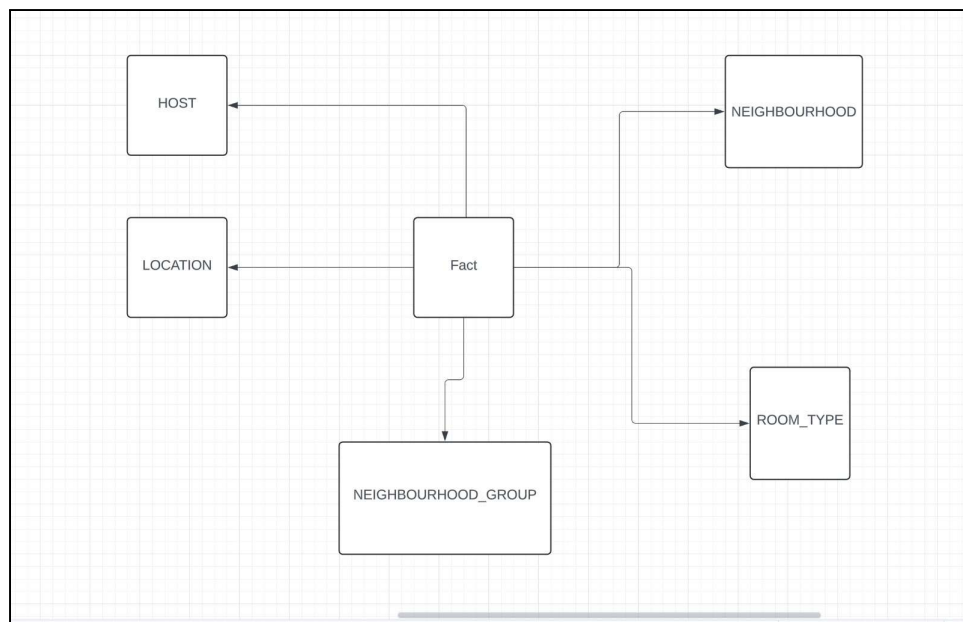
# Main execution
bucket_name = 'group-1-final-project-finalsdata'
file_key = 'final_airbnb_dataset.csv'
database_details = {
    'host': 'database-2.c5s0q6g0qg6i.us-east-2.rds.amazonaws.com',
    'user': 'admin',
    'password': 'anveshalluri',
    'database': 'airbnb_db'
}

print("Script started.")
data = load_data_from_s3(bucket_name, file_key)
insert_data_to_rds(data, database_details)
print("Script executed successfully.")
```

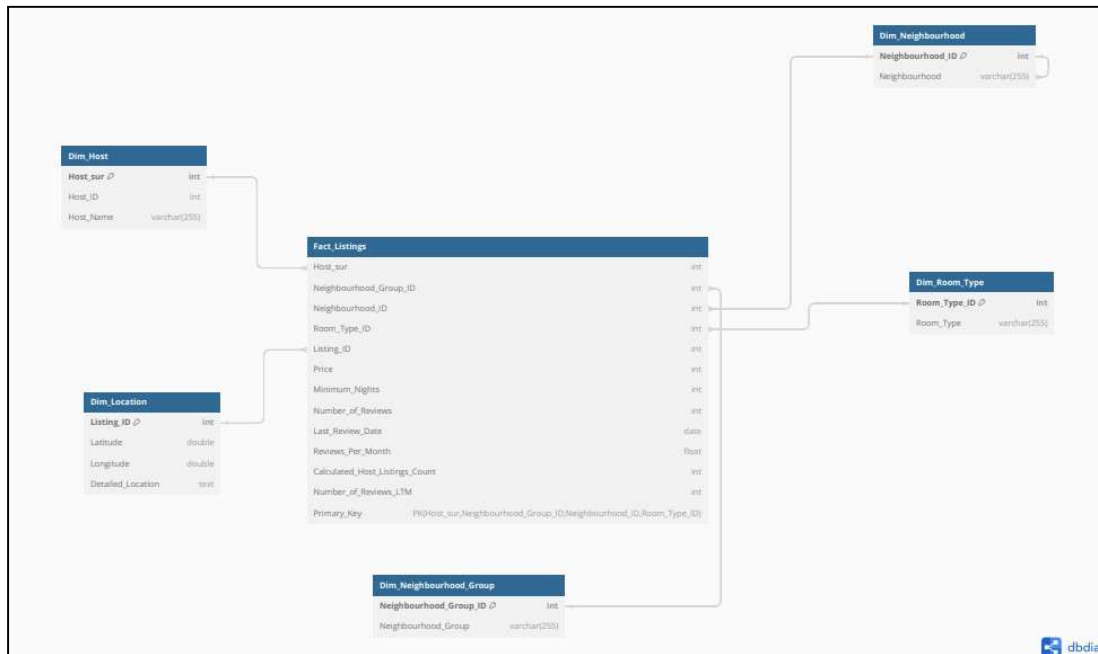
Script started.
Connecting to S3 to retrieve data...
Data successfully loaded from S3.
Connecting to RDS database...
Inserting data into the database...
Data successfully inserted into the database.
Script executed successfully.

Database Models:

Conceptual Data Model



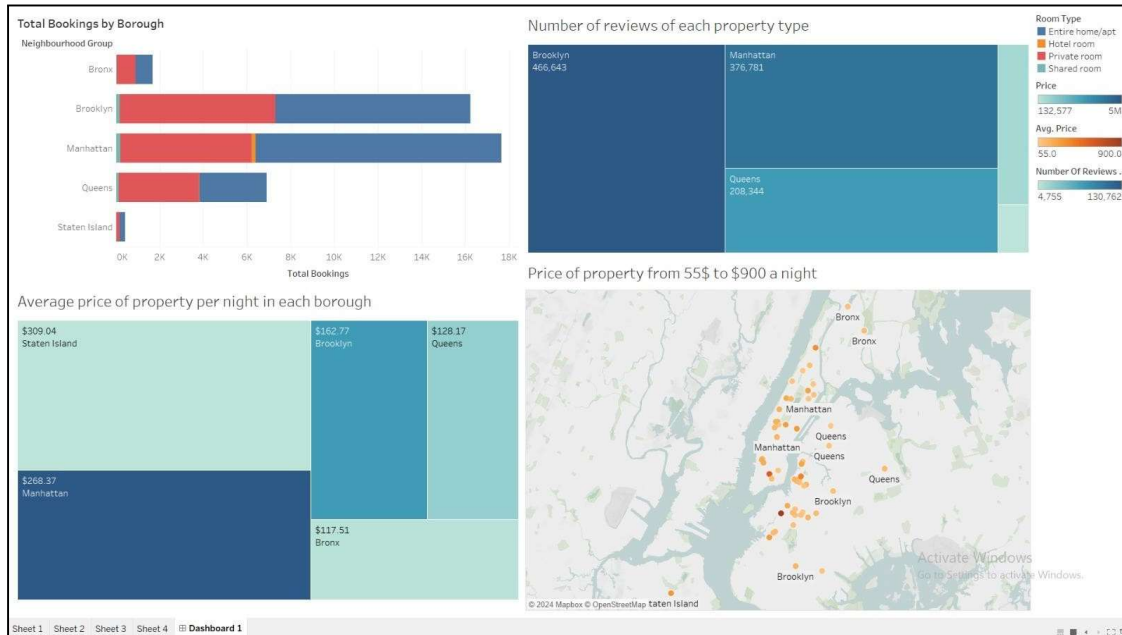
Logical Model



Physical Model



Visualization (using Tableau)



Recommendations from visualization:

Total Bookings by borough

- We saw that Manhattan has the highest number of bookings.
- Brooklyn has more private room bookings than any other borough.
- Staten Island has the lowest number of bookings.
- Manhattan has more bookings for the entire apartment.

Average price of property per night

- Staten Island has the highest price per night (\$309.04) compared to any other borough.
- Bronx has the lowest price per night (\$117.51)

Number of reviews of each property type

- Brooklyn has the highest number of reviews whereas Manhattan is the second highest.
- Staten Island and the Bronx have the least number of reviews.

Price of property per night

- Price of property ranging from average \$55 to \$900 per night in New York City.
- One of the places in Brooklyn has the highest cost per night.
- One place in Queens has the lowest price per night.

Challenges:

- Data Quality Issues:
 - Missing Values: It was common to find missing values in the datasets, which could have impacted our analysis and results. We employed strategies such as imputation and exclusion to deal with these missing values. Imputation involved replacing missing values with estimated values based on other available data points, while exclusion was considered for cases where missing values were too numerous or critical for analysis.
 - Outliers: Outliers, which are data points significantly deviating from the rest of the dataset, posed challenges to our statistical analyses. We employed techniques such as visual inspection and statistical methods like z-score or interquartile range to detect and handle outliers. Additionally, domain knowledge played a crucial role in determining whether outliers were genuine data points or errors.
 - Inconsistencies: Inconsistent data, such as variations in formatting, units, or naming conventions, could have led to errors in our analysis. To mitigate these inconsistencies, we standardized data formats and values and implemented validation checks during data entry or integration.
- AWS Service Configuration:
 - AWS Glue: Configuring AWS Glue for ETL processes involved defining data sources, transformations, and target destinations. We faced challenges in mapping complex data transformations, handling schema changes, and optimizing performance for large datasets. Thorough testing and iteration were essential to ensure the accuracy and efficiency of our ETL processes.
 - AWS Lambda: Setting up AWS Lambda functions for real-time data handling required defining triggers, writing code to process incoming data, and configuring event sources. Challenges included managing dependencies, optimizing code for performance and cost efficiency, and ensuring seamless integration with other AWS services like S3 or DynamoDB.
- Integration Complexity:
 - Multiple Data Sources: Integrating data from diverse sources such as structured databases, unstructured files, and APIs introduced complexity in data ingestion and processing. Challenges included data format conversion, data validation, and synchronization of data refresh schedules to ensure consistency and timeliness.
 - AWS Services Integration: Seamlessly integrating AWS services like S3, Glue, Lambda, and RDS required careful planning and coordination. Challenges arose in configuring service permissions, managing data pipelines, and troubleshooting connectivity issues. We implemented robust monitoring and error handling mechanisms to ensure smooth data flow and minimize disruptions in our data warehouse solution.
- Importance of Data Profiling:

- Conducted thorough data profiling and analysis early in the project: Identified potential data quality issues such as missing values, outliers, and inconsistencies, analyzed data distribution, patterns, and relationships to understand the dataset's characteristics, and detected anomalies that could impact the accuracy and reliability of subsequent analysis.
- Devised appropriate cleansing strategies based on profiling insights:

Further Improvement

1. **Integration with Additional Data Sources:** Incorporating data from additional sources beyond Airbnb listings, such as local events data, weather patterns, or economic indicators can provide a more comprehensive understanding of hotel booking trends and enhance the value of insights.
2. **Realtime data :** AWS Lambda for real-time event processing and Amazon S3 for data storage and management, you can develop a scalable and efficient dynamic pricing optimization system. This system can adapt to changing market conditions, competitor prices, and demand fluctuations to maximize revenue and profitability for your business.