

Building a Scalable Data Pipeline for Airbnb Listings Analytics in NYC

Bronze Layer



Upload the Raw data in DataBricks
and write it in S3 in Delta format

Silver Layer



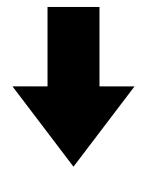
Perform all Transformations and cleaning
process and store the data in S3

Gold Layer

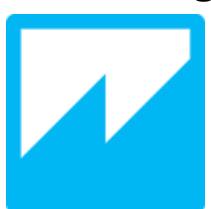
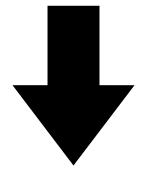


Perform all Aggregate operations
'and store the data in S3

AWS Glue



AWS Athena



Using Glue Crawler table is
created and visualisation is
done using Athena

Objective:

The project addresses key challenges in hospitality analytics by providing data-driven insights into Airbnb listings and host behaviours. The pipeline will support:

- Revenue optimization – Price benchmarking and anomaly detection
- Operational efficiency – Automated host performance tracking
- Market analysis – Neighbourhood-level demand patterns
- Customer experience – Review trends and availability monitoring

The solution must balance scalability, cost-efficiency, and business agility while handling geographic and temporal data. Business Goal: This pipeline delivers tangible business outcomes:

- 15-25% improvement in optimal pricing identification
 - 40% reduction in manual market analysis time
 - Actionable host performance benchmarks
-

The solution transforms raw listing data into strategic assets using AWS services, enabling data-driven decision making for hosts, managers, and analysts. Dataset Source: Kaggle Title: Airbnb New York Dataset URL: <https://www.kaggle.com/datasets/dgomonov/new-york-city-airbnb-open-data> Description: This dataset contains Airbnb New York Dataset listings including host id, host names, id, name prices, minimum_night, reviews ect., Prerequisite:

- Databricks Community version
- AWS Account

Bronze Layer

Step 1: Create S3 Bucket for "Bronze", "Silver" and "Gold"

Go to AWS Search Console S3 Create Bucket

Provide Bucket name

Click on "Create Bucket" [Refer Image 1]

S3 Bucket Flow Chart:

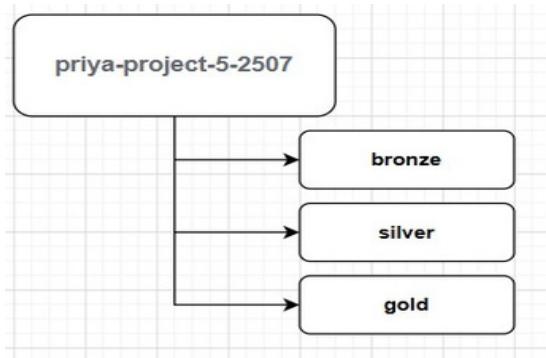


Image 1:

The screenshot shows the AWS S3 console for the bucket "priya-project-5-2507". The "Objects" tab is selected, displaying five objects: "bronze/", "gold/", "Queryoutput/", "silver/", and "silveroverall/". Each object is a folder. The interface includes standard S3 actions like Copy S3 URI, Copy URL, Download, Open, Delete, Actions, Create folder, and Upload. The bottom of the screen shows CloudShell, Feedback, and various AWS links.

Step 2: Update the IAM Role in IAM USER to read and write files to S3 bucket from Databricks and vice versa

Go to AWS Search Console IAM User Project_user_Priya Add Permissions Create Inline Policy

Add the following code

Create the new permissions[Refer Image 2]

JSON Code:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3>ListBucket",  
                "s3GetObject"  
            ],  
            "Resource": [  
                "arn:aws:s3:::priya-project-5-2507 "  
            ]  
        }  
    ]  
}
```

```

        "arn:aws:s3:::priya-project-5-2507 /*"
    ]
}
]
}

```

Image 2:

The screenshot shows the AWS IAM User Permissions page for a user named 'Project_user_Priya'. The 'Permissions' tab is selected. The table lists eight attached policies:

Policy name	Type	Attached via
AdministratorAccess	AWS managed - job function	Directly
AmazonAthenaFullAccess	AWS managed	Directly
AmazonKinesisFullAccess	AWS managed	Directly
AmazonS3FullAccess	AWS managed	Directly
AWSGlueConsoleFullAccess	AWS managed	Directly
JAMFullAccess	AWS managed	Directly
s3access	Customer inline	Inline
s3accessstodatabricks	Customer inline	Inline

Step 3: Create IAM Role

Go to AWS Search Console IAM Roles Create Role

Select “AWS Service” in “Trusted entity type”

Select “Glue” in “Use case”

Click Next

Add the following Permissions

- [AmazonS3FullAccess](#)
- [AWSGlueServiceRole](#)
- [CloudWatchEventsFullAccess](#)
- [AmazonAthenaFullAccess](#)
- [AWSQuicksightAthenaAccess](#)

Provide Role Name as “glue_service_role” [Refer Image 3]

Image 3:

The screenshot shows the AWS IAM Role Policies page for a role named 'glue-service-role'. A green banner at the top indicates that policies have been successfully attached. The table lists five attached policies:

Policy name	Type	Attached entities
AmazonAthenaFullAccess	AWS managed	4
AmazonS3FullAccess	AWS managed	6
AWSGlueServiceRole	AWS managed	4
AWSQuicksightAthenaAccess	AWS managed	1
CloudWatchEventsFullAccess	AWS managed	3

Step 4: Upload the data into Bronze layer

Upload the data in Databricks platform

Write the raw data in AWS S3 bucket in delta format using Databricks platform[Refer Image 4,5,6]

Image 4:

```
#Importing Packages
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql import *
from pyspark.sql.types import *
from pyspark.sql.functions import *
from pyspark.sql import types
from pyspark.sql.window import *
from pyspark.sql.dataframe import DataFrame
spark = SparkSession.builder.appName("Airbnb Listings in NYC").getOrCreate()
```

1. Ingest all CSV files into Databricks using PySpark.

```
#Creating Data Frames for Sales,Distributor and products data
airbnb_df = spark.read.csv("dbfs:/FileStore/shared_uploads/priyaraman0712@gmail.com/AB_NYC_2019.csv",
                           header=True,
                           inferSchema=True,
                           multiLine=True,
                           quote='"',
                           escape='',
                           mode="DROPMALFORMED",
                           ignoreLeadingWhiteSpace=True,
                           ignoreTrailingWhiteSpace=True)
```

Image 5:

```
import boto3
from delta.tables import DeltaTable

# Replace with your credentials and bucket name
aws_access_key = "*****"
aws_secret_key = "*****"
region_name = "ap-south-1" # Use your region
bucket_name = "priya-project-5-2507"
prefix = "" # optional: folder inside your bucket, set to empty string if not used

# Create boto3 session
session = boto3.Session(
    aws_access_key_id=aws_access_key,
    aws_secret_access_key=aws_secret_key,
    region_name=region_name
)

s3 = session.resource('s3')

# Set AWS credentials via Hadoop config
spark._jsc.hadoopConfiguration().set("fs.s3a.access.key", aws_access_key)
spark._jsc.hadoopConfiguration().set("fs.s3a.secret.key", aws_secret_key)
spark._jic.hadoopConfiguration().set("fs.s3a.endpoint", "s3.amazonaws.com")
spark._jic.hadoopConfiguration().set("fs.s3a.impl", "org.apache.hadoop.fs.s3a.S3AFileSystem")
airbnb_df.write \
    .format("delta") \
    .mode("overwrite") \
    .option("header", "true") \
    .option("multiline", True) \
    .option("escape", "\\") \
    .option("quote", "\\") \
    .save("s3://priya-project-5-2507/bronze/")
```

Image 6:

The screenshot shows the AWS S3 console interface. The top navigation bar includes the AWS logo, a search bar, and user information (Asia Pacific (Mumbai), Priya_Raman_0712). Below the navigation bar, the path is shown as Amazon S3 > Buckets > priya-project-5-2507 > bronze/. On the left, there's a sidebar with 'bronze/' selected. The main area displays a table of objects under the 'Objects' tab. The table has columns for Name, Type, Last modified, Size, and Storage class. There are three parquet files listed:

Name	Type	Last modified	Size	Storage class
_delta_log/	Folder	-	-	-
part-00000-116b5917-7834-42b3-a950-06639f4999a8.c000.snappy.parquet	parquet	July 25, 2025, 15:17:48 (UTC+05:30)	2.4 MB	Standard
part-00000-a4c5c74a-f135-4e5a-978f-c9bc18be67fd.c000.snappy.parquet	parquet	July 25, 2025, 15:16:12 (UTC+05:30)	2.4 MB	Standard

Silver Layer

Step 5: Perform the Transformation task in the Silver layer

1. Clean and standardize the pricing field(remove 0 or negative values). [Refer Image 9]
2. Remove duplicate product entries and fix missing values. [Refer Image 7 & 8]
3. Change the Datatypes as per data[Refer Image 10]
4. Dropping null values in “name” column
5. Write the file in S3 in parquet format[Refer Image 11]

Image 7:

Databricks Notebook titled "Project 5 - Airbnb Listings in NYC 2025-07-25 10:59:23 (Python)". The code cell contains:

```
#Drop Duplicated
airbnb_df.count()
airbnb_df.dropDuplicates()
airbnb_df.count()
```

Output: Out[6]: 48895

The next cell shows the count of null values across all columns:

```
null_counts = airbnb_df.select([
    sum(when(col(c).isNull(), 1).otherwise(0)).alias(c)
    for c in airbnb_df.columns
])

null_counts.display()
```

Output: null_counts: pyspark.sql.dataframe.DataFrame = [id: long, name: long ... 14 more fields]

A preview of the data table shows one row with all null values:

id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude
1	0	16	0	21	0	0	0

1 row

Image 8:

Databricks Notebook titled "Project 5 - Airbnb Listings in NYC 2025-07-25 10:59:23 (Python)". The code cell contains:

```
airbnb_df.dropna()
airbnb_df.count()
```

Output: Out[8]: 48895

The next cell shows the count of null values again:

```
#Dropping Null value in name
airbnb_df = airbnb_df.dropna(subset=["name"])

airbnb_df
```

Output: airbnb_df: pyspark.sql.dataframe.DataFrame = [id: integer, name: string ... 14 more fields]

The final cell shows the count of the cleaned dataset:

```
null_counts = airbnb_df.select([count(when(col(c).isNull(), c)).alias(c) for c in airbnb_df.columns])
display(null_counts)
```

Output: null_counts: pyspark.sql.dataframe.DataFrame = [id: long, name: long ... 14 more fields]

A preview of the data table shows one row with all null values:

id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude
1	0	0	0	21	0	0	0

1 row

Image 9:

Databricks Notebook titled "Project 5 - Airbnb Listings in NYC 2025-07-25 10:59:23 (Python)". The code cell contains:

```
#Dropping 0 or Negative pricing
airbnb_df = filter(when("price") >= 0).count()
airbnb_df = airbnb_df.filter("price" >= 0)

airbnb_df
```

Output: Out[13]: 48866

The final cell prints the schema of the dataset:

```
airbnb_df.printSchema()
```

Output:

```
root
 |-- id: integer (nullable = true)
 |-- name: string (nullable = true)
 |-- host_id: integer (nullable = true)
 |-- host_name: string (nullable = true)
 |-- neighbourhood_group: string (nullable = true)
 |-- neighbourhood: string (nullable = true)
 |-- latitude: double (nullable = true)
 |-- longitude: double (nullable = true)
 |-- room_type: string (nullable = true)
 |-- price: integer (nullable = true)
 |-- availability_365: integer (nullable = true)
 |-- number_of_reviews: integer (nullable = true)
 |-- last_review: date (nullable = true)
 |-- reviews_per_month: double (nullable = true)
```

Image 10:

```

from pyspark.sql.functions import to_date, date_format
airbnb_df = airbnb_df.withColumn(
    "last_review", date_format(to_date("last_review", "dd-MMM-yyyy"), "yyyy-MM-dd")
)

```

16

```

display(airbnb_df.groupBy("neighbourhood_group").agg(count(col("id")).alias("cnt")).orderBy(col("cnt").desc())))

```

17

neighbourhood_group	cnt
Manhattan	21651
Brooklyn	20089
Queens	5666
Bronx	1089
Staten Island	373

5 rows

18

```

display(airbnb_df)

```

Image 11:

```

# #writing to Silver Layer w.r.t neighbourhood_group
# #Already Client is established

# airbnb_df.write \
#     .format("csv") \
#     .mode("overwrite") \
#     .option("header", "true") \
#     .option("multiline", True) \
#     .option("escape", "\\") \
#     .option("quote", "\\") \
#     .partitionBy("neighbourhood_group") \
#     .save("s3://priya-project-5-2507/silver/")
airbnb_df.write \
    .format("parquet") \
    .mode("overwrite") \
    .partitionBy("neighbourhood_group") \
    .save("s3://priya-project-5-2507/silver/")

```

19

```

airbnb_df.write \
    .format("parquet") \
    .mode("overwrite") \

```

20

Output:

Amazon S3 > Buckets > priya-project-5-2507 > silver/

silver/

Objects (6)

Name	Type	Last modified	Size	Storage class
<u>SUCCESS</u>	-	July 28, 2025, 20:30:12 (UTC+05:30)	0 B	Standard
neighbourhood_group=Bronx/	Folder	-	-	-
neighbourhood_group=Brooklyn/	Folder	-	-	-
neighbourhood_group=Manhattan/	Folder	-	-	-
neighbourhood_group=Queens/	Folder	-	-	-

Gold Layer

Step 6: Perform aggregate function

- Identify the average price for room type [Refer Image 12]
- Identify the average review by area[Refer Image 13]
- Identify the average review by neighbourhood[Refer Image 14]
- Identify the average price by area[Refer Image 15]
- Identify the room_type by area[Refer Image 16]
- Identify the top_host[Refer Image 17]
- Identify the average min_night_requirement[Refer Image 18]
- Identify the average high_availability[Refer Image 19]
- Identify the average host_performance_summary[Refer Image 20]
- Write the data into S3 Bucket in Parquet format[Refer Image 21,22,23]

Image 12:

Databricks Notebook titled "Project 5 - Airbnb Listings in NYC 2025-07-25 10:59:23 (Python)". The code cell contains the following Python code:

```
neighbourhood_group: string (nullable = true)
neighbourhood: string (nullable = true)
latitude: double (nullable = true)
longitude: double (nullable = true)
room_type: string (nullable = true)
price: integer (nullable = true)
minimum_nights: integer (nullable = true)
number_of_reviews: integer (nullable = true)
last_review: string (nullable = true)
reviews_per_month: double (nullable = true)
calculated_host_listings_count: integer (nullable = true)
availability_365: integer (nullable = true)
```

The output shows the results of the aggregation:

room_type	avg_price
Entire home/apt	211.8
Private room	89.81
Shared room	70.2

Image 13:

Databricks Notebook titled "Project 5 - Airbnb Listings in NYC 2025-07-25 10:59:23 (Python)". The code cell contains the following Python code:

```
#Average Review per month Rate by Area
avg_review_area = airbnb_df.groupby("neighbourhood_group").agg(round(avg("reviews_per_month"),2).alias("avg_reviews_per_month")).orderBy(col("avg_reviews_per_month").desc())
avg_review_area.show()
```

The output shows the results of the aggregation:

neighbourhood_group	avg_reviews_per_month
Queens	1.94
Staten Island	1.87
Bronx	1.84
Brooklyn	1.28
Manhattan	1.27

Image 14:

Databricks Notebook Project 5 - Airbnb Listings in NYC 2025-07-25 10:59:23 (Python)

```
#Average Review Rate by neighbourhood
avg_review_neighbourhood = airbnb_df.groupBy("neighbourhood").agg(round(avg("number_of_reviews"),2).alias("avg_number_of_reviews")).orderBy(col("avg_number_of_reviews").desc())
avg_review_neighbourhood.show()
```

neighbourhood	avg_number_of_reviews
Richmondtown	79.0
Elttingville	76.0
Mount Eden	70.0
Springfield Gardens	69.99
Tompkinsville	57.14
Huguenot	55.67
Manhattan Beach	50.63
Highbridge	48.81
South Ozone Park	48.68
East Morrisania	47.22
Clifton	47.2
Baychester	44.29
Woodlawn	44.0
Allerton	42.93
Jamaica	42.9
Mott Haven	42.37
City Island	42.17
Bay Terrace	41.5

only showing top 20 rows

26

Image 15:

Databricks Notebook Project 5 - Airbnb Listings in NYC 2025-07-25 10:59:23 (Python)

```
#Price Benchmarking by Area
avg_price_area = airbnb_df.groupBy("neighbourhood_group").agg(round(avg("price"),2).alias("avg_price_area")).orderBy(col("avg_price_area").desc())
avg_price_area.show()
```

neighbourhood_group	avg_price_area
Manhattan	196.89
Brooklyn	124.45
Staten Island	114.81
Queens	99.52
Bronx	87.54

only showing top 20 rows

27

Image 16:

Databricks Notebook Project 5 - Airbnb Listings in NYC 2025-07-25 10:59:23 (Python)

```
#Listings Summary by Neighborhood and Room Type
room_type_area_summary = airbnb_df.groupBy("neighbourhood_group","room_type").agg(round(avg("price"),2).alias("avg_price_area"),round(avg("number_of_reviews"),2).alias("avg_no_of_reviews"),round(max("availability_365"),2).alias("avg_availability_365")).orderBy(col("neighbourhood_group").desc())
room_type_area_summary.show()
```

neighbourhood_group	room_type	avg_price_area	avg_no_of_reviews	avg_availability_365
Staten Island	Private room	62.29	30.16	365
Staten Island	Shared room	57.44	1.56	312
Staten Island[Entire home/apt]	Entire home/apt	173.85	33.28	365
Queens	Shared room	69.92	13.86	365
Queens	Private room	71.76	27.75	365
Queens[Entire home/apt]	Entire home/apt	147.05	28.93	365
Manhattan[Entire home/apt]	Entire home/apt	249.26	17.82	365
Manhattan	Private room	116.79	26.21	365
Manhattan	Shared room	88.98	21.4	365
Brooklyn	Private room	76.55	21.08	365
Brooklyn[Entire home/apt]	Entire home/apt	178.35	27.95	365
Brooklyn	Shared room	50.77	14.08	365
Bronx[Entire home/apt]	Entire home/apt	127.51	30.68	365
Bronx	Shared room	58.61	7.32	365
Bronx	Private room	66.89	24.97	365

28

29

Image 17:

Project 5 - Airbnb Listings in NYC 2025-07-25 10:59:23 (Python)

```
#Top Hosts by Listing Count
top_host = airbnb_df.groupBy("neighbourhood_group","host_id","host_name").agg(sum(col("calculated_host_listings_count")).alias("listing_count")).orderBy(col("listing_count").desc())
top_host.show(10)
```

neighbourhood_group	host_id	host_name	listing_count
Manhattan	219537861	Sonder (NYC)	106929
Manhattan	107434423	Blueground	53360
Manhattan	30283594	Kara	16641
Manhattan	122430511	Sonder	9216
Manhattan	16098958	Jeremy & Laura	9216
Manhattan	61391963	Corporate Housing	8281
Queens	137358866	Kazuya	8137
Manhattan	22541573	Ken	7482
Manhattan	200380610	Pranjal	4225
Manhattan	1475015	Mike	2704

only showing top 10 rows

Image 18:

Project 5 - Airbnb Listings in NYC 2025-07-25 10:59:23 (Python)

```
#Minimum Night Requirements by Area
min_night_requirement = airbnb_df.groupBy("neighbourhood_group","neighbourhood").agg(round(min("minimum_nights"),2).alias("minimum_nights"))
min_night_requirement.show()
```

neighbourhood_group	neighbourhood	minimum_nights
Staten Island	Willowbrook	4
Staten Island	Richmondtown	3
Staten Island	Rossville	3
Staten Island	Bay Terrace, Staten Island	3
Staten Island	Grant City	3
Staten Island	Houland Hook	3
Bronx	Castle Hill	2
Queens	Heponit	2
Bronx	Spuerten Duyvill	2
Bronx	Co-op City	2
Staten Island	Silver Lake	2
Brooklyn	Bergen Beach	2
Staten Island	Todt Hill	2
Staten Island	Westleigh	2
Staten Island	Lighthouse Hill	2
Queens	Glendale	1
Brooklyn	Windsor Terrace	1
Queens	Bayswater	1

only showing top 20 rows

Image 19:

Project 5 - Airbnb Listings in NYC 2025-07-25 10:59:23 (Python)

```
#High Availability Listings (> 300 Days)
high_availability = airbnb_df.groupBy("id","name","neighbourhood_group","neighbourhood").agg(sum(col("availability_365")).alias("availability")).filter(col("availability")>>0)
distinct_count = high_availability.select("id","name","neighbourhood_group","neighbourhood").distinct().count()
print("Number of high-availability listings:", distinct_count)
high_availability.show()
```

id	name	neighbourhood_group	neighbourhood	availability
1197899	Sunny Bedroom in ...	Brooklyn	Clinton Hill	318
1332471	2 BD ROOM APT. I...	Queens	Queens Village	363
1628156	A Tree Grows in ...	Brooklyn	Canarsie	317
4103970	Private room in h...	Brooklyn	Greenpoint	335
7341406	Private Spacious ...	Brooklyn	Bedford-Stuyvesant	312
9384721	North of Madison ...	Manhattan	Midtown	365
9900100	Sunny, Cozy Doubli...	Brooklyn	East Flatbush	326
11097962	Home, Sweet, Marin...!	Manhattan	East Harlem	348
11554255	PRIVATE ROOM ❤ ...	Queens	Jamaica	318
12972783	Private Room for ...	Queens	Maspeth	364
13369841	charming LFS Back...	Manhattan	Lower East Side	365
15775049	Room in charming ...	Brooklyn	Bedford-Stuyvesant	311
19829900	Modern Front Apar...	Brooklyn	Bensonhurst	325
20799932	Luxury 2 Bedroom ...	Manhattan	Midtown	339
21568652	Habitacion Privada	Queens	Ozone Park	363
23271408	Perfect shared ma...	Manhattan	Lower East Side	322
23302837	Basement studio w...	Brooklyn	Bedford-Stuyvesant	365
23549904	Lofty 2 Bedroom i...	Manhattan	Gramercy	322

only showing top 20 rows

Image 20:

Project 5 - Airbnb Listings in NYC 2025-07-25 10:59:23 (Python)

```
#host Performance Summary
host_performance_summary = airbnb_df.groupBy("host_id","host_name","neighbourhood_group").agg(sum("calculated_host_listings_count").alias("listing_count")).orderBy(col("listing_count").desc())
host_performance_summary.show(50)
```

host_id	host_name	neighbourhood_group	listing_count
131647128	Emily	Manhattan	625
230192510	Zach	Brooklyn	625
137358066	Kazuya	Brooklyn	618
242362235	Yuval	Queens	529
3191545	Kylie	Manhattan	529
221206420	Adam	Manhattan	529
9864136	Anthony	Brooklyn	494
187434423	Blueground	Brooklyn	464
16437254	Benjamin	Brooklyn	441
252604696	Erin	Manhattan	400
48146336	Trina	Manhattan	400
7245581	Michael	Manhattan	361
26377263	Stat!	Queens	344
95459395	Bluebird	Manhattan	324
134184451	Hillside Hotel	Queens	324
213781715	Anting	Manhattan	297
216235179	Nina	Brooklyn	289
177174475	Alberto	Manhattan	289

only showing top 50 rows

Image 21:

databricks

Project 5 - Airbnb Listings in NYC 2025-07-25 10:59:23 (Python)

Import notebook

```
# Writing in s3 Gold layer
avg_price_room_type.write \
    .format("csv") \
    .mode("overwrite") \
    .option("header", "true") \
    .option("multiline", True) \
    .option("escape", "\\") \
    .option("quote", "") \
    .save("s3://priya-project-5-2507/gold/avg_price_room_type/")

avg_review_area.write \
    .format("csv") \
    .mode("overwrite") \
    .option("header", "true") \
    .option("multiline", True) \
    .option("escape", "\\") \
    .option("quote", "") \
    .save("s3://priya-project-5-2507/gold/avg_review_area/")

avg_review_neighbourhood.write \
    .format("csv") \
    .mode("overwrite") \
    .option("header", "true") \
    .option("multiline", True) \
    .option("escape", "\\") \
    .option("quote", "") \
    .save("s3://priya-project-5-2507/gold/avg_review_neighbourhood/")

avg_price_area.write \
    .format("csv") \
    .mode("overwrite") \
    .option("header", "true") \
    .option("multiline", True) \
    .option("escape", "\\") \
    .option("quote", "") \
    .save("s3://priya-project-5-2507/gold/avg_price_area/")
```

34

Image 22:

databricks

Project 5 - Airbnb Listings in NYC 2025-07-25 10:59:23 (Python)

Import notebook

```
avg_price_area.write \
    .format("csv") \
    .mode("overwrite") \
    .option("header", "true") \
    .option("multiline", True) \
    .option("escape", "\\") \
    .option("quote", "") \
    .save("s3://priya-project-5-2507/gold/avg_price_area/*")

room_type_area_summary.write \
    .format("csv") \
    .mode("overwrite") \
    .option("header", "true") \
    .option("multiline", True) \
    .option("escape", "\\") \
    .option("quote", "") \
    .save("s3://priya-project-5-2507/gold/room_type_area_summary/*")

top_host.write \
    .format("csv") \
    .mode("overwrite") \
    .option("header", "true") \
    .option("multiline", True) \
    .option("escape", "\\") \
    .option("quote", "") \
    .save("s3://priya-project-5-2507/gold/top_host/*")

min_right_requirement.write \
    .format("csv") \
    .mode("overwrite") \
    .option("header", "true") \
    .option("multiline", True) \
    .option("escape", "\\") \
    .option("quote", "") \
    .save("s3://priya-project-5-2507/gold/min_right_requirement/*")
```

Image 23:

```

.option("quote", "\"") \
.save("s3://priya-project-5-2507/gold/top_host/")
min_night_requirement.write \
.format("csv") \
.mode("overwrite") \
.option("header", "true") \
.option("multiline", True) \
.option("escape", "\\") \
.option("quote", "\\") \
.save("s3://priya-project-5-2507/gold/min_night_requirement/")
high_availability.write \
.format("csv") \
.mode("overwrite") \
.option("header", "true") \
.option("multiline", True) \
.option("escape", "\\") \
.option("quote", "\\") \
.save("s3://priya-project-5-2507/gold/high_availability/")
host_performance_summary.write \
.format("csv") \
.mode("overwrite") \
.option("header", "true") \
.option("multiline", True) \
.option("escape", "\\") \
.option("quote", "\\") \
.save("s3://priya-project-5-2507/gold/host_performance_summary/")

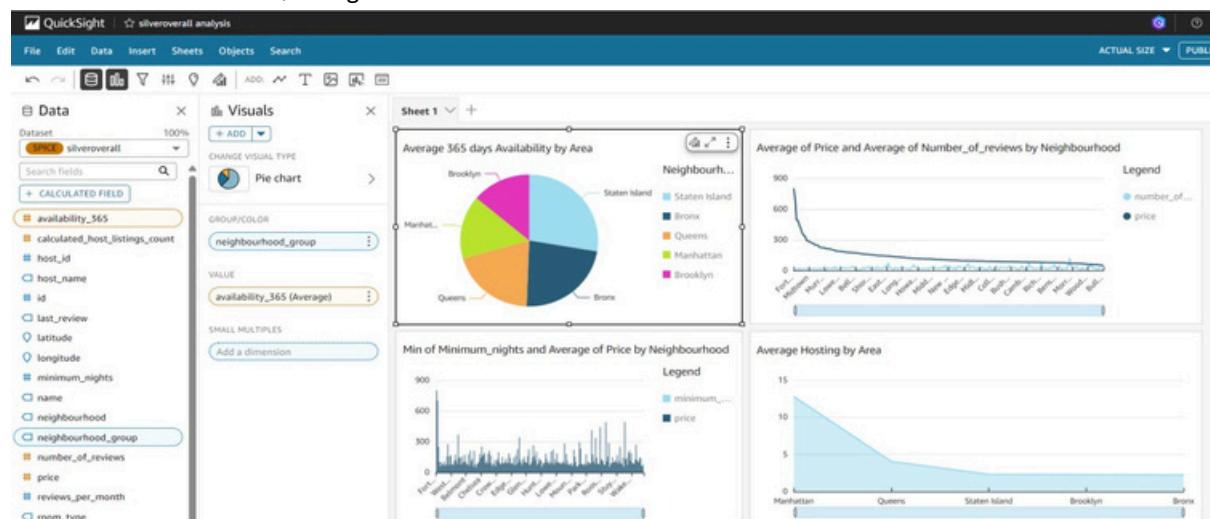
```

Output:

The screenshot shows the 'gold' folder in an Amazon S3 bucket. The folder contains several sub-folders: 'avg_price_area/', 'avg_price_room_type/', 'avg_review_area/', 'avg_review_neighbourhood/', 'high_availability/', 'host_performance_summary/', and 'min_night_requirement/'. The 'Actions' dropdown menu is open, showing options like 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', and 'Upload'.

Visualization

Visualization is done in QuickSight



Appendix:

Databricks link: Tested the Transformation and Cleaning process for Raw is done using PySpark language activity in Databricks platform.

<https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/1230577161393971/3859233799733772/6414225150628508/latest.html>