

Building a Scalable Data Pipeline for Airbnb Listings Analytics in NYC

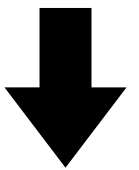
Bronze Layer

Silver Layer

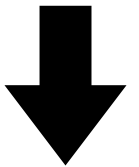
Gold Layer



AWS Glue



AWS Athena



AWS QuickSight



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

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

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

Perform all Aggregate operations 'and store the data in S3

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:

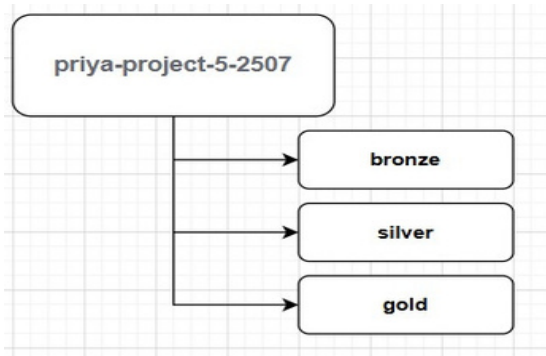
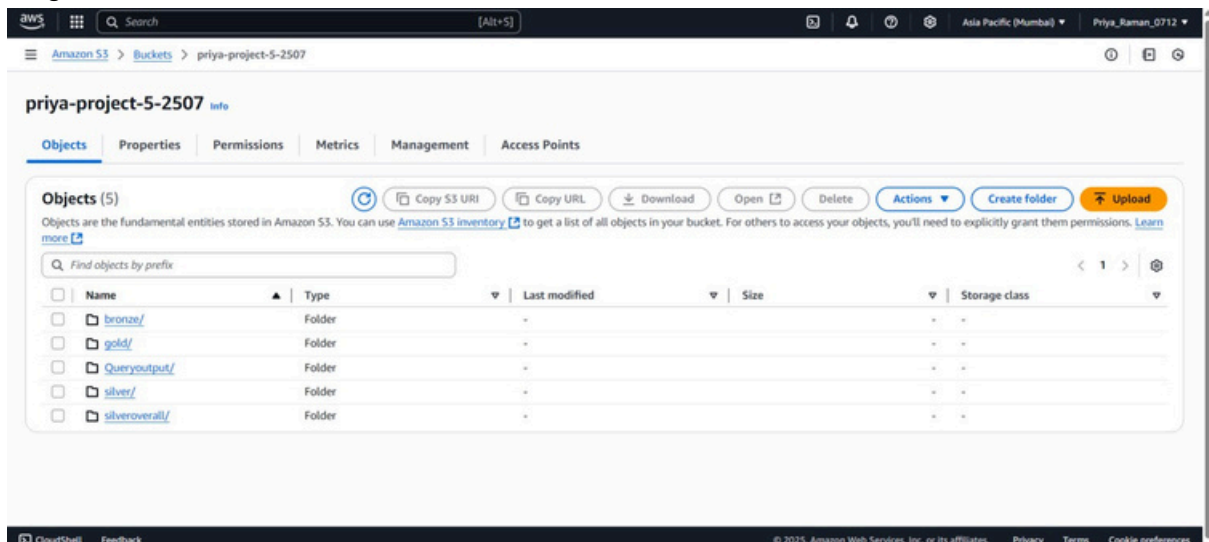


Image 1:



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",
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::priya-project-5-2507 ",

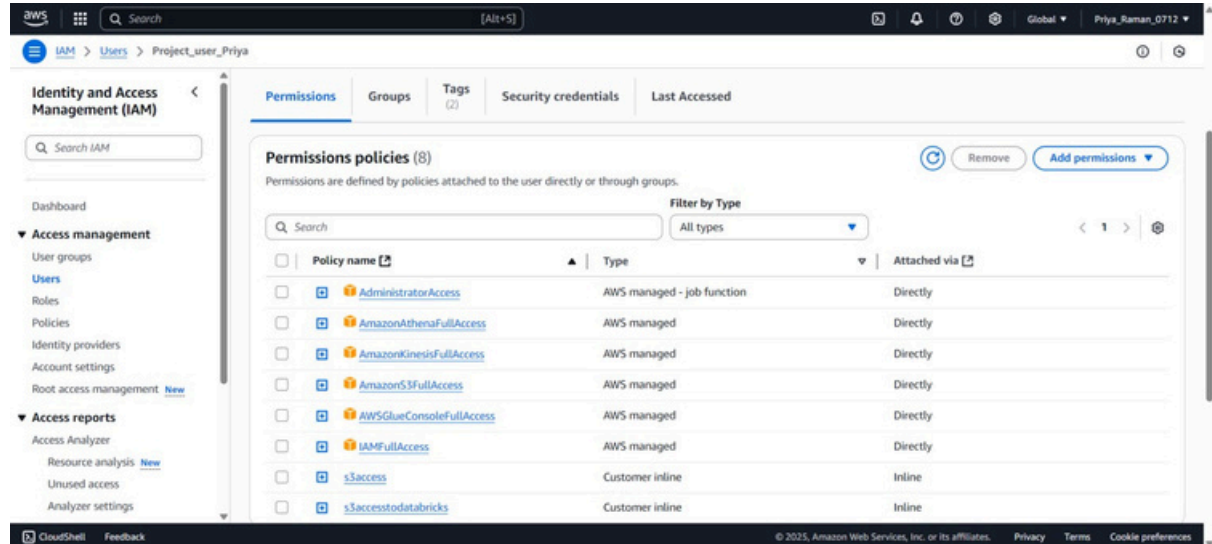
```

```

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

```

Image 2:



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"

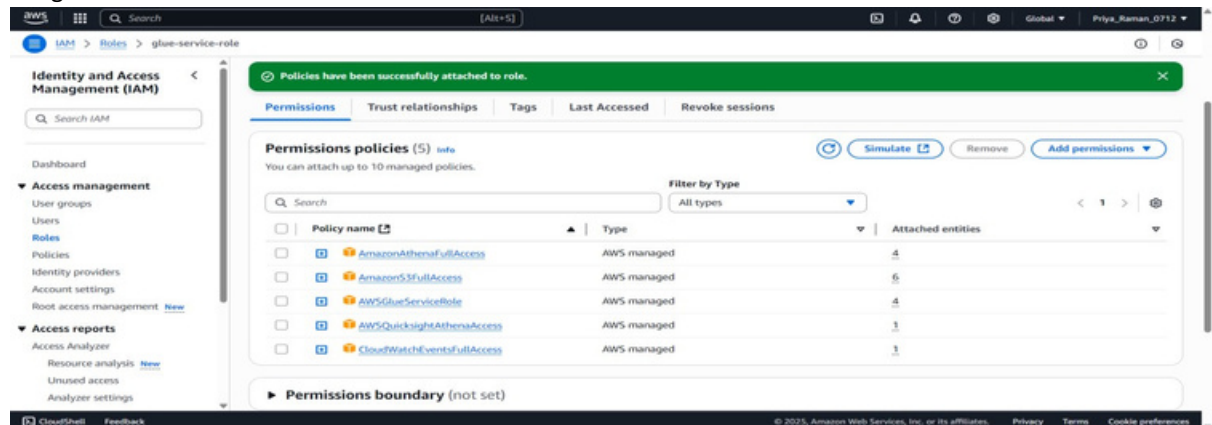
Click Next

Add the following Permissions

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

Provide Role Name as "glue_service_role" [Refer Image 3]

Image 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:

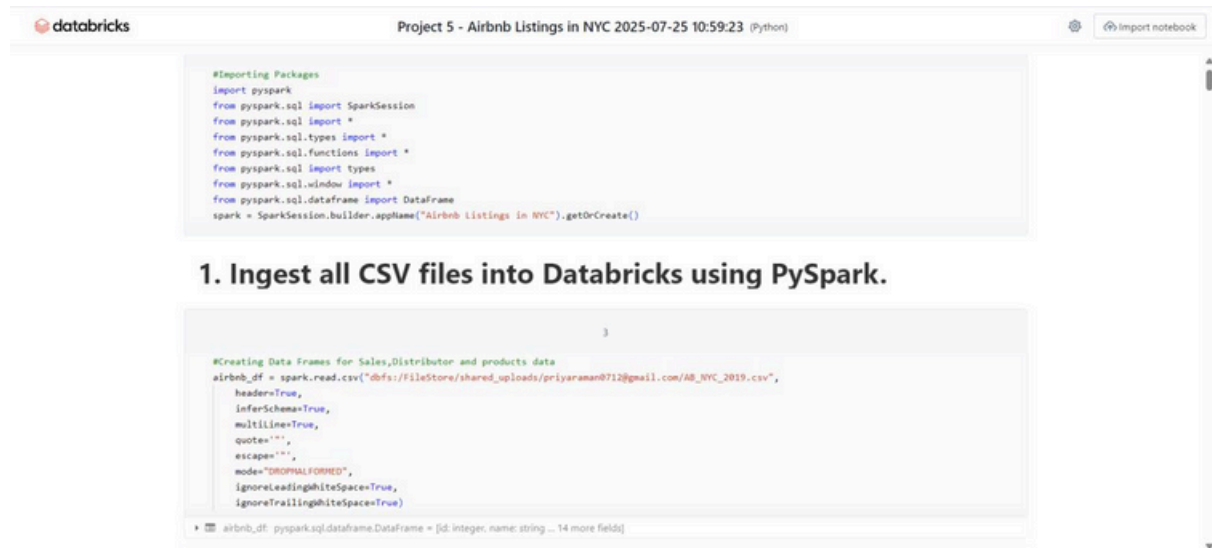


Image 5:

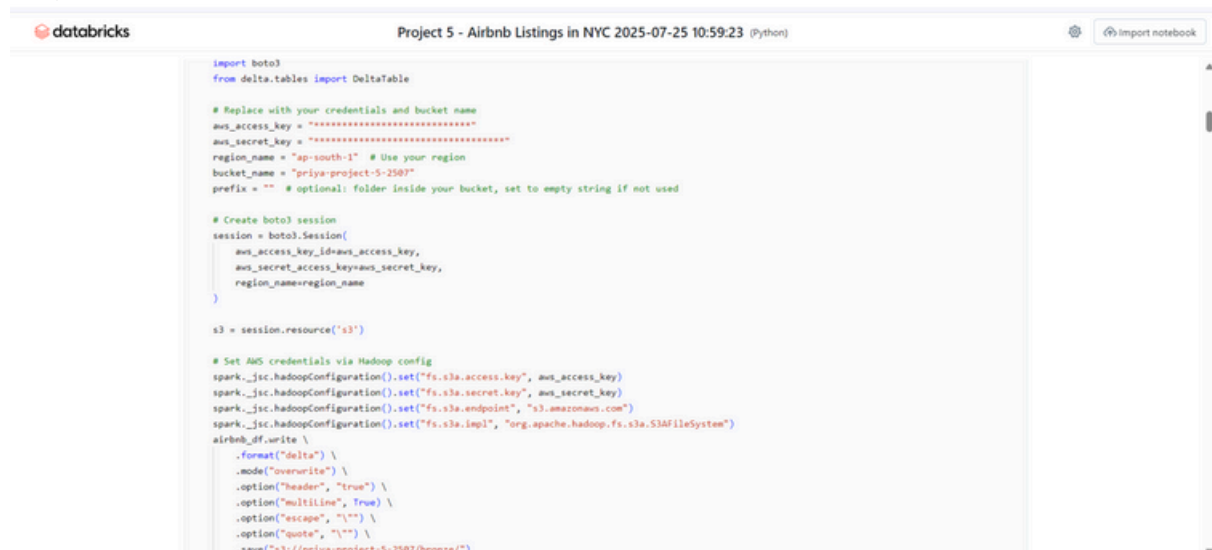
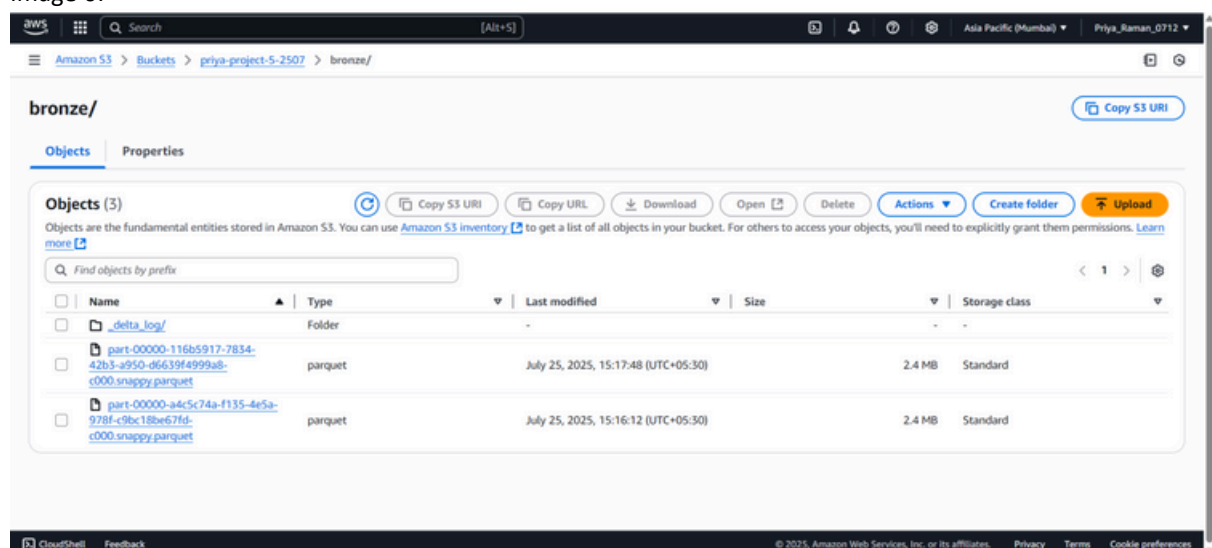


Image 6:



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:

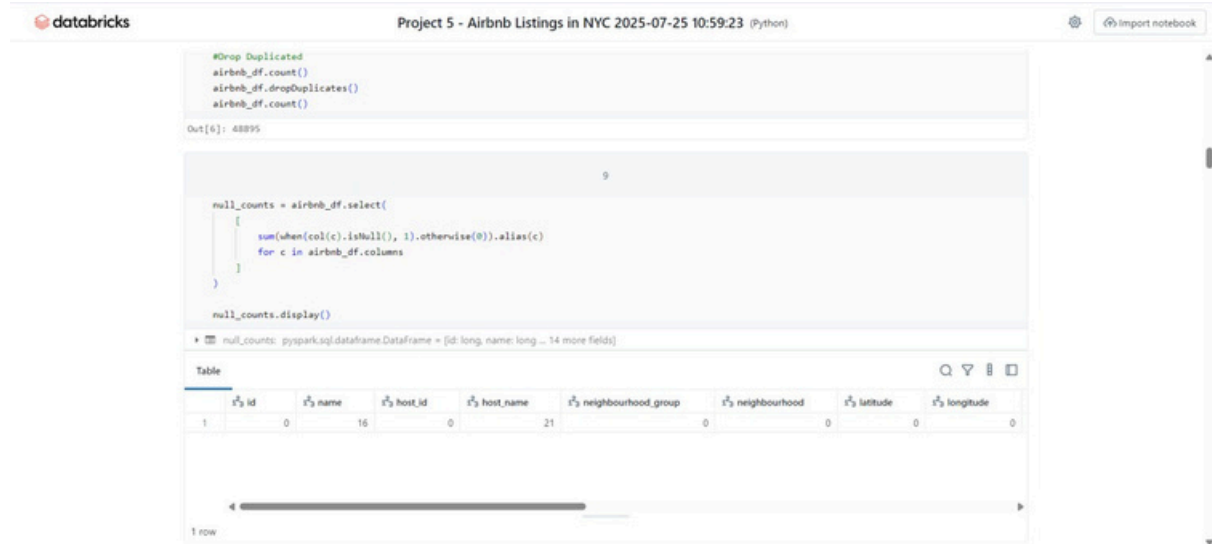


Image 8:

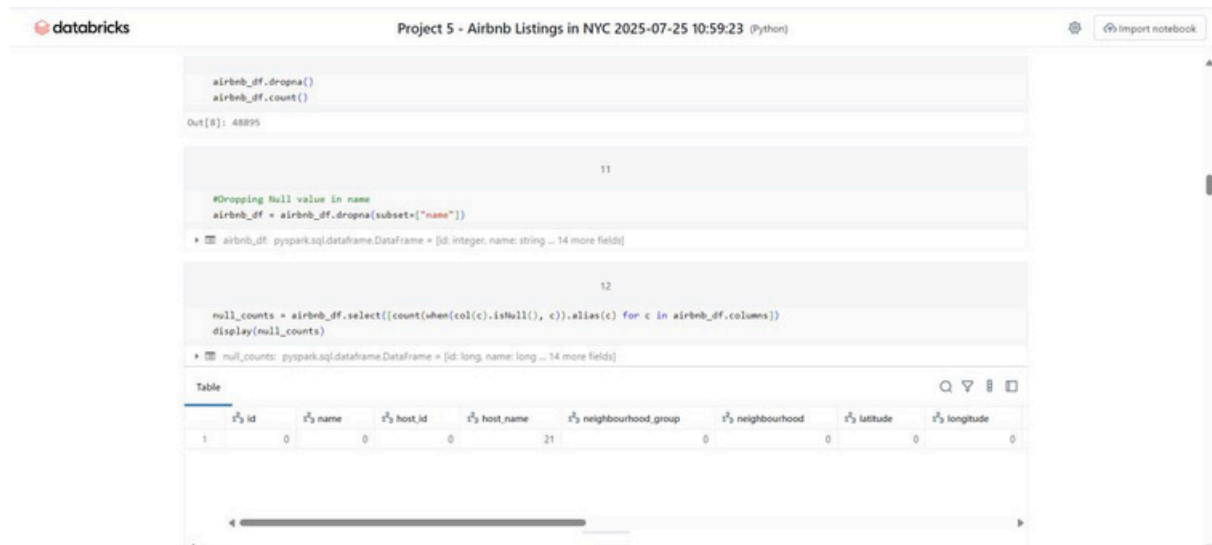


Image 9:

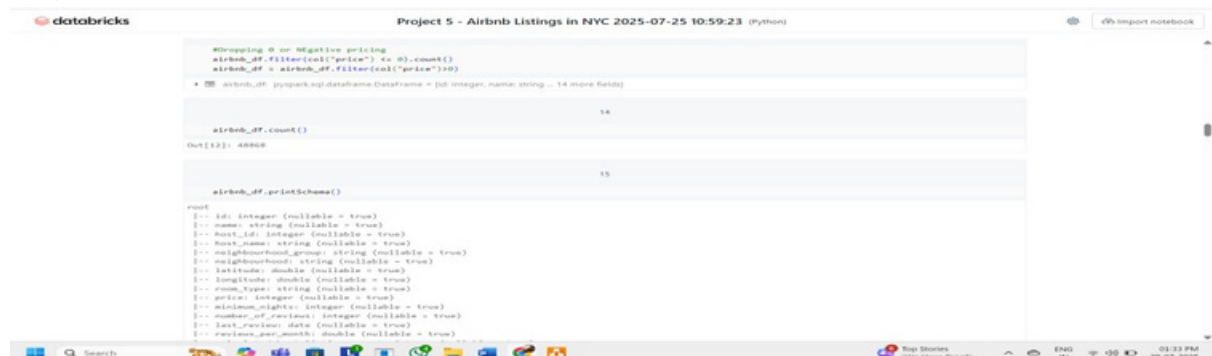


Image 10:

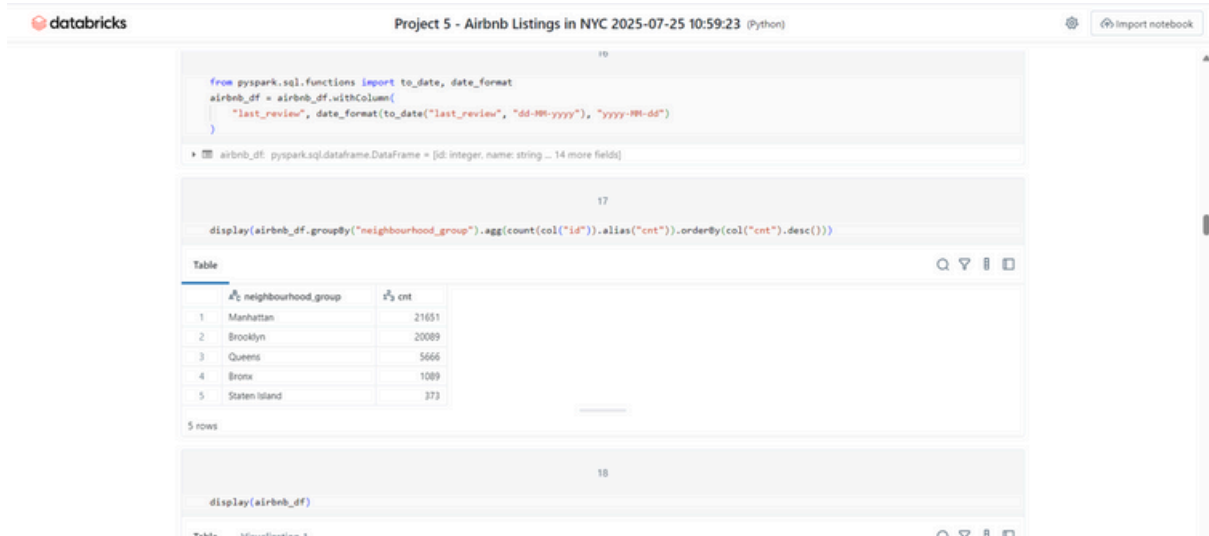
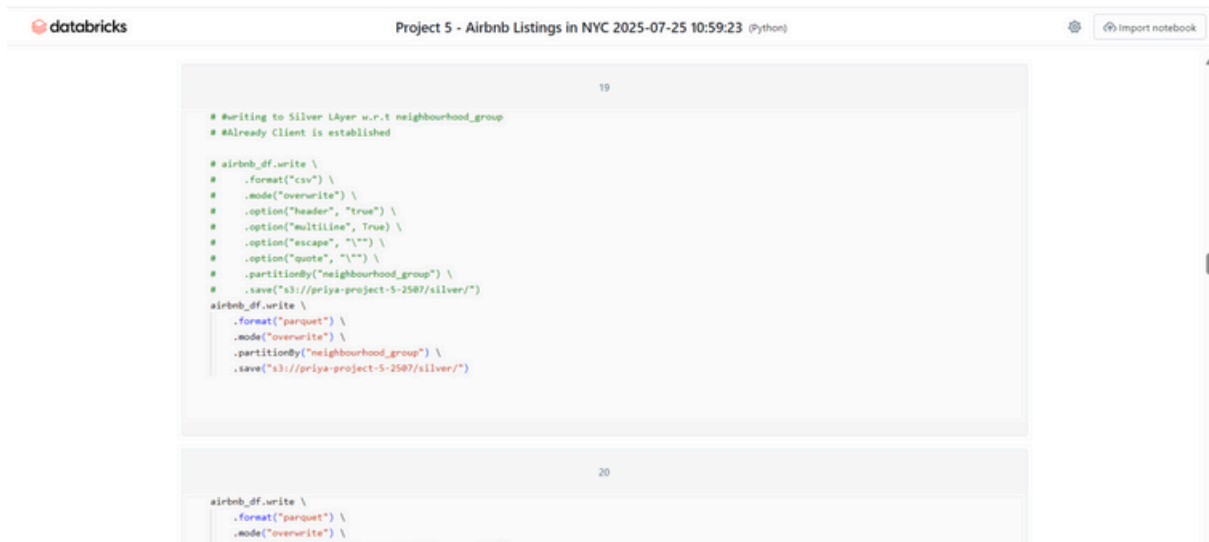
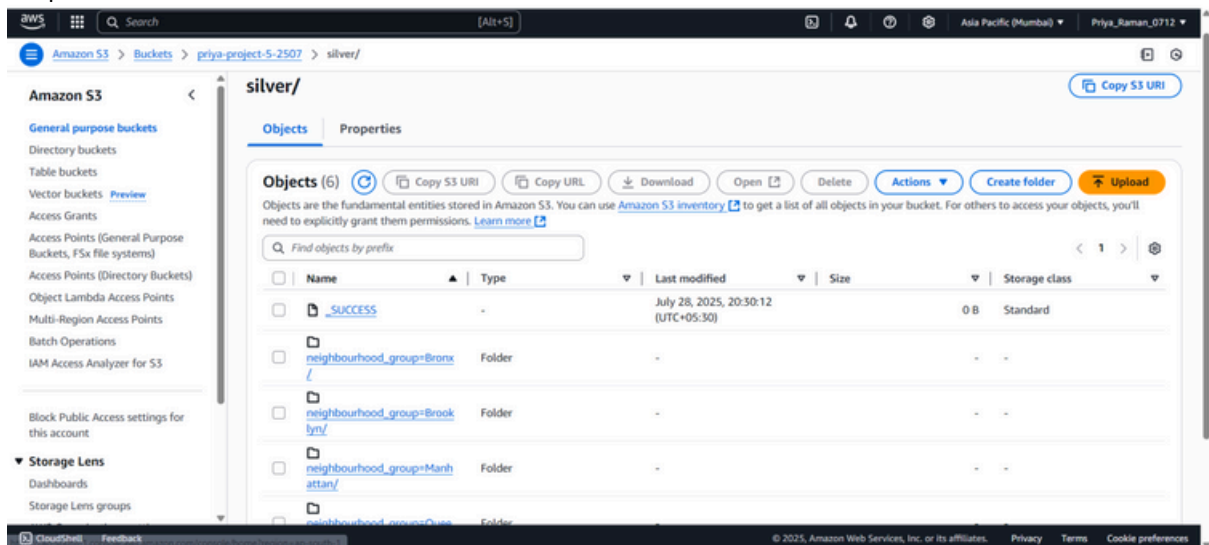


Image 11:



Output:



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 Buckert in Parquet format [Refer Image 21,22,23]

Image 12:

The screenshot shows a Databricks notebook interface. At the top, it says "Project 5 - Airbnb Listings in NYC 2025-07-25 10:59:23 (Python)". Below the header, there is a schema definition for a table with columns: neighbourhood_group, neighbourhood, latitude, longitude, room_type, price, minimum_nights, number_of_reviews, last_review, reviews_per_month, calculated_host_listings_count, and availability_365. Below the schema, a SQL query is executed: `avg_price_room_type = airbnb_df.groupBy("room_type").agg(round(avg(col("price")),2).alias("avg_price")).orderBy(col("avg_price").desc())`. The result is displayed as a table with two columns: room_type and avg_price. The data rows are: Entire home/apt (211.8), Private room (89.81), and Shared room (70.2).

```
-- 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)

24

#Average prices by room_type
avg_price_room_type = airbnb_df.groupBy("room_type").agg(round(avg(col("price")),2).alias("avg_price")).orderBy(col("avg_price").desc())
avg_price_room_type.show()

+-----+
| room_type|avg_price|
+-----+
| Entire home/apt| 211.8|
| Private room| 89.81|
| Shared room| 70.2|
+-----+
```

Image 13:

The screenshot shows a Databricks notebook interface. At the top, it says "Project 5 - Airbnb Listings in NYC 2025-07-25 10:59:23 (Python)". Below the header, a SQL query is executed: `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())`. The result is displayed as a table with two columns: neighbourhood_group and avg_reviews_per_month. The data rows are: Queens (1.94), Staten Island (1.87), Bronx (1.84), Brooklyn (1.28), and Manhattan (1.27). Below this, another SQL query is shown: `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())`.

```
25

#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()

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

26

#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())
```

Image 14:

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

26

```
#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()
```

avg_review_neighbourhood: pyspark.sql.dataframe.DataFrame = [neighbourhood: string, avg_number_of_reviews: double]

Richmondton	79.0
Eltingville	76.0
Mount Eden	70.0
Springfield Gardens	69.09
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

27

Image 15:

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

only showing top 20 rows

27

```
#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()
```

avg_price_area: pyspark.sql.dataframe.DataFrame = [neighbourhood_group: string, avg_price_area: double]

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

Image 16:

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

28

```
#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()
```

room_type_area_summary: pyspark.sql.dataframe.DataFrame = [neighbourhood_group: string, room_type: string -- 3 more fields]

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	173.85	33.28	365
Queens	Shared room	69.82	13.86	365
Queens	Private room	71.76	27.75	365
Queens	Entire home/apt	147.05	28.93	365
Manhattan	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	178.35	27.95	365
Brooklyn	Shared room	50.77	14.08	365
Bronx	Entire home/apt	127.51	38.68	365
Bronx	Shared room	58.61	7.32	365
Bronx	Private room	66.89	24.97	365

29

Image 17:

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

29

```
#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)
```

top_host: pyspark.sql.dataframe.DataFrame = [neighbourhood_group: string, host_id: integer ... 2 more fields]

neighbourhood_group	host_id	host_name	listing_count
Manhattan	219517861	Sonder (NYC)	106929
Manhattan	107434423	Blueground	53360
Manhattan	30283594	Kara	14641
Manhattan	12243851	Sonder	9216
Manhattan	16098958	Jeremy & Laura	9216
Manhattan	61391963	Corporate Housing	8281
Queens	137358866	Kazuya	8137
Manhattan	22543573	Ken	7482
Manhattan	280380610	Franjoli	4225
Manhattan	1475015	Hike	2704

only showing top 10 rows

30

Image 18:

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

30

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

min_night_requirement: pyspark.sql.dataframe.DataFrame = [neighbourhood_group: string, neighbourhood: string ... 1 more field]

neighbourhood_group	neighbourhood	minimum_nights
Staten Island	Willowbrook	4
Staten Island	Richmondton	3
Staten Island	Rossville	3
Staten Island	Bay Terrace, Stat...	3
Staten Island	Grant City	3
Staten Island	Houland Hook	3
Bronx	Castle Hill	2
Queens	Hopsonit	2
Bronx	Spyuten Duyvil	2
Bronx	Co-op City	2
Staten Island	Silver Lake	2
Brooklyn	Bergen Beach	2
Staten Island	Todd Hill	2
Staten Island	Westerleigh	2
Staten Island	Lighthouse Hill	2
Queens	Glendale	1
Brooklyn	Windsor Terrace	1
Queens	Bayswater	1

only showing top 20 rows

31

Image 19:

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

31

```
#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")>300)
distinct_count = high_availability.select("id", "name", "neighbourhood_group", "neighbourhood").distinct().count()
print("Number of high-availability listings:", distinct_count)
high_availability.show()
```

high_availability: pyspark.sql.dataframe.DataFrame = [id: integer, name: string ... 3 more fields]

id	name	neighbourhood_group	neighbourhood	availability
1197899	Sunny Bedroom in ...	Brooklyn	Clinton Hill	318
1332473	2 BED ROOM APT. I...	Queens	Queens Village	363
1628156	A Tree Grows in B...	Brooklyn	Canarsie	317
4103970	Private room in B...	Brooklyn	Greenpoint	335
7341406	Private Spacious ...	Brooklyn	Bedford-Stuyvesant	312
9384723	North of Madison ...	Manhattan	Midtown	365
9900100	Sunny, Cozy Doubl...	Brooklyn	East Flatbush	326
11097962	Home, Sweet, Harl...	Manhattan	East Harlem	348
11554255	♥ PRIVATE ROOM ♥ ...	Queens	Jamaica	318
12972783	Private Room for ...	Queens	Maspeth	364
13360841	Charming LES Back...	Manhattan	Lower East Side	365
15775049	Room in charming ...	Brooklyn	Bedford-Stuyvesant	311
19829900	Modern Front Apar...	Brooklyn	Bensonhurst	325
20799032	Luxury 2 Bedroom ...	Manhattan	Midtown	339
21568652	Habitacion Privada	Queens	Ozone Park	363
23271408	Perfect shared ma...	Manhattan	Lower East Side	322
23382837	Basement studio w...	Brooklyn	Bedford-Stuyvesant	365
23549904	Lofty 2 Bedroom i...	Manhattan	Gramercy	322

only showing top 20 rows

32

Image 20:

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

33

```
#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_performance_summary: pyspark.sql.dataframe.DataFrame = [host_id: integer, host_name: string ... 2 more fields]

[131647128]	Emily	Manhattan	625
[230192510]	Zach	Brooklyn	625
[137358866]	Kazuya	Brooklyn	618
[242962235]	Yuval	Queens	529
[3191545]	Kyle	Manhattan	529
[221200420]	Adam	Manhattan	529
[9864136]	Anthony	Brooklyn	494
[187434423]	Blueground	Brooklyn	464
[6437254]	Benjamin	Brooklyn	441
[252684696]	Erin	Manhattan	408
[48146336]	Irina	Manhattan	408
[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

34

Image 21:

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

34

```
# 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/")
```

35

Image 22:

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

35

```
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_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/")
```

36

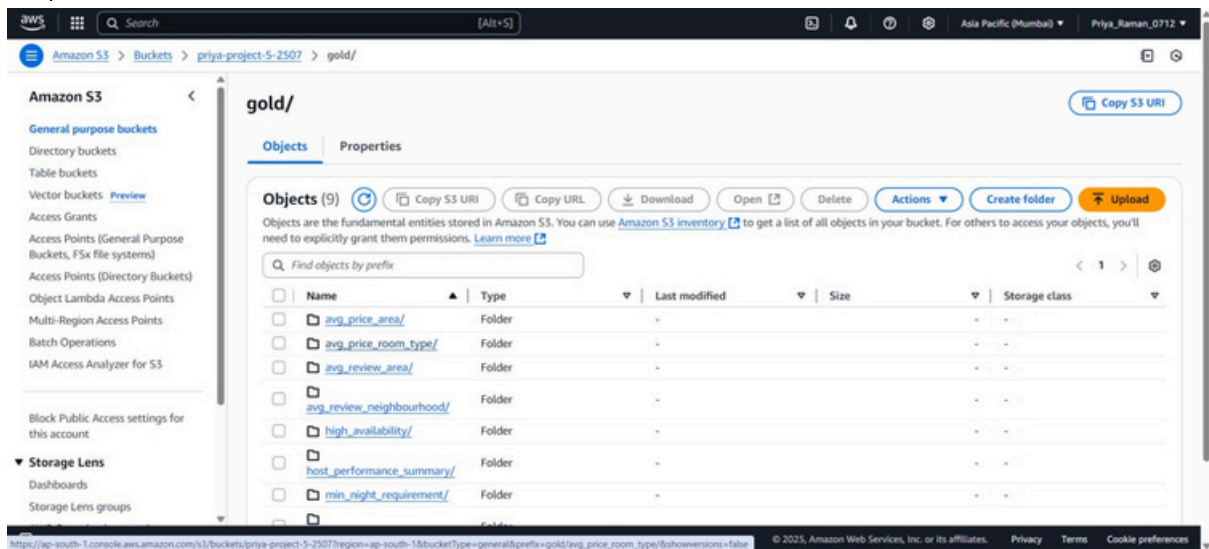
Image 23:

```

.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/")

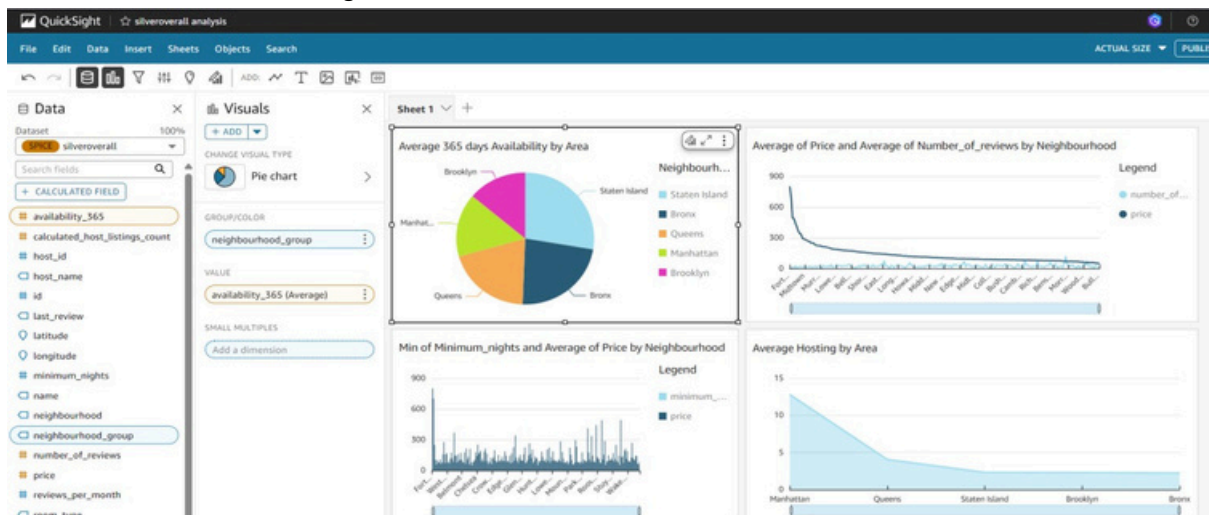
```

Output:



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>