

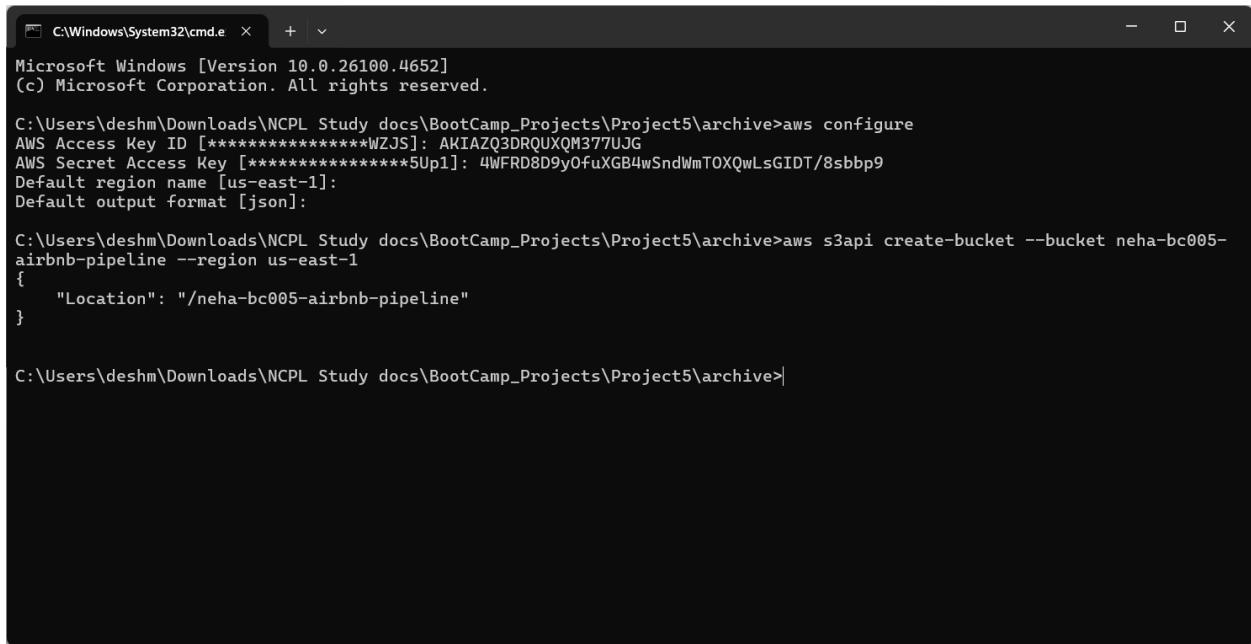
Building a Scalable Data Pipeline for Airbnb Listings Analytics in NYC

[Airbnb Listings Data Pipeline- Business-Focused Data Pipeline for Hospitality Insights]

[Bootcamp- AWS Data Engineering Project- 5]

1. Create an S3 Bucket Using CLI

Command: aws s3api create-bucket --bucket neha-bc005-airbnb-pipeline --region us-east-1



```
C:\Windows\System32\cmd.exe + - 
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

C:\Users\deshm\Downloads\NCPL Study docs\BootCamp_Projects\Project5\archive>aws configure
AWS Access Key ID [*****WZJS]: AKIAZQ3DRQUXQM377UJG
AWS Secret Access Key [*****5Up1]: 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
Default region name [us-east-1]:
Default output format [json]

C:\Users\deshm\Downloads\NCPL Study docs\BootCamp_Projects\Project5\archive>aws s3api create-bucket --bucket neha-bc005-airbnb-pipeline --region us-east-1
{
    "Location": "/neha-bc005-airbnb-pipeline"
}

C:\Users\deshm\Downloads\NCPL Study docs\BootCamp_Projects\Project5\archive>
```

2. Configure S3 in AWS

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with navigation links like 'Amazon S3', 'General purpose buckets', 'Storage Lens', and 'CloudShell'. The main area is titled 'General purpose buckets' and shows one bucket named 'neha-bc005-airbnb-pipeline'. Below the bucket list is an 'Account snapshot' section with a 'View dashboard' button and a note about Storage Lens providing visibility into storage usage and activity trends. Another section, 'External access summary - new', is also visible.

3. Create sub folders in S3

Commands:

```
aws s3api put-object --bucket neha-bc005-airbnb-pipeline --key bronze/
aws s3api put-object --bucket neha-bc005-airbnb-pipeline --key silver/
aws s3api put-object --bucket neha-bc005-airbnb-pipeline --key gold/
```

The screenshot shows a Windows Command Prompt window with three lines of command history. The first line creates a folder named 'bronze' in the S3 bucket. The second line creates a folder named 'silver'. The third line creates a folder named 'gold'. Each command includes the full path from the root of the bucket.

```
C:\Windows\System32\cmd.exe > aws s3api put-object --bucket neha-bc005-airbnb-pipeline --key bronze/
{
  "ETag": "\"d41d8cd98f00b204e9800998ecf8427e\"",
  "ChecksumCRC64NVME": "AAAAAAA=",
  "ChecksumType": "FULL_OBJECT",
  "ServerSideEncryption": "AES256"
}

C:\Users\deshm\Downloads\NCPL Study docs\BootCamp_Projects\Project5\archive>aws s3api put-object --bucket neha-bc005-airbnb-pipeline --key silver/
{
  "ETag": "\"d41d8cd98f00b204e9800998ecf8427e\"",
  "ChecksumCRC64NVME": "AAAAAAA=",
  "ChecksumType": "FULL_OBJECT",
  "ServerSideEncryption": "AES256"
}

C:\Users\deshm\Downloads\NCPL Study docs\BootCamp_Projects\Project5\archive>aws s3api put-object --bucket neha-bc005-airbnb-pipeline --key gold/
{
  "ETag": "\"d41d8cd98f00b204e9800998ecf8427e\"",
  "ChecksumCRC64NVME": "AAAAAAA=",
  "ChecksumType": "FULL_OBJECT",
  "ServerSideEncryption": "AES256"
}
```

4. Configure in S3

The screenshot shows the AWS S3 console with the bucket 'neha-bc005-airbnb-pipeline'. The left sidebar has sections for General purpose buckets, Storage Lens, and IAM Access Analyzer for S3. The main area shows 'Objects (3)'. There are three entries: 'bronze/' (Folder), 'gold/' (Folder), and 'silver/' (Folder). Each entry has a small icon, a name, a type (Folder), and a last modified date.

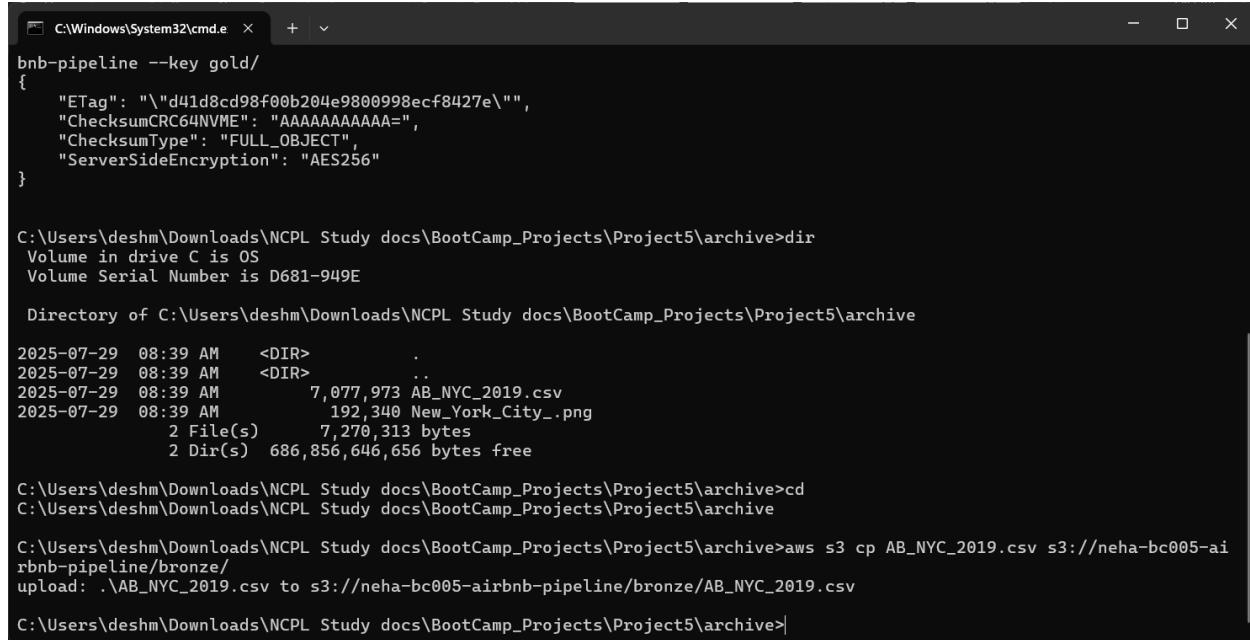
5. Create IAM Role for Glue

The screenshot shows the AWS IAM console with the role 'glue_service_role'. The left sidebar has sections for Identity and Access Management (IAM), Access management, and Access reports. The main area shows 'Permissions policies (5) Info'. It lists five managed policies: 'AmazonAthenaFullAccess', 'AmazonS3FullAccess', 'AWSGlueServiceRole', 'AWSQuickSightAthenaAccess', and 'CloudWatchEventsFullAccess'. Each policy is listed with its name, type (AWS managed), and the number of attached entities.

Bronze Layer

- Upload airbnb_nyc.csv to S3 (Bronze Layer) through CLI

Command: aws s3 cp AB_NYC_2019.csv s3://neha-bc005-airbnb-pipeline/bronze/



```

C:\Windows\System32\cmd.exe + -
bnb-pipeline --key gold/
{
    "ETag": "\"d41d8cd98f00b204e9800998ecf8427e\"",
    "ChecksumCRC64NVME": "AAAAAAAAAA=",
    "ChecksumType": "FULL_OBJECT",
    "ServerSideEncryption": "AES256"
}

C:\Users\deshm\Downloads\NCPL Study docs\BootCamp_Projects\Project5\archive>dir
 Volume in drive C is OS
 Volume Serial Number is D681-949E

Directory of C:\Users\deshm\Downloads\NCPL Study docs\BootCamp_Projects\Project5\archive

2025-07-29  08:39 AM      <DIR>        .
2025-07-29  08:39 AM      <DIR>        ..
2025-07-29  08:39 AM           7,077,973 AB_NYC_2019.csv
2025-07-29  08:39 AM           192,340 New_York_City_.png
2 File(s)       7,270,313 bytes
2 Dir(s)   686,856,646,656 bytes free

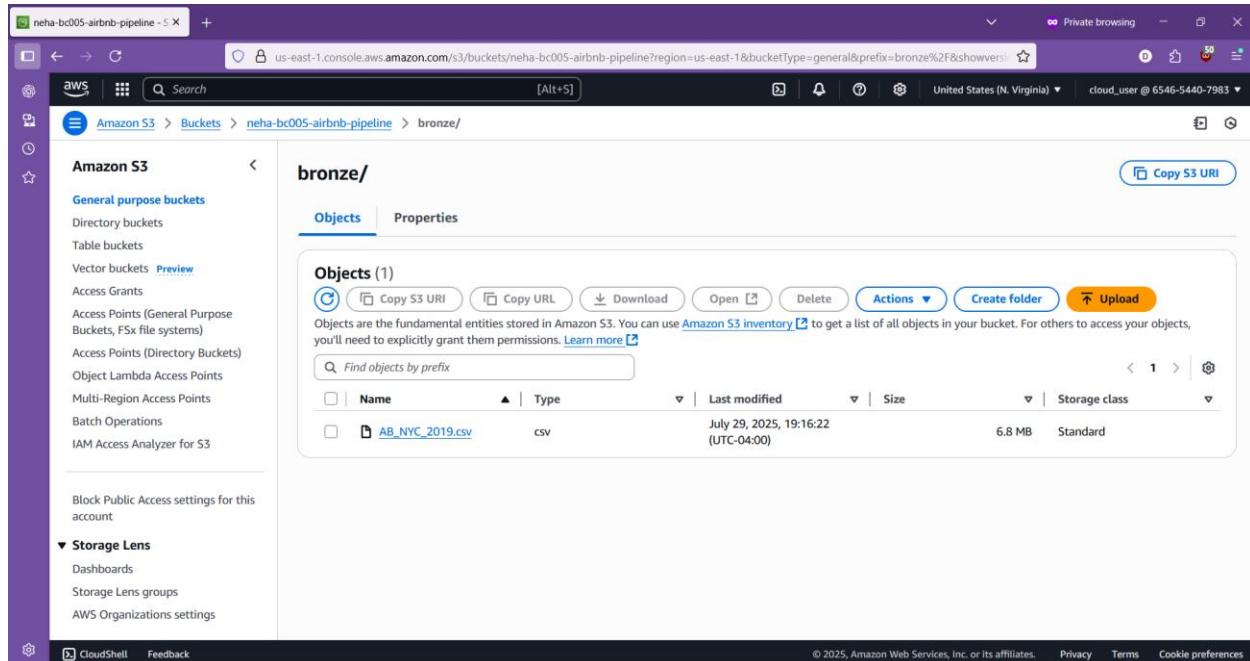
C:\Users\deshm\Downloads\NCPL Study docs\BootCamp_Projects\Project5\archive>cd
C:\Users\deshm\Downloads\NCPL Study docs\BootCamp_Projects\Project5\archive

C:\Users\deshm\Downloads\NCPL Study docs\BootCamp_Projects\Project5\archive>aws s3 cp AB_NYC_2019.csv s3://neha-bc005-airbnb-pipeline/bronze/
upload: ./AB_NYC_2019.csv to s3://neha-bc005-airbnb-pipeline/bronze/AB_NYC_2019.csv

C:\Users\deshm\Downloads\NCPL Study docs\BootCamp_Projects\Project5\archive>

```

- Configure in S3



The screenshot shows the AWS S3 console interface. The left sidebar navigation bar includes links for General purpose buckets, Storage Lens, and CloudShell. The main content area displays the 'bronze/' folder under the 'neha-bc005-airbnb-pipeline' bucket. The 'Objects' tab is selected, showing one object: 'AB_NYC_2019.csv'. The object details are as follows:

Name	Type	Last modified	Size	Storage class
AB_NYC_2019.csv	csv	July 29, 2025, 19:16:22 (UTC-04:00)	6.8 MB	Standard

Silver Layer - Clean Airbnb Data with AWS Glue

1. Create a Glue Crawler for Bronze Data

Crawler properties

- Name:** bronze_airbnb_crawler
- IAM role:** glue_service_role
- Database:** airbnb_pipeline_db
- Description:** -
- Security configuration:** -
- Lake Formation configuration:** -
- State:** READY
- Table prefix:** -

Maximum table threshold: -

Advanced settings: [button]

Crawler runs [button] | Schedule | Data sources | Classifiers | Tags

Crawler runs (0)
The list of crawler runs for this crawler.

Filter data | Filter by a date and time range | Start time (UTC) ▲ End time (UTC) ▼ Current/last duration ▼ Status ▼ DPU hours ▼ Table changes ▼

Crawler properties

- Name:** bronze_airbnb_crawler
- IAM role:** glue_service_role
- Database:** airbnb_pipeline_db
- Description:** -
- Security configuration:** -
- Lake Formation configuration:** -
- State:** READY
- Table prefix:** -

Maximum table threshold: -

Advanced settings: [button]

Crawler runs (1 -)
The list of crawler runs for this crawler.

Filter data | Filter by a date and time range | Start time (UTC) ▲ End time (UTC) ▼ Current/last duration ▼ Status ▼ DPU hours ▼ Table changes ▼

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
July 29, 2025 at 23:26:58	-	10 s	Running	-	-

Crawler properties

- Name:** bronze_airbnb_crawler
- IAM role:** glue_service_role
- Database:** airbnb_pipeline_db
- Description:** -
- Security configuration:** -
- Lake Formation configuration:** -
- Table prefix:** -
- State:** READY

Maximum table threshold: -

Advanced settings:

Crawler runs (1)

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
July 29, 2025 at 23:26:58	July 29, 2025 at 23:28:05	01 min 07 s	Completed	-	-

Table created:

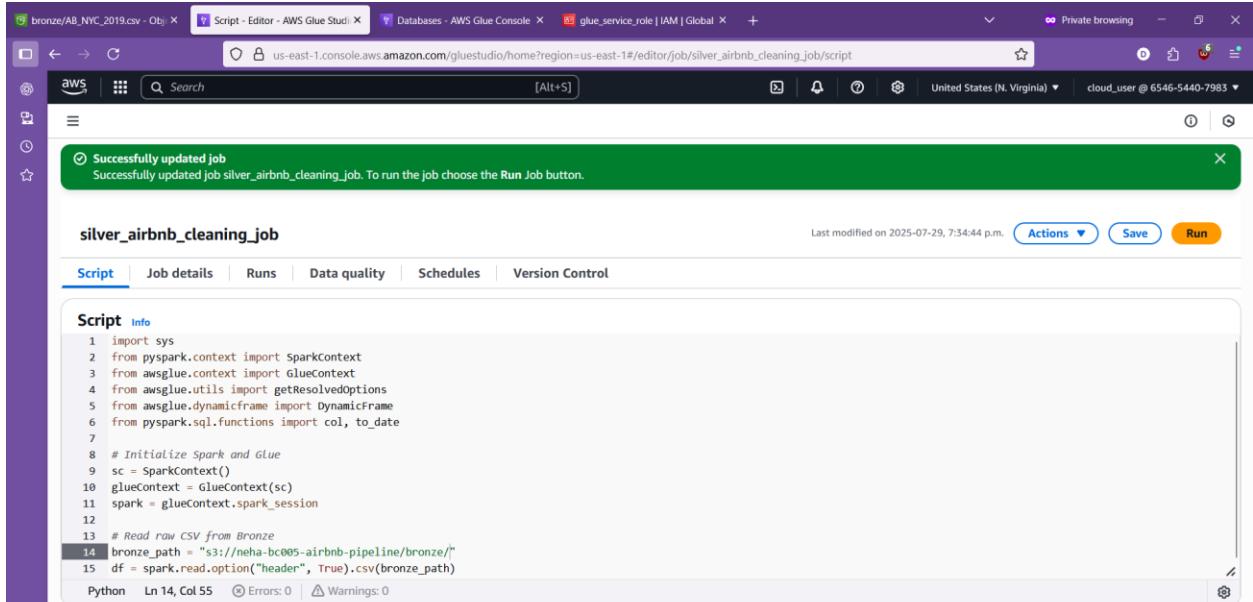
Database properties

- Name:** airbnb_pipeline_db
- Description:** -
- Location:** -
- Created on (UTC):** July 29, 2025 at 23:26:18

Tables (1)

Name	Database	Location	Classification	Deprecated	View data	Data quality	Column stats...
bronze	airbnb_pipeline_dt	s3://neha-bc005-a	CSV	-	Table data	View data quality	View statistics

2. Create a Glue Job to Clean and Enrich Data



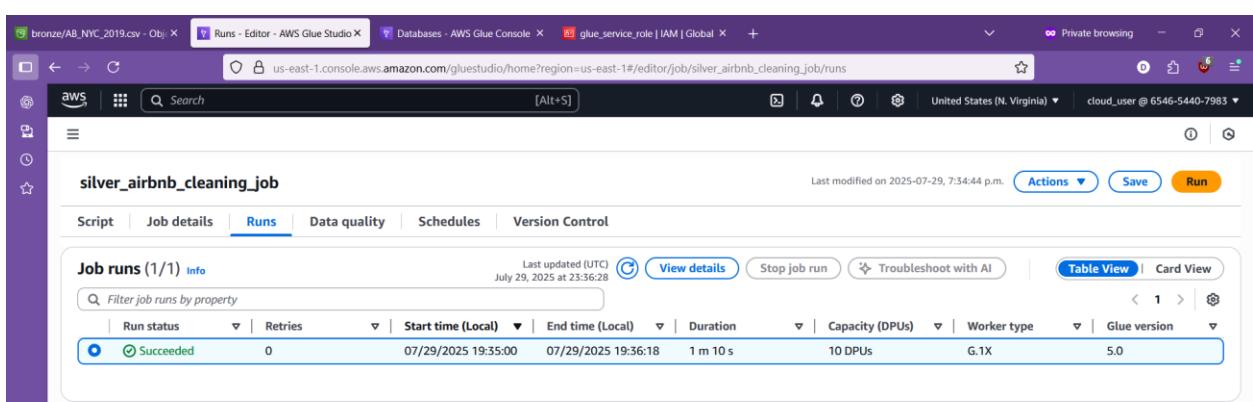
The screenshot shows the AWS Glue Studio interface with a success message: "Successfully updated job silver_airbnb_cleaning_job. To run the job choose the Run Job button." Below this, the "silver_airbnb_cleaning_job" script is displayed in Python:

```

1 import sys
2 from pyspark.context import SparkContext
3 from awsglue.context import GlueContext
4 from awsglue.utils import getResolvedOptions
5 from awsglue.dynamicframe import DynamicFrame
6 from pyspark.sql.functions import col, to_date
7
8 # Initialize Spark and Glue
9 sc = SparkContext()
10 glueContext = GlueContext(sc)
11 spark = glueContext.spark_session
12
13 # Read raw CSV from Bronze
14 bronze_path = "s3://neha-bc005-airbnb-pipeline/bronze/"
15 df = spark.read.option("header", True).csv(bronze_path)
    
```

Python Ln 14, Col 55 Errors: 0 Warnings: 0

Runs - Editor - AWS Glue Studio



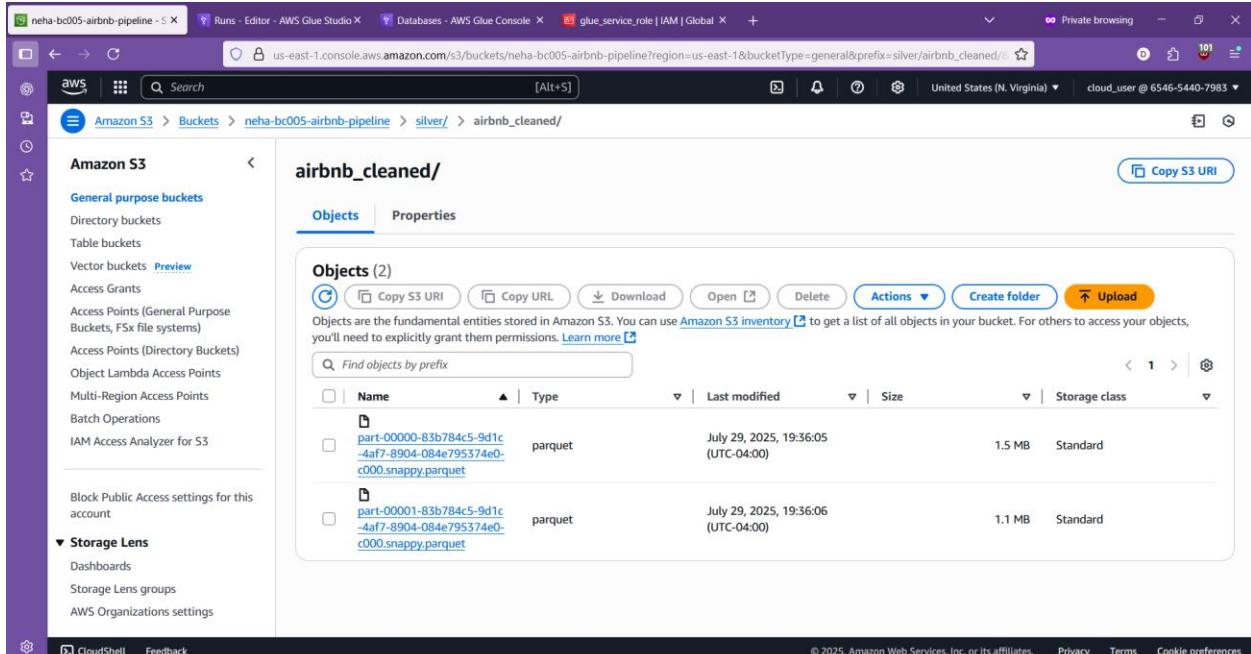
The "Runs" tab shows one job run (1/1) last updated on July 29, 2025 at 23:36:28. The run status is Succeeded, with 0 retries. The run details table includes:

Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)	Worker type	Glue version
Succeeded	0	07/29/2025 19:35:00	07/29/2025 19:36:18	1 m 10 s	10 DPUs	G.1X	5.0

Run details

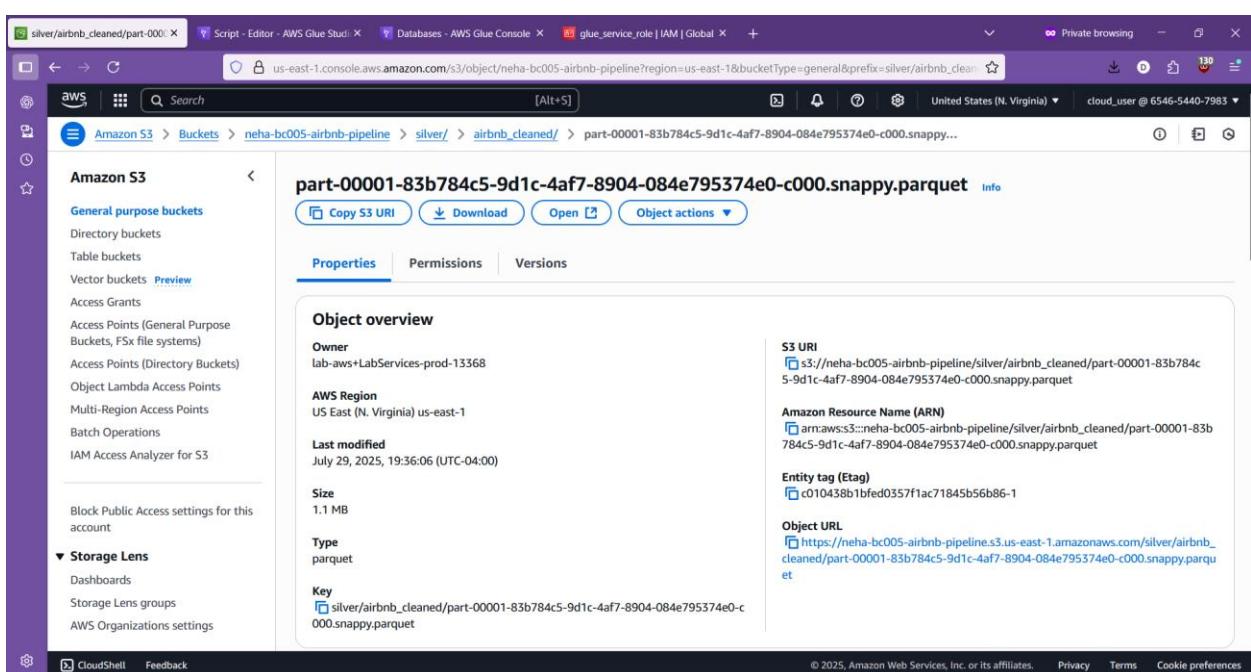
Job name	Start time (Local)	Glue version	Last modified on (Local)
silver_airbnb_cleaning_job	07/29/2025 19:35:00	5.0	07/29/2025 19:36:18
Id	End time (Local)	Worker type	Log group name
jr_f94e4c84305035cc2a289fb0b063aa22682f0c940e1207/29/2025 19:36:18 7930b6ef7d695e6dfaae0		G.1X	/aws-glue/jobs
Run status	Start-up time	Max capacity	Number of workers
Succeeded	8 seconds	10 DPUs	10
Retry attempt number	Execution time	Execution class	Timeout

3. Verify output in S3



The screenshot shows the AWS Glue Studio interface with the 'neha-bc005-airbnb-pipeline - 5' run selected. The main view displays the 'airbnb_cleaned' bucket in Amazon S3. The left sidebar shows navigation options like 'Amazon S3', 'General purpose buckets', 'Storage Lens', and 'CloudShell'. The right panel shows the 'Objects' tab with two parquet files listed:

Name	Type	Last modified	Size	Storage class
part-00000-83b784c5-9d1c-4af7-8904-084e795374e0-c000.snappy.parquet	parquet	July 29, 2025, 19:36:05 (UTC-04:00)	1.5 MB	Standard
part-00001-83b784c5-9d1c-4af7-8904-084e795374e0-c000.snappy.parquet	parquet	July 29, 2025, 19:36:06 (UTC-04:00)	1.1 MB	Standard



The screenshot shows the AWS Glue Studio interface with the 'silver/airbnb_cleaned/part-0001' object selected. The left sidebar shows navigation options like 'Amazon S3', 'General purpose buckets', 'Storage Lens', and 'CloudShell'. The right panel shows the 'Properties' tab for the file 'part-00001-83b784c5-9d1c-4af7-8904-084e795374e0-c000.snappy.parquet'. The 'Object overview' section includes details such as Owner (lab-aws+LabServices-prod-13368), AWS Region (US East (N. Virginia) us-east-1), Last modified (July 29, 2025, 19:36:06 (UTC-04:00)), Size (1.1 MB), Type (parquet), and Key (silver/airbnb_cleaned/part-00001-83b784c5-9d1c-4af7-8904-084e795374e0-c000.snappy.parquet). The 'S3 URI' is listed as s3://neha-bc005-airbnb-pipeline/silver/airbnb_cleaned/part-00001-83b784c5-9d1c-4af7-8904-084e795374e0-c000.snappy.parquet.

4. Crawler for silver data

Crawler properties

Name: silver_airbnb_crawler	IAM role: glue_service_role	Database: airbnb_pipeline_db	State: READY
Description: -	Security configuration: -	Lake Formation configuration: -	Table prefix: silver_
Maximum table threshold: -			

Advanced settings

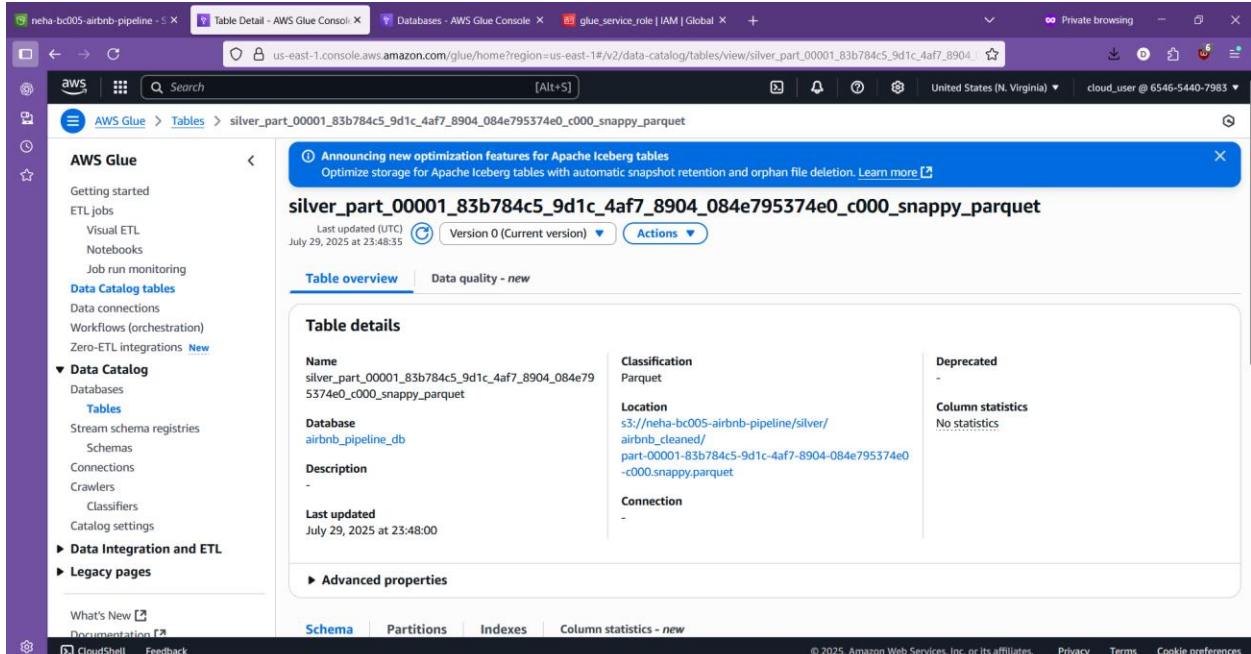
Crawler runs (1)

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
July 29, 2025 at 23:46:44	July 29, 2025 at 23:48:00	01 min 15 s	Completed	-	1 table change, 0 partition changes

Tables (2)

View and manage all available tables.

Name	Database	Location	Classification	Deprecated	View data	Data quality	Column stat...
bronze	airbnb_pipeline_dt	s3://neha-bc005-a	CSV	-	Table data	View data quality	View statistics
silver_part_00001	airbnb_pipeline_dt	s3://neha-bc005-a	Parquet	-	Table data	View data quality	View statistics



The screenshot shows the AWS Glue Table Detail page for the table `silver_part_00001_83b784c5_9d1c_4af7_8904_084e795374e0_c000_snappy_parquet`. The table was last updated on July 29, 2025, at 23:48:00. It has no partitions or indexes. The schema includes columns: id (int), name (string), host_id (int), host_name (string), neighbourhood_group (string), neighbourhood (string), latitude (double), longitude (double), room_type (string), price (double), minimum_nights (int), number_of_reviews (int), last_review (date), reviews_per_month (double), calculated_host_listings_count (int), and availability_365 (int). The table is located in the `s3://neha-bc005-airbnb-pipeline/silver/airbnb_cleaned/part-00001-83b784c5-9d1c-4af7-8904-084e795374e0-c000.snappy.parquet` path.

#	Column name	Data type	Partition key	Comment
1	id	int	-	-
2	name	string	-	-
3	host_id	int	-	-
4	host_name	string	-	-
5	neighbourhood_group	string	-	-
6	neighbourhood	string	-	-
7	latitude	double	-	-
8	longitude	double	-	-
9	room_type	string	-	-
10	price	double	-	-
11	minimum_nights	int	-	-
12	number_of_reviews	int	-	-
13	last_review	date	-	-
14	reviews_per_month	double	-	-
15	calculated_host_listings_count	int	-	-
16	availability_365	int	-	-

Athena Query: Average Price by Neighborhood and Room Type

Query:

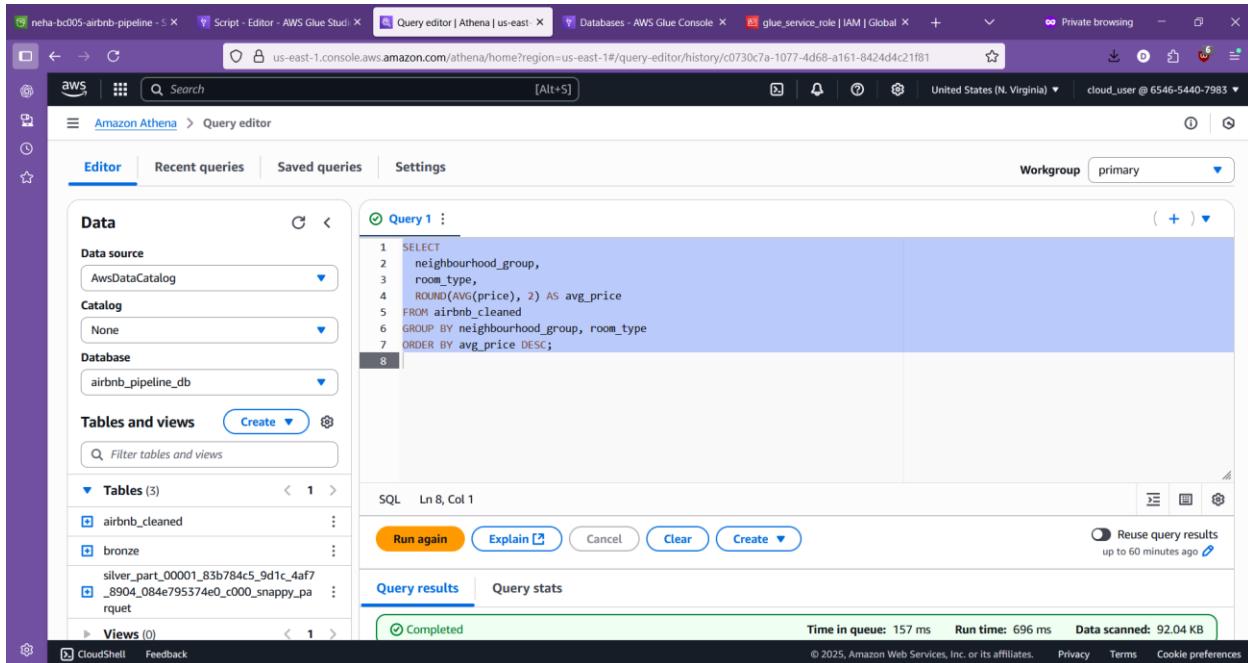
SELECT

neighbourhood_group,

room_type,

ROUND(AVG(price), 2) AS avg_price

FROM airbnb_cleaned
 GROUP BY neighbourhood_group, room_type
 ORDER BY avg_price DESC;

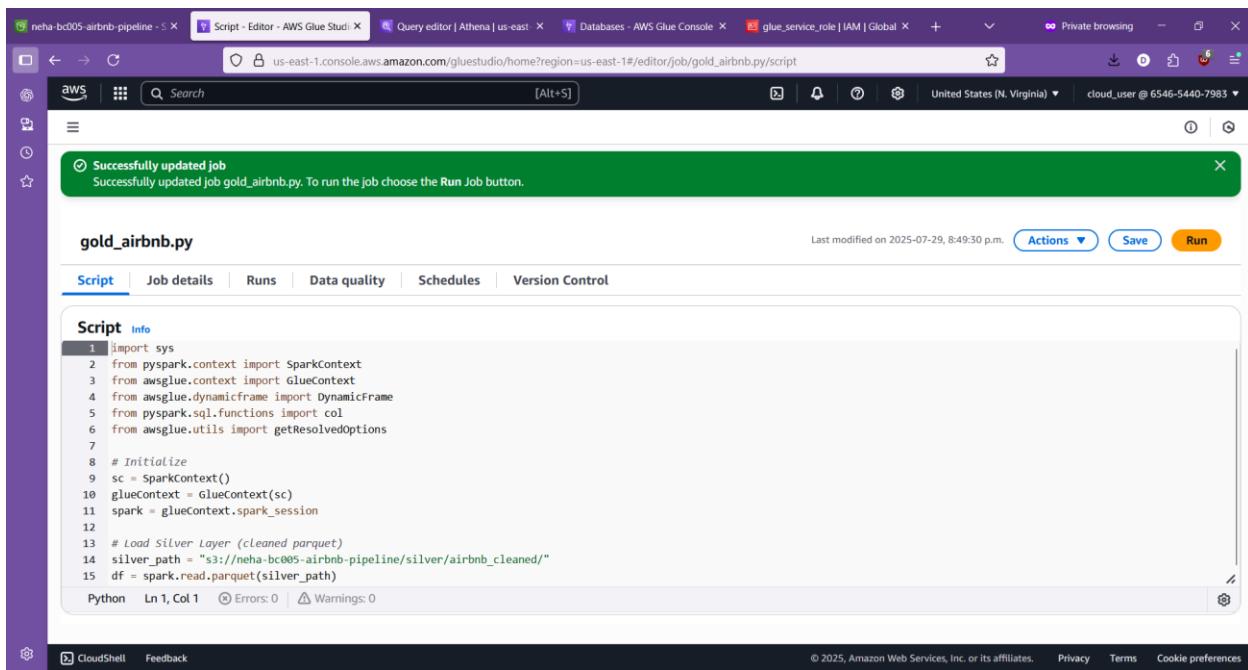


```

SELECT
    neighbourhood_group,
    room_type,
    ROUND(AVG(price), 2) AS avg_price
FROM airbnb_cleaned
GROUP BY neighbourhood_group, room_type
ORDER BY avg_price DESC;
  
```

Gold Layer – Business-Ready Data

1. Create AWS Glue Job for Gold Layer



```

import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.dynamicframe import DynamicFrame
from pyspark.sql.functions import col
from awsglue.utils import getResolvedOptions

# Initialize
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

# Load Silver Layer (cleaned parquet)
silver_path = "s3://neha-bc005-airbnb-pipeline/silver/airbnb_cleaned/"
df = spark.read.parquet(silver_path)
  
```

Last modified on 2025-07-29, 8:49:30 p.m.

Job runs (1/1) Info

Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)	Worker type	Glue version
Succeeded	0	07/29/2025 20:49:38	07/29/2025 20:51:15	1 m 22 s	2 DPUs	G.1X	5.0

Run details

Job name	gold_airbnb.py	Start time (Local)	07/29/2025 20:49:38	Glue version	5.0	Last modified on (Local)	07/29/2025 20:51:15
Id	jr_ae9ceb6cae91576c9f92cad6577fd6899d30b3965	End time (Local)	07/29/2025 20:51:15	Worker type	G.1X	Log group name	/aws-glue/jobs
Run status	Succeeded	Start-up time	14 seconds	Max capacity	2 DPUs	Number of workers	2
Retry attempt number		Execution time		Execution class		Timeout	

2. Verify the output in S3

Amazon S3

General purpose buckets

- Directory buckets
- Table buckets
- Vector buckets [Preview](#)
- Access Grants
- Access Points (General Purpose Buckets, FSx file systems)
- Access Points (Directory Buckets)
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

- Dashboards
- Storage Lens groups
- AWS Organizations settings

gold/

Objects

Name	Type	Last modified	Size	Storage class
dim_listings/	Folder	-	-	-
dim_location/	Folder	-	-	-
fact_reviews/	Folder	-	-	-

Objects (4)

Name	Type	Last modified	Size	Storage class
part-00000-ca001ac1- b64d-45ca-bb78- c717bde5cb8- c000.snappy.parquet	parquet	July 29, 2025, 20:51:01 (UTC-04:00)	431.0 KB	Standard
part-00001-ca001ac1- b64d-45ca-bb78- c717bde5cb8- c000.snappy.parquet	parquet	July 29, 2025, 20:51:01 (UTC-04:00)	426.1 KB	Standard
part-00002-ca001ac1- b64d-45ca-bb78- c717bde5cb8- c000.snappy.parquet	parquet	July 29, 2025, 20:51:01 (UTC-04:00)	428.8 KB	Standard
part-00003-ca001ac1- b64d-45ca-bb78- c717bde5cb8- c000.snappy.parquet	parquet	July 29, 2025, 20:51:01 (UTC-04:00)	430.4 KB	Standard

Objects (4)

Name	Type	Last modified	Size	Storage class
part-00000-60cb59ae-14d9- -461f-bfcc-38a7f12e5517- c000.snappy.parquet	parquet	July 29, 2025, 20:51:03 (UTC-04:00)	238.5 KB	Standard
part-00001-60cb59ae-14d9- -461f-bfcc-38a7f12e5517- c000.snappy.parquet	parquet	July 29, 2025, 20:51:03 (UTC-04:00)	226.2 KB	Standard
part-00002-60cb59ae-14d9- -461f-bfcc-38a7f12e5517- c000.snappy.parquet	parquet	July 29, 2025, 20:51:03 (UTC-04:00)	235.3 KB	Standard
part-00003-60cb59ae-14d9- -461f-bfcc-38a7f12e5517- c000.snappy.parquet	parquet	July 29, 2025, 20:51:03 (UTC-04:00)	230.5 KB	Standard

Amazon S3

General purpose buckets

Objects (2)

Name	Type	Last modified	Size	Storage class
part-00000-848f4b58-420d-4b13-b76d-5ba72ced7134-c000.snappy.parquet	parquet	July 29, 2025, 20:51:04 (UTC-04:00)	244.4 KB	Standard
part-00001-848f4b58-420d-4b13-b76d-5ba72ced7134-c000.snappy.parquet	parquet	July 29, 2025, 20:51:04 (UTC-04:00)	158.1 KB	Standard

3. Create Crawlers for Gold Tables

There are 3 Gold folders:

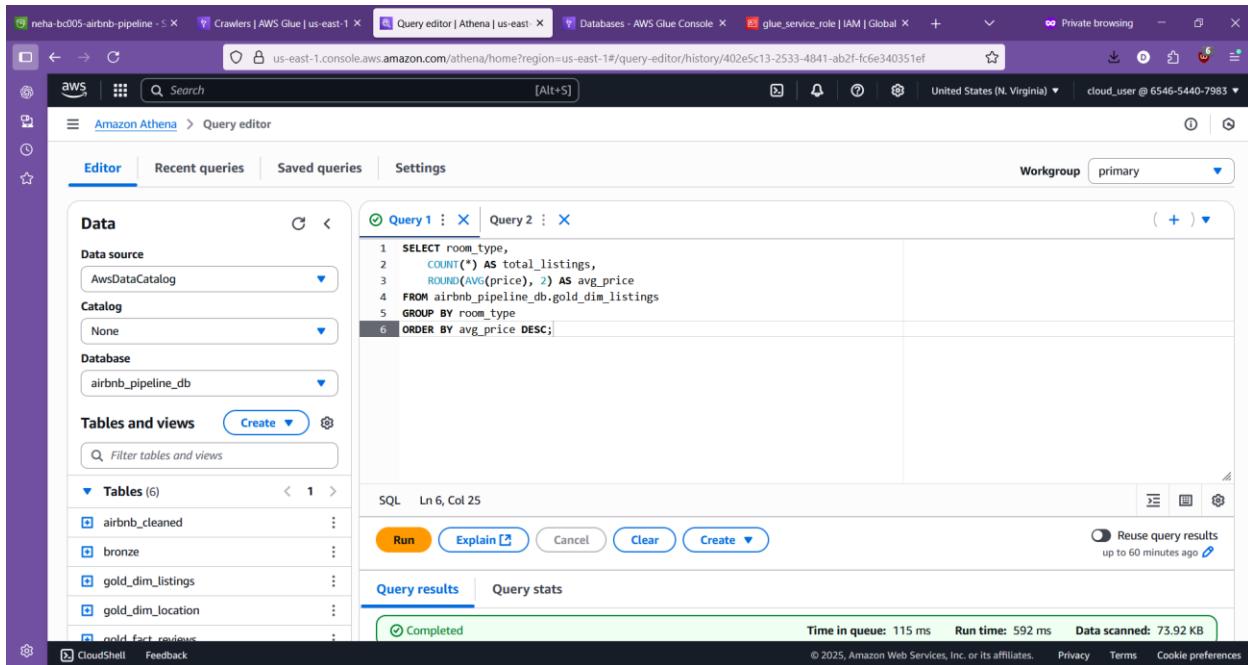
- gold/dim_listings/
- gold/dim_location/
- gold/fact_reviews/

Create one crawler for each.

Name	State	Last run	Last run time...	Log	Table cha..
bronze_airbnb_crawler	Ready	Succeeded	July 29, 2025 at ...	View log	1 created
gold_dim_listings_crawler	Ready	Succeeded	July 30, 2025 at ...	View log	1 created
gold_dim_location_crawler	Ready	Succeeded	July 30, 2025 at ...	View log	1 created
gold_fact_reviews_crawler	Ready	Succeeded	July 30, 2025 at ...	View log	1 created
silver_airbnb_crawler	Ready	Succeeded	July 29, 2025 at ...	View log	1 created

4. Query Gold Tables in Athena

4.1 Listings Overview

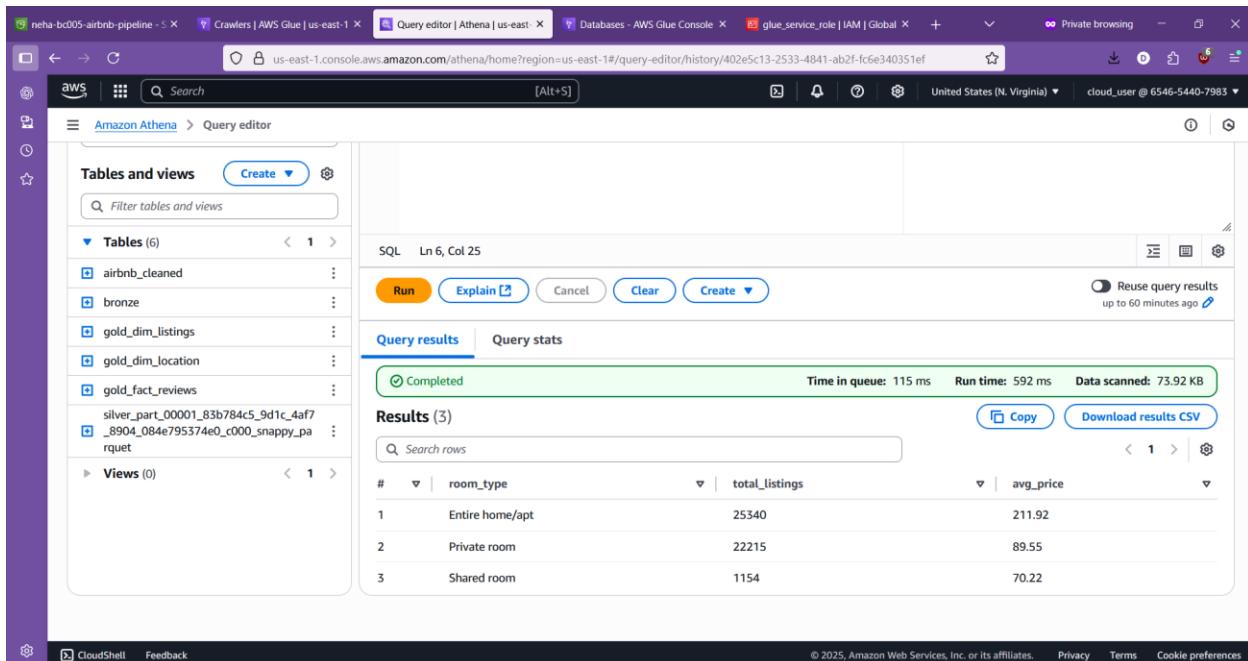


The screenshot shows the AWS Athena Query Editor interface. The left sidebar displays the Data Catalog configuration, including the Data source (AwsDataCatalog), Catalog (None), and Database (airbnb_pipeline_db). The Tables and views section lists several tables: airbnb_cleaned, bronze, gold_dim_listings, gold_dim_location, gold_fact_reviews, silver_part_00001_83b784c5_9d1c_4af7, and _8904_084e795374e0_c000_snappy_paquet. The main panel shows a completed query in the Query 1 tab:

```

1 SELECT room_type,
2     COUNT(*) AS total_listings,
3     ROUND(AVG(price), 2) AS avg_price
4 FROM airbnb_pipeline_db.gold_dim_listings
5 GROUP BY room_type
6 ORDER BY avg_price DESC;
    
```

The query was run successfully with the following metrics: Time in queue: 115 ms, Run time: 592 ms, and Data scanned: 73.92 KB.



The screenshot shows the AWS Athena Query Editor interface with the results of the completed query. The left sidebar shows the same catalog and table list as the previous screenshot. The main panel displays the results in the Query results tab:

#	room_type	total_listings	avg_price
1	Entire home/apt	25340	211.92
2	Private room	22215	89.55
3	Shared room	1154	70.22

4.2 Most Expensive Neighborhoods

Data

Data source: AwsDataCatalog

Catalog: None

Database: airbnb_pipeline_db

Tables and views:

- airbnb_cleaned
- bronze
- gold_dim_listings
- gold_dim_location
- gold_fact_reviews

Query 1

```
1 SELECT l.neighbourhood, AVG(dl.price) AS avg_price
2 FROM airbnb_pipeline_db.gold_dim_listings dl
3 JOIN airbnb_pipeline_db.gold_dim_location l ON dl.id = l.id
4 GROUP BY l.neighbourhood
5 ORDER BY avg_price DESC
6 LIMIT 10;
```

Query 2

SQL Ln 7, Col 1

Run again Explain Cancel Clear Create

Reuse query results up to 60 minutes ago

Query results Completed Time in queue: 107 ms Run time: 1.028 sec Data scanned: 498.81 KB

Completed Time in queue: 107 ms Run time: 1.028 sec Data scanned: 498.81 KB

Results (10)

#	neighbourhood	avg_price
1	Fort Wadsworth	800.0
2	Woodrow	700.0
3	Sea Gate	548.333333333334
4	Tribeca	490.638418079096
5	Riverdale	442.09090909090907
6	Prince's Bay	409.5
7	Battery Park City	367.5571428571429
8	Randall Manor	352.94444444444446
9	Flatiron District	341.925
10	NoHo	297.85526315789474

4.3 Availability and Engagement

The screenshot shows the AWS Athena Query Editor interface. On the left, the Data source is set to AwsDataCatalog, Catalog to None, and Database to airbnb_pipeline_db. The Tables and views section lists several tables: airbnb_cleaned, bronze, gold_dim_listings, gold_dim_location, gold_fact_reviews, silver_part_00001_83b784c5_9d1c_4af7, and _8904_084e795574e0_c000_snappy_p. The main area displays a completed SQL query (Query 1) with a run time of 1.044 sec and data scanned of 515.85 KB. The results table shows the following data:

#	neighbourhood_group	avg_availability	avg_reviews
1	Queens	144.19	1.93
2	Staten Island	198.36	1.85
3	Bronx	165.45	1.83
4	Brooklyn	100.13	1.28
5	Manhattan	111.86	1.27

ML Use Case: Price Prediction Model

Objective:

Predict a listing's price based on its attributes like:

- room type, location, number of reviews, availability, etc.

This will help with:

- Price benchmarking, Anomaly detection, Host pricing suggestions

Build Price Prediction in Databricks (with scikit-learn)

Databricks script:

<https://community.cloud.databricks.com/editor/notebooks/641076367412301?o=1049562611814646#command/641076367412311>

```

Neha_BC005_ML_Price_Prediction Python
File Edit View Run Help Last edit was now
Run all My Cluster Share Publish
Python
1
2
3 minutes ago (32s)
4
5 spark._jsc.hadoopConfiguration().set("fs.s3a.endpoint", "s3.amazonaws.com")
6
7 #
8 # -----
9 # Load Silver Layer Parquet Data from S3
10 #
11 df = spark.read.parquet("s3a://neha-bc005-airbnb-pipeline/silver/airbnb_cleaned/")
12
13 # Optional: Convert to Pandas for ML
14 df = df.toPandas()
15
16 # Preview
17 df.head()
18

```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_n
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149.0	1	9	2018-10-19	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225.0	1	45	2019-05-21	
2	3647	THE VILLAGE OF HARLEM - NEW YORK!	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150.0	3	0	None	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89.0	1	270	2019-07-05	

```

Neha_BC005_ML_Price_Prediction Python
File Edit View Run Help Last edit was 1 minute ago
Run all My Cluster Share Publish
Python
1
2
3 minutes ago (4s)
3
4 minutes ago (4s)
# Step 1: Load Cleaned Data From S3
import pandas as pd

df = spark.read.parquet("s3a://neha-bc005-airbnb-pipeline/silver/airbnb_cleaned/").toPandas()
df = df.dropna(subset=["price", "room_type", "neighbourhood_group", "latitude", "longitude"])

(2) Spark Jobs

```

```

3
4 minutes ago (2s)
# Step 2: Encode Categorical Columns
from sklearn.preprocessing import LabelEncoder

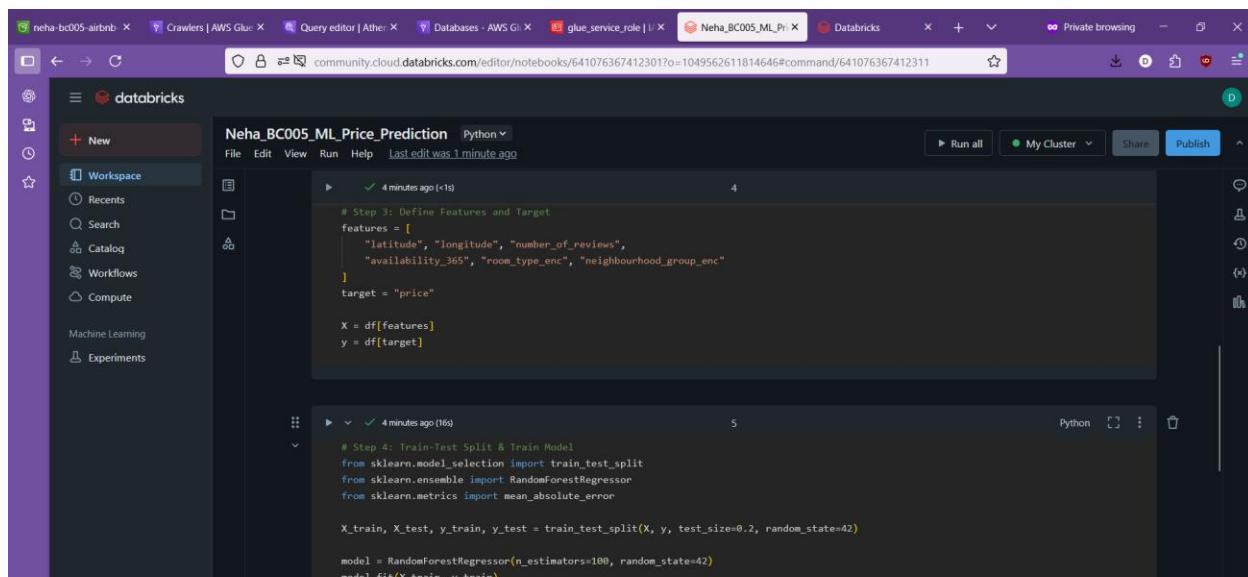
df['room_type_enc'] = LabelEncoder().fit_transform(df['room_type'])
df['neighbourhood_group_enc'] = LabelEncoder().fit_transform(df['neighbourhood_group'])


```

```

4
4 minutes ago (<1s)
# Step 3: Define Features and Target
features = [
    "room_type_enc",
    "neighbourhood_group_enc",
    "latitude",
    "longitude",
    "minimum_nights",
    "number_of_reviews",
    "last_review"
]
target = "price"

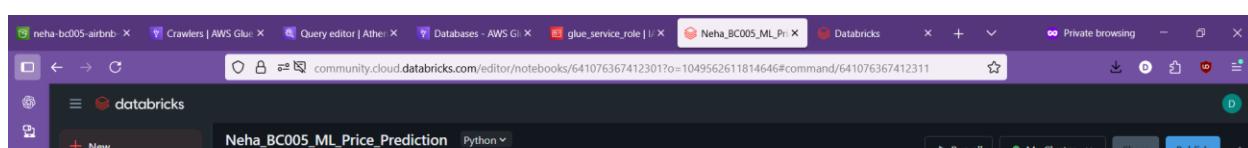
```



```
# Step 3: Define Features and Target
features = [
    "latitude", "longitude", "number_of_reviews",
    "availability_365", "room_type_enc", "neighbourhood_group_enc"
]
target = "price"

X = df[features]
y = df[target]
```

MAE: 70.78728033258058



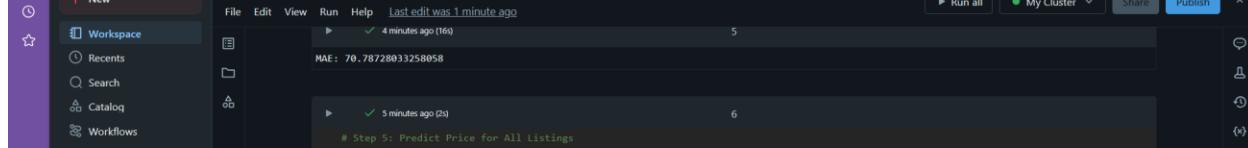
```
# Step 4: Train-Test Split & Train Model
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

preds = model.predict(X_test)
print("MAE:", mean_absolute_error(y_test, preds))
```

MAE: 70.78728033258058



```
# Step 5: Predict Price for All Listings
df["predicted_price"] = model.predict(X)
```



```
# If you already have 'df' as a pandas dataframe:
spark_df = spark.createDataFrame(df[["id", "name", "room_type", "neighbourhood_group", "price", "predicted_price"]])

# Save as CSV to S3 Gold folder
spark_df.write.mode("overwrite").csv("s3a://neha-bc005-airbnb-pipeline/gold/predicted_airbnb_prices/", header=True)
```

(1) Spark Jobs

spark_df: pyspark.sql.dataframe.DataFrame = [id:integer, name:string ... 4 more fields]

[Shift+Enter] to run and move to next cell
[Ctrl+Shift+M] to open the command palette
[Esc H] to see all keyboard shortcuts

Objects (11)

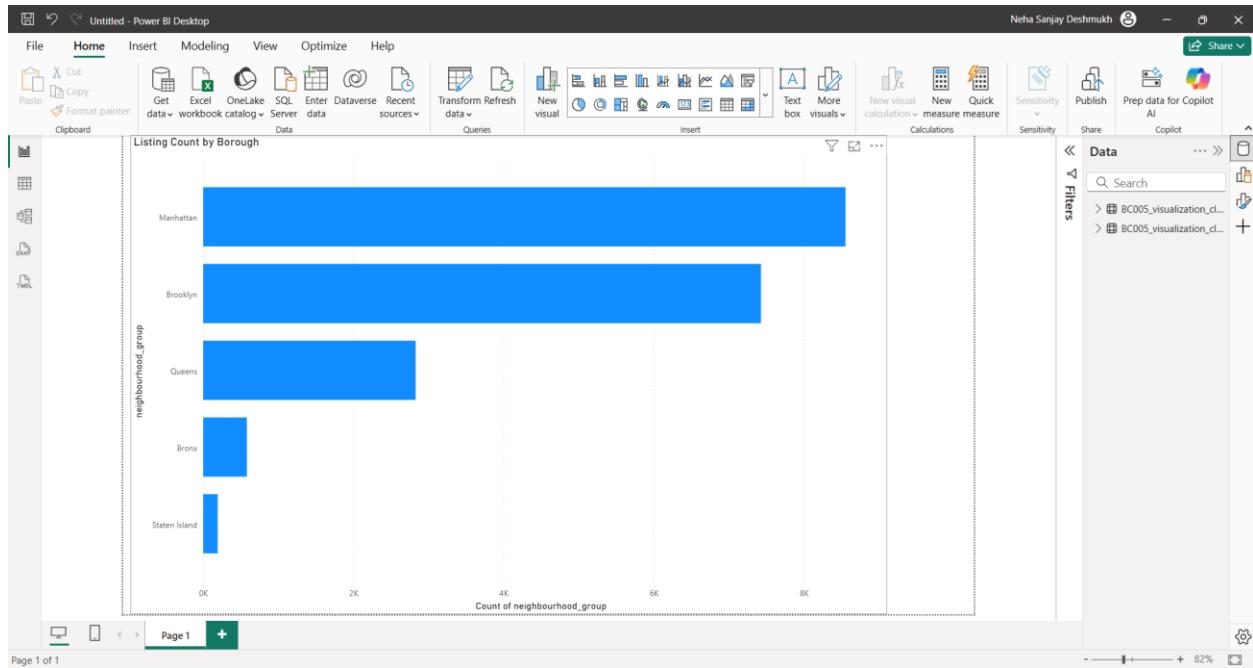
Name	Type	Last modified	Size	Storage class
_committed_7576141998044797751	-	July 29, 2025, 23:26:23 (UTC-04:00)	724.0 B	Standard
_started_7576141998044797751	-	July 29, 2025, 23:26:22 (UTC-04:00)	0 B	Standard
_SUCCESS	-	July 29, 2025, 23:26:24 (UTC-04:00)	0 B	Standard
part-00000-tid-7576141998044797751-cfd73fd4-0ec5-4fea-9ce5-3	CSV	July 29, 2025, 23:26:22 (UTC-04:00)	469.8 KB	Standard

Objects (11)

Name	Type	Last modified	Size	Storage class
_committed_7576141998044797751	-	July 29, 2025, 23:26:23 (UTC-04:00)	498.7 KB	Standard
_started_7576141998044797751	-	July 29, 2025, 23:26:23 (UTC-04:00)	507.5 KB	Standard
_SUCCESS	-	July 29, 2025, 23:26:23 (UTC-04:00)	509.3 KB	Standard

Visualization

1. Listing Count by Borough - Shows how listings are distributed across NYC.



2. Average Price by Room Type - Pricing differences among Entire home, Private room, Shared room.

