

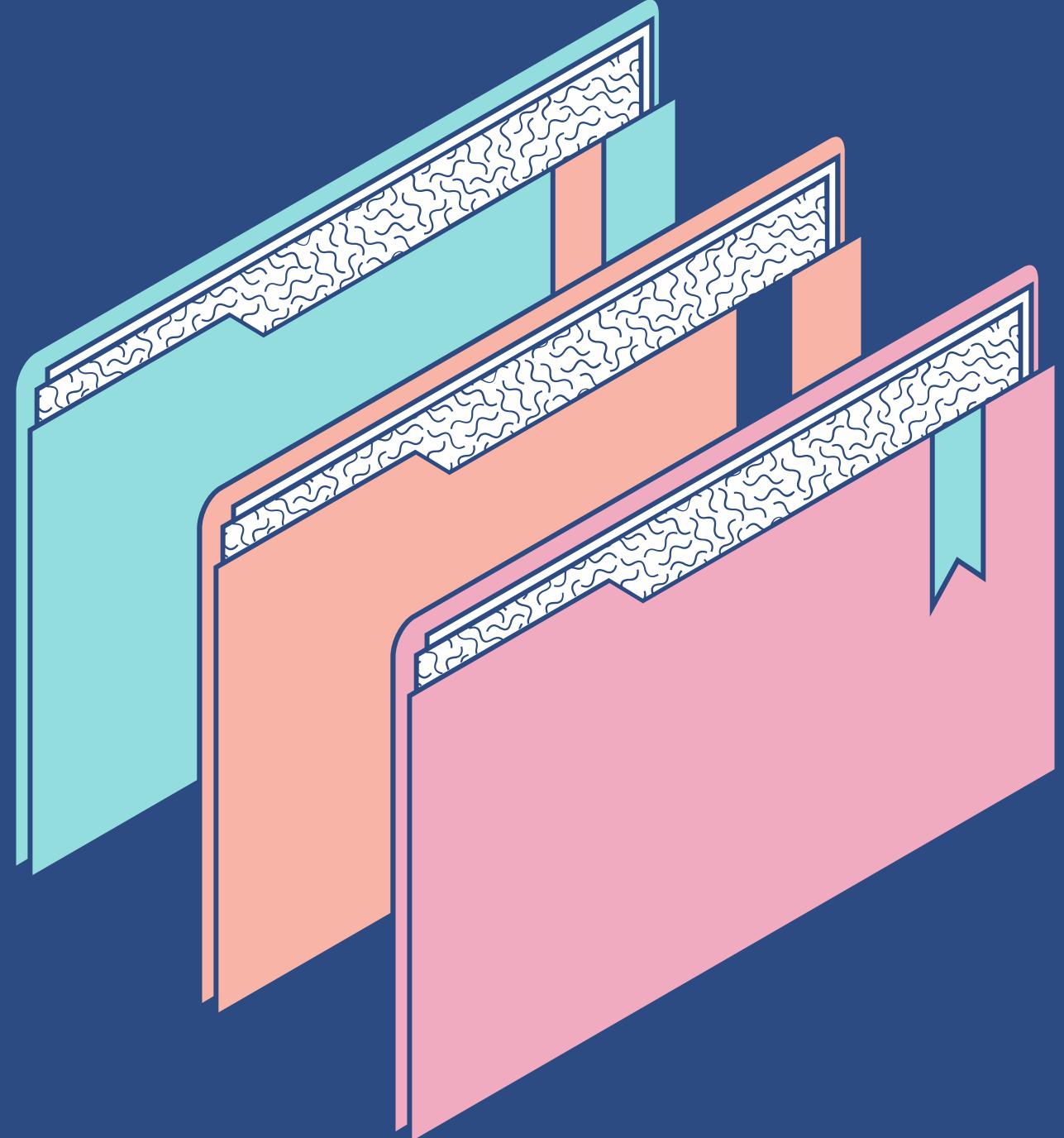


Airbnb Ratings & Pricing Insights using AWS Big Data Pipeline

Saurabh Ghumnar



Agenda



KEY TOPICS DISCUSSED IN THIS PRESENTATION

- Problem Statement
- Dataset Overview
- Dataset Schema
- AWS Architecture Diagram
- Lambda Automation
- Glue Crawler Setup
- Athena Queries and Insights
- Machine Learning with EMR
- Final Outputs
- Challenges and Resolutions

Problem Statement

***Objective:* Discover actionable insights from Airbnb data using cloud-native analytics.**

This project focuses on answering key questions like:

- What factors influence customer ratings across cities?
- How do pricing trends correlate with customer satisfaction?
- Can we identify overpriced underperformers?
- How do we scale this analysis using modern cloud tools?



Dataset Overview

- We used publicly available Airbnb datasets covering listings and reviews from:
 - Los Angeles
 - New York City These were uploaded to:
 - S3 Bucket: airbnb/raw/
 - Subfolders: LA_Listings/, NY_Listings/, Reviews/, Ratings/

Dataset Schema

- Key attributes across the datasets include:
 - listing id, host name, room type, price, review scores rating
 - Reviews metadata with timestamps and sentiment-rich text This structured schema enabled meaningful groupings (by room type, neighborhood, etc.) and served as the basis for ML feature engineering.

us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/84178ff8-09db-4d7a-b33c-9fb796be65c4

Relaunch

Search [Option+S]

United States (N. Virginia)

voclabs/user3783176=Aishwary_Joshi @ 7492-3

Amazon Athena > Query editor

Editor Recent queries Saved queries Settings Workgroup primary

Data

Data source: AwsDataCatalog

Catalog: None

Database: airbnb_reviews_db

Tables and views: Create

Filter tables and views

Tables (5): airbnb_la_listings, airbnb_ny_listings, airbnb_ratings, airbnb_reviews, airbnb_sample

Views (0)

Query 8: Average Rating by Room Type

```
1 -- 2 Average Rating by Room Type
2
3 SELECT
4   "room type",
5   AVG("review scores rating") AS avg_rating,
6   COUNT(*) AS total_listings
7 FROM "AwsDataCatalog"."airbnb_reviews_db"."airbnb_la_listings"
8 GROUP BY "room type"
9 ORDER BY avg_rating DESC;
```

SQL Ln 10, Col 1

Run again Explain Cancel Clear Create Reuse query up to 60 minutes

Completed Time in queue: 100 ms Run time: 1.047 sec Data scanned: 21.36

Results (4) Copy Download results

Search rows

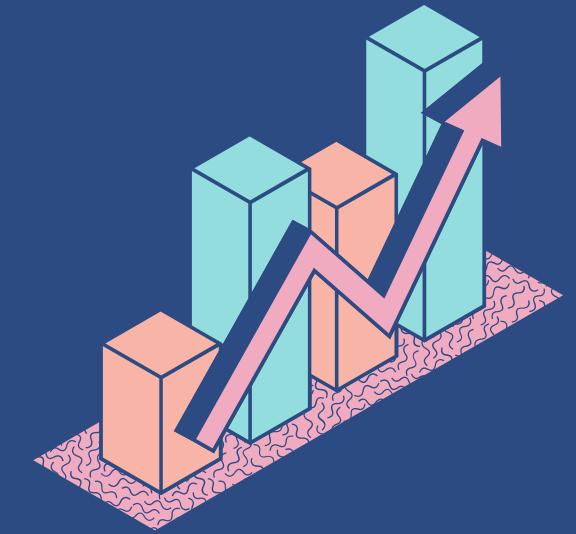
AWS Architecture Diagram



LAMBDA AUTOMATION



- Lambda triggers on file upload in S3
- Script handles:
 1. Glue DB: airbnb_reviews_db
 2. Dynamic crawler setup
 3. Table creation & monitoring



```

import boto3
import urllib.parse
import time

s3 = boto3.client('s3')
glue = boto3.client('glue')

GLUE_DATABASE_NAME = "airbnb_reviews_db"
GLUE_ROLE_NAME = "LabRole"
CRAWLER_PREFIX = "crawler_for_"
TABLE_PREFIX = "airbnb_"

def lambda_handler(event, context):
    for record in event['Records']:
        bucket = record['s3']['bucket']['name']
        key = urllib.parse.unquote_plus(record['s3']['object']['key'])

    print(f"✓ File uploaded: s3://{bucket}/{key}")

    key_parts = key.split('/')
    if len(key_parts) < 4:
        print("⚠ File not under expected path /airbnb/raw/<folder>/<file.csv>")
        return

    table_folder = key_parts[2] # FIXED: correctly gets folder name
    table_name_suffix = table_folder.replace('-', '_').lower()
    final_table_name = TABLE_PREFIX + table_name_suffix
    crawler_name = CRAWLER_PREFIX + table_name_suffix
    s3_target_path = f"s3://{bucket}/airbnb/raw/{table_folder}/"

    try:
        glue.create_database(DatabaseInput={'Name': GLUE_DATABASE_NAME})
        print(f"✓ Created Glue database: {GLUE_DATABASE_NAME}")
    except glue.exceptions.AlreadyExistsException:
        print(f"ℹ️ Glue database {GLUE_DATABASE_NAME} already exists")

    try:
        glue.create_crawler(
            Name=crawler_name,
            Role=GLUE_ROLE_NAME,
            DatabaseName=GLUE_DATABASE_NAME,
            Targets={'S3Targets': [{'Path': s3_target_path}]},
            TablePrefix=TABLE_PREFIX,
            SchemaChangePolicy={
                'UpdateBehavior': 'UPDATE_IN_DATABASE',
                'DeleteBehavior': 'DEPRECATE_IN_DATABASE'
            }
        )
        print(f"✓ Created crawler: {crawler_name}")
    except glue.exceptions.AlreadyExistsException:
        print(f"ℹ️ Crawler {crawler_name} already exists")

    try:
        glue.start_crawler(Name=crawler_name)
        print(f"➤ Started crawler: {crawler_name}")
    except glue.exceptions.CrawlerRunningException:
        print(f"⚠️ Crawler {crawler_name} is already running")

    # Wait for crawler to finish
    print("➤ Waiting for crawler to finish...")
    while True:
        status = glue.get_crawler(Name=crawler_name)['Crawler']['State']
        if status == 'READY':
            break
        time.sleep(5)

    print(f"✓ Crawler finished. Looking for table: {final_table_name} in DB: {GLUE_DATABASE_NAME}")

    try:
        glue.start_crawler(Name=crawler_name)
        print(f"➤ Started crawler: {crawler_name}")
    except glue.exceptions.CrawlerRunningException:
        print(f"⚠️ Crawler {crawler_name} is already running")

    # Wait for crawler to finish
    print("➤ Waiting for crawler to finish...")
    while True:
        status = glue.get_crawler(Name=crawler_name)['Crawler']['State']
        if status == 'READY':
            break
        time.sleep(5)

    print(f"✓ Crawler finished. Looking for table: {final_table_name} in DB: {GLUE_DATABASE_NAME}")

    try:
        response = glue.get_table(
            DatabaseName=GLUE_DATABASE_NAME,
            Name=final_table_name
        )
        print(f"✓ Table found: {final_table_name}")
    except glue.exceptions.EntityNotFoundException:
        print(f"✗ Table {final_table_name} NOT found in database {GLUE_DATABASE_NAME}.")
        print(f"🔍 Check if the file at {s3_target_path} is correctly formatted and non-empty.")
    return {"status": "done"}

```

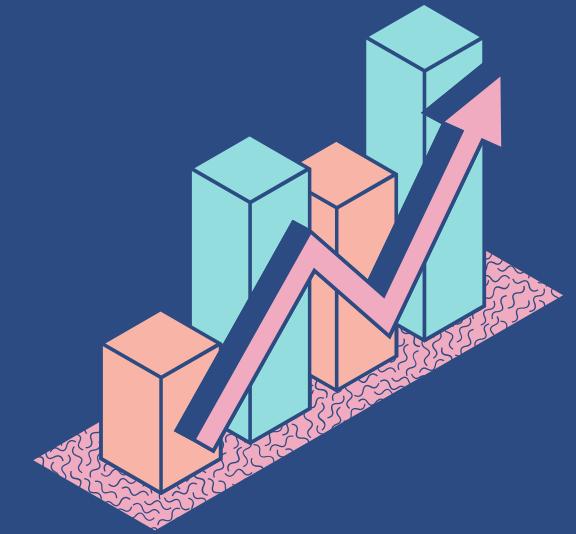
Lambda Code Snippet

(lambda_code.py)

Glue Crawler Setup



- 5 Glue tables auto-created
- Crawlers run on individual S3 folders
- Tables: LA, NY Listings, Ratings, Reviews, Sample



0
B
J
E
C
T
S

CloudShell Feedback

Launch AWS Academy Learner | Console Home | final-project-airbnb-reviews-aj61?region=us-east-1&bucketType=general&prefix=airbnb/raw/&showv... Relaunch to update

aws Search [Option+S] United States (N. Virginia) vocabs/user3783176=Aishwary_Joshi @ 7492-3810-0078

Amazon S3 > Buckets > final-project-airbnb-reviews-aj61 > airbnb/ > raw/

raw/ Copy S3 URI

Objects Properties

Objects (5)

Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix < 1 > ⚙️

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	LA_Listings/	Folder	-	-	-
<input type="checkbox"/>	NY_Listings/	Folder	-	-	-
<input type="checkbox"/>	Ratings/	Folder	-	-	-
<input type="checkbox"/>	Reviews/	Folder	-	-	-
<input type="checkbox"/>	Sample/	Folder	-	-	-

Block Public Access settings for this account

Storage Lens

- Dashboards
- Storage Lens groups
- AWS Organizations settings

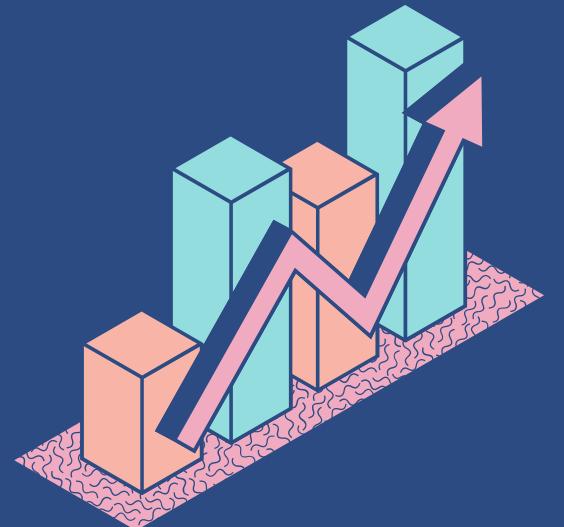
Feature spotlight 11

AWS Marketplace for S3

Glue Databases And Tables



- Database: airbnb_reviews_db
- Location mapped to: s3://final-project-airbnb-reviews...
- Format: CSV with structured schema



Launch AWS Academy Learner | Console Home | Databases - AWS Glue Cons...

us-east-1.console.aws.amazon.com/glue/home?region=us-east-1#/v2/data-catalog/databases/view/airbnb_reviews_db?catalogId=749238100078

Relaunch to update

AWS Glue | Search [Option+S] | United States (N. Virginia) | voclabs/user3783176=Aishwary_Joshi @ 7492-3810-0078

AWS Glue > Databases > airbnb_reviews_db

AWS Glue

- Getting started
- ETL jobs
- Visual ETL
- Notebooks
- Job run monitoring
- Data Catalog tables
- Data connections
- Workflows (orchestration)
- Zero-ETL integrations [New](#)

Data Catalog

Databases

- Tables
- Stream schema registries
- Schemas
- Connections
- Crawlers
- Classifiers
- Catalog settings

Data Integration and ETL

Legacy pages

What's New [New](#)

Documentation [New](#)

AWS Marketplace

Enable compact mode

airbnb_reviews_db

Announcing new optimization features for Apache Iceberg tables
Optimize storage for Apache Iceberg tables with automatic snapshot retention and orphan file deletion. [Learn more](#)

Last updated (UTC)
May 12, 2025 at 00:47:35

Edit Delete

Database properties

Name	Description	Location	Created on (UTC)
airbnb_reviews_db	-	-	May 10, 2025 at 01:47:17

Tables (5)

Last updated (UTC)
May 12, 2025 at 00:47:36

Add tables using crawler Add table

View and manage all available tables.

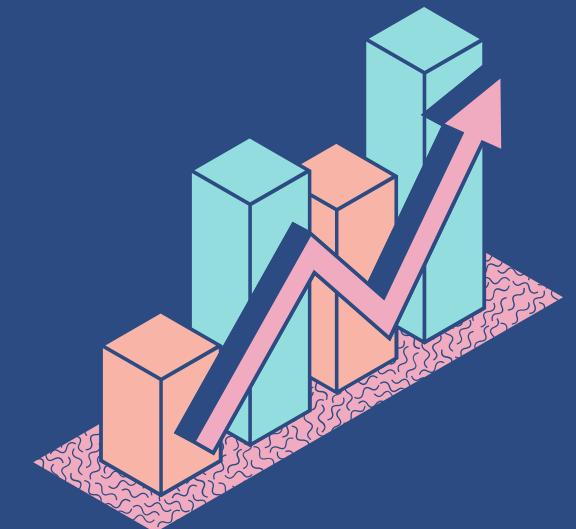
<input type="checkbox"/>	Name	Database	Location	Classification	Deprecated	View data	Data quality	Column stati...
<input type="checkbox"/>	airbnb_la_listings	airbnb_reviews_db	s3://final-project-1	CSV	-	Table data	View data quality	View statistics
<input type="checkbox"/>	airbnb_ny_listings	airbnb_reviews_db	s3://final-project-1	CSV	-	Table data	View data quality	View statistics
<input type="checkbox"/>	airbnb_ratings	airbnb_reviews_db	s3://final-project-1	CSV	-	Table data	View data quality	View statistics
<input type="checkbox"/>	airbnb_reviews	airbnb_reviews_db	s3://final-project-1	CSV	-	Table data	View data quality	View statistics
<input type="checkbox"/>	airbnb_sample	airbnb_reviews_db	s3://final-project-1	CSV	-	Table data	View data quality	View statistics

T
A
B
L
E
S

Querying with Amazon Athena



- SQL queries run directly on S3 through Glue tables
- Queries written to answer specific business questions





Screenshot of the AWS Glue Crawlers page in the AWS Console.

The browser tabs show: Launch AWS Academy Learner, Console Home | Console Hom, Crawlers - AWS Glue Console, and us-east-1.console.aws.amazon.com/glue/home?region=us-east-1#/v2/data-catalog/crawlers.

The AWS Glue sidebar includes sections for Getting started, ETL jobs, Visual ETL, Notebooks, Job run monitoring, Data Catalog tables, Data connections, Workflows (orchestration), Zero-ETL integrations (New), Data Catalog (with sub-sections like Databases, Tables, Stream schema registries, Schemas, Connections, Crawlers, Classifiers, Catalog settings), Data Integration and ETL, and Legacy pages.

The main content area displays a blue banner about Apache Iceberg optimization features. The "Crawlers" section shows a table with 9 entries, all of which are "Ready" and have "Succeeded" in the last run. The table columns are: Name, State, Schedule, Last run, Last run time..., Log, and Table.

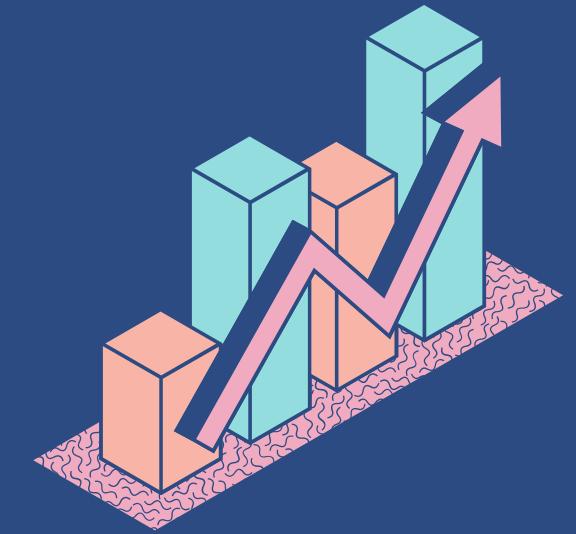
Name	State	Schedule	Last run	Last run time...	Log	Table
crawler_for_la_listings	Ready		Succeeded	May 10, 2025 at ...	View log	1 cre
crawler_for_ny_listings	Ready		Succeeded	May 10, 2025 at ...	View log	1 cre
crawler_for_ratings	Ready		Succeeded	May 10, 2025 at ...	View log	1 cre
crawler_for_reviews	Ready		Succeeded	May 10, 2025 at ...	View log	1 cre
crawler_for_sample	Ready		Succeeded	May 10, 2025 at ...	View log	1 cre

At the bottom, there are links for What's New, Documentation, AWS Marketplace, and a toggle for Enable compact mode.

Insights from SQL Queries



- Avg. Rating by Room Type (Query 9)
- Avg. Rating by Neighborhood (Query 11)
- Top Reviewed Listings (Query 10)



SQL Queries

The image displays two side-by-side screenshots of the Amazon Athena Query editor interface, showing the 'Editor' tab selected. Both screenshots show a list of recent queries at the top, followed by a query editor area and a results section.

Left Screenshot (Query 8):

- Data:** Data source: AwsDataCatalog; Catalog: None; Database: airbnb_reviews_db.
- Tables and views:** Tables (5): airbnb_la_listings, airbnb_ny_listings, airbnb_ratings, airbnb_reviews, airbnb_sample.
- Query Editor:** SQL:

```
-- Top 10 Most Reviewed Listings
SELECT
    "listing id",
    "name",
    "number of reviews"
FROM "AwsDataCatalog"."airbnb_reviews_db"."airbnb_la_listings"
ORDER BY "number of reviews" DESC
LIMIT 10;
```
- Results:** Time in queue: 96 ms, Run time: 925 ms, Data scanned: 21.36 MB. Results (10) - Search rows.

Right Screenshot (Query 11):

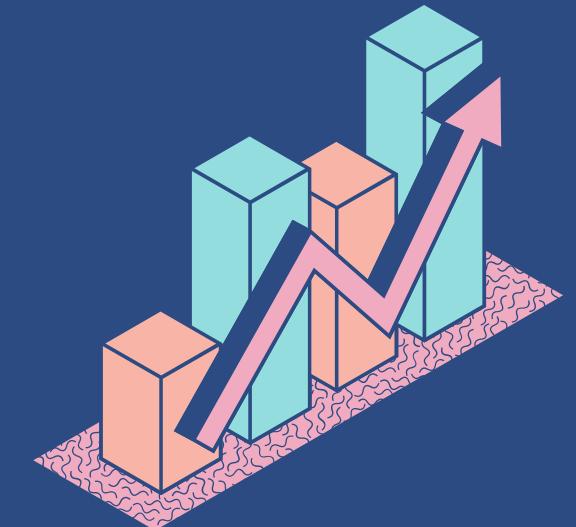
- Data:** Data source: AwsDataCatalog; Catalog: None; Database: airbnb_reviews_db.
- Tables and views:** Tables (5): airbnb_la_listings, airbnb_ny_listings, airbnb_ratings, airbnb_reviews, airbnb_sample.
- Query Editor:** SQL:

```
-- Average Scores by City Neighborhood ( Shows which LA neighborhoods get better ratings on average.)
SELECT
    "neighbourhood cleansed",
    AVG("review scores rating") AS avg_rating,
    COUNT(*) AS listing_count
FROM "AwsDataCatalog"."airbnb_reviews_db"."airbnb_la_listings"
GROUP BY "neighbourhood cleansed"
HAVING COUNT(*) > 5
ORDER BY avg_rating DESC;
```
- Results:** Time in queue: 101 ms, Run time: 810 ms, Data scanned: 21.36 MB. Results (245) - Search rows.

Advanced Analytics Using Athena



- Diving deeper, we ran queries like:
 1. Price vs Review Score (binned) (Query 10)
 2. Listings with High Price but Low Rating:
potential anomalies (Query 15)
 3. Top Hosts by Rating (Query 12)



PRICE
Vs
REVIEW

The screenshot shows the AWS Athena Query editor interface. The top navigation bar includes tabs for 'Launch AWS Academy Learner', 'Console Home | Console Home', and 'Query editor | Athena | us-east-1'. The main title bar displays the URL 'us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/d7733620-9790-4897-b3fa-5285135cf26'. The top right corner features a 'Relaunch to update' button and a user profile icon.

The left sidebar, titled 'Data', contains dropdown menus for 'Data source' (AwsDataCatalog), 'Catalog' (None), and 'Database' (airbnb_reviews_db). Below these are sections for 'Tables and views' with a 'Create' button and a 'Filter tables and views' search bar. A list of tables is shown under 'Tables (5)': airbnb_la_listings, airbnb_ny_listings, airbnb_ratings, airbnb_reviews, and airbnb_sample.

The main query editor area shows a list of recent queries: Query 8, Query 9, Query 10, Query 11, Query 12, Query 13, and a placeholder 'Query'. The currently selected query is 'Query 10', which contains the following SQL code:

```
1 -- 3 Price vs Review Scores Analysis (Binned)
2
3 SELECT CASE
4     WHEN price < 100 THEN 'Budget'
5     WHEN price BETWEEN 100 AND 200 THEN 'Mid-Range'
6     ELSE 'Premium'
7 END AS price_range, AVG("review scores rating") AS avg_rating,
8 COUNT(*) AS total_listings FROM "AwsDataCatalog"."airbnb_reviews_db"."airbnb_la_listings"
9 WHERE price IS NOT NULL AND "review scores rating" IS NOT NULL
10 GROUP BY CASE WHEN price < 100 THEN 'Budget'
11     WHEN price BETWEEN 100 AND 200 THEN 'Mid-Range'
12     ELSE 'Premium'
13 END
14 ORDER BY avg_rating DESC;
```

The status bar at the bottom indicates the query was completed: 'Completed', 'Time in queue: 62 ms', 'Run time: 818 ms', and 'Data scanned: 21.36 MB'. There are buttons for 'Run', 'Explain', 'Cancel', 'Clear', and 'Create'. The 'Query results' tab is active, showing a summary of the results.

us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/717366df-28d0-432c-9e73-5e81fa599c91

Relaunch to update

aws Search [Option+S]

Amazon Athena > Query editor

Editor Recent queries Saved queries Settings Workgroup primary

Data

Data source: AwsDataCatalog

Catalog: None

Database: airbnb_reviews_db

Tables and views: Create Filter tables and views

Tables (5): airbnb_la_listings, airbnb_ny_listings, airbnb_ratings, airbnb_reviews, airbnb_sample

Views (0)

Query 10: *-- Listings with High Price but Low Rating*

```
1 -- Listings with High Price but Low Rating
2
3 SELECT
4   "listing id",
5   "name",
6   price,
7   "review scores rating"
8 FROM "AwsDataCatalog"."airbnb_reviews_db"."airbnb_la_listings"
9 WHERE price > 200 AND "review scores rating" < 3
10 ORDER BY price DESC
11 LIMIT 10;
```

SQL Ln 12, Col 1

Run again Explain Cancel Clear Create Reuse query results up to 60 minutes ago

Completed Time in queue: 103 ms Run time: 692 ms Data scanned: 21.36 MB

Results (10) Copy Download results CSV

Search rows

CloudShell Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

HIGH
PRICE
BUT
LOW
RATING

TOP HOSTS BY RATINGS

Launch AWS Academy Learner | Console Home | Console Home | Query editor | Athena | us-east-1# /query-editor

us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor

aws Search [Option+S] United States (N. Virginia) voclabs/user3783176=Aishwary_Joshi @ 7492-3810-0078

Amazon Athena > Query editor

Editor Recent queries Saved queries Settings Workgroup primary

Data Data source: AwsDataCatalog Catalog: None Database: airbnb_reviews_db Tables and views Create Filter tables and views

Tables (5) airbnb_la_listings airbnb_ny_listings airbnb_ratings airbnb_reviews airbnb_sample Views (0)

Query 8 : X Query 9 : X Query 10 : X Query 11 : X Query 12 : X Query 13 : X Query > (+) ▾

```
-- Top Hosts by Rating in LA
SELECT
    "host name",
    AVG("review scores rating") AS avg_rating,
    COUNT(*) AS listings
FROM "AwsDataCatalog"."airbnb_reviews_db"."airbnb_la_listings"
GROUP BY "host name"
HAVING COUNT(*) > 2
ORDER BY avg_rating DESC
LIMIT 10;
```

SQL Ln 11, Col 10

Run Explain Cancel Clear Create Reuse query results up to 60 minutes ago

Query results Query stats

Results

No results Run a query to view results

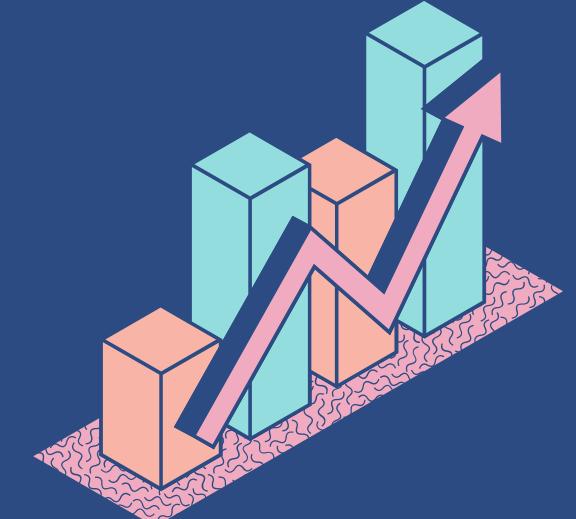
Copy Download results CSV

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Machine Learning With EMR



- To conduct ML experiments:
- Launched Amazon EMR Cluster (JupyterHub + Spark)
- Used Docker to deploy a Jupyter Notebook
- Cluster ID: j-2SWDMEM8YI8ED
- Docker-based Jupyter Notebook setup
- Installed sklearn, xgboost, pandas

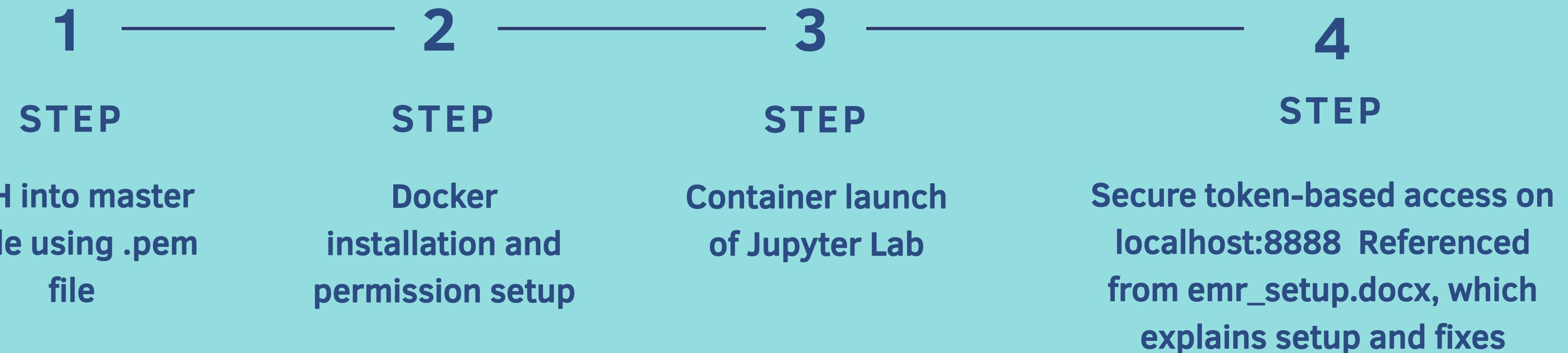


EMR ON EC2 CLUSTER

Launch AWS Academy Learner X Properties > final_cluster > E + us-east-1.console.aws.amazon.com/emr/home?region=us-east-1#/clusterDetails/j-2SWDMEM8YI8ED Relaunch to update : Search [Option+S] United States (N. Virginia) voclabs/user3783176=Aishwary_Joshi @ 7492-3810-0078 Amazon EMR > EMR on EC2: Clusters > final_cluster final_cluster Updated less than a minute ago Terminate Clone in AWS CLI Clone ▼ Summary Cluster info Applications Cluster management Status and time Cluster ID j-2SWDMEM8YI8ED Amazon EMR version emr-6.13.0 Installed applications JupyterHub 1.5.0, Spark 3.4.1 Log destination in Amazon S3 aws-logs-749238100078-us-east-1/elasticmapreduce Status Terminated Cluster ARN arn:aws:elasticmapreduce:us-east-1:749238100078:cluster/j-2SWDMEM8YI8ED Persistent application UIs Spark History Server YARN timeline server Primary node public DNS ec2-3-236-9-104.compute-1.amazonaws.com Connect to the Primary node using SSH Creation time May 11, 2025, 17:54 (UTC-05:00) Elapsed time 48 minutes, 40 seconds End time May 11, 2025, 18:43 (UTC-05:00) Cluster configuration Instance groups Capacity 1 Primary | 1 Core | 1 Task Properties Bootstrap actions Instances (Hardware) Steps Applications Configurations Monitoring Events Tags (0) Operating system Info Amazon Linux release 2.0.20250414.0 Cluster logs Info Archive log files to Amazon S3 Turned on Amazon S3 location s3://aws-logs-749238100078-us-east-1/elasticmapreduce/ Encryption for logs Turned off Cluster termination and node replacement Info Edit Termination option Automatically terminate cluster after idle time Idle time 1 hour Termination protection Off

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

EMR Setup Process



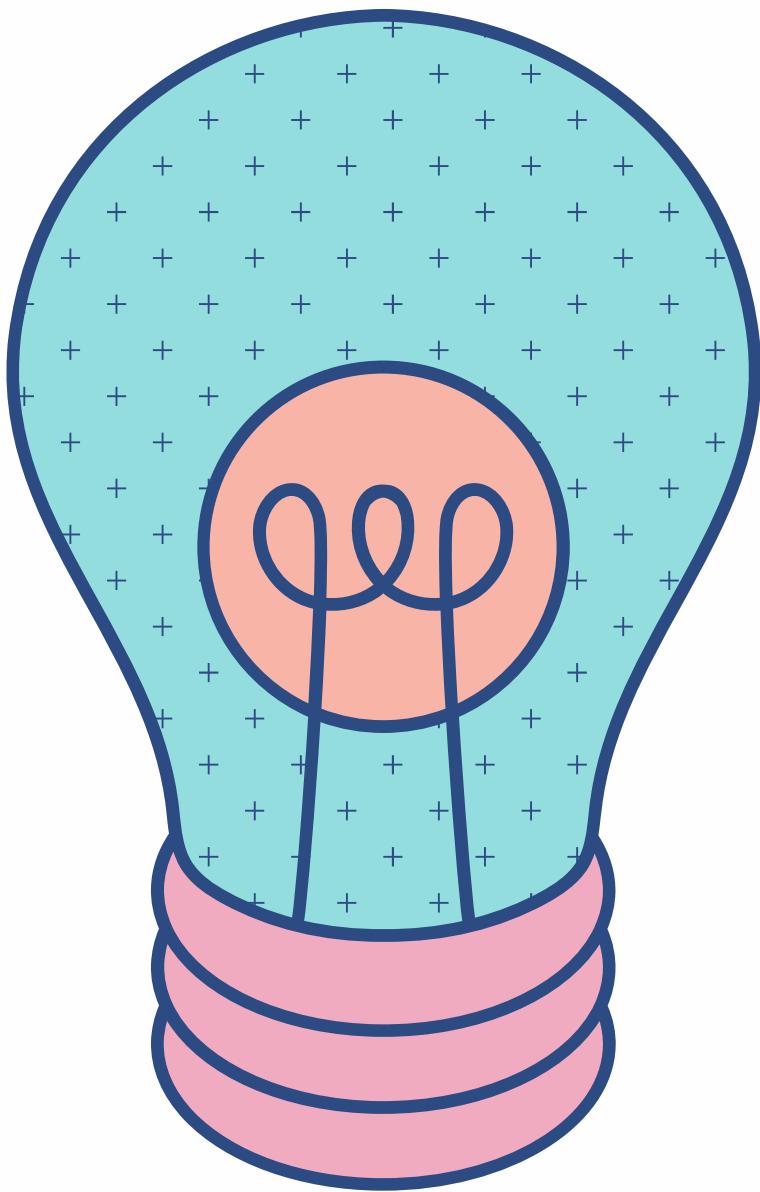
FINAL OUTPUTS

Athena-generated CSVs:

- Average Rating by Room Type.csv
- Top Reviewed Listings.csv
- High Price Low Rating.csv
- Clean table outputs from Glue Crawler

Challenges and Resolutions

Challenge	Resolution
Lambda IAM issues	Solved by assigning correct LabRole
Docker runtime errors	Resolved via manual install
Missing Python packages	<ul style="list-style-type: none">During initial model experimentation in the Jupyter Notebook hosted on EMR, we encountered errors related to missing essential Python libraries such as scikit-learn, xgboost, and pandas.These libraries were not pre-installed in the Docker-based environment we had configured within the EMR master node.We accessed the Docker container's shell environment and used the pip package manager to manually install the required libraries.After installation, we verified that each package was correctly recognized and usable by the active Jupyter kernel.This process was essential for ensuring that the ML code could execute without interruption and remain compatible with the Spark and Hadoop dependencies present on the EMR cluster.
Schema inconsistency	Cleaned during preprocessing phase. Described using lambda_code.py and emr_setup.docx

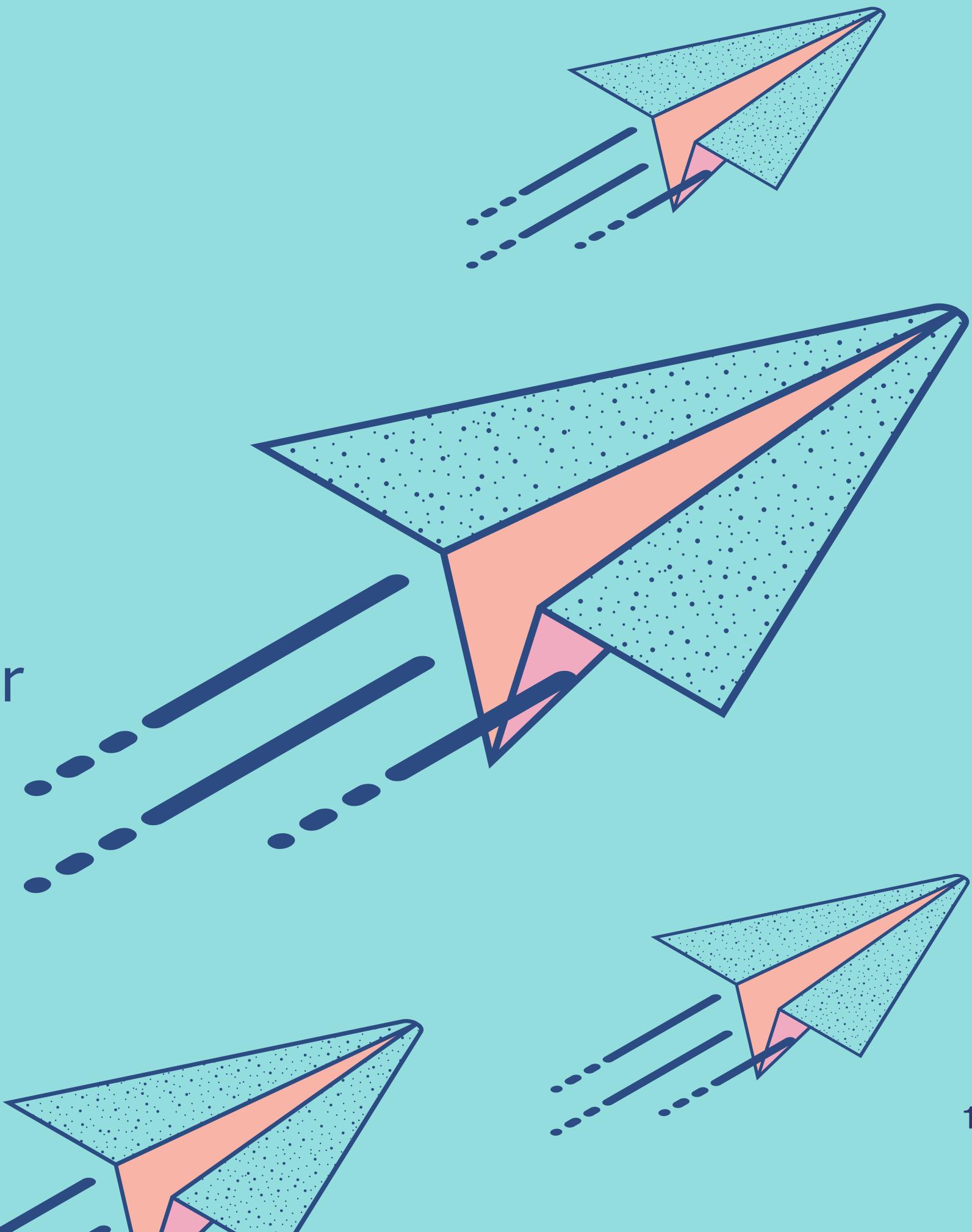


FINAL TAKEAWAYS

- ✓ Fully automated AWS pipeline for structured insights
- ✓ Integrated SQL, Serverless, and ML in one workflow
- ✓ Production-ready setup with scalability potential

Future Enhancements

- Add Stream Ingestion via Kinesis
- Use SageMaker for model training
- Add BI Layer: Amazon QuickSight for real-time dashboards





THANK YOU!