Cloud Computing Technology (UEC634)

Lab Report

Experiment-8

Submitted by:-

Aarushi Agarwal	102215273
Akshi Sharma	102215183
Diya Goyal	102215255
Jatin Chhabra	102215309

Submitted to:-

Dr. Geetanjali

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY, PATIALA, PUNJAB, INDIA

Group Leader:- Aarushi Agarwal (102215273)

Aim- To demonstrate the setup and utilization of Amazon Athena for serverless SQL querying of data stored in Amazon S3.

Theory

Amazon Athena is a serverless, interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL. It eliminates the need for complex extract, transform, and load (ETL) processes by allowing users to query data in its original storage location. Athena is built on an open-source Presto distributed SQL engine and leverages a massively parallel processing architecture to deliver fast query performance regardless of data size.

Architectural Overview

Athena operates on a unique serverless architecture where computing resources are allocated dynamically for each query and released once execution completes. This differs significantly from traditional data warehousing solutions that require provisioning and maintaining dedicated compute clusters. The service integrates directly with the AWS Glue Data Catalog, which stores metadata about the data sources and schema information required for query processing.

Query Processing Pipeline

When a user submits a query to Athena:

1. The service parses and validates the SQL syntax
2. Generates an optimized execution plan
3. Allocates distributed compute resources to process the query
4. Reads data directly from S3 using schema definitions from the Glue Data Catalog
5. Processes the data using MPP (Massively Parallel Processing) techniques
6. Returns results to the user and stores them in a designated S3 location

Performance Optimization

Query performance in Athena can be optimized through several techniques:

- Partitioning data based on commonly filtered columns (date, region, category) ● Using columnar storage formats (Parquet, ORC) instead of row-based formats

- Compressing data to reduce I/O operations
- Bucketing data for join-intensive workloads
- Converting large numbers of small files into fewer, larger files

Security and Governance

Athena integrates with AWS IAM for access control, AWS KMS for encryption, and AWS Lake Formation for fine-grained data access. This allows organizations to implement comprehensive security policies while maintaining data accessibility for analysis.

Data Lake Integration

Athena is a key component in modern data lake architectures, enabling SQL-based analysis on data lakes without moving data. This approach:

- Maintains a single source of truth for data
- Reduces data duplication and associated storage costs
- Minimizes data transfer latency
- Simplifies data governance by centralizing access controls
- Allows for schema-on-read flexibility

Supported Data Formats

Athena supports querying data in various formats including:

- CSV and TSV (comma and tab-separated values)
- JSON (JavaScript Object Notation)
- Parquet and ORC (columnar storage formats)
- Avro (row-based format)
- Text files with custom delimiters
- Apache logs and other semi-structured formats

Key Advantages

Key advantages observed include:

- Simplified setup compared to traditional data warehousing solutions
- Pay-per-query pricing model that scales with usage
- Native integration with the AWS ecosystem, particularly S3 and Glue
- Support for standard SQL and various data formats
- Quick time-to-insight for data analysis tasks

Data Storage Organization

- **Separation of Source Data and Query Results**: Two different folders should be created—one for the original dataset and another for query results—to avoid overlapping outputs and ensure clean separation of raw data and Athena's output files. This separation prevents Athena from recursively reading its own output files in subsequent queries.
- **Folder-Level Selection Limitation**: When configuring data sources or result locations, Athena accepts only folder paths, not specific files. This is by design, as Athena needs to manage multiple files within a location, especially for query results where each query generates a new file.
- **Query Result Configuration**: A separate folder must be used for query outputs (not the same as the original dataset path) to avoid data duplication and recursive reading of result files in future queries, which could lead to errors or inflated costs from scanning unnecessary data.

Amazon Athena vs. Traditional SQL Engines

Feature	**Amazon Athena**	**Traditional SQL Engines**
Deployment Model	Serverless (No infrastructure to manage)	Requires manual setup on servers/machines

Data Storage	Queries data directly from Amazon S3	Stores data internally in a local database
Cost Model	Pay-per-query (per TB scanned)	Licensing or subscription-based; fixed or per-instance

Steps (along with Snapshots):-

Open **Amazon S3** and click on **Create Bucket**

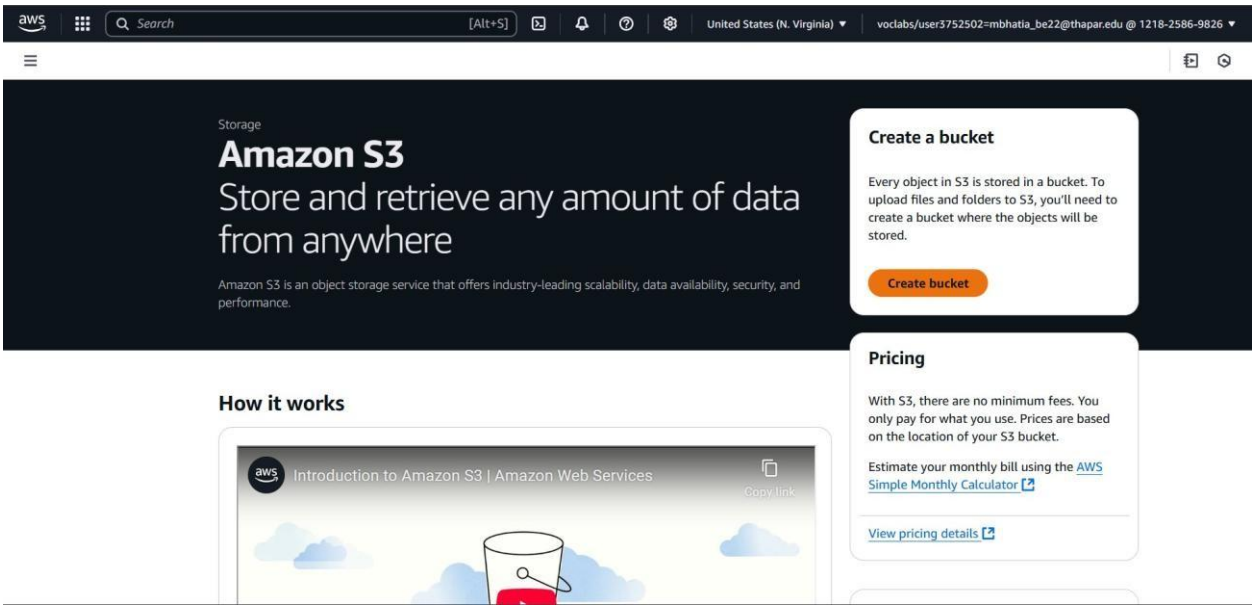


Figure 1: Amazon S3

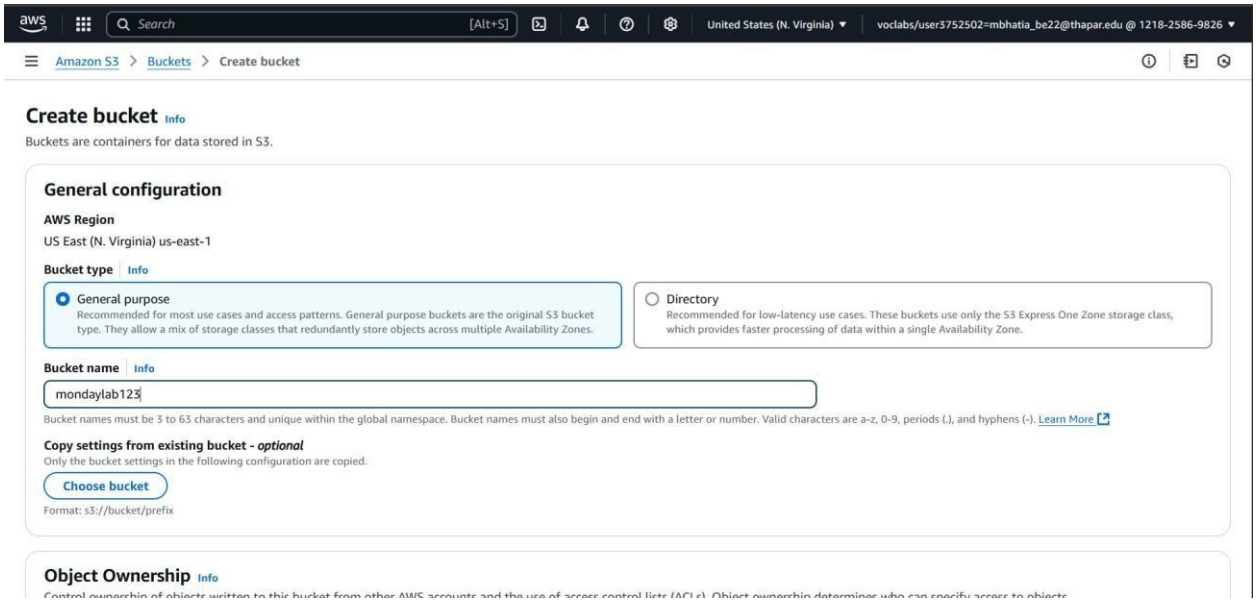


Figure 2: Creating a bucket Bucket created successfully

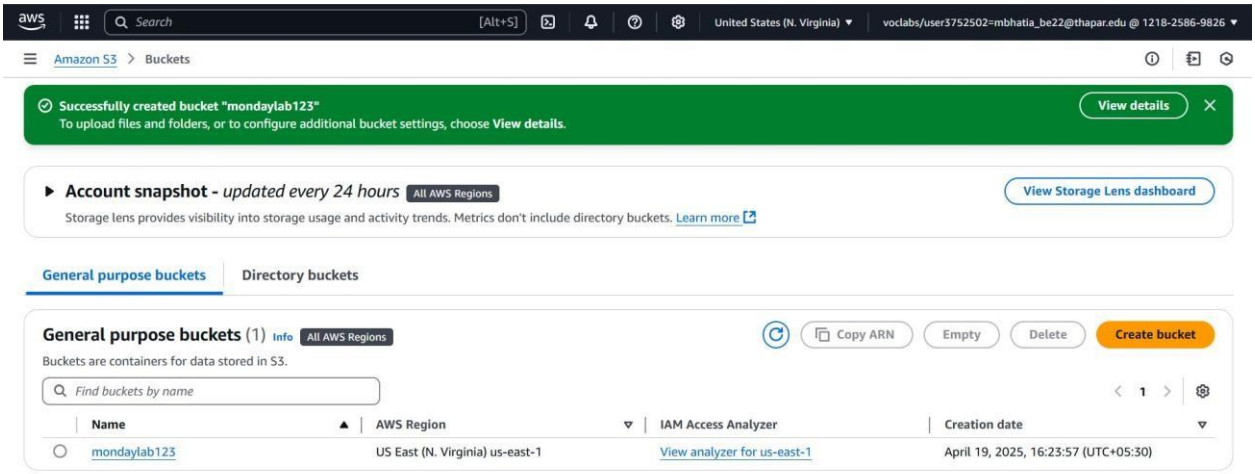


Figure 3: Click on the created bucket

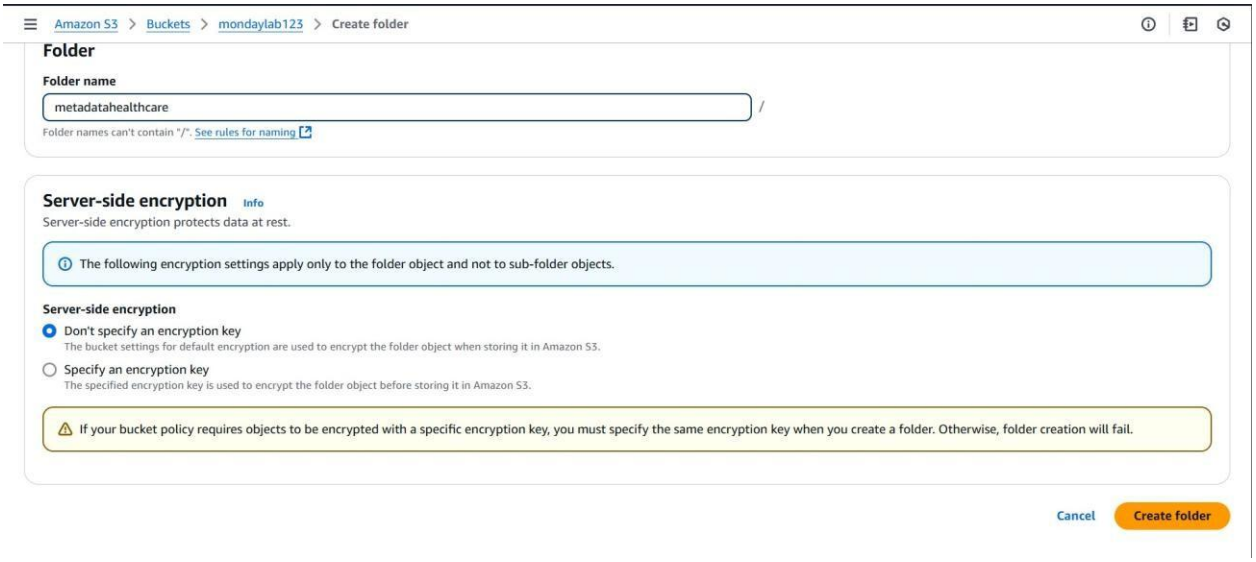


Figure 4: Create a folder for metadata

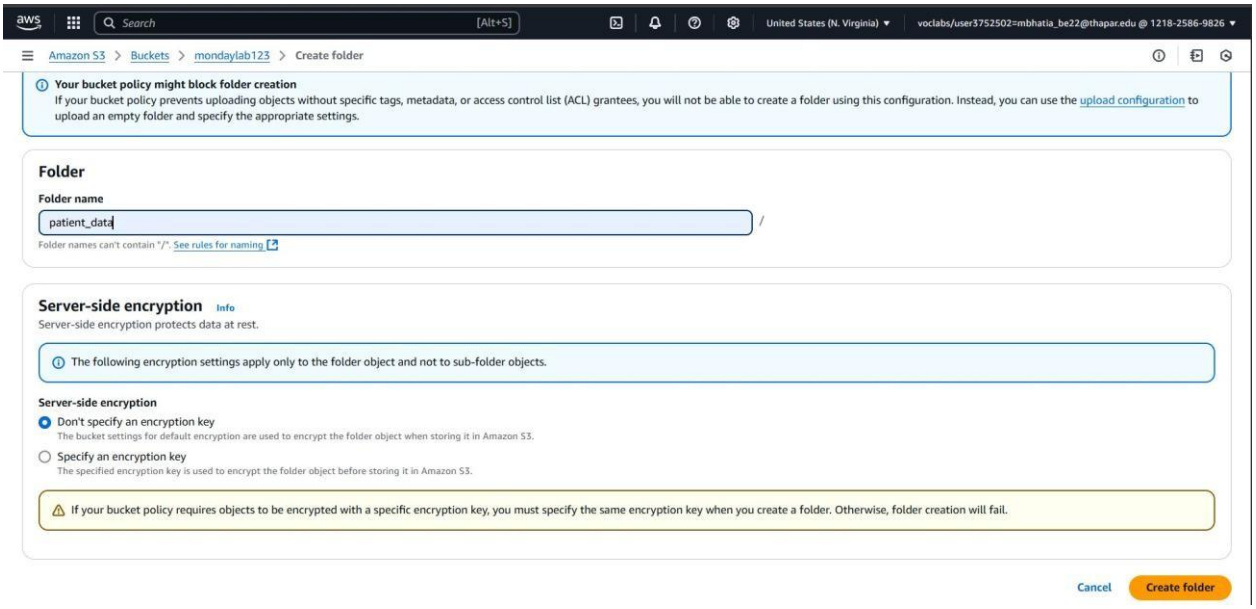


Figure 5: Create another folder for original data

Both folders created successfully

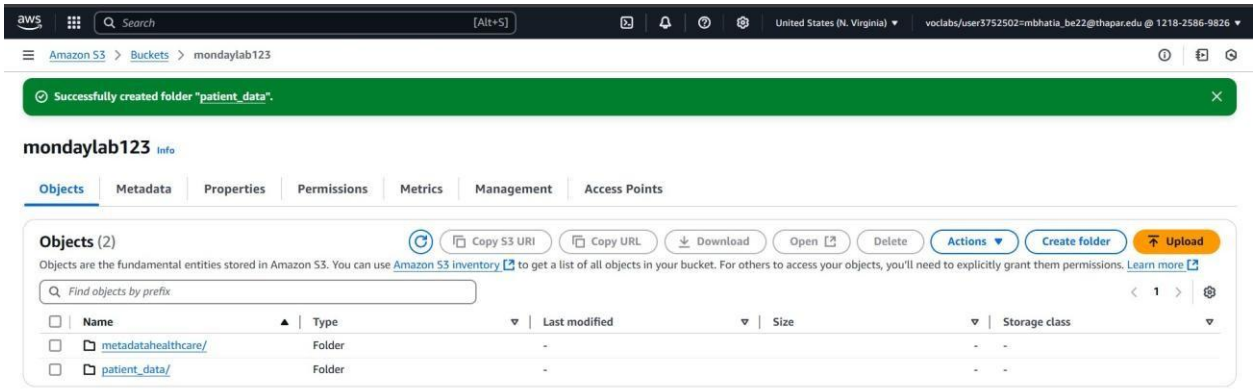


Figure 6: click on patient_data

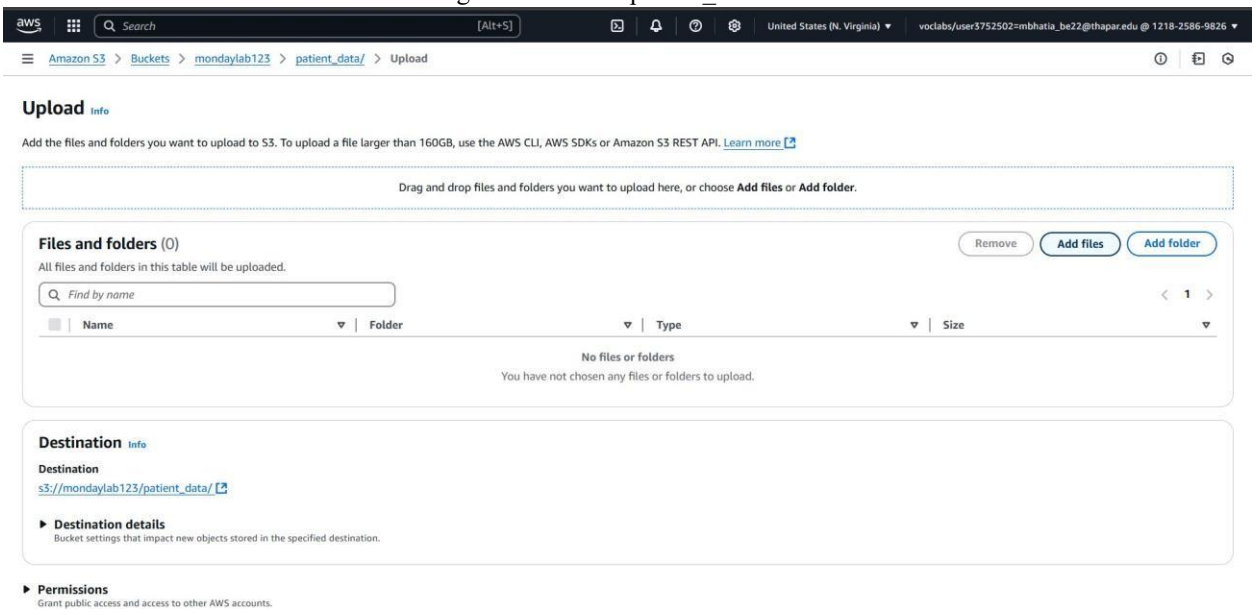


Figure 7: click on add files and add the .csv file

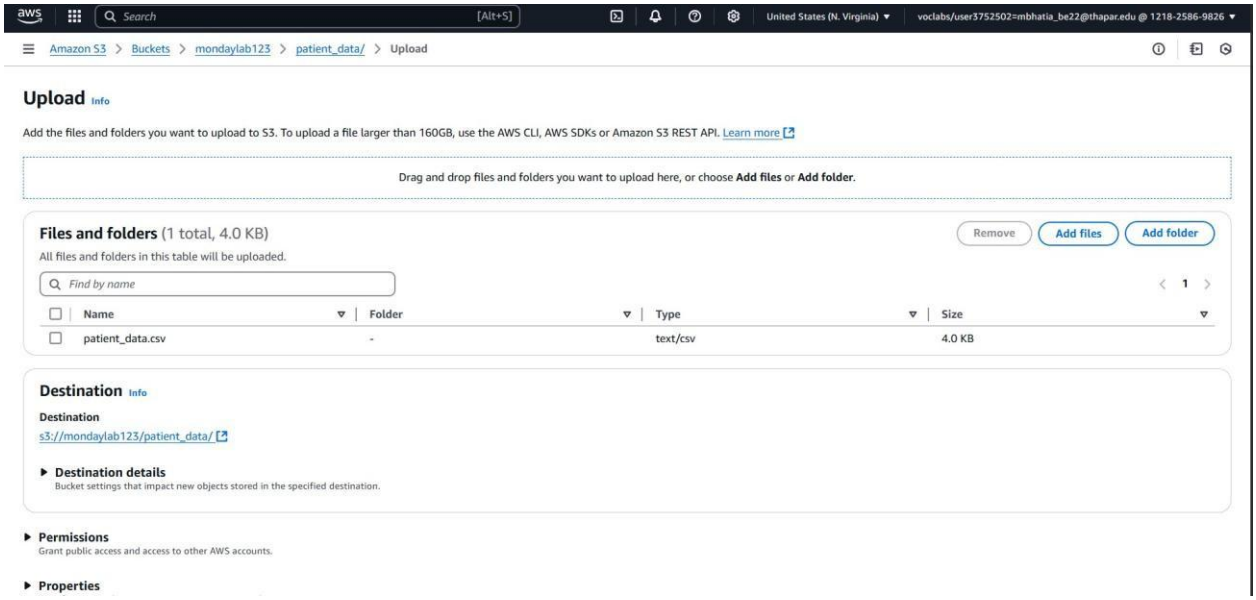


Figure 8: Click on upload

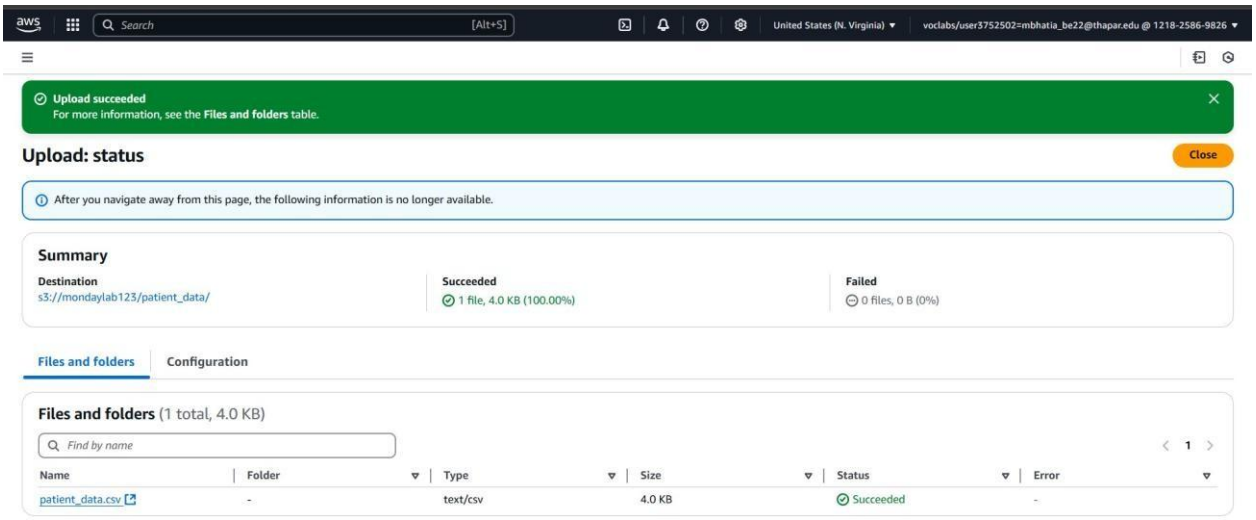


Figure 9: File uploaded successfully

	A	B	C	D	E	F	G	H	I
1	PatientID	Age	Gender	Diagnosis	BMI	Smoker	BloodPres	CholesterolLevel	
2	1	69	Female	Hypertens	31.8	No	112.7	208.3	
3	2	32	Female	None	37.7	No	123.1	208.4	
4	3	89	Male	Hypertens	23.4	Yes	133.3	126.4	
5	4	78	Male	Diabetes	24	Yes	113	173	
6	5	38	Male	Hypertens	17.8	No	110.7	196.7	
7	6	41	Male	Diabetes	31	No	114.7	198.3	
8	7	20	Male	None	31.5	No	134.3	207.8	
9	8	39	Male	None	20.7	No	141.8	209.1	
10	9	70	Female	Diabetes	28.1	No	125.4	245.5	
11	10	19	Male	Asthma	31.1	Yes	128.7	209.4	
12	11	47	Male	Diabetes	26.1	No	110.6	190.6	
13	12	55	Male	Asthma	29.2	Yes	133.2	152	

Figure 10: patient_data.csv

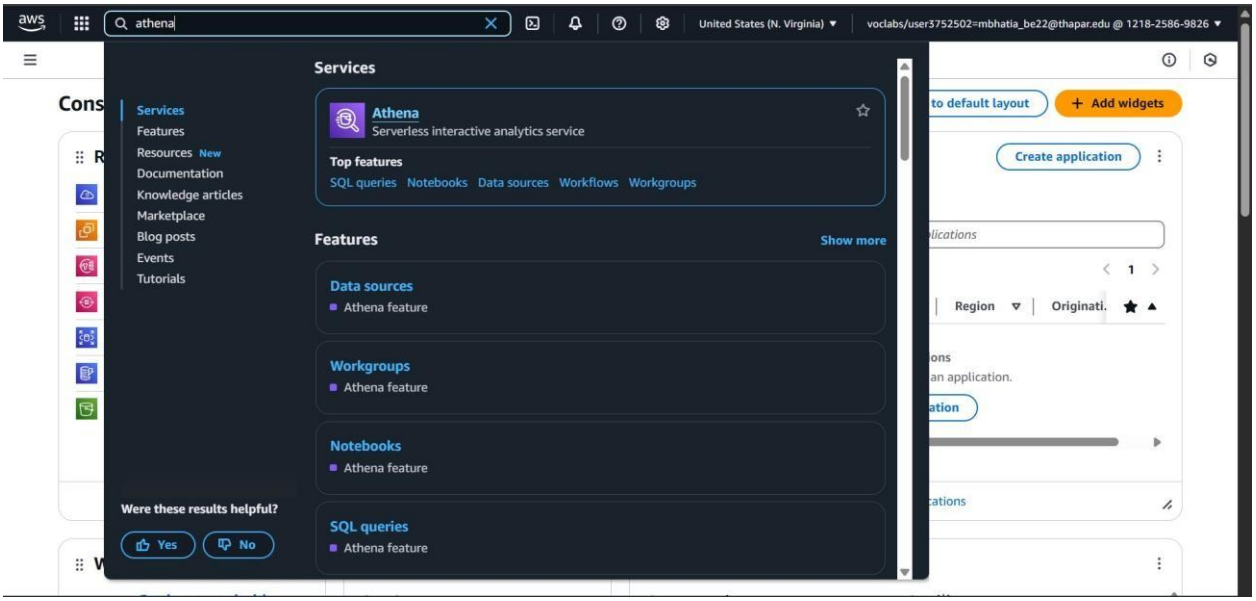


Figure 11: Go to Athena

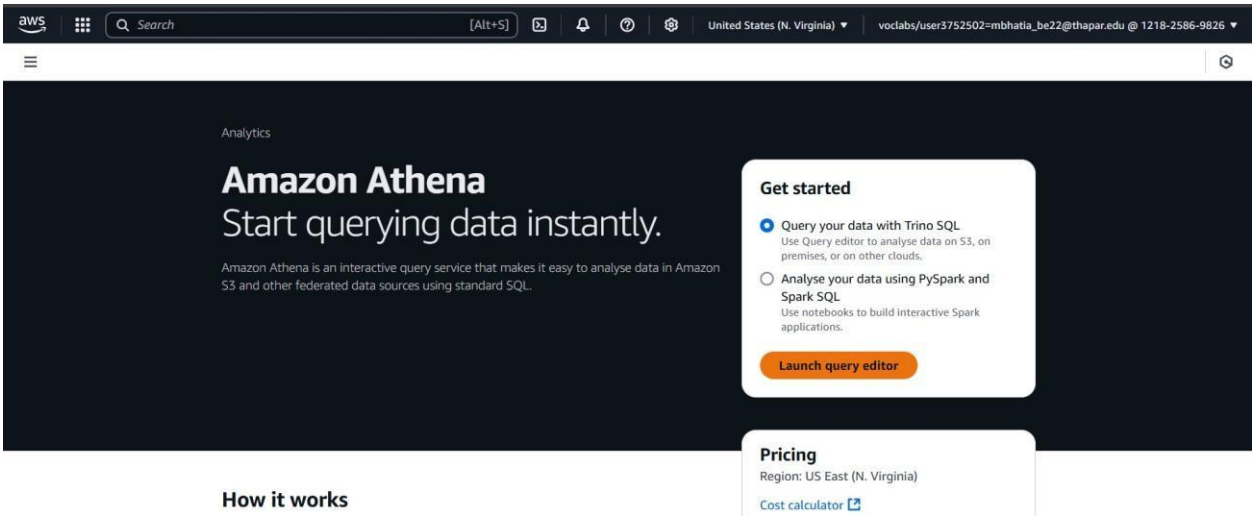


Figure 12: Click on Launch query editor

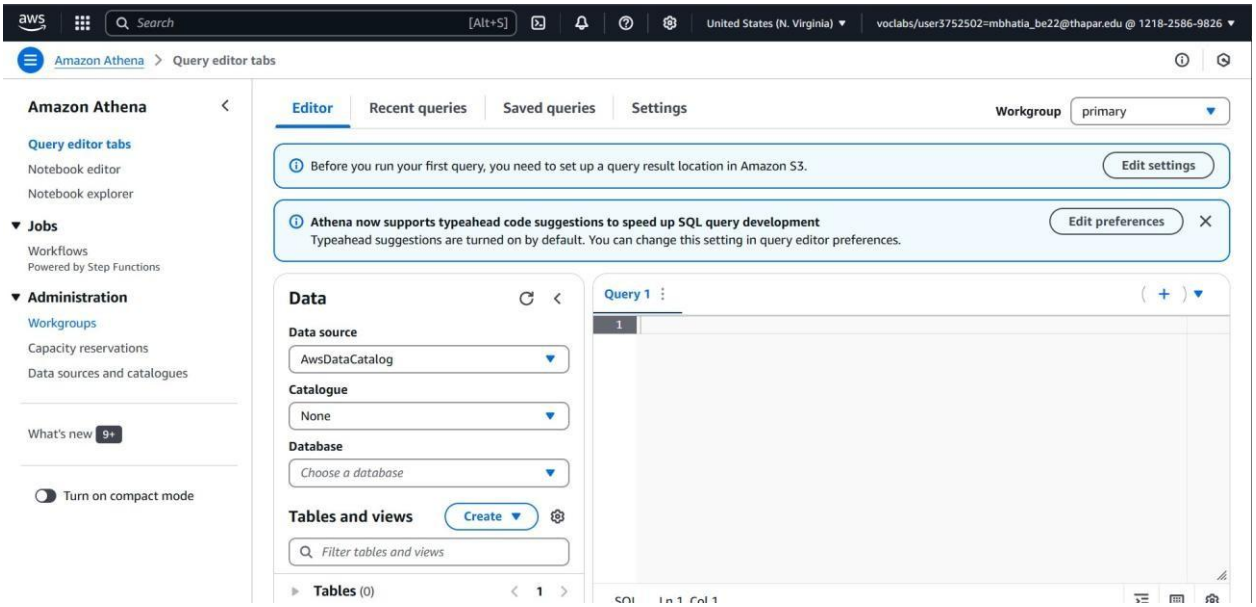


Figure 13: In the side panel go to Workgroups

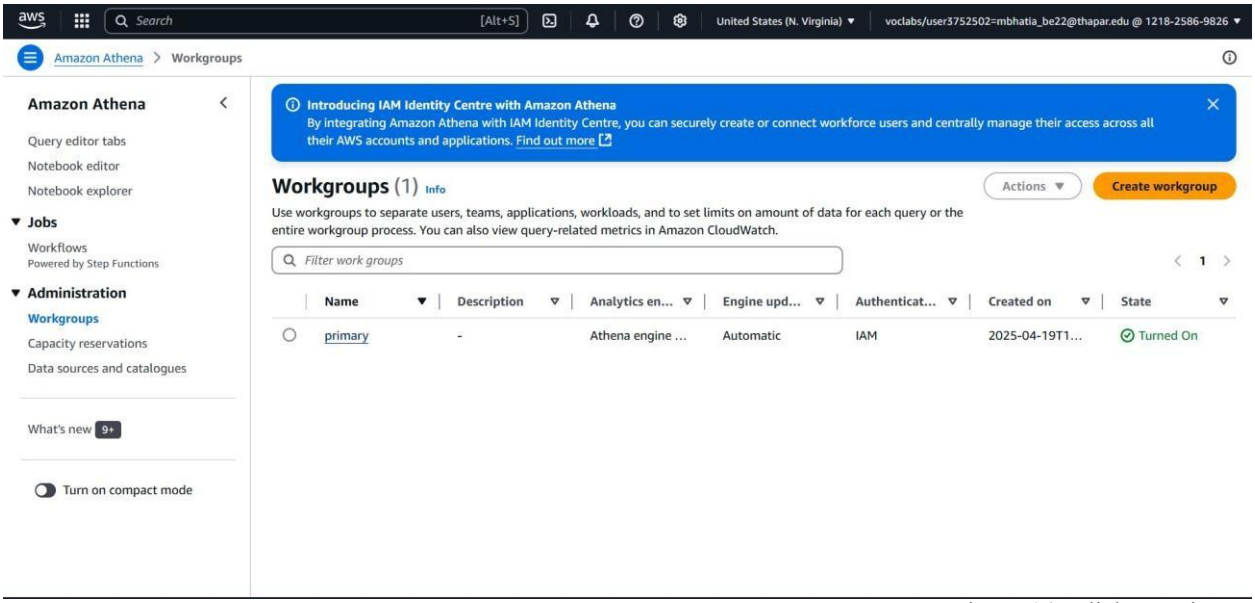


Figure 14: Click on primary

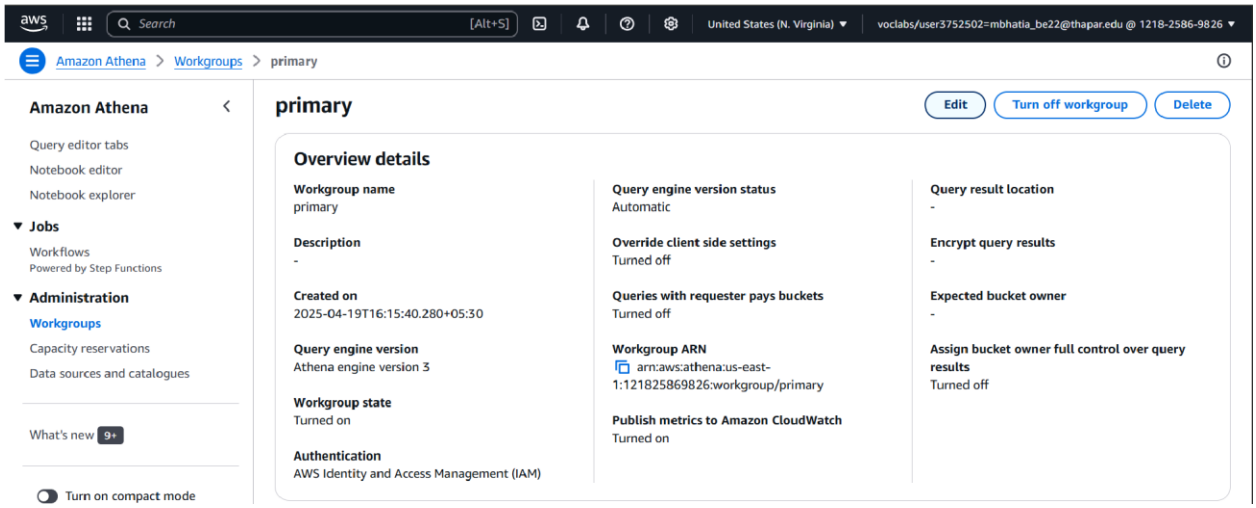


Figure 15: Click on Edit

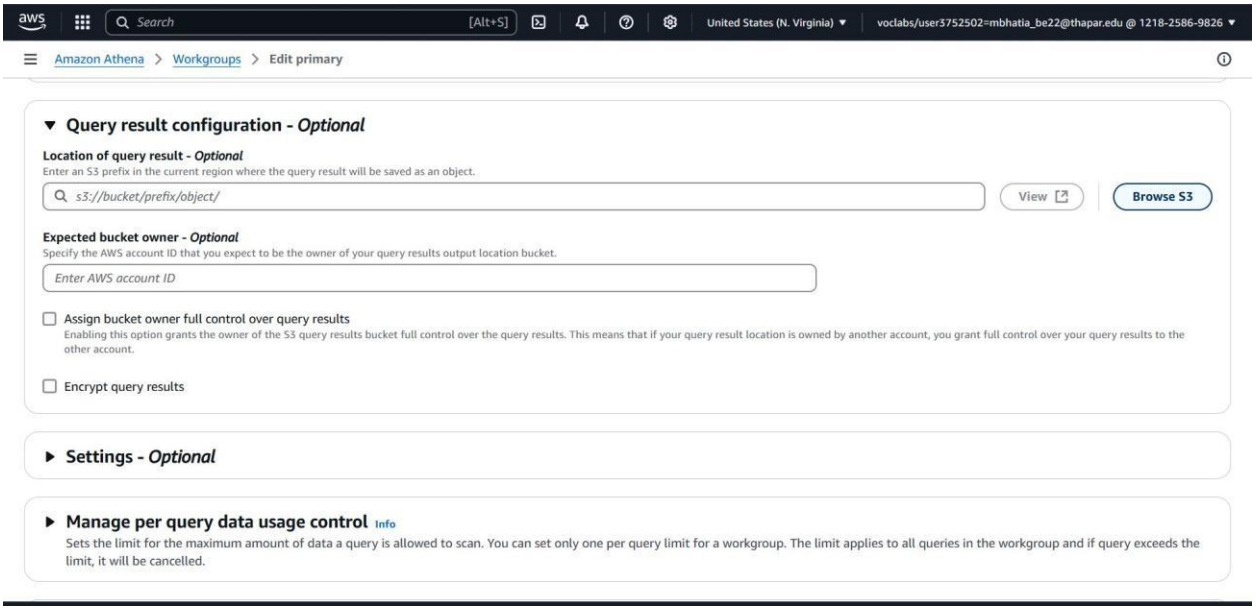


Figure 16: Under Query result config, click on Browse S3

Figure 17: Select the metadata folder in the bucket

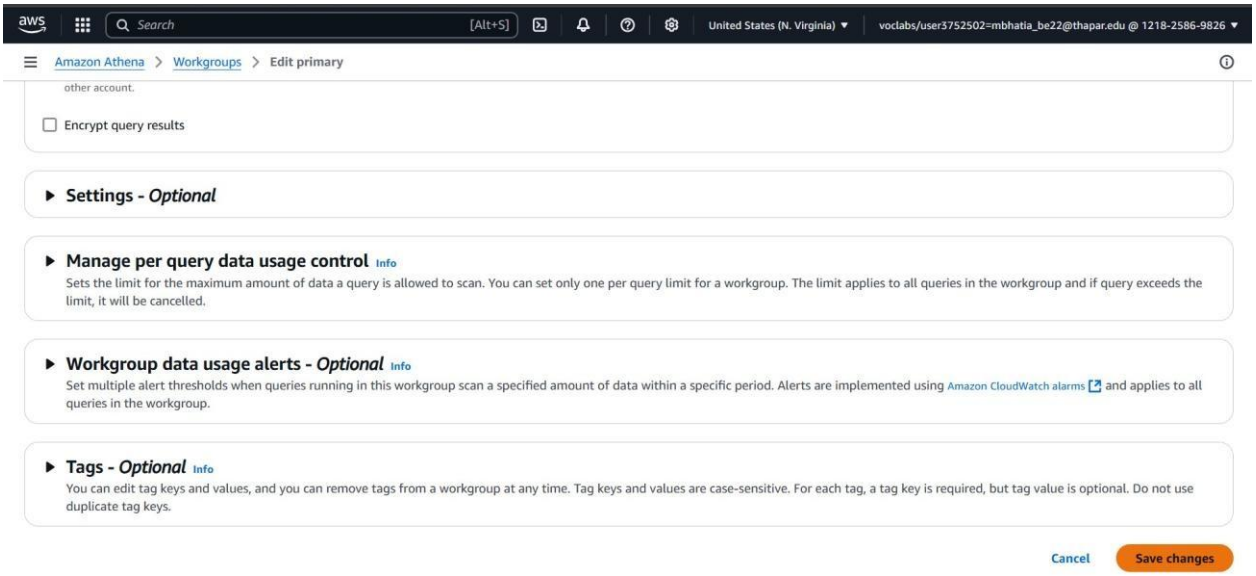


Figure 18: Click on save changes

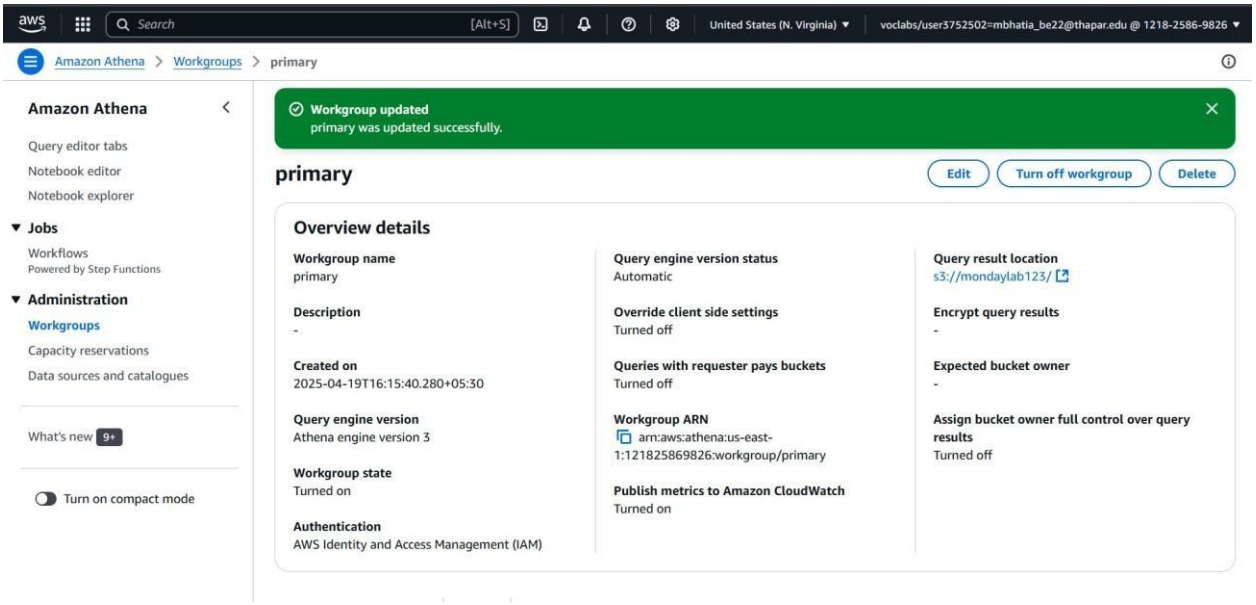


Figure 19: Go to query editor in the side panel

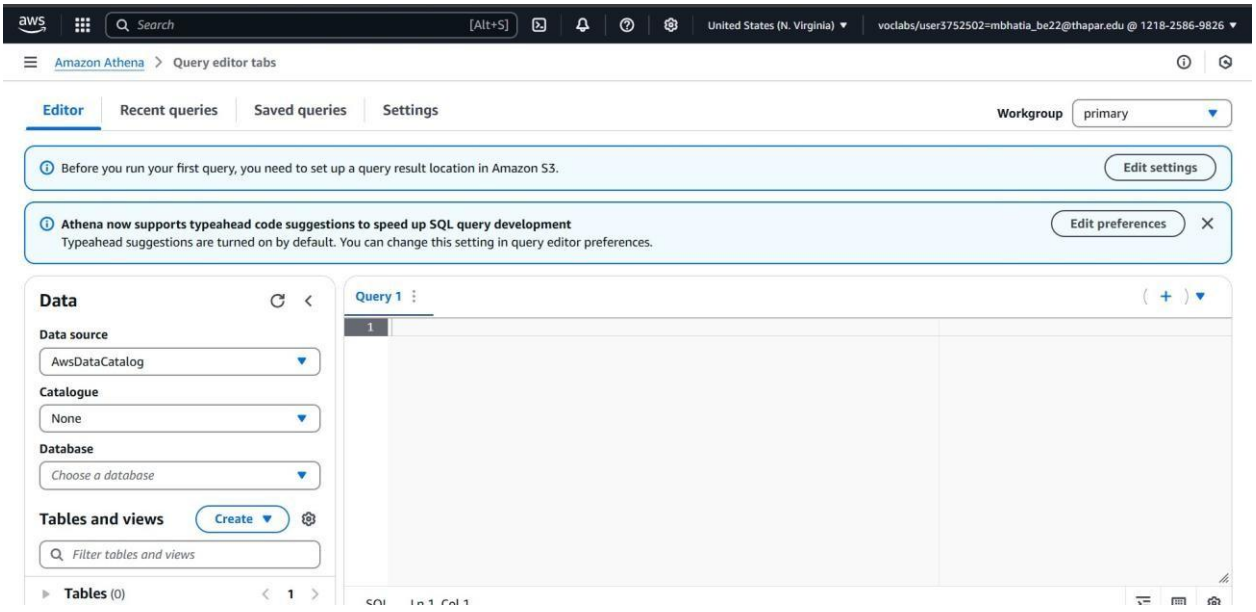


Figure 20: Click on create

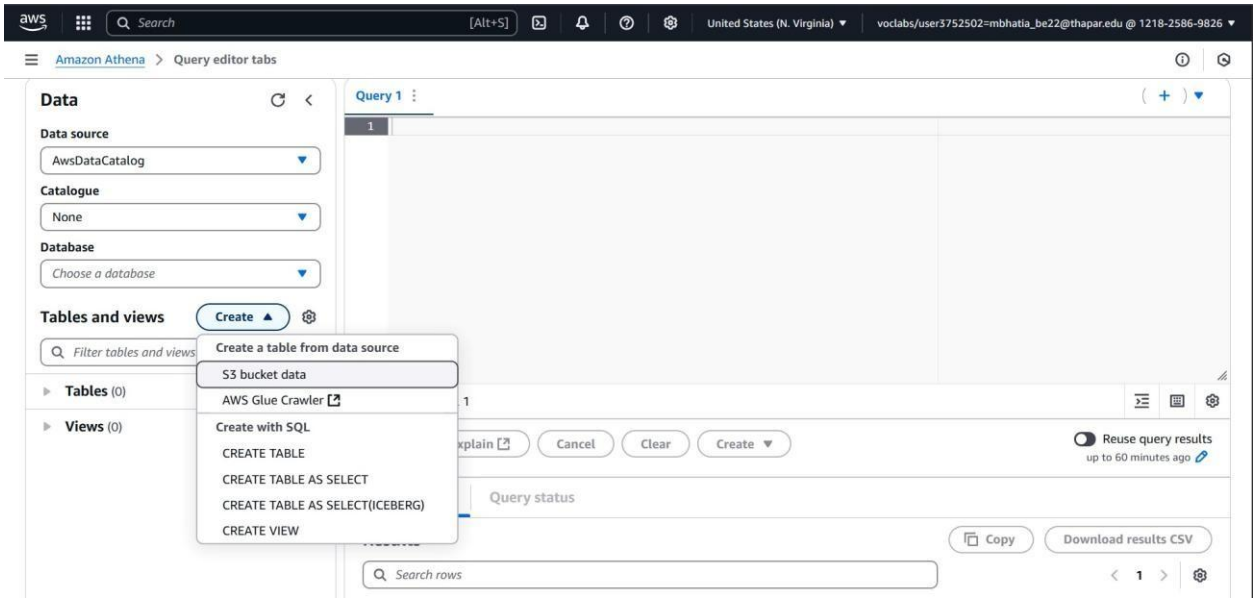


Figure 21: Choose S3 bucket data

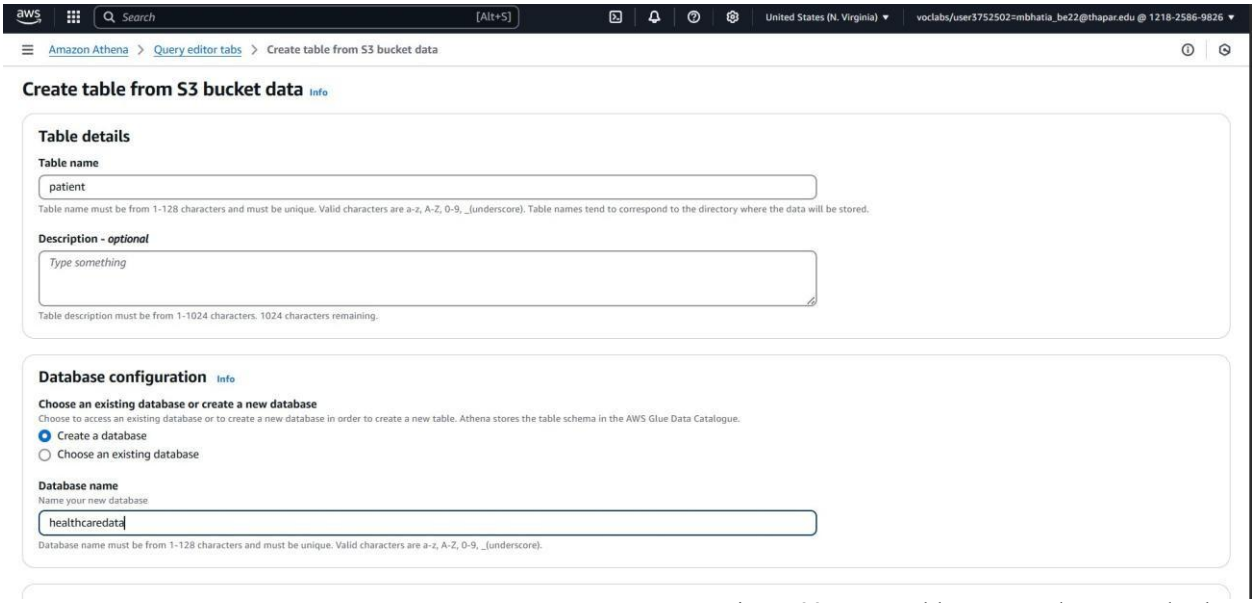


Figure 22: Enter table name and create a database

Figure 23: Specify S3 location for source data

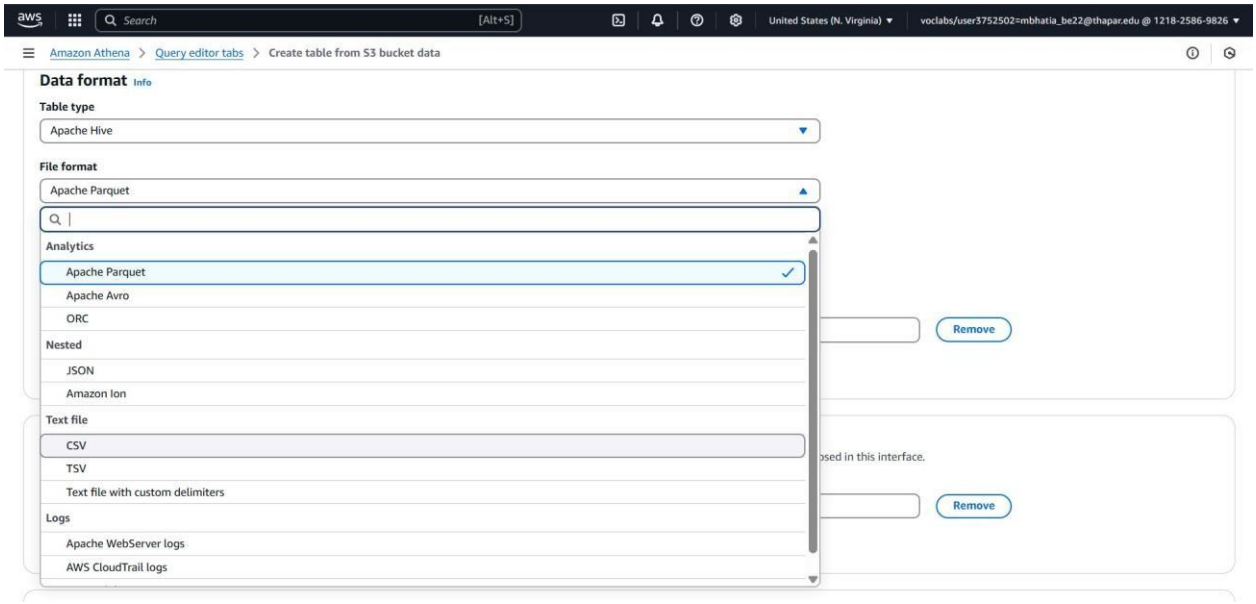


Figure 24: Specify data format as CSV

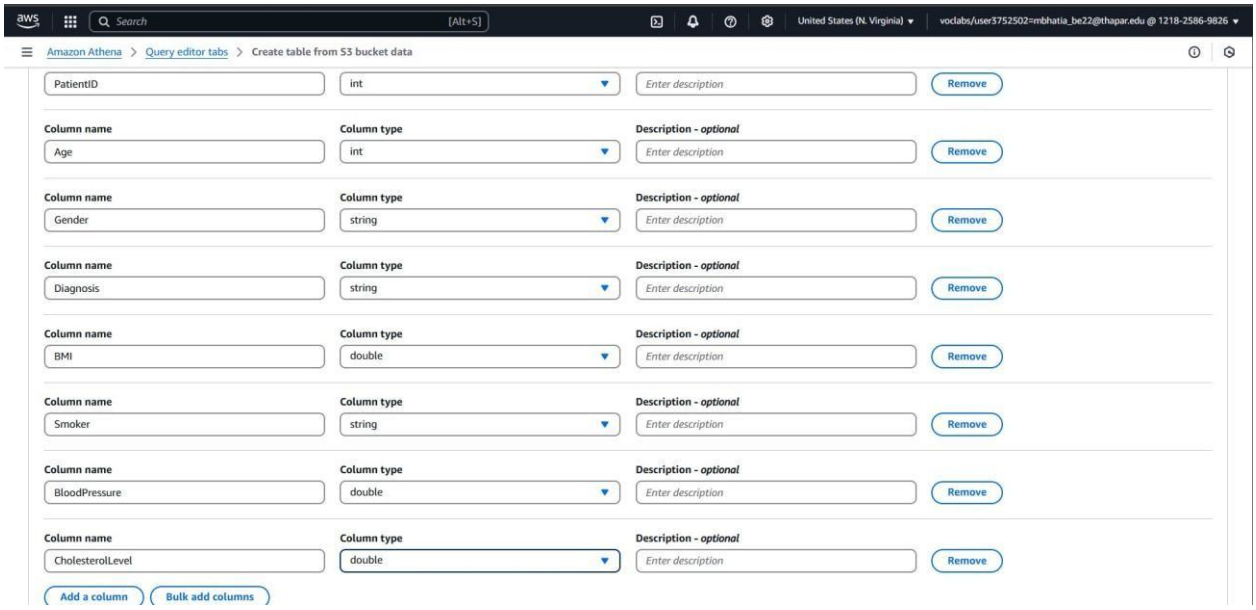


Figure 25: Define column structure

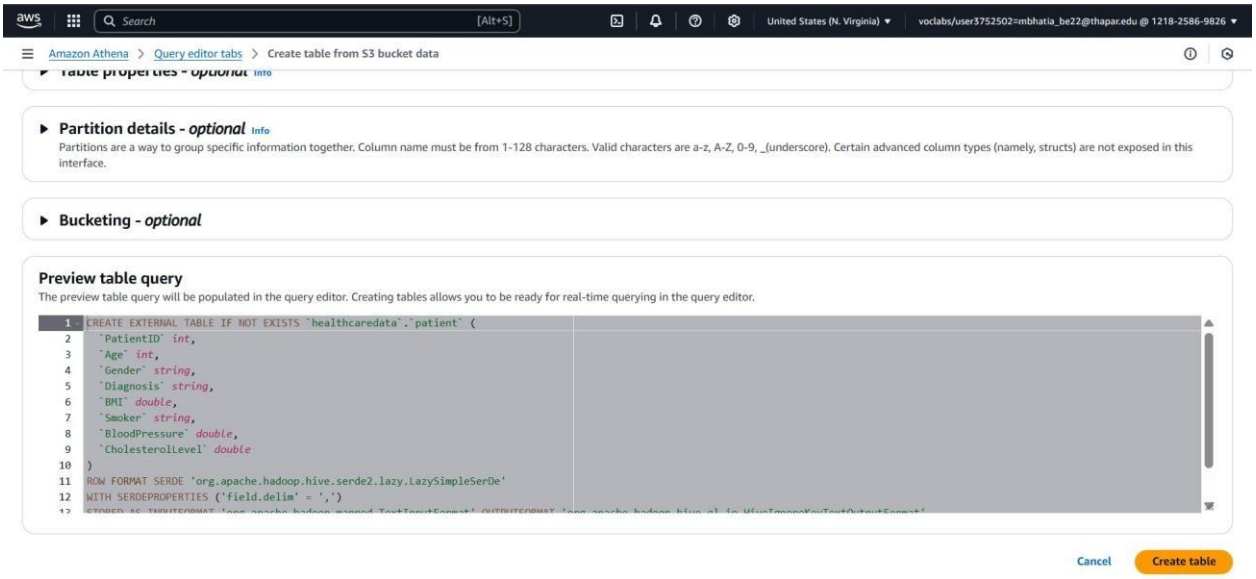


Figure 26: Click on create table

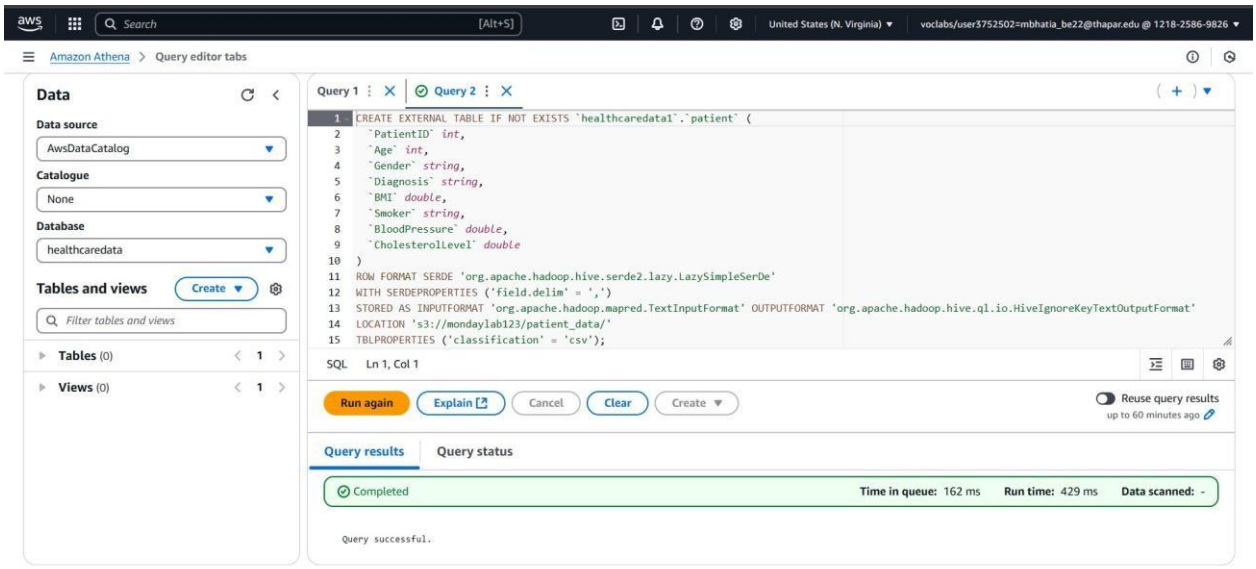


Figure 27: Table creation successful

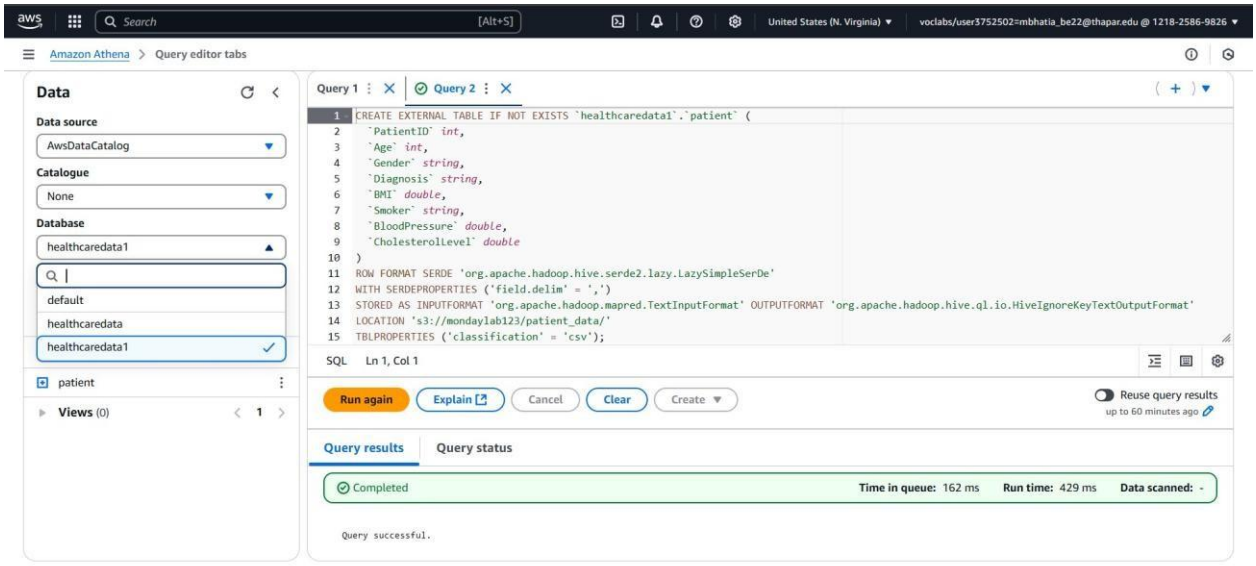


Figure 28:Click on refresh and select database

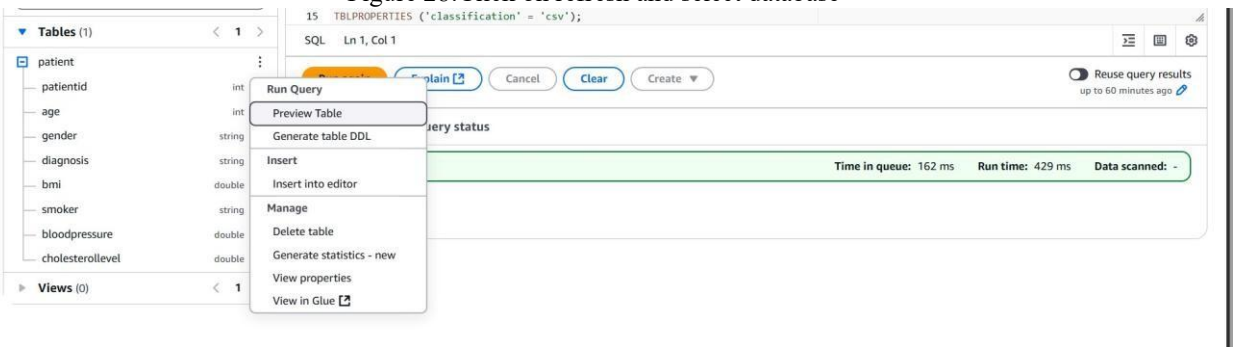


Figure 29: Click on three dots and select preview table

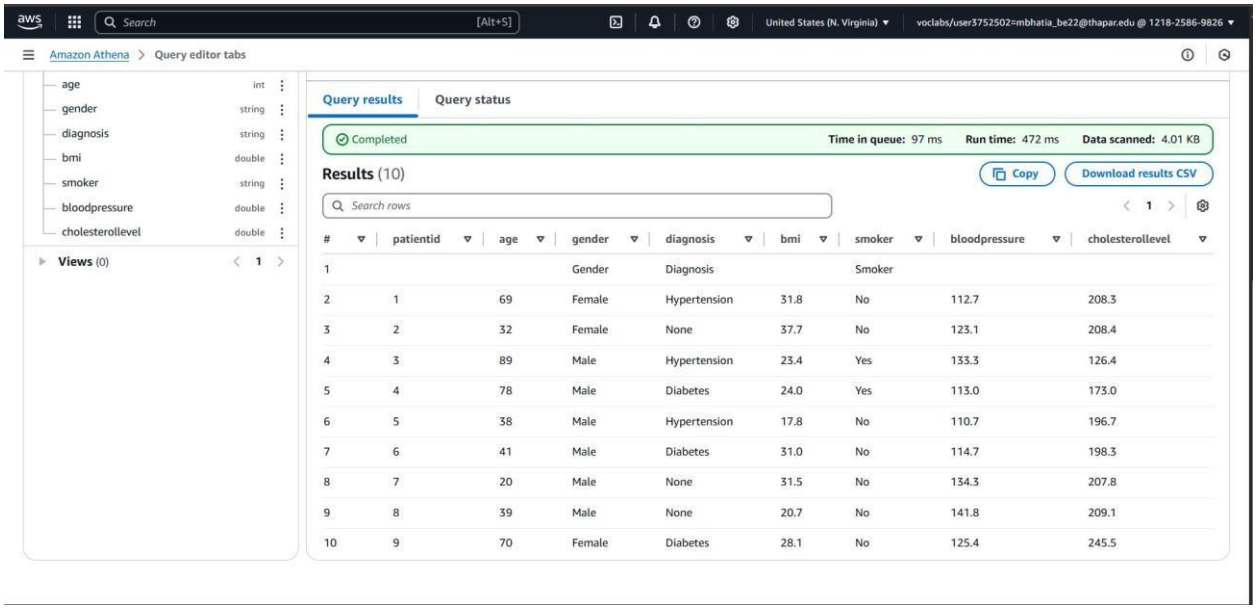


Figure 30: Results display showing patient records from the CSV file

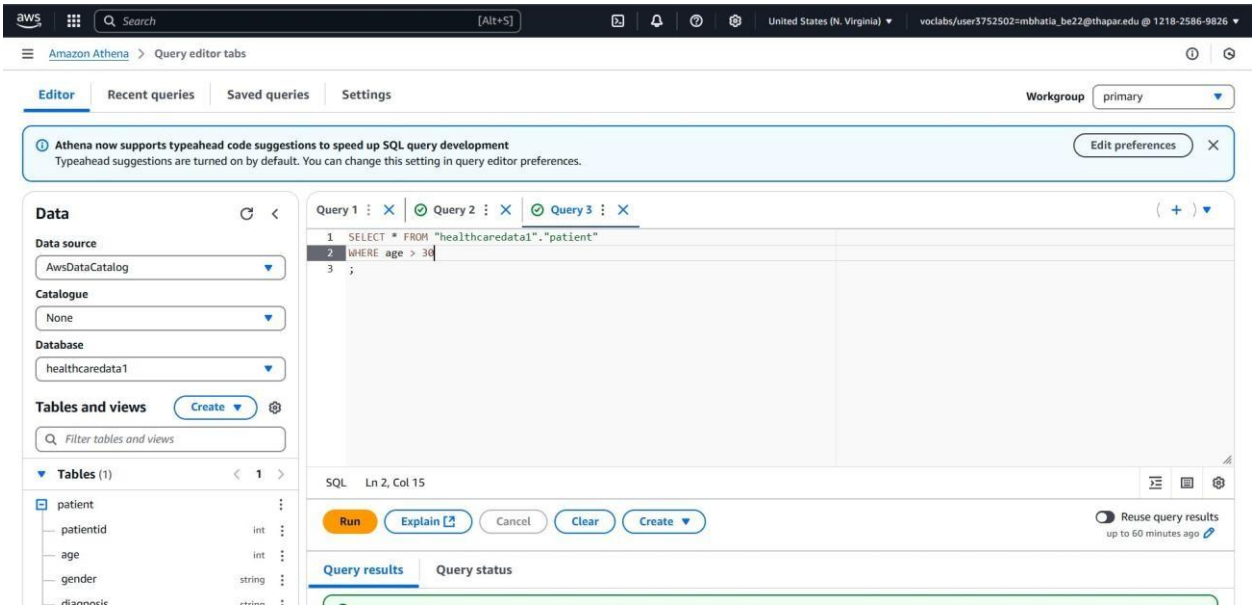
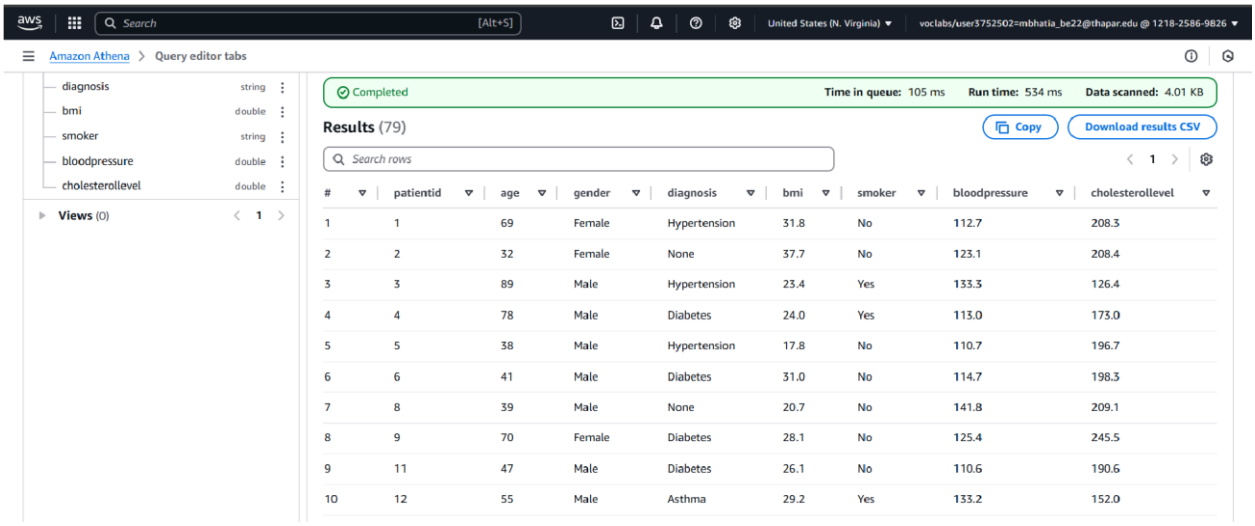


Figure 31:Writing more complex query with filtering conditions



#	patientid	age	gender	diagnosis	bmi	smoker	bloodpressure	cholesterollevel
1	1	69	Female	Hypertension	31.8	No	112.7	208.3
2	2	32	Female	None	37.7	No	123.1	208.4
3	3	89	Male	Hypertension	23.4	Yes	133.3	126.4
4	4	78	Male	Diabetes	24.0	Yes	113.0	173.0
5	5	38	Male	Hypertension	17.8	No	110.7	196.7
6	6	41	Male	Diabetes	31.0	No	114.7	198.3
7	8	39	Male	None	20.7	No	141.8	209.1
8	9	70	Female	Diabetes	28.1	No	125.4	245.5
9	11	47	Male	Diabetes	26.1	No	110.6	190.6
10	12	55	Male	Asthma	29.2	Yes	133.2	152.0

Figure 32: Filtered query results showing specific patient information

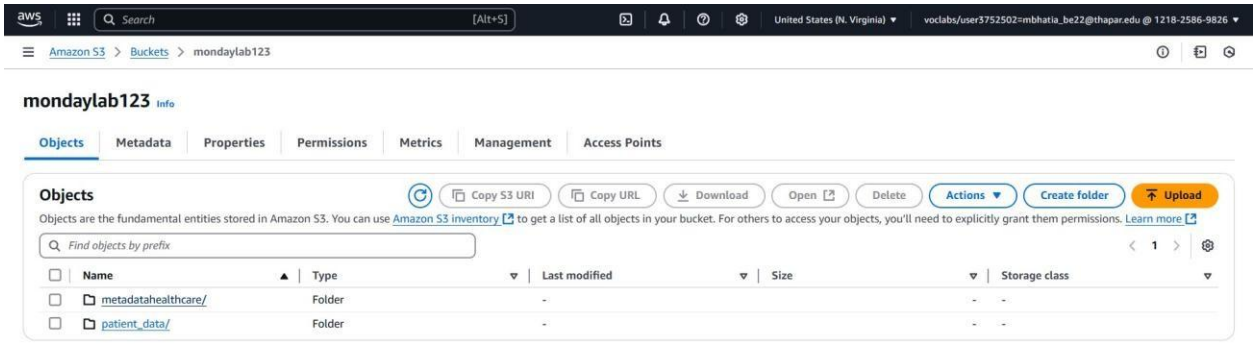


Figure 33: Go to bucket and open metadata folder

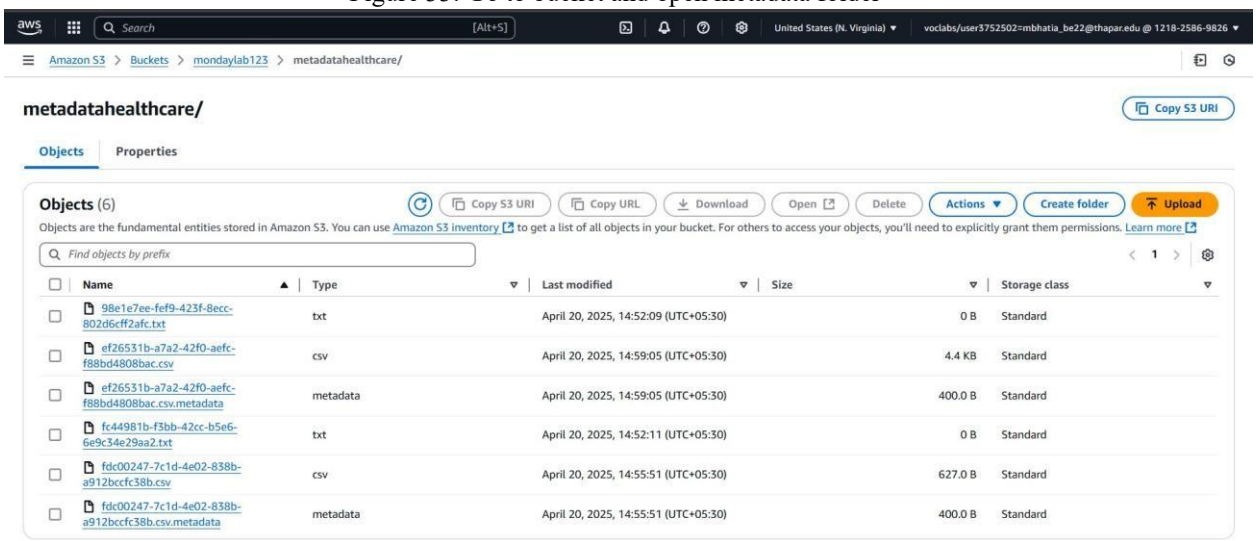


Figure 34: Folder containing all the metadata

Results

1. Successfully created and configured an Athena database and connection to S3 data sources.

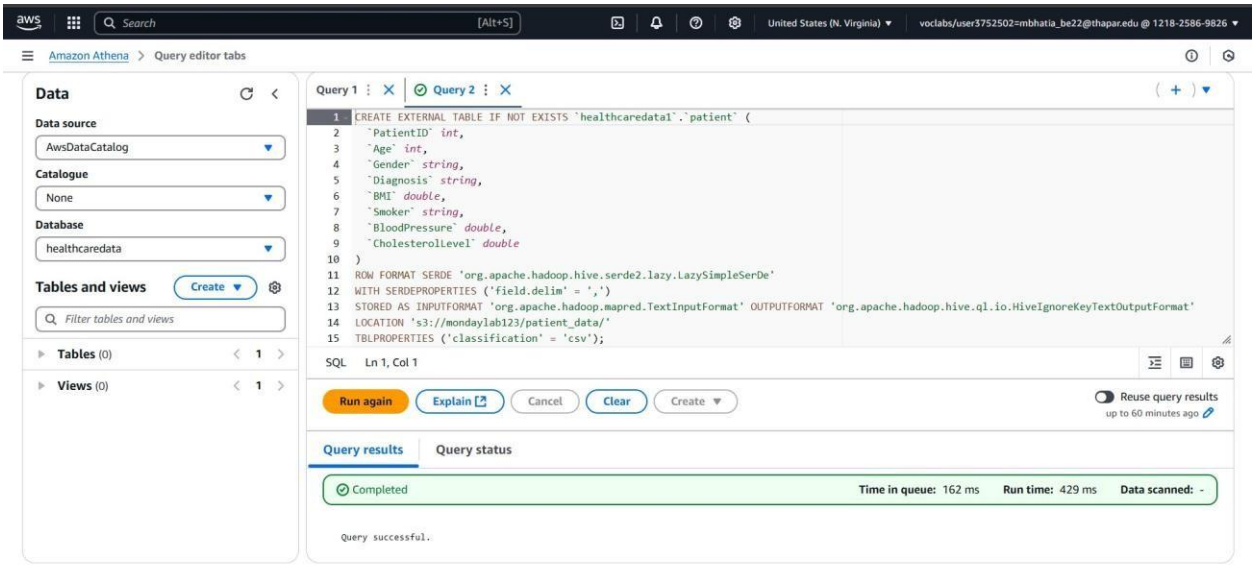


Figure 35: Database connected successfully

2. Defined table schemas that correctly mapped to the underlying data structure.

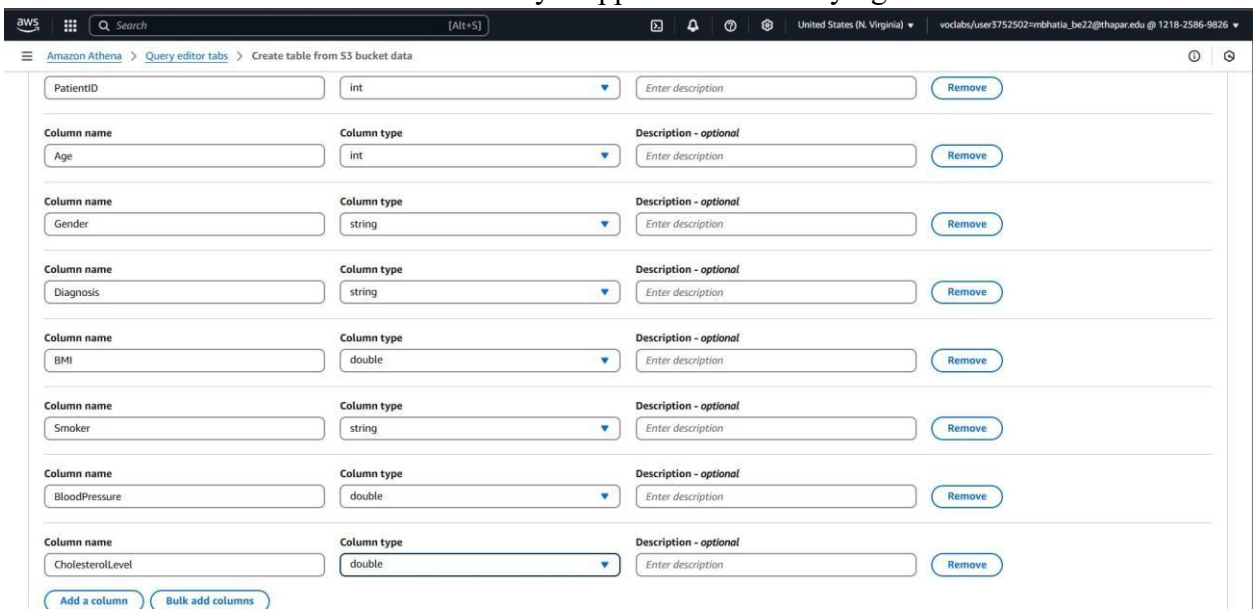


Figure 36: Define column structure

3. Executed SQL queries that returned expected results from the data stored in S3.

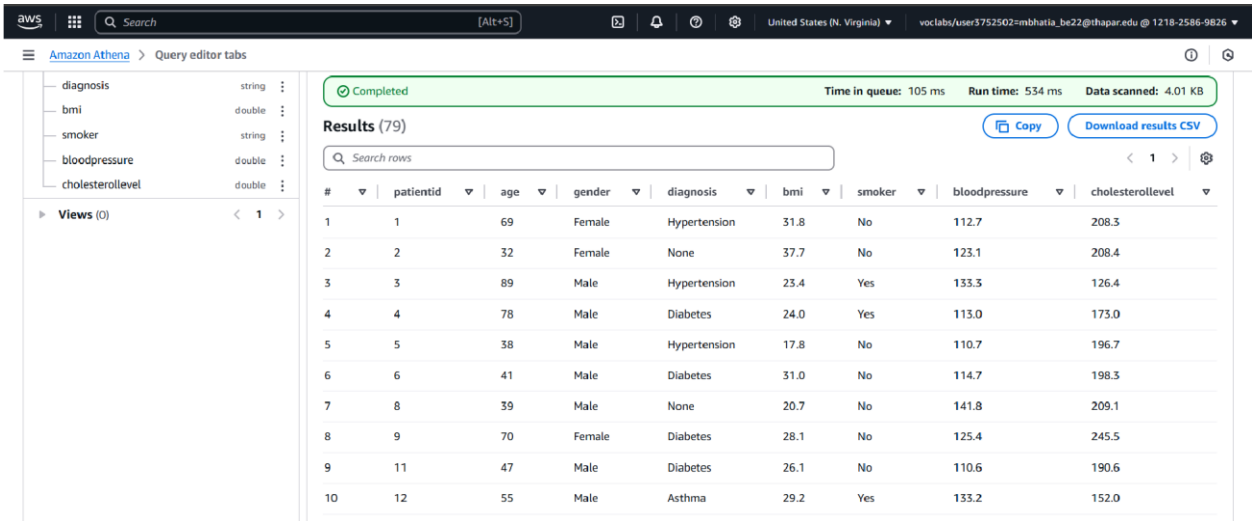


Figure 37: Filtered query results showing specific patient information

4. Created separate folder for metadata

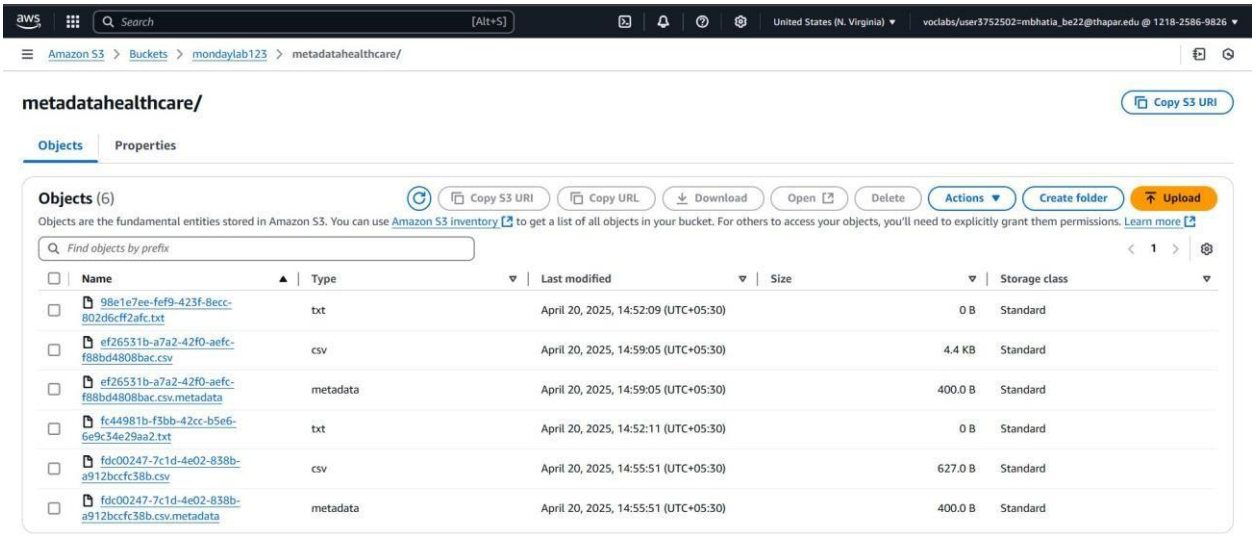


Figure 38: Metadata folder

Troubleshooting & Common Pitfalls

- Incorrect folder selection can lead to failed query execution in Athena.
- Schema mismatch during table creation may result in errors or incorrect data mapping.
- Missing query result location configuration can cause Athena to throw execution errors.

Conclusion

Amazon Athena provides a powerful, cost-effective solution for analyzing data stored in S3 without the need for complex data warehouse infrastructure. Through this lab, we demonstrated how Athena allows users to quickly set up and query large datasets using familiar SQL syntax, making it an excellent choice for ad-hoc queries, data exploration, and analytics workloads.

The serverless nature of Athena eliminates the need to provision and manage computing resources, allowing users to focus on data analysis rather than infrastructure management. This makes it particularly useful for intermittent workloads where maintaining a dedicated database would be inefficient.

Feedback:

Akshi Sharma

- Explain Use of Original Data Folder vs Metadata Folder:
  - "Two different folders were created—one for the original dataset and another for query results—to avoid overlapping outputs and ensure clean separation of raw data and Athena's output files."[Included]
    - "A separate folder is used for query outputs (not the same as the original dataset path) to avoid data duplication and recursive reading of result files in future queries."[Included]
- The report does not mention the limitation that you can only choose a folder or bucket, not an individual file, when selecting a data set for performing queries or when configuring the result location.
  - "In Athena, the create query configuration or result configuration accepts a folder path only, not a specific file. This ensures Athena can automatically create and manage result files in case of result configuration."[Included] ● Explain the reason behind each step
    - Please incorporate brief justifications after each setup step wherever possible to help readers understand why those settings were applied.

Diya Goyal

The lab report overall is well-structured, but some improvements are as follows:

1) We can emphasize the explanation of *why* each step is taken, for example, explaining why separate folders were created for metadata and original data, and why Athena requires folder-level paths instead of file-level for queries and outputs. [This explanation is already included in the 'Data Storage Organization' section.]

2) We can also include a brief section at the beginning or end of the report that outlines realworld use cases of Amazon Athena, such as log analysis, data lake querying, or ad-hoc business analytics, to help readers understand its practical relevance beyond the lab. [Included]

Jatin Chhabra

- Including small comparison table between Athena and Traditional SQL Engines can make report more understanding. Highlighting what makes Athena unique will add depth to the theory section. [Included]
- Add a Troubleshooting / Common Pitfalls Section. Including likely missteps (permissions, metadata mismatch, etc.) adds practical value. [Included] Examples you can mention:

    1. Incorrect folder selection can lead to failed query executions.
    2. Mismatched schema definitions cause table creation to fail.
    3. Forgetting to configure query result location = Athena errors.

- For consistent heading format convert some bold figure labels into proper Step titles (e.g., “Step 1: Create Bucket in S3”) for better structure.
- Highlight results more visually instead of just inline with paragraphs, maybe put final query results or key outputs inside a callout box or shaded background. [Results are already wellstructured, and shaded backgrounds may reduce the formal look of the report.]