Optimized Batch & Stream Data Processing for Enhanced Inventory Management

Submitted in partial fulfilment of the requirements for the degree of

Post Graduate Diploma in Data Engineering

by

Sudarshan P (G23AI1046)

Utkarsh Gupta (G23AI1048)

Under the guidance of

Dr. Pradip Samal

IIT Jodhpur

Indian Institute of Technology Jodhpur
Advance Data Engineering in Cloud
Trimester-3 (July 2024)

॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

End-to-End Project

https://github.com/utkarshgupta98/advance_data_engineering

Problem Statement

An e-commerce company aims to enhance inventory management and supply chain efficiency through analysis of batched sales data from multiple online platforms. The objective is to optimize inventory levels, minimize stockouts, and streamline fulfilment processes.

Real-Life Use Case: Batch Data Processing for Inventory Management

Scenario:

- **Online Sales Platforms:** Transactional data including orders, product details, and customer information.
- **Inventory Systems:** Stock levels, warehouse locations, and logistics data.
- **Supplier Information:** Lead times, pricing, and availability.

Challenges:

1. **Data Integration:**
    - Integrating batch data from diverse online platforms and internal systems.
    - Ensuring data consistency and accuracy across different sources.

2. **Inventory Management:**
    - Providing insights into inventory levels and sales trends through batch processing.
    - Predicting demand fluctuations and adjusting inventory accordingly.

3. **Scalability:**

- Handling large volumes of batched transactional data from global online platforms.
- Scaling to accommodate peak shopping periods and seasonal demand variations.

4. **Operational Efficiency:**
    - Optimizing procurement and fulfillment processes based on batched data insights.
    - Minimizing storage costs and reducing excess inventory.

Solution:

Using AWS Services to Build a Batch Data Inventory Management System:

1. **Data Ingestion:**
    - **Amazon S3:** Upload batch data files (CSV, JSON) containing sales, inventory, and supplier information.
    - **AWS Kinesis Data Streams or AWS Direct Connect:** Implement the data ingestion mechanism to stream data from a source to Amazon S3.

2. **Data Processing:**
    - **AWS Glue:** Develop and test the data processing pipeline using Apache Spark.
        - Apply data transformation and cleansing techniques to prepare the data for aggregation and analysis.
        - Implement data partitioning and indexing strategies to optimize query performance.
        - Update the GitHub repository with the code and configuration files for data ingestion and processing.

3. **Data Storage:**
    - **Amazon RDS (Relational Database Service):**
        - Set up an RDS instance (e.g., MySQL, PostgreSQL) to store structured transactional data:

        - Sales transactions.
        - Product details.
        - Inventory levels.
        - Supplier information.

4. **Analytics and Reporting:**
    - **Amazon Athena:**
        - Query data directly in Amazon S3 for ad-hoc analysis and reporting.
    - **Quicksight**
        - Visualize data using Amazon QuickSight for interactive dashboards.

5. **Security and Access Control:**
    - **AWS IAM:** Manage access to RDS and other AWS services:
        - Ensure data privacy and compliance with regulatory requirements.

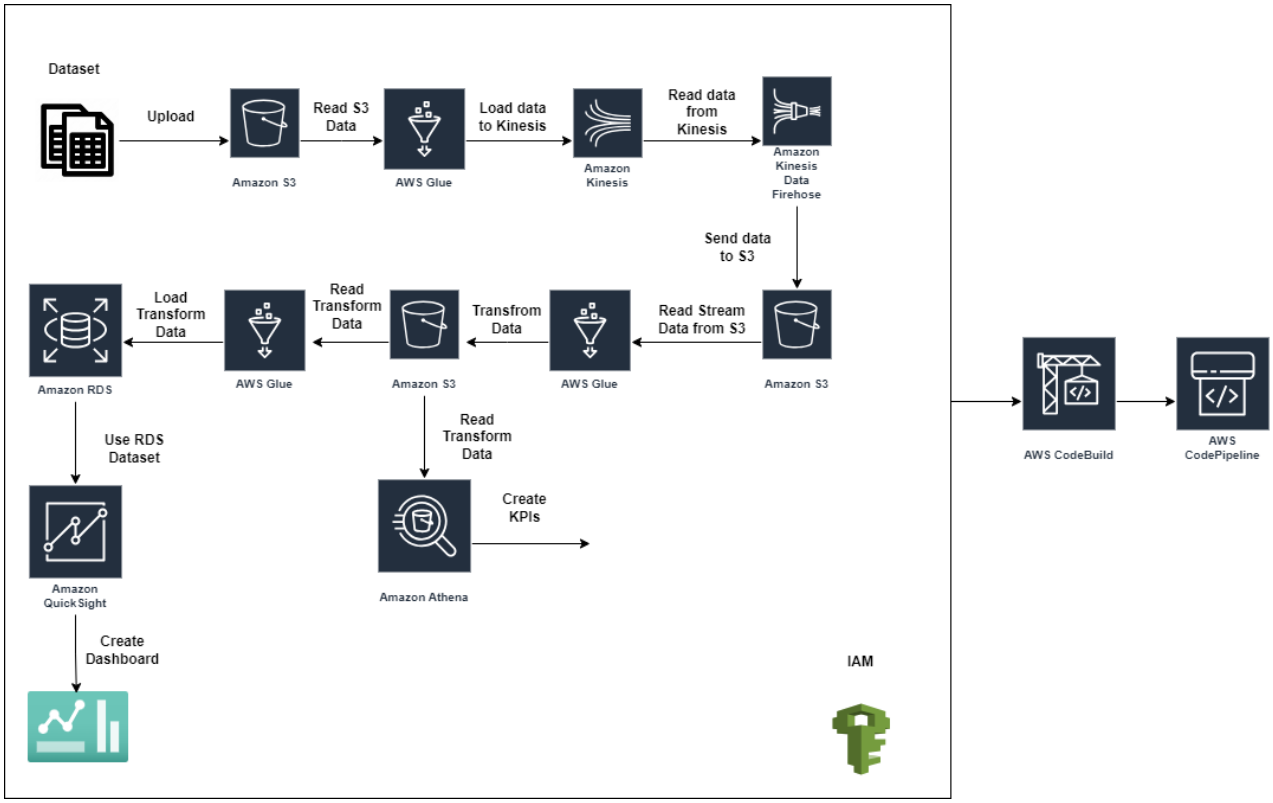
End-to-End Data Engineering Platform:

1. **Integration:**
    - Integrate the components developed to create a complete end-to-end data engineering platform.
2. **Data Management:**
    - Implement data retention policies and configure data lifecycle management in Amazon S3.
3. **Security Best Practices:**
    - Enable encryption for data at rest and in transit.
    - Set up proper access controls using AWS IAM for authentication and authorization.
4. **Cost Optimization:**
    - Utilize cost-effective storage options like Amazon S3 Glacier for long-term data retention.
    - Monitor and optimize resource utilization to reduce costs.

5. **Testing and Validation:**
    - Conduct thorough testing of the platform, including data ingestion, processing, aggregation, and visualization, to ensure data integrity and performance.

6. **CI/CD Automation:**
    - Implement continuous integration and continuous deployment (CI/CD) processes using AWS CodePipeline and AWS CodeBuild to automate deployment and updates.

Architecture Diagram:



Key Performance Indicators (KPIs):

- **Inventory Turnover Ratio:** measures how efficiently inventory is managed by indicating how many times inventory is sold and replaced over a period.

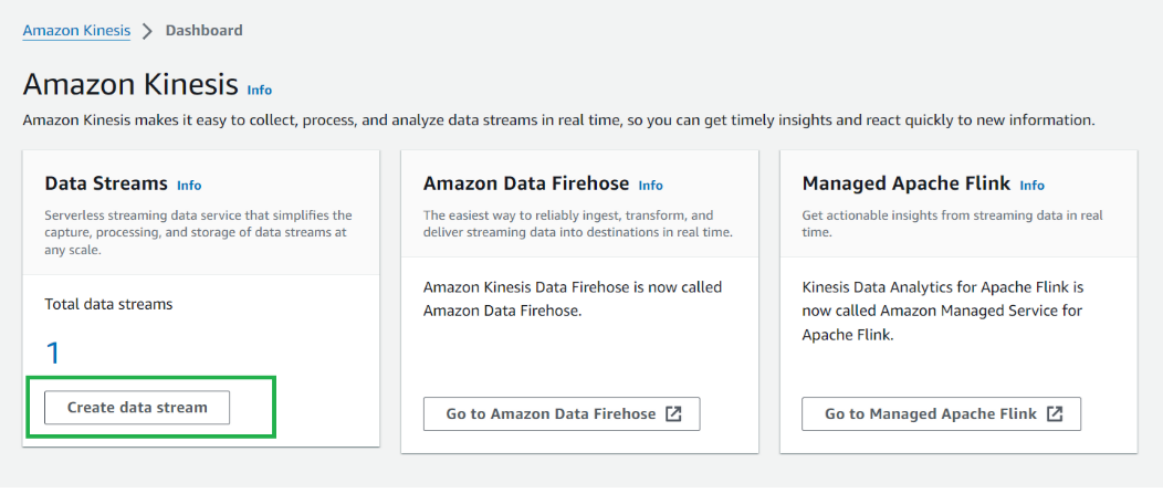
- **Stockout Rate:** measures the percentage of time products are out of stock.
- **COGS to Revenue Ratio:** measures the efficiency of managing inventory costs relative to revenue.
- **Customer Acquisition Cost (CAC):** measures the average cost of acquiring a new customer.
- **Returning Customers**: The total number of unique customers who made purchases on July 11, 2011.
- **Average Order Value (AOV):** Measures the average amount spent per order over the course of the year 2011.
- **Gross Margin:** Measures the profitability of products by comparing the revenue (InvoiceTotal) to the cost of goods sold (Quantity * UnitPrice).
- **COGS to Revenue Ratio:** Efficiency in managing inventory costs relative to revenue.

Benefits:

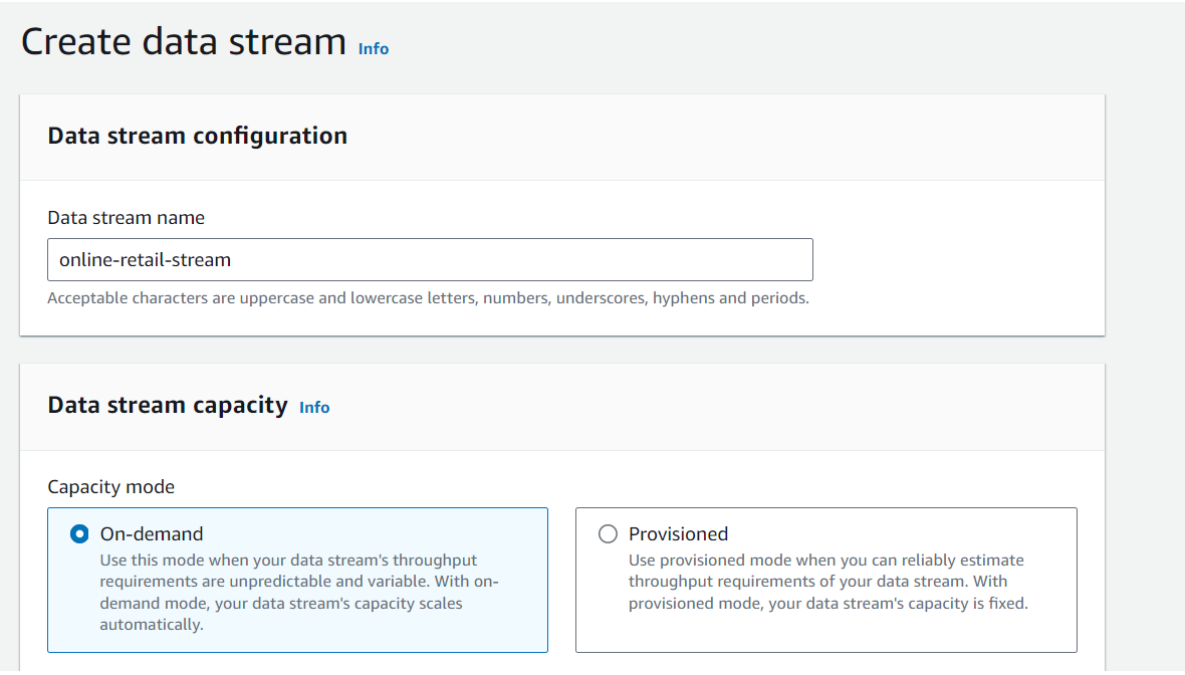
- **Improved Inventory Management:** Enhanced control over inventory levels and reduced stockouts.
- **Efficient Supply Chain Operations:** Optimized procurement and fulfillment processes.
- **Scalability:** AWS services scale with business growth.
- **Cost Efficiency:** Managed services minimize infrastructure costs.
- **Security and Compliance:** Secure handling of data ensures compliance with regulations.

Part 1: Kinesis Setup

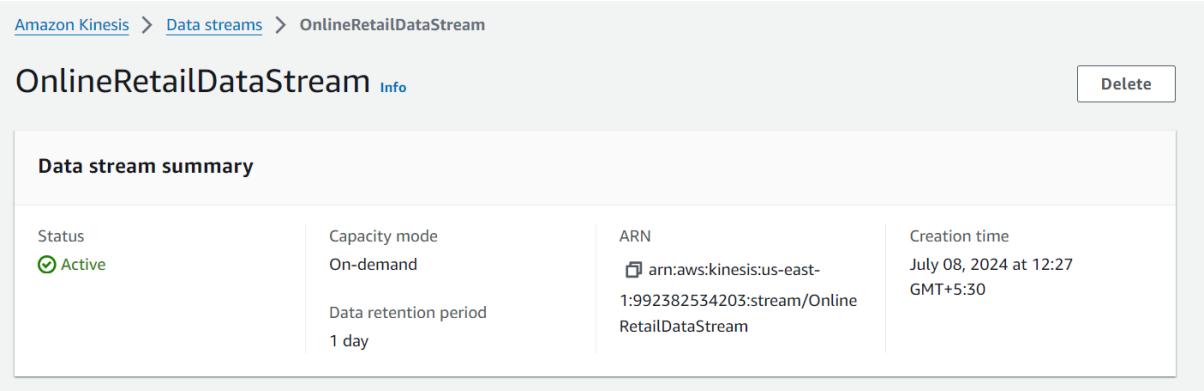
1. Create a Kinesis Data Stream:
   - Open the AWS Management Console.
   - Navigate to Kinesis > Data Streams.
   - Click Create data stream.
   - Enter the stream name (e.g., OnlineRetailDataStream).
   - Specify the number of shards based on your expected data volume.
   - Click Create stream.



2. Enter Details
   a. Give stream name
   b. Choose On-demand or provisioned as per requirement (I have chosen on-demand here)
   c. Click on Create data stream

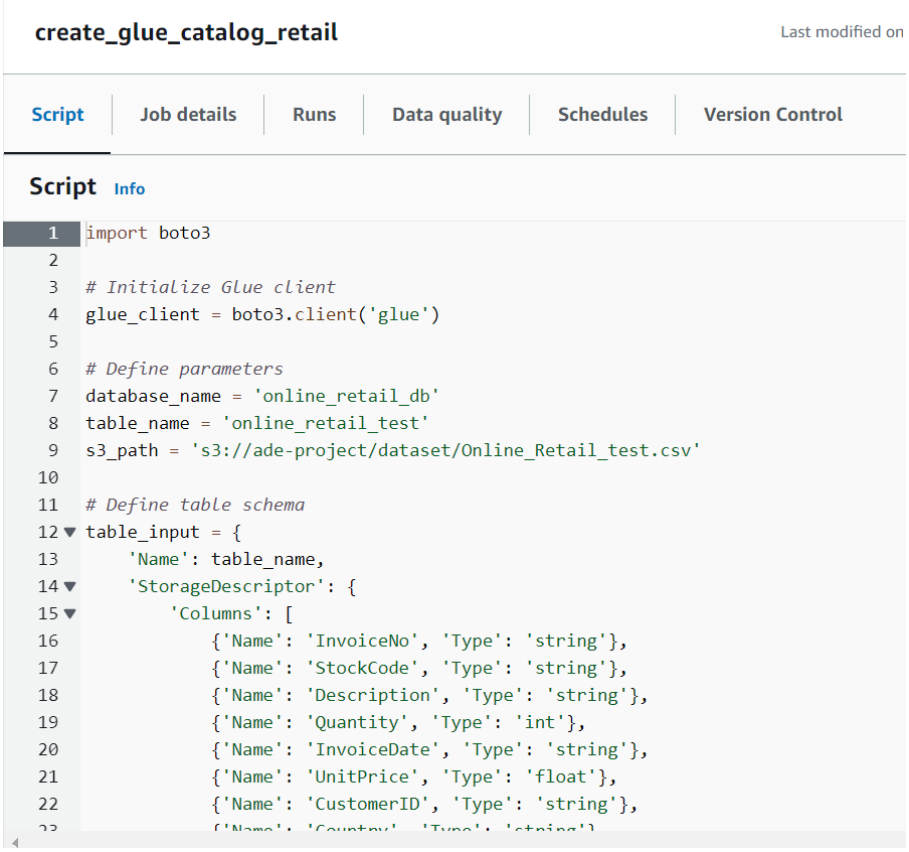


3. Check the status if it is Active



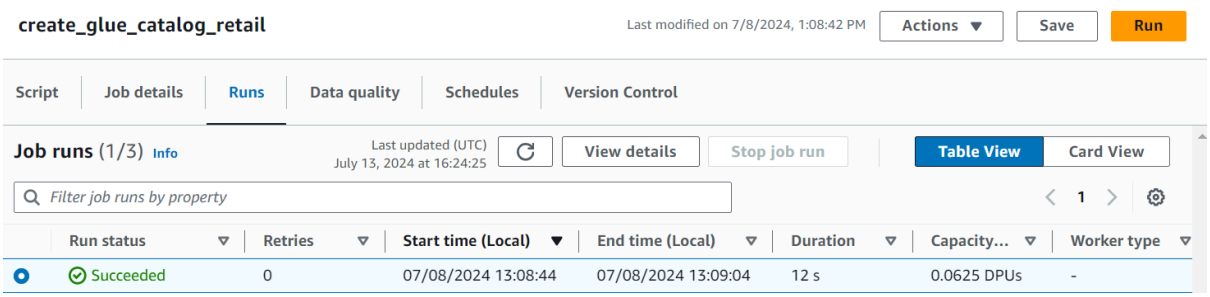
Part 2: Glue Setup to send data to Kinesis

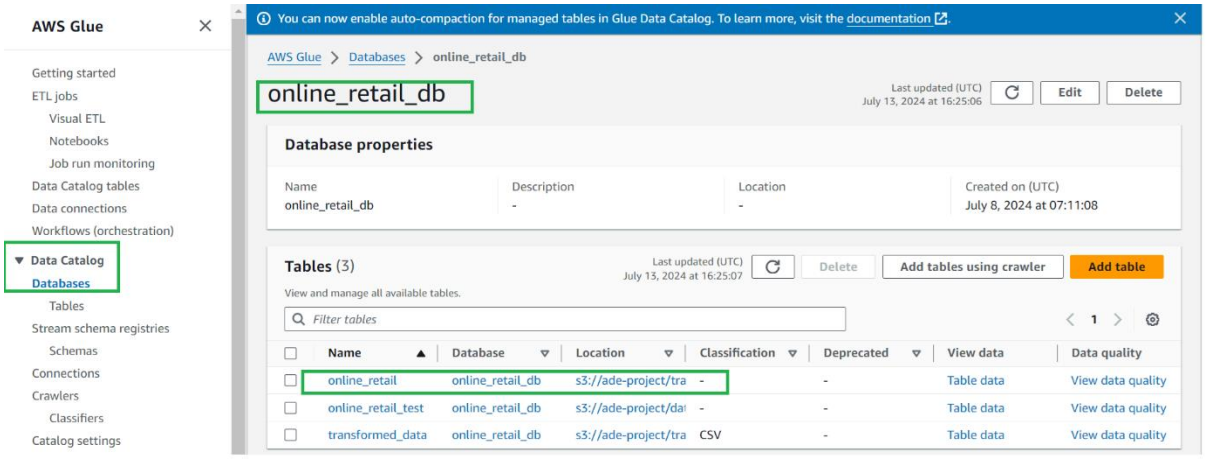
1. Create Python Shell Glue job to create glue catalog for Retail data that we will stream



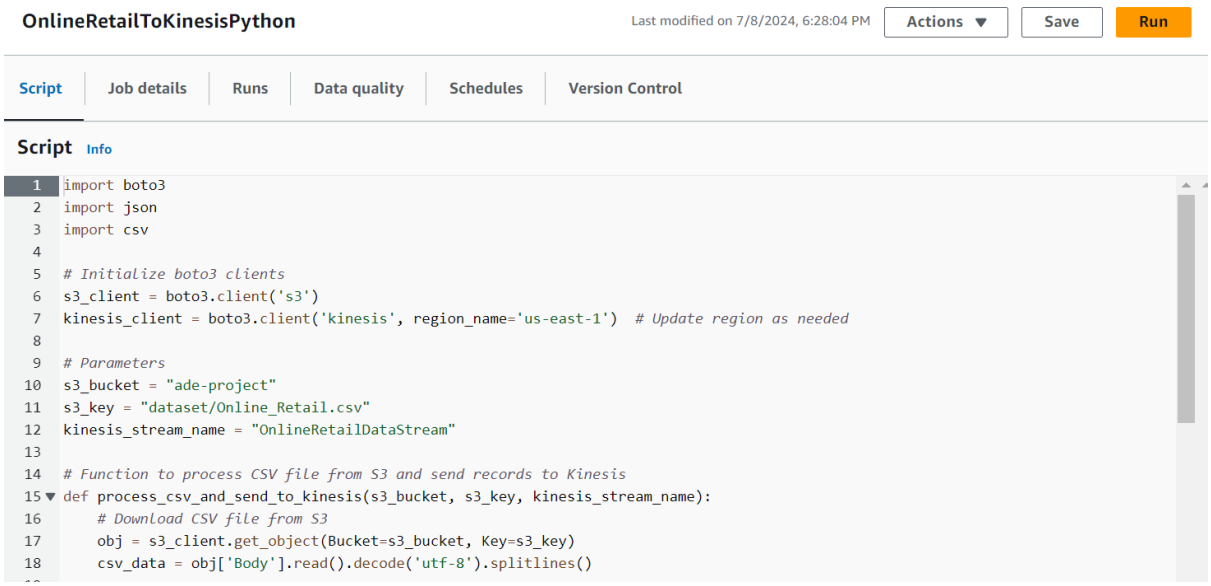

```python
import boto3

# Initialize Glue client
glue_client = boto3.client('glue')

# Define parameters
database_name = 'online_retail_db'
table_name = 'online_retail_test'
s3_path = 's3://ade-project/dataset/Online_Retail_test.csv'

# Define table schema
table_input = {
    'Name': table_name,
    'StorageDescriptor': {
        'Columns': [
            {'Name': 'InvoiceNo', 'Type': 'string'},
            {'Name': 'StockCode', 'Type': 'string'},
            {'Name': 'Description', 'Type': 'string'},
            {'Name': 'Quantity', 'Type': 'int'},
            {'Name': 'InvoiceDate', 'Type': 'string'},
            {'Name': 'UnitPrice', 'Type': 'float'},
            {'Name': 'CustomerID', 'Type': 'string'},
```

2. Also, have the dataset in an s3 location where you want to stream the data from
3. Once the job succeeds, check the glue catalog

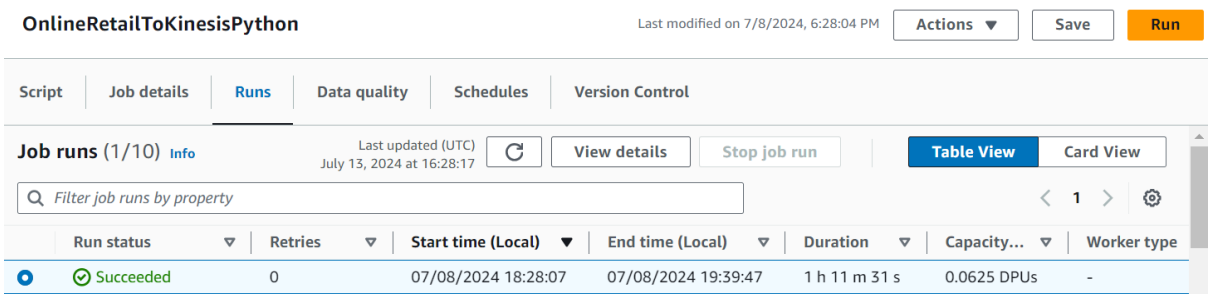




4. Next Create a Python Shell Glue to send the data to kinesis stream
    a. Use boto3 Client to connect to Kinesis Stream
    b. Give s3 location to send the data from



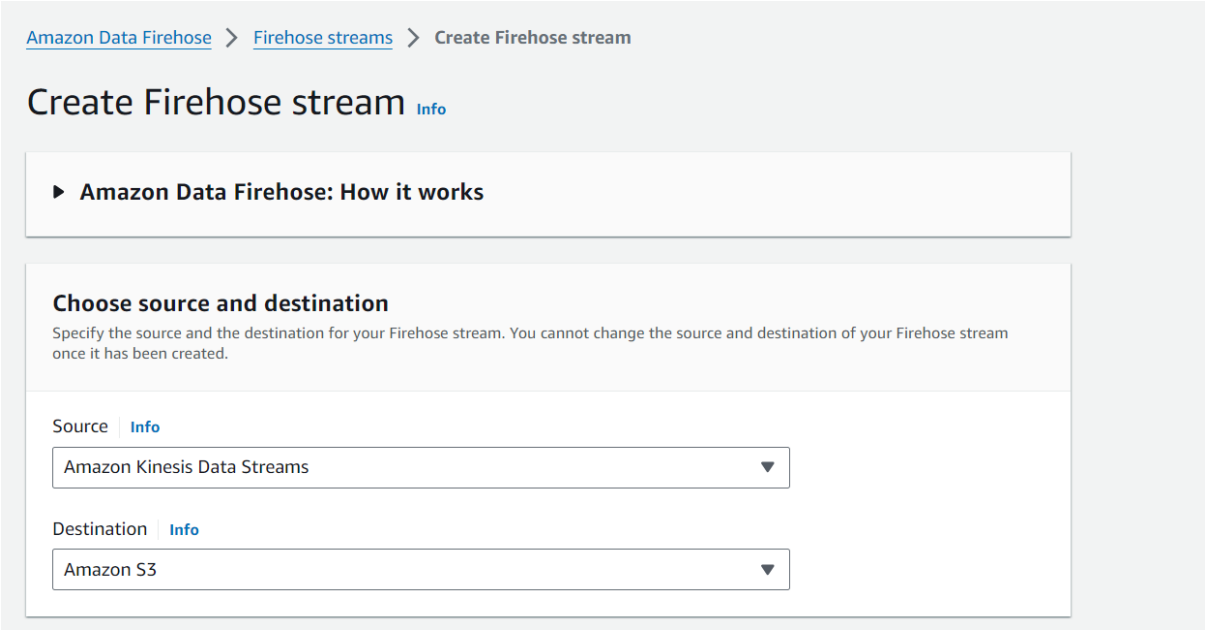
5. Once the job succeeds, proceed to create Data Firehose to get that data



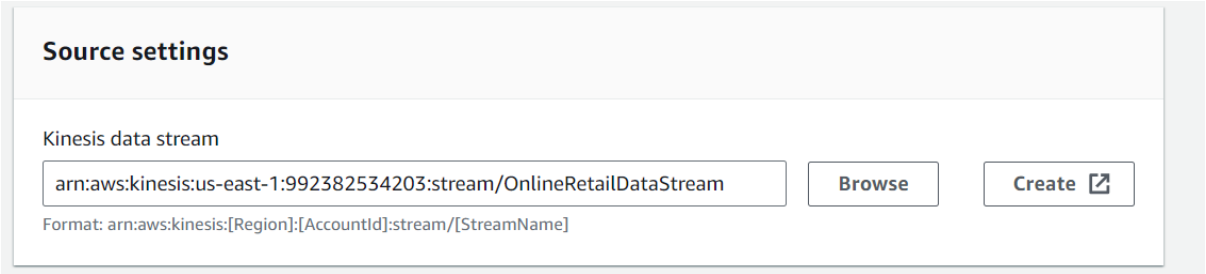
6. Also, you can monitor the data has arrived in monitoring tab of kinesis

Part 3: Setup Data Firehose

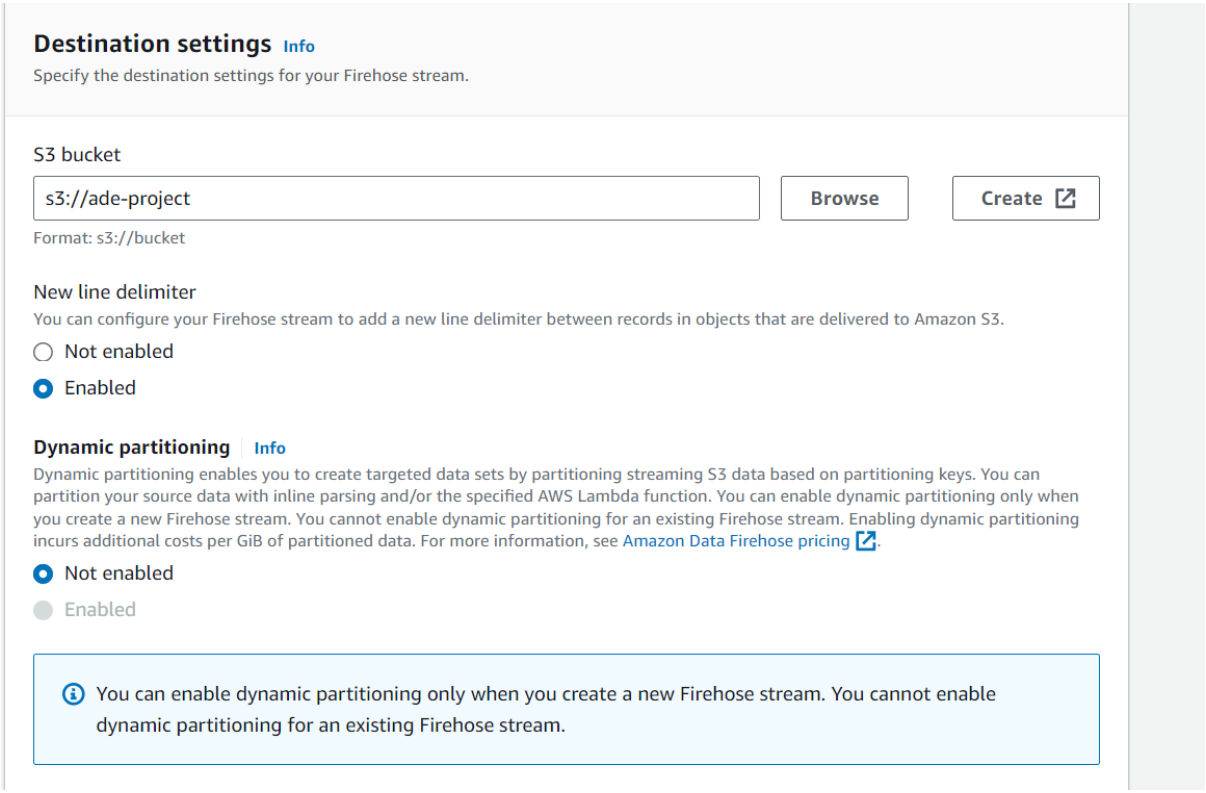
1. Click on Create Firehose stream
2. Choose Source and Destination



3. Give source stream



4. Give the destination settings



S3 bucket prefix - *optional*

By default, Amazon Data Firehose appends the prefix "YYYY/MM/dd/HH" (in UTC) to the data it delivers to Amazon S3. You can override this default by specifying a custom prefix that includes expressions that are evaluated at runtime.

stream_data/

You can repeat the same keys in your S3 bucket prefix. Maximum S3 bucket prefix characters: 1024.

S3 bucket error output prefix - *optional*

You can specify an S3 bucket error output prefix to be used in error conditions. This prefix can include expressions for Amazon Data Firehose to evaluate at runtime.

stream_data/error/

S3 bucket and S3 error output prefix time zone | Info

Choose a time zone that you want to use for date and time in S3 prefixes

UTC

5. Provide proper IAM role to access S3 and Kinesis Data Stream

Service access Info

Edit

Amazon Data Firehose uses this IAM role for all the permissions that the Firehose stream needs. To specify different roles for the different permissions, use the API or the CLI.

IAM role

KinesisFirehoseServiceRole-KDS-S3-DfiuG-us-east-1-1720425224099

6. Once the setup is done and Firehose is active, whenever data comes in Kinesis Data Stream Firehose will put the data to s3 destination location in part files

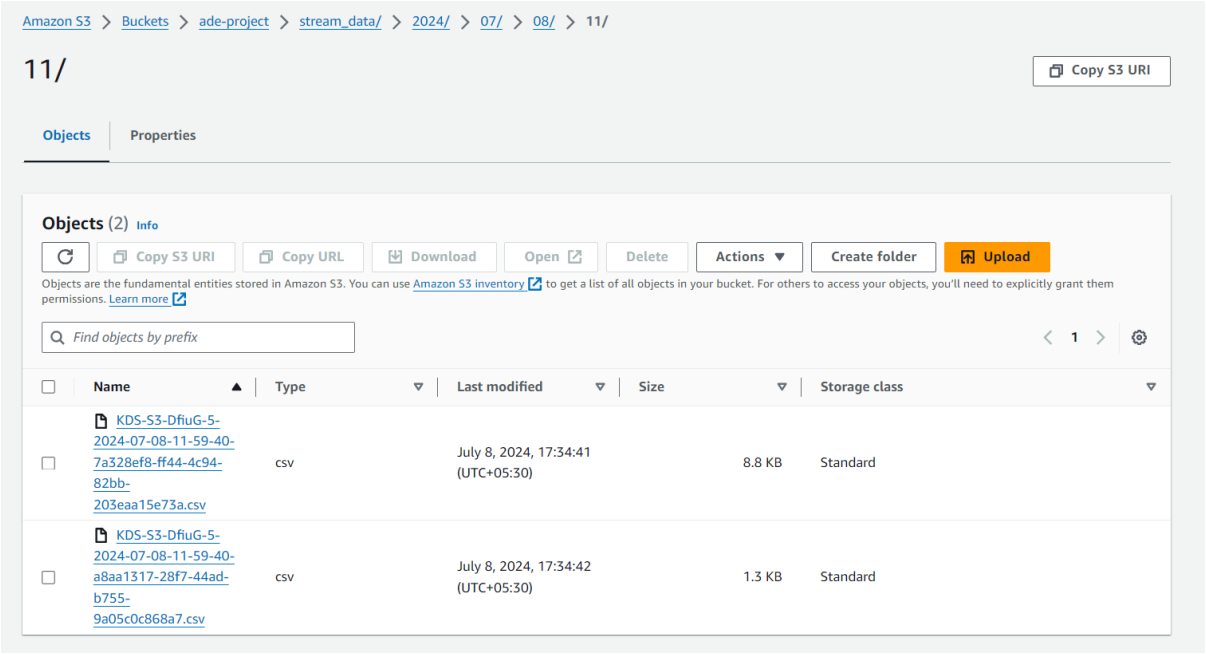
Amazon Data Firehose > Firehose streams > KDS-S3-DfiuG

KDS-S3-DfiuG Info

Delete Firehose stream

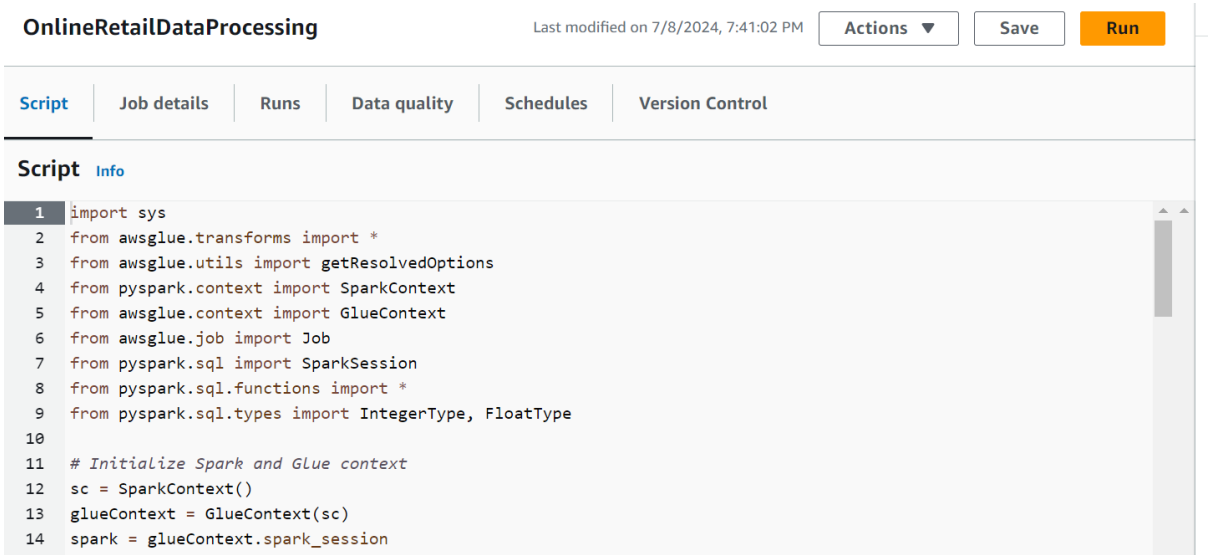
Firehose stream details

Status	Destination	Data transformation	Creation time
Active	Amazon S3	Not enabled	July 08, 2024 at 13:26 GMT+5:30
Source	**ARN**	**Dynamic partitioning**	**Error logs status**
Amazon Kinesis Data Streams	arn:aws:firehose:us-east-1:992382534203:deliverystream/KDS-S3-DfiuG	Not enabled	0 Destination error logs



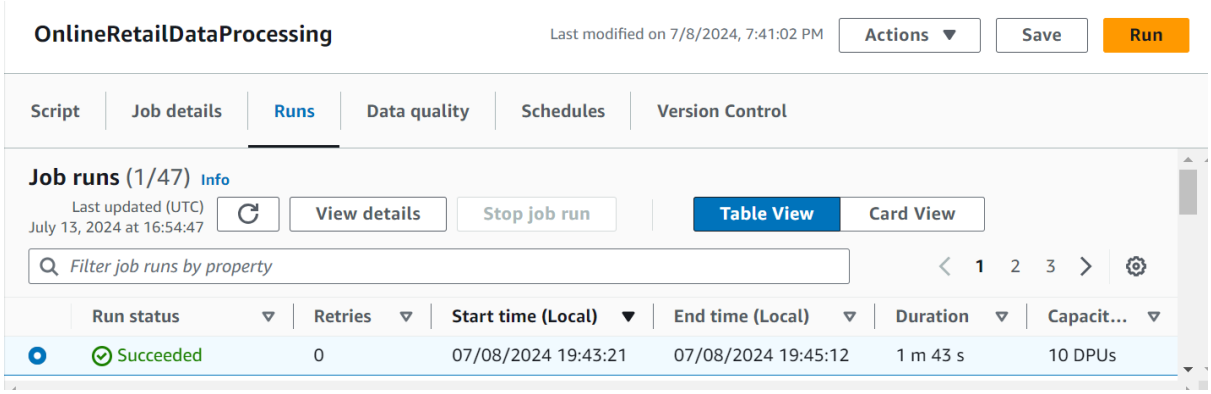
Part 4: Glue Job to for transform Stream data into Cleansed data

1. Create Spark Glue Job

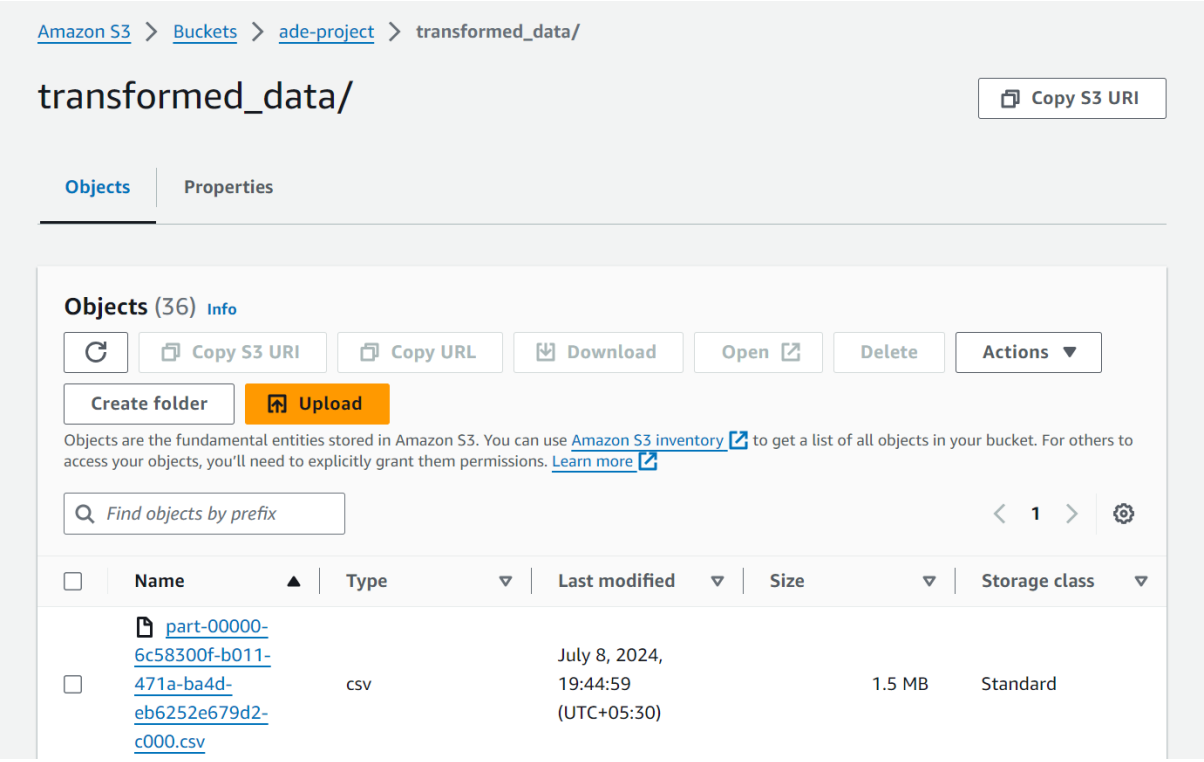


2. Store the data in an S3 location

3. Check if Job succeeded

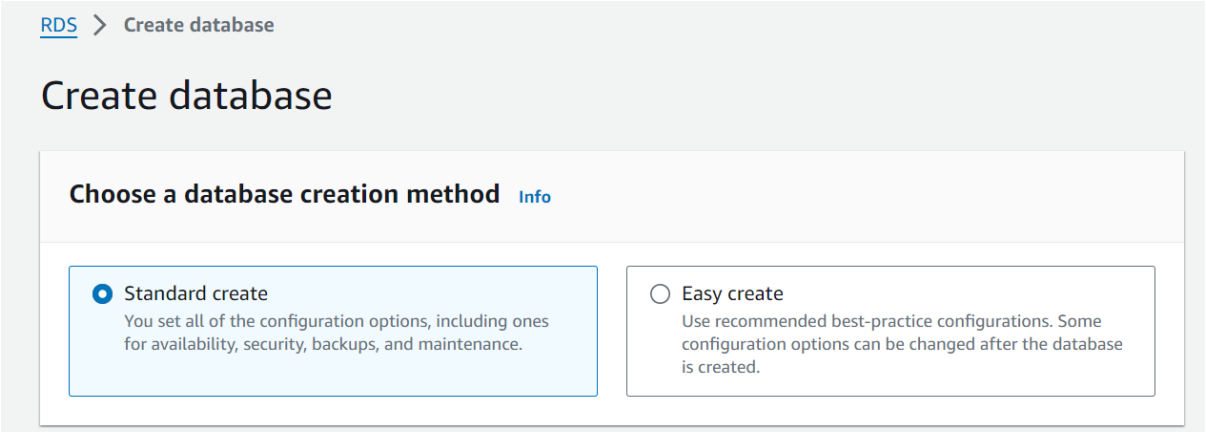


4. Check s3 for transformed data

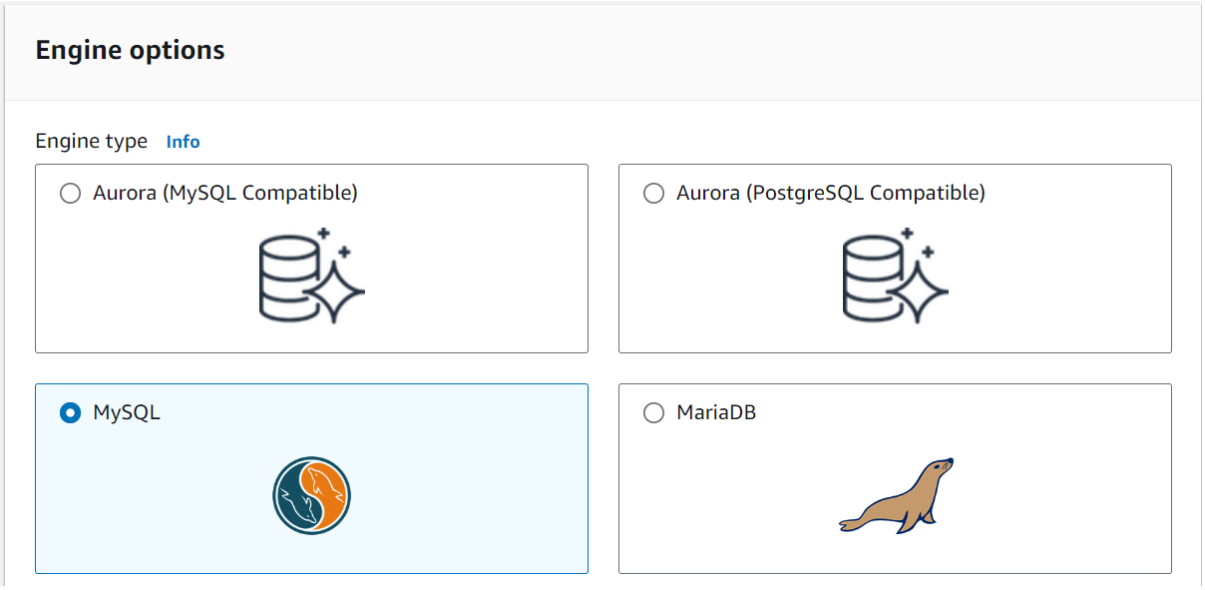


Part 5: Create RDS instance

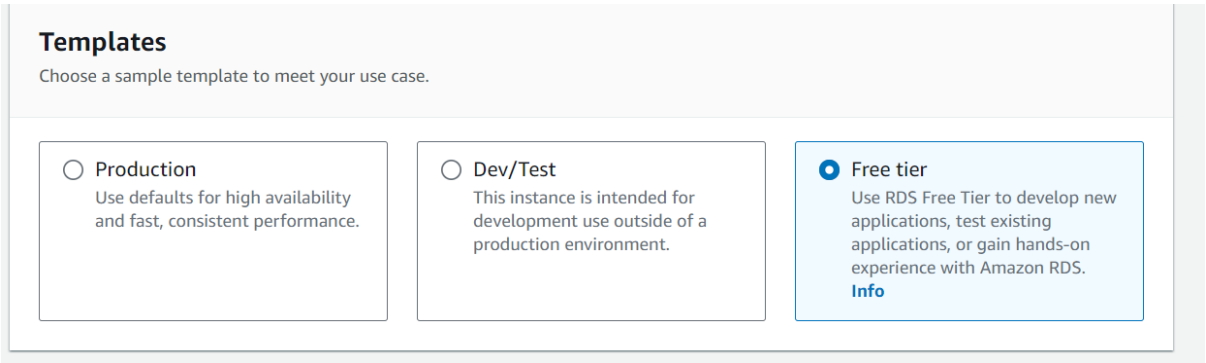
1. Go to RDS console and Create DB instance



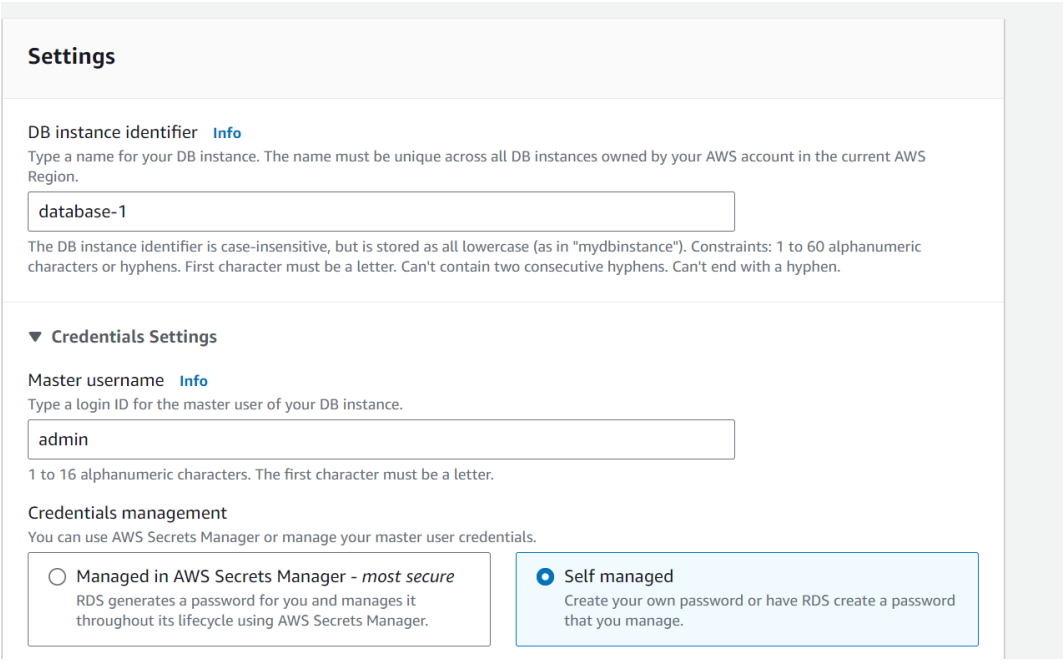
2. Choose MySQL

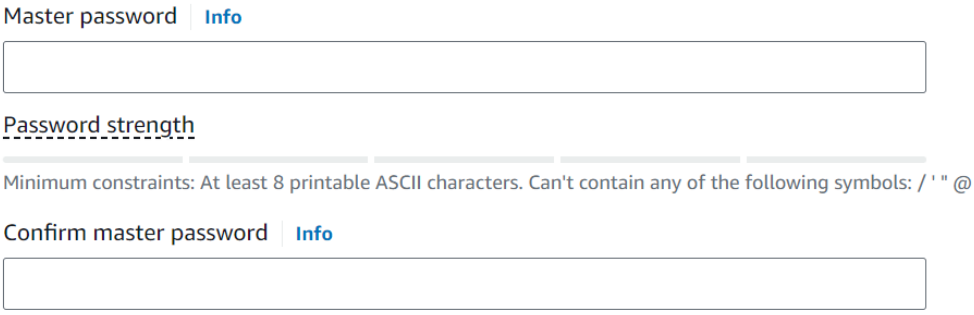


3. Choose Free Tier

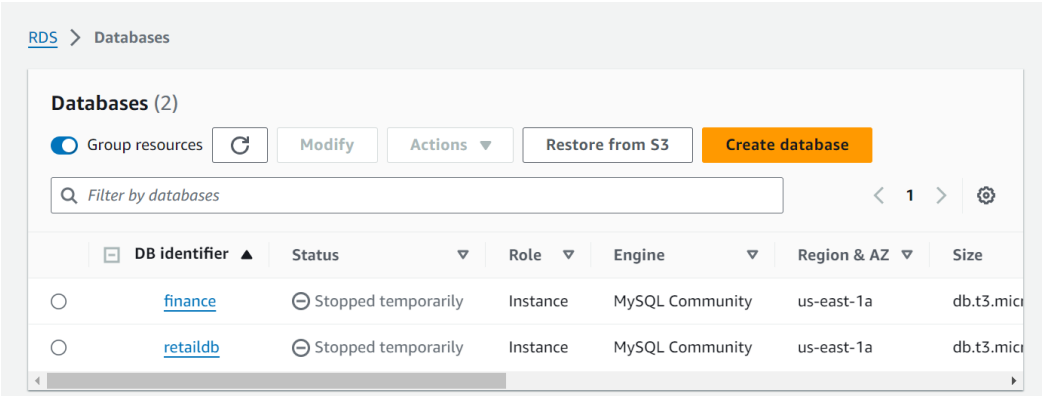


4. Give DB instance and DB username and password



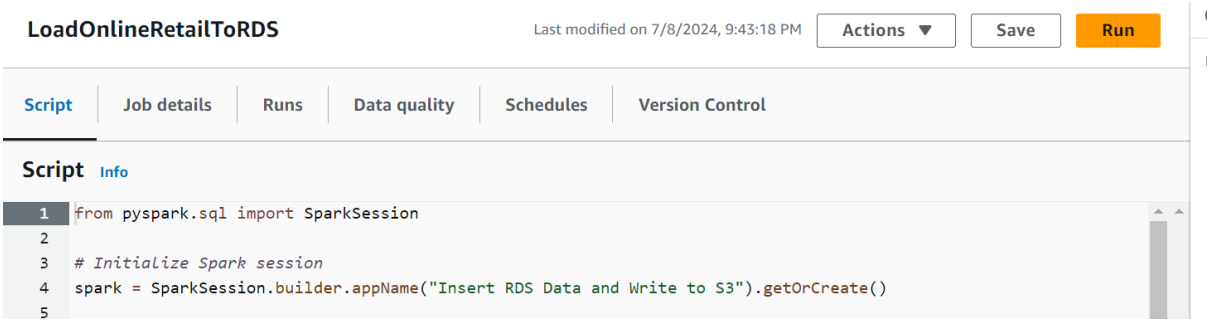


5. Keep everything else as it is.
6. Once the you create it should have status as available (Here I have stopped it for cost purpose)



Part 6: Load Transformed Data To RDS

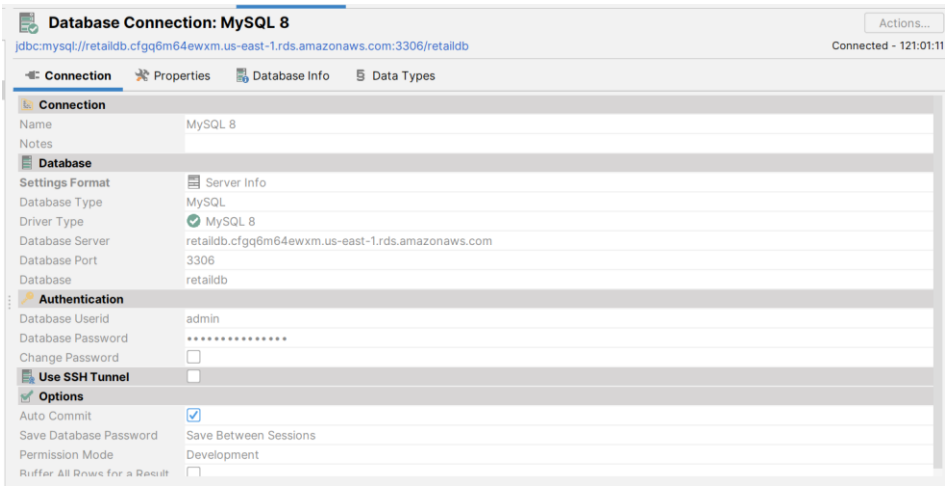
1. Create Spark Glue Job to load s3 data to RDS



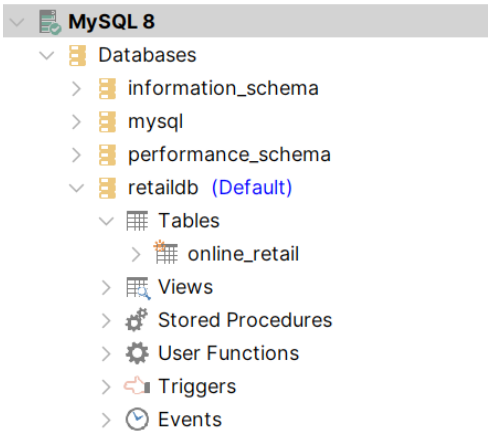
2. Provide connection details

```
rds_options = {
    "url": "jdbc:mysql://retaildb.cfgq6m64ewxm.us-east-1.rds.amazonaws.com:3306/retaildb",
    "user": "admin",
    "password": "",
    "driver": "com.mysql.cj.jdbc.Driver"
}
```

3. Once the data is loaded check the data from local as well
    a. Create connection using DBVisualizer

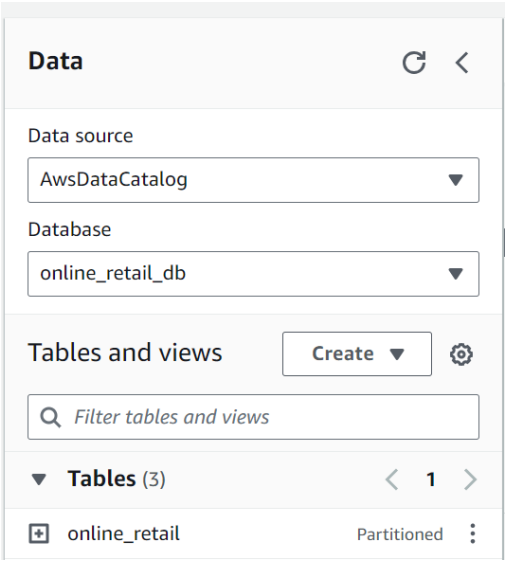


    b. Check if table is created

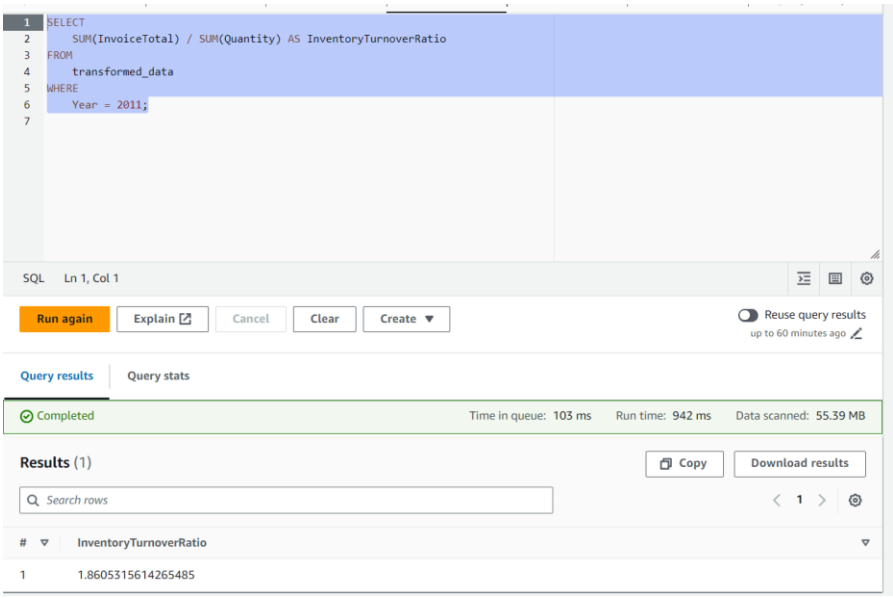


Part 7: Setup Athena and KPIs

1. Go to Athena Console
2. Use the correct database and Table



3. Run aggregation KPI queries:
    a. **Inventory Turnover Ratio** measures how efficiently inventory is managed by indicating how many times inventory is sold and replaced over a period.



b. **Stockout Rate** measures the percentage of time products are out of stock.

```sql
SELECT
    COUNT(DISTINCT InvoiceNo) AS StockoutCount
FROM
    transformed_data
WHERE
    Quantity = 0
    AND Year = 2011;
```

SQL Ln 8, Col 1

Run again | Explain | Cancel | Clear | Create

Reuse query results
up to 60 minutes ago

Query results | Query stats

Completed — Time in queue: 97 ms — Run time: 550 ms — Data scanned: 55.39 MB

Results (1) — Copy — Download results

Search rows

#	StockoutCount
1	0

c. **COGS to Revenue Ratio** measures the efficiency of managing inventory costs relative to revenue.

```sql
SELECT
    SUM(Quantity * UnitPrice) / SUM(InvoiceTotal) AS COGSToRevenueRatio
FROM
    transformed_data
WHERE
    Year = 2011;
```

SQL Ln 6, Col 16

Run again | Explain | Cancel | Clear | Create

Reuse query results
up to 60 minutes ago

Query results | Query stats

Completed — Time in queue: 67 ms — Run time: 1.004 sec — Data scanned: 55.39 MB

Results (1) — Copy — Download results

Search rows

#	COGSToRevenueRatio
1	1.000000000674781

d. **Customer Acquisition Cost (CAC)** measures the average cost of acquiring a new customer.

```sql
SELECT
    SUM(InvoiceTotal) / COUNT(DISTINCT CustomerID) AS CAC
FROM
    transformed_data
WHERE
    Year = 2011;
```

SQL Ln 7, Col 1

Run again | Explain | Cancel | Clear | Create

Reuse query results
up to 60 minutes ago

Query results | Query stats

Completed | Time in queue: 101 ms | Run time: 827 ms | Data scanned: 55.39 MB

Results (1)

Copy | Download results

Search rows

#	CAC
1	2106.045868010359

e. **Returning Customers**: The total number of unique customers who made purchases on July 11, 2011.

```sql
SELECT
    COUNT(DISTINCT CustomerID) AS ReturningCustomers,
    COUNT(DISTINCT CASE WHEN Quantity > 0 THEN CustomerID ELSE NULL END) AS TotalCustomers
FROM
    transformed_data
WHERE
    Year = 2011
    AND Month = 7
    AND DayOfMonth = 11;
```

SQL Ln 10, Col 1

Run again | Explain | Cancel | Clear | Create

Reuse query results
up to 60 minutes ago

Query results | Query stats

Completed | Time in queue: 116 ms | Run time: 787 ms | Data scanned: 55.39 MB

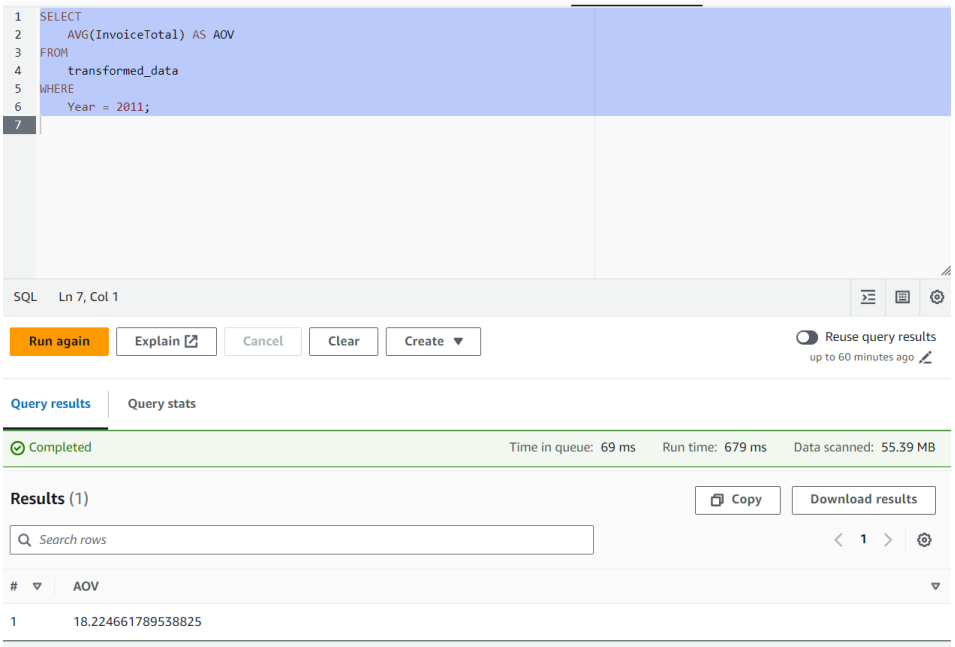
Results (1)

Copy | Download results

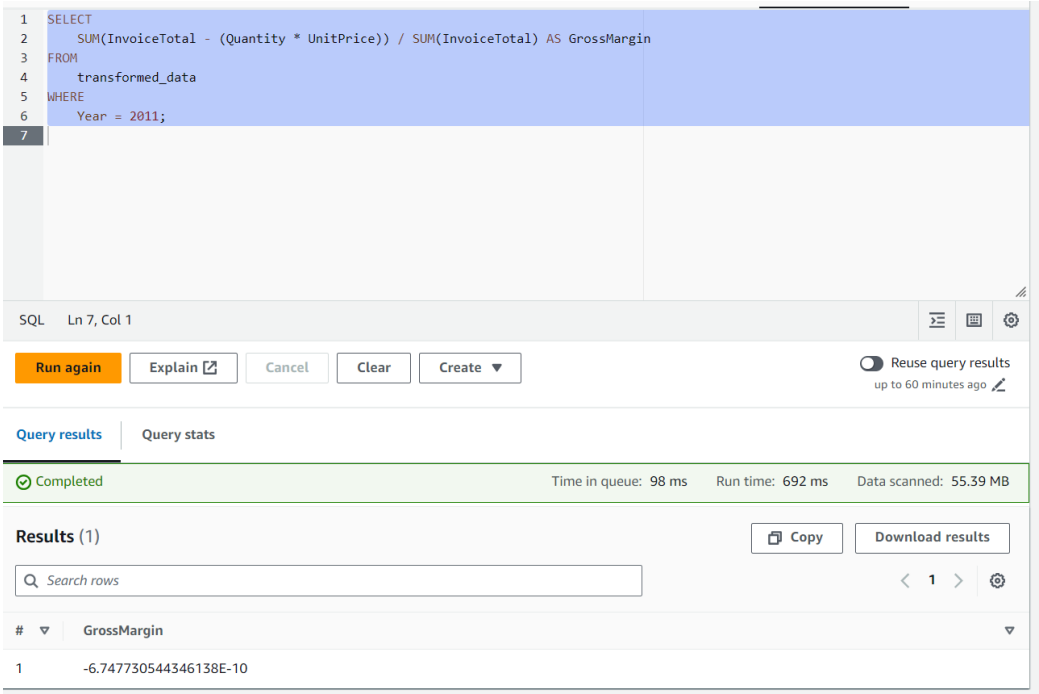
Search rows

#	ReturningCustomers	TotalCustomers
1	49	43

f. **Average Order Value (AOV)**: Measures the average amount spent per order over the course of the year 2011.

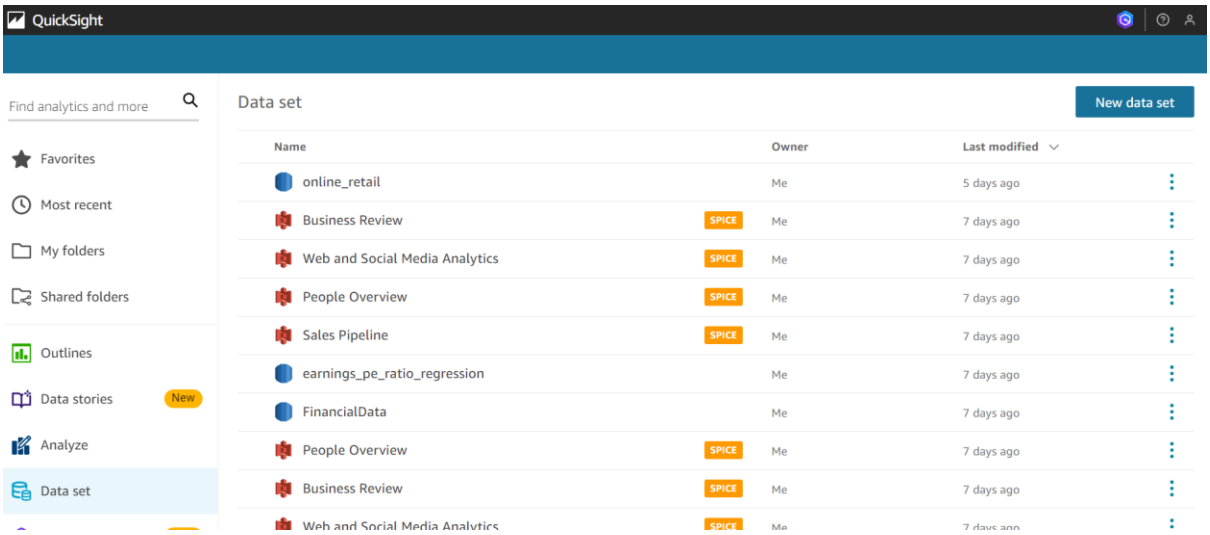


g. Gross Margin: Measures the profitability of products by comparing the revenue (InvoiceTotal) to the cost of goods sold (Quantity * UnitPrice).

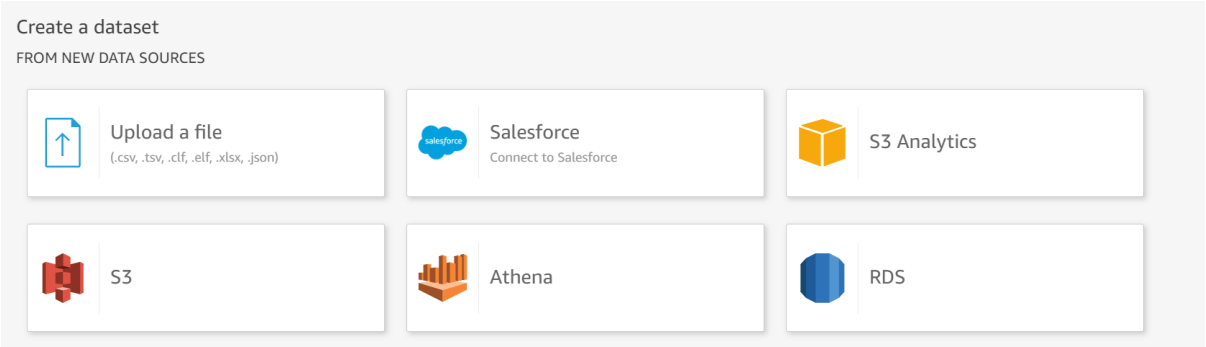


Part 8: Quicksight and Visualizations

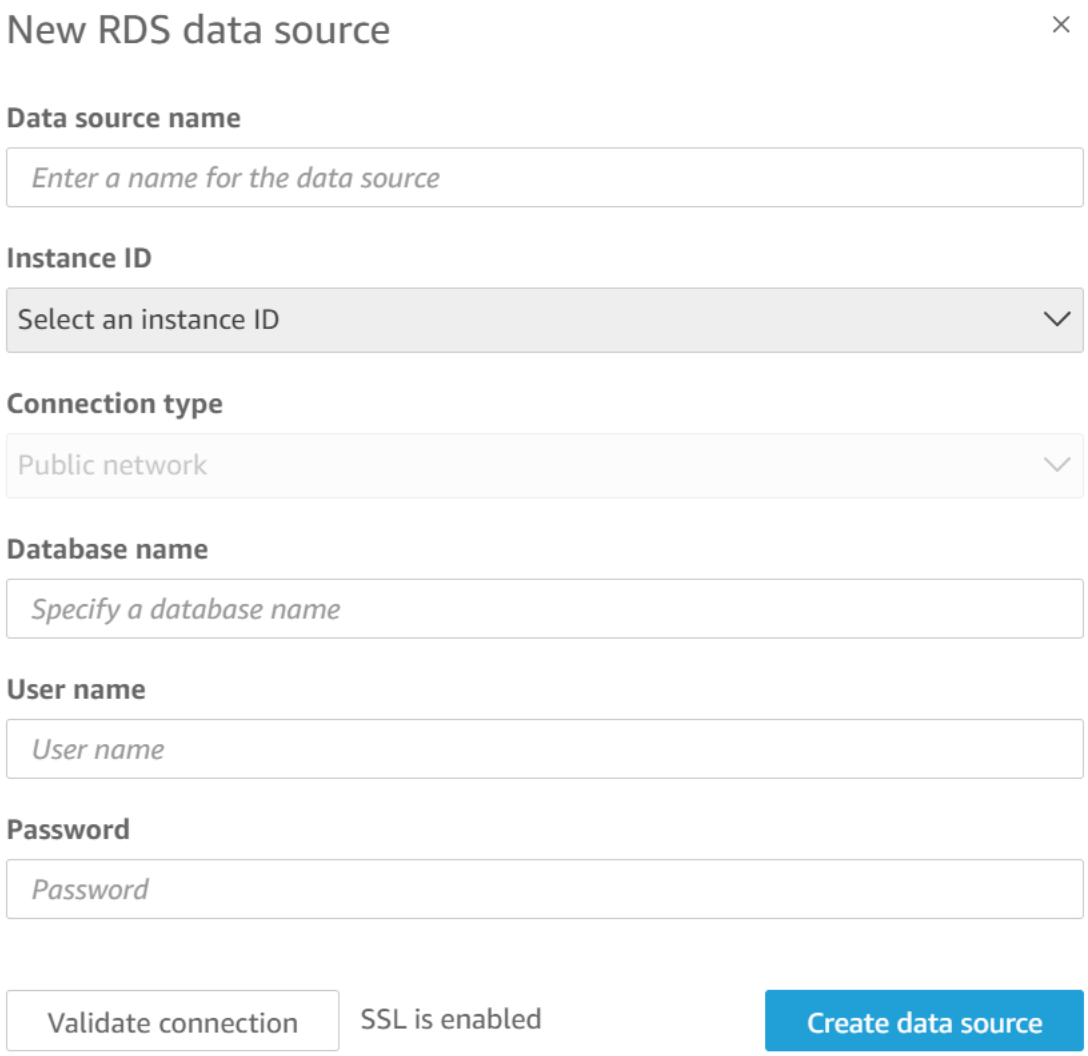
1. Go to Quicksight and login
2. Choose the dataset to have visualization on



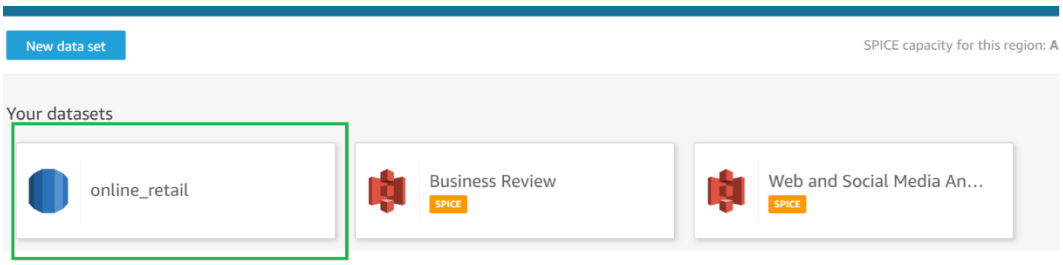
3. Click on new dataset and choose RDS



4. Enter RDS details and test the connection

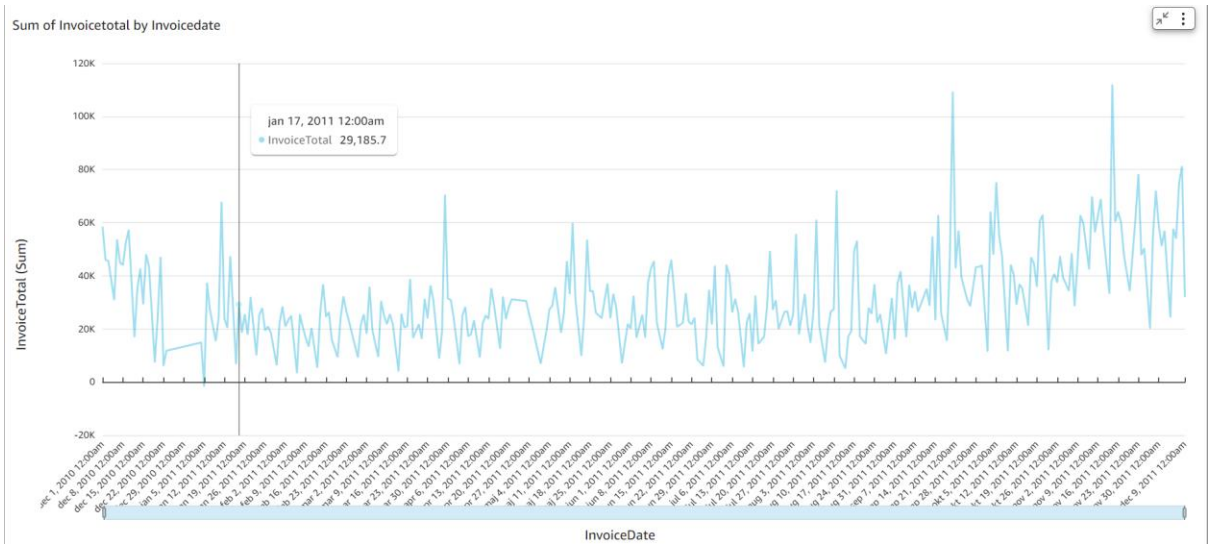


5. Once created, it will be listed in the datasets

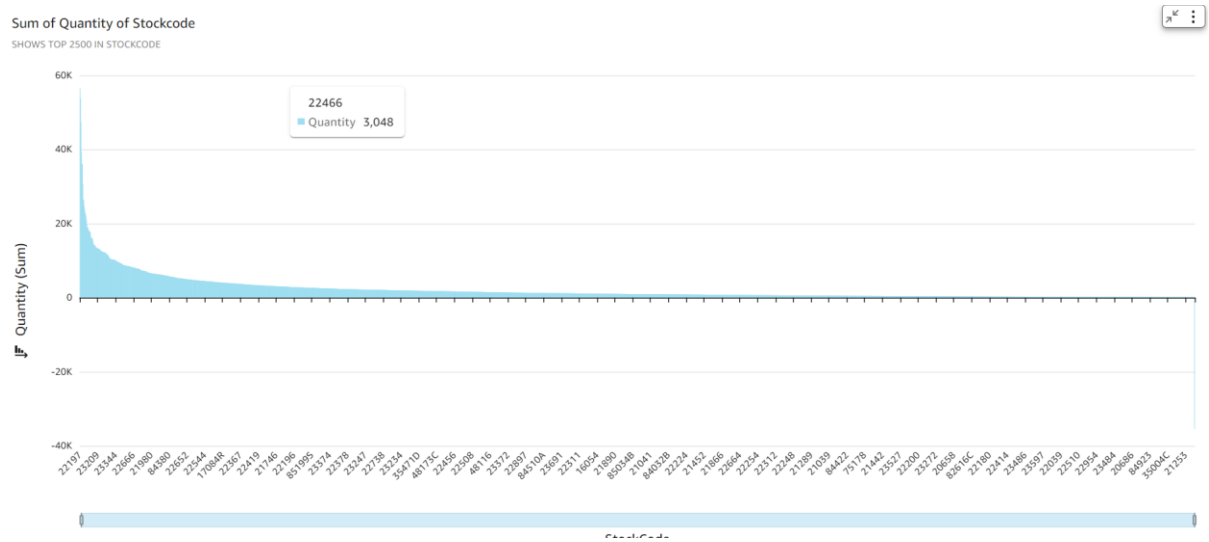


6. Create Dashboard

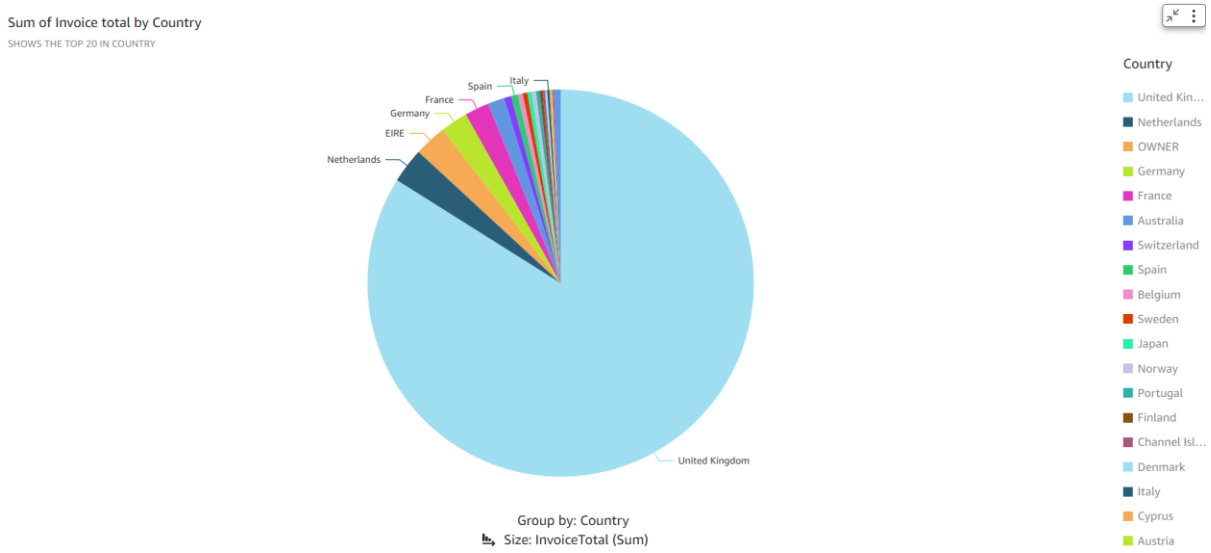
   a. Sum of Invoice total by invoice date



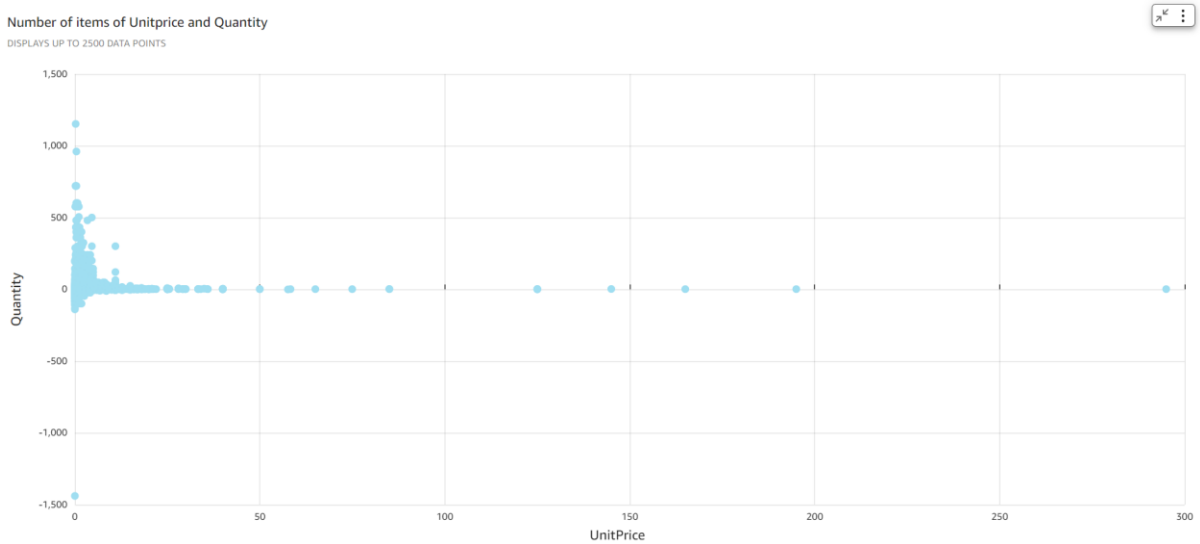
   b. Sum of Quantity of Stockcode



c. Sum of invoice total by country



d. Number of items of Unitprice and Quantity

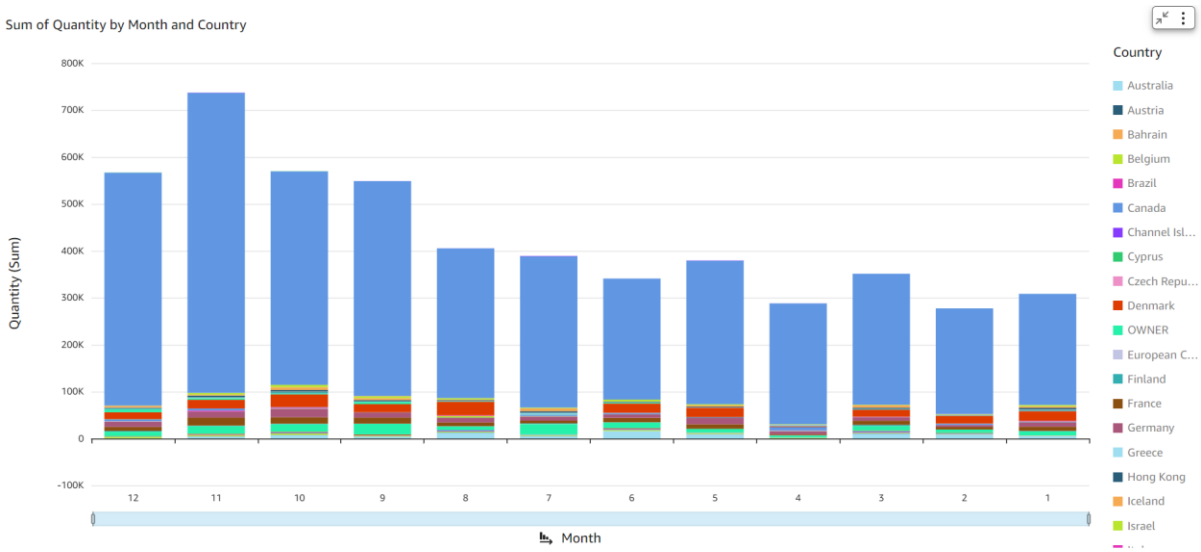


e. Sum of Invoicetotal, Sum of Quantity, and Sum of Unitprice by Month

Sum of Invoicetotal, Sum of Quantity, and Sum of Unitprice by Month

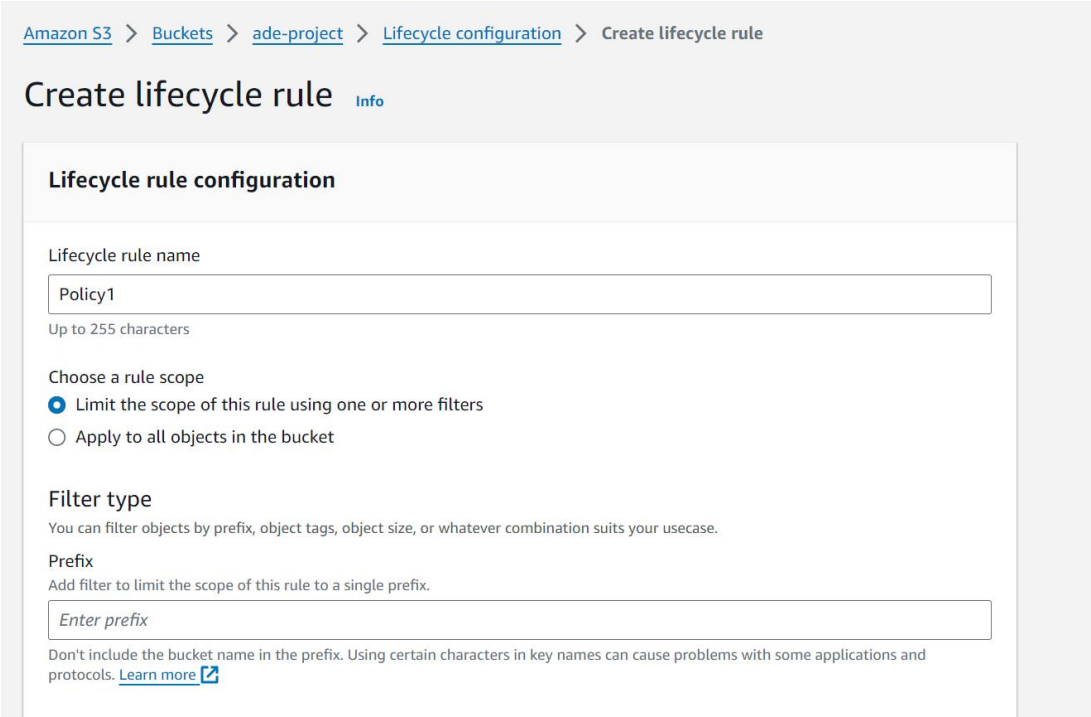
Rows	InvoiceTotal	Quantity	UnitPrice
1	558,448.56	308,281	172,003.69
2	497,026.41	277,374	126,841.95
3	682,013.98	351,165	170,778.3
4	492,367.84	288,237	128,689.46
5	722,094.1	379,652	190,058.09
6	689,977.23	340,945	200,032.62
7	680,156.99	389,051	171,424.58
8	681,386.46	405,450	149,831.85
9	1,017,596.68	548,669	198,308.68
10	1,069,368.23	569,749	261,626.83
11	1,456,145.8	737,182	323,943.25
12	1,179,424.67	566,747	392,533.67

f. Sum of Quantity by Month and Country

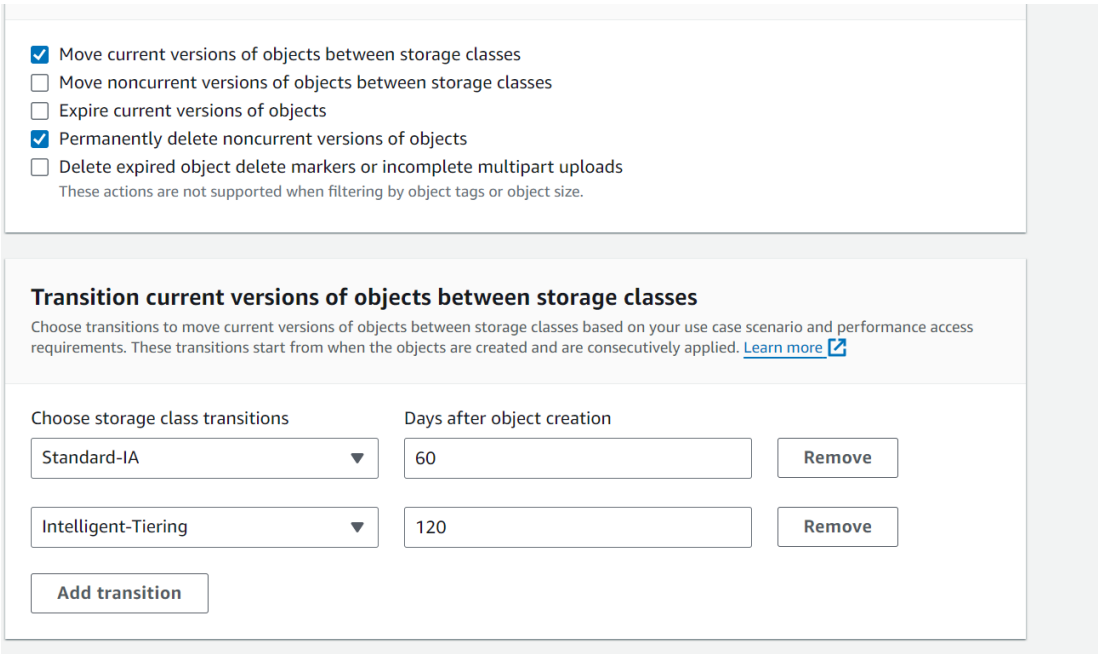


Part 9: S3 lifecycle

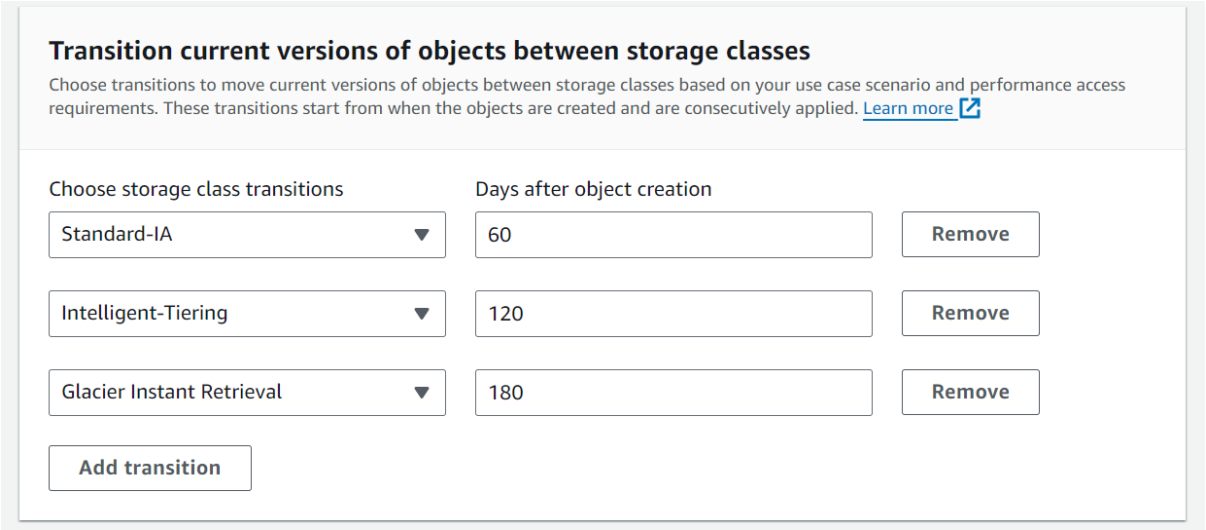
1. Go to Lifecycle Rule in Management of S3 bucket



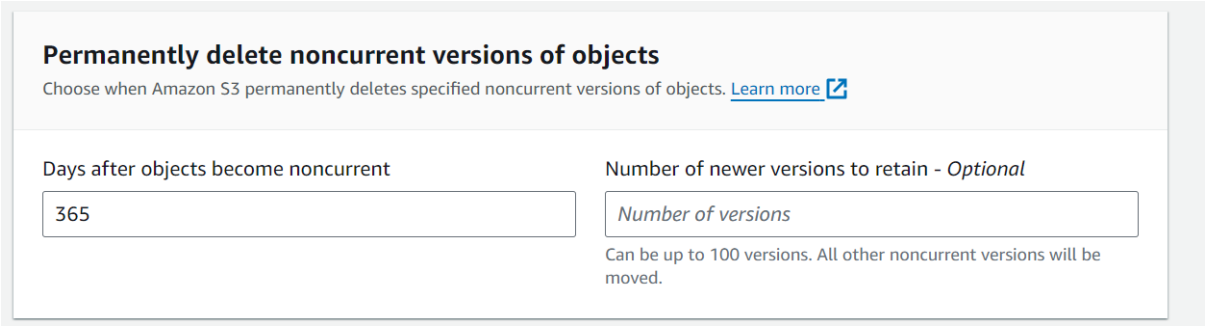
2. Add Rules and Transitions



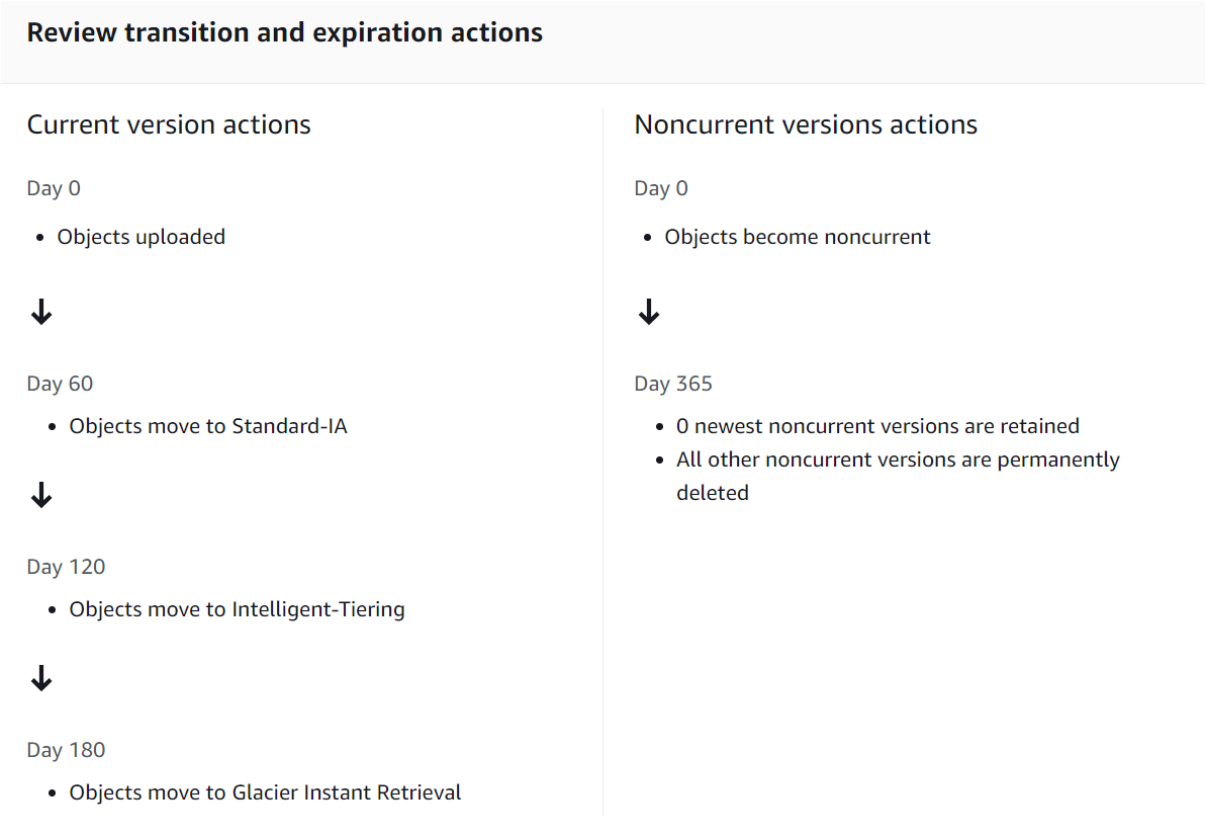
3. Add Glacier also



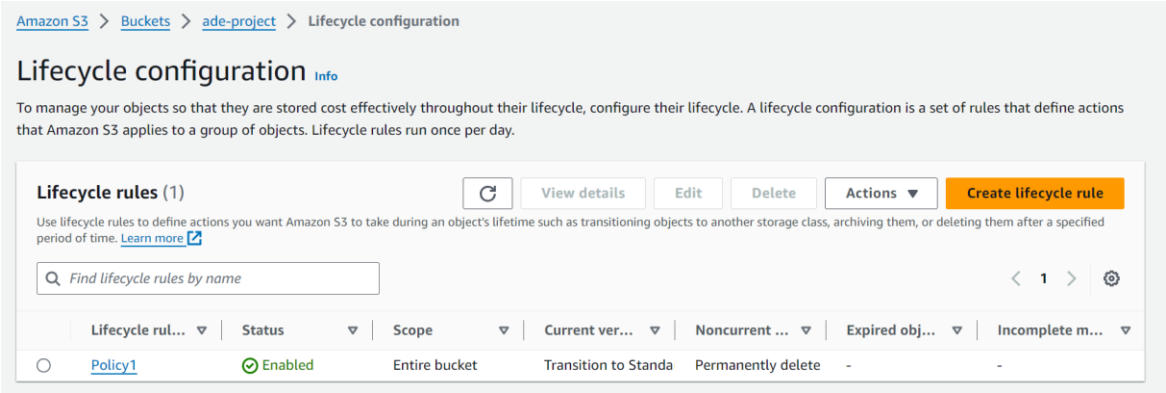
4. Add permanent Delete Also



5. Review transition and expiration actions

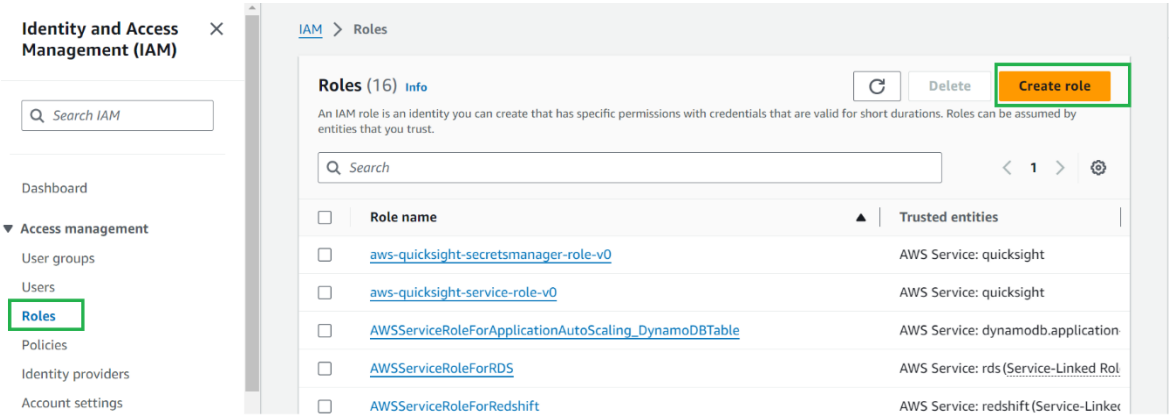


6. Once create it will be added as below

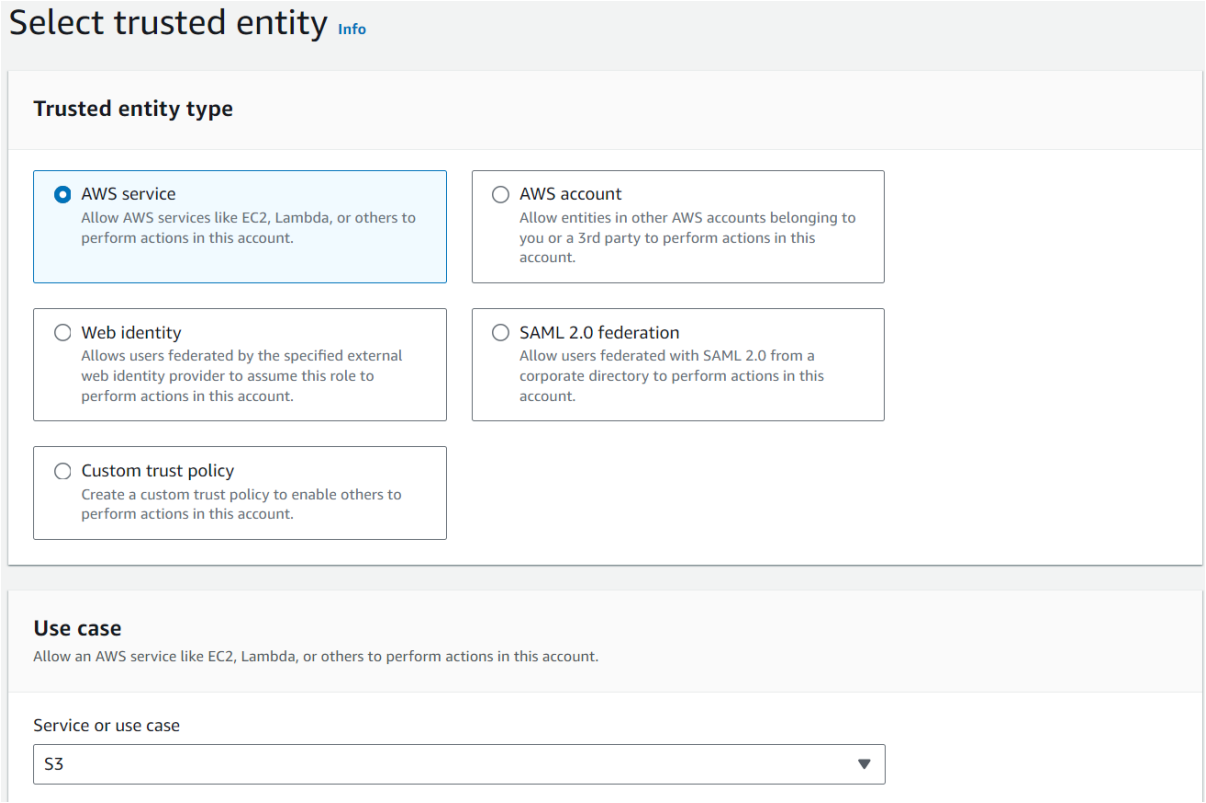


Part 10: IAM Role

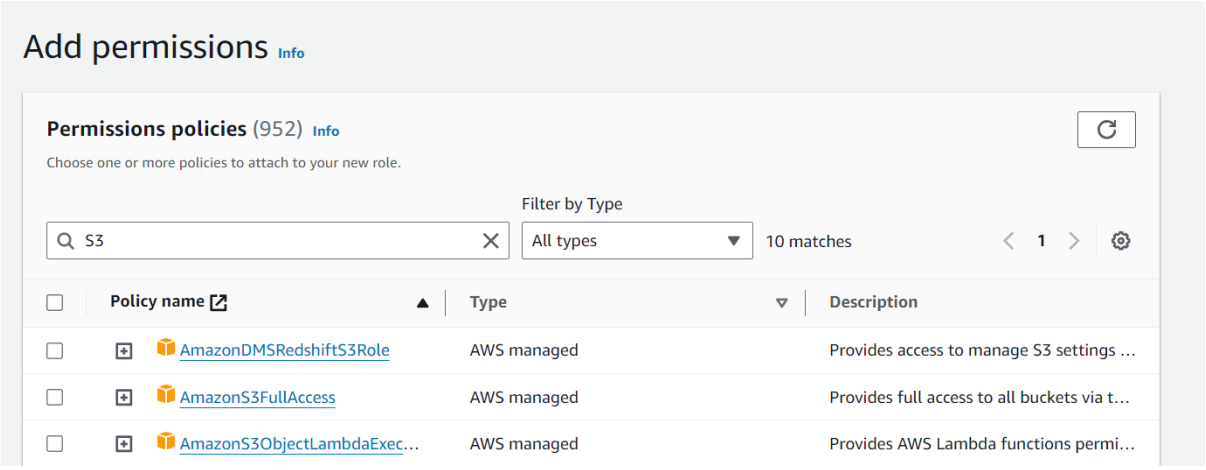
1. Go to IAM Role console
2. Click on Roles and Create Role



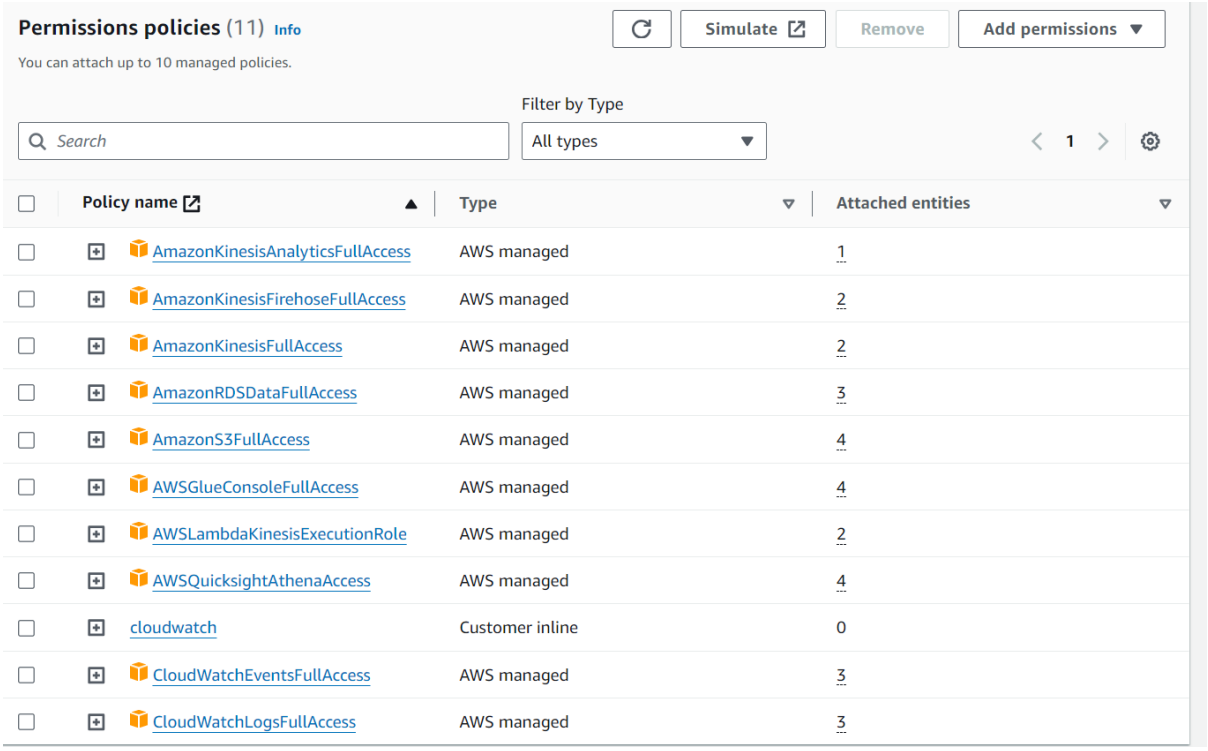
3. Choose AWS service and service to add permissions



4. Add necessary permissions

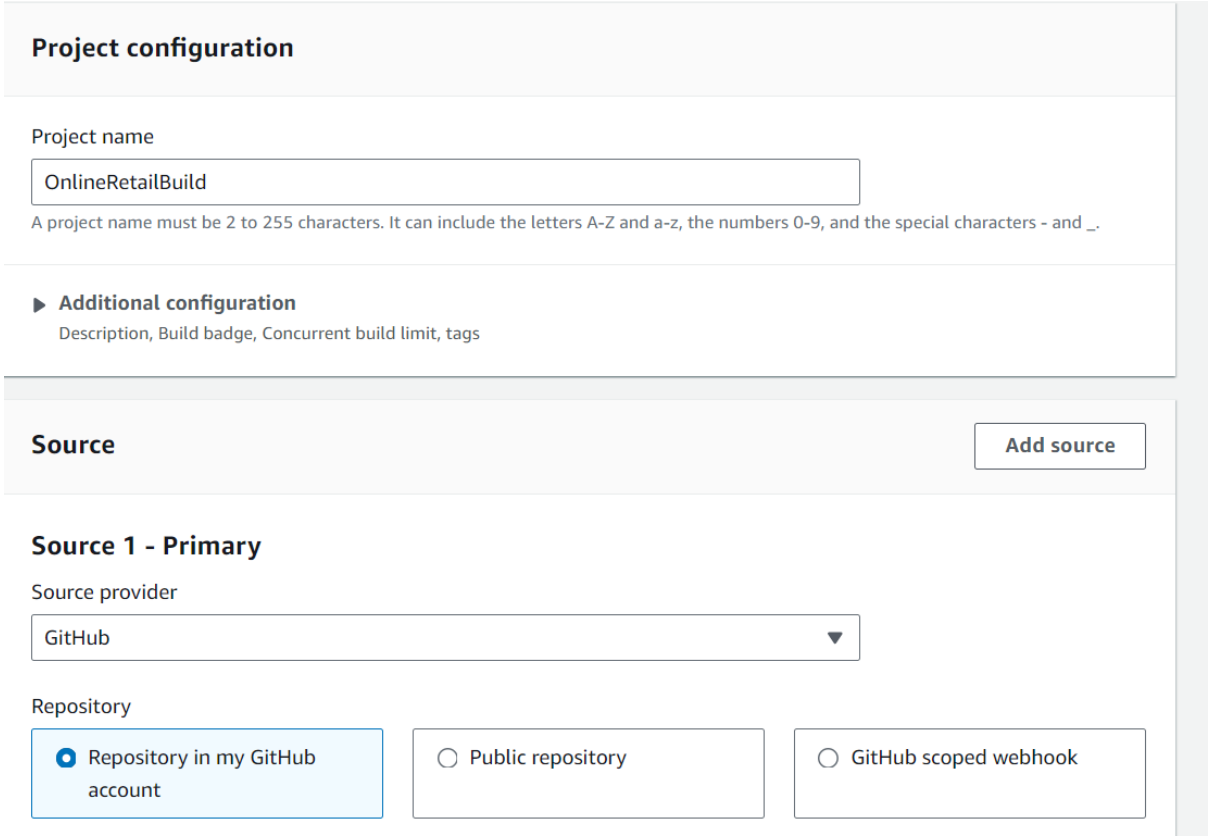


5. Finally add all the permissions required to the Role as below

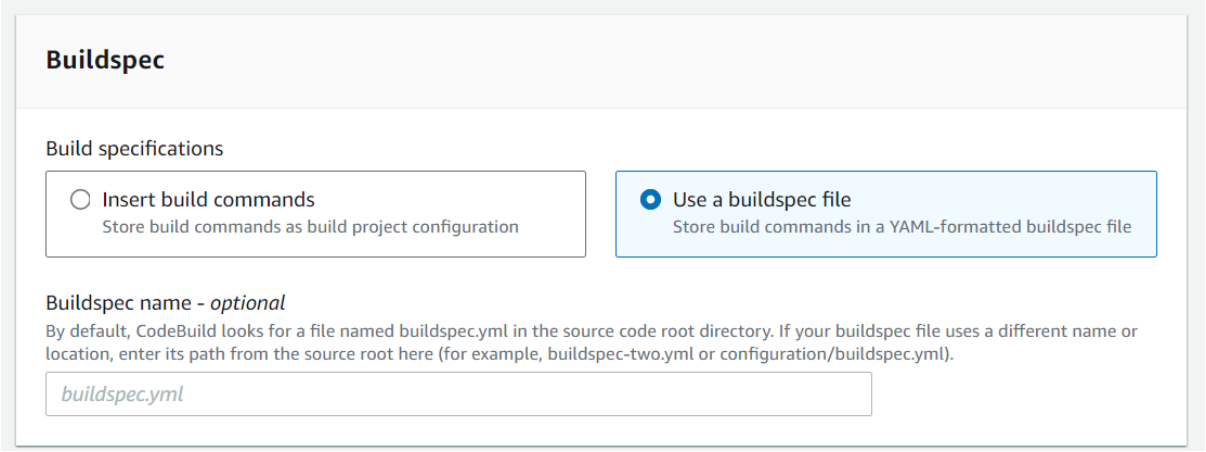


Part 11: AWS CodeBuild

1. Go to AWS CodeBuild console
2. Click on Create Project
3. Provide the details

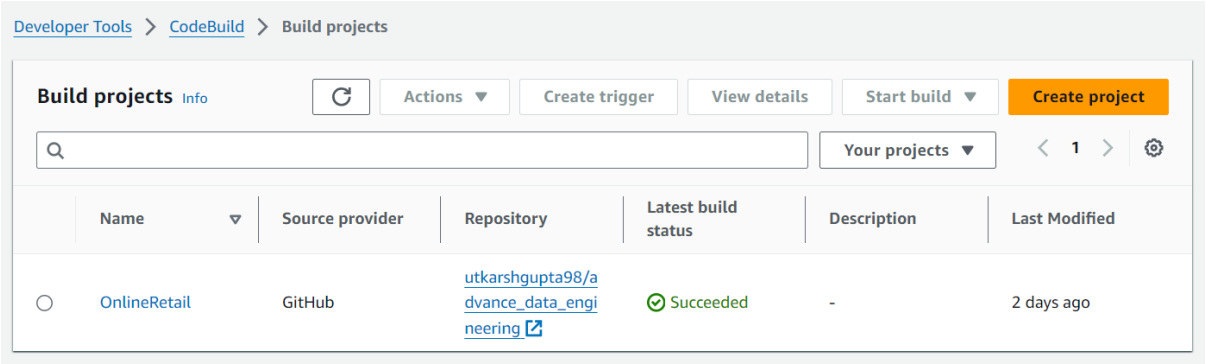


4. Create a buildspec.yaml file and push to your github

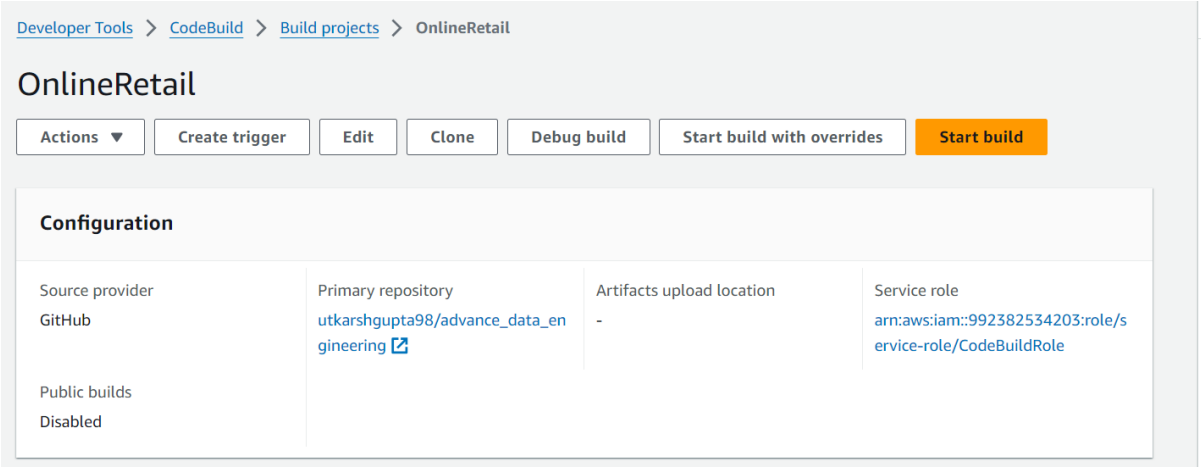


5. Keep everything else as it is

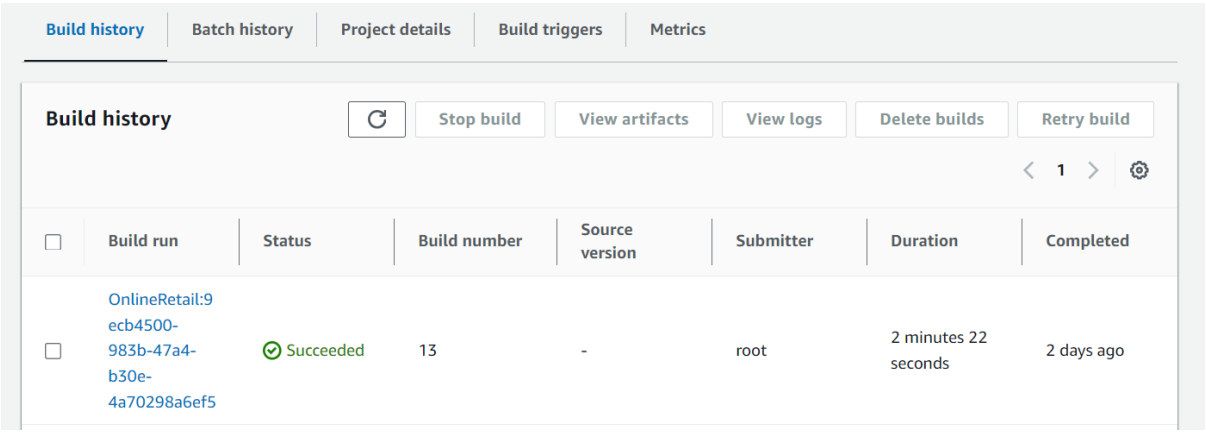
6. Once created it will come up as below



7. Click on Start build and remember to provide proper permission to IAM

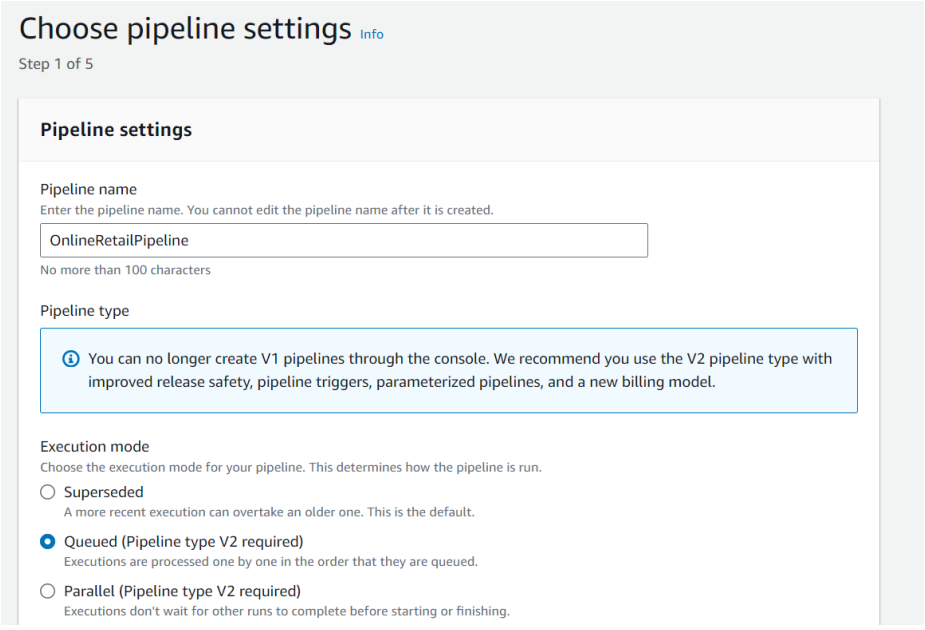


8. Check if Build is succeeded

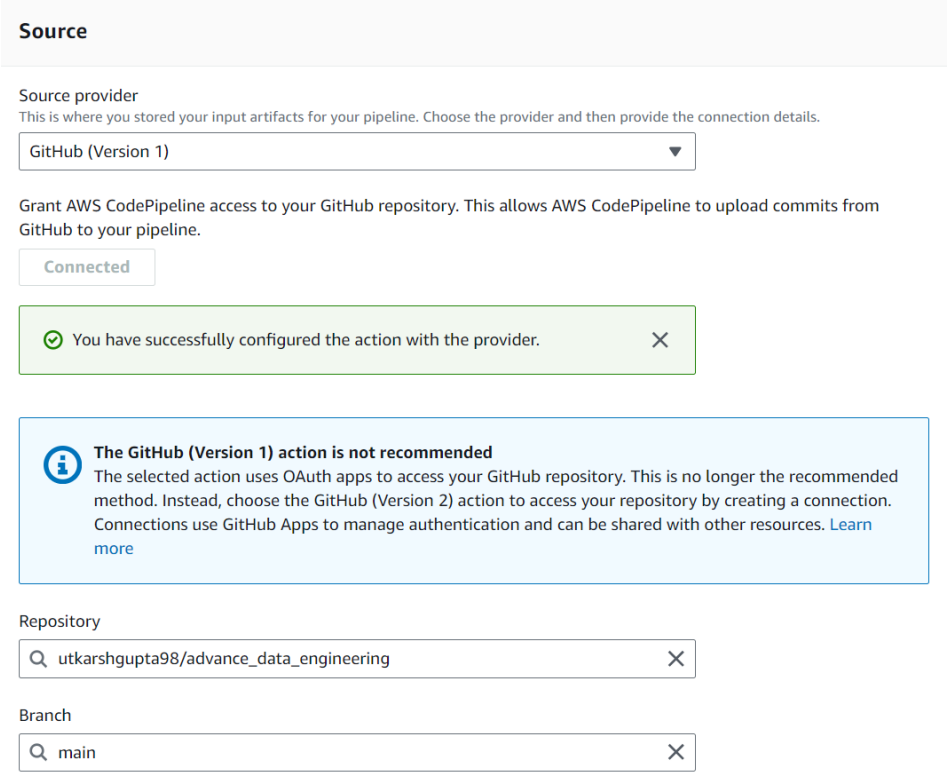


Part 12: AWS CodePipeline

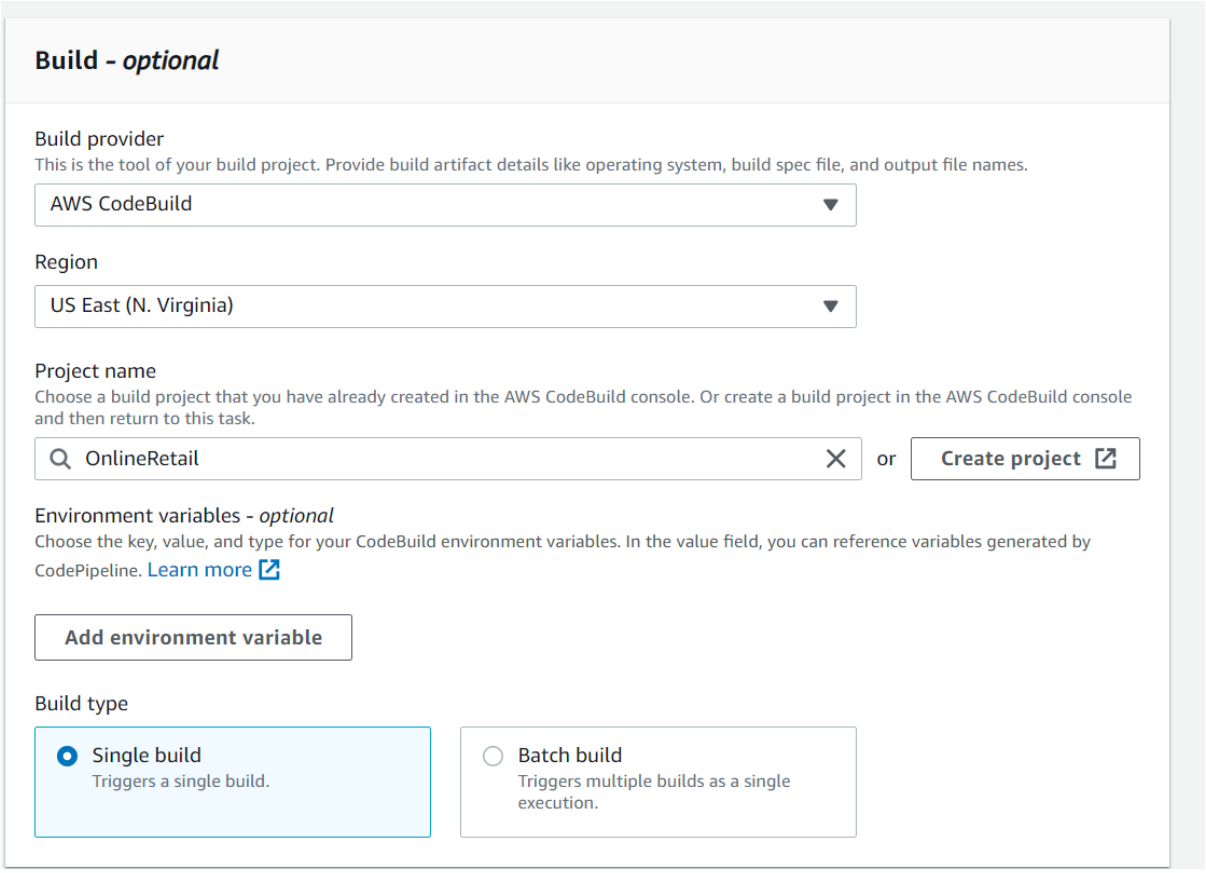
1. Go to AWS CodePipeline console
2. Click on Create Pipeline
3. Give details



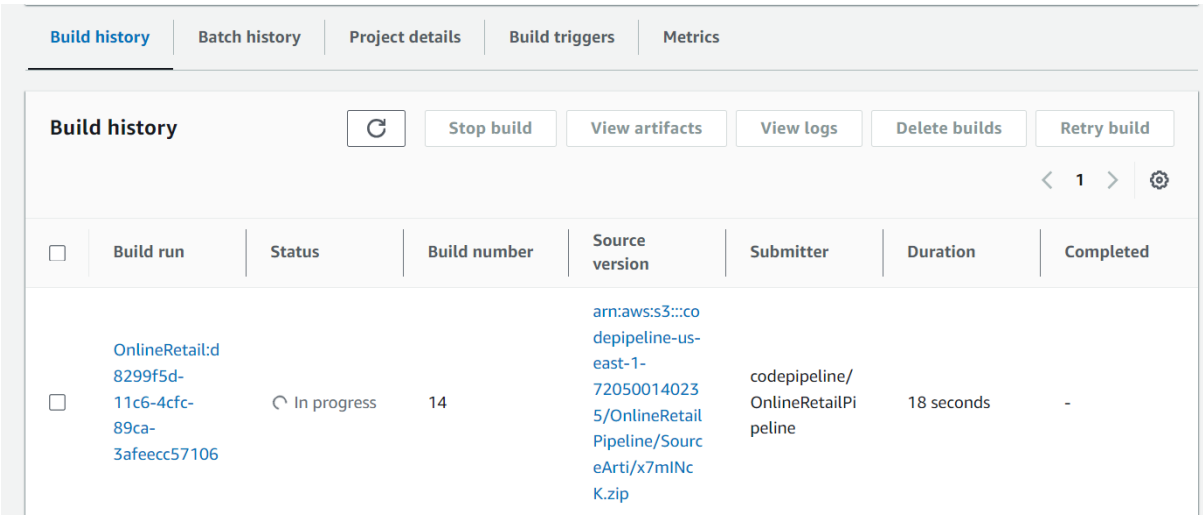
4. Provide Source Details



5. Add CodeBuild



6. Keep everything else as it is
7. Once you create a build will be triggered on CodeBuild



8. Check if all stages are green