Ashish Panchal
Assignment – 6
E-Commerce Data Pipeline

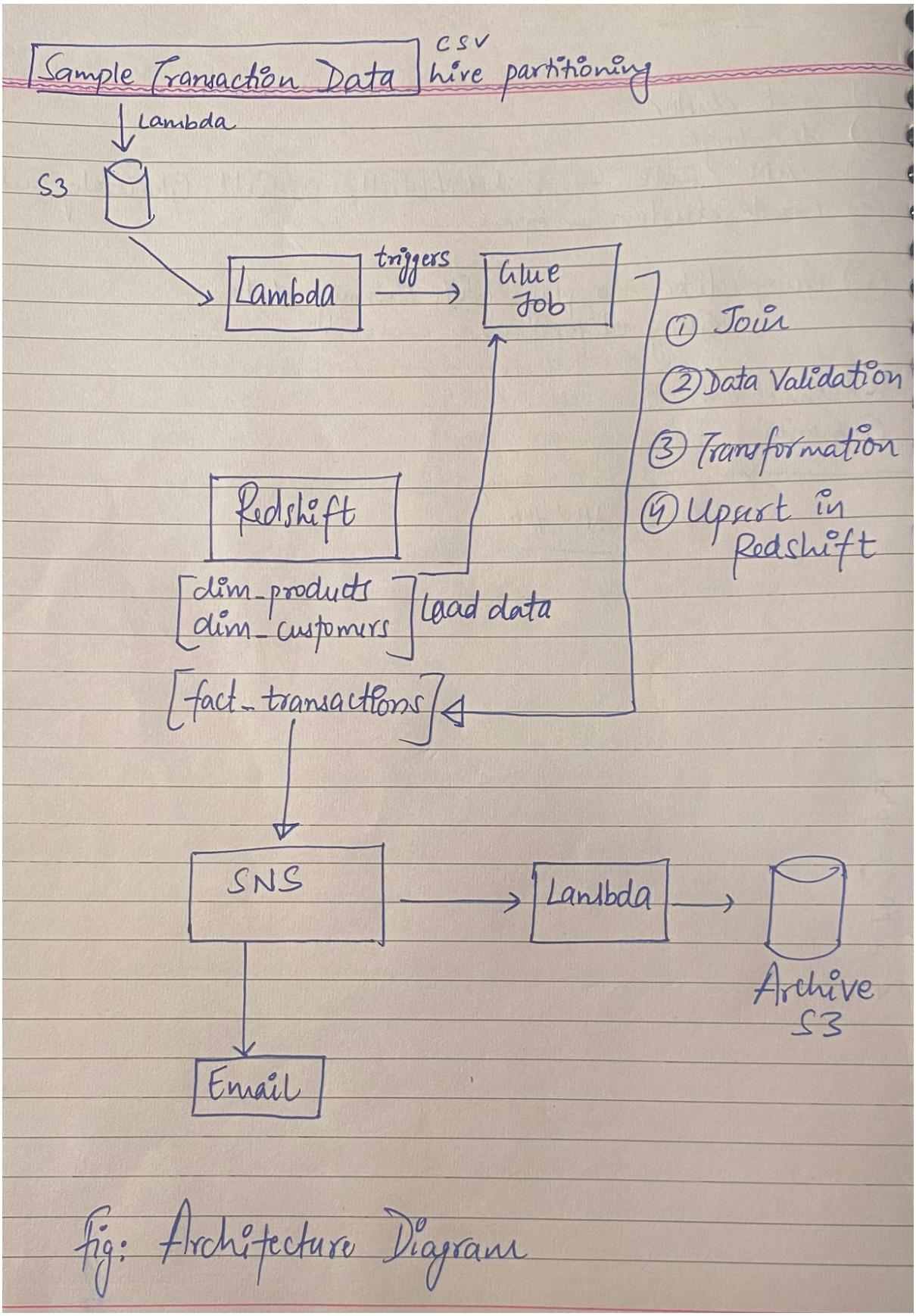
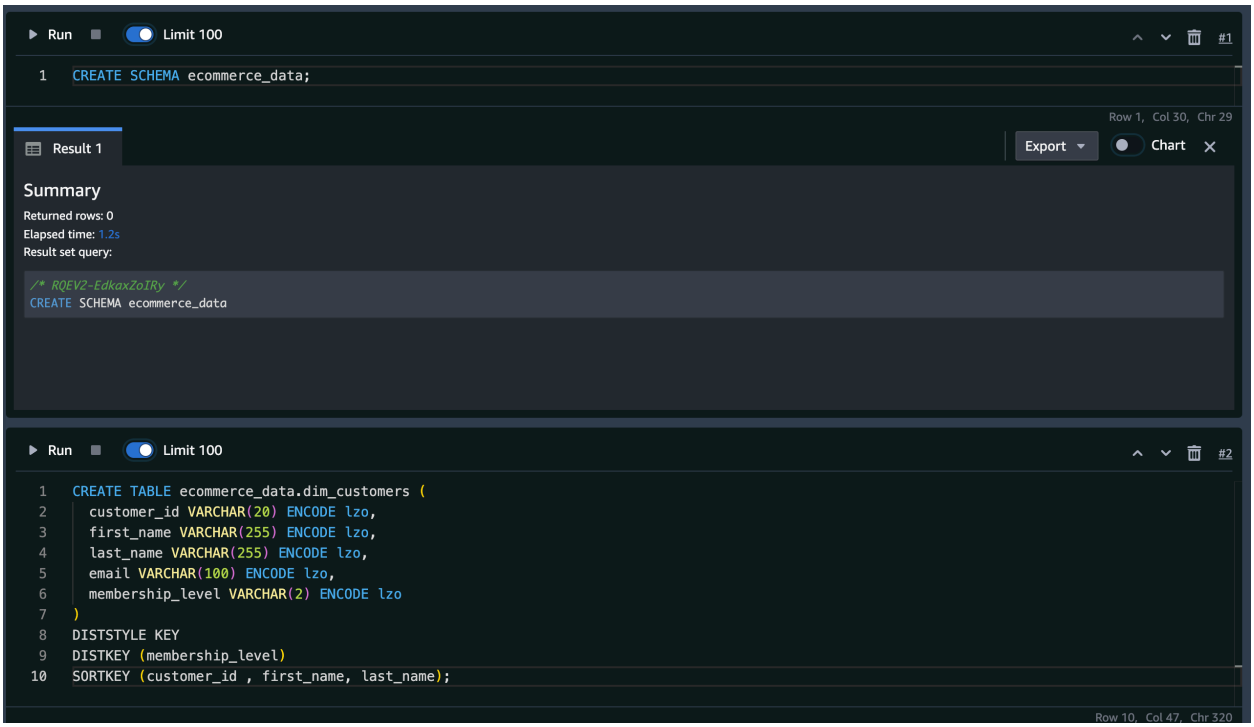
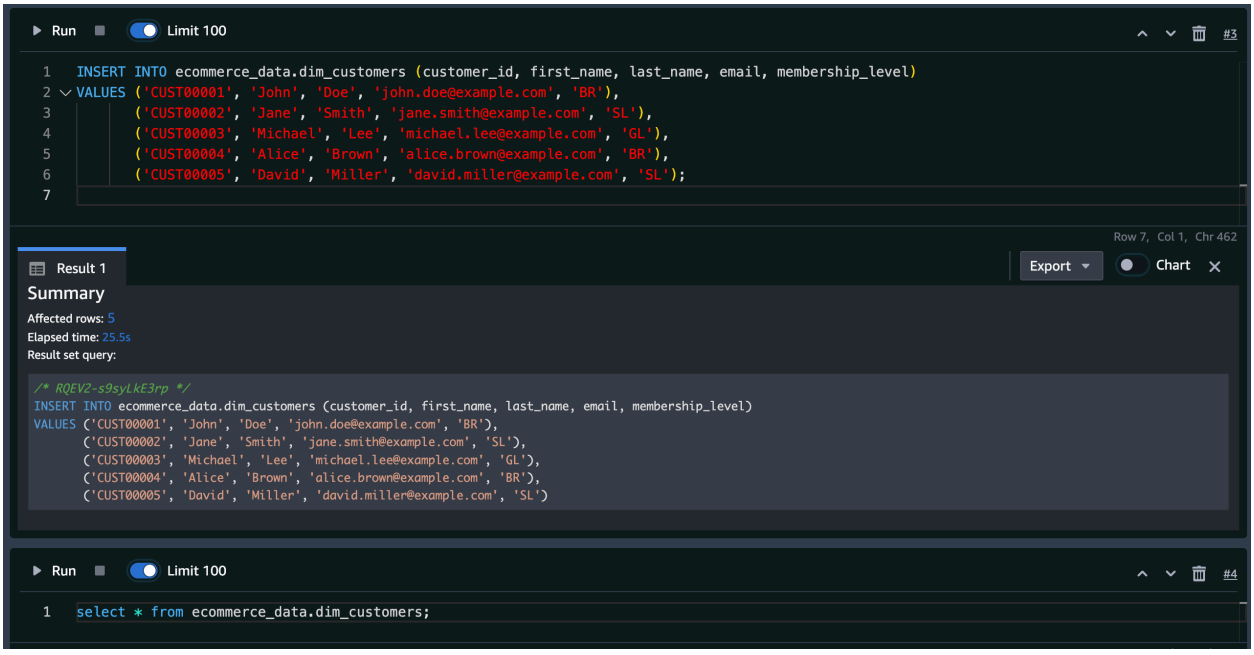


fig: Architecture Diagram

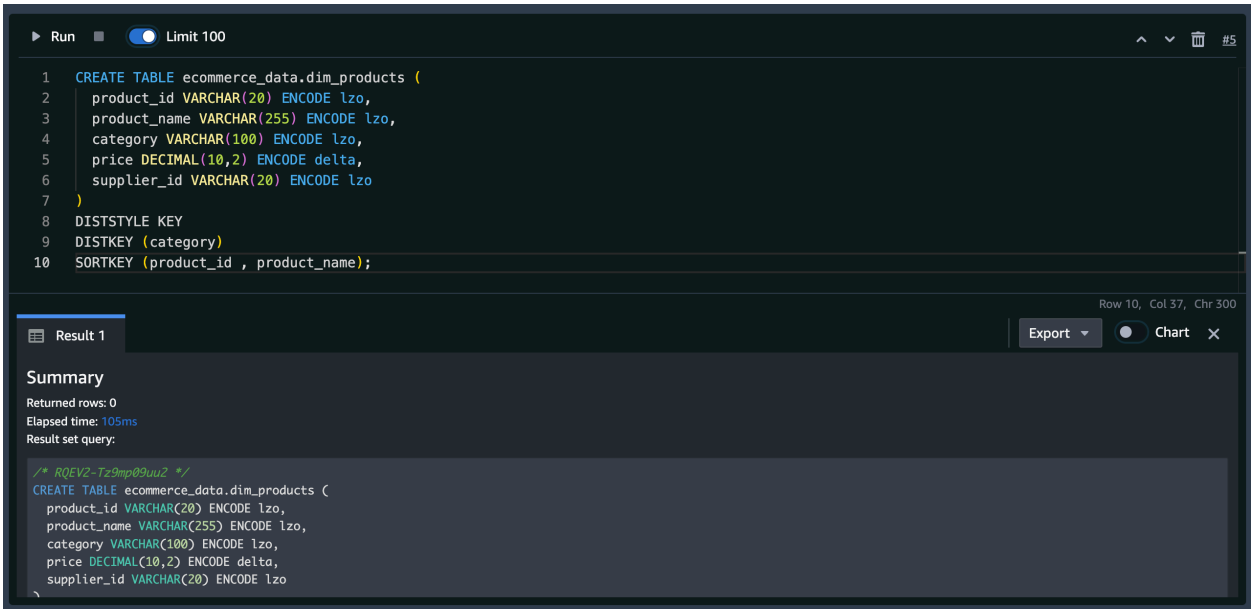
Ashish Panchal
Assignment – 6
E-Commerce Data Pipeline

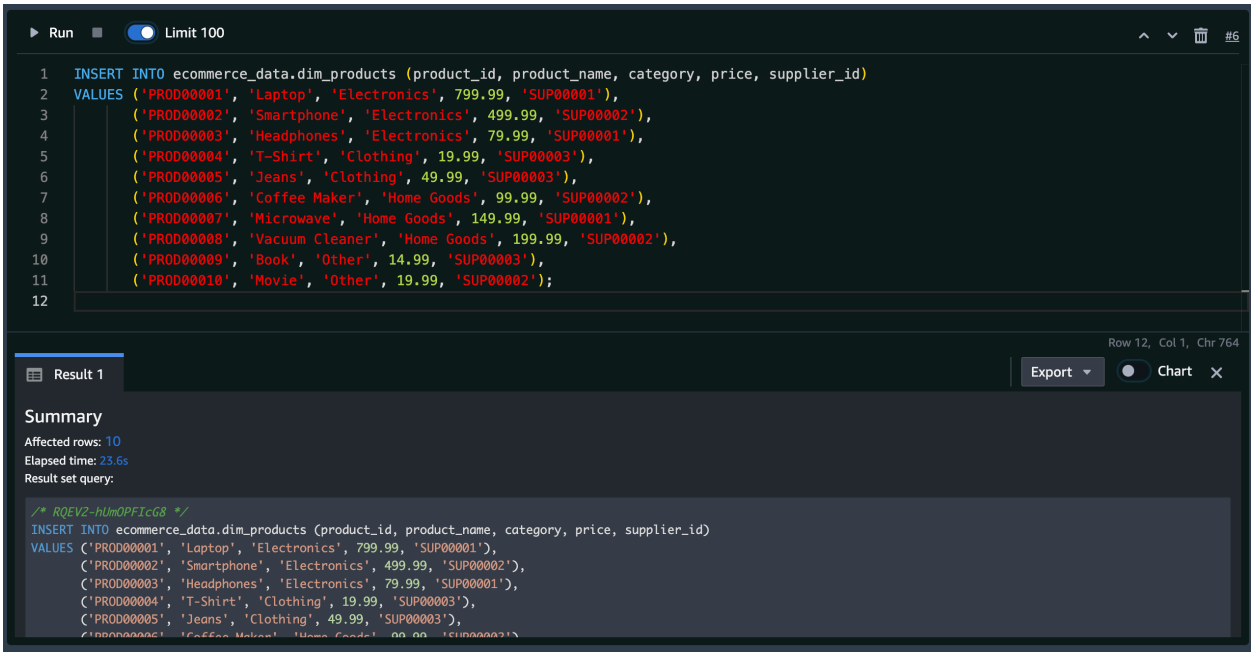
Create Tables in AWS Redshift and Load Data



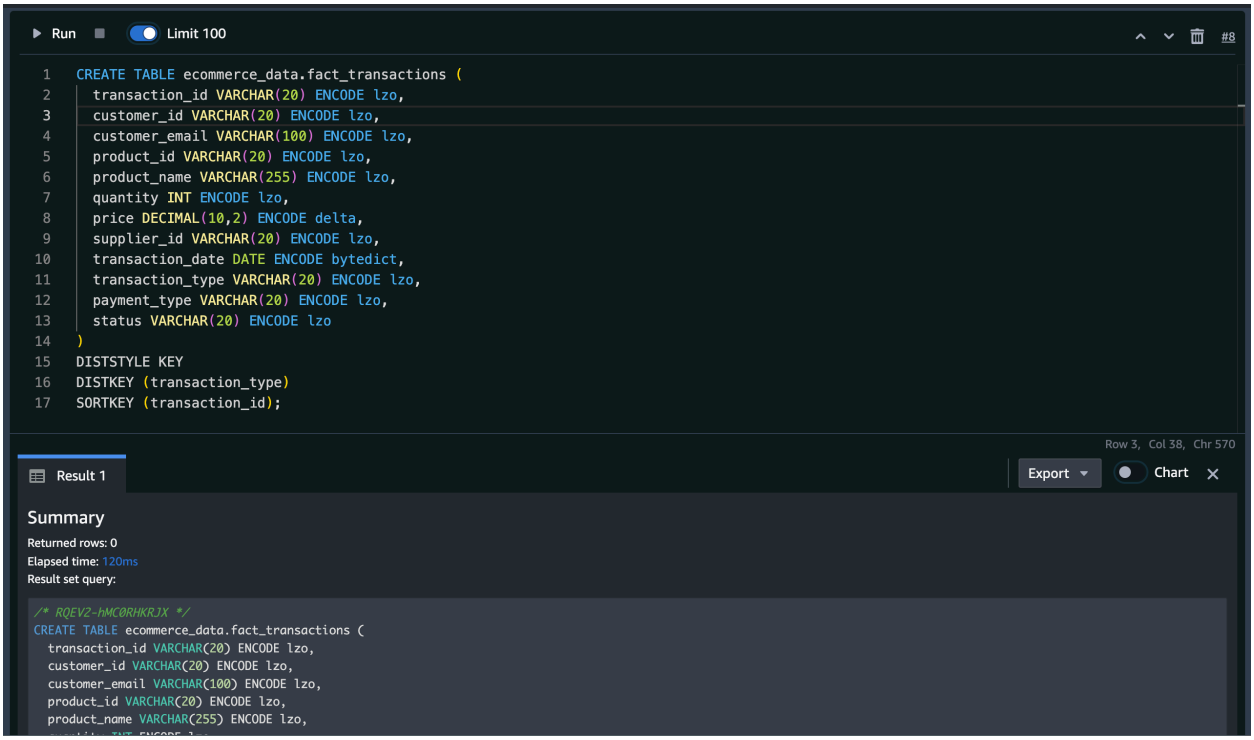


Ashish Panchal
Assignment – 6
E-Commerce Data Pipeline





Ashish Panchal
Assignment – 6
E-Commerce Data Pipeline

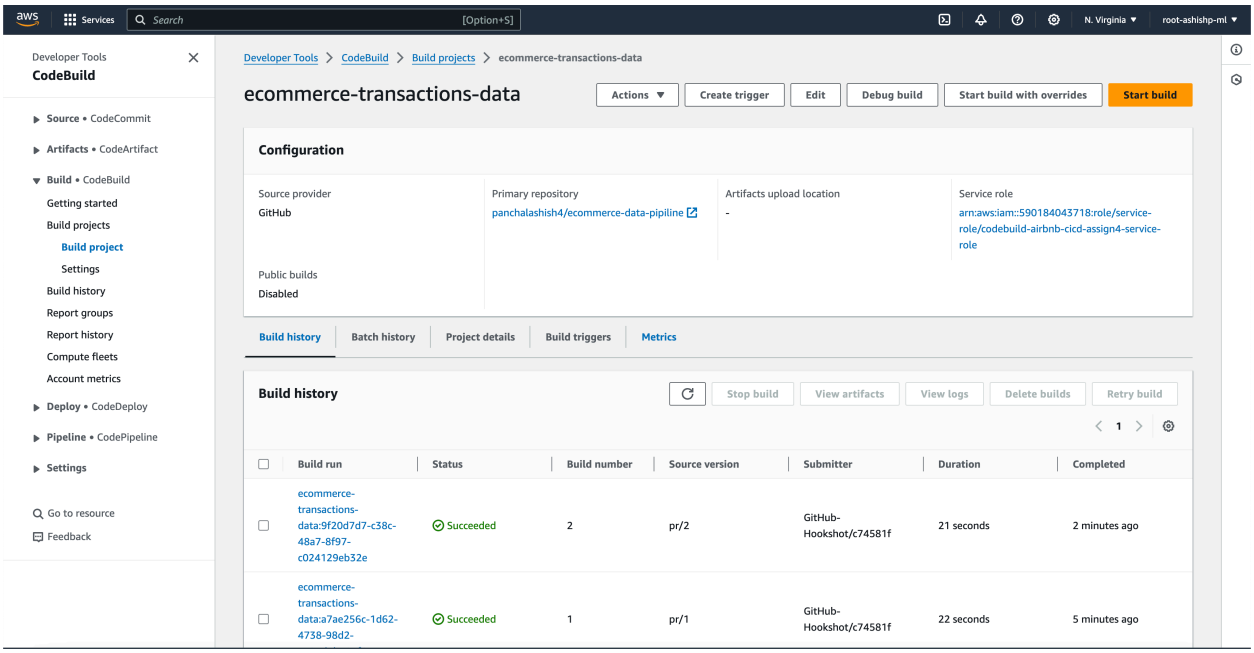


Generate Mock Data for the customers based on products. Use Code build for lambda. This lambda function will generate data and upload csv files into S3 in Hive Partitioning manner.

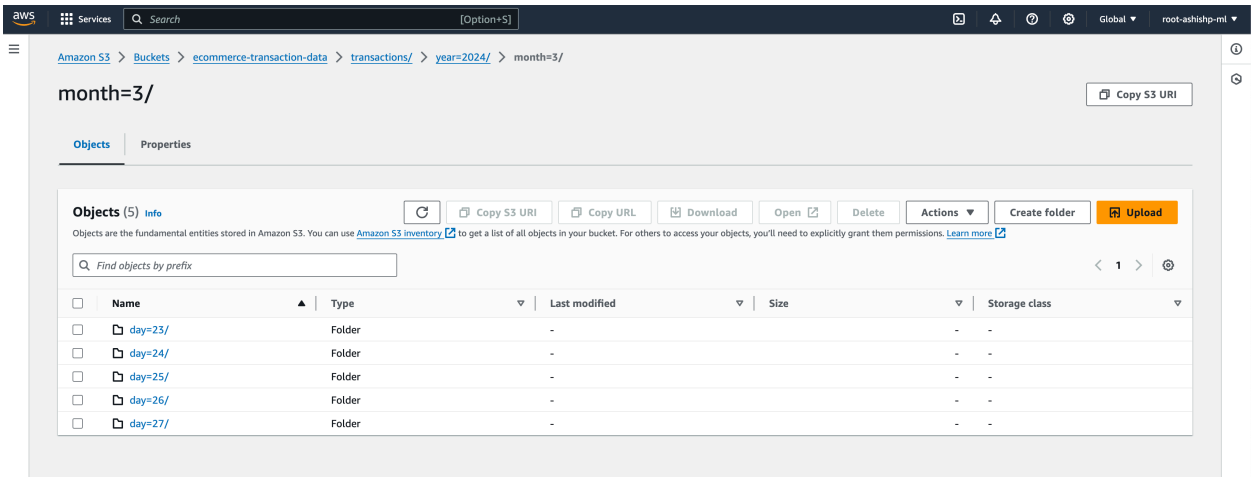
Python Code Link: https://github.com/panchalashish4/ecommerce-data-pipiline/tree/main/generate_data

Ashish Panchal
Assignment – 6
E-Commerce Data Pipeline

CodeBuild:

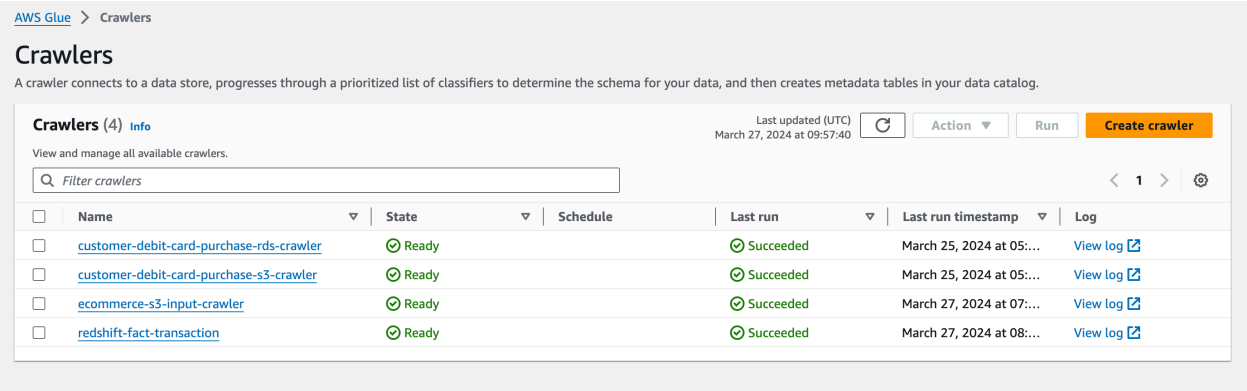


Test Lambda to Generate Data and Upload into S3

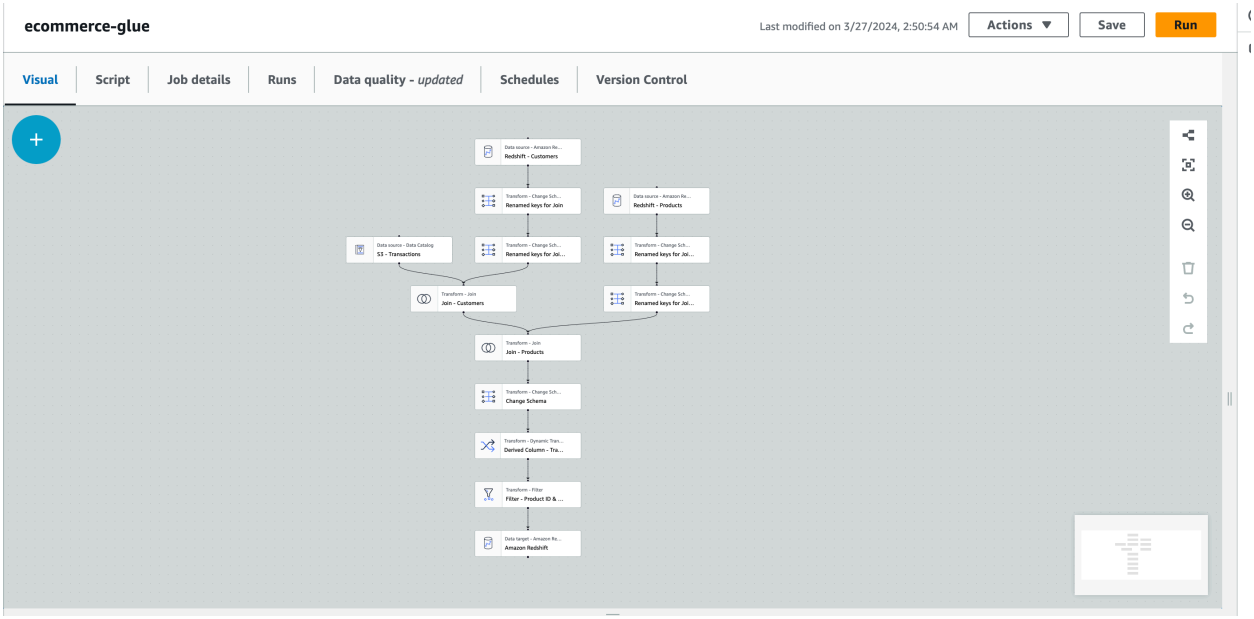


Ashish Panchal
Assignment – 6
E-Commerce Data Pipeline

Create crawlers in AWS Glue for new S3 bucket and AWS Redshift Tables



Create Glue ETL Flow as per our usecase



AWS Glue Script

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue import DynamicFrame
import gs_derived
```

Ashish Panchal
Assignment – 6
E-Commerce Data Pipeline

```
import re

args = getResolvedOptions(sys.argv, ['JOB_NAME'])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

# Script generated for node S3 - Transactions
S3Transactions_node1711523386952 =
glueContext.create_dynamic_frame.from_catalog(database="ecommerce-db",
table_name="s3_inputtransactions",
transformation_ctx="S3Transactions_node1711523386952")

# Script generated for node Redshift - Products
RedshiftProducts_node1711523340997 =
glueContext.create_dynamic_frame.from_catalog(database="ecommerce-db",
table_name="redshift_dev_ecommerce_data_dim_products",
redshift_tmp_dir="s3://aws-glue-assets-590184043718-us-east-1/temporary/",
transformation_ctx="RedshiftProducts_node1711523340997")

# Script generated for node Redshift - Customers
RedshiftCustomers_node1711523211825 =
glueContext.create_dynamic_frame.from_catalog(database="ecommerce-db",
table_name="redshift_dev_ecommerce_data_dim_customers",
redshift_tmp_dir="s3://aws-glue-assets-590184043718-us-east-1/temporary/",
transformation_ctx="RedshiftCustomers_node1711523211825")

# Script generated for node Renamed keys for Join - Products
RenamedkeysforJoinProducts_node1711523548475 =
ApplyMapping.apply(frame=RedshiftProducts_node1711523340997,
mappings=[("price", "decimal(10,2)", "price", "decimal"), ("product_id",
"string", "product_id", "string"), ("category", "string", "category",
"string"), ("product_name", "string", "product_name", "string"),
("supplier_id", "string", "supplier_id", "string")],
transformation_ctx="RenamedkeysforJoinProducts_node1711523548475")

# Script generated for node Renamed keys for Join
RenamedkeysforJoin_node1711523465265 =
ApplyMapping.apply(frame=RedshiftCustomers_node1711523211825,
mappings=[("customer_id", "string", "customer_id", "string"), ("first_name",
"string", "first_name", "string"), ("last_name", "string", "last_name",
"string"), ("email", "string", "email", "string"), ("membership_level",
"string", "membership_level", "string")],
transformation_ctx="RenamedkeysforJoin_node1711523465265")

# Script generated for node Renamed keys for Join - Products
RenamedkeysforJoinProducts_node1711531733910 =
ApplyMapping.apply(frame=RenamedkeysforJoinProducts_node1711523548475,
mappings=[("product_id", "string", "rs_product_id", "string"),
("product_name", "string", "rs_product_name", "string"), ("category",
"string", "rs_category", "string"), ("price", "decimal", "rs_price",
"decimal"), ("supplier_id", "string", "rs_supplier_id", "string")],
transformation_ctx="RenamedkeysforJoinProducts_node1711531733910")
```

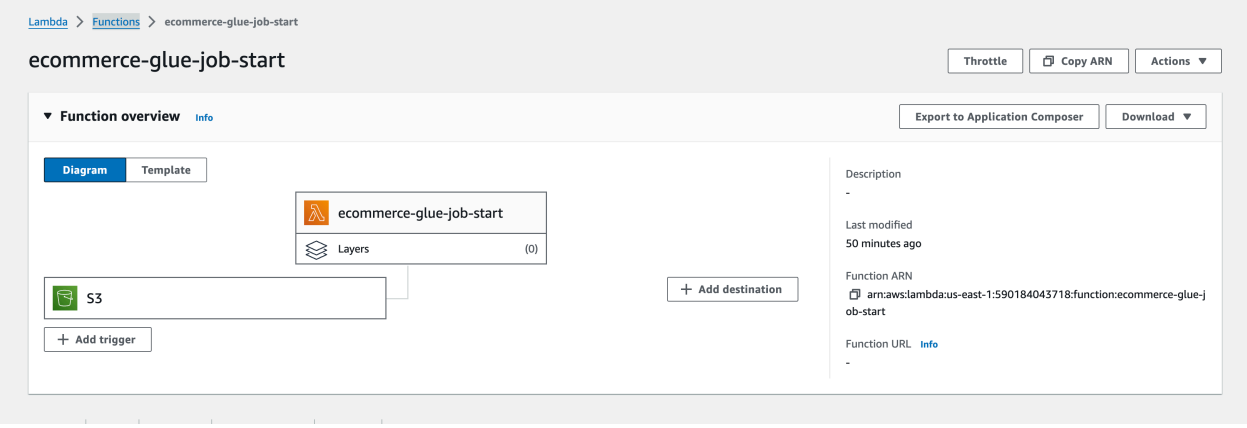
Ashish Panchal
Assignment – 6
E-Commerce Data Pipeline

```
# Script generated for node Renamed keys for Join - Customers
RenamedkeysforJoinCustomers_node1711531701689 =
ApplyMapping.apply(frame=RenamedkeysforJoin_node1711523465265,
mappings=[("customer_id", "string", "rs_customer_id", "string"),
("first_name", "string", "rs_first_name", "string"), ("last_name", "string",
"rs_last_name", "string"), ("email", "string", "rs_email", "string"),
("membership_level", "string", "rs_membership_level", "string")],
transformation_ctx="RenamedkeysforJoinCustomers_node1711531701689")

# Script generated for node Join - Customers
JoinCustomers_node1711523428722 =
Join.apply(frame1=S3Transactions_node1711523386952,
frame2=RenamedkeysforJoinCustomers_node1711531701689, keys1=["customer_id"],
keys2=["rs_customer_id"],
transformation_ctx="JoinCustomers_node1711523428722")

# Script generated for node Join - Products
JoinProducts_node1711523515185 =
Join.apply(frame1=JoinCustomers_node1711523428722,
frame2=RenamedkeysforJoinProducts_node1711531733910, keys1=["product_id"],
keys2=["rs_product_id"], transformation_ctx="JoinProducts_node1711523515185")

# Script generated for node Change Schema
ChangeSchema_node1711523572022 =
ApplyMapping.apply(frame=JoinProducts_node1711523515185, mappings=[("price",
"double", "price", "decimal"), ("product_id", "string", "product_id",
"varchar"), ("customer_id", "string", "customer_id", "varchar"),
("transaction_date", "string", "transaction_date", "date"), ("payment_type",
"string", "payment_type", "varchar"), ("transaction_id", "string",
"transaction_id", "varchar"), ("status", "string", "status", "varchar"),
("rs_email", "string", "customer_email", "varchar"), ("rs_product_name",
"string", "product_name", "varchar"), ("rs_supplier_id", "string",
"supplier_id", "varchar")],
transformation_ctx="ChangeSchema_node1711523572022")

# Script generated for node Derived Column - Transaction_Type
DerivedColumnTransaction_Type_node1711523858206 =
ChangeSchema_node1711523572022.gs_derived(colName="transaction_type",
expr="case when price < 100 then 'small' when price > 100 and price < 500
then 'medium' when price > 500 then 'large' else 'not defined' end")

# Script generated for node Filter - Product ID & Customer ID
FilterProductIDCustomerID_node1711525042051 =
Filter.apply(frame=DerivedColumnTransaction_Type_node1711523858206, f=lambda
row: (bool(re.match("^PROD[0-9]{5}$", row["product_id"])) and
bool(re.match("^CUST[0-9]{5}$", row["customer_id"]))),
transformation_ctx="FilterProductIDCustomerID_node1711525042051")

# Script generated for node Amazon Redshift
AmazonRedshift_node1711532392573 =
glueContext.write_dynamic_frame.from_options(frame=FilterProductIDCustomerID_
node1711525042051, connection_type="redshift",
connection_options={"postactions": "BEGIN; DELETE FROM
ecommerce_data.fact_transactions USING
```

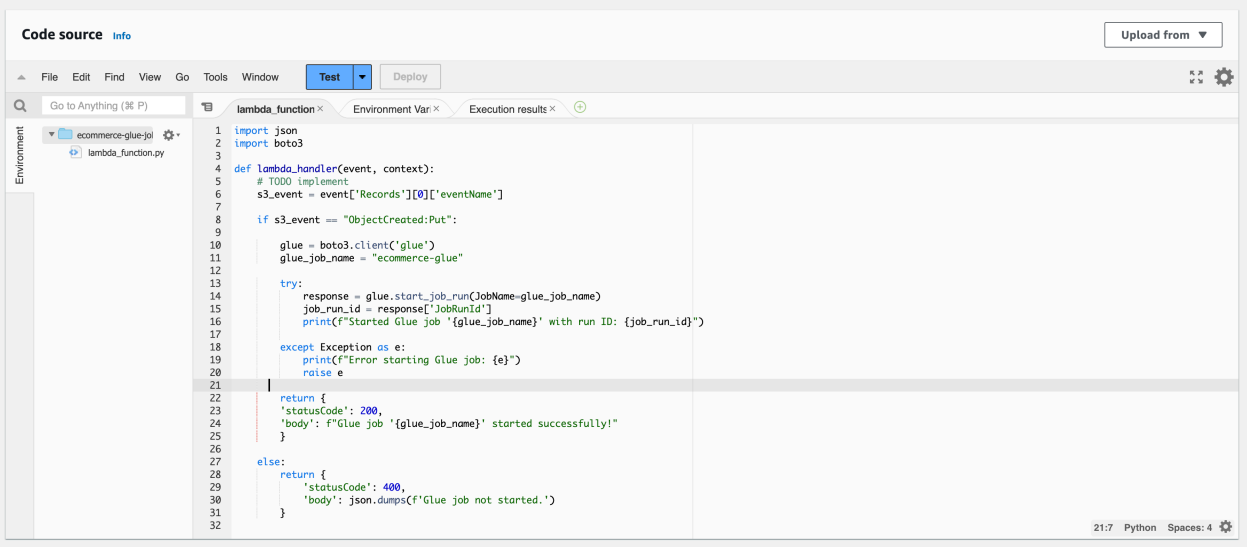
Ashish Panchal
Assignment – 6
E-Commerce Data Pipeline

```
ecommerce_data.fact_transactions_temp_09nd7y WHERE
ecommerce_data.fact_transactions_temp_09nd7y.transaction_id =
ecommerce_data.fact_transactions.transaction_id; INSERT INTO
ecommerce_data.fact_transactions SELECT * FROM
ecommerce_data.fact_transactions_temp_09nd7y; DROP TABLE
ecommerce_data.fact_transactions_temp_09nd7y; END;", "redshiftTmpDir":
"s3://aws-glue-assets-590184043718-us-east-1/temporary/",
"useConnectionProperties": "true", "dbtable":
"ecommerce_data.fact_transactions_temp_09nd7y", "connectionName": "Redshift
connection", "preactions": "CREATE TABLE IF NOT EXISTS
ecommerce_data.fact_transactions (price DECIMAL, product_id VARCHAR,
customer_id VARCHAR, transaction_date DATE, payment_type VARCHAR,
transaction_id VARCHAR, status VARCHAR, customer_email VARCHAR, product_name
VARCHAR, supplier_id VARCHAR, transaction_type VARCHAR); DROP TABLE IF EXISTS
ecommerce_data.fact_transactions_temp_09nd7y; CREATE TABLE
ecommerce_data.fact_transactions_temp_09nd7y AS SELECT * FROM
ecommerce_data.fact_transactions WHERE 1=2;"},
transformation_ctx="AmazonRedshift_node1711532392573")

job.commit()
```

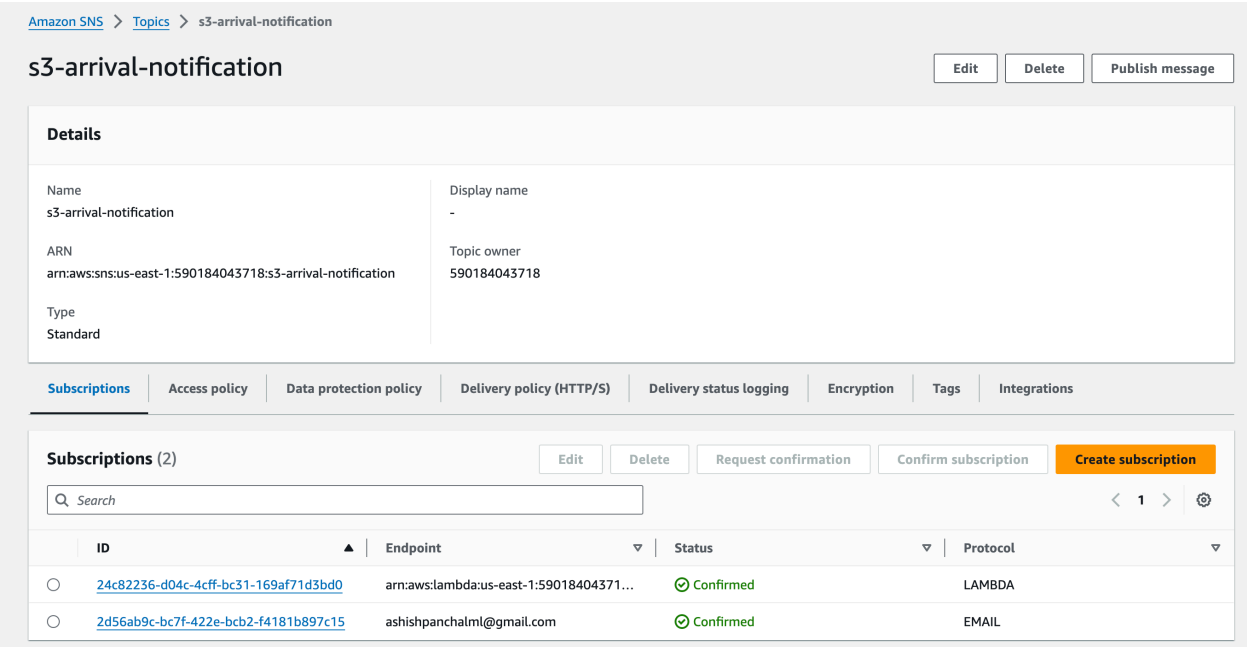
Create lambda to start glue job when data is generated in S3:



Ashish Panchal
Assignment – 6
E-Commerce Data Pipeline

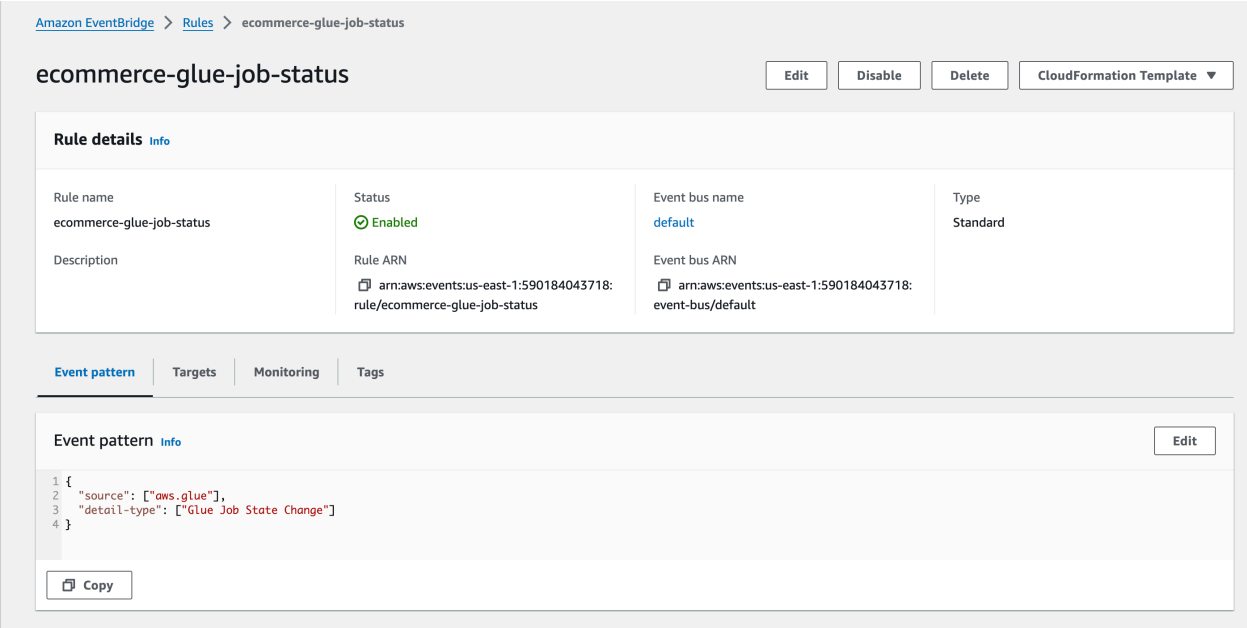


Create SNS Topic to send email when Glue ETL job is done, and it can be used in final lambda to archive the data

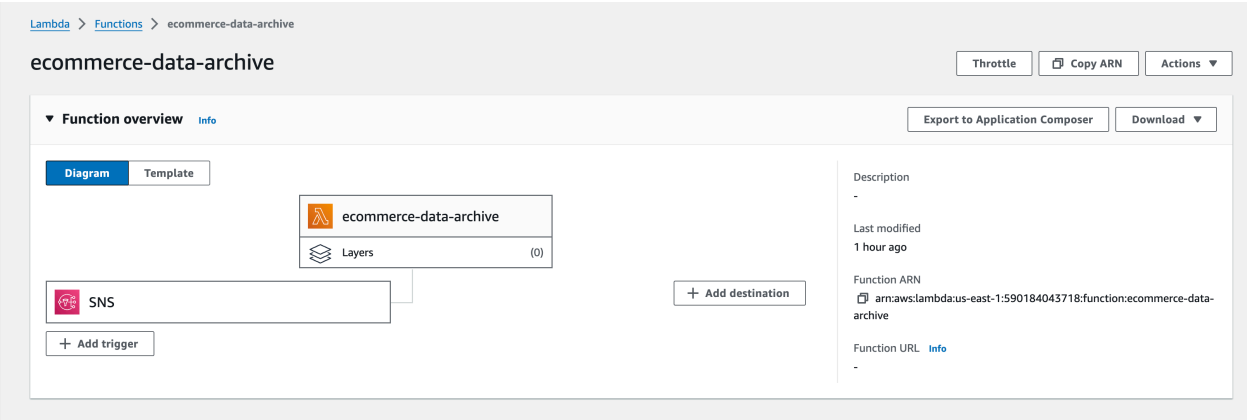


Ashish Panchal
Assignment – 6
E-Commerce Data Pipeline

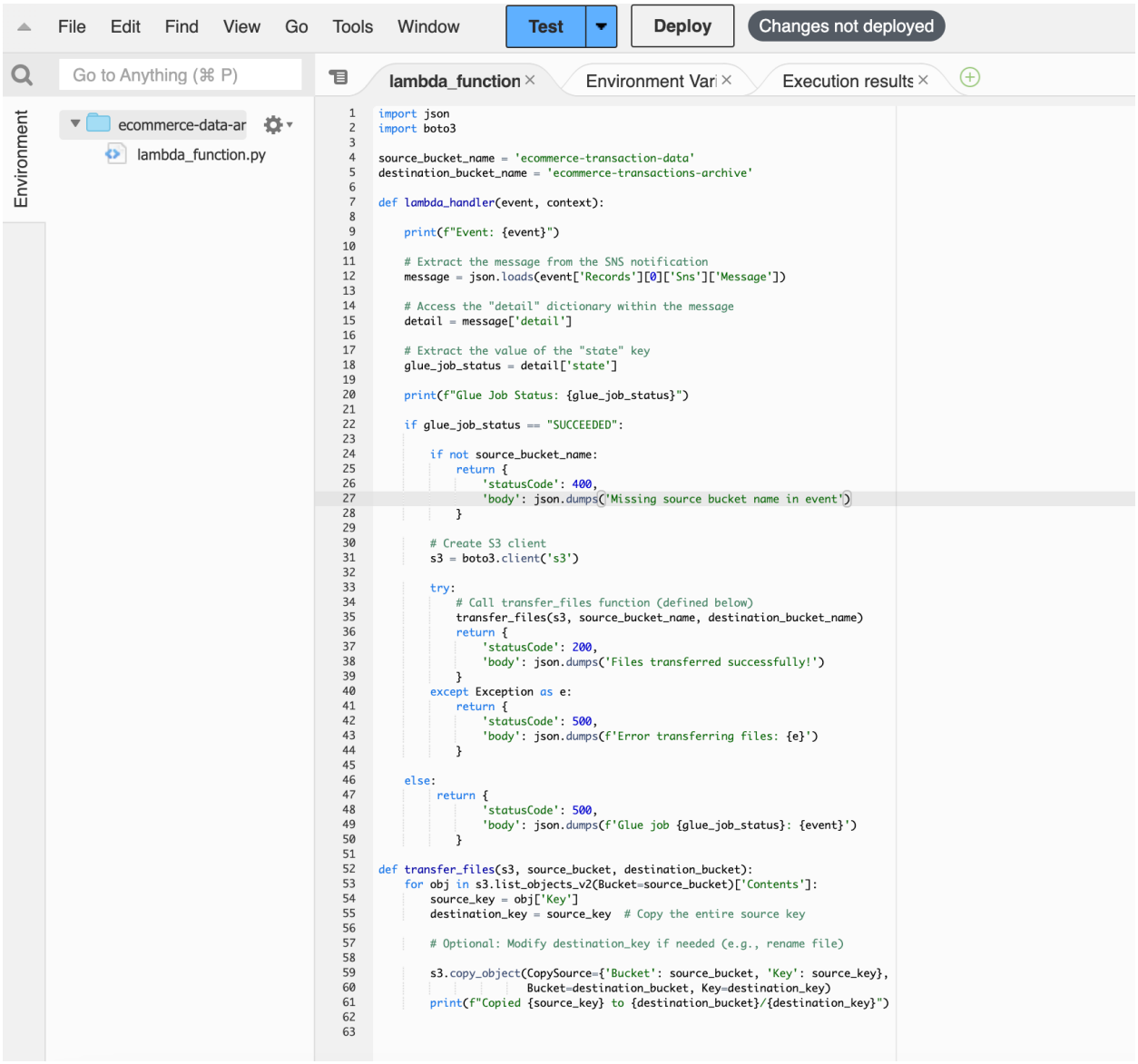
Create EventBridge Rule to trigger SNS when Glue Job Changes Status:



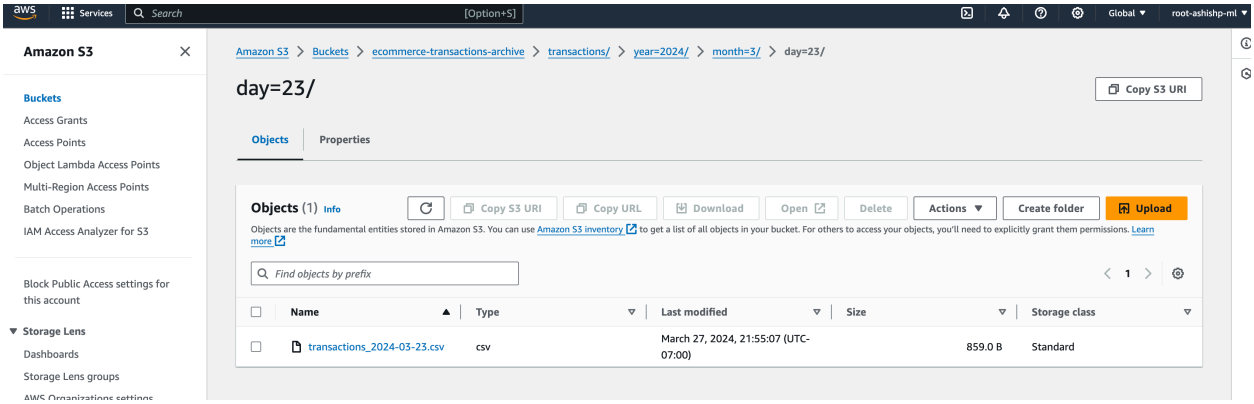
Create final Lambda which will archive data after SNS notification:



Ashish Panchal
Assignment – 6
E-Commerce Data Pipeline

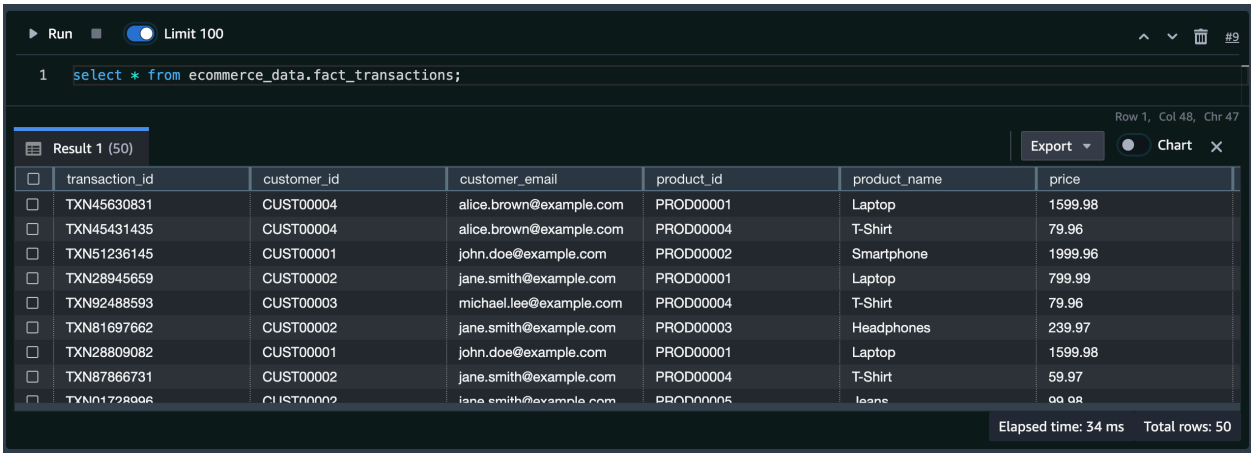


Create final S3 bucket where data is archived:



Ashish Panchal
Assignment – 6
E-Commerce Data Pipeline

Check if data is populated in RedShift fact_transactions table:



Once we execute the generate_mock_data Lambda, this whole pipeline flows as seen in the screenshots and executes/passes through different AWS services and finally data is archived in S3 bucket.

This completes the assignment.