Capstone Project

Requirement Analysis

Objectives:

1. Develop a scalable Data and Analytics Infrastructure on the cloud.
2. Enable data availability across different entities of the organization.
3. Ingest a variety of data sources efficiently.
4. Leverage a common infrastructure to overcome performance bottlenecks and reduce costs.
5. Reduce new ETL and BI development effort with increased productivity.
6. Support operational and BI reporting & dashboards, ad-hoc reporting, and analytical applications.

Key Challenges in current architecture:

1. Operational silos leading to fragmented data.
2. Performance bottlenecks affecting data availability.
3. Aging Architecture using Teradata as EDW.
4. Lack of a single version of truth across the enterprise.
5. Manual reporting on Excel and limited executive-level dashboards.
6. Limited support for modern analytics tools and data sources.

Proposed Execution Strategy:

1. Data Sources: Ingest data from multiple sources like csv and live API.
2. Storage Services: Use S3 for storing processed and unprocessed data.
3. Data Processing and Ingestion: Use Amazon Kinesis for high performance real-time data streaming, Glue for processing batch data, Lambda to trigger any jobs on events.
4. Data Structuring: Using AWS Glue Crawler, Glue Data Catalog to structure raw data from S3 do ETL and store the processed data in S3.
5. Data Lake: Using S3 buckets to store all the processed data.
6. Data Analysis: Using Athena to run queries, do analysis and build dashboards to analyze all the data.
7. Data Communication: Using SNS and SMTP protocol to communicate with users and admins, to send emails to specific users etc.

Functional Requirements:

1. Data Ingestion:
    a. Capability to ingest structured and unstructured data.
    b. Real-time and batch data ingestion functionalities.
    c. Integration with third-party sources, including ClickStream, Social Media, and External HR Data.

2. Scalable Data Warehouse:
    a. Cloud-based scalable data warehouse solution.
    b. Ability to handle large volumes of data efficiently.

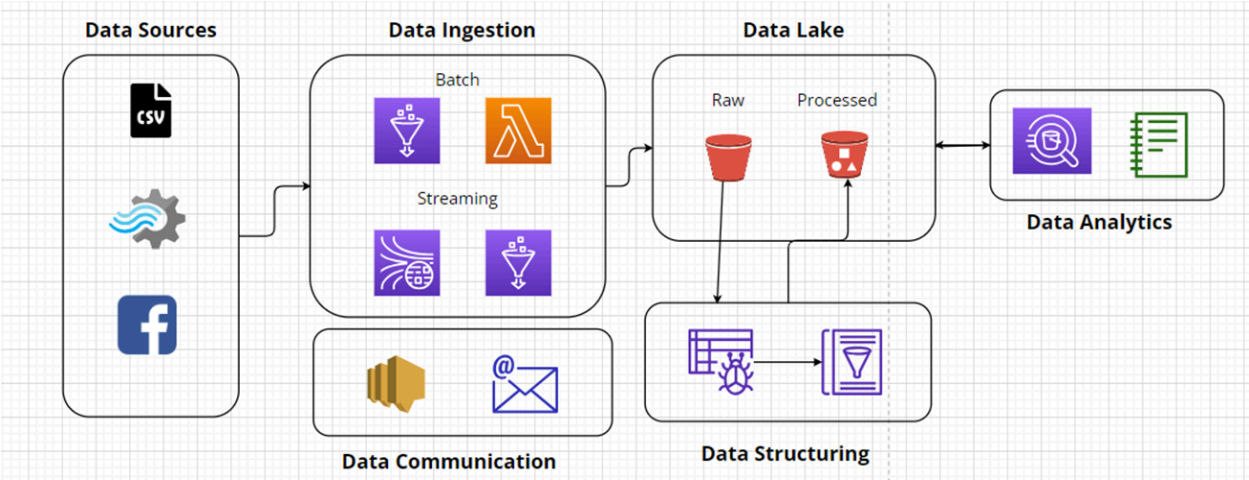
    c. Support for parallel processing to enhance performance.

3. Reporting and Visualization:
    a. Ad-hoc reporting features for flexible data exploration.
    b. Executive-level dashboards providing a holistic view.
    c. Integration with external reporting tools, such as Power BI.

Non-Functional Requirements:

1. Performance:
    a. Reduced data latency, ensuring data availability within two hours.
    b. Improved query performance for reporting and analytics.
2. Security:
    a. Implementation of role-based access control for data security.
    b. Encryption for data in transit and at rest.
    c. Compliance with relevant data protection regulations.
3. Scalability:
    a. Horizontal scalability to accommodate increased data volume.
    b. Provision for scaling resources based on demand.
    c. Support for future data growth.
4. Reliability:
    a. High availability of the data platform to minimize downtime.
    b. Implementation of disaster recovery and backup mechanisms.
5. Usability:
    a. User-friendly interfaces for reporting and analytical tools.
    b. Intuitive data exploration features for end-users.
6. Interoperability:
    a. Seamless integration with existing systems and applications.
    b. Compatibility with various data formats and sources.
7. Availability:
    a. 24/7 availability of critical data and reporting services.
    b. Minimal scheduled maintenance windows to ensure continuous operation.
8. Auditability:
    a. Logging and auditing capabilities to track data access and modifications.
    b. Generation of comprehensive audit reports for compliance purposes.
9. Monitoring and Alerting:
    a. Real-time monitoring of data platform performance.
    b. Configurable alerting system for proactive issue resolution.
10. Comprehensive Documentation:
    a. Well-documented system architecture, configurations, and processes.
    b. User manuals and guides for end-users and administrators.

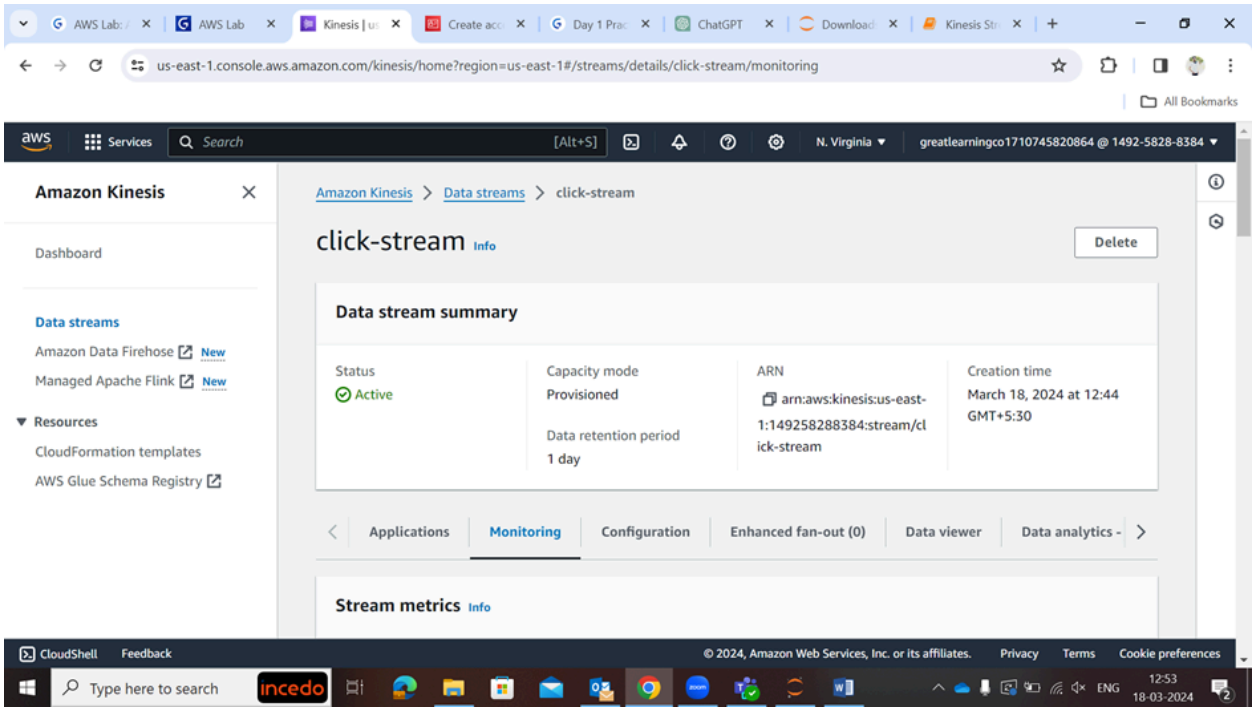
High Level Architecture:



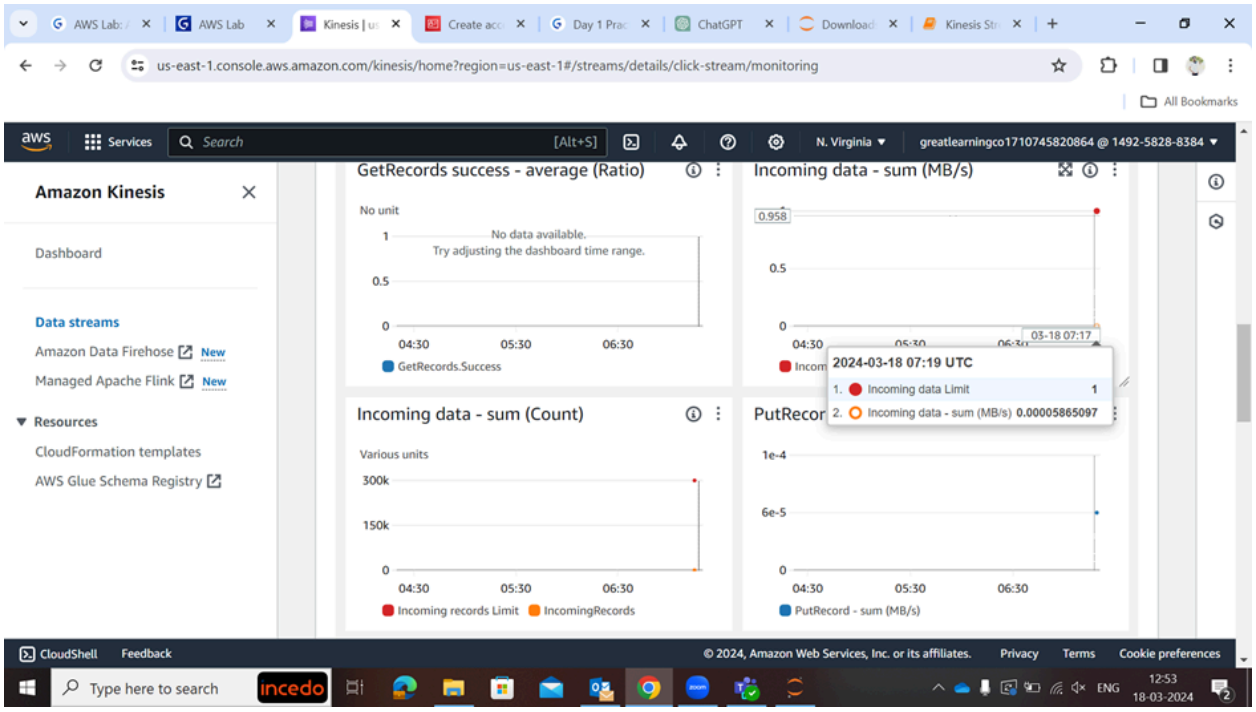
AWS Kinesis for Data Ingestion:

1. Records are generated using Faker library in Python simulating a real time streaming.
2. These records are put into the kinesis stream using boto3 AWS SDK.
3. Then a simple Glue job converts them into a CSV file. This CSV file then goes for further ETL using Glue Studio.

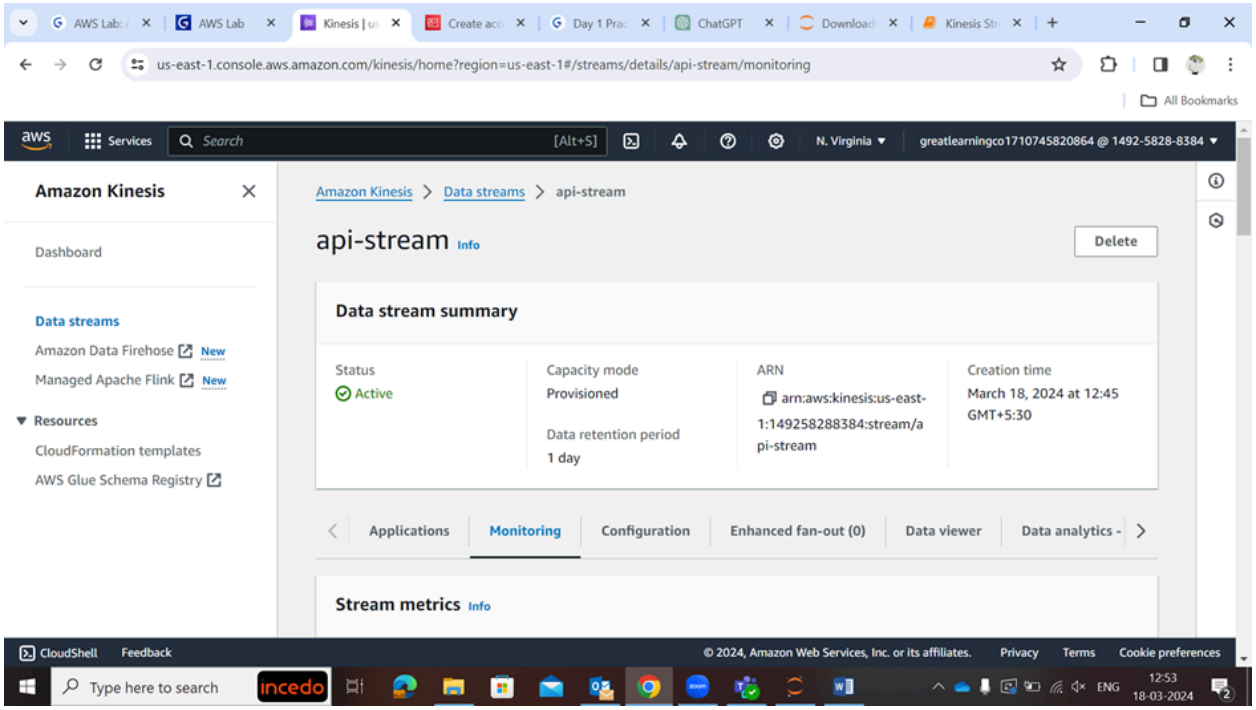
Clickstream kinesis stream:



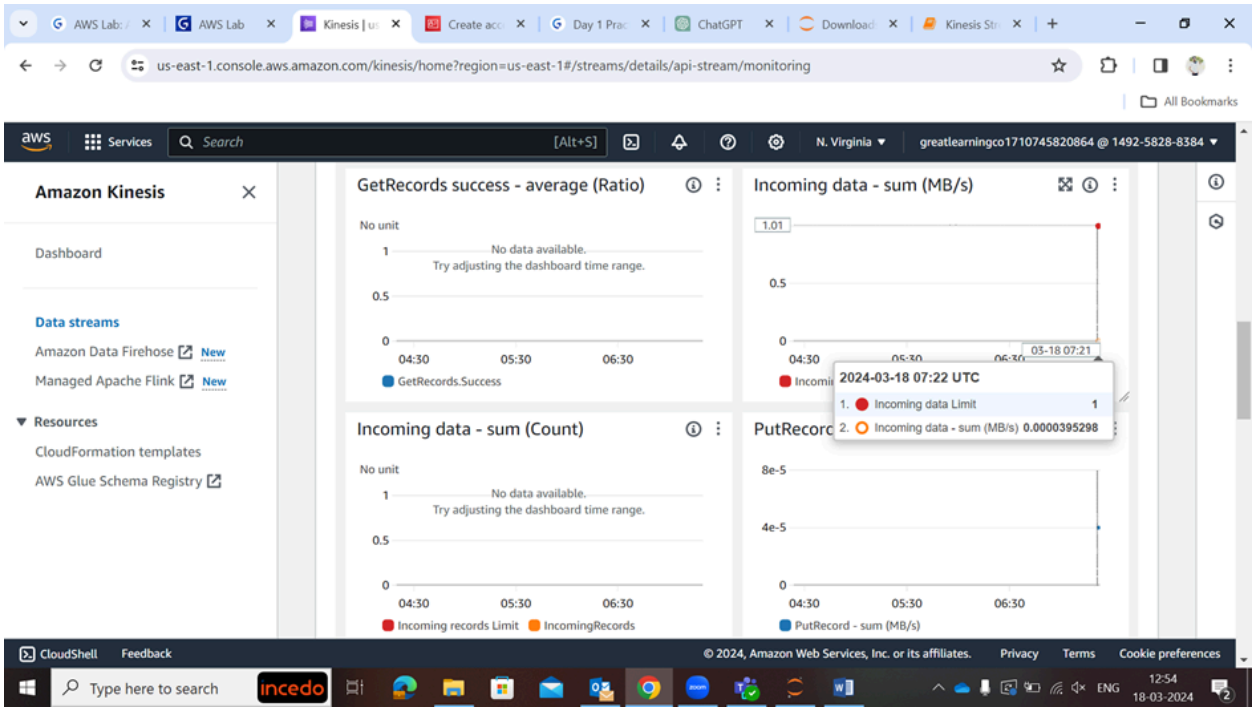
Data is put in stream:



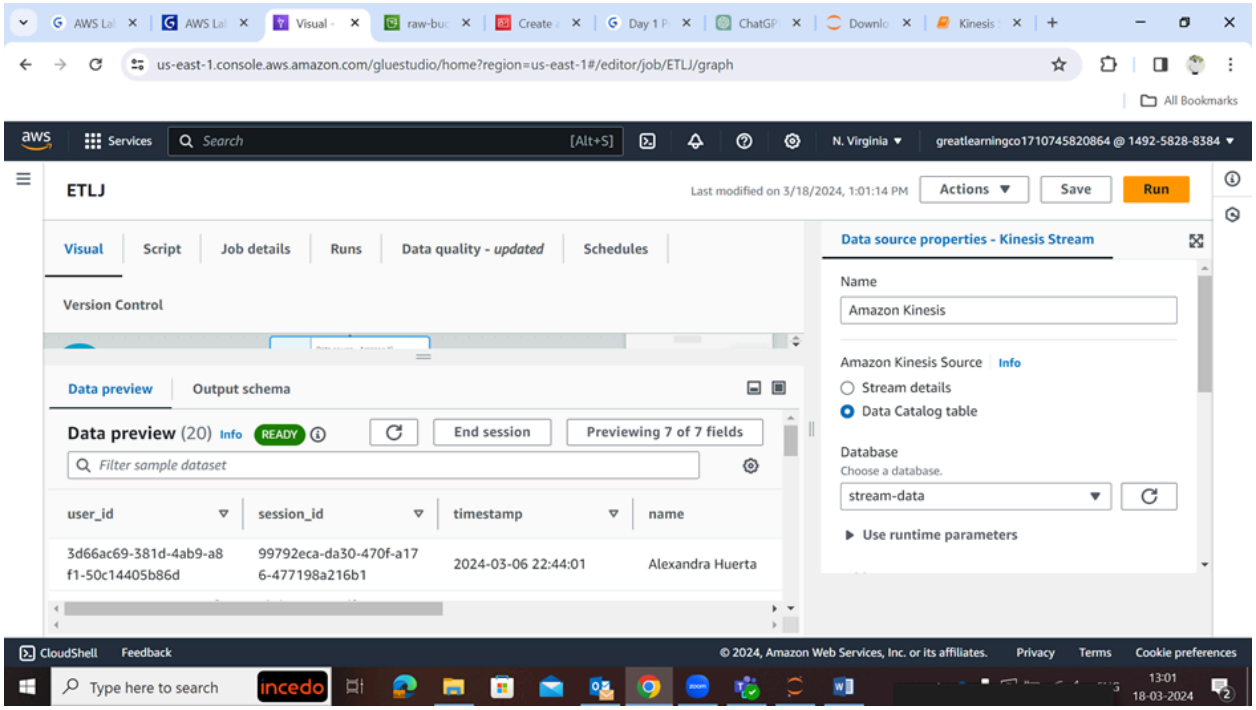
API-steam:



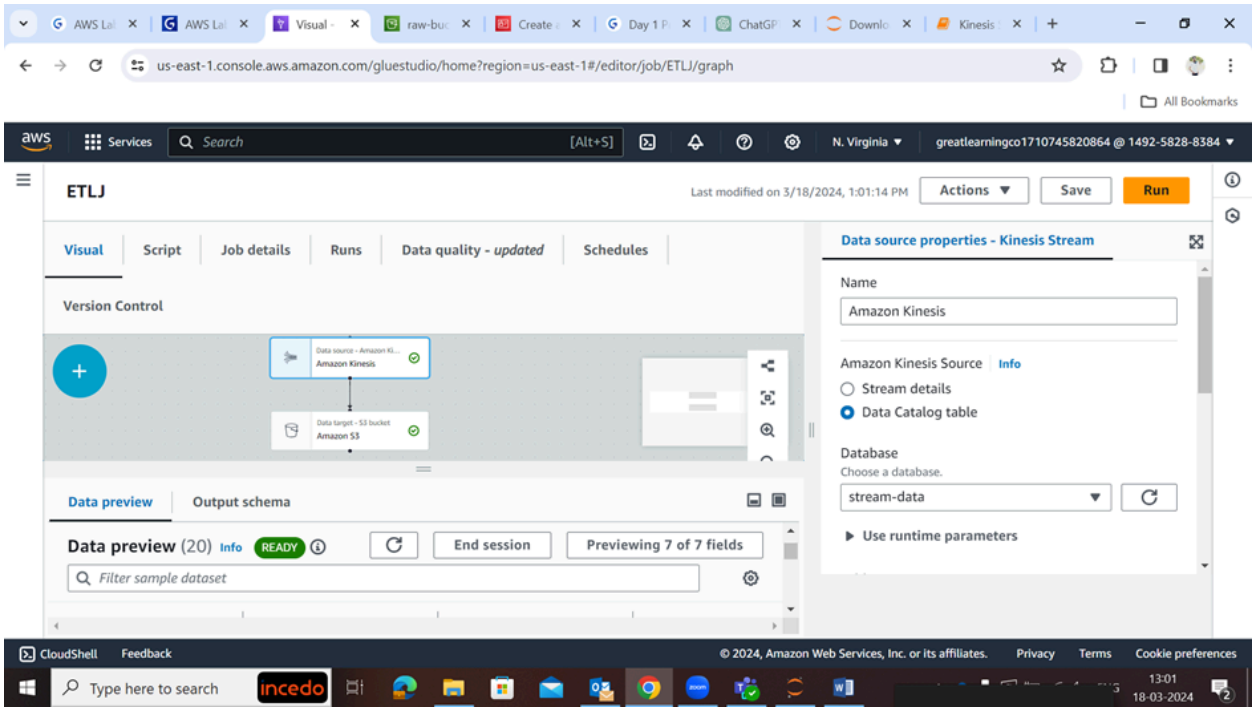
Data is put in stream:



Data preview:



Data is put in bucket as CSV:



Extract Transformation and Load

AWS Glue for ETL

Transforms for all datasets:

1. Source is S3 bucket.
2. Custom Transform: This is performed to clean the abnormalities in the date columns of the dataset. Some dates were of format dd-mm-yyyy while some were of format mm/dd/yyyy. All of these were replaced by the standard mm/dd/yyyy format. Also the dates of the format mm-dd-yy were like 09-06-2021 extra zeroes have been removed: 9/6/2021.

Date columns before:

A	B	C	D	E	F	G	H	I	J	K	L	M
customerN	dateofbirth	citizenship	currCount	employme	incomeInT	marketSeg	maritalSta	stateCode	city	country	accountopeningdate	gender
b90ec527-	11/27/1995	US	USD	worker	$200685.8	standard	Married	FL	Zephyrhills	United Sta	09-06-2021	Male
f8162ec9-	05-03-2002	US	USD	self-emplo	$121102.5	basic	Married	CT	Waterbury	United Sta	1/30/2021	Female
8c50fecc-4	7/21/1994	US	USD	employee	$234625.6	standard	Widowed	NY	Staten Isla	United Sta	6/23/2021	Female
5afb4208-	3/14/1996	US	USD	employee	$152978.8	HNI	Divorced	LA	Baton Rou	United Sta	05-04-2021	Male
3fed47d1-	08-02-1995	US	USD	worker	$168933.2	standard	Widowed	FL	Miami	United Sta	08-06-2021	Female
8e8ced95-	2/16/1999	US	USD	self-emplo	$214317.7	standard	Separated	FL	Panama Ci	United Sta	1/20/2021	Male
b0319a45-	6/28/1993	US	USD	worker	$159816.26		Separated	FL	Tallahasse	United Sta	03-07-2021	Male
2bdeb405-	12/18/1991	US	USD	employee	$342441.2	basic	Separated	OH	Akron	United Sta	1/18/2021	Male
78ebd95c-	10/16/1993	US	USD	self-emplo	$111660.5	basic	Widowed	MO	Kansas Cit	United Sta	6/20/2021	Female
b7fac2b9-4	06-04-1999	US	USD	worker	$385312.7	standard	Separated	OK	Oklahoma	United Sta	3/22/2021	Female
9b508fe5-	2/18/1995	US	USD	employee	$293493.5	standard	Divorced	AZ	Glendale	United Sta	3/30/2021	Male

Date columns after:

A	B	C	D	E	F	G	H	I	J	K	L	M
customerNumber	dateofbirth	citizenship	currCount	employme	incomeInT	marketSeg	maritalSta	stateCode	city	country	accountopeningdat	gender
b90ec527-d54c-4546-8872	11/27/1995	US	USD	worker	########	standard	Married	FL	Zephyrhill	United Sta	9/6/2021	Male
f8162ec9-2f9d-4412-8c22-	5/3/2002	US	USD	self-emplo	########	basic	Married	CT	Waterbury	United Sta	1/30/2021	Female
8c50fecc-abef-43f9-bfe1-8	7/21/1994	US	USD	employee	########	standard	Widowed	NY	Staten Isla	United Sta	6/23/2021	Female
5afb4208-19a0-4ef0-a12b-	3/14/1996	US	USD	employee	########	HNI	Divorced	LA	Baton Rou	United Sta	5/4/2021	Male
3fed47d1-606c-4fb5-a35d-	8/2/1995	US	USD	worker	########	standard	Widowed	FL	Miami	United Sta	8/6/2021	Female
8e8ced95-ddc1-4fb1-b3ae	2/16/1999	US	USD	self-emplo	########	standard	Separated	FL	Panama Ci	United Sta	1/20/2021	Male
b0319a45-4e62-4049-a33c	6/28/1993	US	USD	worker	########		Separated	FL	Tallahasse	United Sta	3/7/2021	Male
2bdeb405-0a05-4f65-90f6	12/18/1991	US	USD	employee	########	basic	Separated	OH	Akron	United Sta	1/18/2021	Male
78ebd95c-8975-4bc8-a78f	10/16/1993	US	USD	self-emplo	########	basic	Widowed	MO	Kansas Cit	United Sta	6/20/2021	Female
b7fac2b9-a461-48db-9129	6/4/1999	US	USD	worker	########	standard	Separated	OK	Oklahoma	United Sta	3/22/2021	Female
9b508fe5-65aa-43c7-90c3-	2/18/1995	US	USD	employee	########	standard	Divorced	AZ	Glendale	United Sta	3/30/2021	Male

3. Timestamp transform: This transform is performed on the clickstream dataset where the timestamp column has the data in epoch format of Python(seconds). This transform converts that data into timestamp format.

Timestamp column before:

Timestamp column after:

clean_clickstream - Notepad

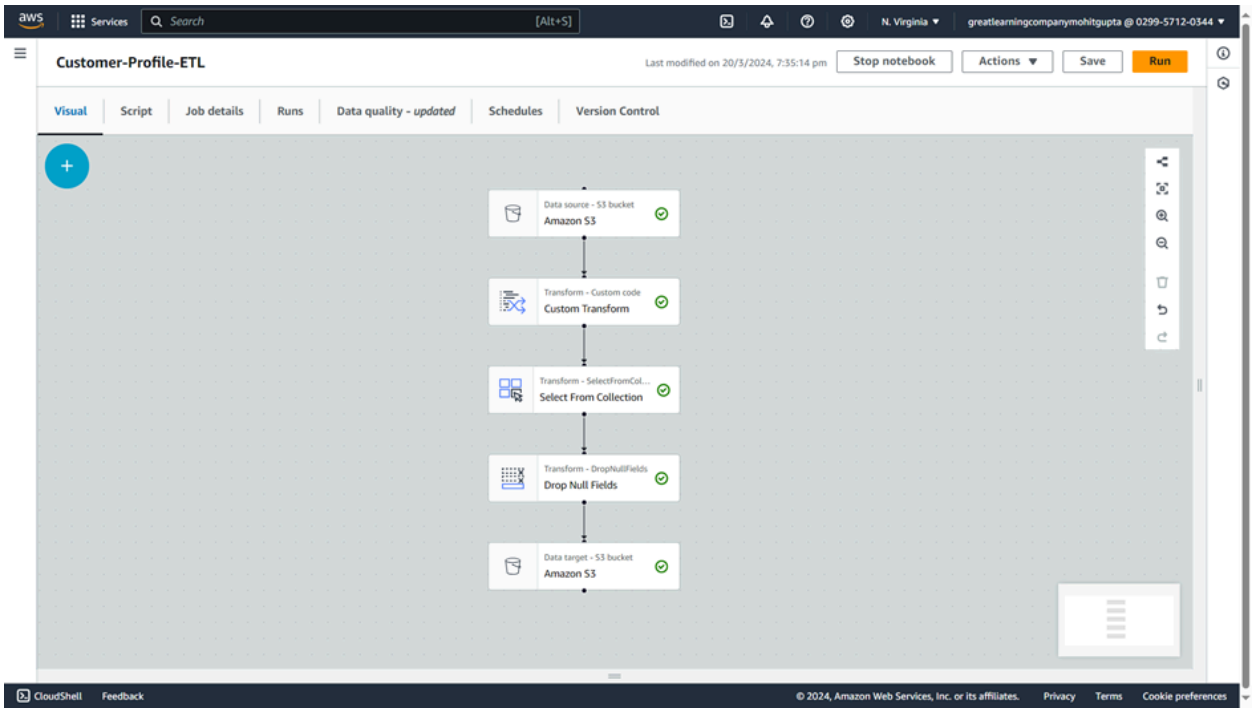
File Edit Format View Help

```
timestamp,session_id,user_id,page_url,referrer,action,product_id,category,price
"2024-03-20 14:59:13.0","e14245d6-e977-41ec-8f8f-fc9d7f2aa5c3","1eda003a-80b0-4829-a2a7-40a2f43cf561","https://d
"2024-03-20 14:59:13.0","a40e4fa0-6928-4c8f-80ea-75800f9f2ed7","52043015-e3c4-43d9-bf18-331538d8fabd","https://c
"2024-03-20 14:59:13.0","8f659e91-d970-4200-b143-c51dd49d8d19","5c397e4c-f844-4e2f-87a3-48d06c570671","http://mo
"2024-03-20 14:59:13.0","8632ff43-651d-434f-bd6a-492cba797985","2551ed4b-c1fb-459b-9187-016eeecaab07","https://n
"2024-03-20 14:59:13.0","f9d0e36e-35ff-4fc5-a086-eccef83128af","08ea00df-892a-47bb-84cd-3b24dfacf86e","http://va
"2024-03-20 14:59:13.0","f7d1080a-42b0-44e1-9ebc-b6f4dc58c6c1","37bada43-57ed-4048-8b3d-cc1e74b050b1","https://g
"2024-03-20 14:59:13.0","01abcc1f-639b-47a3-95b1-33d6570a0f6d","530ccb26-c218-4795-941e-dcf5634f8f0c","https://m
"2024-03-20 14:59:13.0","5cb0cf03-5d8f-4d47-b67a-4ad930369c08","657f161b-f629-4ca7-b3df-1865738222fc","http://ne
"2024-03-20 14:59:13.0","ddad5f05-b791-42ad-add9-88a50a8998ad","3c8b14ec-1c2a-4ca0-adb9-0284ee3af0a6","http://bl
"2024-03-20 14:59:13.0","77efe9ff-e9cb-4db2-bfb0-37bcd700b011","49c4b3de-ad1c-4e1f-9ede-3544608dfc65","http://ww
"2024-03-20 14:59:13.0","9ba610ff-39a4-47eb-a77d-29767cad70f1","9f3eddf3-0a1a-4644-b018-a5aeabebf140","https://a
"2024-03-20 14:59:13.0","cc201d10-dc66-4449-b866-14f5b2449b2e","704ca150-0a6c-4812-a41d-92bb6df89ede","https://h
"2024-03-20 14:59:13.0","d6494f1d-2a31-426a-ab22-ad568d6d42af","8371b560-7978-45b0-90ae-4957cd02a1f6","https://r
"2024-03-20 14:59:13.0","a1848a70-efe5-4c84-bd7e-fbe7aaa2e88e","82668632-133d-4c27-966f-71bb4938fd1f","https://n
"2024-03-20 14:59:13.0","7633d4c6-22eb-4763-a5bc-a8afabead891","6783ac2b-8993-4e02-a691-f40739ad392f","https://w
"2024-03-20 14:59:13.0","14e56a21-b925-4773-b424-1221e25f4a0d","c78c8bac-54d1-4293-94c4-8d5c3e65afa6","http://ww
"2024-03-20 14:59:13.0","ad8bdec4-8ec0-45fe-9a76-0c66ba5a5148","fc2b5871-b414-46b9-837e-18e40d2aac04","http://ww
"2024-03-20 14:59:13.0","7ccecfb7-120e-4d13-8ccc-12dd19ea8e38","ab3415f5-28a5-453b-bb53-98b1a288f331","https://w
"2024-03-20 14:59:13.0","6cd37081-47d4-4ecd-abc0-b72e40d2c080","4eea8538-12b0-4c0b-8085-acb413ebec11","http://el
"2024-03-20 14:59:13.0","304ca85f-b141-4a4d-bf7e-ead4a2c09b8b","23a08b5e-7b72-479d-9413-894a16eacfa3","http://ww
"2024-03-20 14:59:13.0","e4ccc8ae-7b1c-464e-8858-958d3036ae90","eb01f07a-1396-4fa8-a3b8-95e189383a1d","http://wa
```

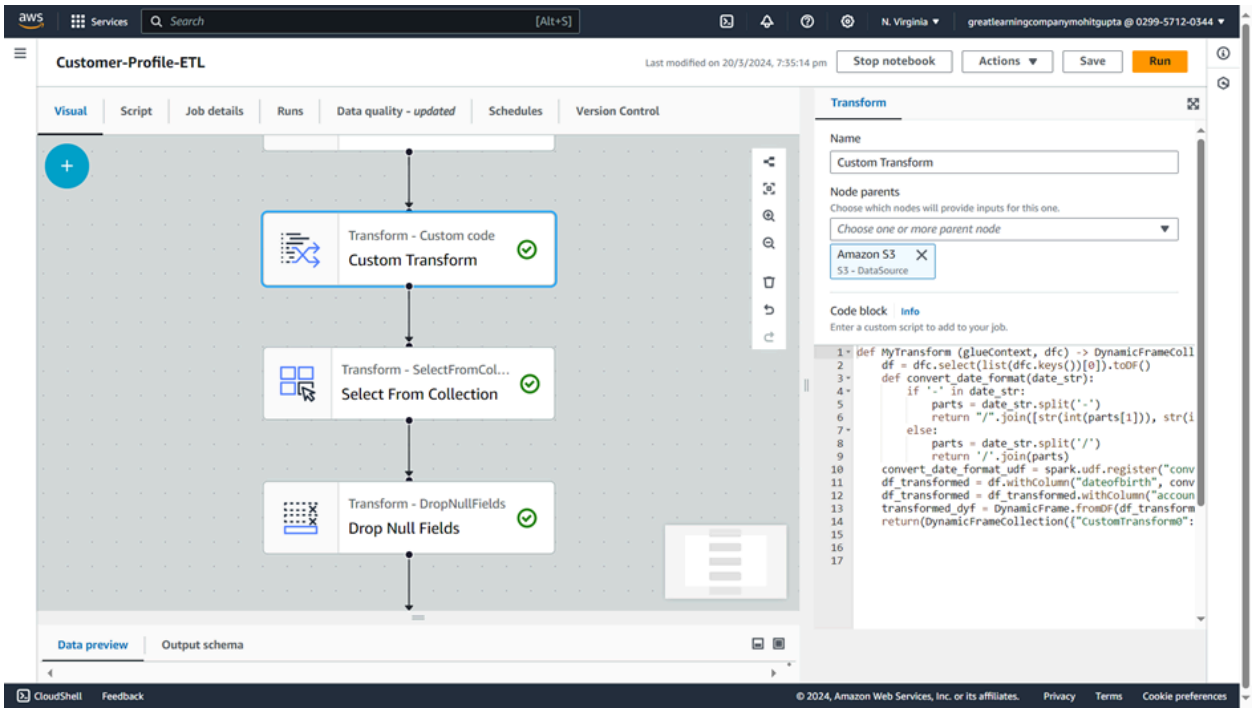
Ln 1, Col 1 | 100% | Unix (LF) | UTF-8

4. Destination is also an S3 bucket.

ETL of Customer Profile data:



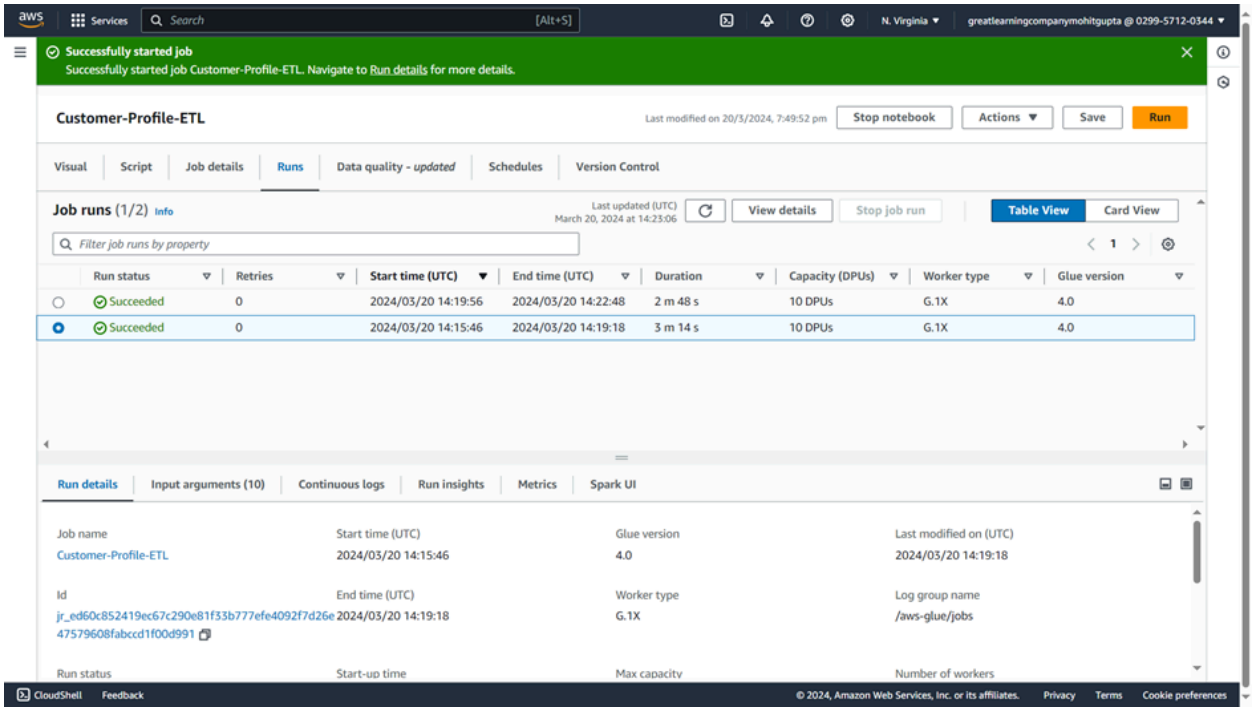
The custom transform:



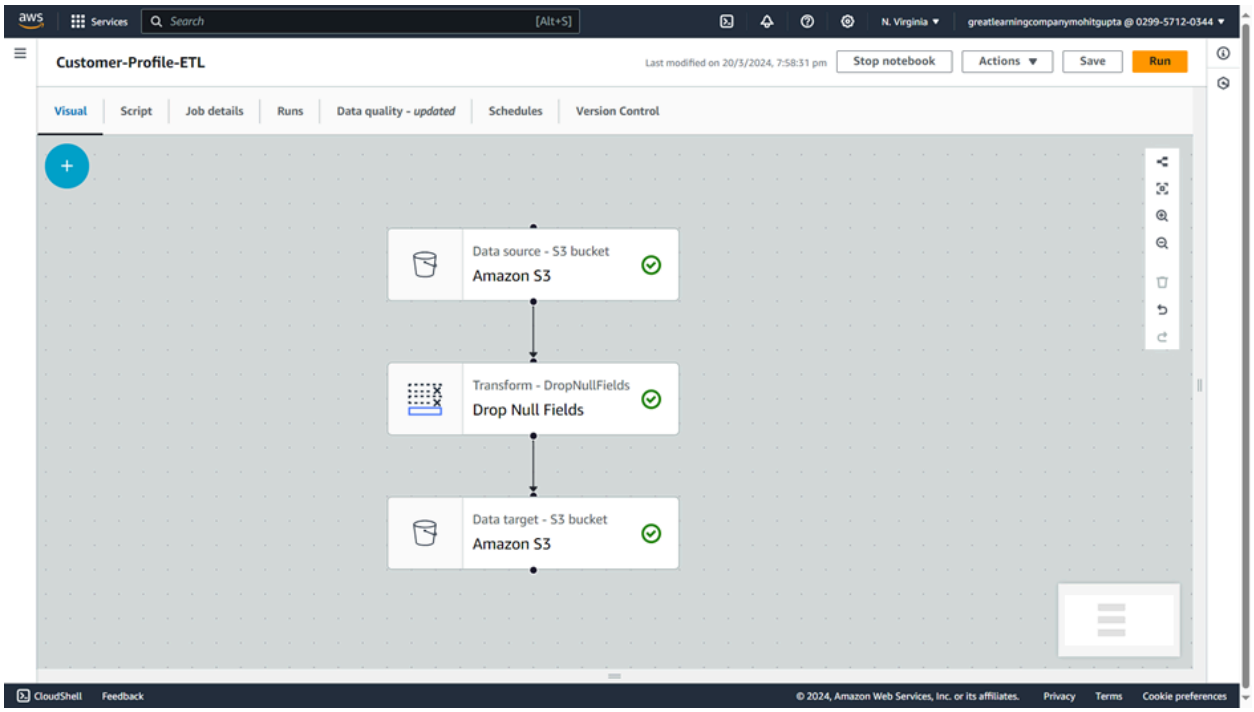
Code:

```
def MyTransform (glueContext, dfc) -> DynamicFrameCollection:
    df = dfc.select(list(dfc.keys())[0]).toDF()
    def convert_date_format(date_str):
        if '-' in date_str:
            parts = date_str.split('-')
            return "/".join([str(int(parts[1])), str(int(parts[0])), parts[2]])
        else:
            parts = date_str.split('/')
            return '/'.join(parts)
    convert_date_format_udf = spark.udf.register("convert_date_format", convert_date_format)
    df_transformed = df.withColumn("dateofbirth", convert_date_format_udf(df["dateofbirth"]))
    df_transformed = df_transformed.withColumn("accountopeningdate",
convert_date_format_udf(df_transformed["accountopeningdate"]))
    transformed_dyf = DynamicFrame.fromDF(df_transformed, glueContext, "transformed_dyf")
    return(DynamicFrameCollection({"CustomTransform0": transformed_dyf}, glueContext))
```

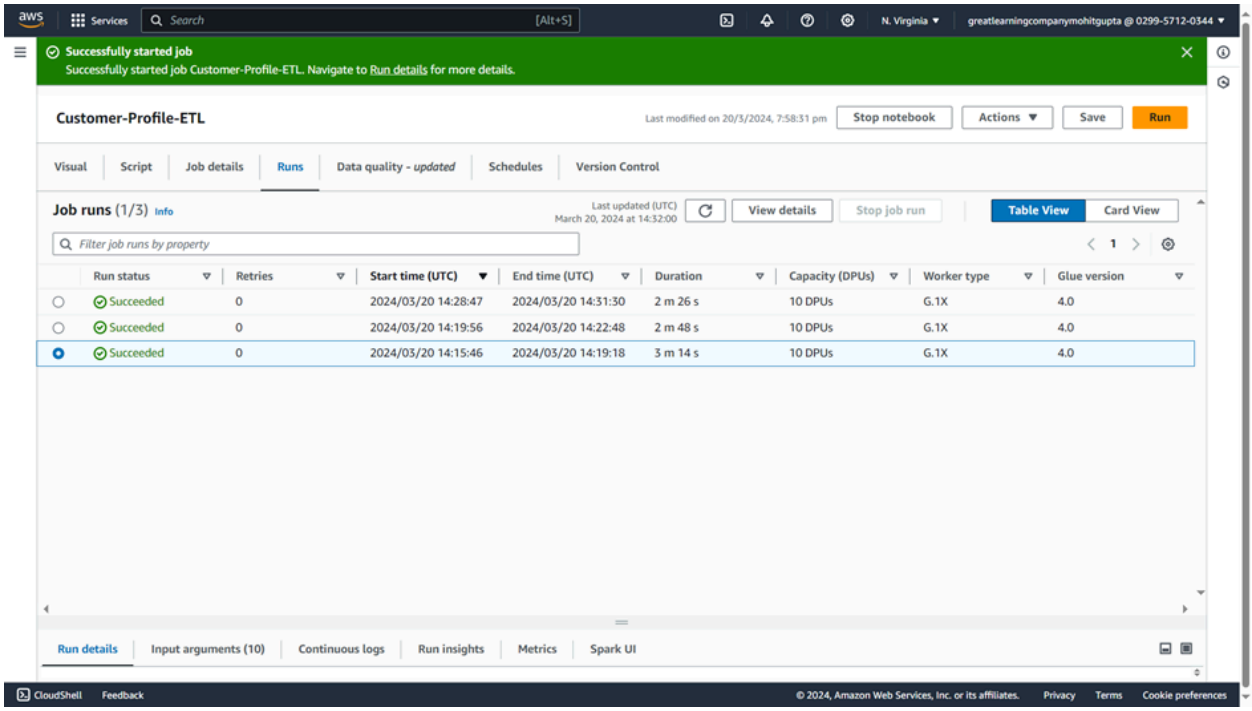
Job run succeeded:



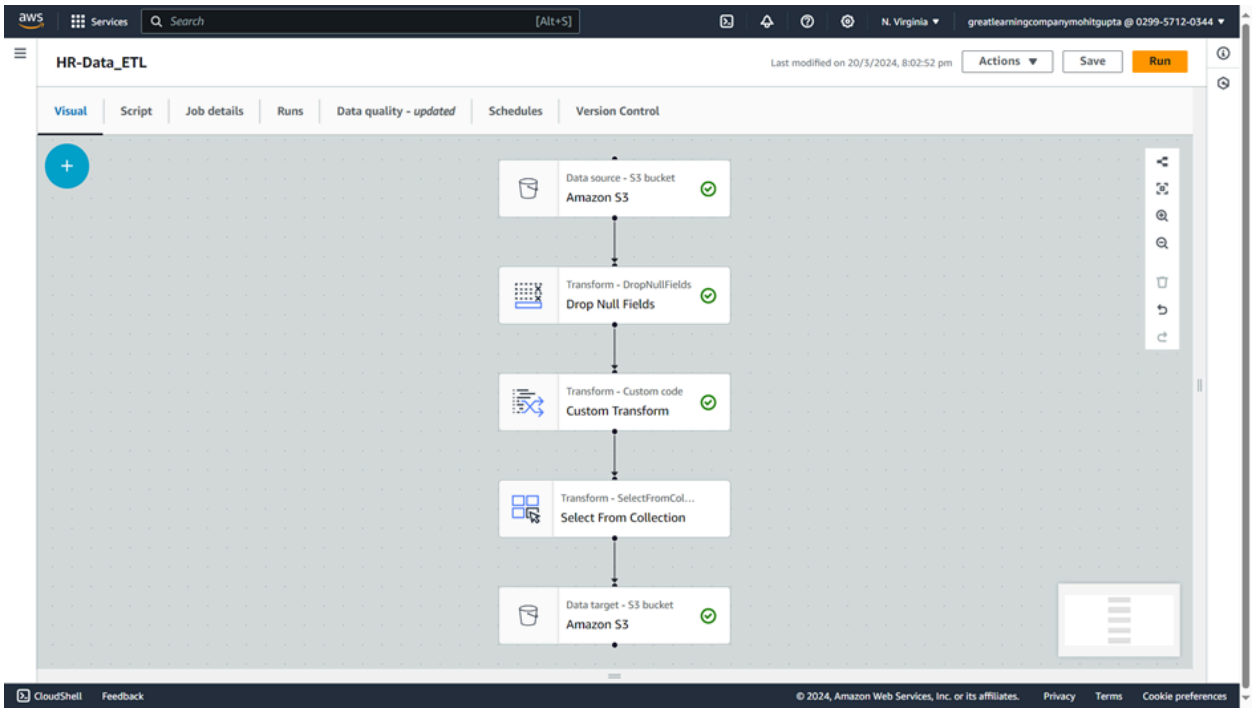
ETL of Customer Transactions data:



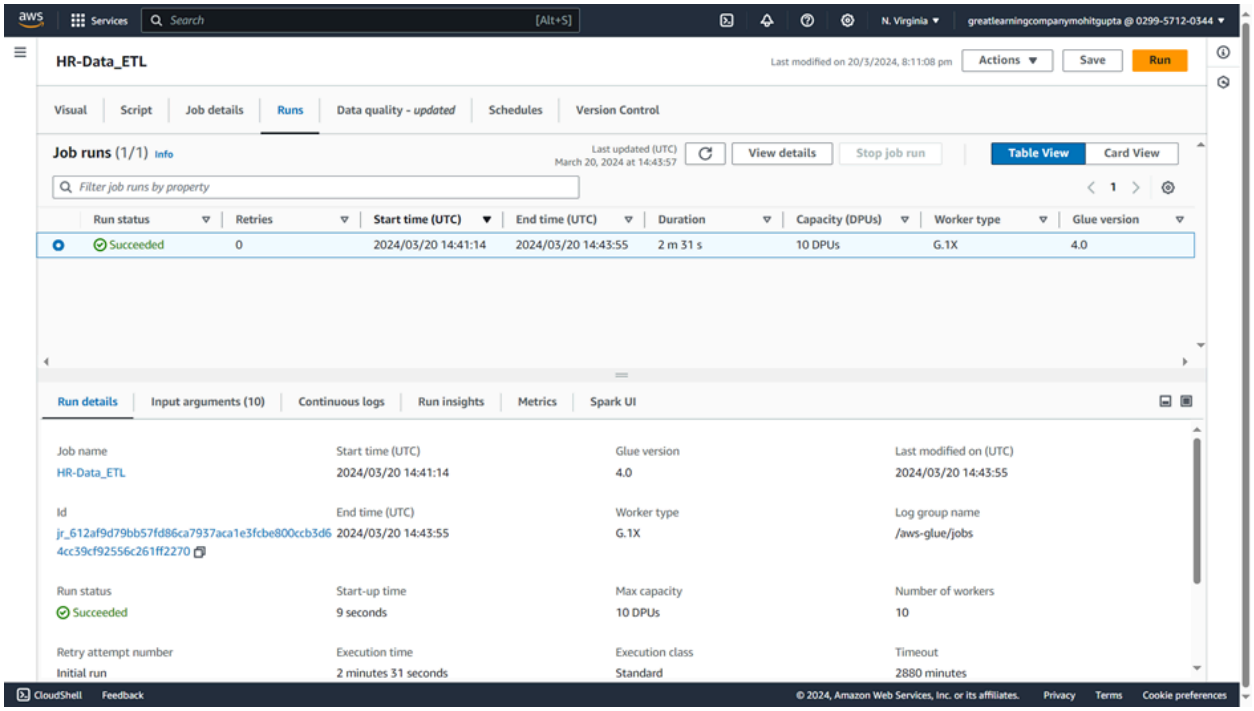
Job run Succeeded:



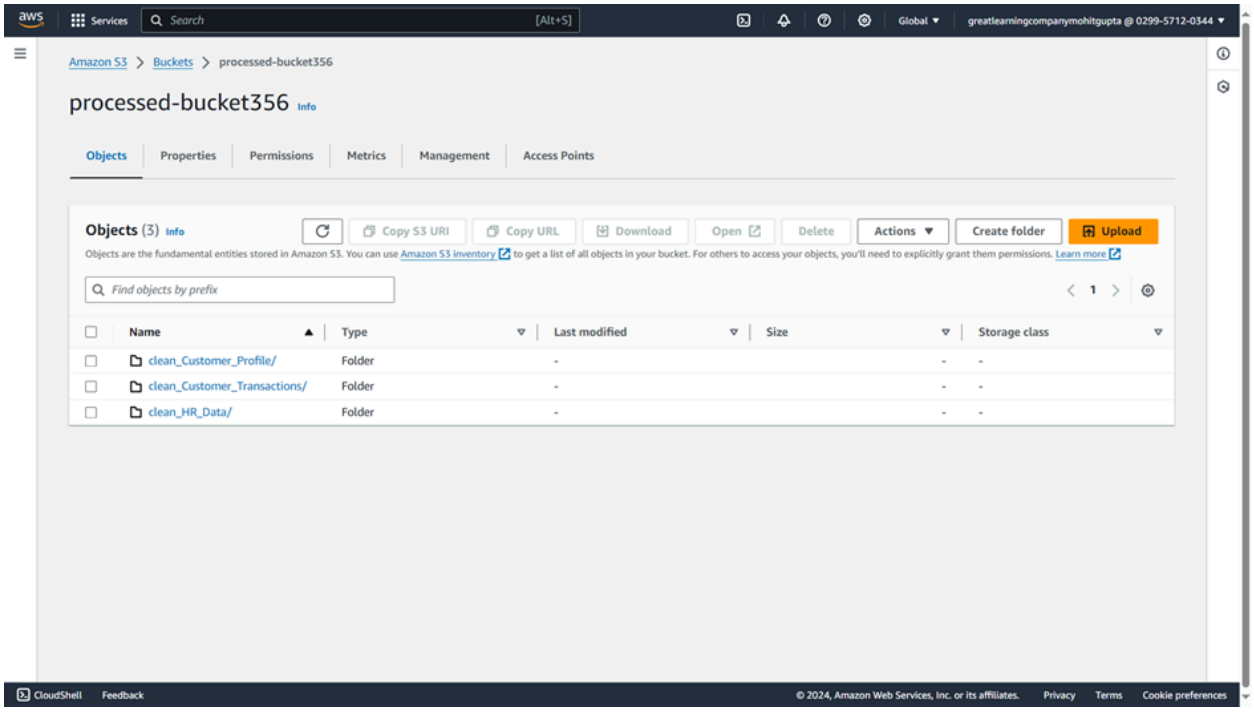
ETL of HR Data:



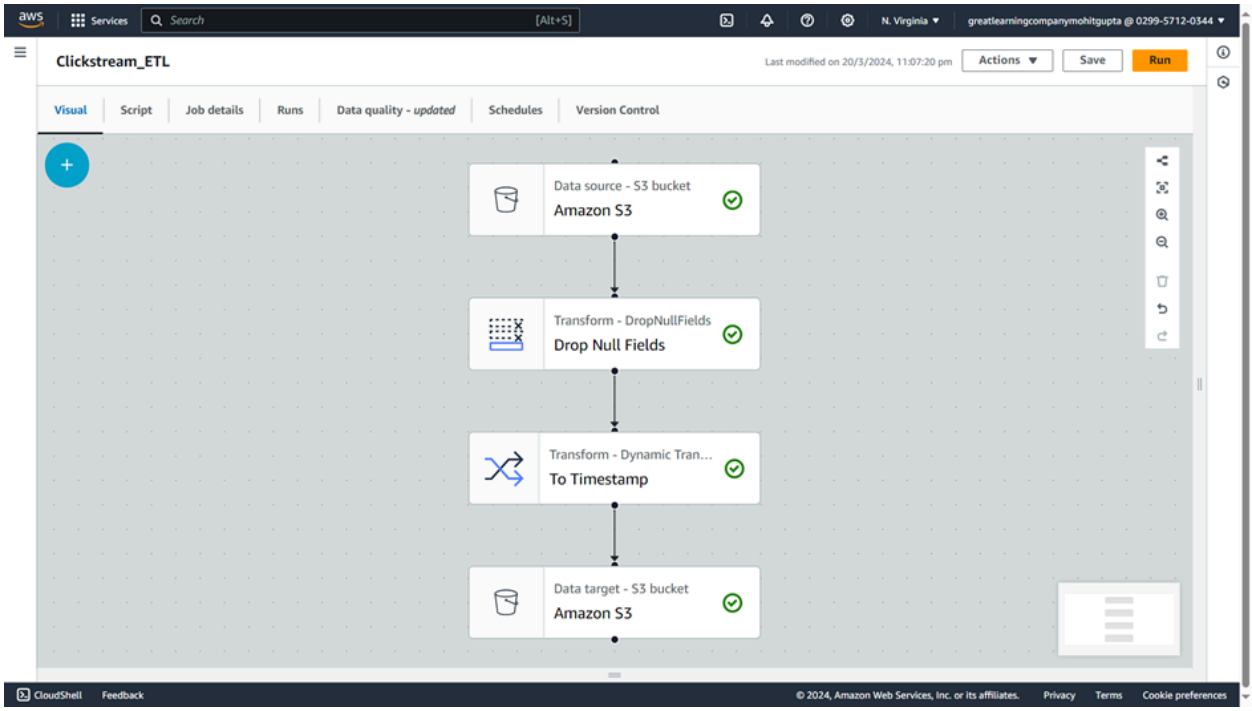
Job run succeeded:



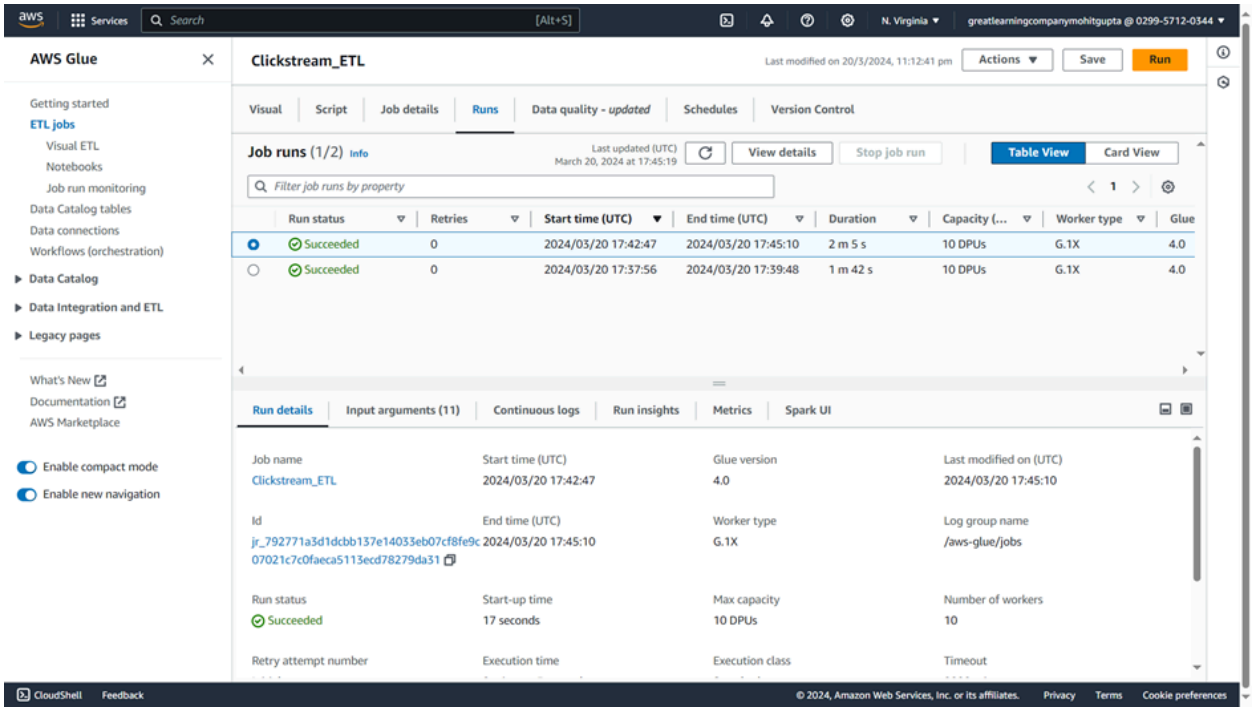
Bucket screenshots:



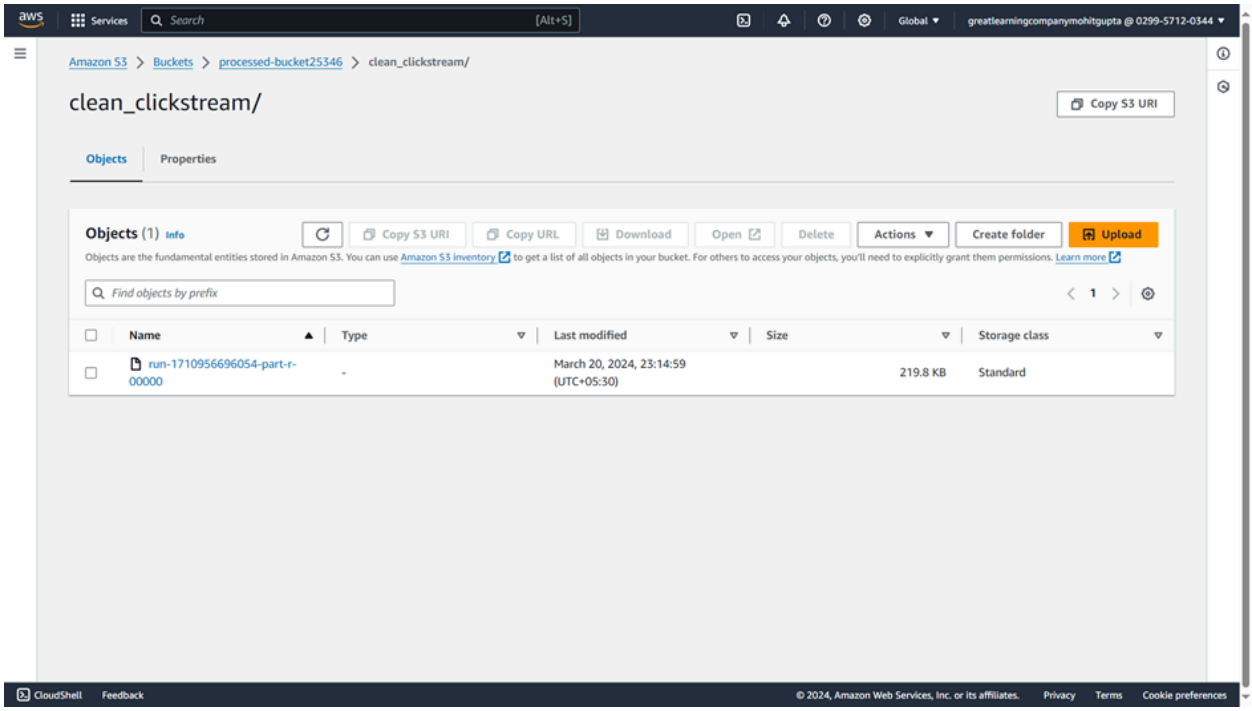
ETL of Clickstream data:



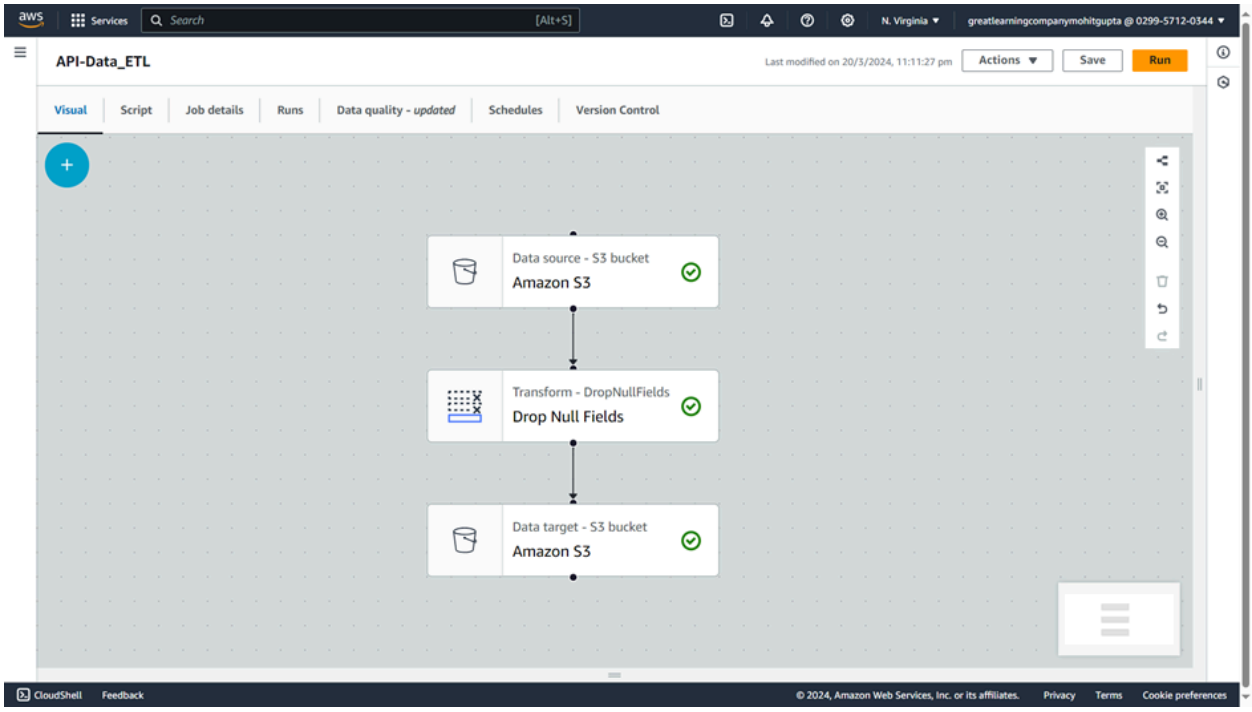
Job run succeeded:



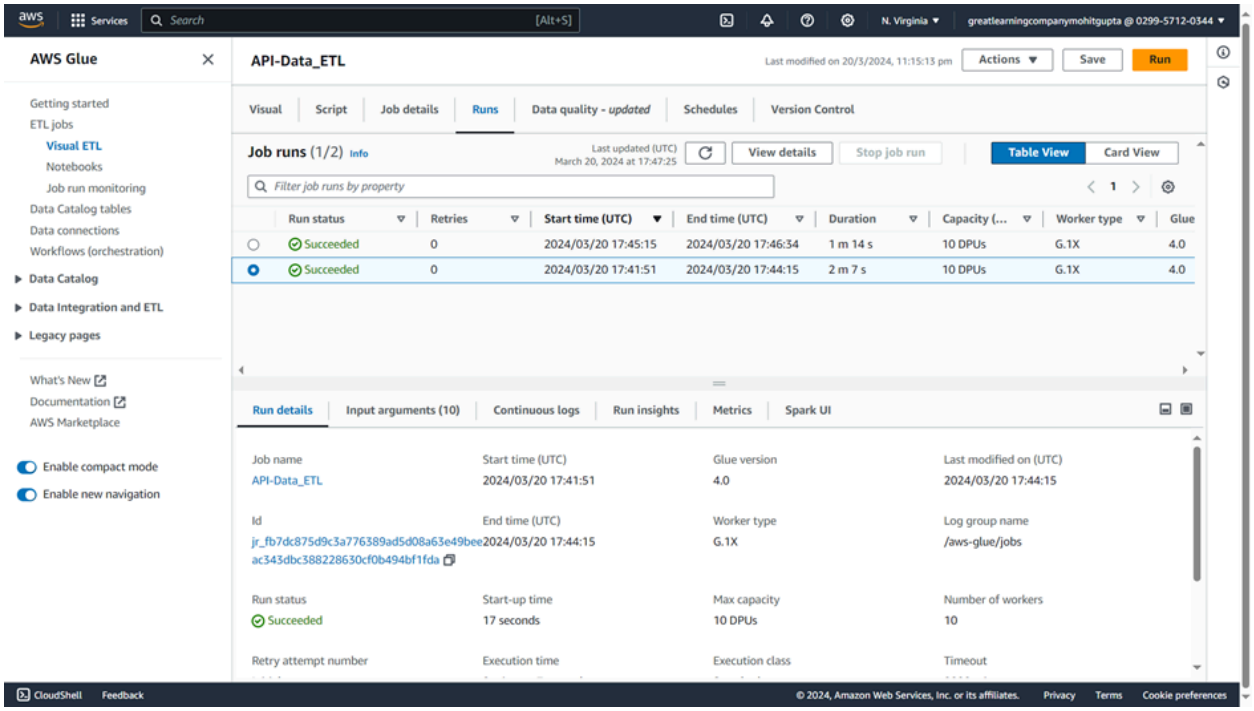
Bucket screenshot:



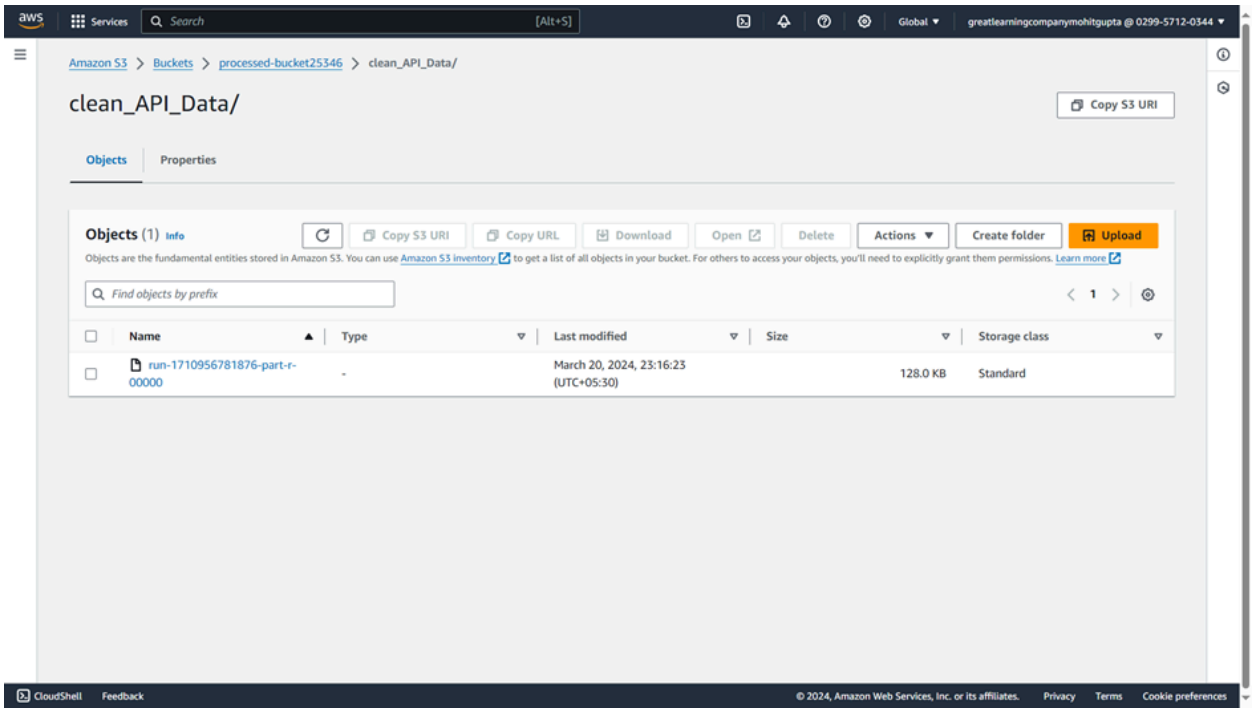
ETL of API Data:



Job run succeeded:

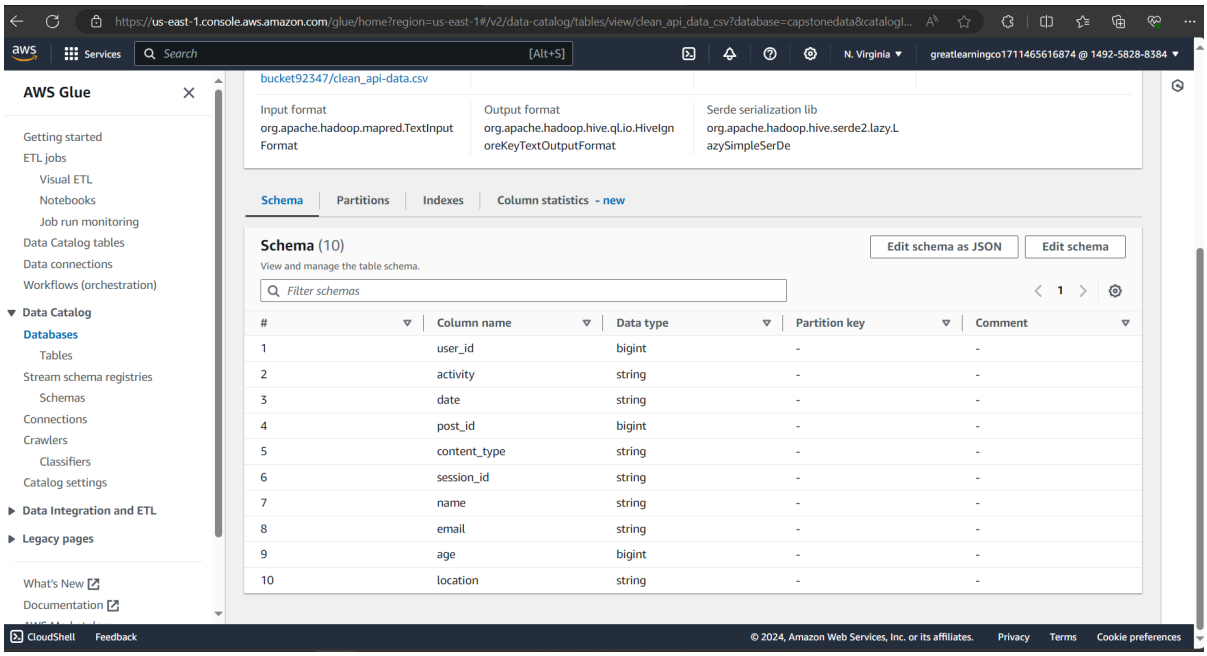


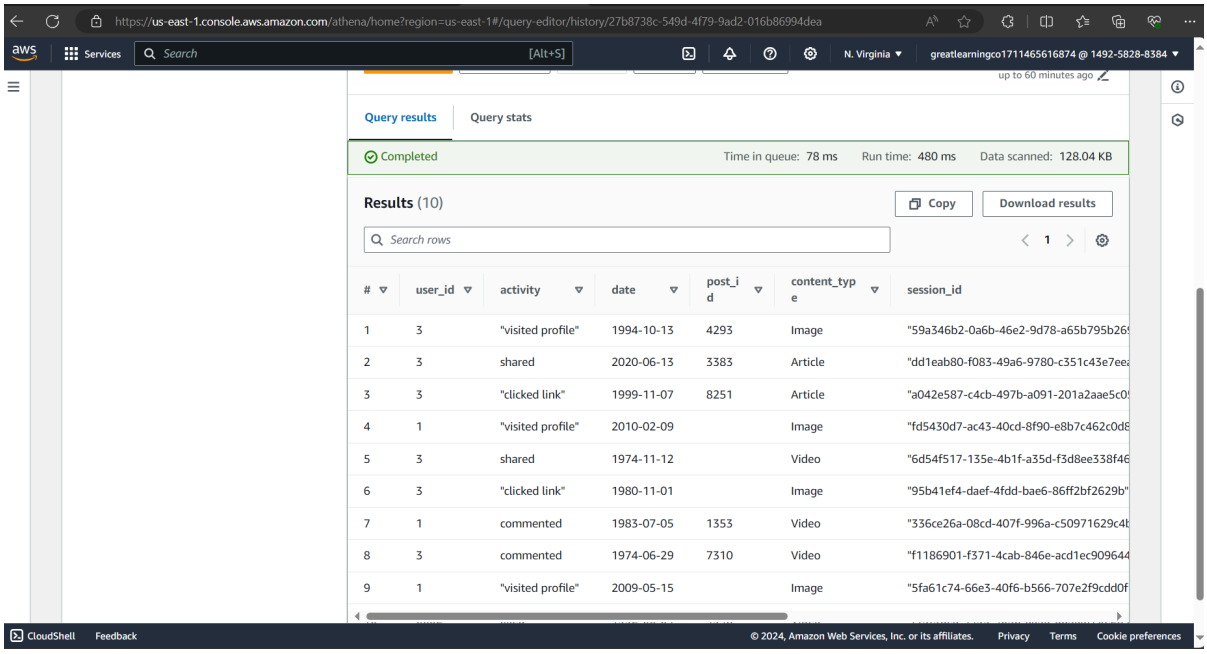
Bucket screenshot:



Load

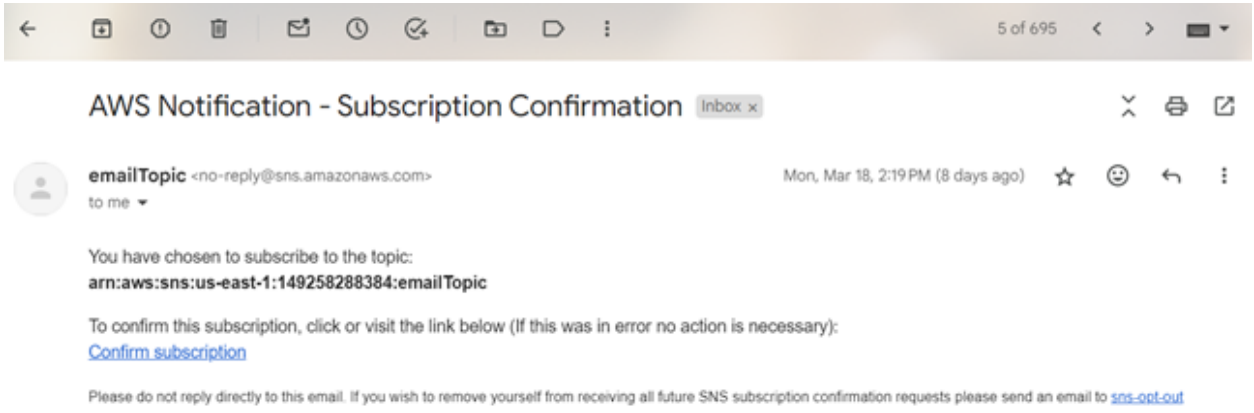
1. Data is loaded into Glue databases and data catalog using glue crawlers from processed S3 buckets.
2. The schema is automatically defined and tables are populated.
3. Now, we can perform sql queries on this data using Athena.





Communication: Sending email to user using SMTP and to admin using SNS

1. Creating SNS topic for admin.
2. Subscribing admin email to the topic so they receive emails.



5 of 695

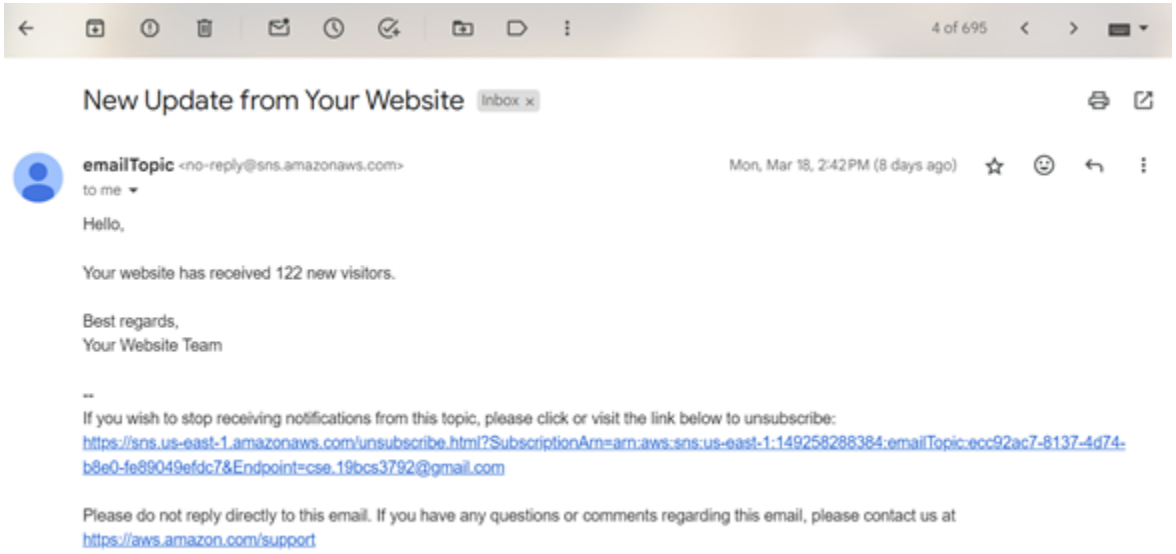
AWS Notification - Subscription Confirmation Inbox ×

emailTopic <no-reply@sns.amazonaws.com> Mon, Mar 18, 2:19 PM (8 days ago)
to me

You have chosen to subscribe to the topic:
arn:aws:sns:us-east-1:149258288384:emailTopic

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
Confirm subscription

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to sns-opt-out



4 of 695

New Update from Your Website Inbox ×

emailTopic <no-reply@sns.amazonaws.com> Mon, Mar 18, 2:42 PM (8 days ago)
to me

Hello,

Your website has received 122 new visitors.

Best regards,
Your Website Team

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:149258288384:emailTopic:ecc92ac7-8137-4d74-b8e0-fe89049efdc7&Endpoint=cse.19bcs3792@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at
https://aws.amazon.com/support

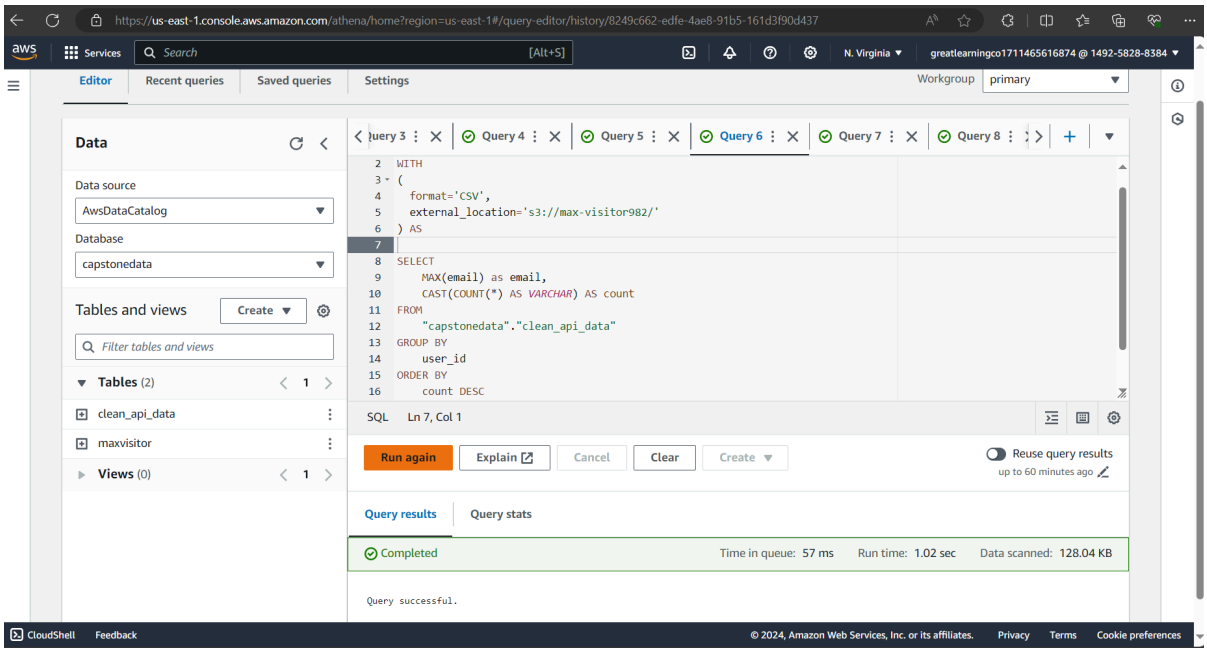
3. Writing SQL qwerty in Athena to output the email ID and the count of visits in a separate S3 bucket (max-visitor982).

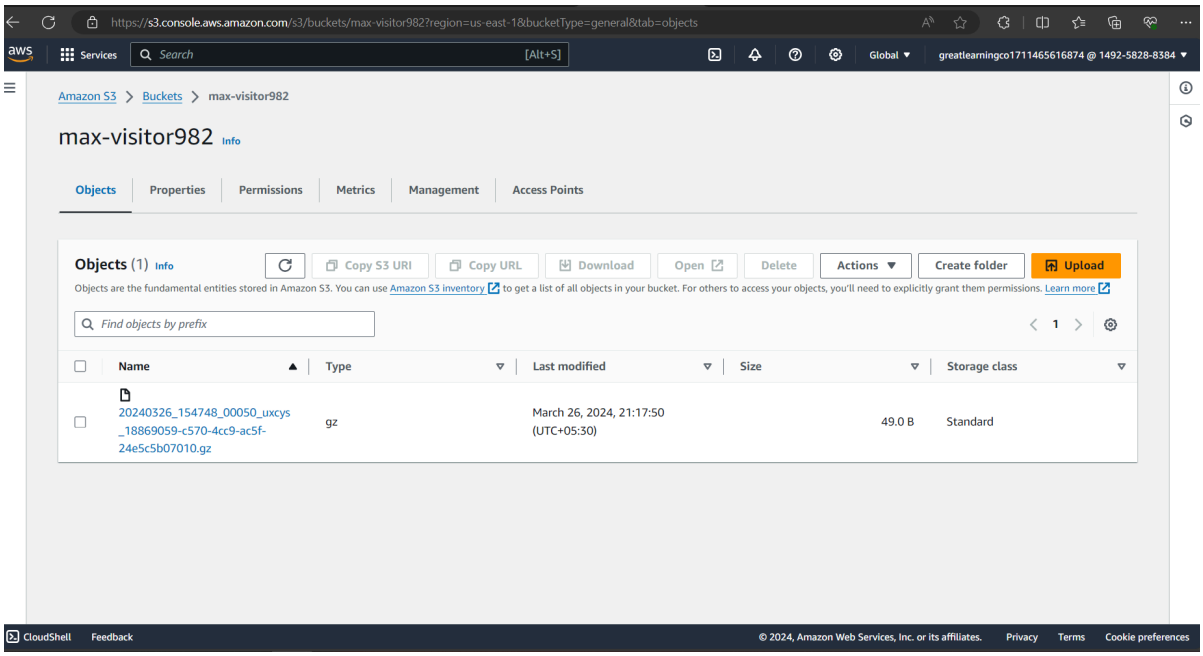
SQL Query:

```
CREATE TABLE "capstonedata"."maxvisitor"
WITH
(
  format='CSV',
  external_location='s3://max-visitor982/'
) AS

SELECT
    MAX(email) as email,
```

```
    CAST(COUNT(*) AS VARCHAR) AS count
FROM
    "capstonedata"."clean_api_data"
GROUP BY
    user_id
ORDER BY
    count DESC
LIMIT 1;
```

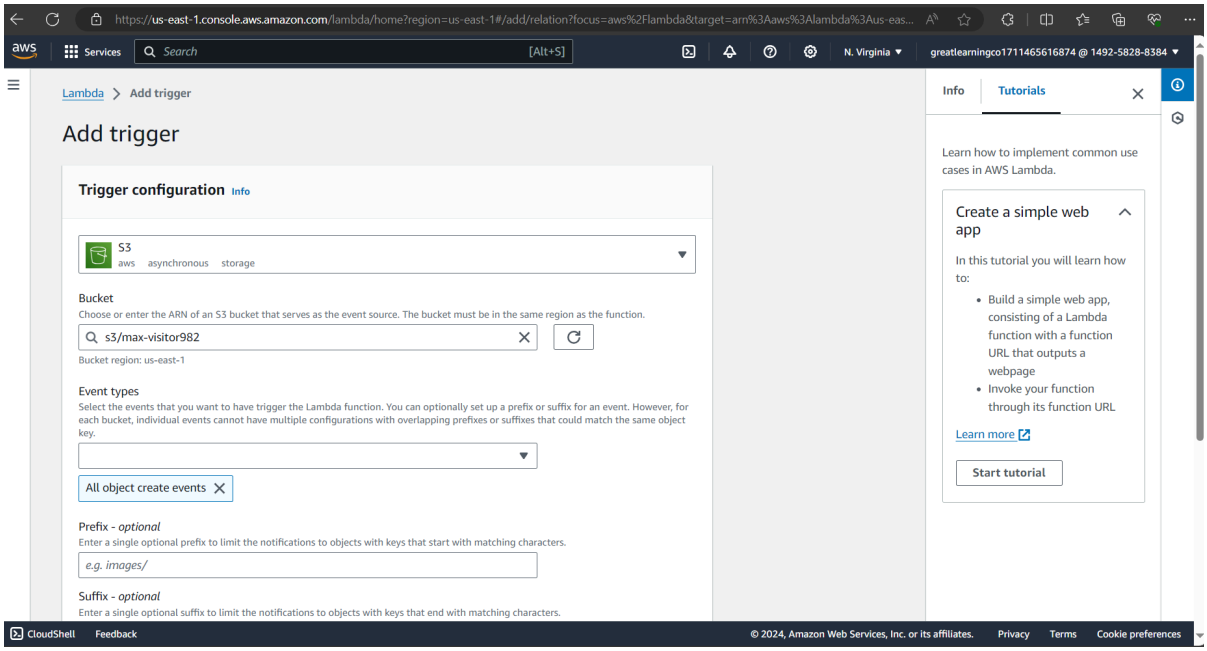
This saves the csv containing email and count in the S3 bucket specified in compressed gz format.

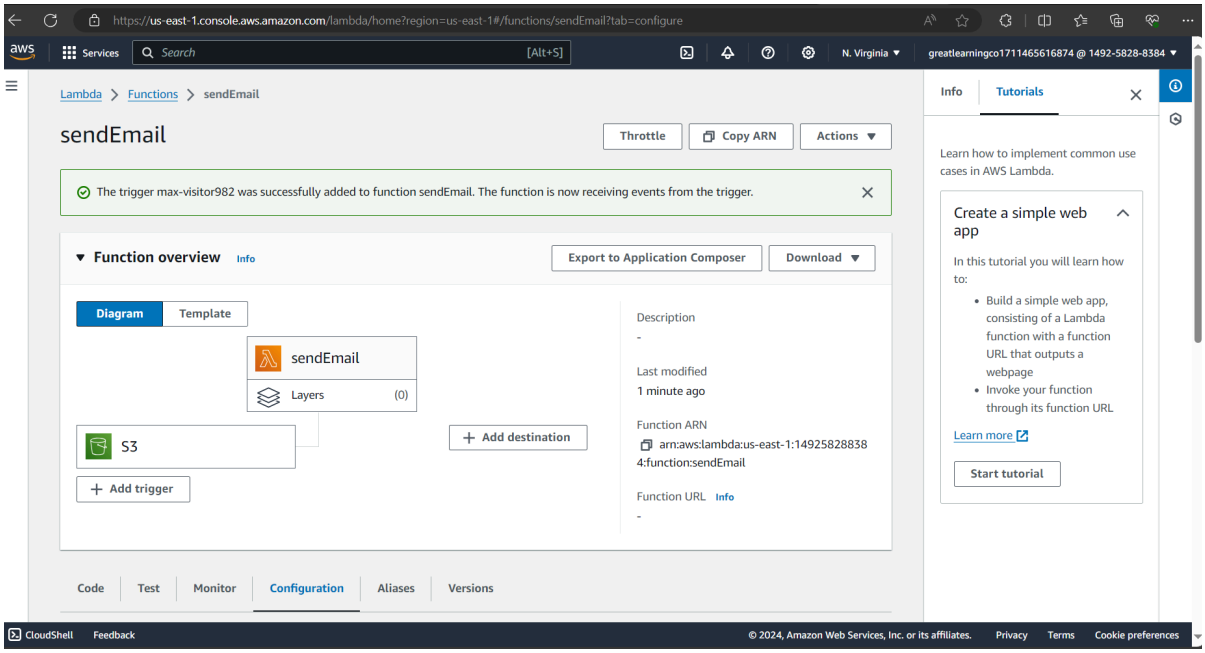


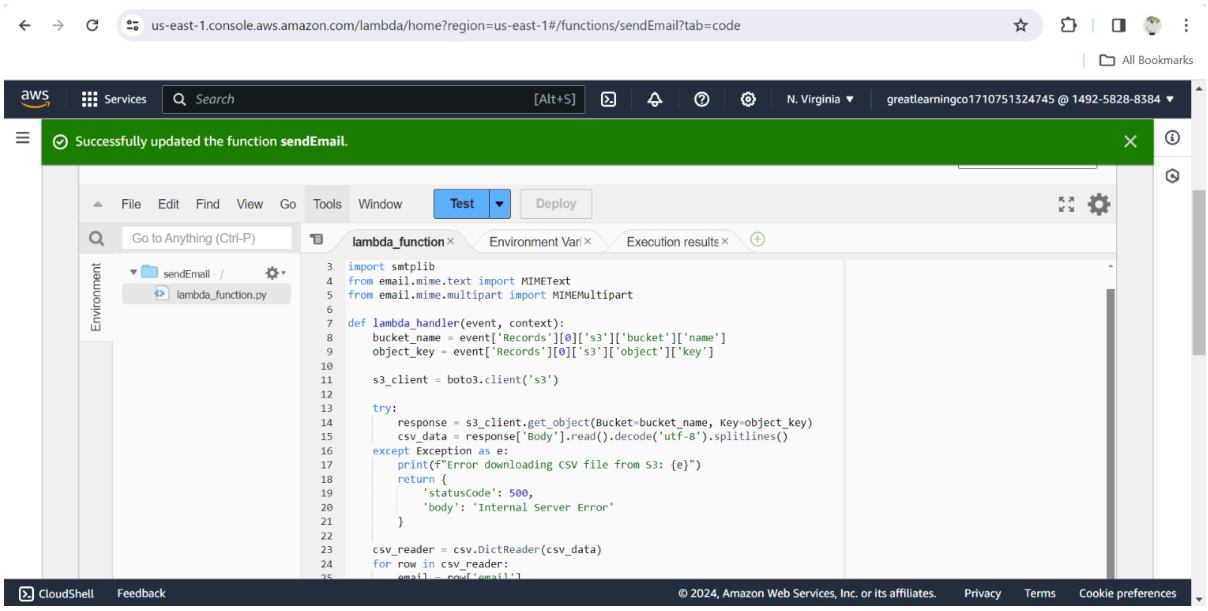


4. Writing lambda function code to give discount coupons to the most frequently visited user using SMTP protocol.
5. The lambda triggers when any changes happen in the S3 bucket created earlier (max-visitor982). It picks up email and number of visits from that S3 bucket where the email of the top visitor is stored after running query on Athena.

Trigger:

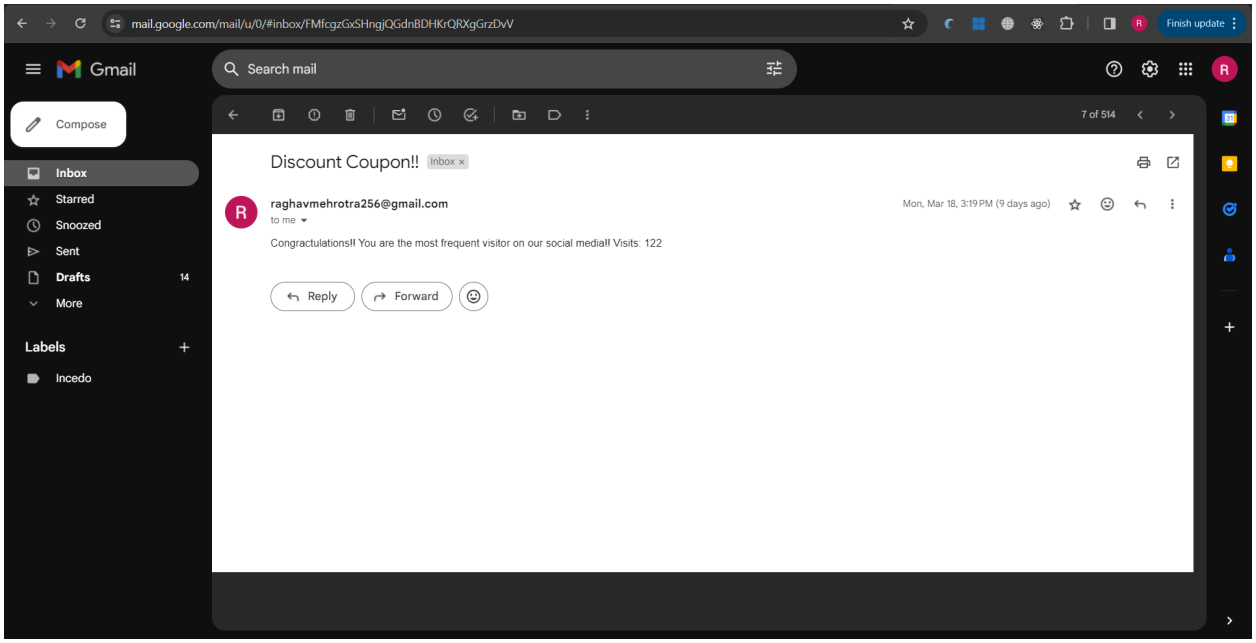






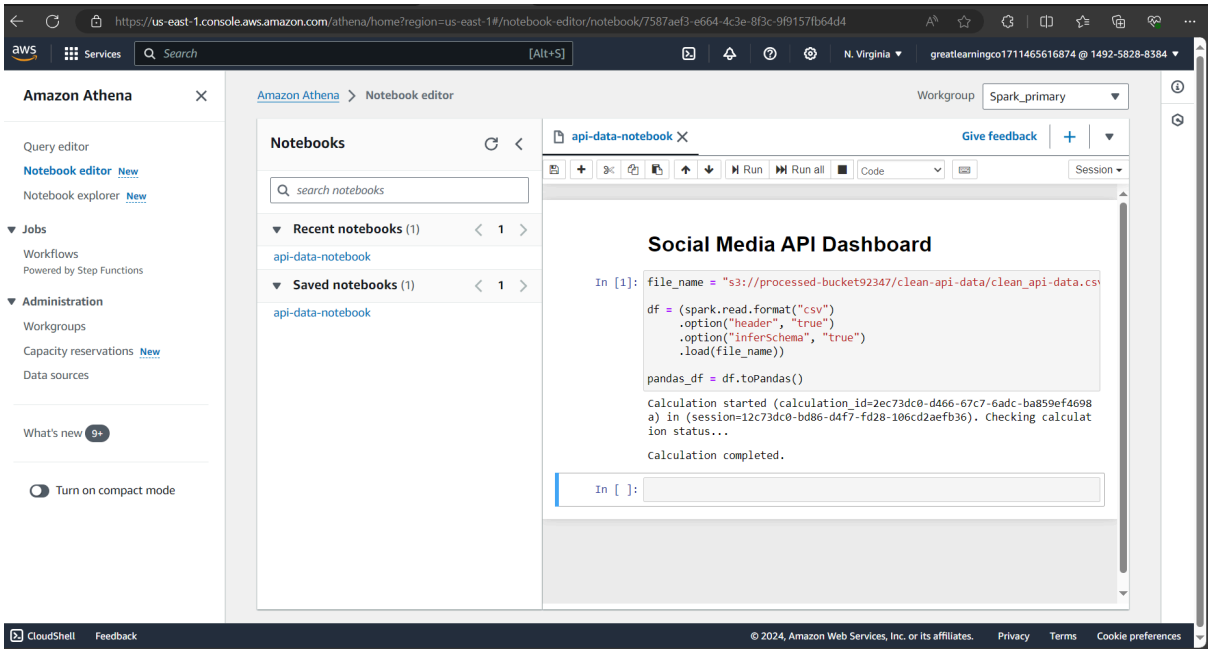

```
import boto3
import csv
import random
import string
import smtplib
import gzip
import io

from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

def lambda_handler(event, context):
	bucket_name = event['Records'][0]['s3']['bucket']['name']
	object_key = event['Records'][0]['s3']['object']['key']
	s3_client = boto3.client('s3')

	try:
		response = s3_client.get_object(Bucket=bucket_name, Key=object_key)
		compressed_data = response['Body'].read()
		with gzip.open(io.BytesIO(compressed_data)) as f:
			csv_data = f.read().decode('utf-8').splitlines()

	except Exception as e:
		print(f"Error downloading CSV file from S3: {e}")import
		return {
			'statusCode': 500,
			'body': 'Internal Server Error'
		}

	csv_reader = csv.DictReader(csv_data)
	for row in csv_reader:
		email = row['email']
```

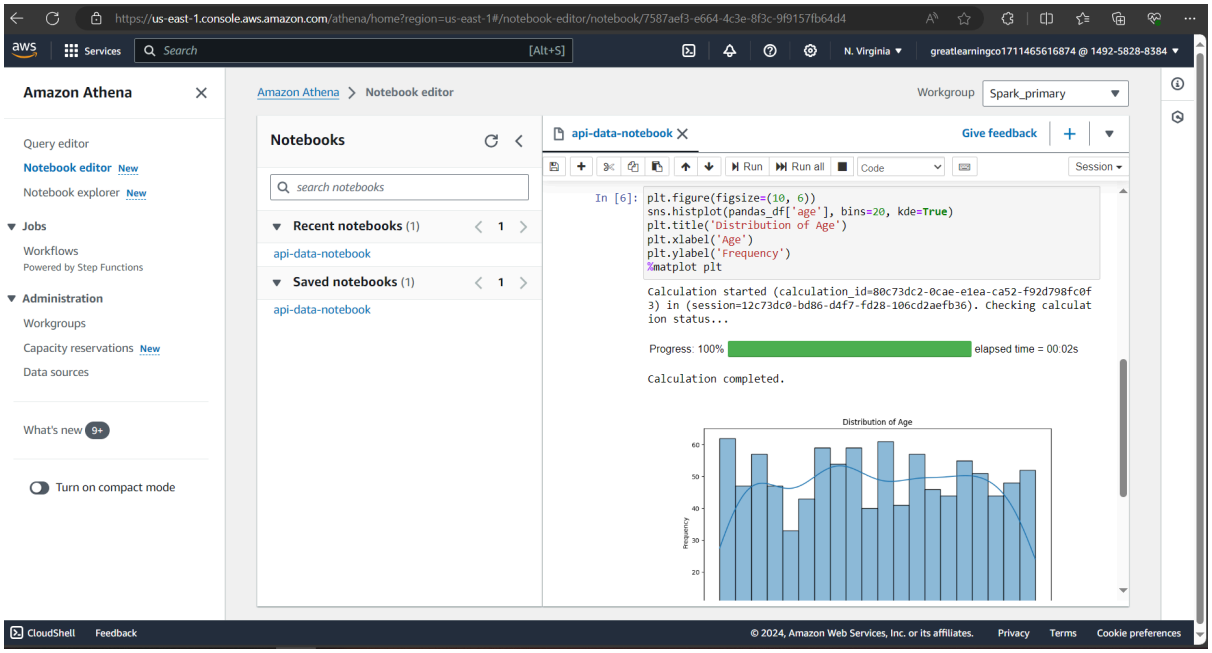
```
            count = row['count']
            send_email(email, count)


def send_email(recipient_email, count):
        # Retrieve S3 bucket and object key from the event
        sender_email = email
        subject = "Discount Coupon!!"
        characters = string.ascii_letters + string.digits
        coupon_code = ''.join(random.choice(characters) for i in range(8))
        message = "Congratulations!! You are the most frequent visitor on our social media!! Visits:
"+count+". Your discount coupon is: "+coupon_code+"."

        msg = MIMEMultipart()
        msg['From'] = sender_email
        msg['To'] = recipient_email
        msg['Subject'] = subject

        msg.attach(MIMEText(message, 'plain'))

        try:
            smtp_server = smtplib.SMTP('smtp.gmail.com', 587)
            smtp_server.starttls()
            smtp_server.login(sender_email, 'dvpojbnfkdotjdxi')
            smtp_server.sendmail(sender_email, recipient_email, msg.as_string())
            smtp_server.quit()
            print(f"Email sent to {recipient_email}")
        except Exception as e:
            print(f"Error sending email to {recipient_email}: {e}")
```



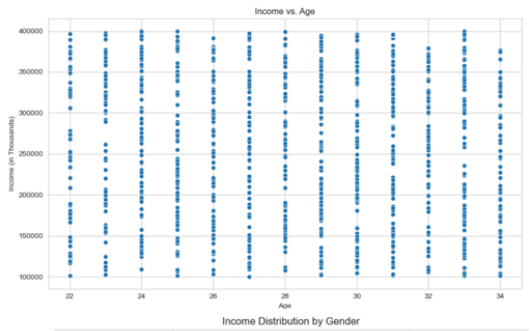
Visualization and Dashboard

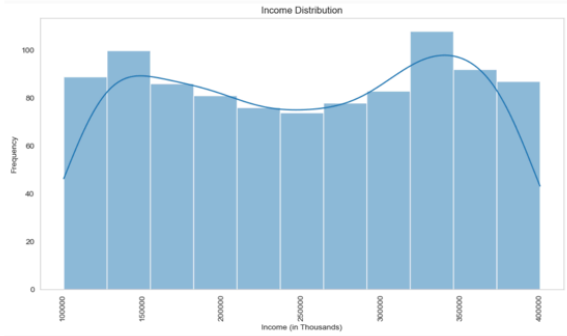
This is accomplished using AWS Athena's embedded notebooks. By loading the cleaned csv into a spark dataframe and then converting it to a pandas dataframe, we can create several interactive graphs using plotly, matplotlib and seaborn libraries.

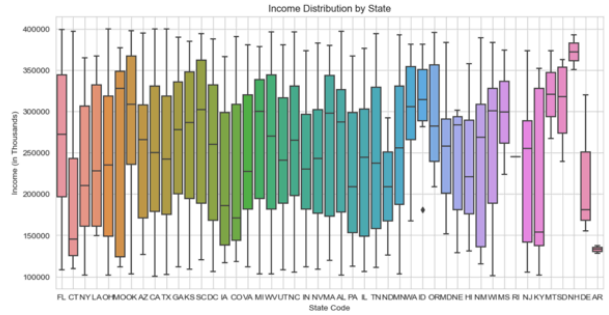




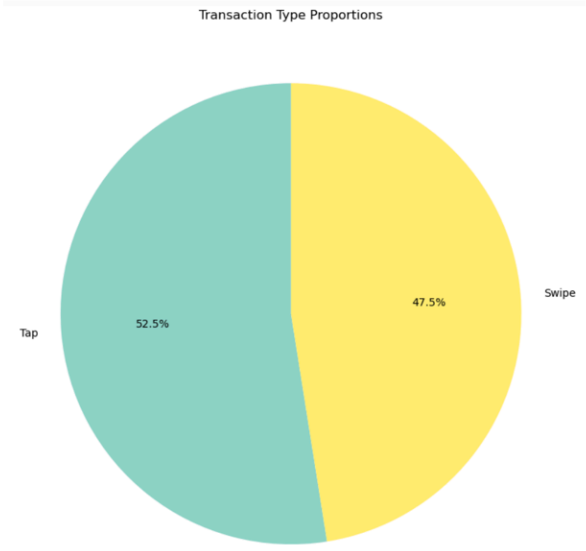
Customer Profile Dashboard

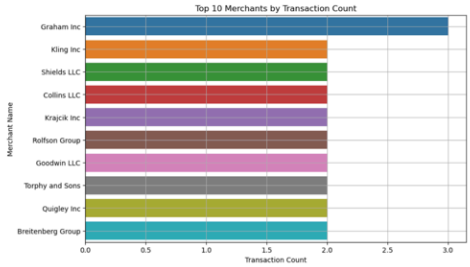


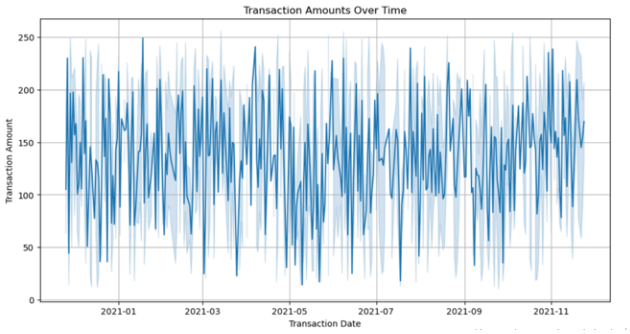




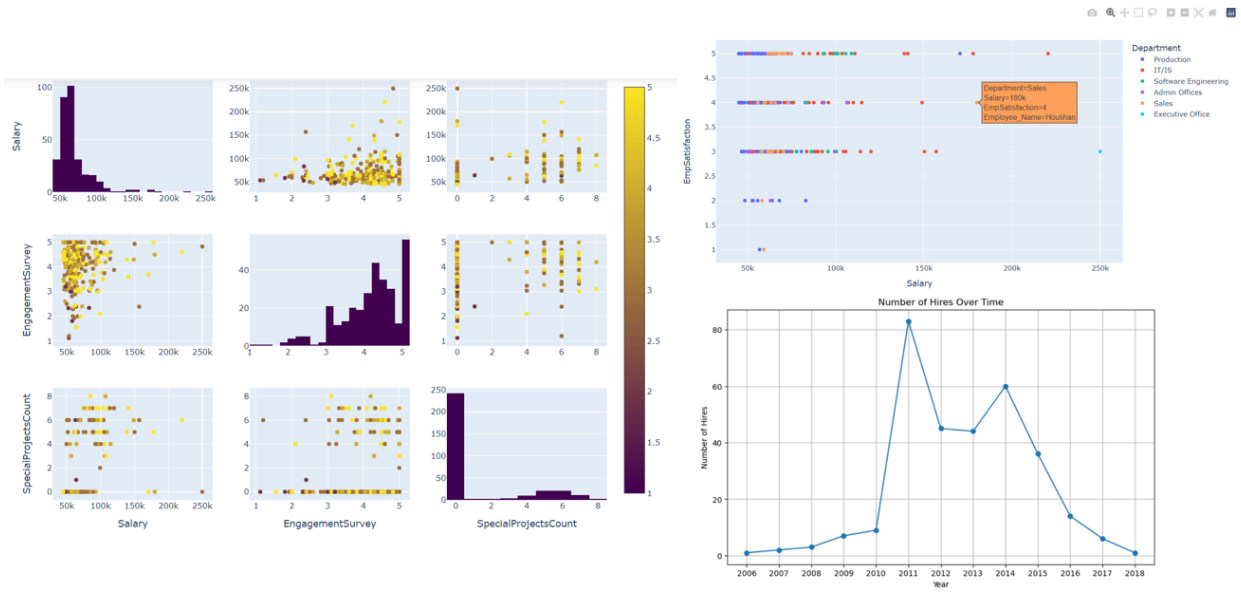
Customer Transactions Dashboard



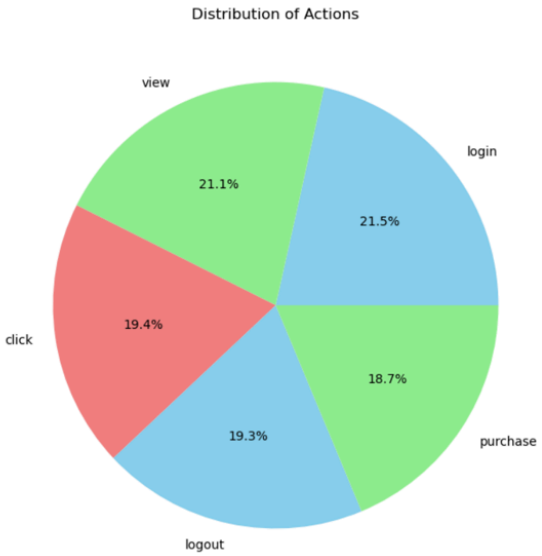


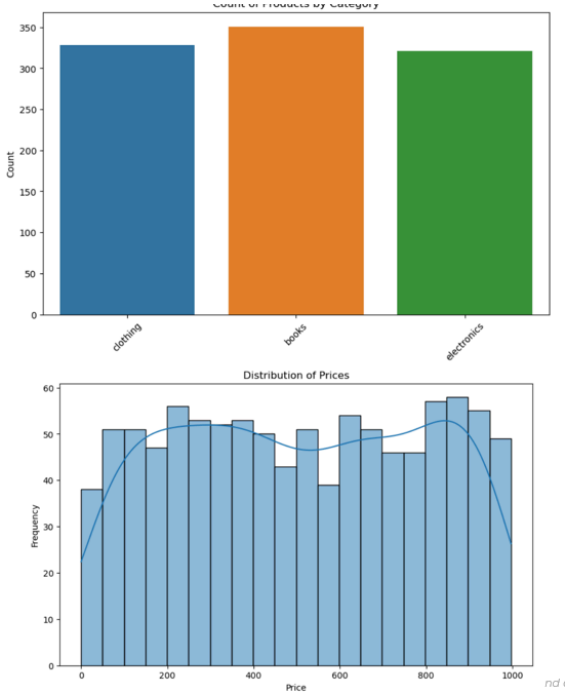


HR Data Dashboard



Clickstream Data Dashboard





API Data Dashboard