

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:

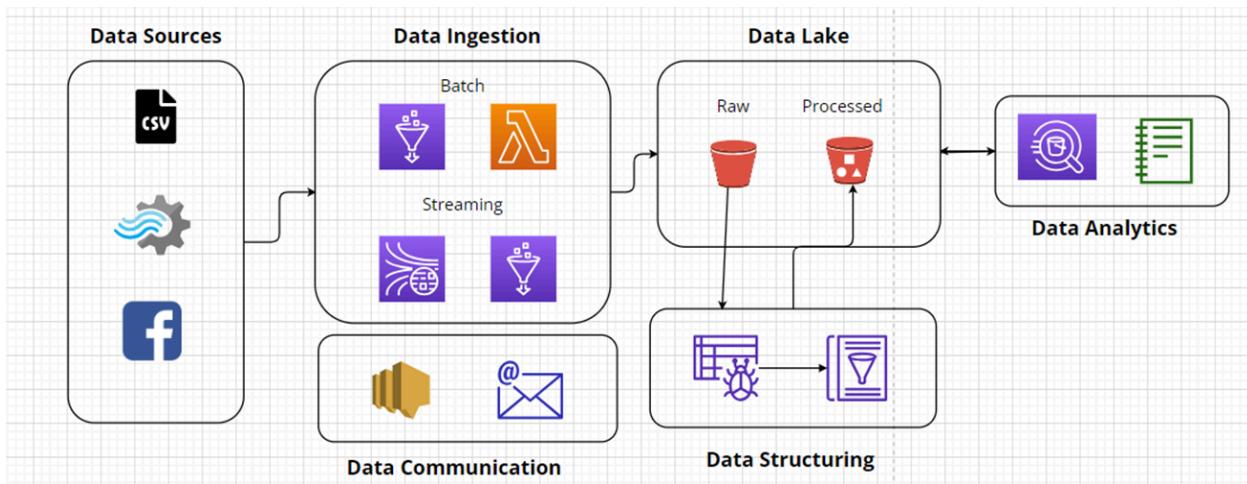
1. Data Ingestion:
 - a. Capability to ingest structured and unstructured data.
 - b. Real-time and batch data ingestion functionalities.
 - c. Integration with third-party sources, including ClickStream, Social Media, and External HR Data.
2. Scalable Data Warehouse:
 - a. Cloud-based scalable data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

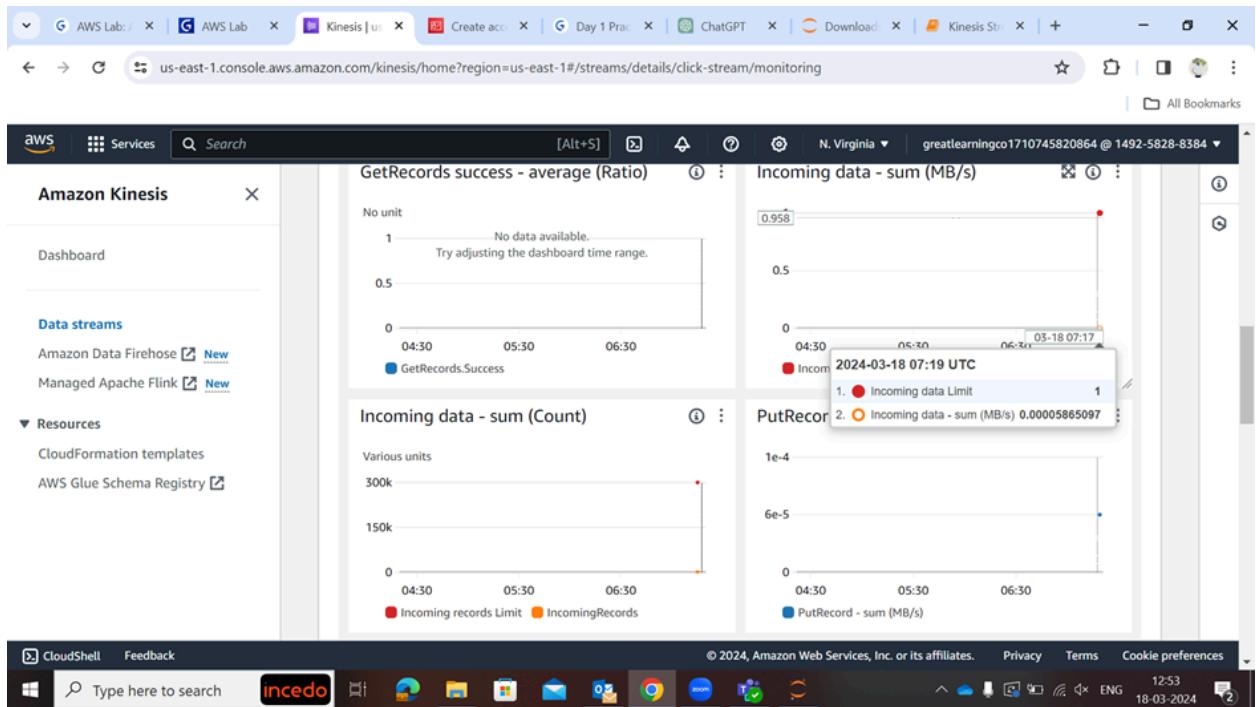
1. Records are generated using Faker library in Python simulating a real time streaming.
2. These records are put into the kinesis stream using boto3 AWS SDK.
3. Then a simple Glue job converts them into a CSV file. This CSV file then goes for further ETL using Glue Studio.

Clickstream kinesis stream:

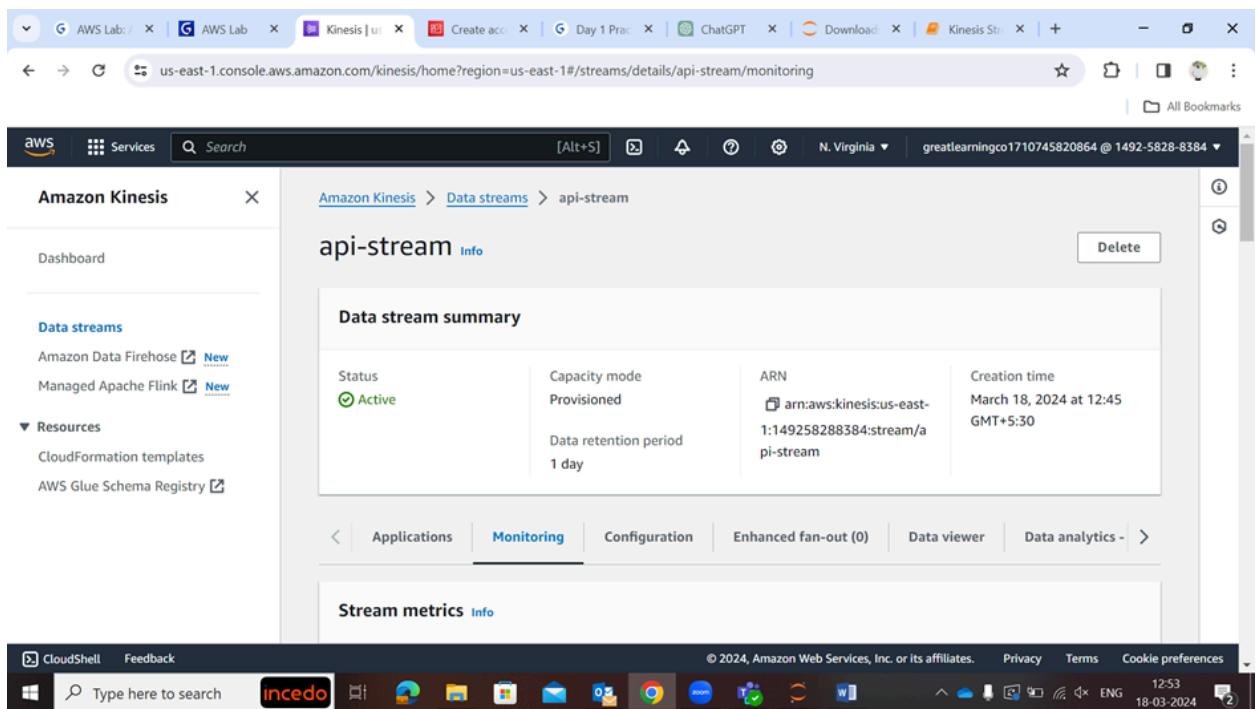
Screenshot of the AWS Kinesis Data Stream details page for the "click-stream" stream. The page shows the following information:

- Data stream summary:**
 - Status: Active
 - Capacity mode: Provisioned
 - ARN: arn:aws:kinesis:us-east-1:149258288384:stream/click-stream
 - Creation time: March 18, 2024 at 12:44 GMT+5:30
- Monitoring:** The active tab, showing Stream metrics and Info.
- Stream metrics:** A table showing various metrics over time.

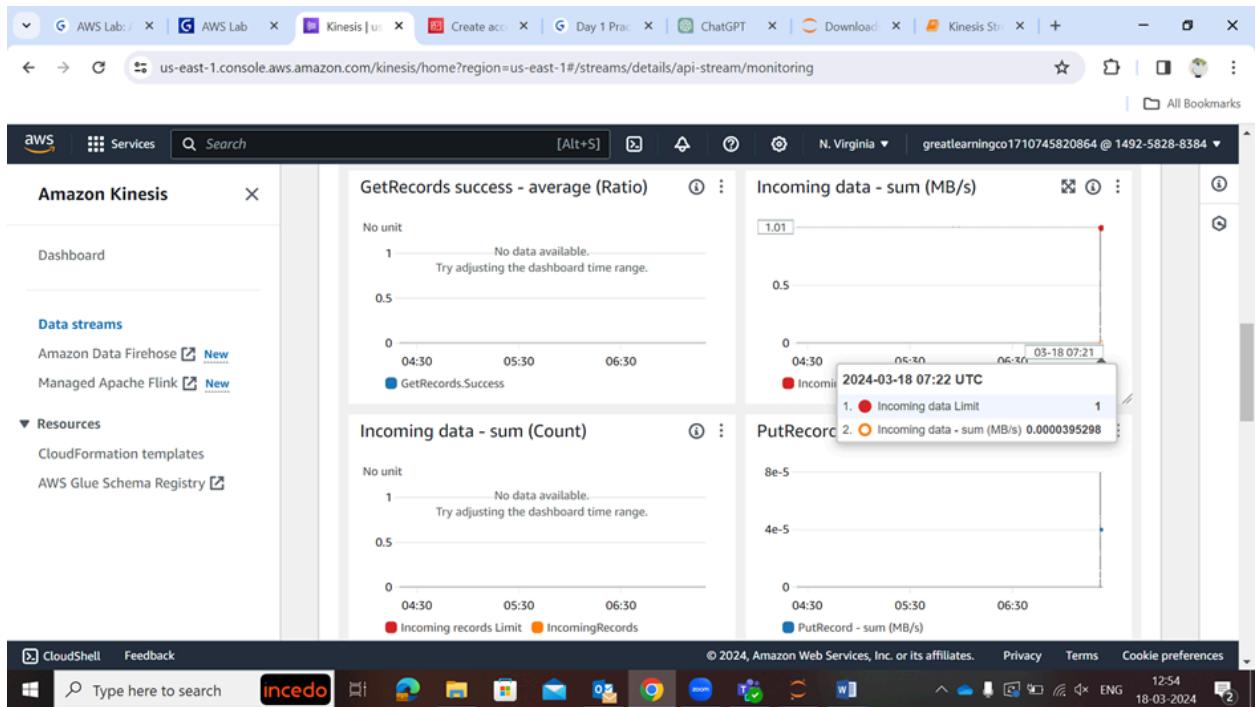
Data is put in stream:



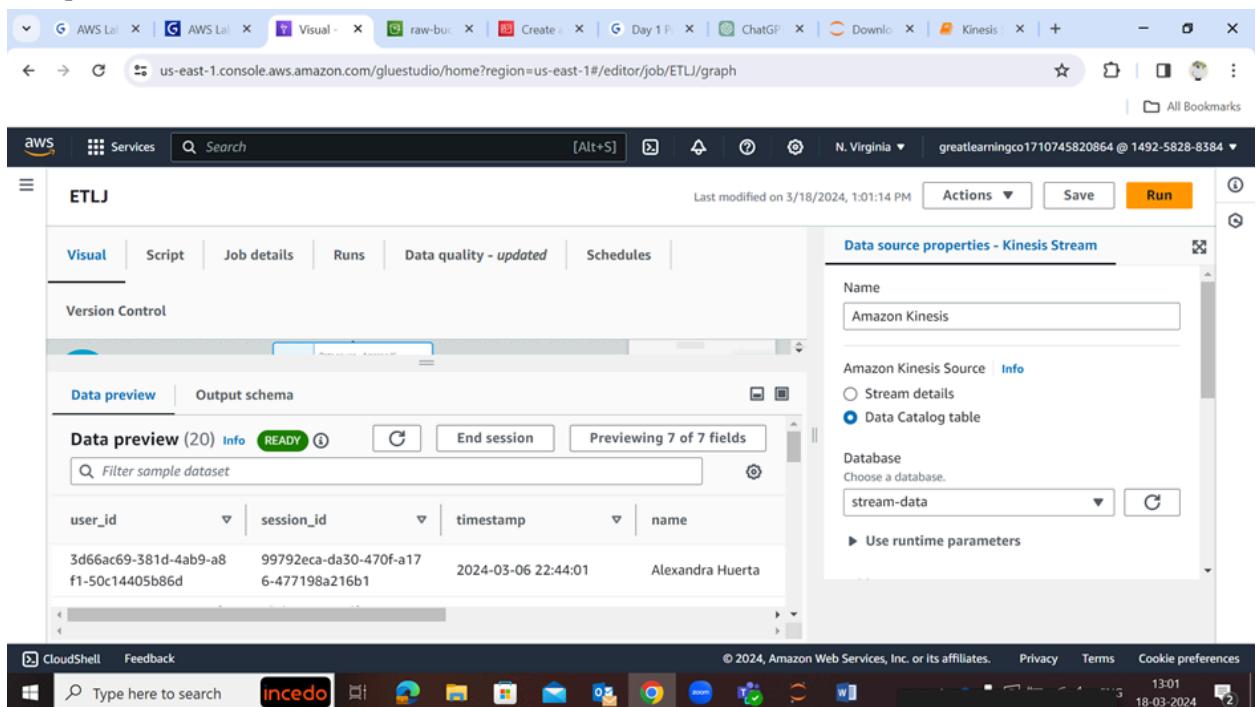
API-stream:



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:

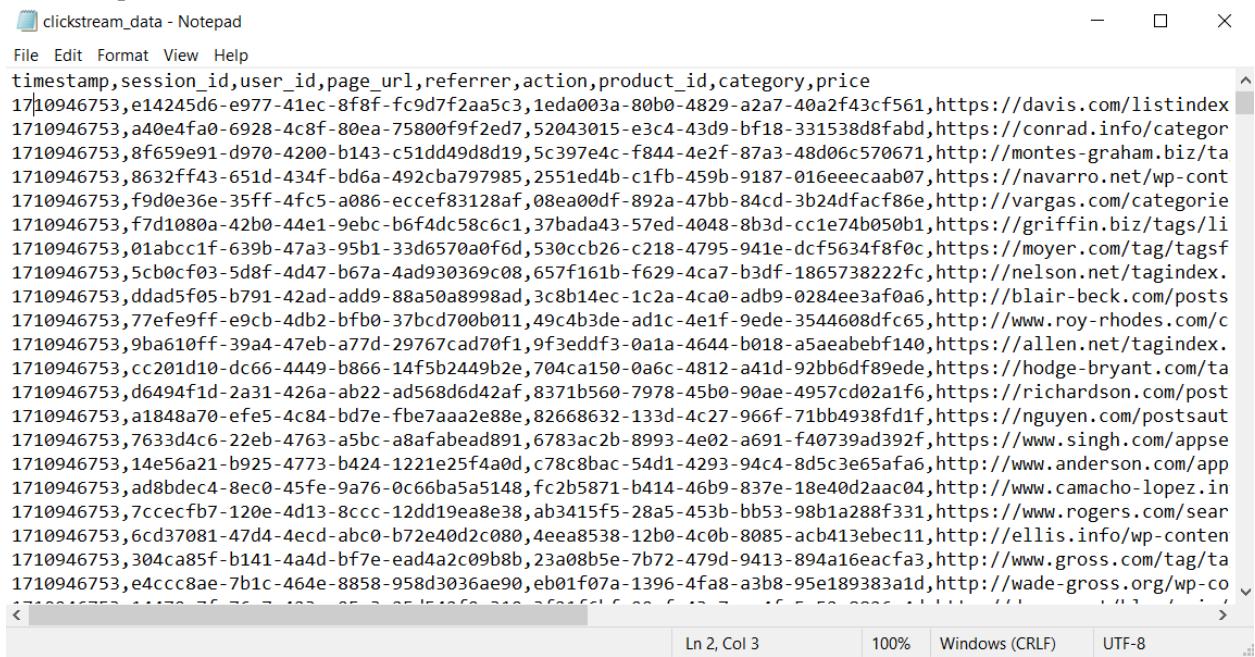
A	B	C	D	E	F	G	H	I	J	K	L	M
customer	dateofbirth	citizenship	currCount	employme	incomeInT	marketSeg	maritalSta	stateCode	city	country	accountopeningdate	gender
b90ec527-	11/27/1995	US	USD	worker	\$200685.8	standard	Married	FL	Zephyrhills	United Sta	09-06-2021	Male
f8162ec9-	05-03-2002	US	USD	self-emplo	\$121102.5	basic	Married	CT	Waterbury	United Sta	1/30/2021	Female
8c50fecc-	7/21/1994	US	USD	employee	\$234625.6	standard	Widowed	NY	Staten Isla	United Sta	6/23/2021	Female
5abf4208-	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-	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	employe	incomeInT	marketSeg	maritalSta	stateCode	city	country	accountopeningdat	gender
b90ec527-d54c-4546-8872	11/27/1995	US	USD	worker	##### standard	Married	FL	Zephyrhills	United Sta	9/6/2021	Male	
f8162ec9-2f9d-4412-8c22-	5/3/2002	US	USD	self-emplc	##### 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-emplc	##### standard	Separated	FL	Panama Ci	United Sta	1/20/2021	Male	
b0319a45-e62-4049-a33c	6/28/1993	US	USD	worker	#####	Separated	FL	Tallahasse	United Sta	3/7/2021	Male	
2deb405-0a05-4f65-90f6	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-emplc	##### basic	Widowed	MO	Kansas Cit	United Sta	6/20/2021	Female	
b7fac2b9-a461-48d8-9129	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:



```

File Edit Format View Help
timestamp,session_id,user_id,page_url,referrer,action,product_id,category,price
17|10946753,e14245d6-e977-41ec-8f8f-fc9d7f2aa5c3,1eda003a-80b0-4829-a2a7-40a2f43cf561,https://davis.com/listindex
17|10946753,a40e4fa0-6928-4c8f-80ea-75800f9f2ed7,52043015-e3c4-43d9-bf18-331538d8fabd,https://conrad.info/categor
17|10946753,8f659e91-d970-4200-b143-c51dd49d819,5c397e4c-f844-4e2f-87a3-48d06c570671,http://montes-graham.biz/ta
17|10946753,8632ff43-651d-434f-bd6a-492cba797985,2551ed4b-c1fb-459b-9187-016eeecaab07,https://navarro.net/wp-cont
17|10946753,f9d0e36e-35ff-4fc5-a086-eccef83128af,08ea00df-892a-47bb-84cd-3b24dfacf86e,http://vargas.com/categorie
17|10946753,f7d1080a-42b0-44e1-9ebc-b6f4dc58c6c1,37bada43-57ed-4048-8b3d-cc1e74b050b1,https://griffin.biz/tags/li
17|10946753,01abcc1f-639b-47a3-95b1-33d6570a0f6d,530ccb26-c218-4795-941e-dcf5634f8f0c,https://moyer.com/tag/tagsf
17|10946753,5cb0cf03-5d8f-4d47-b67a-4ad930369c08,657f161b-f629-4ca7-b3df-1865738222fc,http://nelson.net/tag/index.
17|10946753,ddad5f05-b791-42ad-add9-88a50a8998ad,3c8b14ec-1c2a-4ca0-adb9-0284ee3af0a6,http://blair-beck.com/posts
17|10946753,77efe9ff-e9cb-4db2-bfb0-37bcd700b011,49c4b3de-ad1c-4e1f-9ede-3544608dfc65,http://www.roy-rhodes.com/c
17|10946753,9ba610ff-39a4-47eb-a77d-29767cad70f1,9f3eddff-0a1a-4644-b018-a5eabebf140,https://allen.net/tagindex.
17|10946753,cc201d10-dc66-4449-b866-14f5b2449b2e,704ca150-0a6c-4812-a41d-92bb6df89ede,https://hodge-bryant.com/ta
17|10946753,d6494f1d-2a31-426a-ab22-ad568d6d42af,8371b560-7978-45b0-90ae-4957cd02a1f6,https://richardson.com/post
17|10946753,a1848a70-efef-4c84-bd7e-fbe7aaa2e88e,82668632-133d-4c27-966f-71bb4938fd1f,https://nguyen.com/postsaut
17|10946753,7633d4c6-22eb-4763-a5bc-a8afabead891,6783ac2b-8993-4e02-a691-f40739ad392f,https://www.singh.com/appse
17|10946753,14e56a21-b925-4773-b424-1221e25f4a0d,c78c8bac-54d1-4293-94c4-8d5c3e65afa6,http://www.anderson.com/app
17|10946753,ad8bdec4-8ec0-45fe-9a76-0c66ba5a5148,fc2b5871-b414-46b9-837e-18e40d2aac04,http://www.camacho-lopez.in
17|10946753,7ccecfb7-120e-4d13-8ccc-12dd19ea8e38,ab3415f5-28a5-453b-bb53-98b1a288f331,https://www.rogers.com/sear
17|10946753,6cd37081-47d4-4ecd-abc0-b72e40d2c080,4eea8538-12b0-4c0b-8085-acb413ebec11,http://ellis.info/wp-conten
17|10946753,304ca85f-b141-4a4d-bf7e-ead4a2c09b8b,23a08b5e-7b72-479d-9413-894a16eacfa3,http://www.gross.com/tag/ta
17|10946753,e4ccc8ae-7b1c-464e-8858-958d3036ae90,eb01f07a-1396-4fa8-a3b8-95e189383a1d,http://wade-gross.org/wp-co

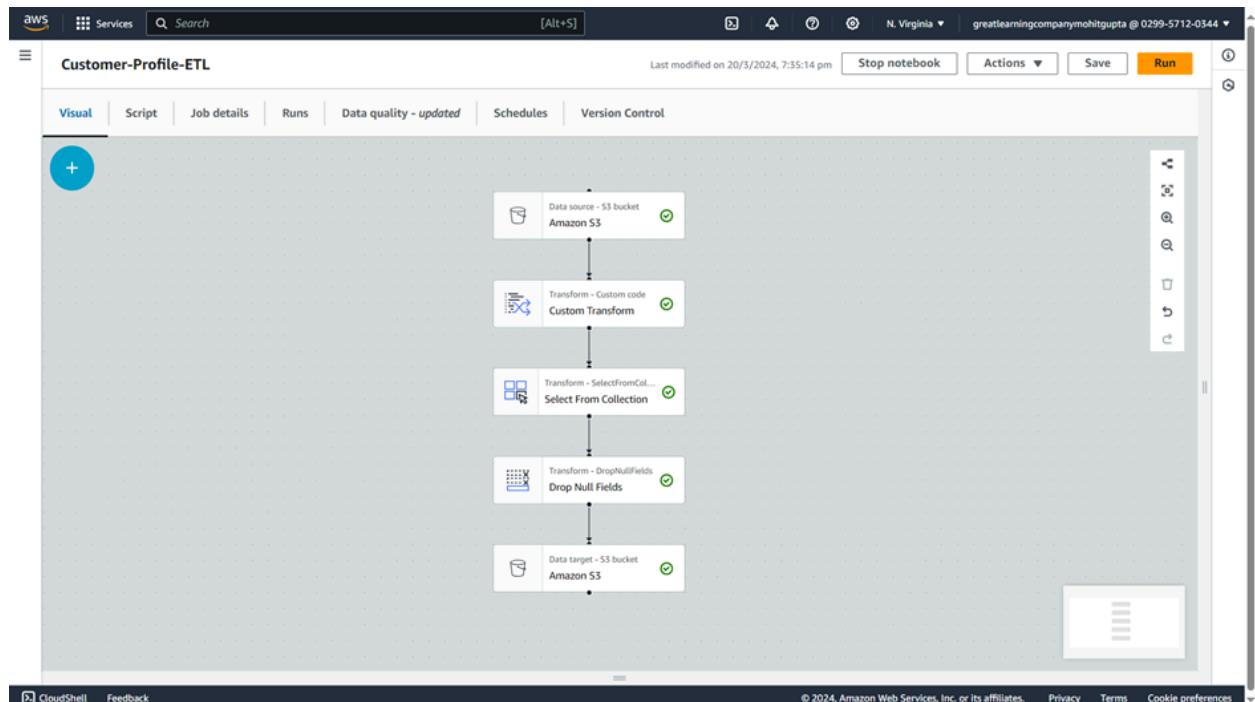
```

Timestamp column after:

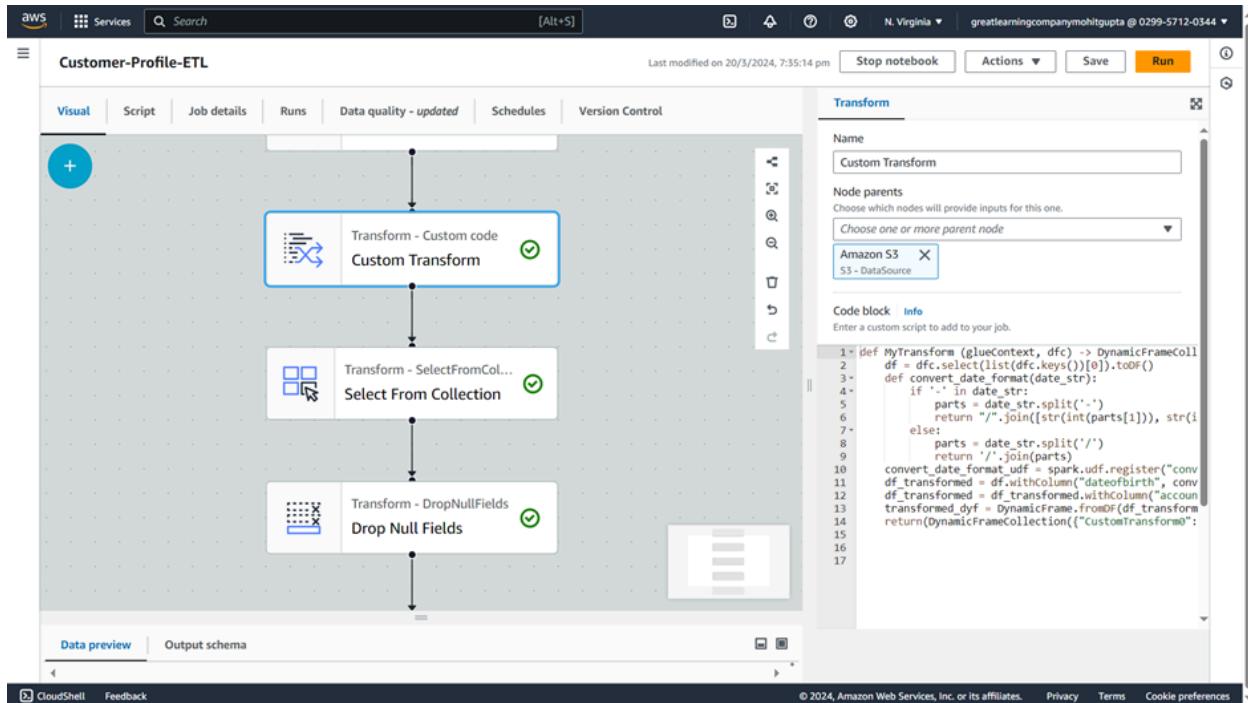
```
clean_clickstream - Notepad
File Edit Format View Help
|timestamp,session_id,user_id,page_url,referrer,action,product_id,category,price
"2024-03-20 14:59:13.0","e14245d6-e977-41ec-8f8f-fc9d7f2aa5c3","1eda003a-80b0-4829-a2a7-40a2f43cf561","https://d
"2024-03-20 14:59:13.0","a40e4fa0-6928-4c8f-80ea-75800f9f2ed7","52043015-e3c4-43d9-bf18-331538d8fabd","https://c
"2024-03-20 14:59:13.0","8f6599e91-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-9187-016eeeaca0b7","https://n
"2024-03-20 14:59:13.0","f9d0e36e-35ff-4fc5-a086-eccef83128af","08ea00df-892a-47bb-84cd-3b24dfac86e","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","77efef9f-e9cb-4db2-bfb0-37bcd700b011","49c4b3de-ad1c-4e1f-9ede-3544608dbf65","http://ww
"2024-03-20 14:59:13.0","9ba610ff-39a4-47eb-a77d-29767cad70f1","973eddf3-0a1a-4644-b018-a5aeabef140","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-5d41-4293-94c4-8d5c3e65afa6","http://ww
"2024-03-20 14:59:13.0","ad8bdec4-8ec0-45fe-9a76-0c66ba5a5148","fc2b5871-b414-46b9-837e-18e40d2aa04","http://ww
"2024-03-20 14:59:13.0","7cccef7b-120e-4d13-8ccc-12dd19ea8e38","ab3415f5-28a5-453b-bb53-98b1ba88f331","https://w
"2024-03-20 14:59:13.0","6cd37081-47d4-4ecd-abc0-b72e40d2c080","4eea8538-12b0-4c0b-8085-acb413ebecc1","http://e1
"2024-03-20 14:59:13.0","304ca85f-b141-4a4d-bf7e-ead4a2c09b8b","23a08b5e-7b72-479d-9413-894a16eacf3","http://ww
"2024-03-20 14:59:13.0","e4ccc8ae-7b1c-464e-8858-958d3036ae90","eb01f07a-1396-4fa8-a3b8-95e189383a1d","http://wa
```

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:

The screenshot shows the AWS Glue job run details for the "Customer-Profile-ETL" job. At the top, a green banner indicates "Successfully started job Customer-Profile-ETL. Navigate to Run details for more details." Below this, the "Runs" tab is selected. The "Job runs (1/2) Info" section shows two runs: one succeeded at 14:19:56 and another succeeded at 14:15:46. The "Run details" tab is selected, displaying the job name, start time, glue version, and last modified time for each run. The "Input arguments (10)" tab is also visible.

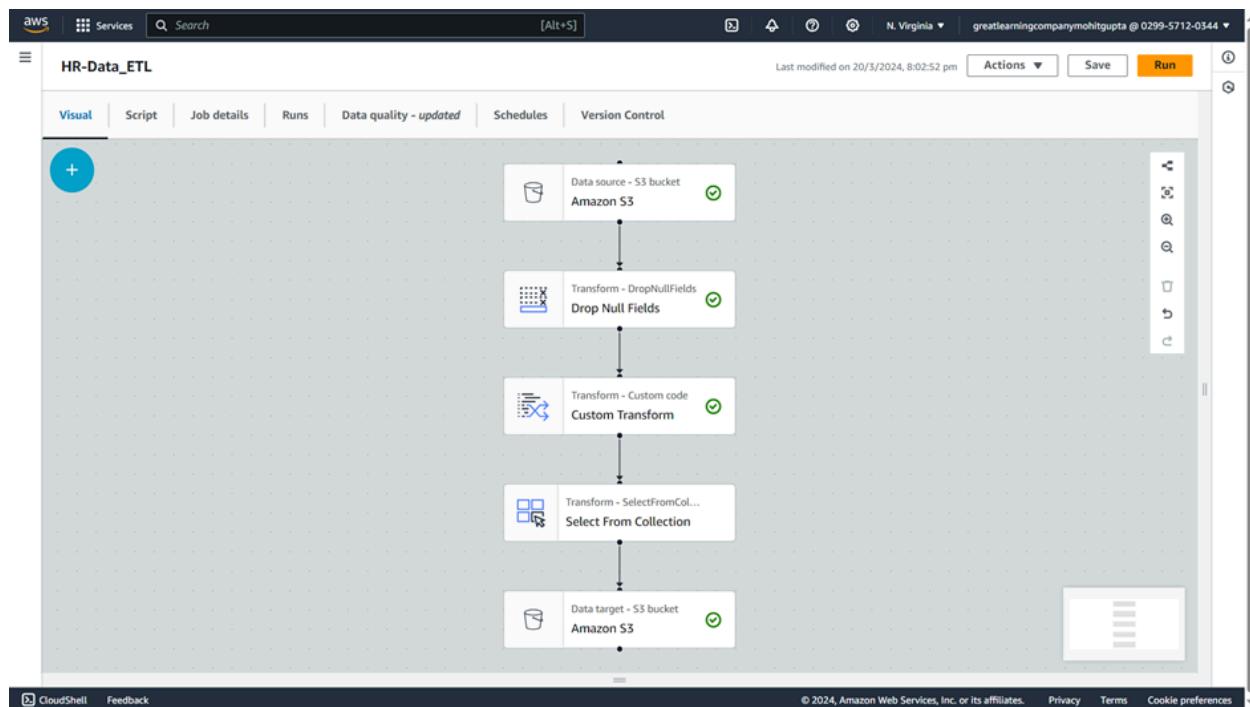
ETL of Customer Transactions data:

The screenshot shows the AWS Glue visual editor interface for creating ETL jobs. A simple pipeline is defined with three main components: an "Amazon S3" data source, a "Drop Null Fields" transform, and an "Amazon S3" data target. The components are connected sequentially, with arrows indicating the flow from source to transform to target. The "Visual" tab is selected at the top.

Job run Succeeded:

The screenshot shows the AWS Glue job run details page for the job "Customer-Profile-ETL". The top navigation bar includes the AWS logo, Services, Search, and account information (N. Virginia, greatlearningcompanymohitgupta @ 0299-5712-0344). A green banner at the top indicates "Successfully started job Customer-Profile-ETL. Navigate to Run details for more details." The main content area has tabs for Visual, Script, Job details, Runs (selected), Data quality - updated, Schedules, and Version Control. Under the "Runs" tab, it says "Job runs (1/3) Info" and "Last updated (UTC) March 20, 2024 at 14:52:00". There is a table showing three runs, all of which have "Succeeded" status. The table columns include Run status, Retries, Start time (UTC), End time (UTC), Duration, Capacity (DPUs), Worker type, and Glue version. The third run is highlighted with a blue border. Below the table, there are tabs for Run details, Input arguments (10), Continuous logs, Run insights, Metrics, and Spark UI.

ETL of HR Data:



Job run succeeded:

The screenshot shows the AWS Glue Job Run Details page for the job **HR-Data_ETL**. The job run status is **Succeeded**, and it completed at **2024/03/20 14:41:14** with a duration of **2 m 31 s**. The run used **10 DPUs** and was of type **G.1X** with **Glue version 4.0**.

Run status	Retries	Start time (UTC)	End time (UTC)	Duration	Capacity (DPUs)	Worker type	Glue version
Succeeded	0	2024/03/20 14:41:14	2024/03/20 14:43:55	2 m 31 s	10 DPUs	G.1X	4.0

Run details | Input arguments (10) | Continuous logs | Run insights | Metrics | Spark UI

Job name: **HR-Data_ETL** | Start time (UTC): **2024/03/20 14:41:14** | Glue version: **4.0** | Last modified on (UTC): **2024/03/20 14:43:55**

Id: **jr_612af9d79bb57fd86ca7937aca1e3fcbe800ccb3d6 4cc39cf92556c261ff2270** | End time (UTC): **2024/03/20 14:43:55** | Worker type: **G.1X** | Log group name: **/aws-glue/jobs**

Run status: **Succeeded** | Start-up time: **9 seconds** | Max capacity: **10 DPUs** | Number of workers: **10**

Retry attempt number: **Initial run** | Execution time: **2 minutes 31 seconds** | Execution class: **Standard** | Timeout: **2880 minutes**

Bucket screenshots:

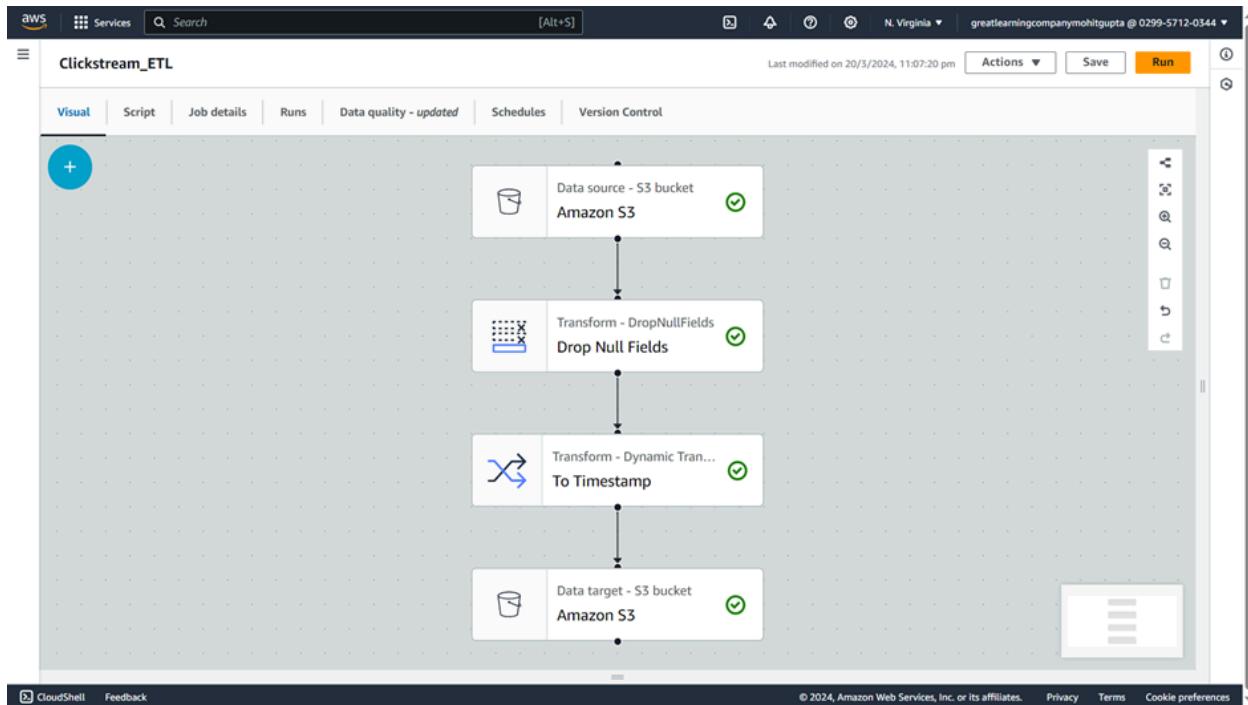
The screenshot shows the AWS S3 Bucket **processed-bucket356**. The bucket contains three objects, all of which are folders:

- clean_Customer_Profile/**
- clean_Customer_Transactions/**
- clean_HR_Data/**

Objects (3) Info

Name	Type	Last modified	Size	Storage class
clean_Customer_Profile/	Folder	-	-	-
clean_Customer_Transactions/	Folder	-	-	-
clean_HR_Data/	Folder	-	-	-

ETL of Clickstream data:



Job run succeeded:

The screenshot shows the AWS Glue service dashboard under the 'Runs' tab for the 'Clickstream_ETL' job. It displays the following information:

Run status	Retries	Start time (UTC)	End time (UTC)	Duration	Capacity (DPU)	Worker type	Glue
Succeeded	0	2024/03/20 17:42:47	2024/03/20 17:45:10	2 m 5 s	10 DPU	G.1X	4.0
Succeeded	0	2024/03/20 17:37:56	2024/03/20 17:39:48	1 m 42 s	10 DPU	G.1X	4.0

Below the table, the 'Run details' section provides specific details for the first run:

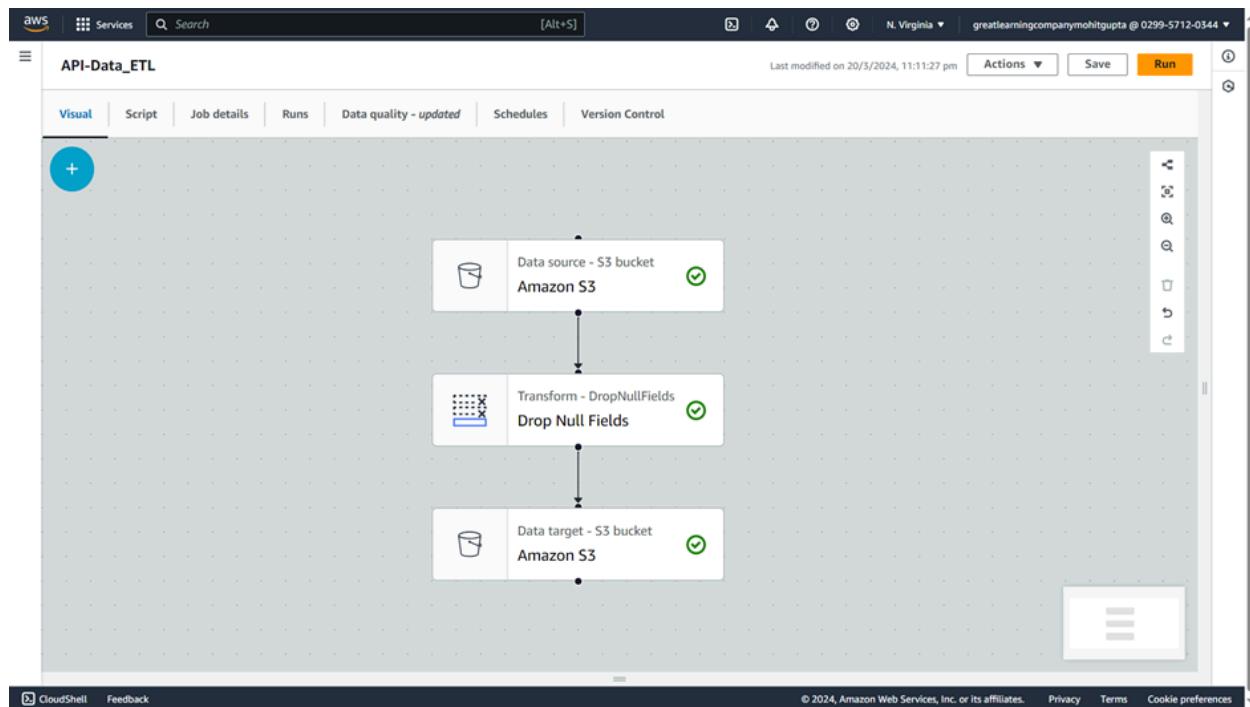
Job name	Start time (UTC)	Glue version	Last modified on (UTC)
Clickstream_ETL	2024/03/20 17:42:47	4.0	2024/03/20 17:45:10
Id	End time (UTC)	Worker type	Log group name
jr_792771a3d1dcbb137e14033eb07cf8fe9c 07021c7c0faeca5113ecd78279da31	2024/03/20 17:45:10	G.1X	/aws-glue/jobs
Run status	Start-up time	Max capacity	Number of workers
Succeeded	17 seconds	10 DPU	10
Retry attempt number	Execution time	Execution class	Timeout

Bucket screenshot:

The screenshot shows the AWS S3 console interface. The top navigation bar includes the AWS logo, Services, a search bar, and user information: greatlearningcompanymohitgupta @ 0299-5712-0344. Below the navigation is a breadcrumb trail: Amazon S3 > Buckets > processed-bucket25346 > clean_clickstream/. The main content area displays the 'Objects' tab for the 'clean_clickstream/' bucket. A single object is listed:

Name	Type	Last modified	Size	Storage class
run-1710956696054-part-r-00000	-	March 20, 2024, 23:14:59 (UTC+05:30)	219.8 KB	Standard

ETL of API Data:



Job run succeeded:

The screenshot shows the AWS Glue console with the job **API-Data_ETL**. The **Runs** tab is selected, displaying two successful runs. The first run started at 2024/03/20 17:45:15 and ended at 2024/03/20 17:46:34, with a duration of 1m 14s and 10 DPUUs. The second run started at 2024/03/20 17:41:51 and ended at 2024/03/20 17:44:15, with a duration of 2m 7s and 10 DPUUs. Both runs have a status of **Succeeded** and 0 retries.

Run status	Retries	Start time (UTC)	End time (UTC)	Duration	Capacity (DPUUs)	Worker type	Glue
Succeeded	0	2024/03/20 17:45:15	2024/03/20 17:46:34	1m 14 s	10 DPUUs	G.1X	4.0
Succeeded	0	2024/03/20 17:41:51	2024/03/20 17:44:15	2m 7 s	10 DPUUs	G.1X	4.0

Bucket screenshot:

The screenshot shows the Amazon S3 console with the bucket **clean_API_Data**. One object is listed: **run-1710956781876-part-r-00000**. The object was last modified on March 20, 2024, at 23:16:23 (UTC+05:30) and has a size of 128.0 KB. The storage class is Standard.

Name	Type	Last modified	Size	Storage class
run-1710956781876-part-r-00000	-	March 20, 2024, 23:16:23 (UTC+05:30)	128.0 KB	Standard

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.

The screenshot shows the AWS Glue Data Catalog interface. On the left, there's a sidebar with navigation links like 'Getting started', 'ETL jobs', 'Visual ETL', 'Notebooks', 'Job run monitoring', 'Data Catalog tables', 'Data connections', and 'Workflows (orchestration)'. Under 'Data Catalog', there are sections for 'Databases', 'Tables', 'Stream schema registries', 'Schemas', 'Connections', 'Crawlers', 'Classifiers', and 'Catalog settings'. The main panel displays a table titled 'Schema (10)' for a table named 'bucket92347/clean_api-data.csv'. The table has columns for '#', 'Column name', 'Data type', 'Partition key', and 'Comment'. The data includes columns for user_id (bigint), activity (string), date (string), post_id (bigint), content_type (string), session_id (string), name (string), email (string), age (bigint), and location (string). At the top of the main panel, it shows 'Input format: org.apache.hadoop.mapred.TextInputFormat' and 'Output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'.

The screenshot shows the AWS Athena query results page. At the top, it says 'https://us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/27b8738c-549d-4f79-9ad2-016b86994dea' and 'N. Virginia'. The main area is titled 'Query results' and shows a table titled 'Results (10)'. The table has columns for '#', 'user_id', 'activity', 'date', 'post_id', 'content_type', and 'session_id'. The data rows are:

- 1 3 "visited profile" 1994-10-13 4293 Image "59a346b2-0a6b-46e2-9d78-a65b795b26"
- 2 3 shared 2020-06-13 3383 Article "dd1eab80-f083-49a6-9780-c351c43e7ea"
- 3 3 "clicked link" 1999-11-07 8251 Article "a042e587-c4cb-497b-a091-201a2aae5c0"
- 4 1 "visited profile" 2010-02-09 Image "fd5430d7-ac43-40cd-8f90-e8b7c462c0d8"
- 5 3 shared 1974-11-12 Video "6d54f517-135e-4b1f-a35d-f3d8ee338f46"
- 6 3 "clicked link" 1980-11-01 Image "95b41ef4-dae9-4fd1-bae6-86ff2bf2629b"
- 7 1 commented 1983-07-05 1353 Video "336ce26a-08cd-407f-996a-c50971629c4l"
- 8 3 commented 1974-06-29 7310 Video "f1186901-f571-4cab-846e-acd1ec909644"
- 9 1 "visited profile" 2009-05-15 Image "5fa61c74-66e5-40f6-b566-707e2f9cdd0f"

At the bottom, it says 'CloudShell Feedback' and '© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences'.

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.

The screenshot shows an email from 'emailTopic <no-reply@sns.amazonaws.com>' to the user. The subject is 'AWS Notification - Subscription Confirmation'. The email body contains the following text:

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](#).

The screenshot shows an email from 'emailTopic <no-reply@sns.amazonaws.com>' to the user. The subject is 'New Update from Your Website'. The email body contains the following text:

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.19bc3792@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 query 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.

The screenshot shows the AWS Athena Query Editor interface. The left sidebar displays the Data source (AwsDataCatalog) and Database (capstonedata). Under Tables and views, there are two tables: clean_api_data and maxvisitor. The main area contains the following SQL query:

```

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

```

The status bar at the bottom indicates the query completed successfully with the following metrics: Time in queue: 57 ms, Run time: 1.02 sec, Data scanned: 128.04 KB.

The screenshot shows the AWS S3 console for the max-visitor982 bucket. The Objects tab is selected, displaying one object: 20240326_154748_00050_uxcys_18869059-c570-4cc9-ac5f-24e5c5b07010.gz. The object was uploaded on March 26, 2024, at 21:17:50 (UTC+05:30) and has a size of 49.0 B. The storage class is Standard.

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:

The screenshot shows the AWS Lambda 'Add trigger' configuration page. The 'Trigger configuration' section is open, showing an S3 trigger for the 's3/max-visitor982' bucket. The 'Event types' dropdown is set to 'All object create events'. The 'Prefix - optional' field contains 'e.g. images/'. The 'Suffix - optional' field is empty. The right sidebar displays a tutorial titled 'Create a simple web app' with steps to build a Lambda function that outputs a webpage.

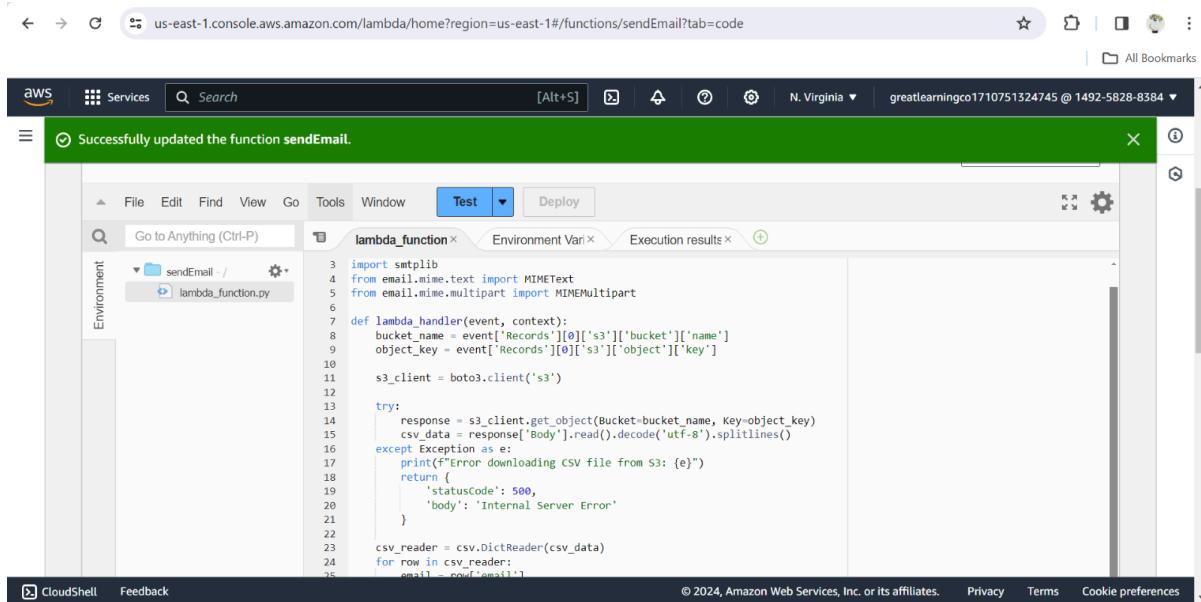
Trigger configuration

- Bucket:** s3/max-visitor982
- Event types:** All object create events
- Prefix - optional:** e.g. images/

Create a simple web app

- In this tutorial you will learn how to:
 - Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
 - Invoke your function through its function URL

[Learn more](#) [Start tutorial](#)



```
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)
        csv_data = response['Body'].read().decode('utf-8').splitlines()
    except Exception as e:
        print(f"Error downloading CSV file from S3: {e}")
        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)

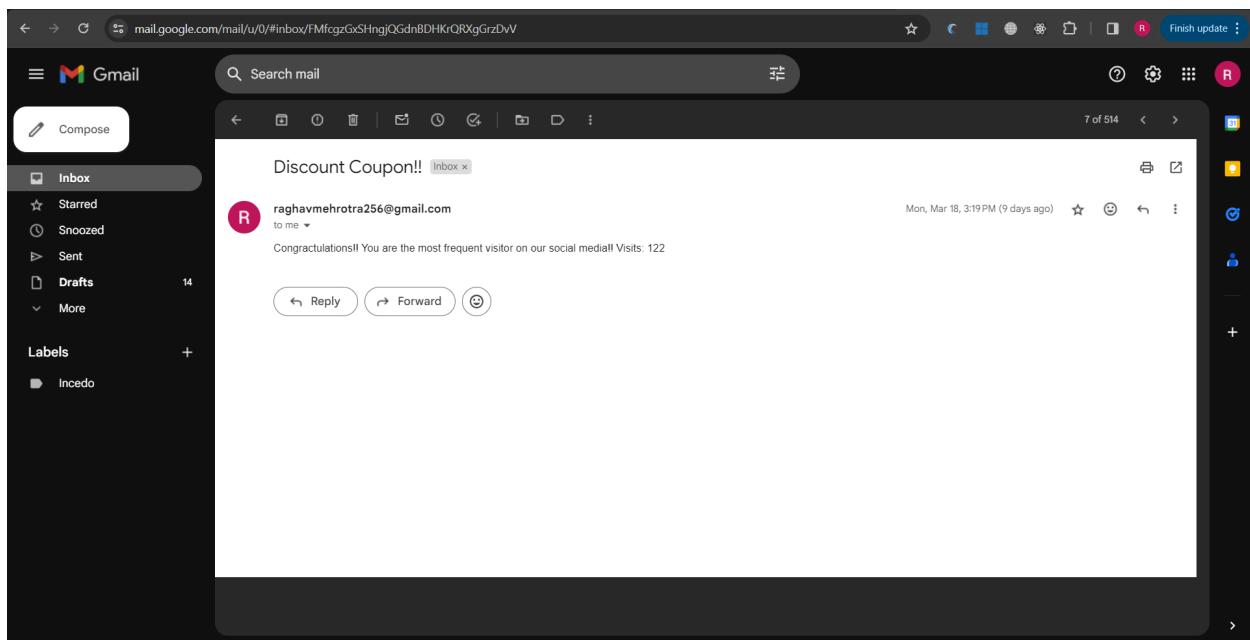
def send_email(recipient_email, count):
    # Retrieve S3 bucket and object key from the event
    sender_email = email
    subject = "Discount Coupon!!"
    characters = string.ascii_letters + string.digits
    coupon_code = ''.join(random.choice(characters) for i in range(8))
    message = "Congratulations!!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

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

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

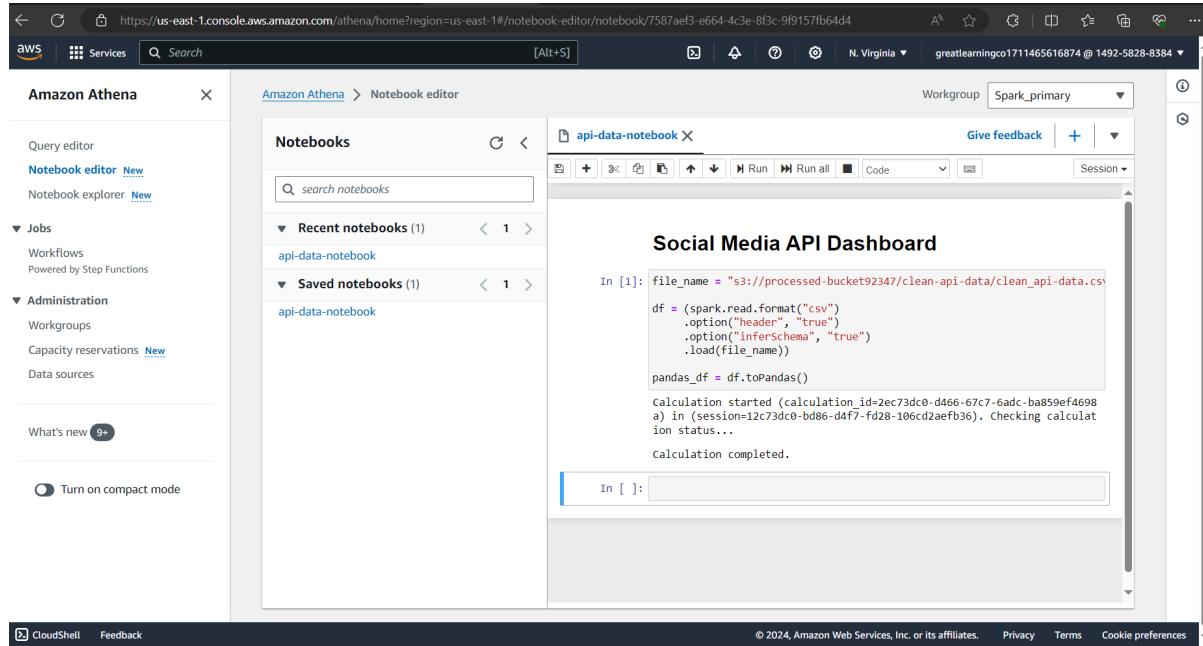
try:
    smtp_server = smtplib.SMTP('smtp.gmail.com', 587)
    smtp_server.starttls()
    smtp_server.login(sender_email, 'dvojbnfkdotjdx')
    smtp_server.sendmail(sender_email, recipient_email, msg.as_string())
    smtp_server.quit()
    print(f'Email sent to {recipient_email}')
except Exception as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

```



Visualization and Dashboard

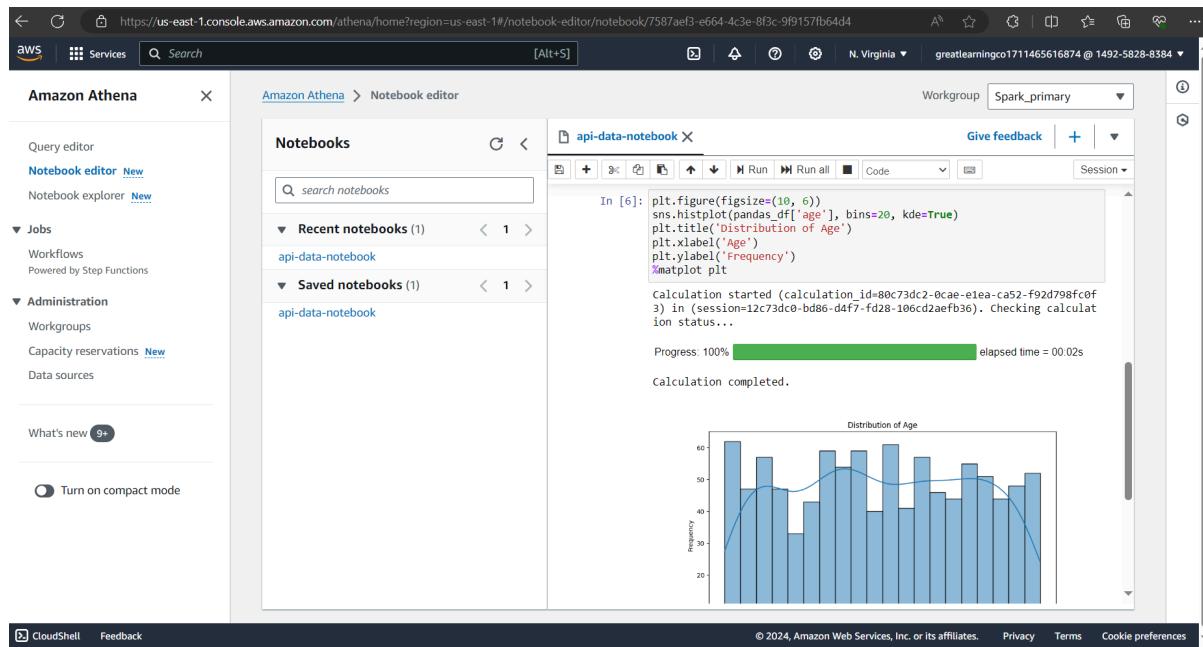
This is accomplished using AWS Athena's embedded notebooks. By loading the cleaned csv into a spark dataframe and then converting it to a pandas dataframe, we can create several interactive graphs using plotly, matplotlib and seaborn libraries.



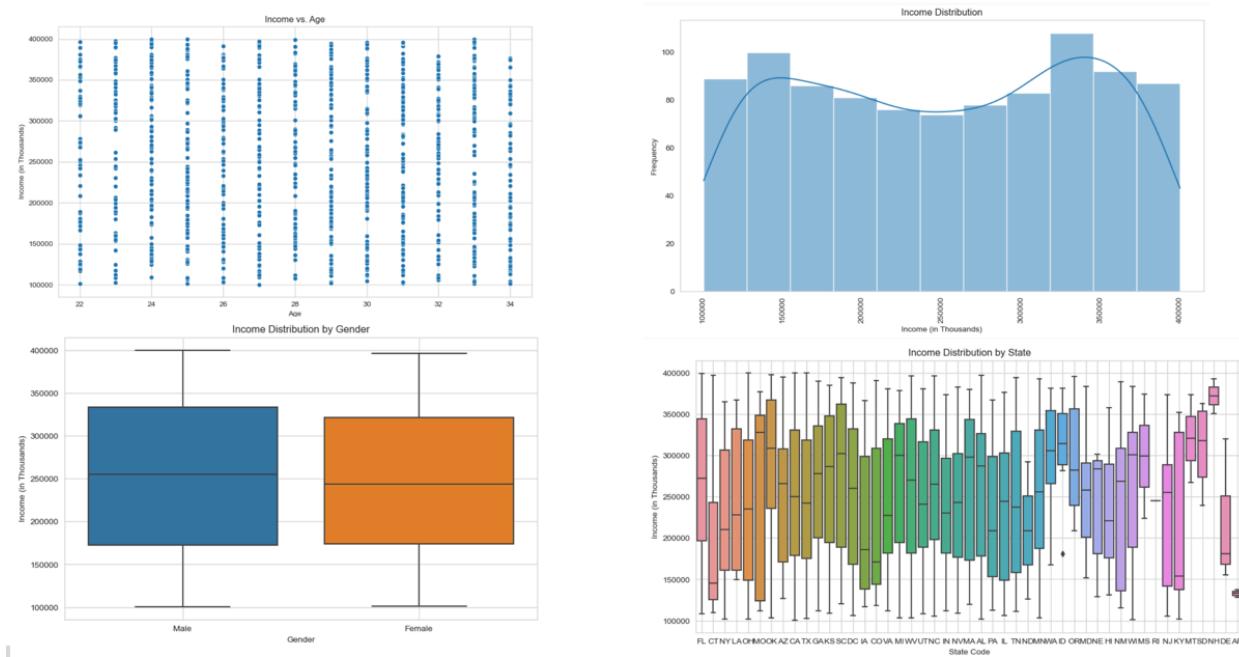
The screenshot shows the AWS Athena Notebook Editor interface. On the left, the sidebar includes 'Query editor', 'Notebook editor New' (which is selected), 'Notebook explorer New', 'Jobs', 'Workflows', 'Administration' (with 'Workgroups', 'Capacity reservations New', and 'Data sources'), and 'What's new'. The main area shows a 'Notebooks' list with 'Recent notebooks' and 'Saved notebooks', both containing 'api-data-notebook'. The notebook editor window displays a script titled 'Social Media API Dashboard' in 'In [1]'. The code reads a CSV file from S3, creates a DataFrame, and converts it to a Pandas DataFrame. The output shows the calculation started and completed.

```
file_name = "s3://processed-bucket92347/clean-api-data/clean_api-data.csv"
df = (spark.read.format("csv")
      .option("header", "true")
      .option("inferSchema", "true")
      .load(file_name))
pandas_df = df.toPandas()
```

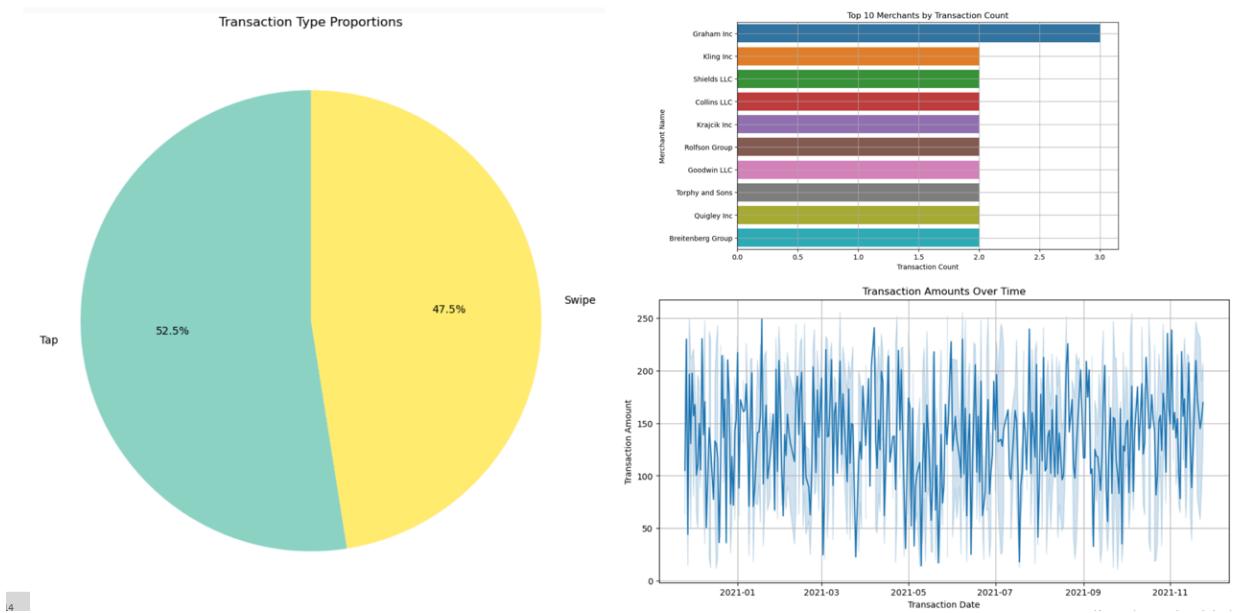
Calculation started (calculation_id=2ec73dc0-d466-67c7-6adc-ba859ef4698
a) in (session=12c73dc0-bd8e-d4f7-fd28-106cd2aeefb36). Checking calculat
ion status...
Calculation completed.



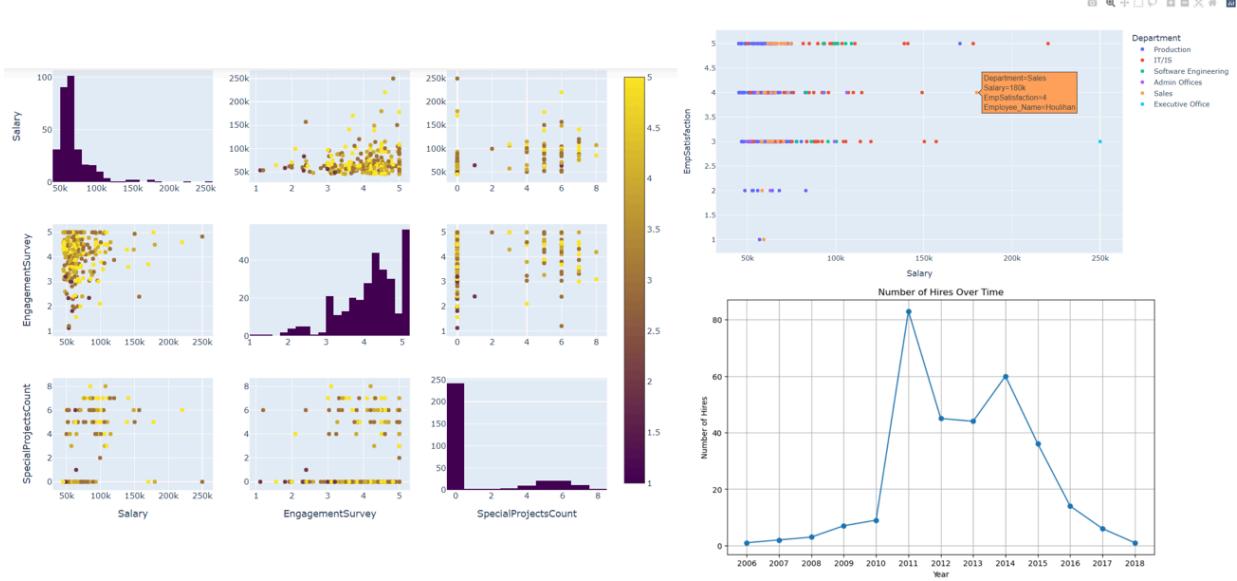
Customer Profile Dashboard



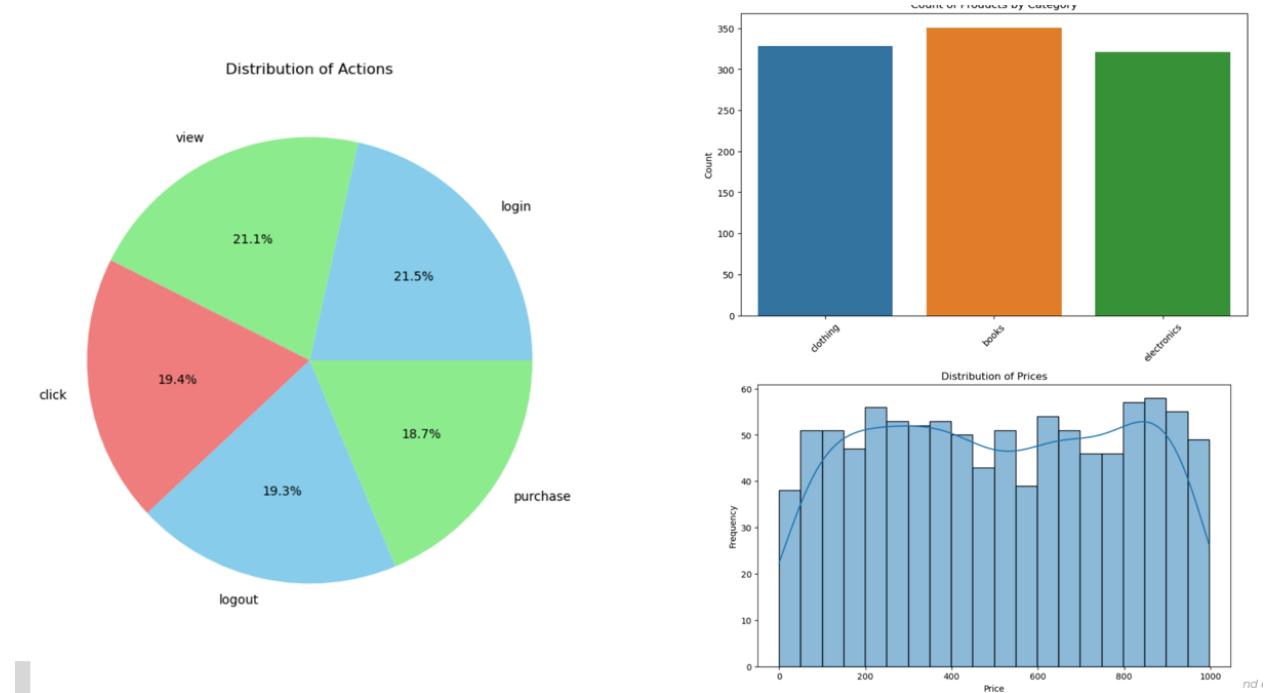
Customer Transactions Dashboard



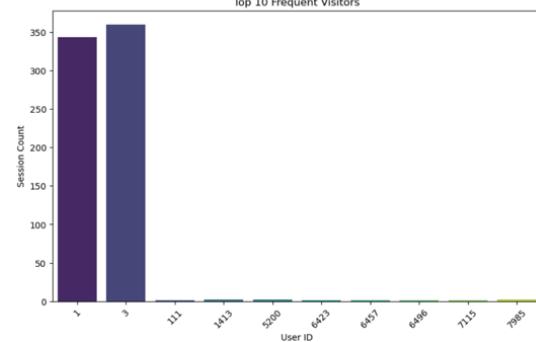
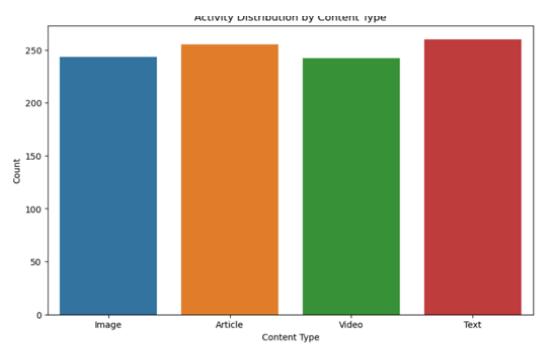
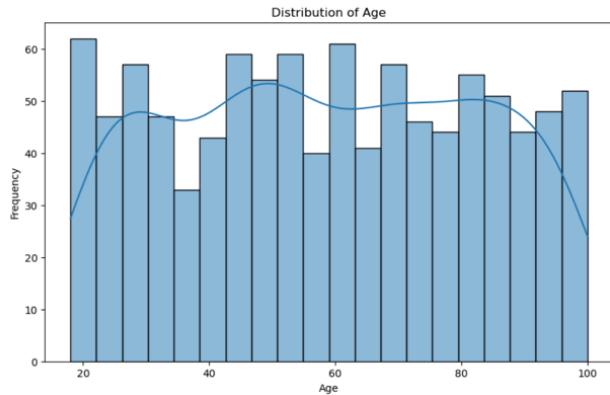
HR Data Dashboard



Clickstream Data Dashboard



API Data Dashboard



17