Smart City Real-Time Streaming & Analytics Pipeline

A real-time streaming architecture built with **Kafka, Spark Structured Streaming, AWS S3, Athena, Glue, and Redshift** to simulate, process, and analyze smart city data including vehicles, GPS, weather, traffic, and emergency events.

Table of Content

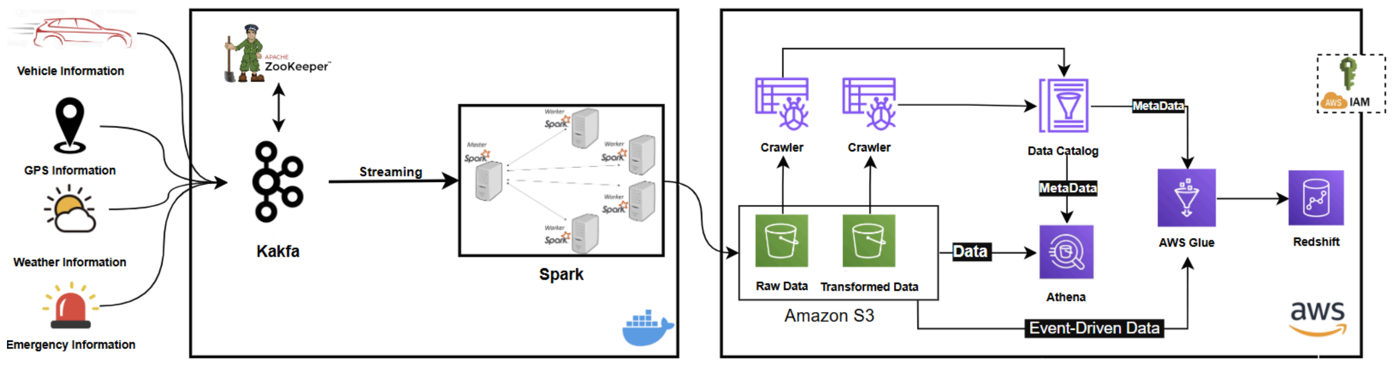
Project Overview

This project simulates a smart city environment where real-time data is continuously generated, streamed, processed, enriched, stored, and made queryable for analytics. The system supports:

- **Real-time data simulation** using Python.
- **Kafka** for stream ingestion.
- **Spark Structured Streaming** for transformation and joining.

- **S3 as a Data Lake** for storage.
- **Glue crawler + Athena** for schema detection and querying.
- **Redshift** as a data warehouse for high-performance BI analytics.

Architecture Diagram



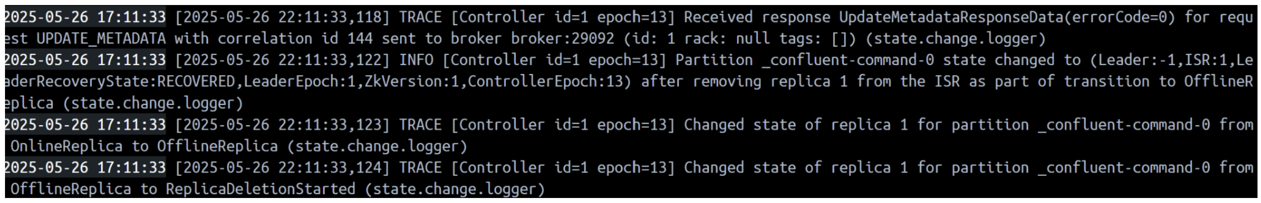
Tech Stack

Docker	Orchestrates Kafka, Spark, Zookeeper
Apache Kafka	Stream ingestion and message bus
Apache Spark	Real-time stream processing
AWS S3	Raw and enriched data storage
AWS Glue Crawler	Schema discovery for Athena
Glue Data Catalog	Central metadata store for table schemas and locations
AWS Athena	SQL-like queries over S3 data
AWS Glue Job	Scheduled ETL from S3 to Redshift
Amazon Redshift	Data warehouse for fast, cocurrent and advanced queries and ingestion into BI tools

End-to-End Data Flow Explanation Data Simulation:

I developed a Python-based data simulator that generates smart city data including vehicle, weather, emergency, traffic, and GPS readings. Each data type is streamed in real-time and sent to corresponding Kafka topics using Kafka Python producers.

Kafka Topic Setup: I created and configured five Kafka topics (vehicle_data, gps_data, weather_data, traffic_data, and emergency_data) inside a Docker-based Kafka-Zookeeper cluster. This setup ensures high-throughput, scalable ingestion for each data stream.

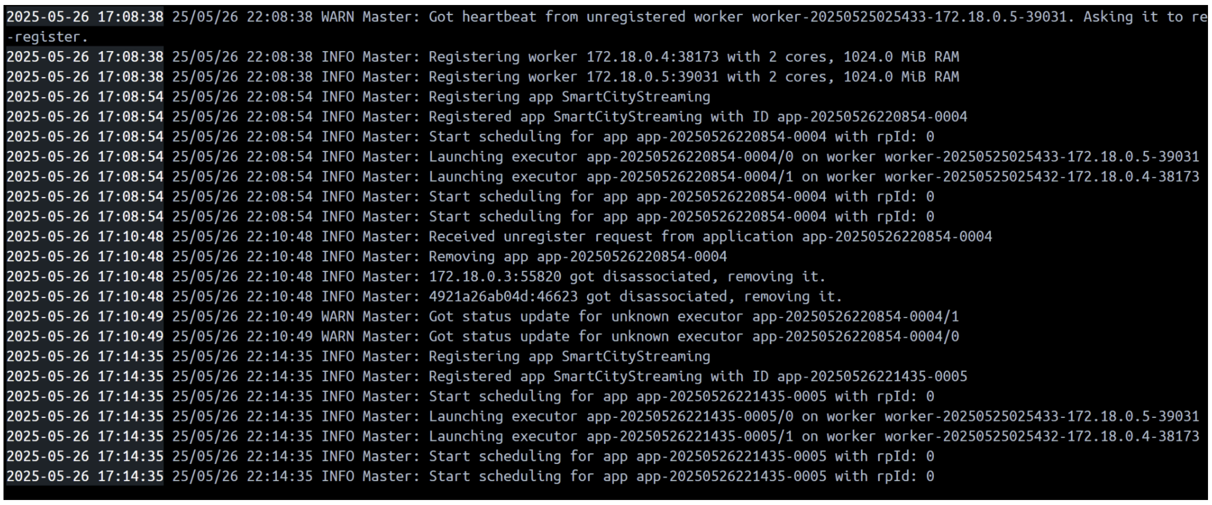


```
2025-05-26 17:11:33 [2025-05-26 22:11:33,118] TRACE [Controller id=1 epoch=13] Received response UpdateMetadataResponseData(errorCode=0) for requ
est UPDATE_METADATA with correlation id 144 sent to broker broker:29092 (id: 1 rack: null tags: []) (state.change.logger)
2025-05-26 17:11:33 [2025-05-26 22:11:33,122] INFO [Controller id=1 epoch=13] Partition _confluent-command-0 state changed to (Leader:-1,ISR:1,Le
aderRecoveryState:RECOVERED,LeaderEpoch:1,ZkVersion:1,ControllerEpoch:13) after removing replica 1 from the ISR as part of transition to OfflineR
eplica (state.change.logger)
2025-05-26 17:11:33 [2025-05-26 22:11:33,123] TRACE [Controller id=1 epoch=13] Changed state of replica 1 for partition _confluent-command-0 from
 OnlineReplica to OfflineReplica (state.change.logger)
2025-05-26 17:11:33 [2025-05-26 22:11:33,124] TRACE [Controller id=1 epoch=13] Changed state of replica 1 for partition _confluent-command-0 from
 OfflineReplica to ReplicaDeletionStarted (state.change.logger)
```

This image captures the moment when all Kafka topics are live and receiving data.

Spark Streaming Application (My Role): I used two Spark worker nodes, each with 2 cores, and implemented a PySpark Structured Streaming job that:

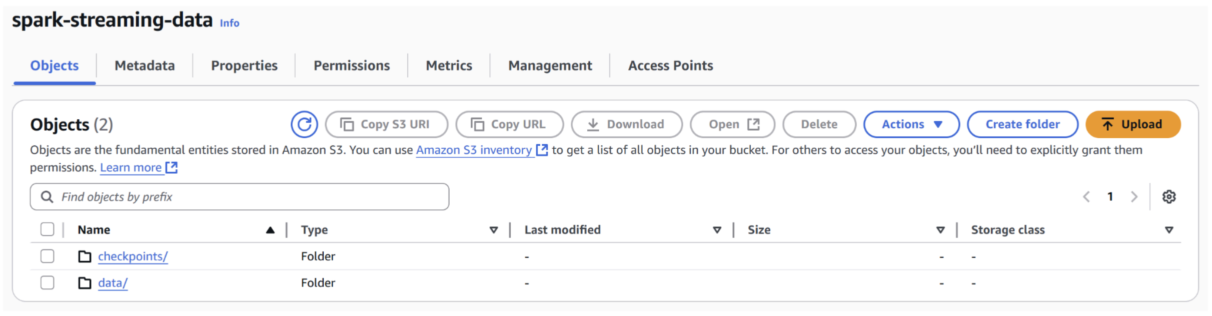
- Subscribes to all Kafka topics.
- Applies custom schemas to parse JSON payloads.
- Cleans location fields and extracts latitude and longitude.
- Uses with **Watermark** and performs stream joins between vehicle, weather, and emergency streams based on timestamp windows to tolerate late data.
- Writes both individual and joined streams to respective S3 locations in Parquet format using checkpointing to guarantee exactly-once semantics.



```
2025-05-26 17:08:38 25/05/26 22:08:38 WARN Master: Got heartbeat from unregistered worker worker-20250525025433-172.18.0.5-39031. Asking it to re
-register.
2025-05-26 17:08:38 25/05/26 22:08:38 INFO Master: Registering worker 172.18.0.4:38173 with 2 cores, 1024.0 MiB RAM
2025-05-26 17:08:38 25/05/26 22:08:38 INFO Master: Registering worker 172.18.0.5:39031 with 2 cores, 1024.0 MiB RAM
2025-05-26 17:08:54 25/05/26 22:08:54 INFO Master: Registering app SmartCityStreaming
2025-05-26 17:08:54 25/05/26 22:08:54 INFO Master: Registered app SmartCityStreaming with ID app-20250526220854-0004
2025-05-26 17:08:54 25/05/26 22:08:54 INFO Master: Start scheduling for app app-20250526220854-0004 with rpId: 0
2025-05-26 17:08:54 25/05/26 22:08:54 INFO Master: Launching executor app-20250526220854-0004/0 on worker worker-20250525025433-172.18.0.5-39031
2025-05-26 17:08:54 25/05/26 22:08:54 INFO Master: Launching executor app-20250526220854-0004/1 on worker worker-20250525025432-172.18.0.4-38173
2025-05-26 17:08:54 25/05/26 22:08:54 INFO Master: Start scheduling for app app-20250526220854-0004 with rpId: 0
2025-05-26 17:08:54 25/05/26 22:08:54 INFO Master: Start scheduling for app app-20250526220854-0004 with rpId: 0
2025-05-26 17:10:48 25/05/26 22:10:48 INFO Master: Received unregister request from application app-20250526220854-0004
2025-05-26 17:10:48 25/05/26 22:10:48 INFO Master: Removing app app-20250526220854-0004
2025-05-26 17:10:48 25/05/26 22:10:48 INFO Master: 172.18.0.3:55820 got disassociated, removing it.
2025-05-26 17:10:48 25/05/26 22:10:48 INFO Master: 4921a26ab04d:46623 got disassociated, removing it.
2025-05-26 17:10:49 25/05/26 22:10:49 WARN Master: Got status update for unknown executor app-20250526220854-0004/1
2025-05-26 17:10:49 25/05/26 22:10:49 WARN Master: Got status update for unknown executor app-20250526220854-0004/0
2025-05-26 17:14:35 25/05/26 22:14:35 INFO Master: Registering app SmartCityStreaming
2025-05-26 17:14:35 25/05/26 22:14:35 INFO Master: Registered app SmartCityStreaming with ID app-20250526221435-0005
2025-05-26 17:14:35 25/05/26 22:14:35 INFO Master: Start scheduling for app app-20250526221435-0005 with rpId: 0
2025-05-26 17:14:35 25/05/26 22:14:35 INFO Master: Launching executor app-20250526221435-0005/0 on worker worker-20250525025433-172.18.0.5-39031
2025-05-26 17:14:35 25/05/26 22:14:35 INFO Master: Launching executor app-20250526221435-0005/1 on worker worker-20250525025432-172.18.0.4-38173
2025-05-26 17:14:35 25/05/26 22:14:35 INFO Master: Start scheduling for app app-20250526221435-0005 with rpId: 0
2025-05-26 17:14:35 25/05/26 22:14:35 INFO Master: Start scheduling for app app-20250526221435-0005 with rpId: 0
```

Spark console log showing successful subscription and processing of Kafka streams
Here you can see Spark's successful launch and continuous reading from Kafka topics. The logs also confirm schema application, transformation steps, and real-time batch updates to the S3 destination.

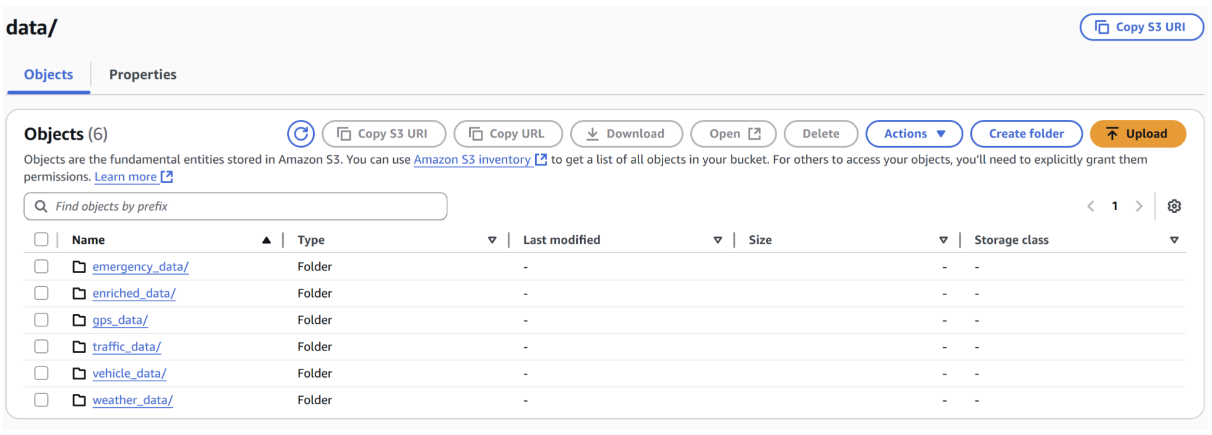
Data Lake Storage in S3: I configured Spark to write to AWS S3 in append mode with exactly-once processing via checkpointing. Data is structured under different folders: /vehicle_data, /weather_data, /emergency_data, and /enriched_data.



Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
checkpoints/	Folder	-	-	-
data/	Folder	-	-	-



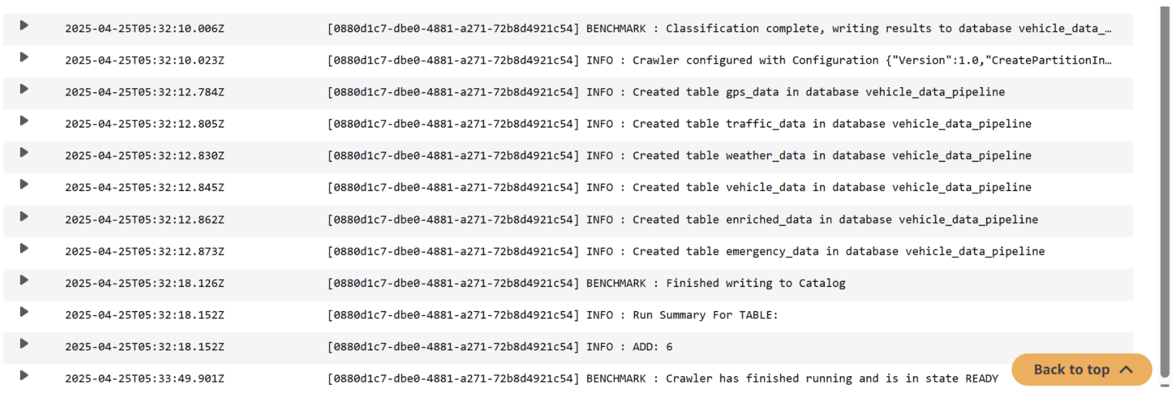
Objects (6)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
emergency_data/	Folder	-	-	-
enriched_data/	Folder	-	-	-
gps_data/	Folder	-	-	-
traffic_data/	Folder	-	-	-
vehicle_data/	Folder	-	-	-
weather_data/	Folder	-	-	-

This image shows the organized data in S3—each folder representing a Kafka topic like vehicle_data or weather_data, and one for joined enriched data. The files inside are stored in Parquet format for efficiency.

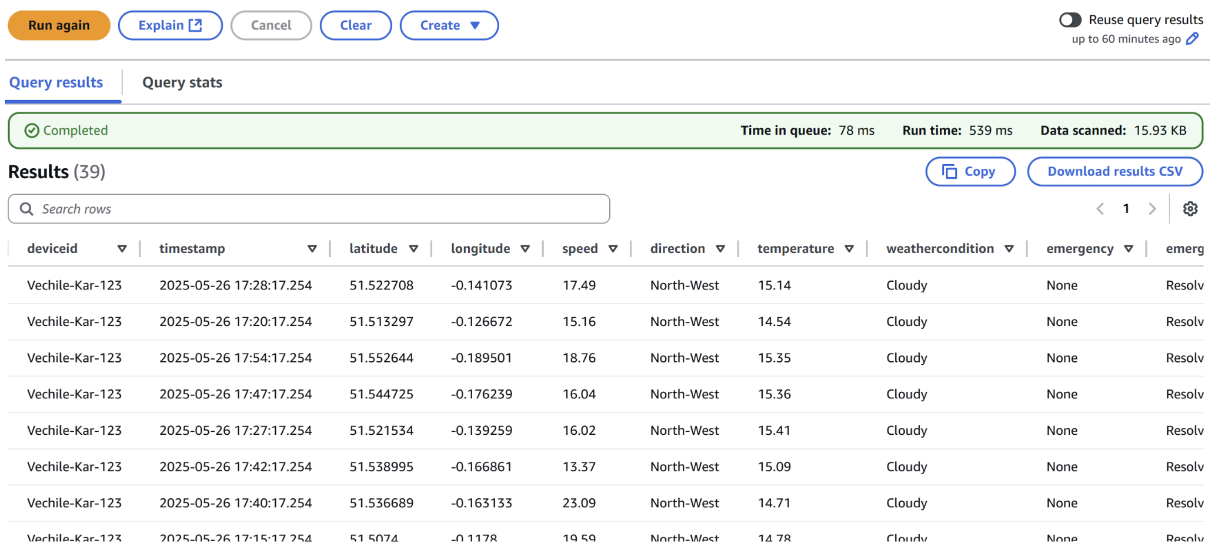
Glue Crawler: Configured a Glue Crawler to crawl S3 folders and infer schemas for all datasets. These schemas are added to the AWS Glue Data Catalog and are accessible from Athena.



```
2025-04-25T05:32:10.006Z    [0880d1c7-dbe0-4881-a271-72b8d4921c54] BENCHMARK : Classification complete, writing results to database vehicle_data_…
2025-04-25T05:32:10.023Z    [0880d1c7-dbe0-4881-a271-72b8d4921c54] INFO : Crawler configured with Configuration {"Version":1.0,"CreatePartitionIn…
2025-04-25T05:32:12.784Z    [0880d1c7-dbe0-4881-a271-72b8d4921c54] INFO : Created table gps_data in database vehicle_data_pipeline
2025-04-25T05:32:12.805Z    [0880d1c7-dbe0-4881-a271-72b8d4921c54] INFO : Created table traffic_data in database vehicle_data_pipeline
2025-04-25T05:32:12.830Z    [0880d1c7-dbe0-4881-a271-72b8d4921c54] INFO : Created table weather_data in database vehicle_data_pipeline
2025-04-25T05:32:12.845Z    [0880d1c7-dbe0-4881-a271-72b8d4921c54] INFO : Created table vehicle_data in database vehicle_data_pipeline
2025-04-25T05:32:12.862Z    [0880d1c7-dbe0-4881-a271-72b8d4921c54] INFO : Created table enriched_data in database vehicle_data_pipeline
2025-04-25T05:32:12.873Z    [0880d1c7-dbe0-4881-a271-72b8d4921c54] INFO : Created table emergency_data in database vehicle_data_pipeline
2025-04-25T05:32:18.126Z    [0880d1c7-dbe0-4881-a271-72b8d4921c54] BENCHMARK : Finished writing to Catalog
2025-04-25T05:32:18.152Z    [0880d1c7-dbe0-4881-a271-72b8d4921c54] INFO : Run Summary For TABLE:
2025-04-25T05:32:18.152Z    [0880d1c7-dbe0-4881-a271-72b8d4921c54] INFO : ADD: 6
2025-04-25T05:33:49.901Z    [0880d1c7-dbe0-4881-a271-72b8d4921c54] BENCHMARK : Crawler has finished running and is in state READY
```

This screenshot demonstrates the crawler's role in reading data from S3, automatically inferring schemas, and updating the AWS Glue Data Catalog. The resulting tables are available in Athena for SQL access.

Athena Query Layer: validated and queried the output using Athena, verifying that the Glue Crawler successfully detected and registered tables for all Parquet datasets in S3.



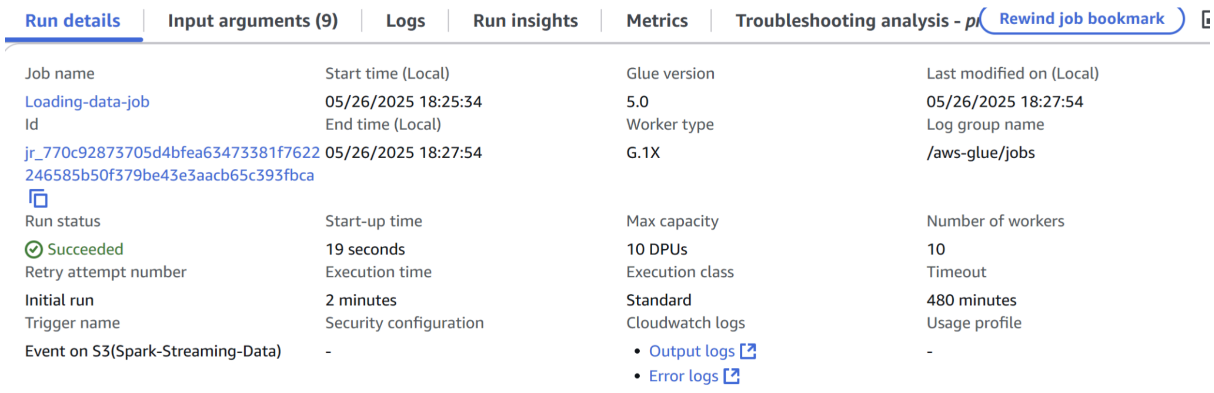
Completed Time in queue: 78 ms Run time: 539 ms Data scanned: 15.93 KB

Results (39)

deviceid	timestamp	latitude	longitude	speed	direction	temperature	weathercondition	emergency	emerg
Vechile-Kar-123	2025-05-26 17:28:17.254	51.522708	-0.141073	17.49	North-West	15.14	Cloudy	None	Resolv
Vechile-Kar-123	2025-05-26 17:20:17.254	51.513297	-0.126672	15.16	North-West	14.54	Cloudy	None	Resolv
Vechile-Kar-123	2025-05-26 17:54:17.254	51.552644	-0.189501	18.76	North-West	15.35	Cloudy	None	Resolv
Vechile-Kar-123	2025-05-26 17:47:17.254	51.544725	-0.176239	16.04	North-West	15.36	Cloudy	None	Resolv
Vechile-Kar-123	2025-05-26 17:27:17.254	51.521534	-0.139259	16.02	North-West	15.41	Cloudy	None	Resolv
Vechile-Kar-123	2025-05-26 17:42:17.254	51.538995	-0.166861	13.37	North-West	15.09	Cloudy	None	Resolv
Vechile-Kar-123	2025-05-26 17:40:17.254	51.536689	-0.163133	23.09	North-West	14.71	Cloudy	None	Resolv
Vechile-Kar-123	2025-05-26 17:15:17.254	51.5074	-0.1178	19.59	North-West	14.78	Cloudy	None	Resolv

This screenshot shows how Athena allows interactive querying on top of the S3 Parquet files, utilizing the schemas created by Glue Crawler. A sample SQL query and its results demonstrate how easy it is to explore data.

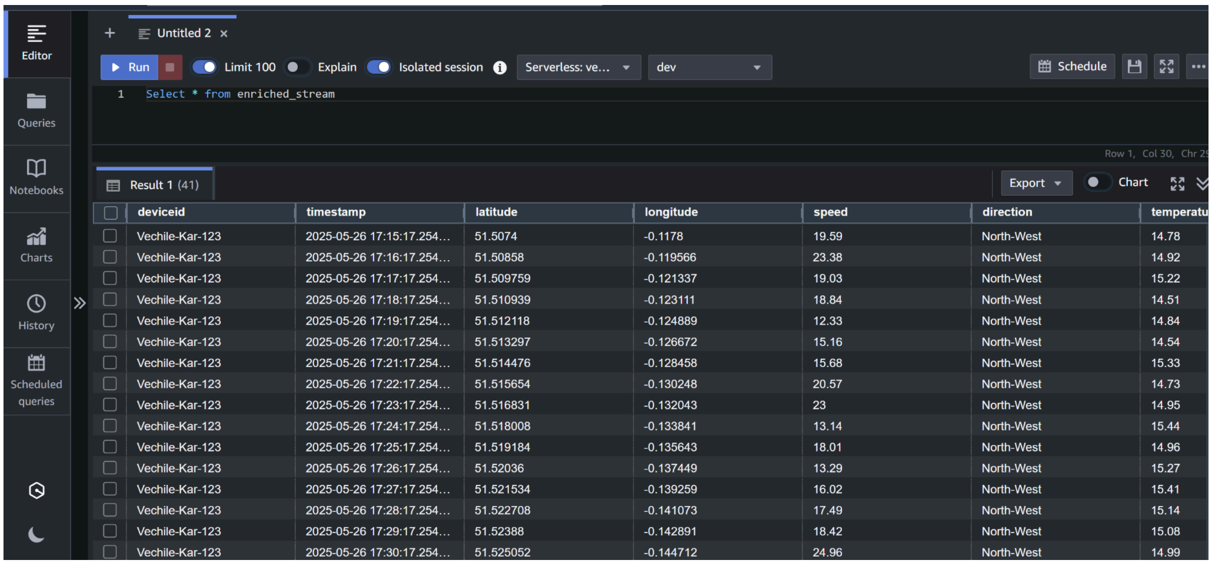
Glue Job & Redshift: Configured an event-driven AWS Glue Job that is triggered automatically whenever new enriched data is written to a specific S3 path. This job loads the enriched data into a Redshift data warehouse table. I ensured the S3-Redshift connection and schema mapping were correctly set up. Since Spark has already performed the stream processing and data transformation, the Glue job focuses solely on loading. However, if additional transformation is required in the future, this step is a suitable place to perform it before ingestion into Redshift.



Job name	Start time (Local)	Glue version	Last modified on (Local)
Loading-data-job	05/26/2025 18:25:34	5.0	05/26/2025 18:27:54
Id	End time (Local)	Worker type	Log group name
jr_770c92873705d4bfea63473381f7622246585b50f379be43e3aacb65c393fbca	05/26/2025 18:27:54	G.1X	/aws-glue/jobs
Run status	Start-up time	Max capacity	Number of workers
Succeeded	19 seconds	10 DPUs	10
Retry attempt number	Execution time	Execution class	Timeout
Initial run	2 minutes	Standard	480 minutes
Trigger name	Security configuration	Cloudwatch logs	Usage profile
Event on S3(Spark-Streaming-Data)	-	• Output logs • Error logs	-

Captured from the Glue console, this image confirms that the ETL job is active and regularly transforming data from S3 to Redshift. It ensures that the data warehouse remains updated and synchronized.

Redshift Query Layer: Confirmed the loaded data results in Redshift using SQL queries and sample data previews. Redshift serves as the final layer where data is ready for visualization and reporting.



```
Select * from enriched_stream
```

deviceid	timestamp	latitude	longitude	speed	direction	temperatu
Vechile-Kar-123	2025-05-26 17:15:17.254...	51.5074	-0.1178	19.59	North-West	14.78
Vechile-Kar-123	2025-05-26 17:16:17.254...	51.50858	-0.119566	23.38	North-West	14.92
Vechile-Kar-123	2025-05-26 17:17:17.254...	51.509759	-0.121337	19.03	North-West	15.22
Vechile-Kar-123	2025-05-26 17:18:17.254...	51.510939	-0.123111	18.84	North-West	14.51
Vechile-Kar-123	2025-05-26 17:19:17.254...	51.512118	-0.124889	12.33	North-West	14.84
Vechile-Kar-123	2025-05-26 17:20:17.254...	51.513297	-0.126672	15.16	North-West	14.54
Vechile-Kar-123	2025-05-26 17:21:17.254...	51.514476	-0.128458	15.68	North-West	15.33
Vechile-Kar-123	2025-05-26 17:22:17.254...	51.515654	-0.130248	20.57	North-West	14.73
Vechile-Kar-123	2025-05-26 17:23:17.254...	51.516831	-0.132043	23	North-West	14.95
Vechile-Kar-123	2025-05-26 17:24:17.254...	51.518008	-0.133841	13.14	North-West	15.44
Vechile-Kar-123	2025-05-26 17:25:17.254...	51.519184	-0.135643	18.01	North-West	14.96
Vechile-Kar-123	2025-05-26 17:26:17.254...	51.52036	-0.137449	13.29	North-West	15.27
Vechile-Kar-123	2025-05-26 17:27:17.254...	51.521534	-0.139259	16.02	North-West	15.41
Vechile-Kar-123	2025-05-26 17:28:17.254...	51.522708	-0.141073	17.49	North-West	15.14
Vechile-Kar-123	2025-05-26 17:29:17.254...	51.52388	-0.142891	18.42	North-West	15.08
Vechile-Kar-123	2025-05-26 17:30:17.254...	51.525052	-0.144712	24.96	North-West	14.99

This image displays the enriched data table inside Amazon Redshift. It serves as the final stop in the pipeline and enables fast queries and integration with BI tools such as Power BI or QuickSight.

Kafka Topics & Schemas

Topics Simulated:

- `vehicle_data`
- `gps_data`
- `weather_data`
- `Traffic_data`
- `emergency_data`

Spark Streaming & Data Processing

- Subscribes to all Kafka topics.
- Parses JSON and extracts fields.
- Cleans `location` field → splits into `latitude`, `longitude`.
- Joins `vehicle`, `weather`, and `Emergency` streams on exact timestamps.
- Selects enriched fields like `deviceId`, `speed`, `weatherCondition`, and `emergencyType`.
- Writes both raw and enriched data to **S3** in **Parquet format**.
- Can be able to send processed results to an Application, If we want a real time result.

Data Lake & Data Warehouse

S3 (Data Lake)

- Stores partitioned data by topic: `/data/vehicle_data`, `/data/weather_data`, etc.
- Stores enriched joined data under `/data/enriched_data`.

Athena + Glue

- Glue Crawler detects schema and builds external tables in Athena.
- Enables fast, serverless SQL queries directly on S3 data.

Redshift (Warehouse)

- AWS Glue Job runs every 5 minutes.
- Extracts new records from S3.
- Loads into pre-defined Redshift tables.

Automation & Scheduling

- Kafka producers generate data every few seconds.
- Spark streams are **triggered every 1 minute**.
- Glue Crawler runs periodically or on-demand.
- Configured an event-driven AWS Glue Job that is triggered automatically whenever new enriched data is written to a specific S3 path.

Glue Crawler & Glue Job Details

Glue Crawler

- Scans S3 data locations such as `/data/vehicle_data`, `/data/enriched_data`, etc.
- Automatically infers schema and updates the AWS Glue Data Catalog.
- Enables Athena to query Parquet files using standard SQL.

Glue Job

- A scheduled ETL job that runs every 5 minutes.
- Reads data from enriched Parquet files stored in S3.
- Transforms and loads the data into Amazon Redshift tables.
- Ensures that Redshift remains updated for real-time dashboarding and analytics.