

Project Documentation

Step 1: EC2 Instance and MySQL Setup

1. **EC2 Instance Creation:** An EC2 instance was initially created in the Tokyo region, requiring a new keypair and region-specific security groups.
2. **Security Group Configuration:** A new security group was set up with permissions for SSH, HTTP, HTTPS, and MySQL access to facilitate project operations.
3. **MySQL 5.6 Installation:**

Update yum packages:

```
sudo yum update -y
```

Install MySQL 5.7 community release:

```
sudo yum install -y https://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm
```

Import MySQL GPG key:

```
sudo rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022
```

Install MySQL community server:

```
sudo yum install -y mysql-community-server
```

Start MySQL service:

```
sudo systemctl start mysqld
```

Enable MySQL service to start on boot:

```
sudo systemctl enable mysqld
```

4. **MySQL Password Configuration:**

Retrieve temporary password:

```
sudo grep 'temporary password' /var/log/mysqld.log
```

Connect to MySQL:

```
mysql -u root -p
```

Change temporary password to Midhunproj@1:

```
ALTER USER 'root'@'localhost' IDENTIFIED BY 'Midhunproj@1';
```

```
LAST.COM
The authenticity of host 'ec2-18-183-62-248.ap-northeast-1.compute.amazonaws.com (18.183.62.248)' can't be established.
ED25519 key fingerprint is SHA256:doUr90AggNqAF5pTsao+0Gm8e6+aKo4QIg24TruGZ/0.
This host key is known by the following other names/addresses:
  C:\Users\midhun.krishna\.ssh\known_hosts:65: ec2-54-168-236-160.ap-northeast-1.compute.amazonaws.com
  C:\Users\midhun.krishna\.ssh\known_hosts:69: ec2-43-207-189-77.ap-northeast-1.compute.amazonaws.com
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-183-62-248.ap-northeast-1.compute.amazonaws.com' (ED25519) to the list of known hosts
.
      #_
  , \_ #####_      Amazon Linux 2023
  ~~ \#####\
  ~~  \###|
  ~~   \#/ ____ https://aws.amazon.com/linux/amazon-linux-2023
  ~~   V~' '-->
  ~~~   /
  ~~~. /_/
  ~~~/_/ _/
  ~~~/_m/_
Last login: Wed Jul 30 15:15:59 2025 from 49.36.41.141
[ec2-user@ip-172-31-7-251 ~]$ ls
```

Database and Table Creation in EC2 Instance

1. **Create Database:** create database midhun_proj;
2. **Select Database:** use midhun_proj;
3. **Create Dimension Tables:**
 - o futurecart_calendar_details

SQL

```
CREATE TABLE futurecart_calendar_details (
    calendar_date DATE,
    date_desc VARCHAR(50),
    week_day_nbr SMALLINT,
    week_number SMALLINT,
    week_name VARCHAR(50),
    year_week_number INT,
    month_number SMALLINT,
    month_name VARCHAR(50),
    quarter_number SMALLINT,
```

```
quarter_name VARCHAR(50),  
half_year_number SMALLINT,  
half_year_name VARCHAR(50),  
geo_region_cd CHAR(2)  
);
```

- o futurecart_call_center_details

SQL

```
CREATE TABLE futurecart_call_center_details (  
call_center_id VARCHAR(10),  
call_center_vendor VARCHAR(50),  
location VARCHAR(50),  
country VARCHAR(50)  
);
```

- o futurecart_case_category_details

SQL

```
CREATE TABLE futurecart_case_category_details (  
category_key VARCHAR(10),  
sub_category_key VARCHAR(10),  
category_description VARCHAR(50),  
sub_category_description VARCHAR(50),  
priority VARCHAR(10)  
);
```

- o futurecart_case_country_details

SQL

```
CREATE TABLE futurecart_case_country_details (  
id INT,
```

```
Name VARCHAR(75),  
Alpha_2 VARCHAR(2),  
Alpha_3 VARCHAR(3)  
);  
    o  futurecart_case_priority_details
```

SQL

```
CREATE TABLE futurecart_case_priority_details (  
Priority_key VARCHAR(5),  
priority VARCHAR(20),  
severity VARCHAR(100),  
SLA VARCHAR(100)
```

);

```
    o  futurecart_employee_details
```

SQL

```
CREATE TABLE futurecart_employee_details (  
emp_key INT,  
first_name VARCHAR(100),  
last_name VARCHAR(100),  
email VARCHAR(100),  
gender VARCHAR(10),  
ldap VARCHAR(50),  
hire_date DATE,  
manager VARCHAR(50)
```

);

```
    o  futurecart_product_details
```

SQL

```
CREATE TABLE futurecart_product_details (
product_id VARCHAR(50),
department VARCHAR(100),
brand VARCHAR(100),
commodity_desc VARCHAR(100),
sub_commodity_desc VARCHAR(100)
);
```

- o futurecart_survey_question_details

SQL

```
CREATE TABLE futurecart_survey_question_details (
question_id VARCHAR(50),
question_desc VARCHAR(755),
response_type VARCHAR(50),
`range` VARCHAR(50),
negative_response_range VARCHAR(50),
neutral_response_range VARCHAR(50),
positive_response_range VARCHAR(50)
);
```

(Note:range is a reserved keyword in MySQL, hence the backticks).

3.1 Loading Data into MySQL from S3

1. **Create MySQL files directory:** sudo mkdir -p /var/lib/mysql-files/
2. **Copy data files from S3 to EC2 instance:**
 - o aws s3 cp s3://midhun-tokyo/futurecart_calendar_details.txt /var/lib/mysql-files/
 - o aws s3 cp s3://midhun-tokyo/futurecart_call_center_details.txt /var/lib/mysql-files/

- aws s3 cp s3://midhun-tokyo/futurecart_case_category_details.txt /var/lib/mysql-files/
- aws s3 cp s3://midhun-tokyo/futurecart_case_country_details.txt /var/lib/mysql-files/
- aws s3 cp s3://midhun-tokyo/futurecart_case_priority_details.txt /var/lib/mysql-files/
- aws s3 cp s3://midhun-tokyo/futurecart_employee_details.txt /var/lib/mysql-files/
- aws s3 cp s3://midhun-tokyo/futurecart_product_details.txt /var/lib/mysql-files/
- aws s3 cp s3://midhun-tokyo/futurecart_survey_question_details.txt /var/lib/mysql-files/

3. Load data into MySQL tables:

- futurecart_calendar_details: LOAD DATA INFILE '/var/lib/mysql-files/futurecart_calendar_details.txt' INTO TABLE futurecart_calendar_details FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n' IGNORE 1 LINES;

```
Database changed
mysql> LOAD DATA INFILE '/var/lib/mysql-files/futurecart_calendar_details.txt'
      -> INTO TABLE futurecart_calendar_details
      -> FIELDS TERMINATED BY '\t'
      -> LINES TERMINATED BY '\n'
      -> IGNORE 1 LINES;
Query OK, 21553 rows affected (0.21 sec)
Records: 21553 Deleted: 0 Skipped: 0 Warnings: 0
```

- futurecart_call_center_details: LOAD DATA INFILE '/var/lib/mysql-files/futurecart_call_center_details.txt' INTO TABLE futurecart_call_center_details FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n' IGNORE 1 LINES;

```
mysql> LOAD DATA INFILE '/var/lib/mysql-files/futurecart_call_center_details.txt'
      -> INTO TABLE futurecart_call_center_details
      -> FIELDS TERMINATED BY '\t'
      -> LINES TERMINATED BY '\n'
      -> IGNORE 1 LINES;
Query OK, 25 rows affected (0.00 sec)
```

- futurecart_case_category_details: LOAD DATA INFILE '/var/lib/mysql-files/futurecart_case_category_details.txt' INTO TABLE

```
futurecart_case_category_details FIELDS TERMINATED BY '\t' LINES  
TERMINATED BY '\n' IGNORE 1 LINES;
```

```
mysql> LOAD DATA INFILE '/var/lib/mysql-files/futurecart_case_category_details.txt'  
-> INTO TABLE futurecart_case_category_details  
-> FIELDS TERMINATED BY '\t'  
-> LINES TERMINATED BY '\n'  
-> IGNORE 1 LINES;  
Query OK, 27 rows affected (0.00 sec)  
Records: 27 Deleted: 0 Skipped: 0 Warnings: 0
```

- futurecart_case_country_details: LOAD DATA INFILE '/var/lib/mysql-files/futurecart_case_country_details.txt' INTO TABLE futurecart_case_country_details FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n' IGNORE 1 LINES;

```
mysql> LOAD DATA INFILE '/var/lib/mysql-files/futurecart_case_country_details.txt'  
-> INTO TABLE futurecart_case_country_details  
-> FIELDS TERMINATED BY '\t'  
-> LINES TERMINATED BY '\n'  
-> IGNORE 1 LINES;  
Query OK, 193 rows affected (0.01 sec)  
Records: 193 Deleted: 0 Skipped: 0 Warnings: 0
```

- futurecart_case_priority_details: LOAD DATA INFILE '/var/lib/mysql-files/futurecart_case_priority_details.txt' INTO TABLE futurecart_case_priority_details FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n' IGNORE 1 LINES;

```
mysql> LOAD DATA INFILE '/var/lib/mysql-files/futurecart_case_priority_details.txt'  
-> INTO TABLE futurecart_case_priority_details  
-> FIELDS TERMINATED BY '\t'  
-> LINES TERMINATED BY '\n'  
-> IGNORE 1 LINES;  
Query OK, 25 rows affected (0.00 sec)  
Records: 25 Deleted: 0 Skipped: 0 Warnings: 0
```

- futurecart_employee_details: LOAD DATA INFILE '/var/lib/mysql-files/futurecart_employee_details.txt' INTO TABLE futurecart_employee_details FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n' IGNORE 1 LINES;

```
mysql> LOAD DATA INFILE '/var/lib/mysql-files/futurecart_employee_details.txt'
-> INTO TABLE futurecart_employee_details
-> FIELDS TERMINATED BY '\t'
-> LINES TERMINATED BY '\n'
-> IGNORE 1 LINES;
Query OK, 300024 rows affected (2.18 sec)
Records: 300024 Deleted: 0 Skipped: 0 Warnings: 0
```

- futurecart_product_details: LOAD DATA INFILE '/var/lib/mysql-files/futurecart_product_details.txt' INTO TABLE futurecart_product_details FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n' IGNORE 1 LINES;

```
mysql> LOAD DATA INFILE '/var/lib/mysql-files/futurecart_product_details.txt'
-> INTO TABLE futurecart_product_details
-> FIELDS TERMINATED BY '\t'
-> LINES TERMINATED BY '\n'
-> IGNORE 1 LINES;
Query OK, 92353 rows affected (0.45 sec)
Records: 92353 Deleted: 0 Skipped: 0 Warnings: 0
```

- futurecart_survey_question_details: LOAD DATA INFILE '/var/lib/mysql-files/futurecart_survey_question_details.txt' INTO TABLE futurecart_survey_question_details FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n' IGNORE 1 LINES;

```
mysql> LOAD DATA INFILE '/var/lib/mysql-files/futurecart_product_details.txt'
-> INTO TABLE futurecart_product_details
-> FIELDS TERMINATED BY '\t'
-> LINES TERMINATED BY '\n'
-> IGNORE 1 LINES;
Query OK, 92353 rows affected (0.45 sec)
Records: 92353 Deleted: 0 Skipped: 0 Warnings: 0
```

3.2 EMR Cluster Setup and Sqoop Import to Hive

1. **Create EMR Cluster:** An EMR cluster was created with Hadoop, Sqoop, Spark, and Hive.
2. **Grant MySQL Privileges to EMR Cluster:**
 - GRANT ALL PRIVILEGES ON ** TO 'root'@'%' IDENTIFIED BY 'Midhunproj@1';
 - Modified MySQL config file to set bind-address to 0.0.0.0 to allow external connections.

- Restart MySQL service: sudo systemctl restart mysql
- Verify MySQL is listening on 0.0.0.0:3306: sudo netstat -tulnp | grep mysqld
- Connect to MySQL from EMR cluster: mysql -h 172.31.7.251 -u root -p

```
41 package(s) needed for security, out of 42 available
Run "sudo yum update" to apply all updates.
```

```
EEEEEEEEEEEEEEEEEE MMMMMMM          MMMMMMM RRRRRRRRRRRRRRR
E:::::::::::::E M::::::M           M:::::M R:::::::::::R
EE:::::EEEEEEE:::E M::::::M       M:::::M R::::RRRRR:::::R
   E:::E     EEEE M::::::M       M:::::M RR:::R      R:::R
   E:::E     M:::::M:::M   M:::M::::M   R:::R      R:::R
   E:::::EEEEEEEEE M:::::M M:::M M::::M R::::RRRRR:::::R
   E:::::::::::E M:::::M M:::M::::M M:::::M R:::::::::::R
   E:::::EEEEEEEEE M:::::M M:::::M M:::::M R::::RRRRR:::::R
   E:::::E     M:::::M M:::::M M:::::M R::::R      R:::R
   E:::E     EEEE M:::::M   MMM  M:::::M R::::R      R:::R
EE:::::EEEEEEE:::E M:::::M       M:::::M R::::R      R:::R
E:::::::::::::E M:::::M           M:::::M RR:::::R      R:::R
EEEEEEEEEEEEEEEEEE MMMMMMM          MMMMM RRRRRRR RRRRRR
```

```
[ec2-user@ip-172-31-41-223 ~]$ nano script.py
[ec2-user@ip-172-31-41-223 ~]$ rm script.py
[ec2-user@ip-172-31-41-223 ~]$ nano script.py
[ec2-user@ip-172-31-41-223 ~]$ spark-submit \
>   --packages "com.amazon.redshift:redshift-jdbc42:2.1.0.9" \
>   script.py
::: loading settings :: url = jar:file:/usr/lib/spark/jars/ivy-2.5.1.jar!/org/
Ivy Default Cache set to: /home/ec2-user/.ivy2/cache
The jars for the packages stored in: /home/ec2-user/.ivy2/jars
com.amazon.redshift#redshift-jdbc42 added as a dependency
... resolving dependencies ... org.apache.spark#spark-submit-parent-89072d19-5f
```

```
[ec2-user@ip-172-31-47-151 ~]$ mysql -h 172.31.7.251 -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.44 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> |
```

3. Install JDBC Driver for Sqoop:

- Download JDBC driver: wget
<https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.49.tar.gz>
- Extract archive: tar -xvzf mysql-connector-java-5.1.49.tar.gz
- Copy JAR to Sqoop lib directory: sudo cp mysql-connector-java-5.1.49/mysql-connector-java-5.1.49-bin.jar /usr/lib/sqoop/lib/

4. HDFS Permissions for EC2 user:

- Create user directory: sudo -u hdfs hdfs dfs -mkdir /user/ec2-user
- Change ownership: sudo -u hdfs hdfs dfs -chown ec2-user:ec2-user /user/ec2-user

5. Sqoop Import Data to Hive Tables:

- futurecart_calendar_details: sqoop import --connect jdbc:mysql://172.31.7.251:3306/midhun_proj --username root --password Midhunproj@1 --table futurecart_calendar_details --hive-import --hive-table futurecart_calendar_details --create-hive-table --m 1
- futurecart_call_center_details: sqoop import --connect jdbc:mysql://172.31.7.251:3306/midhun_proj --username root --password Midhunproj@1 --table futurecart_call_center_details --hive-import --hive-table futurecart_call_center_details --create-hive-table --m 1
- futurecart_case_category_details: sqoop import --connect jdbc:mysql://172.31.7.251:3306/midhun_proj --username root --password Midhunproj@1 --table futurecart_case_category_details --hive-import --hive-table futurecart_case_category_details --create-hive-table --m 1
- futurecart_case_country_details: sqoop import --connect jdbc:mysql://172.31.7.251:3306/midhun_proj --username root --password Midhunproj@1 --table futurecart_case_country_details --hive-import --hive-table futurecart_case_country_details --create-hive-table --m 1
- futurecart_case_priority_details: sqoop import --connect jdbc:mysql://172.31.7.251:3306/midhun_proj --username root --password Midhunproj@1 --table futurecart_case_priority_details --hive-import --hive-table futurecart_case_priority_details --create-hive-table --m 1

- futurecart_employee_details: sqoop import --connect jdbc:mysql://172.31.7.251:3306/midhun_proj --username root --password Midhunproj@1 --table futurecart_employee_details --hive-import --hive-table futurecart_employee_details --create-hive-table --m 1
- futurecart_product_details: sqoop import --connect jdbc:mysql://172.31.7.251:3306/midhun_proj --username root --password Midhunproj@1 --table futurecart_product_details --hive-import --hive-table futurecart_product_details --create-hive-table --m 1
- futurecart_survey_question_details: sqoop import --connect jdbc:mysql://172.31.7.251:3306/midhun_proj --username root --password Midhunproj@1 --table futurecart_survey_question_details --hive-import --hive-table futurecart_survey_question_details --create-hive-table --m 1

```
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: true
hive> show tables;
OK
futurecart_calendar_details
futurecart_call_center_details
futurecart_case_category_details
futurecart_case_country_details
futurecart_case_priority_details
futurecart_employee_details
futurecart_product_details
futurecart_survey_question_details
Time taken: 0.67 seconds, Fetched: 8 row(s)
hive>
```

Uploading Dimension Data to S3 using Spark

1. Create Python script (`dimension_export.py`):

Python

```
import sys

from pyspark.sql import SparkSession

from pyspark.sql.utils import AnalysisException
```

```
dimension_tables = [
    "futurecart_calendar_details",
    "futurecart_call_center_details",
    "futurecart_case_category_details",
```

```
"futurecart_case_country_details",
"futurecart_case_priority_details",
"futurecart_employee_details",
"futurecart_product_details",
"futurecart_survey_question_details"
]
```

```
s3_output_base_path = "s3a://midhun-tokyo/dimension_data_json/"
```

```
def process_table(table_name, spark, s3_path):
    try:
        print(f"Processing table: {table_name}")
        df = spark.sql(f"SELECT * FROM {table_name}")
        output_path = f"{s3_path}{table_name}.json"
        df.write.mode("overwrite").json(output_path)
        print(f"Successfully wrote data from {table_name} to {output_path}")
    except AnalysisException as e:
        print(f"Error processing table {table_name}: {e}")
        print("Please check if the Hive table exists and if the Spark session has access to it.")
    except Exception as e:
        print(f"An unexpected error occurred while processing table {table_name}: {e}")
```

```
if __name__ == "__main__":
    spark = SparkSession.builder \
        .appName("Hive to S3 Dimension Data Export") \
        .enableHiveSupport()
```

```
.getOrCreate()

for table in dimension_tables:
    process_table(table, spark, s3_output_base_path)
```

```
spark.stop()
print("All tables have been processed. Spark session stopped.")
```

2. Run Spark job: spark-submit dimension_export.py

<input type="checkbox"/>	Name	Type	Last modified
<input type="checkbox"/>	futurecart_calendar_details.json/	Folder	-
<input type="checkbox"/>	futurecart_call_center_details.json/	Folder	-
<input type="checkbox"/>	futurecart_case_category_details.json/	Folder	-
<input type="checkbox"/>	futurecart_case_country_details.json/	Folder	-

3.4 Generating and Uploading Historical Data to HDFS

1. **Fetch historical data generation script:** aws s3 cp s3://midhun-tokyo/generate_historical_data.py ~/generate_historical_data.py
2. **Run script:** python3 generate_historical_data.py
3. **Upload historical data (cases) to HDFS:**
 - o Create HDFS directory: sudo -u hdfs hadoop fs -mkdir -p /user/hdfs/historical_data_jsonl/cases

- Copy local files to /tmp: sudo cp -r /home/ec2-user/historical_data_jsonl/cases /tmp/
- Change ownership to hdfs: sudo chown -R hdfs:hdfs /tmp/cases
- Put files into HDFS: sudo -u hdfs hadoop fs -put /tmp/cases/* /user/hdfs/historical_data_jsonl/cases/

```
[ec2-user@ip-172-31-47-151 ~]$ sudo -u hdfs hadoop fs -mkdir -p /user/hdfs/historical_data_jsonl/surveys
[ec2-user@ip-172-31-47-151 ~]$ sudo cp -r /home/ec2-user/historical_data_jsonl/surveys /tmp/
[ec2-user@ip-172-31-47-151 ~]$ sudo chown -R hdfs:hdfs /tmp/surveys
[ec2-user@ip-172-31-47-151 ~]$ sudo -u hdfs hadoop fs -put /tmp/surveys/* /user/hdfs/historical_data_jsonl/surveys/
```

4. Upload historical data (surveys) to HDFS:

- Create HDFS directory: sudo -u hdfs hadoop fs -mkdir -p /user/hdfs/historical_data_jsonl/surveys
- Copy local files to /tmp: sudo cp -r /home/ec2-user/historical_data_jsonl/surveys /tmp/
- Change ownership to hdfs: sudo chown -R hdfs:hdfs /tmp/surveys
- Put files into HDFS: sudo -u hdfs hadoop fs -put /tmp/surveys/* /user/hdfs/historical_data_jsonl/surveys/

3.5 Creating External Hive Tables for Historical Data

1. futurecart_case_details:

SQL

```
CREATE EXTERNAL TABLE IF NOT EXISTS futurecart_case_details (
    case_no varchar(700),
    create_timestamp varchar(700),
    last_modified_timestamp varchar(700),
    created_employee_key varchar(700),
    call_center_id varchar(700),
    status varchar(700),
    category varchar(700),
    sub_category varchar(700),
    communication_mode varchar(700),
```

```

country_cd varchar(700),
product_code varchar(700)
)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
STORED AS TEXTFILE
LOCATION '/user/hdfs/historical_data_jsonl/cases/';

```

2. **futurecart_case_survey_details:**

SQL

```

CREATE EXTERNAL TABLE IF NOT EXISTS futurecart_case_survey_details (
survey_id varchar(700),
case_no varchar(700),
survey_timestamp varchar(700),
Q1 varchar(700),
Q2 varchar(700),
Q3 varchar(700),
Q4 varchar(700),
Q5 varchar(700)
)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
LOCATION '/user/hdfs/historical_data_jsonl/surveys';

```

```

25/07/30 06:31:48 INFO MemoryStore: MemoryStore cleared
25/07/30 06:31:48 INFO BlockManager: BlockManager stopped
25/07/30 06:31:48 INFO BlockManagerMaster: BlockManagerMaster stopped
25/07/30 06:31:48 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stop
25/07/30 06:31:49 INFO SparkContext: Successfully stopped SparkContext
All historical tables have been processed. Spark session stopped.
25/07/30 06:31:49 INFO ShutdownHookManager: Shutdown hook called
25/07/30 06:31:49 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-575db8ba-5abd-4fe6-a12f-4996df
25/07/30 06:31:49 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-191600ab-366a-4596-ae75-8c43e6
25/07/30 06:31:49 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-191600ab-366a-4596-ae75-8c43e6
da9ff8ab97c
25/07/30 06:31:51 INFO MetricsSystemImpl: Stopping s3a-file-system metrics system...
25/07/30 06:31:51 INFO MetricsSystemImpl: s3a-file-system metrics system stopped.
25/07/30 06:31:51 INFO MetricsSystemImpl: s3a-file-system metrics system shutdown complete.

```

3.6 Uploading Historical Data from Hive to S3 using Spark

1. Create Python script (bucket_export.py):

Python

```
import sys
from pyspark.sql import SparkSession
from pyspark.sql.utils import AnalysisException

historical_tables = [
    "futurecart_case_details",
    "futurecart_case_survey_details"
]

s3_output_base_path = "s3a://midhun-tokyo/historical_data_json_export/"

def process_table(table_name, spark, s3_path):
    try:
        print(f"Processing historical table: {table_name}")
        df = spark.sql(f"SELECT * FROM {table_name}")
        output_path = f"{s3_path}{table_name}/"
        df.write.mode("overwrite").json(output_path)
        print(f"Successfully wrote data from {table_name} to {output_path}")
    except AnalysisException as e:
        print(f"Error processing table {table_name}: {e}")
        print("Please check if the Hive table exists and if the Spark session has access to it.")
    except Exception as e:
        print(f"An unexpected error occurred while processing table {table_name}: {e}")
```

```

if __name__ == "__main__":
    spark = SparkSession.builder \
        .appName("Historical Hive to S3 JSON Export") \
        .enableHiveSupport() \
        .getOrCreate()

    for table in historical_tables:
        process_table(table, spark, s3_output_base_path)

    spark.stop()
    print("All historical tables have been processed. Spark session stopped.")

```

2. Download Hive HCatalog Core JAR: wget

<https://repo1.maven.org/maven2/org/apache/hive/hcatalog/hive-hcatalog-core/3.1.3/hive-hcatalog-core-3.1.3.jar>

3. Run Spark job with JAR: spark-submit --jars /home/ec2-user/hive-hcatalog-core-3.1.3.jar bucket_export.py

	Name	Type	Last Modified
<input type="checkbox"/>	futurecart_case_details/	Folder	-
<input type="checkbox"/>	futurecart_case_survey_details/	Folder	-

Step 4: Redshift Setup and Data Loading

1. Create Tables in Redshift: 10 tables were created in Redshift.

2. Load Data from S3 to Redshift:

SQL

```
COPY midhun_proj.public.futurecart_product_details FROM 's3://midhun-tokyo/dimension_data_json/futurecart_product_details.json/'
```

```
IAM_ROLE 'arn:aws:iam::008673239246:role/Midhun-Redshift-s3fullaccess'
```

```
JSON 'auto';
```

- Loading reported successful for futurecart_case_details (1000 records) and futurecart_case_survey_details (800 records).

A screenshot of the AWS Redshift console. The left sidebar shows a tree structure with 'midhun_proj' expanded, revealing 'public' and 'Tables'. The 'Tables' node is selected, highlighted in blue. To its right, the number '10' indicates the count of tables. Below this, a list of ten tables is displayed, each represented by a small icon and a truncated table name.

A screenshot of the AWS Redshift console showing the results of a recent load operation. At the top, there are two tabs: 'Result 1' and 'Result 2', with 'Result 1' being the active tab. Below the tabs, the word 'Summary' is displayed. In the bottom right corner of the summary area, there is an 'Info' message with a small info icon, stating: "Load into table 'futurecart_case_details' completed, 1000 record(s) loaded successfully."

Result 1 Result 2

Summary

Info:

- Load into table 'futurecart_case_survey_details' completed, 800 record(s) loaded successfully.

Step 5: Real-Time Data Generation and Kinesis Integration

- Real-Time Data Generation to Kinesis:** A script was run on an EC2 instance to send real-time data to a Kinesis data stream. The region and Kinesis data stream name were configured in the Python file.

```

ory': 'SCAT10', 'communication_mode': 'Email', 'country_cd': 'TG', 'product_code': '15452140', 'last_modified_timestamp': '2025-07-29 15:24:09', 'cr
eate_timestamp': '2025-07-29 15:24:09'}
Sent to Kinesis: {'case_no': '600939', 'created_employee_key': '487044', 'call_center_id': 'C-102', 'status': 'Open', 'category': 'CAT3', 'sub_categ
ory': 'SCAT13', 'communication_mode': 'Email', 'country_cd': 'NE', 'product_code': '10355849', 'last_modified_timestamp': '2025-07-29 15:34:14', 'cr
eate_timestamp': '2025-07-29 15:34:14'}
Sent to Kinesis: {'case_no': '600940', 'created_employee_key': '240778', 'call_center_id': 'C-113', 'status': 'Open', 'category': 'CAT3', 'sub_categ
ory': 'SCAT13', 'communication_mode': 'Call', 'country_cd': 'BB', 'product_code': '10313101', 'last_modified_timestamp': '2025-07-29 15:34:14', 'cre
ate_timestamp': '2025-07-29 15:34:14'}
Sent to Kinesis: {'case_no': '600941', 'created_employee_key': '67402', 'call_center_id': 'C-121', 'status': 'Closed', 'category': 'CAT3', 'sub_cate
gory': 'SCAT8', 'communication_mode': 'Email', 'country_cd': 'BE', 'product_code': '9878805', 'last_modified_timestamp': '2025-07-29 16:04:19', 'cre
ate_timestamp': '2025-07-29 15:34:19'}
Sent survey to Kinesis: {'survey_id': 'S-500007', 'case_no': '600941', 'survey_timestamp': '2025-07-29 16:14:19', 'Q1': 7, 'Q2': 4, 'Q3': 2, 'Q4': 'Y',
'Q5': 2}
Sent to Kinesis: {'case_no': '600942', 'created_employee_key': '487425', 'call_center_id': 'C-102', 'status': 'Open', 'category': 'CAT3', 'sub_categ
ory': 'SCAT15', 'communication_mode': 'Chat', 'country_cd': 'DE', 'product_code': '8380016', 'last_modified_timestamp': '2025-07-29 15:24:24', 'cre
ate_timestamp': '2025-07-29 15:24:24'}
Sent to Kinesis: {'case_no': '600943', 'created_employee_key': '452299', 'call_center_id': 'C-123', 'status': 'Open', 'category': 'CAT3', 'sub_categ
ory': 'SCAT15', 'communication_mode': 'Chat', 'country_cd': 'EH', 'product_code': '1126253', 'last_modified_timestamp': '2025-07-29 15:24:24', 'cre
ate_timestamp': '2025-07-29 15:24:24'}
Sent to Kinesis: {'case_no': '600944', 'created_employee_key': '247659', 'call_center_id': 'C-119', 'status': 'Open', 'category': 'CAT3', 'sub_categ
ory': 'SCAT15', 'communication_mode': 'Email', 'country_cd': 'FO', 'product_code': '8065238', 'last_modified_timestamp': '2025-07-29 15:24:24', 'cre
ate_timestamp': '2025-07-29 15:24:24'}
Sent to Kinesis: {'case_no': '600945', 'created_employee_key': '471602', 'call_center_id': 'C-106', 'status': 'Open', 'category': 'CAT3', 'sub_categ
ory': 'SCAT15', 'communication_mode': 'Email', 'country_cd': 'CX', 'product_code': '17215079', 'last_modified_timestamp': '2025-07-29 15:24:24', 'cr
eate_timestamp': '2025-07-29 15:24:24'}
Sent to Kinesis: {'case_no': '600946', 'created_employee_key': '290601', 'call_center_id': 'C-117', 'status': 'Open', 'category': 'CAT3', 'sub_categ
ory': 'SCAT15', 'communication_mode': 'Email', 'country_cd': 'SI', 'product_code': '12648192', 'last_modified_timestamp': '2025-07-29 15:24:24', 'cr
eate_timestamp': '2025-07-29 15:24:24'}
Sent to Kinesis: {'case_no': '600947', 'created_employee_key': '273941', 'call_center_id': 'C-124', 'status': 'Open', 'category': 'CAT3', 'sub_categ
ory': 'SCAT9', 'communication_mode': 'Chat', 'country_cd': 'AS', 'product_code': '1098074', 'last_modified_timestamp': '2025-07-29 15:34:29', 'creat
e_timestamp': '2025-07-29 15:34:29'}
Sent to Kinesis: {'case_no': '600948', 'created_employee_key': '34874', 'call_center_id': 'C-109', 'status': 'Open', 'category': 'CAT3', 'sub_categ
ory': 'SCAT9', 'communication_mode': 'Email', 'country_cd': 'TO', 'product_code': '880177', 'last_modified_timestamp': '2025-07-29 15:34:29', 'create
_timestamp': '2025-07-29 15:34:29'}
|
```

- Spark Script for Kinesis to Redshift:** A Spark script was used to capture data from Kinesis and load it into Redshift.

Python code:

```

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, from_json, to_timestamp, when, lit

from pyspark.sql.types import StructType, StructField, StringType, IntegerType

```

```
import os
import shutil
import boto3
import json
import time
import threading
from queue import Queue
import traceback

def start_spark_session():
    """
    Starts and gives back a SparkSession set up for handling data streams.
    It makes Spark run faster for big data tasks.
    """

    return SparkSession.builder \
        .appName("LiveDataProcessor") \
        .config("spark.jars.packages", "com.amazon.redshift:redshift-jdbc42:2.1.0.9") \
        .config("spark.sql.adaptive.enabled", "true") \
        .config("spark.sql.adaptive.coalescePartitions.enabled", "true") \
        .config("spark.serializer", "org.apache.spark.serializer.KryoSerializer") \
        .getOrCreate()

def define_data_structures():
    """
    Sets up the exact layouts (blueprints) for two types of incoming data:
    customer calls and customer survey answers.
    """
```

These blueprints help Spark understand the data correctly.

"""

```
customer_call_data_structure = StructType([
    StructField("call_id", StringType(), True),
    StructField("agent_id", StringType(), True),
    StructField("service_center_id", StringType(), True),
    StructField("status", StringType(), True),
    StructField("category", StringType(), True),
    StructField("sub_category", StringType(), True),
    StructField("talk_channel", StringType(), True),
    StructField("country_code", StringType(), True),
    StructField("product_code", StringType(), True),
    StructField("last_change_time", StringType(), True),
    StructField("create_time", StringType(), True)
])
```

```
customer_survey_data_structure = StructType([
    StructField("survey_id", StringType(), True),
    StructField("call_id", StringType(), True),
    StructField("survey_time", StringType(), True),
    StructField("score_q1", IntegerType(), True),
    StructField("score_q2", IntegerType(), True),
    StructField("score_q3", IntegerType(), True),
    StructField("comment_q4", StringType(), True),
    StructField("score_q5", IntegerType(), True)
])
```

])

```
return customer_call_data_structure, customer_survey_data_structure
```

class KinesisDataFetcher:

"""

Handles getting data from an AWS Kinesis stream safely, even with multiple tasks at once.

It keeps pulling new data and puts it into a waiting list (queue) for processing.

"""

```
def __init__(self, stream_name, aws_region):
```

"""

Sets up the KinesisDataFetcher with stream details and connects to Kinesis.

Args:

stream_name (str): The name of the Kinesis stream to get data from.

aws_region (str): The AWS region where the Kinesis stream is.

"""

```
self.stream_name = stream_name
```

```
self.aws_region = aws_region
```

```
self.kinesis_client = boto3.client('kinesis', region_name=aws_region)
```

```
self.data_storage_queue = Queue()
```

```
self.is_running = False
```

```
self.shard_cursors = {} # Keeps track of where we left off in each Kinesis shard
```

```
def start_fetching(self):
    """
    Starts the background task that continuously gets data from Kinesis.

    """
    self.is_running = True
    fetch_thread = threading.Thread(target=self._data_fetch_loop)
    fetch_thread.daemon = True # Lets the main program close even if this task is
    still running
    fetch_thread.start()
    return fetch_thread
```

```
def stop_fetching(self):
    """
    Tells the data fetching task to stop gently.

    """
    self.is_running = False
```

```
def get_stored_data_batch(self, max_items=100):
    """
    Gets a group of records from the internal data waiting list.
```

Args:

max_items (int): The most records to get in one group.

Returns:

list: A list of data items, each being a Kinesis record.

```
....  
  
batch_of_data = []  
  
while not self.data_storage_queue.empty() and len(batch_of_data) < max_items:  
    batch_of_data.append(self.data_storage_queue.get())  
  
return batch_of_data
```

```
def _data_fetch_loop(self):
```

```
....
```

The main loop for the background data fetching task. It keeps getting records from Kinesis sections (shards) and adds them to the queue.

```
....
```

```
print("⌚ Starting Kinesis data fetching task...")
```

```
while self.is_running:
```

```
try:
```

```
    # Get stream details to find active sections
```

```
    stream_info =
```

```
    self.kinesis_client.describe_stream(StreamName=self.stream_name)
```

```
    shards = stream_info['StreamDescription']['Shards']
```

```
    # Go through each section to get records
```

```
    for shard in shards: # Checking all sections
```

```
        shard_id = shard['ShardId']
```

```
        # Get or refresh the section marker (cursor)
```

```
        if shard_id not in self.shard_cursors:
```

```
response = self.kinesis_client.get_shard_iterator(
    StreamName=self.stream_name,
    ShardId=shard_id,
    ShardIteratorType='LATEST' # Start from the newest records
)
self.shard_cursors[shard_id] = response['ShardIterator']

# Get records using the current section marker
try:
    response = self.kinesis_client.get_records(
        ShardIterator=self.shard_cursors[shard_id],
        Limit=50 # Max records to get per try
    )

    records = response.get('Records', [])
    if records:
        print(f"📦 Got {len(records)} records from section {shard_id}")

    for record in records:
        try:
            raw_record_data = record['Data']
            if isinstance(raw_record_data, bytes):
                raw_record_data = raw_record_data.decode('utf-8')

            # Save important info along with the main data
            processed_record = {
```

```

        "json_data": str(raw_record_data),
        "partitionKey": record.get('PartitionKey', ''),
        "sequenceNumber": record.get('SequenceNumber', ''),
        "approximateArrivalTimestamp":
            str(record.get('ApproximateArrivalTimestamp', ''))

    }

    self.data_storage_queue.put(processed_record)

except Exception as e:
    print(f"⚠️ Problem handling one record: {str(e)}")

# Update the section marker for the next fetch

if response.get('NextShardIterator'):

    self.shard_cursors[shard_id] = response['NextShardIterator']

else:

    # If the next marker is empty, the current one has expired.

    # Get a new one starting from the latest point.

    response = self.kinesis_client.get_shard_iterator(
        StreamName=self.stream_name,
        ShardId=shard_id,
        ShardIteratorType='LATEST'
    )

    self.shard_cursors[shard_id] = response['ShardIterator']

except Exception as e:
    print(f"⚠️ Problem reading records from section {shard_id}: {str(e)}")

```

```
# Wait a bit before getting the next round of data
time.sleep(5)

except Exception as e:
    print(f"⚠ Problem during Kinesis data fetching loop: {str(e)}")
    time.sleep(10) # Wait longer if there's a big error
```

```
def sort_and_understand_data(raw_records, call_data_structure,
                             survey_data_structure):
```

```
    """
```

Takes raw Kinesis records, tries to turn their data into JSON,
and puts them into "customer call" or "customer survey" groups
based on what key words are inside.

Args:

raw_records (list): A list of raw Kinesis data items.
call_data_structure (StructType): The blueprint for customer call data.
survey_data_structure (StructType): The blueprint for customer survey data.

Returns:

tuple: Two lists: (call_events, survey_events).

Each list holds the understood JSON data.

```
    """
```

```
call_events = []
```

```
survey_events = []
```

```

for record in raw_records:

    json_text = record['json_data']

    # A simple way to guess the data type based on words it contains

    is_survey = any(field.name in json_text for field in survey_data_structure.fields if
                    field.dataType == IntegerType()) or \
                any(field_name in json_text for field_name in ['survey_id', 'survey_time',
                    '"Q1":', '"Q2":'])

    is_call = any(field.name in json_text for field in call_data_structure.fields if
                  field.dataType == StringType()) or \
                any(field_name in json_text for field_name in ['call_center_id',
                    'talk_channel', 'created_employee_key', '"status":'])

try:

    event_data = json.loads(json_text)

    if is_survey:

        survey_events.append(event_data)

    elif is_call:

        call_events.append(event_data)

    else:

        print(f" ? Data not put into a group: {json_text[:50]}...")

except json.JSONDecodeError:

    print(f" ! Bad JSON format for record: {json_text[:50]}...")

except Exception as e:

    print(f" ! Problem understanding record: {str(e)} - Data: {json_text[:50]}...")

```

```
    return call_events, survey_events
```

```
def send_data_to_warehouse_batch(spark_session, event_list, table_name,  
data_structure, warehouse_login_info):
```

```
    """
```

Sends a group of processed data items to the chosen Redshift table.

It also changes time formats to fit the table's needs.

Args:

spark_session (SparkSession): The active SparkSession.

event_list (list): A list of data items, each being an event.

table_name (str): The name of the Redshift table to write to.

data_structure (StructType): The Spark blueprint matching the table.

warehouse_login_info (dict): Details for connecting to Redshift.

```
    """
```

```
if not event_list:
```

```
    return
```

```
try:
```

```
    # Make a Spark DataFrame from the list of events and apply the blueprint
```

```
    df_to_save = spark_session.createDataFrame(event_list, data_structure)
```

```
    # Change time formats based on the table name
```

```
    if table_name == "customer_interactions_data":
```

```
        df_to_save = df_to_save.select(
```

```
            col("call_id"),
```

```

    col("agent_id"),
    col("service_center_id"),
    col("status"),
    col("category"),
    col("sub_category"),
    col("talk_channel"),
    col("country_code"),
    col("product_code"),

    to_timestamp(col("last_change_time"), "yyyy-MM-dd
HH:mm:ss").alias("last_change_time"),
    to_timestamp(col("create_time"), "yyyy-MM-dd
HH:mm:ss").alias("create_time")
)

elif table_name == "customer_feedback_data":

    df_to_save = df_to_save.select(
        col("survey_id"),
        col("call_id"),
        to_timestamp(col("survey_time"), "yyyy-MM-dd
HH:mm:ss").alias("survey_time"),
        col("score_q1"),
        col("score_q2"),
        col("score_q3"),
        col("comment_q4"),
        col("score_q5")
)

# Send the DataFrame to Redshift using JDBC (Java Database Connectivity)

```

```
df_to_save.write \  
.format("jdbc") \  
.option("url", warehouse_login_info["url"]) \  
.option("dbtable", table_name) \  
.option("user", warehouse_login_info["user"]) \  
.option("password", warehouse_login_info["password"]) \  
.option("driver", warehouse_login_info["driver"]) \  
.mode("append") \  
.save()
```

```
print(f" ✅ Successfully sent {len(event_list)} records to {table_name}")
```

```
except Exception as e:
```

```
    print(f" ❌ Problem sending to {table_name}: {str(e)}")  
    traceback.print_exc()
```

```
def check_warehouse_connection(spark_session, warehouse_login_info):
```

```
    """
```

```
    Tests if we can connect to the Redshift data warehouse by running a simple query.
```

Args:

spark_session (SparkSession): The active SparkSession.

warehouse_login_info (dict): Details for connecting to Redshift.

Returns:

bool: True if the connection works, False otherwise.

```
"""
print("🔍 Checking data warehouse connection...")

try:
    # Try to read from a dummy table or run a simple select to test the connection
    test_df = spark_session.read \
        .format("jdbc") \
        .option("url", warehouse_login_info["url"]) \
        .option("dbtable", "(SELECT 1 as connection_test_col) as test_query") \
        .option("user", warehouse_login_info["user"]) \
        .option("password", warehouse_login_info["password"]) \
        .option("driver", warehouse_login_info["driver"]) \
        .load()

    # Make Spark actually try to connect
    result = test_df.collect()
    print(f"✅ Data warehouse connection successful!")

    return True

except Exception as e:
    print(f"❌ Data warehouse connection failed: {str(e)}")
    traceback.print_exc()
    return False

def run_main_data_pipeline():
    """
```

This is the main part of the live data streaming system.

It manages getting data from Kinesis, processing it with Spark, and saving it to Redshift.

.....

```
print("🚀 Starting Live Data Processing Pipeline...")
print("=" * 60)

# Start Spark Session
spark = start_spark_session()

spark.sparkContext.setLogLevel("WARN") # Set Spark logging to show fewer
messages

# Set up data blueprints
customer_call_data_structure, customer_survey_data_structure =
define_data_structures()

# Data Warehouse (Redshift) connection details
warehouse_login_info = {
    "url": "jdbc:redshift://midhun-proj.008673239246.ap-northeast-1.redshift-
serverless.amazonaws.com:5439/midhun_proj",
    "user": "admin",
    "password": "Midhunproj1",
    "driver": "com.amazon.redshift.jdbc.Driver"
}

print(f"🔗 Data Warehouse Address: {warehouse_login_info['url']}")
```

```
print(f" User: {warehouse_login_info['user']}")  
  
# Check if we can connect to the data warehouse  
  
if not check_warehouse_connection(spark, warehouse_login_info):  
  
    print("✖ Data warehouse connection check failed. Stopping pipeline.")  
  
    return  
  
# Start getting data from Kinesis  
  
data_fetcher = KinesisDataFetcher("Midhun-proj", "ap-northeast-1")  
  
fetch_thread_link = data_fetcher.start_fetching()  
  
  
print("=" * 60)  
  
print("⌚ LIVE DATA SYSTEM STATUS")  
  
print("=" * 60)  
  
print("✓ Kinesis Source Stream: Midhun-proj")  
  
print("✓ How it Works: Background fetching + small-group Spark processing")  
  
print("✓ Customer Call Data → customer_interactions_data table")  
  
print("✓ Customer Survey Data → customer_feedback_data table")  
  
print("✓ Processing Every: 15 seconds")  
  
print("✓ Data Safety: No special custom functions, direct Spark actions")  
  
print("=" * 60)  
  
print("⌚ System running... Press Ctrl+C to stop")  
  
print("=" * 60)  
  
  
try:
```

```
processed_groups_count = 0

while True:

    # Get a group of raw data from the Kinesis fetcher's storage
    current_raw_data_batch = data_fetcher.get_stored_data_batch()

    if current_raw_data_batch:

        print(f" 📦 Processing group {processed_groups_count} with
{len(current_raw_data_batch)} records...")

        # Understand and sort the raw data into different event types
        call_events, survey_events = sort_and_understand_data(
            current_raw_data_batch, customer_call_data_structure,
            customer_survey_data_structure
        )

        # Send the sorted data to their correct Redshift tables
        if call_events:

            print(f" 📋 Found {len(call_events)} customer call events")
            send_data_to_warehouse_batch(
                spark, call_events, "customer_interactions_data",
                customer_call_data_structure, warehouse_login_info
            )

        if survey_events:

            print(f" 🎨 Found {len(survey_events)} customer survey events")
            send_data_to_warehouse_batch(
```

```
spark, survey_events, "customer_feedback_data",
customer_survey_data_structure, warehouse_login_info
)

if not call_events and not survey_events:
    print(f"⚠️ No good data found in this group.")

processed_groups_count += 1

else:
    print("⏳ No new data in storage, waiting for Kinesis to get more...")

# Wait for a set time before processing the next group
time.sleep(15)

except KeyboardInterrupt:
    print("\n🔴 System stopped by user (Ctrl+C).")
except Exception as e:
    print(f"✖️ A major problem happened in the system: {str(e)}")
    traceback.print_exc() # Show full error details for fixing
finally:
    print("🔄 Stopping Kinesis data fetcher...")
    data_fetcher.stop_fetching()
    fetch_thread_link.join(timeout=5) # Wait for the fetching task to finish

    print("🔌 Closing Spark session...")
```

```
spark.stop() # Stop Spark

print("✅ System shutdown complete.")

if __name__ == "__main__":
    run_main_data_pipeline()
```

- **Spark-submit command:** spark-submit --packages "com.amazon.redshift:redshift-jdbc42:2.1.0.9" script.py

3. Redshift Table Definition for Real-Time Cases:

SQL

```
CREATE TABLE IF NOT EXISTS futurecart_case_details (
    case_no varchar(700),
    create_timestamp varchar(700),
    last_modified_timestamp varchar(700),
    created_employee_key varchar(700),
    call_center_id varchar(700),
    status varchar(700),
    category varchar(700),
    sub_category varchar(700),
    communication_mode varchar(700),
    country_cd varchar(700),
    product_code varchar(700)
)
```

Initial data:

```

50
51 select * from futurecart_case_survey_details;

```

Row 50, Col 1, Chr 2900

Result 1 (1600)

	survey_id	case_no	survey_timestamp	q1	q2
1	S-514032	696729	2025-07-21 05:10:51	1	4
2	S-514264	649950	2025-07-21 05:10:51	9	8
3	S-571629	617917	2025-07-21 05:10:51	5	2
4	S-560236	682240	2025-07-21 05:10:51	10	4
5	S-554023	698346	2025-07-21 05:10:51	9	8
6	S-575510	660200	2025-07-21 05:10:51	1	2

K < 1 to 100 of 1600 rows > X Row 51, Col 39, Chr 2893

Query ID 5346 Elapsed time: 144 ms Total rows: 1600

```

50
51 select * from futurecart_case_survey_details;

```

Row 51, Col 39, Chr 2893

Result 1 (2014)

	case_no	create_timestamp	last_modified_timestamp	created_employee_key	call_center_id
1	610253	2025-07-21 05:10:51	2025-07-21 05:10:51	223210	C-114
2	605965	2025-07-21 05:10:51	2025-07-21 05:10:51	277102	C-115
3	613774	2025-07-21 05:10:51	2025-07-21 05:10:51	262430	C-108
4	620044	2025-07-21 05:10:51	2025-07-21 05:10:51	233478	C-102
5	647629	2025-07-21 05:10:51	2025-07-21 05:10:51	270911	C-106
6	643200	2025-07-21 05:10:51	2025-07-21 05:10:51	250067	C-100

K < 1 to 100 of 2014 rows > X Row 51, Col 39, Chr 2893

Query ID 5672 Elapsed time: 25 ms Total rows: 2014

Data after the spark streaming:

```

50
51 select * from futurecart_case_survey_details;

```

Row 50, Col 1, Chr 2900

Result 1 (1608)

	survey_id	case_no	survey_timestamp	q1	q2
1	S-514032	696729	2025-07-21 05:10:51	1	4
2	S-514264	649950	2025-07-21 05:10:51	9	8
3	S-571629	617917	2025-07-21 05:10:51	5	2
4	S-560236	682240	2025-07-21 05:10:51	10	4
5	S-554023	698346	2025-07-21 05:10:51	9	8
6	S-575510	660200	2025-07-21 05:10:51	1	2

K < 1 to 100 of 1608 rows > X Row 51, Col 39, Chr 2893

Query ID 6041 Elapsed time: 25 ms Total rows: 1608

Row 51, Col 1, Chr 2893

The screenshot shows a Redshift query results interface. At the top, there's a header bar with 'Result 1 (2022)' on the left, 'Export' and 'Chart' buttons on the right, and a status message 'Row 51, Col 1, Chr 2893'. Below the header is a table with six columns: 'case_no', 'create_timestamp', 'last_modified_timest...', 'created_employee_key', and 'call_center_id'. The table contains 2022 rows of data. At the bottom of the table, there are navigation buttons for 'K < 1 to 100 of 2022 rows > X' and a status bar with 'Query ID 5986 Elapsed time: 22 ms Total rows: 2022'.

	case_no	create_timestamp	last_modified_timest...	created_employee_key	call_center_id
1	610253	2025-07-21 05:10:51	2025-07-21 05:10:51	223210	C-114
2	605965	2025-07-21 05:10:51	2025-07-21 05:10:51	277102	C-115
3	613774	2025-07-21 05:10:51	2025-07-21 05:10:51	262430	C-108
4	620044	2025-07-21 05:10:51	2025-07-21 05:10:51	233478	C-102
5	647629	2025-07-21 05:10:51	2025-07-21 05:10:51	270911	C-106
6	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
7	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
8	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
9	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
10	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
11	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
12	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
13	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
14	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
15	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
16	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
17	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
18	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
19	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
20	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
21	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
22	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
23	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
24	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
25	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
26	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
27	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
28	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
29	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
30	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
31	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
32	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
33	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
34	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
35	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
36	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
37	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
38	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
39	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
40	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
41	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
42	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
43	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
44	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
45	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
46	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
47	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
48	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
49	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
50	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
51	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
52	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
53	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
54	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
55	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
56	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
57	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
58	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
59	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
60	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
61	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
62	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
63	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
64	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
65	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
66	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
67	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
68	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
69	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
70	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
71	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
72	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
73	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
74	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
75	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
76	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
77	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
78	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
79	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
80	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
81	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
82	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
83	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
84	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
85	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
86	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
87	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
88	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
89	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
90	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
91	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
92	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
93	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
94	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
95	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
96	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
97	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
98	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
99	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100
100	649200	2025-07-21 05:10:51	2025-07-21 05:10:51	250007	C-100

Step 6: Data Analysis and Querying in Redshift

The following queries were executed in Redshift to analyze the data:

- **Total Cases:**

- Query: `SELECT COUNT(case_no) AS total_cases FROM futurecart_case_details;`

- Result: 1080 cases.

The screenshot shows a database interface with two main sections. The top section is titled 'Schedule' and contains a SQL query:

```
1 SELECT
2     COUNT(case_no) AS total_cases
3 FROM
4     futurecart_case_details;
```

The bottom section is titled 'Result 1 (1)' and displays the query results in a table:

	total_cases
	1080

- **Total Open Cases Last 1 Hour:**

- Query: `SELECT COUNT(case_no) AS total_open_cases_last_1_hour FROM futurecart_case_details WHERE status = 'Open' AND create_timestamp >= DATEADD(hour, -1, GETDATE());`
- Result: 3

The screenshot shows a database interface with two main sections. The top section is titled 'Schedule' and contains a SQL query:

```
1 SELECT
2     COUNT(case_no) AS total_open_cases_last_1_hour
3 FROM
4     futurecart_case_details
5 WHERE
6     status = 'Open' AND create_timestamp >= DATEADD(hour, -1, GETDATE());
```

The bottom section is titled 'Result 1 (1)' and displays the query results in a table:

	total_open_cases_la...
	3

- **Total Closed Cases Last 1 Hour:**

- Query: `SELECT COUNT(case_no) AS total_closed_cases_last_1_hour FROM futurecart_case_details WHERE status = 'Closed' AND last_modified_timestamp >= DATEADD(hour, -1, GETDATE());`
- Result: 8

The screenshot shows a dark-themed database interface. At the top, there are several icons: a calendar labeled 'Schedule', a save icon, a refresh icon, and a three-dot menu icon. Below these are two code editor panes. The left pane contains a SQL query:

```

63  SELECT
64    COUNT(case_no) AS total_closed_cases_last_1_hour
65  FROM
66    futurecart_case_details
67  WHERE
68    status = 'Closed' AND last_modified_timestamp >= DATEADD(hour, -1, GETDATE());
69
70  SELECT

```

The right pane shows the results of the query:

Result 1 (1)	
total_closed_cases_l...	8

- **Total Priority Cases (CAT1 or CAT2, or specific sub-categories):**

- Query: `SELECT COUNT(*) AS total_priority_cases FROM futurecart_case_details WHERE category IN ('CAT1', 'CAT2') OR sub_category IN ('SCAT1', 'SCAT2', 'SCAT3', 'SCAT4', 'SCAT5');`
- Result: 1064

The screenshot shows a dark-themed database interface. At the top, there are several icons: a save icon, a refresh icon, and a three-dot menu icon. Below these are two code editor panes. The left pane contains a SQL query:

```

90
91  SELECT COUNT(*) AS total_priority_cases
92  FROM futurecart_case_details
93  WHERE category IN ('CAT1', 'CAT2')
94    OR sub_category IN ('SCAT1', 'SCAT2', 'SCAT3', 'SCAT4', 'SCAT5');

```

The right pane shows the results of the query:

Result 1 (1)	
total_priority_cases	1064

- **Total Surveys Last Hour:**

- Query: `SELECT COUNT(*) AS total_surveys_last_hour FROM futurecart_case_survey_details WHERE CAST(survey_timestamp AS timestamp) >= GETDATE() - INTERVAL '1 hour';`
- Result: 8

A screenshot of a database query results window. The query is:

```
5
6  SELECT COUNT(*) AS total_surveys_last_hour
7  FROM futurecart_case_survey_details
8  WHERE CAST(survey_timestamp AS timestamp) >= GETDATE() - INTERVAL '1 hour';
```

The result table shows one row with the value 8.

total_surveys_last_h...
8

Buttons at the top right include "Export" and "Chart".

- **Open Cases Today:**

- Query: `SELECT COUNT(*) AS open_cases_today FROM futurecart_case_details WHERE status = 'Open' AND CAST(create_timestamp AS timestamp)::date = CURRENT_DATE;`
- Result: 14

A screenshot of a database query results window. The query is:

```
00  SELECT COUNT(*) AS open_cases_today
01  FROM futurecart_case_details
02  WHERE status = 'Open'
03  AND CAST(create_timestamp AS timestamp)::date = CURRENT_DATE;
```

The result table shows one row with the value 14.

open_cases_today
14

- **Open Cases This Week:**

- Query: `SELECT COUNT(*) AS open_cases_this_week FROM futurecart_case_details WHERE status = 'Open' AND DATE_TRUNC('week', CAST(create_timestamp AS timestamp)) = DATE_TRUNC('week', CURRENT_DATE);`
- Result: 220

The screenshot shows a database query results window. The query is:

```

105  SELECT COUNT(*) AS open_cases_this_week
106  FROM futurecart_case_details
107  WHERE status = 'Open'
108  AND DATE_TRUNC('week', CAST(create_timestamp AS timestamp)) = DATE_TRUNC('week', CURRENT_DATE)

```

The result table has one row labeled "Result 1 (1)" with a single column "open_cases_this_week" containing the value "220".

- **Open Cases This Month:**

- Query: `SELECT COUNT(*) AS open_cases_this_month FROM futurecart_case_details WHERE status = 'Open' AND DATE_TRUNC('month', CAST(create_timestamp AS timestamp)) = DATE_TRUNC('month', CURRENT_DATE);`
- Result: 1036

The screenshot shows a database query results window. The query is:

```

SELECT COUNT(*) AS open_cases_this_month
FROM futurecart_case_details
WHERE status = 'Open'
AND DATE_TRUNC('month', CAST(create_timestamp AS timestamp)) = DATE_TRUNC('month', CURRENT_DATE)

```

The result table has one row labeled "Result 1 (1)" with a single column "open_cases_this_mo..." containing the value "1036".

- **Closed Cases Today:**

- Query: `SELECT COUNT(*) AS closed_cases_today FROM futurecart_case_details WHERE status = 'Closed' AND CAST(create_timestamp AS timestamp)::date = CURRENT_DATE;`
- Result: 8

```

115  SELECT COUNT(*) AS closed_cases_today
116  FROM futurecart_case_details
117  WHERE status = 'Closed'
118  AND CAST(create_timestamp AS timestamp)::date = CURRENT_DATE;

```

The screenshot shows a PostgreSQL terminal window. The query in the command line is identical to the one listed above. The results pane shows a single row labeled "closed_cases_today" with a value of 8. There are buttons for "Result 1 (1)", "Export", and a chart icon.

- **Closed Cases This Week:**

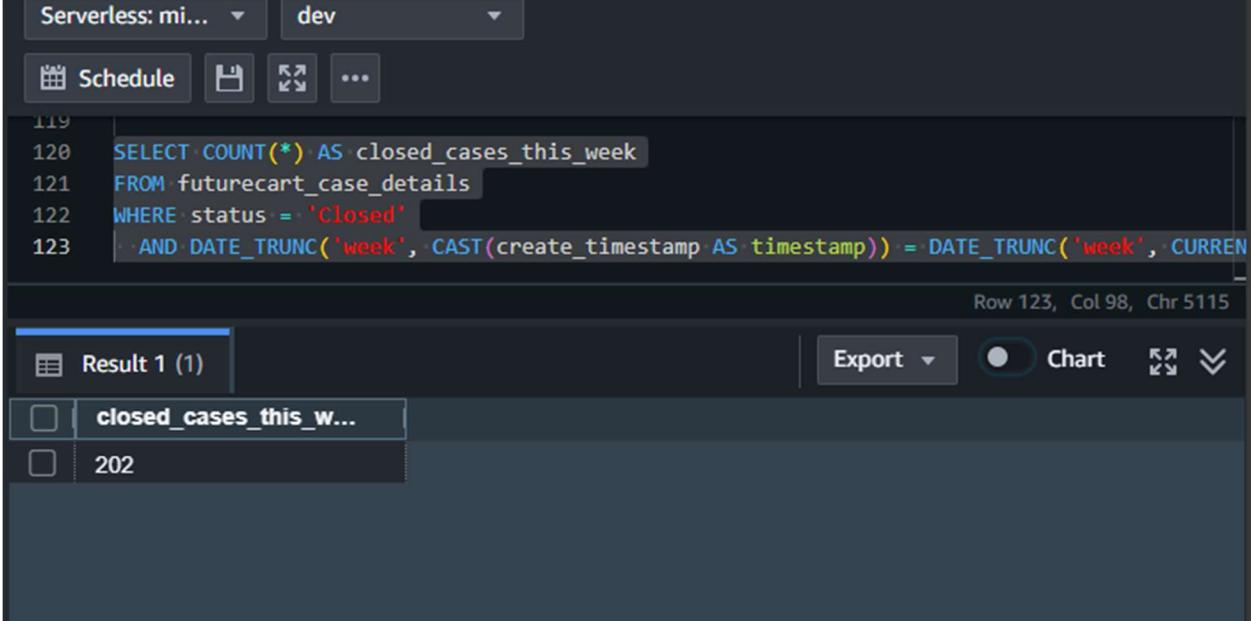
- Query: `SELECT COUNT(*) AS closed_cases_this_week FROM futurecart_case_details WHERE status = 'Closed' AND DATE_TRUNC('week', CAST(create_timestamp AS timestamp)) = DATE_TRUNC('week', CURRENT_DATE);`
- Result: 202

The screenshot shows a PostgreSQL terminal window. The query in the command line is identical to the one listed above. The results pane shows a single row labeled "closed_cases_this_w..." with a value of 202. There are buttons for "Result 1 (1)", "Export", and a chart icon. The status bar at the bottom right indicates "Row 123, Col 98, Chr 5115".

- **Closed Cases This Month:**

- Query:

```
SELECT COUNT(*) AS closed_cases_this_month FROM futurecart_case_details WHERE status = 'Closed' AND DATE_TRUNC('month', CAST(create_timestamp AS timestamp)) = DATE_TRUNC('month', CURRENT_DATE);
```
- Result: 986



The screenshot shows a database interface with the following details:

- Serverless: mi...** dropdown menu.
- dev** environment dropdown menu.
- Schedule** button.
- ...** button.
- Query Editor:**

```
119
120  SELECT COUNT(*) AS closed_cases_this_week
121  FROM futurecart_case_details
122  WHERE status = 'Closed'
123  AND DATE_TRUNC('week', CAST(create_timestamp AS timestamp)) = DATE_TRUNC('week', CURRENT_DATE)
```
- Result Panel:**
 - Result 1 (1)** tab selected.
 - closed_cases_this_w...** column header.
 - 202** value in the result row.
- Export** button.
- Chart** button.
- Row 123, Col 98, Chr 5115** status bar.

- **Total Surveys Today:**

- Query:

```
SELECT COUNT(*) AS total_surveys_today FROM futurecart_case_survey_details WHERE CAST(survey_timestamp AS timestamp)::date = CURRENT_DATE;
```
- Result: 8

```
129
130     SELECT COUNT(*) AS total_surveys_today
131     FROM futurecart_case_survey_details
132    WHERE CAST(survey_timestamp AS timestamp)::date = CURRENT_DATE;
```

Result 1 (1)

	total_surveys_today
	8

Export ▾

- **Total Surveys This Week:**

- Query: `SELECT COUNT(*) AS total_surveys_this_week FROM futurecart_case_survey_details WHERE DATE_TRUNC('week', CAST(survey_timestamp AS timestamp)) = DATE_TRUNC('week', CURRENT_DATE);`
- Result: 328

Run Limit 100 Explain Isolated session i

Serverless: mi... dev

Schedule

```
132 WHERE CAST(survey_timestamp AS timestamp)::date = CURRENT_DATE;
133
134     SELECT COUNT(*) AS total_surveys_this_week
135     FROM futurecart_case_survey_details
136    WHERE DATE_TRUNC('week', CAST(survey_timestamp AS timestamp)) = DATE_TRUNC('week', CURREN
```

Row 136, Col 98, Chr 5643

Result 1 (1)

	total_surveys_this_w...
	328

Export ▾ Chart ▾ ▾

- **Total Surveys This Month:**

- Query: `SELECT COUNT(*) AS total_surveys_this_month FROM futurecart_case_survey_details WHERE DATE_TRUNC('month', CAST(survey_timestamp AS timestamp)) = DATE_TRUNC('month', CURRENT_DATE);`
- Result: 1608

The screenshot shows a database query results window. The SQL query is:

```

137
138   SELECT COUNT(*) AS total_surveys_this_month
139   FROM futurecart_case_survey_details
140   WHERE DATE_TRUNC('month', CAST(survey_timestamp AS timestamp)) = DATE_TRUNC('month', CURRENT_DATE);

```

The result table has one row:

total_surveys_this_m...
1608

Row 140, Col 100, Chr 5828

- **Total Cases by Status:**

- Query: `SELECT status, COUNT(*) AS total_cases FROM futurecart_case_details GROUP BY status;`
- Result: Open: 1036, Closed: 986

The screenshot shows a database query results window. The SQL query is:

```

12
13   SELECT status, COUNT(*) AS total_cases
14   FROM futurecart_case_details
15   GROUP BY status;

```

The result table has two rows:

status	total_cases
Open	1036
Closed	986

- **Cases by Category and Sub-category (Priority and Severity):**

- Query:

```
SELECT category AS priority, sub_category AS severity, COUNT(*) AS total_cases FROM futurecart_case_details GROUP BY category, sub_category ORDER BY category, sub_category;
```

```
6 SELECT category AS priority, sub_category AS severity, COUNT(*) AS total_cases
7 FROM futurecart_case_details
8 GROUP BY category, sub_category
9 ORDER BY category, sub_category;
```

Row 149, Col 33, Chr 6097

priority	severity	total_cases
CAT1	SCAT1	36
CAT1	SCAT10	10
CAT1	SCAT11	30
CAT1	SCAT12	18
CAT1	SCAT13	30
CAT1	SCAT14	10

```
6 SELECT category AS priority, sub_category AS severity, COUNT(*) AS total_cases
7 FROM futurecart_case_details
8 GROUP BY category, sub_category
9 ORDER BY category, sub_category;
```

Row 149, Col 33, Chr 6097

priority	severity	total_cases
CAT1	SCAT1	36
CAT1	SCAT10	10
CAT1	SCAT11	30
CAT1	SCAT12	18
CAT1	SCAT13	30
CAT1	SCAT14	10