**📘 FutureCart CRM Analytics Project Documentation**

**1. 🎯 Business Objective**

FutureCart Inc. aims to enhance customer satisfaction and loyalty through a robust CRM strategy. The company operates over 5000 retail stores and an e-commerce platform across India. A dedicated Customer Care team handles complaints and feedback, which are logged as cases and followed by customer surveys.

The goal is to:

* Improve after-sales service.
* Track and optimize customer support performance.
* Enable real-time and historical insights for leadership decision-making.

**2. 🧩 Business Process Overview**

* Customers contact support via **calls, chat, or email**.
* A **Customer Care Representative (CCR)** logs each interaction as a **case**.
* Cases are categorized and prioritized based on business rules.
* After resolution, customers receive a **survey** to rate their experience.
* All data is collected and analyzed to monitor **KPIs** and improve service.

**3. 📊 Key KPIs Tracked**

| **Type** | **KPI** |
| --- | --- |
| Real-Time | Open/closed cases in last hour, priority cases, survey sentiment |
| Batch | Daily/weekly/monthly case volumes, survey counts, sentiment trends |

**4. 🏗️ Technical Architecture: Lambda-Based Data Mart**

**🔄 Data Flow Layers**

| **Layer** | **Purpose** | **Tools** |
| --- | --- | --- |
| **Extract** | Ingest raw data | Kinesis (real-time), S3 (batch), EC2 MySQL (dimension) |
| **Landing** | Store raw data | EMR + Hive |
| **Transformation** | Clean & model data | Hive/Spark |
| **Load** | Store for analytics | Redshift |
| **Visualization** | Business insights | QuickSight |

**5. 🛠️ Implementation Strategy**

**🔹 Challenge: EMR is shut down nightly by admin (post 11 PM), risking data loss.**

**🔹 Solution:**

* Use **EC2-hosted MySQL** to store dimension data persistently.
* Use **Sqoop** to transfer data from MySQL to Hive when EMR is available.
* Export Hive data to **S3**, then load into **Redshift**.

**6. 📝 Step-by-Step Execution Plan**

**Step 1: Dimension Data Setup**

* Create .txt files for 8 dimension datasets.
* Create MySQL tables on EC2 using SQL scripts.
* Load .txt files into MySQL using LOAD DATA INFILE.

**Step 2: Batch Ingestion**

* Use Sqoop to import MySQL dimension tables into Hive.
* Use Spark to export Hive tables to S3.
* Load S3 data into Redshift.

**Step 3: Historical Data Setup**

* Use generate\_historical\_data.py to simulate 10 days of case and survey data in JSON format.
* Load JSON files into HDFS.
* Create Hive tables for historical case and survey data.

**Step 4: Real-Time Data Simulation**

* Use stream\_to\_kinesis.py to generate real-time case and survey events.
* Send JSON data to Kinesis streams:
  + futurecart\_case\_event
  + futurecart\_survey\_event

**Step 5: Real-Time Processing**

* Create a consumer application to read from Kinesis.
* Parse incoming JSON:
  + If it's a case → load into Redshift case table.
  + If it's a survey → load into Redshift survey table.

**Step 6: Data Export and Load**

* Export transformed Hive data to S3 using Spark DataFrames.
* Load S3 data into Redshift using COPY or Glue.

**Step 7: KPI Queries**

* Write SQL queries on Redshift to calculate KPIs:
  + Case volumes
  + SLA compliance
  + Survey sentiment

**Step 8: Visualization**

* Connect QuickSight to Redshift.
* Build dashboards for CRM KPIs.

**Dimension Data to S3   
  
Dimension Data Workflow** from .txt files on your local system to   
**S3 via EC2 → MariaDB → EMR/Hive → Spark**.

| **Phase** | **Description** |
| --- | --- |
| 1 | Setup EC2 and install MariaDB |
| 2 | Create database and tables |
| 3 | Upload .txt files from local to EC2 |
| 4 | Load data into MariaDB |
| 5 | Setup EMR and Hive |
| 6 | Transfer data from MariaDB to Hive using Sqoop |
| 7 | Export Hive tables to S3 using Spark |

**🧱 Phase 1: Setup EC2 and Install MariaDB**

**🔹 Step 1.1: Launch EC2 Instance**

* **AMI**: Amazon Linux 2
* **Instance Type**: t2.medium or higher
* **Storage**: 20 GB
* **Security Group**:
  + Allow **SSH (port 22)** from your IP
  + Allow **MySQL (port 3306)** from EMR subnet or your IP
  + Allow http
  + Allow redshift

**🔹 Step 1.2: Connect to EC2**

ssh -i your-key.pem ec2-user@<EC2-Public-IP>

**🔹 Step 1.3 Install MariaDB 10.5 on Amazon Linux 2023**

1. **Update packages:**

sudo dnf update -y

1. **Install MariaDB 10.5:**

sudo dnf install mariadb105-server mariadb105 -y

1. **Start and enable MariaDB service:**

sudo systemctl start mariadb

sudo systemctl enable mariadb

1. **Secure MariaDB installation:**

sudo mysql\_secure\_installation

Follow the prompts to set root password and remove test users/databases.

1. **Verify MariaDB is running:**

sudo systemctl status mariadb

**🗃️ Phase 2: Create Database and Tables**

**🔹 Step 2.1: Login to MariaDB**

mysql -u root -p

**🔹 Step 2.2: Create Database**

CREATE DATABASE futurecart\_crm;

USE futurecart\_crm;

**🔹 Step 2.3: Create All 8 Tables**

-- SQL for futurecart\_calendar\_details

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)

    );

-- SQL for futurecart\_call\_center\_details

CREATE TABLE futurecart\_call\_center\_details (

        call\_center\_id VARCHAR(10),

        call\_center\_vendor VARCHAR(50),

        location VARCHAR(50),

        country VARCHAR(50)

    );

-- SQL for futurecart\_case\_category\_details

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),

        sla\_hours INT

    );

-- SQL for futurecart\_case\_country\_details

CREATE TABLE futurecart\_case\_country\_details (

        id INT,

        name VARCHAR(75),

        alpha\_2 VARCHAR(2),

        alpha\_3 VARCHAR(3)

    );

-- SQL for futurecart\_case\_priority\_details

CREATE TABLE futurecart\_case\_priority\_details (

        priority\_key VARCHAR(5),

        priority VARCHAR(20),

        severity VARCHAR(100),

        sla VARCHAR(100)

    );

-- SQL for futurecart\_employee\_details

CREATE TABLE futurecart\_employee\_details (

        emp\_key INT,

        first\_name VARCHAR(50),

        last\_name VARCHAR(50),

        email VARCHAR(100),

        gender VARCHAR(10),

        ldap VARCHAR(50),

        hire\_date DATE,

        manager VARCHAR(50)

    );

-- SQL for futurecart\_product\_details

CREATE TABLE futurecart\_product\_details (

        product\_id VARCHAR(20),

        department VARCHAR(50),

        brand VARCHAR(50),

        commodity\_desc VARCHAR(100),

        sub\_commodity\_desc VARCHAR(100)

    );

-- SQL for futurecart\_survey\_question\_details (// put range in ‘ ‘ as The problem here is that range is a reserved keyword)

CREATE TABLE futurecart\_survey\_question\_details (

question\_id VARCHAR(10),

question\_desc VARCHAR(255),

response\_type VARCHAR(50),

`range` VARCHAR(20),

negative\_response\_range VARCHAR(20),

neutral\_response\_range VARCHAR(20),

positive\_response\_range VARCHAR(20)

);

A screen shot of a computer

AI-generated content may be incorrect.

**📂 Phase 3: Upload .txt Files to EC2**

**🔹 Step 3.1: From Local to EC2**

scp -i your-key.pem futurecart\_\*.txt ec2-user@<EC2-Public-IP>:/home/ec2-user/

**🧾 Phase 4: Load Data into MariaDB**

**🔹 Step 4.1.1: Enable Local Infile (if needed)**

sudo vi /etc/my.cnf

# Add under [mysqld]

local\_infile=1

**ALTERNATIVE**  
**Step 4.1.2: Upload files to S3 then extract it from ec2**aws s3 cp s3://tbsm-source/db/ ~/ --recursive

CHECK FOR THE FILES  
A screen shot of a computer code

AI-generated content may be incorrect.

# Restart MariaDB

sudo systemctl restart mariadb

**Step 4.1.3: Login again and setup permission for loading for marida db  
1. sudo mysql -u root -p**

**2 Allow MariaDB to load local files**

Check MariaDB config allows local file loading. In your MariaDB session:

- SHOW VARIABLES LIKE 'local\_infile';

**3.If it's OFF, enable it:**

- SET GLOBAL local\_infile = 1;

**Step 4.1.4:**

**Use database**use futurecart\_crm;

**🔹 Step 4.2: Load One Table (Example: Calendar)**

LOAD DATA LOCAL INFILE '/home/ec2-user/futurecart\_calendar\_details.txt'

INTO TABLE futurecart\_calendar\_details

FIELDS TERMINATED BY ' '

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

LOAD DATA LOCAL INFILE '/home/ec2-user/futurecart\_call\_center\_details.txt'

INTO TABLE futurecart\_call\_center\_details

FIELDS TERMINATED BY ' '

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

LOAD DATA LOCAL INFILE '/home/ec2-user/futurecart\_case\_category\_details.txt'

INTO TABLE futurecart\_case\_category\_details

FIELDS TERMINATED BY ' '

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

LOAD DATA LOCAL INFILE '/home/ec2-user/futurecart\_case\_country\_details.txt'

INTO TABLE futurecart\_case\_country\_details

FIELDS TERMINATED BY ' '

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

LOAD DATA LOCAL INFILE '/home/ec2-user/futurecart\_case\_priority\_details.txt'

INTO TABLE futurecart\_case\_priority\_details

FIELDS TERMINATED BY ' '

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

LOAD DATA LOCAL INFILE '/home/ec2-user/futurecart\_employee\_details.txt'

INTO TABLE futurecart\_employee\_details

FIELDS TERMINATED BY ' '

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

LOAD DATA LOCAL INFILE '/home/ec2-user/futurecart\_product\_details.txt'

INTO TABLE futurecart\_product\_details

FIELDS TERMINATED BY ' '

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

LOAD DATA LOCAL INFILE '/home/ec2-user/futurecart\_survey\_question\_details.txt'

INTO TABLE futurecart\_survey\_question\_details

FIELDS TERMINATED BY ' '

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

Cross check for count and run one or two select cmd to check data

**A black screen with white text

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**☁️ Phase 5: Setup EMR Cluster**

**🔹 Step 5.1: Launch EMR**

* **Release**: EMR 6.15.0
* **Applications**: Hive, Hadoop, Spark, Sqoop
* **Instance Type**: m5.xlarge (3 nodes)
* **Log URI**: s3://your-bucket/emr-logs/
* **IAM Roles**: ec2-s3-full-access

**🔄 Phase 6: Transfer Data via Sqoop**

**🔹 Step 6.1: SSH into EMR Master**

ssh -i your-key.pem hadoop@<EMR-Master-IP>

Verify HDFS directories and create user directory if needed:

hdfs dfs -ls /

hdfs dfs -mkdir -p /user/ec2-user

hdfs dfs -chown ec2-user /user/ec2-user

Launch Hive CLI and create/use the database:

hive

Inside Hive CLI:

CREATE DATABASE IF NOT EXISTS dimension;

SHOW DATABASES;

USE dimension;

EXIT;

**🔹 Step 6.2: Run Sqoop Import (Example: Calendar Table)**

sqoop import \

--connect jdbc:mysql://172.31.8.1:3306/futurecart\_crm \

--username root \

--password tekavade123 \

--table futurecart\_calendar\_details \

--hive-import \

--hive-database dimension \

--hive-table futurecart\_calendar\_details \

--create-hive-table \

--fields-terminated-by '\t' \

-m 1 \

--driver org.mariadb.jdbc.Driver

**🔹 Step 6.3: Import Remaining Tables**

Repeat the Sqoop import command for all other tables, changing --table and --hive-table accordingly:

sqoop import \

--connect jdbc:mysql://172.31.8.1:3306/futurecart\_crm \

--username root \

--password tekavade123 \

--table futurecart\_call\_center\_details \

--hive-import \

--hive-database dimension \

--hive-table futurecart\_call\_center\_details \

--create-hive-table \

--fields-terminated-by '\t' \

-m 1 \

--driver org.mariadb.jdbc.Driver

Similarly for:

* futurecart\_case\_category\_details
* futurecart\_case\_country\_details
* futurecart\_case\_priority\_details
* futurecart\_employee\_details
* futurecart\_product\_details
* futurecart\_survey\_question\_details

**Note:** Using -m 1 ensures sequential import (no splitting), which worked well for you since some tables might not have primary keys.

**🚀 Phase 7: Export Hive Tables to S3 via Spark**

**🔹 Step 7.1: Create Spark Script (export\_hive\_tables\_to\_s3.py)**

from pyspark.sql import SparkSession

def export\_table(spark, hive\_db, table\_name, s3\_path):

full\_table\_name = f"{hive\_db}.{table\_name}"

print(f"Exporting {full\_table\_name} to {s3\_path} ...")

# Read Hive table

df = spark.table(full\_table\_name)

# Write as Parquet to S3 (overwrite if exists)

df.write.mode("overwrite").parquet(s3\_path)

print(f"Done exporting {full\_table\_name}")

def main():

spark = SparkSession.builder \

.appName("HiveToS3Export") \

.enableHiveSupport() \

.getOrCreate()

hive\_db = "dimension"

s3\_base\_path = "s3://tbsm-destination/db/"

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"

]

for table in tables:

s3\_path = f"{s3\_base\_path}{table}"

export\_table(spark, hive\_db, table, s3\_path)

spark.stop()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**🔹 Step 7.2: Run Spark Job**

spark-submit export\_hive\_tables\_to\_s3.py

**✅ Final Output**

* All 8 dimension tables are:
  + Persisted in EC2 MariaDB
  + Imported into Hive on EMR
  + Exported to S3 in Parquet format

**Step 2: Historical Data Ingestion – Complete Guide**

**Overview**

* Generate 10 days of historical case & survey data as JSON Lines files locally on EMR.
* Upload JSON data to HDFS.
* Create Hive external tables on JSON data.
* Export Hive tables to S3 in Parquet format using Spark.

**Step 2.1: Generate Historical JSON Data**

1. **Create Python script to generate JSON data**

nano generate\_historical\_data.py

1. **Copy-paste this Python code into generate\_historical\_data.py:**

import os

import json

import random

from datetime import datetime, timedelta

NUM\_DAYS = 10

CASES\_PER\_DAY = 100

SURVEYS\_PER\_DAY = 80

OUTPUT\_DIR = "historical\_data\_jsonl"

def random\_case\_no():

return str(random.randint(600000, 700000))

def random\_timestamp(day\_offset):

date = datetime.now() - timedelta(days=day\_offset)

return date.strftime("%Y-%m-%d %H:%M:%S")

def generate\_case(case\_no, day\_offset):

return {

"status": random.choice(["Open", "Closed"]),

"category": f"CAT{random.randint(1,5)}",

"sub\_category": f"SCAT{random.randint(1,20)}",

"last\_modified\_timestamp": random\_timestamp(day\_offset),

"case\_no": case\_no,

"create\_timestamp": random\_timestamp(day\_offset),

"created\_employee\_key": str(random.randint(200000, 300000)),

"call\_center\_id": f"C-{random.randint(100,120)}",

"product\_code": str(random.randint(9000000, 9999999)),

"country\_cd": random.choice(["IN", "US", "BR", "DE", "AU"]),

"communication\_mode": random.choice(["Email", "Call", "Chat"])

}

def generate\_survey(case\_no, day\_offset):

return {

"survey\_id": f"S-{random.randint(500000, 599999)}",

"case\_no": case\_no,

"survey\_timestamp": random\_timestamp(day\_offset),

"Q1": random.randint(1, 10),

"Q2": random.randint(1, 10),

"Q3": random.randint(1, 10),

"Q4": random.choice(["Y", "N"]),

"Q5": random.randint(1, 10)

}

def ensure\_dirs():

os.makedirs(f"{OUTPUT\_DIR}/cases", exist\_ok=True)

os.makedirs(f"{OUTPUT\_DIR}/surveys", exist\_ok=True)

def main():

ensure\_dirs()

for day in range(1, NUM\_DAYS + 1):

case\_nos = [random\_case\_no() for \_ in range(CASES\_PER\_DAY)]

cases = [generate\_case(cn, day) for cn in case\_nos]

surveys = [generate\_survey(cn, day) for cn in random.sample(case\_nos, SURVEYS\_PER\_DAY)]

with open(f"{OUTPUT\_DIR}/cases/case\_data\_day{day}.json", "w") as f:

for record in cases:

f.write(json.dumps(record) + "\n")

with open(f"{OUTPUT\_DIR}/surveys/survey\_data\_day{day}.json", "w") as f:

for record in surveys:

f.write(json.dumps(record) + "\n")

print(f"✅ JSON Lines generated for {NUM\_DAYS} days in '{OUTPUT\_DIR}/'")

if \_\_name\_\_ == "\_\_main\_\_":

main()

1. **Run the script to generate JSON data locally**

python3 generate\_historical\_data.py

* This will create a folder historical\_data\_jsonl with two subfolders: cases and surveys.
* Each contains 10 JSON files — one for each day.

**Step 2.2: Upload JSON Data to HDFS**

1. **Create necessary HDFS directories (run these commands):**

sudo -u hdfs hdfs dfs -mkdir -p /data/historical/cases

sudo -u hdfs hdfs dfs -mkdir -p /data/historical/surveys

1. **Give your user ownership of these directories:**

sudo -u hdfs hdfs dfs -chown -R ec2-user:ec2-user /data/historical

1. **Upload JSON files to HDFS:**

hdfs dfs -put -f historical\_data\_jsonl/cases/\*.json /data/historical/cases/

hdfs dfs -put -f historical\_data\_jsonl/surveys/\*.json /data/historical/surveys/

**Step 2.3: Create Hive External Tables**

1. **Open Hive shell**

hive

1. **Run these commands in Hive shell to create external tables**

CREATE EXTERNAL TABLE case\_details (

status STRING,

category STRING,

sub\_category STRING,

last\_modified\_timestamp STRING,

case\_no STRING,

create\_timestamp STRING,

created\_employee\_key STRING,

call\_center\_id STRING,

product\_code STRING,

country\_cd STRING,

communication\_mode STRING

)

ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'

LOCATION '/data/historical/cases';

CREATE EXTERNAL TABLE case\_survey\_details (

survey\_id STRING,

case\_no STRING,

survey\_timestamp STRING,

Q1 INT,

Q2 INT,

Q3 INT,

Q4 STRING,

Q5 INT

)

ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'

LOCATION '/data/historical/surveys';

1. **Check the data loaded**

SELECT COUNT(\*) FROM case\_details;

SELECT COUNT(\*) FROM case\_survey\_details;

**Step 2.4: Export Hive Tables to S3 as Parquet Using Spark**

1. **Create Spark export script**

nano export\_historical\_to\_s3.py

1. **Paste this Spark Python code**

from pyspark.sql import SparkSession

def main():

spark = SparkSession.builder \

.appName("ExportHistoricalToS3") \

.enableHiveSupport() \

.getOrCreate()

tables = {

"case\_details": "s3://tbsm-core/historical/case/",

"case\_survey\_details": "s3://tbsm-core/historical/survey/"

}

for table, path in tables.items():

print(f"Exporting {table} to {path} ...")

df = spark.sql(f"SELECT \* FROM {table}")

df.write.mode("overwrite").parquet(path)

print(f"Exported {table} successfully.")

spark.stop()

if \_\_name\_\_ == "\_\_main\_\_":

main()

1. **Run the Spark job**

spark-submit export\_historical\_to\_s3.py

**🎉 All done!**

* Historical data JSON generated locally.
* Data uploaded to HDFS.
* Hive tables created on JSON data.
* Data exported from Hive tables to S3 as Parquet.