Capstone Project 9. Comprehensive Data Pipeline with Airflow, Kafka, PySpark and Visualization on for Credit Card Approval Analysis.

Name: Varsha .S
Roll On: 23EE056

Objective:

The objective of this capstone project is to build a robust, scalable, and fault-tolerant end-to-end data pipeline for an e-commerce platform. The system will ingest real-time order and customer data using Kafka, transform and process this data using PySpark, and store the results in a queryable format. The transformed data will be visualized using Plotly or Power BI to generate actionable insights. The entire workflow will be orchestrated and automated using Apache Airflow, enabling scheduled execution, monitoring, and dependency management.

This project demonstrates the ability to design modern data pipelines combining streaming ingestion, batch processing, distributed transformation, orchestration, and business intelligence reporting in a production-ready architecture.

Objectives:

1. Data Extraction: Ingest data from multiple sources using HDFS.
2. Data Transformation: Use PySpark in for data cleaning, transformation, and enrichment.
3. Materialized Views: Create SQL-based materialized views for efficient querying and analysis.
4. Visualization: Develop a visualization layer using Power BI or Plotly to present the Visualization.

Architecture:

1. Input Data Sources (Multiple):
   - Use Data sources such as HDFS, and relational databases (MySQL).
   - Read dataset from multiple sources using Pyspark or Stream Data using Kafka Producer.

2. Materialized Views (SQL):
   - Transform data using PySpark.

- Load the transformed data into SQL Database.
- Create materialized views for optimized querying and reporting.

3. Visualization Layer (Power BI/Dashboard):
    - Create interactive visualization of analysis to visualize key metrics and insights from dataset.

Dataset:

https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction?select=credit_record.csv

Dataset 1

- ID: Unique identifier for each client.
- CODE_GENDER: Gender of the client.
- FLAG_OWN_CAR: Whether the client owns a car.
- FLAG_OWN_REALTY: Whether the client owns property.
- CNT_CHILDREN: The number of children the client has.
- AMT_INCOME_TOTAL: The total annual income of the client.
- NAME_INCOME_TYPE: The category of the client's income source.
- NAME_EDUCATION_TYPE: The highest education level the client has achieved.
- NAME_FAMILY_STATUS: The marital status of the client.
- NAME_HOUSING_TYPE: The client's living situation.
- DAYS_BIRTH: The client's age in days, counted backwards from the current day.
- DAYS_EMPLOYED: How long the client has been employed, counted backwards from the current day. Positive numbers indicate unemployment.
- FLAG_MOBIL: Whether the client owns a mobile phone.
- FLAG_WORK_PHONE: Whether the client has a work phone.
- FLAG_PHONE: Whether the client has a phone.
- FLAG_EMAIL: Whether the client has an email address.
- OCCUPATION_TYPE: The client's occupation.
- CNT_FAM_MEMBERS: The size of the client's family.

The credit_record.csv dataset tracks the credit history of clients, with each record reflecting a monthly snapshot of an individual's credit file.

Dataset 2

- ID: Unique identifier for each client, matching the ID in the application_record.csv.
- MONTHS_BALANCE: The month of the record relative to the current month (0 is current, -1 is previous month, etc.).
- STATUS: The status of the client's credit for that month (e.g., no overdue, days past due, paid off).

Steps to Implement:

1. Data Ingestion Layer (Kafka)

Goal:

Ingest real-time data such as customer orders, product views, cart additions, etc., into the pipeline.

Steps:

- Use a Python Kafka producer to send these simulated events to Kafka topics
- Set up Kafka topics and configure Kafka brokers and Zookeeper locally or on a cluster.
- Use Kafka Consumer within Pyspark Structured Streaming to read incoming data in near real-time.

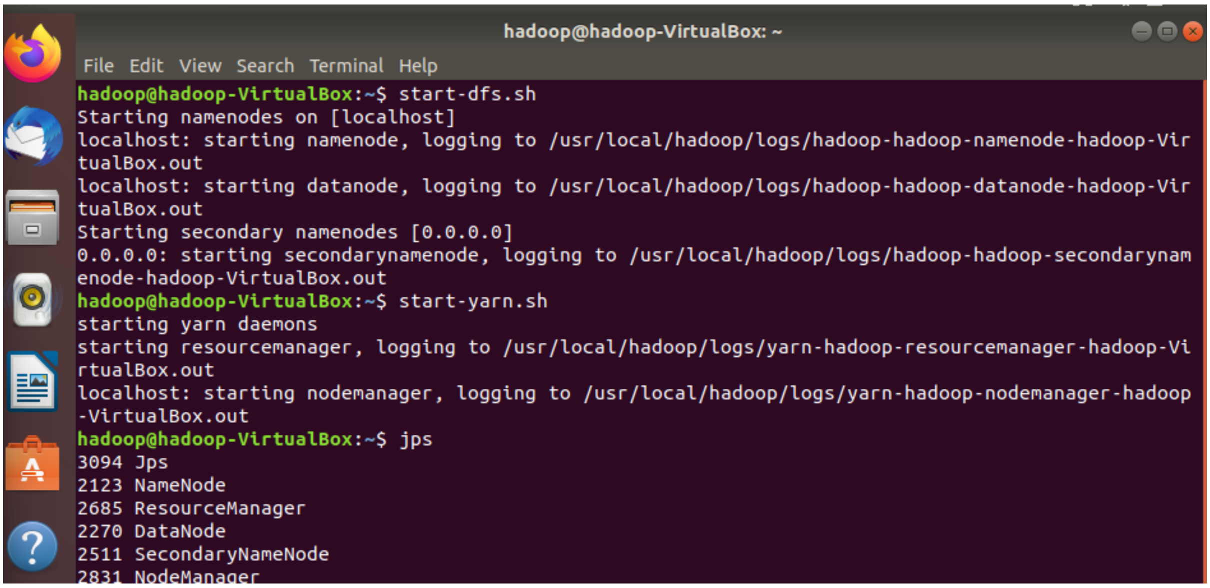
Perform transformations such as:

- Join application + credit records on ID.
- Aggregate credit STATUS by MONTHS_BALANCE.
- Show trends in loan repayment status (e.g., counts of STATUS = 0 = no DPD, 1 = 1–30 DPD, etc.).

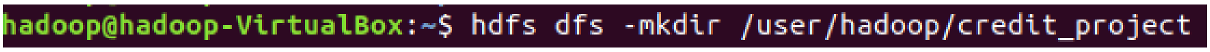
Technologies:

Kafka, Python Producer, JSON/CSV format

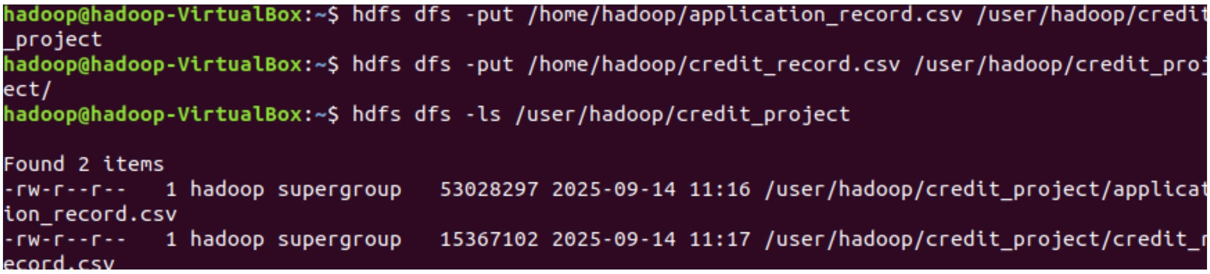
Outputs:


hadoop@hadoop-VirtualBox: ~
File Edit View Search Terminal Help

```
hadoop@hadoop-VirtualBox:~$ start-dfs.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hadoop-namenode-hadoop-Vir
tualBox.out
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hadoop-datanode-hadoop-Vir
tualBox.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hadoop-secondarynam
enode-hadoop-VirtualBox.out
hadoop@hadoop-VirtualBox:~$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-resourcemanager-hadoop-Vi
rtualBox.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-nodemanager-hadoop
-VirtualBox.out
hadoop@hadoop-VirtualBox:~$ jps
3094 Jps
2123 NameNode
2685 ResourceManager
2270 DataNode
2511 SecondaryNameNode
2831 NodeManager
```

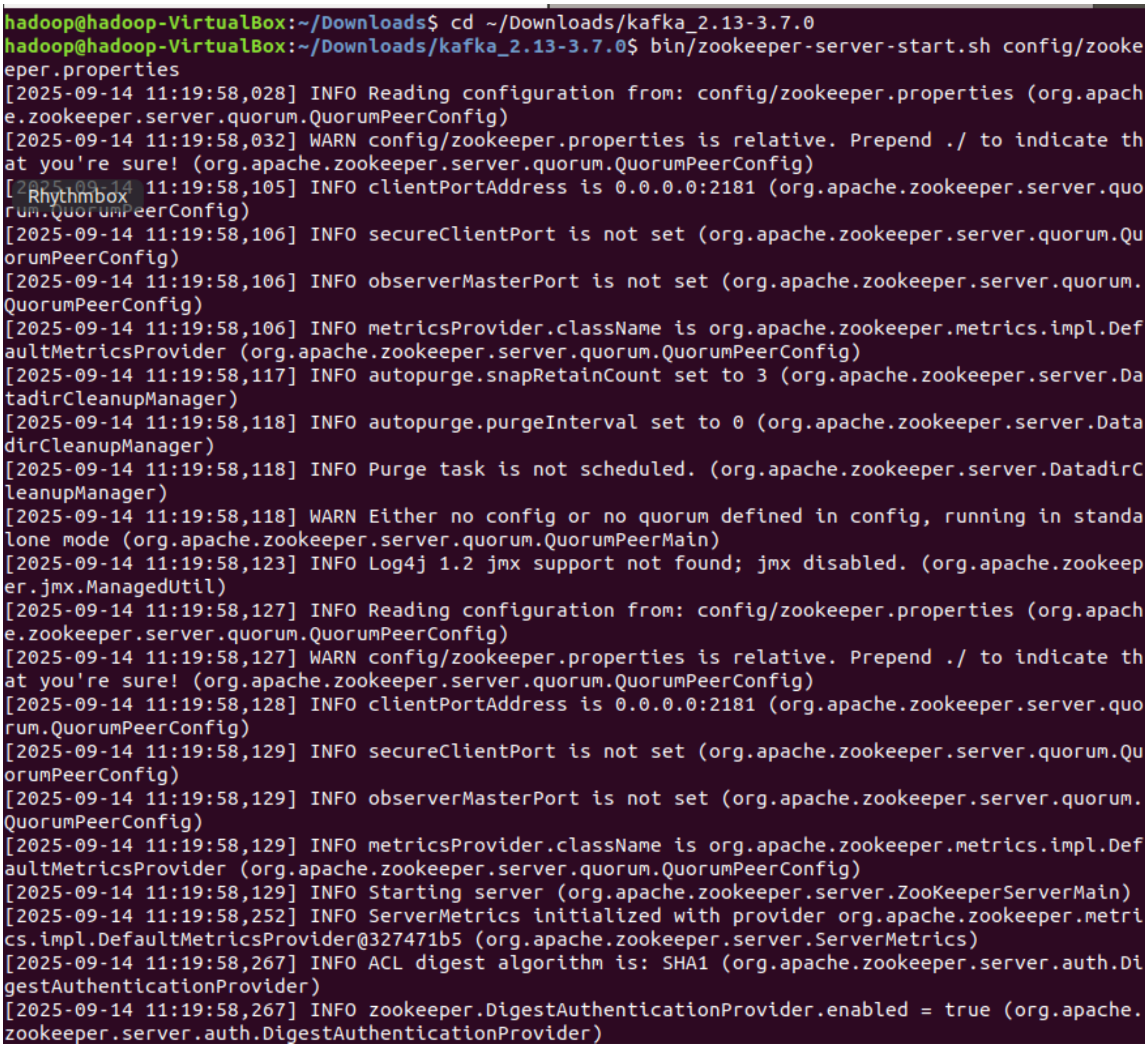


```
hadoop@hadoop-VirtualBox:~$ hdfs dfs -mkdir /user/hadoop/credit_project
```



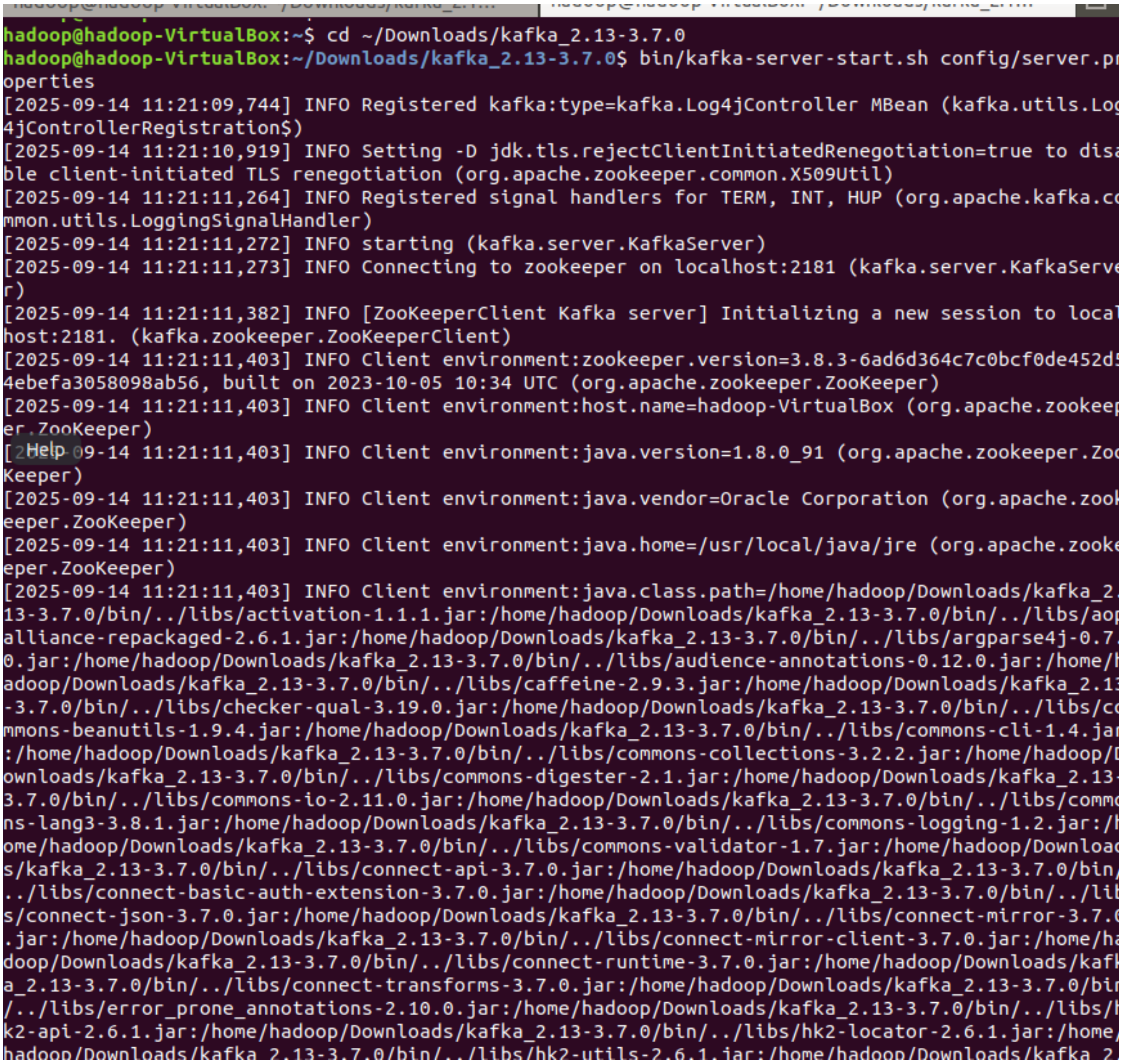
```
hadoop@hadoop-VirtualBox:~$ hdfs dfs -put /home/hadoop/application_record.csv /user/hadoop/credit
_project
hadoop@hadoop-VirtualBox:~$ hdfs dfs -put /home/hadoop/credit_record.csv /user/hadoop/credit_proj
ect/
hadoop@hadoop-VirtualBox:~$ hdfs dfs -ls /user/hadoop/credit_project

Found 2 items
-rw-r--r--   1 hadoop supergroup   53028297 2025-09-14 11:16 /user/hadoop/credit_project/applicat
ion_record.csv
-rw-r--r--   1 hadoop supergroup   15367102 2025-09-14 11:17 /user/hadoop/credit_project/credit_r
ecord.csv
```

START ZOOKEEPER



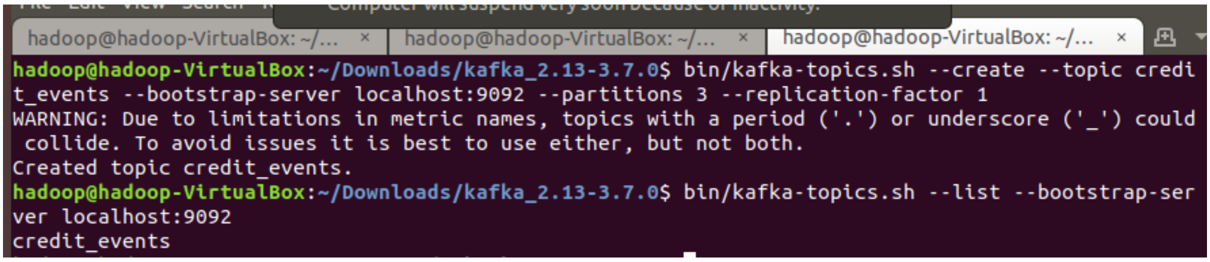
```
hadoop@hadoop-VirtualBox:~/Downloads$ cd ~/Downloads/kafka_2.13-3.7.0
hadoop@hadoop-VirtualBox:~/Downloads/kafka_2.13-3.7.0$ bin/zookeeper-server-start.sh config/zooke
eper.properties
[2025-09-14 11:19:58,028] INFO Reading configuration from: config/zookeeper.properties (org.apach
e.zookeeper.server.quorum.QuorumPeerConfig)
[2025-09-14 11:19:58,032] WARN config/zookeeper.properties is relative. Prepend ./ to indicate th
at you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-09-14 11:19:58,105] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quo
rum.QuorumPeerConfig)
[2025-09-14 11:19:58,106] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.Qu
orumPeerConfig)
[2025-09-14 11:19:58,106] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.
QuorumPeerConfig)
[2025-09-14 11:19:58,106] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.Def
aultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-09-14 11:19:58,117] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.Da
tadirCleanupManager)
[2025-09-14 11:19:58,118] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.Data
dirCleanupManager)
[2025-09-14 11:19:58,118] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirC
leanupManager)
[2025-09-14 11:19:58,118] WARN Either no config or no quorum defined in config, running in standa
lone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2025-09-14 11:19:58,123] INFO Log4j 1.2 jmx support not found; jmx disabled. (org.apache.zookeep
er.jmx.ManagedUtil)
[2025-09-14 11:19:58,127] INFO Reading configuration from: config/zookeeper.properties (org.apach
e.zookeeper.server.quorum.QuorumPeerConfig)
[2025-09-14 11:19:58,127] WARN config/zookeeper.properties is relative. Prepend ./ to indicate th
at you're sure! (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-09-14 11:19:58,128] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quo
rum.QuorumPeerConfig)
[2025-09-14 11:19:58,129] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.Qu
orumPeerConfig)
[2025-09-14 11:19:58,129] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.
QuorumPeerConfig)
[2025-09-14 11:19:58,129] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.Def
aultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-09-14 11:19:58,129] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2025-09-14 11:19:58,252] INFO ServerMetrics initialized with provider org.apache.zookeeper.metri
cs.impl.DefaultMetricsProvider@327471b5 (org.apache.zookeeper.server.ServerMetrics)
[2025-09-14 11:19:58,267] INFO ACL digest algorithm is: SHA1 (org.apache.zookeeper.server.auth.Di
gestAuthenticationProvider)
[2025-09-14 11:19:58,267] INFO zookeeper.DigestAuthenticationProvider.enabled = true (org.apache.
zookeeper.server.auth.DigestAuthenticationProvider)
```

START KAFKA BROKER



```
hadoop@hadoop-VirtualBox:~$ cd ~/Downloads/kafka_2.13-3.7.0
hadoop@hadoop-VirtualBox:~/Downloads/kafka_2.13-3.7.0$ bin/kafka-server-start.sh config/server.pr
operties
[2025-09-14 11:21:09,744] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log
4jControllerRegistration$)
[2025-09-14 11:21:10,919] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disa
ble client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)
[2025-09-14 11:21:11,264] INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.co
mmon.utils.LoggingSignalHandler)
[2025-09-14 11:21:11,272] INFO starting (kafka.server.KafkaServer)
[2025-09-14 11:21:11,273] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServe
r)
[2025-09-14 11:21:11,382] INFO [ZooKeeperClient Kafka server] Initializing a new session to local
host:2181. (kafka.zookeeper.ZooKeeperClient)
[2025-09-14 11:21:11,403] INFO Client environment:zookeeper.version=3.8.3-6ad6d364c7c0bcf0de452d5
4ebefa3058098ab56, built on 2023-10-05 10:34 UTC (org.apache.zookeeper.ZooKeeper)
[2025-09-14 11:21:11,403] INFO Client environment:host.name=hadoop-VirtualBox (org.apache.zookeep
er.ZooKeeper)
[2025-09-14 11:21:11,403] INFO Client environment:java.version=1.8.0_91 (org.apache.zookeeper.Zoo
Keeper)
[2025-09-14 11:21:11,403] INFO Client environment:java.vendor=Oracle Corporation (org.apache.zook
eeper.ZooKeeper)
[2025-09-14 11:21:11,403] INFO Client environment:java.home=/usr/local/java/jre (org.apache.zooke
eper.ZooKeeper)
[2025-09-14 11:21:11,403] INFO Client environment:java.class.path=/home/hadoop/Downloads/kafka_2.
13-3.7.0/bin/../libs/activation-1.1.1.jar:/home/hadoop/Downloads/kafka_2.13-3.7.0/bin/../libs/aop
alliance-repackaged-2.6.1.jar:/home/hadoop/Downloads/kafka_2.13-3.7.0/bin/../libs/argparse4j-0.7.
0.jar:/home/hadoop/Downloads/kafka_2.13-3.7.0/bin/../libs/audience-annotations-0.12.0.jar:/home/h
adoop/Downloads/kafka_2.13-3.7.0/bin/../libs/caffeine-2.9.3.jar:/home/hadoop/Downloads/kafka_2.13
-3.7.0/bin/../libs/checker-qual-3.19.0.jar:/home/hadoop/Downloads/kafka_2.13-3.7.0/bin/../libs/co
mmons-beanutils-1.9.4.jar:/home/hadoop/Downloads/kafka_2.13-3.7.0/bin/../libs/commons-cli-1.4.jar
:/home/hadoop/Downloads/kafka_2.13-3.7.0/bin/../libs/commons-collections-3.2.2.jar:/home/hadoop/D
ownloads/kafka_2.13-3.7.0/bin/../libs/commons-digester-2.1.jar:/home/hadoop/Downloads/kafka_2.13-
3.7.0/bin/../libs/commons-io-2.11.0.jar:/home/hadoop/Downloads/kafka_2.13-3.7.0/bin/../libs/commo
ns-lang3-3.8.1.jar:/home/hadoop/Downloads/kafka_2.13-3.7.0/bin/../libs/commons-logging-1.2.jar:/h
ome/hadoop/Downloads/kafka_2.13-3.7.0/bin/../libs/commons-validator-1.7.jar:/home/hadoop/Download
s/kafka_2.13-3.7.0/bin/../libs/connect-api-3.7.0.jar:/home/hadoop/Downloads/kafka_2.13-3.7.0/bin/
../libs/connect-basic-auth-extension-3.7.0.jar:/home/hadoop/Downloads/kafka_2.13-3.7.0/bin/../lib
s/connect-json-3.7.0.jar:/home/hadoop/Downloads/kafka_2.13-3.7.0/bin/../libs/connect-mirror-3.7.0
.jar:/home/hadoop/Downloads/kafka_2.13-3.7.0/bin/../libs/connect-mirror-client-3.7.0.jar:/home/ha
doop/Downloads/kafka_2.13-3.7.0/bin/../libs/connect-runtime-3.7.0.jar:/home/hadoop/Downloads/kafk
a_2.13-3.7.0/bin/../libs/connect-transforms-3.7.0.jar:/home/hadoop/Downloads/kafka_2.13-3.7.0/bin
/../libs/error_prone_annotations-2.10.0.jar:/home/hadoop/Downloads/kafka_2.13-3.7.0/bin/../libs/h
k2-api-2.6.1.jar:/home/hadoop/Downloads/kafka_2.13-3.7.0/bin/../libs/hk2-locator-2.6.1.jar:/home/
hadoop/Downloads/kafka_2.13-3.7.0/bin/../libs/hk2-utils-2.6.1.jar:/home/hadoop/Downloads/kafka_2
```

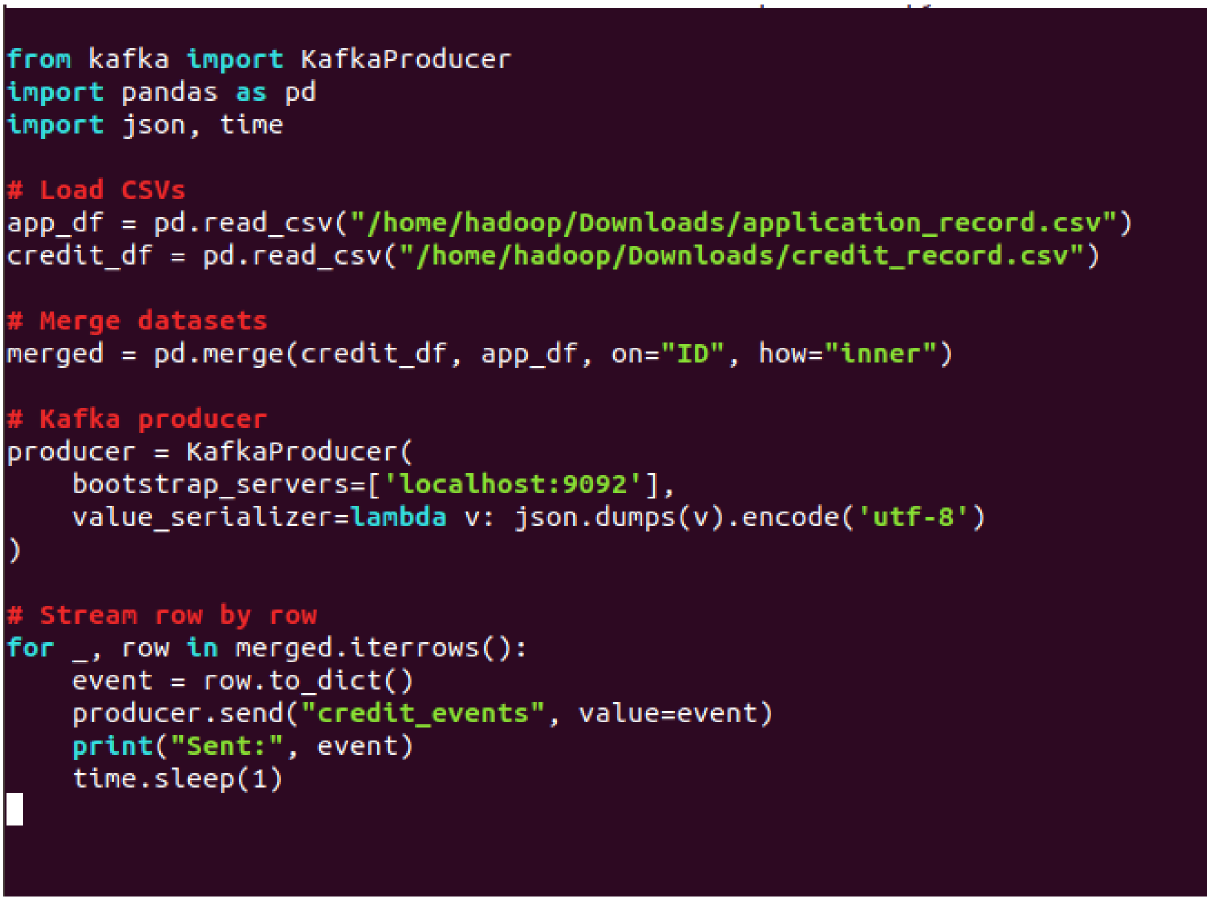
CREATING TOPIC AND VIEWING THEM =>



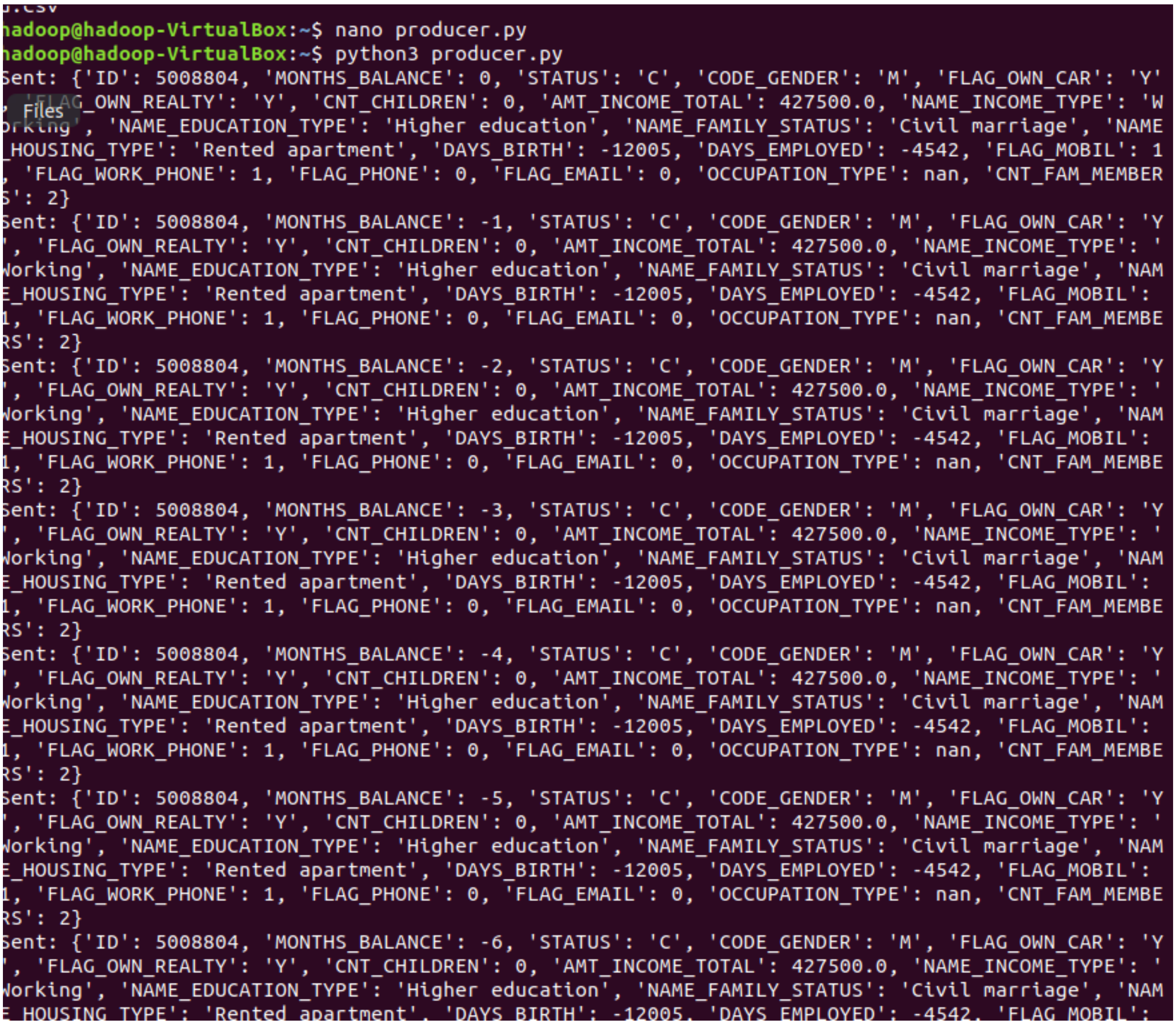
hadoop@hadoop-VirtualBox: ~/...
hadoop@hadoop-VirtualBox: ~/...
hadoop@hadoop-VirtualBox: ~/...

```
hadoop@hadoop-VirtualBox:~/Downloads/kafka_2.13-3.7.0$ bin/kafka-topics.sh --create --topic credi
t_events --bootstrap-server localhost:9092 --partitions 3 --replication-factor 1
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could
 collide. To avoid issues it is best to use either, but not both.
Created topic credit_events.
hadoop@hadoop-VirtualBox:~/Downloads/kafka_2.13-3.7.0$ bin/kafka-topics.sh --list --bootstrap-ser
ver localhost:9092
credit_events
```

```
hadoop@hadoop-VirtualBox:~$ nano producer.py
```

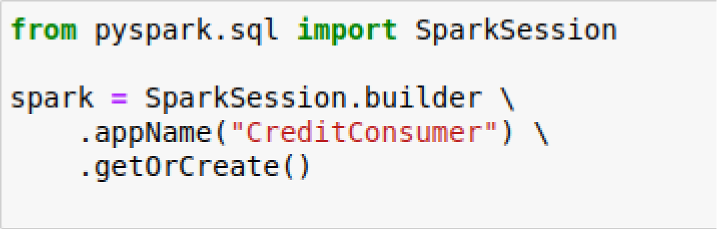


```
from kafka import KafkaProducer
import pandas as pd
import json, time

# Load CSVs
app_df = pd.read_csv("/home/hadoop/Downloads/application_record.csv")
credit_df = pd.read_csv("/home/hadoop/Downloads/credit_record.csv")

# Merge datasets
merged = pd.merge(credit_df, app_df, on="ID", how="inner")

# Kafka producer
producer = KafkaProducer(
    bootstrap_servers=['localhost:9092'],
    value_serializer=lambda v: json.dumps(v).encode('utf-8')
)

# Stream row by row
for _, row in merged.iterrows():
    event = row.to_dict()
    producer.send("credit_events", value=event)
    print("Sent:", event)
    time.sleep(1)
```

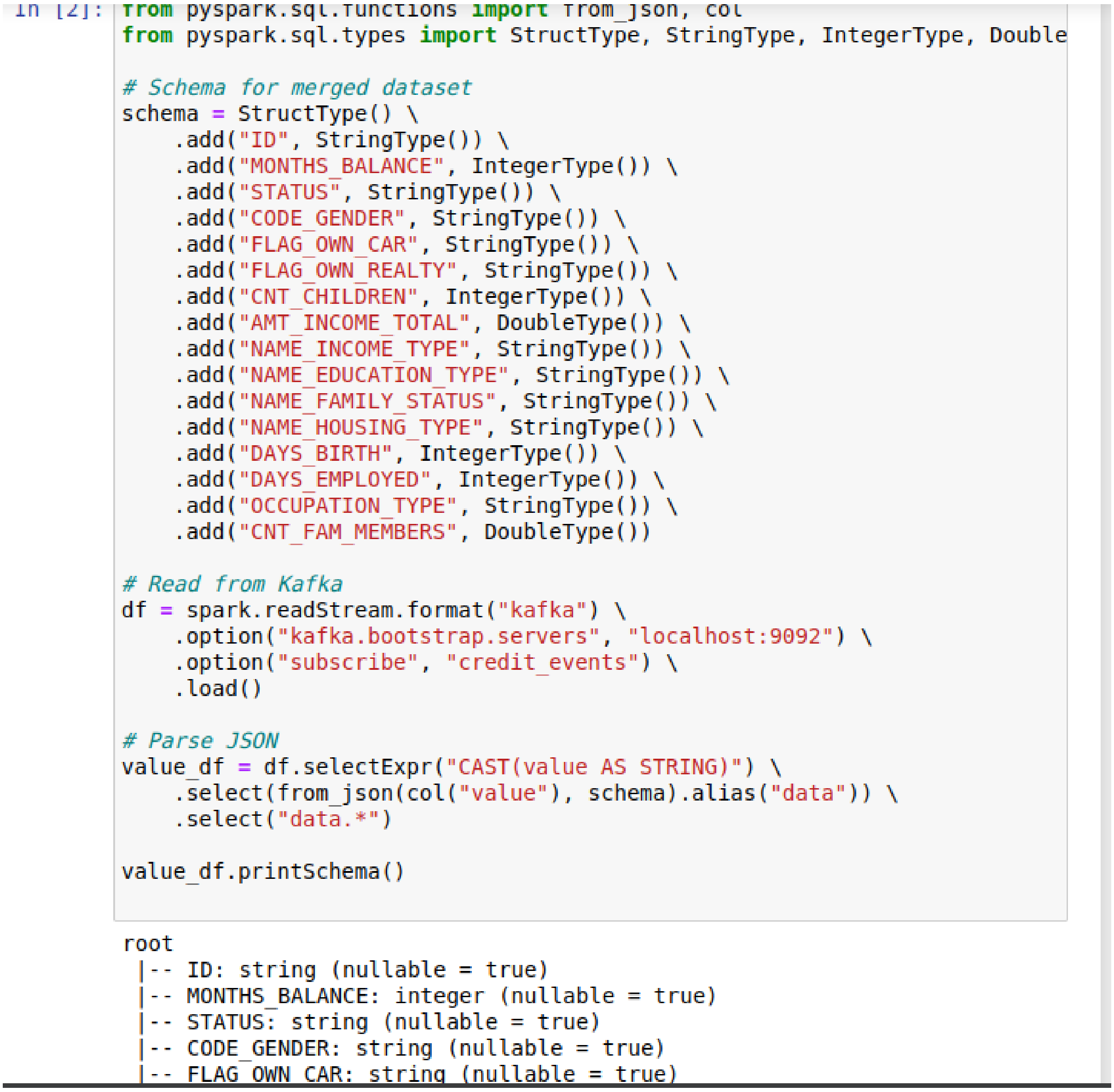


```
d.csv
hadoop@hadoop-VirtualBox:~$ nano producer.py
hadoop@hadoop-VirtualBox:~$ python3 producer.py
Sent: {'ID': 5008804, 'MONTHS_BALANCE': 0, 'STATUS': 'C', 'CODE_GENDER': 'M', 'FLAG_OWN_CAR': 'Y'
, 'FLAG_OWN_REALTY': 'Y', 'CNT_CHILDREN': 0, 'AMT_INCOME_TOTAL': 427500.0, 'NAME_INCOME_TYPE': 'W
orking', 'NAME_EDUCATION_TYPE': 'Higher education', 'NAME_FAMILY_STATUS': 'Civil marriage', 'NAME
_HOUSING_TYPE': 'Rented apartment', 'DAYS_BIRTH': -12005, 'DAYS_EMPLOYED': -4542, 'FLAG_MOBIL': 1
, 'FLAG_WORK_PHONE': 1, 'FLAG_PHONE': 0, 'FLAG_EMAIL': 0, 'OCCUPATION_TYPE': nan, 'CNT_FAM_MEMBER
S': 2}
Sent: {'ID': 5008804, 'MONTHS_BALANCE': -1, 'STATUS': 'C', 'CODE_GENDER': 'M', 'FLAG_OWN_CAR': 'Y
', 'FLAG_OWN_REALTY': 'Y', 'CNT_CHILDREN': 0, 'AMT_INCOME_TOTAL': 427500.0, 'NAME_INCOME_TYPE': '
Working', 'NAME_EDUCATION_TYPE': 'Higher education', 'NAME_FAMILY_STATUS': 'Civil marriage', 'NAM
E_HOUSING_TYPE': 'Rented apartment', 'DAYS_BIRTH': -12005, 'DAYS_EMPLOYED': -4542, 'FLAG_MOBIL':
1, 'FLAG_WORK_PHONE': 1, 'FLAG_PHONE': 0, 'FLAG_EMAIL': 0, 'OCCUPATION_TYPE': nan, 'CNT_FAM_MEMBE
RS': 2}
Sent: {'ID': 5008804, 'MONTHS_BALANCE': -2, 'STATUS': 'C', 'CODE_GENDER': 'M', 'FLAG_OWN_CAR': 'Y
', 'FLAG_OWN_REALTY': 'Y', 'CNT_CHILDREN': 0, 'AMT_INCOME_TOTAL': 427500.0, 'NAME_INCOME_TYPE': '
Working', 'NAME_EDUCATION_TYPE': 'Higher education', 'NAME_FAMILY_STATUS': 'Civil marriage', 'NAM
E_HOUSING_TYPE': 'Rented apartment', 'DAYS_BIRTH': -12005, 'DAYS_EMPLOYED': -4542, 'FLAG_MOBIL':
1, 'FLAG_WORK_PHONE': 1, 'FLAG_PHONE': 0, 'FLAG_EMAIL': 0, 'OCCUPATION_TYPE': nan, 'CNT_FAM_MEMBE
RS': 2}
Sent: {'ID': 5008804, 'MONTHS_BALANCE': -3, 'STATUS': 'C', 'CODE_GENDER': 'M', 'FLAG_OWN_CAR': 'Y
', 'FLAG_OWN_REALTY': 'Y', 'CNT_CHILDREN': 0, 'AMT_INCOME_TOTAL': 427500.0, 'NAME_INCOME_TYPE': '
Working', 'NAME_EDUCATION_TYPE': 'Higher education', 'NAME_FAMILY_STATUS': 'Civil marriage', 'NAM
E_HOUSING_TYPE': 'Rented apartment', 'DAYS_BIRTH': -12005, 'DAYS_EMPLOYED': -4542, 'FLAG_MOBIL':
1, 'FLAG_WORK_PHONE': 1, 'FLAG_PHONE': 0, 'FLAG_EMAIL': 0, 'OCCUPATION_TYPE': nan, 'CNT_FAM_MEMBE
RS': 2}
Sent: {'ID': 5008804, 'MONTHS_BALANCE': -4, 'STATUS': 'C', 'CODE_GENDER': 'M', 'FLAG_OWN_CAR': 'Y
', 'FLAG_OWN_REALTY': 'Y', 'CNT_CHILDREN': 0, 'AMT_INCOME_TOTAL': 427500.0, 'NAME_INCOME_TYPE': '
Working', 'NAME_EDUCATION_TYPE': 'Higher education', 'NAME_FAMILY_STATUS': 'Civil marriage', 'NAM
E_HOUSING_TYPE': 'Rented apartment', 'DAYS_BIRTH': -12005, 'DAYS_EMPLOYED': -4542, 'FLAG_MOBIL':
1, 'FLAG_WORK_PHONE': 1, 'FLAG_PHONE': 0, 'FLAG_EMAIL': 0, 'OCCUPATION_TYPE': nan, 'CNT_FAM_MEMBE
RS': 2}
Sent: {'ID': 5008804, 'MONTHS_BALANCE': -5, 'STATUS': 'C', 'CODE_GENDER': 'M', 'FLAG_OWN_CAR': 'Y
', 'FLAG_OWN_REALTY': 'Y', 'CNT_CHILDREN': 0, 'AMT_INCOME_TOTAL': 427500.0, 'NAME_INCOME_TYPE': '
Working', 'NAME_EDUCATION_TYPE': 'Higher education', 'NAME_FAMILY_STATUS': 'Civil marriage', 'NAM
E_HOUSING_TYPE': 'Rented apartment', 'DAYS_BIRTH': -12005, 'DAYS_EMPLOYED': -4542, 'FLAG_MOBIL':
1, 'FLAG_WORK_PHONE': 1, 'FLAG_PHONE': 0, 'FLAG_EMAIL': 0, 'OCCUPATION_TYPE': nan, 'CNT_FAM_MEMBE
RS': 2}
Sent: {'ID': 5008804, 'MONTHS_BALANCE': -6, 'STATUS': 'C', 'CODE_GENDER': 'M', 'FLAG_OWN_CAR': 'Y
', 'FLAG_OWN_REALTY': 'Y', 'CNT_CHILDREN': 0, 'AMT_INCOME_TOTAL': 427500.0, 'NAME_INCOME_TYPE': '
Working', 'NAME_EDUCATION_TYPE': 'Higher education', 'NAME_FAMILY_STATUS': 'Civil marriage', 'NAM
E_HOUSING_TYPE': 'Rented apartment', 'DAYS_BIRTH': -12005, 'DAYS_EMPLOYED': -4542, 'FLAG_MOBIL':
```

Jupyter Notebook In VM

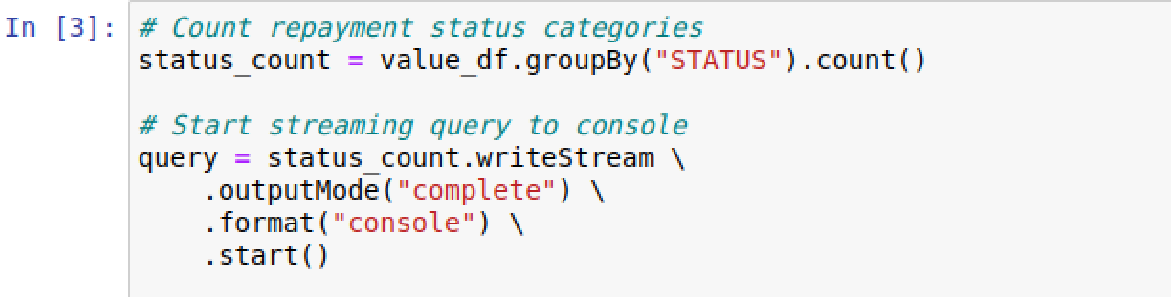


```python
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("CreditConsumer") \
    .getOrCreate()
```

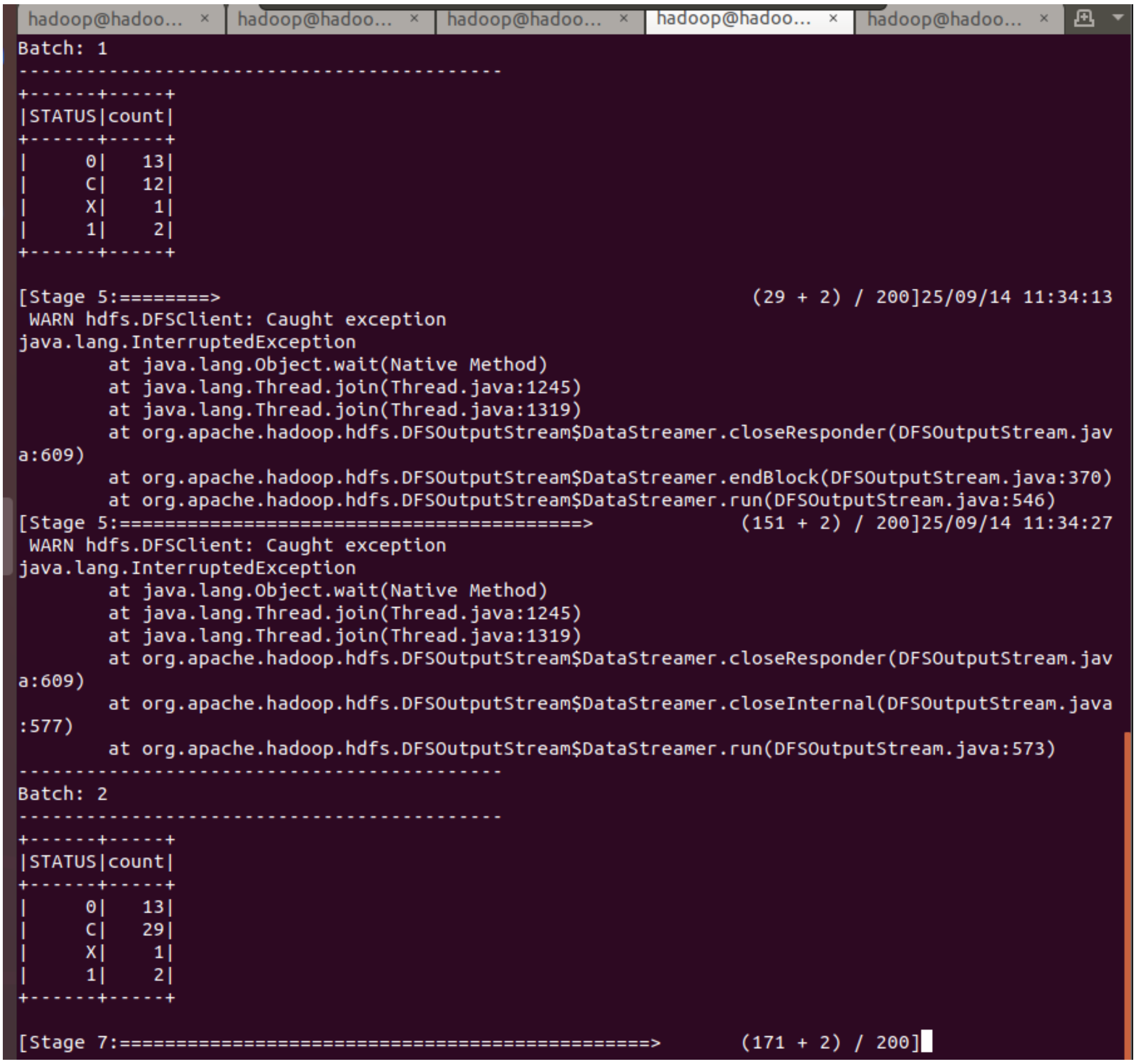


```
In [2]: from pyspark.sql.functions import from_json, col
        from pyspark.sql.types import StructType, StringType, IntegerType, Double

        # Schema for merged dataset
        schema = StructType() \
            .add("ID", StringType()) \
            .add("MONTHS_BALANCE", IntegerType()) \
            .add("STATUS", StringType()) \
            .add("CODE_GENDER", StringType()) \
            .add("FLAG_OWN_CAR", StringType()) \
            .add("FLAG_OWN_REALTY", StringType()) \
            .add("CNT_CHILDREN", IntegerType()) \
            .add("AMT_INCOME_TOTAL", DoubleType()) \
            .add("NAME_INCOME_TYPE", StringType()) \
            .add("NAME_EDUCATION_TYPE", StringType()) \
            .add("NAME_FAMILY_STATUS", StringType()) \
            .add("NAME_HOUSING_TYPE", StringType()) \
            .add("DAYS_BIRTH", IntegerType()) \
            .add("DAYS_EMPLOYED", IntegerType()) \
            .add("OCCUPATION_TYPE", StringType()) \
            .add("CNT_FAM_MEMBERS", DoubleType())

        # Read from Kafka
        df = spark.readStream.format("kafka") \
            .option("kafka.bootstrap.servers", "localhost:9092") \
            .option("subscribe", "credit_events") \
            .load()

        # Parse JSON
        value_df = df.selectExpr("CAST(value AS STRING)") \
            .select(from_json(col("value"), schema).alias("data")) \
            .select("data.*")

        value_df.printSchema()

        root
         |-- ID: string (nullable = true)
         |-- MONTHS_BALANCE: integer (nullable = true)
         |-- STATUS: string (nullable = true)
         |-- CODE_GENDER: string (nullable = true)
         |-- FLAG_OWN_CAR: string (nullable = true)
```

BATCH STREAMING PROCESS



```
In [3]: # Count repayment status categories
        status_count = value_df.groupBy("STATUS").count()

        # Start streaming query to console
        query = status_count.writeStream \
            .outputMode("complete") \
            .format("console") \
            .start()
```



```
Batch: 1
-------------------------------------------
+------+-----+
|STATUS|count|
+------+-----+
|     0|   13|
|     C|   12|
|     X|    1|
|     1|    2|
+------+-----+

[Stage 5:========>                                        (29 + 2) / 200]25/09/14 11:34:13
 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1245)
        at java.lang.Thread.join(Thread.java:1319)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.jav
a:609)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:370)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:546)
[Stage 5:========================================>         (151 + 2) / 200]25/09/14 11:34:27
 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1245)
        at java.lang.Thread.join(Thread.java:1319)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.jav
a:609)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeInternal(DFSOutputStream.java
:577)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:573)
-------------------------------------------
Batch: 2
-------------------------------------------
+------+-----+
|STATUS|count|
+------+-----+
|     0|   13|
|     C|   29|
|     X|    1|
|     1|    2|
+------+-----+

[Stage 7:==============================================>       (171 + 2) / 200]
```

2. Data Transformation Layer (PySpark)

Goal:

Cleanse, enrich, and join raw data to create meaningful datasets for downstream analytics.

Steps:

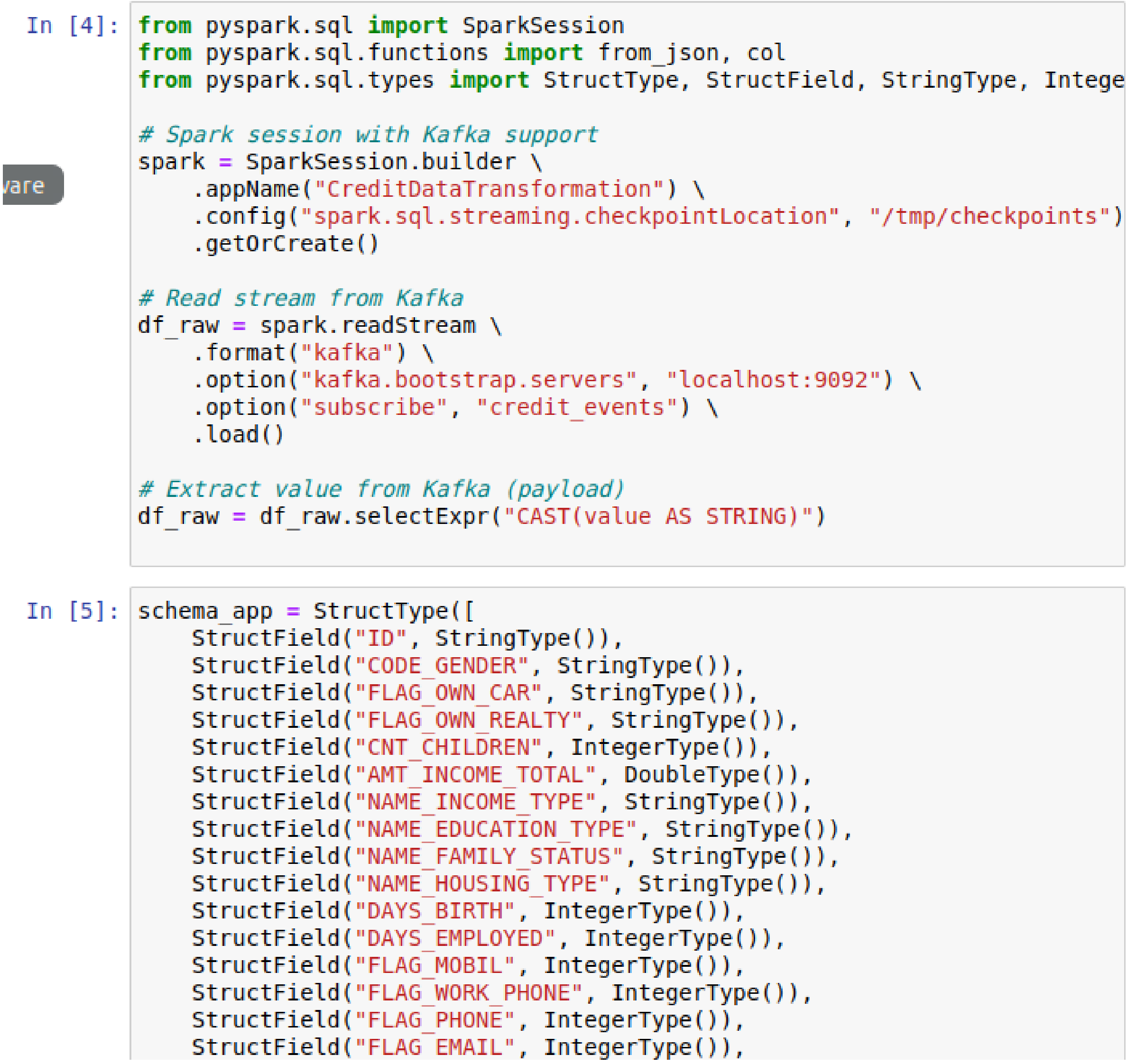
- Read streaming data from Kafka using Pyspark Structured Streaming.
- Parse incoming JSON or CSV payloads and convert to Dataframes.
- Perform transformations:
  - Data type casting
  - Handling missing values

    - Joining with static datasets (if multiple datasets are present)
    - Data Aggregations if required
    - Time-windowed operations (e.g. Rides Booking)
- Write transformed data into:
    - Delta Lake / HDFS / Parquet format for historical records
    - Temporary views for reporting

Technologies:

PySpark, HDFS, Structured Streaming

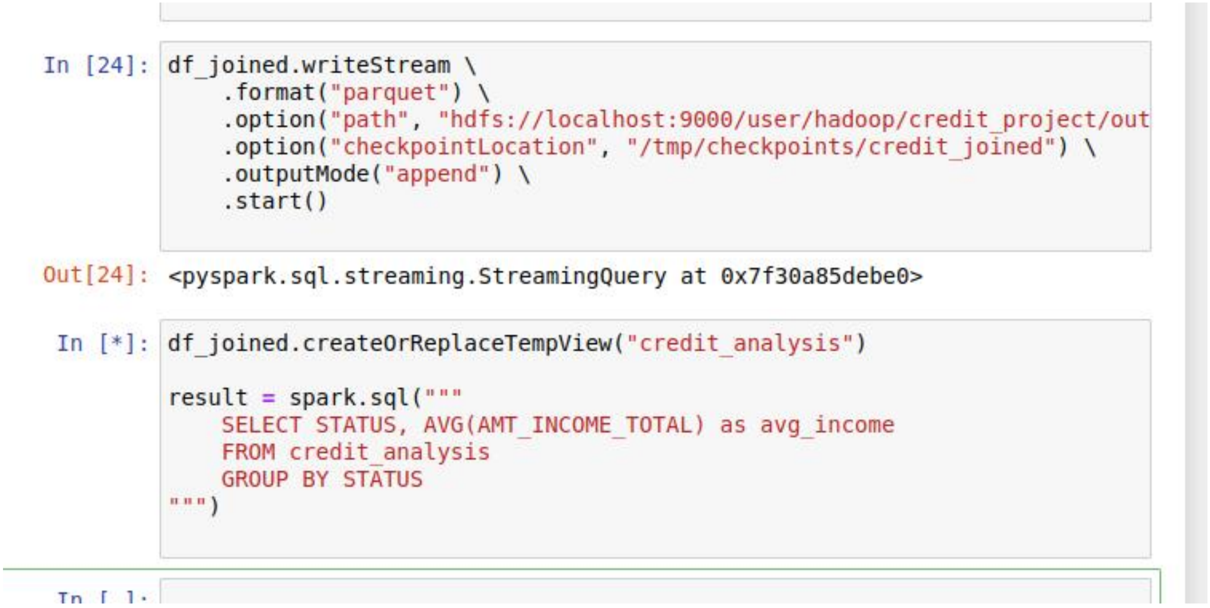
Outputs:



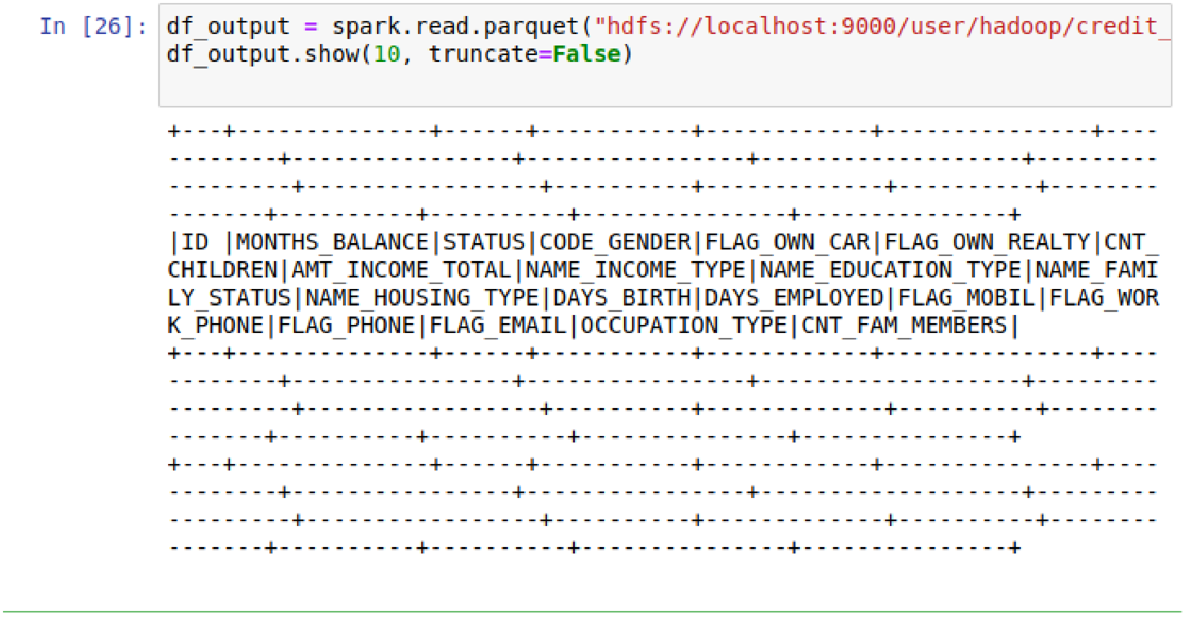
In [4]:
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json, col
from pyspark.sql.types import StructType, StructField, StringType, Intege

# Spark session with Kafka support
spark = SparkSession.builder \
    .appName("CreditDataTransformation") \
    .config("spark.sql.streaming.checkpointLocation", "/tmp/checkpoints")
    .getOrCreate()

# Read stream from Kafka
df_raw = spark.readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "credit_events") \
    .load()

# Extract value from Kafka (payload)
df_raw = df_raw.selectExpr("CAST(value AS STRING)")
```

In [5]:
```
schema_app = StructType([
    StructField("ID", StringType()),
    StructField("CODE_GENDER", StringType()),
    StructField("FLAG_OWN_CAR", StringType()),
    StructField("FLAG_OWN_REALTY", StringType()),
    StructField("CNT_CHILDREN", IntegerType()),
    StructField("AMT_INCOME_TOTAL", DoubleType()),
    StructField("NAME_INCOME_TYPE", StringType()),
    StructField("NAME_EDUCATION_TYPE", StringType()),
    StructField("NAME_FAMILY_STATUS", StringType()),
    StructField("NAME_HOUSING_TYPE", StringType()),
    StructField("DAYS_BIRTH", IntegerType()),
    StructField("DAYS_EMPLOYED", IntegerType()),
    StructField("FLAG_MOBIL", IntegerType()),
    StructField("FLAG_WORK_PHONE", IntegerType()),
    StructField("FLAG_PHONE", IntegerType()),
    StructField("FLAG_EMAIL", IntegerType()),
```

AFTER PARSING, CLEANING AND STORING IT IN PARQUET IN HDFS=>



```
In [24]: df_joined.writeStream \
             .format("parquet") \
             .option("path", "hdfs://localhost:9000/user/hadoop/credit_project/out
             .option("checkpointLocation", "/tmp/checkpoints/credit_joined") \
             .outputMode("append") \
             .start()

Out[24]: <pyspark.sql.streaming.StreamingQuery at 0x7f30a85debe0>

In [*]: df_joined.createOrReplaceTempView("credit_analysis")

        result = spark.sql("""
            SELECT STATUS, AVG(AMT_INCOME_TOTAL) as avg_income
            FROM credit_analysis
            GROUP BY STATUS
        """)

In [ ]:
```



```
In [26]: df_output = spark.read.parquet("hdfs://localhost:9000/user/hadoop/credit_
         df_output.show(10, truncate=False)

         +---+--------------+------+-----------+------------+---------------+----
         --------+----------------+----------------+-------------------+---------
         ---------+-----------------+----------+-------------+----------+--------
         -------+----------+----------+---------------+---------------+
         |ID |MONTHS_BALANCE|STATUS|CODE_GENDER|FLAG_OWN_CAR|FLAG_OWN_REALTY|CNT_
         CHILDREN|AMT_INCOME_TOTAL|NAME_INCOME_TYPE|NAME_EDUCATION_TYPE|NAME_FAMI
         LY_STATUS|NAME_HOUSING_TYPE|DAYS_BIRTH|DAYS_EMPLOYED|FLAG_MOBIL|FLAG_WOR
         K_PHONE|FLAG_PHONE|FLAG_EMAIL|OCCUPATION_TYPE|CNT_FAM_MEMBERS|
         +---+--------------+------+-----------+------------+---------------+----
         --------+----------------+----------------+-------------------+---------
         ---------+-----------------+----------+-------------+----------+--------
         -------+----------+----------+---------------+---------------+
         +---+--------------+------+-----------+------------+---------------+----
         --------+----------------+----------------+-------------------+---------
         ---------+-----------------+----------+-------------+----------+--------
         -------+----------+----------+---------------+---------------+
```

3. Creation of Materialized Views

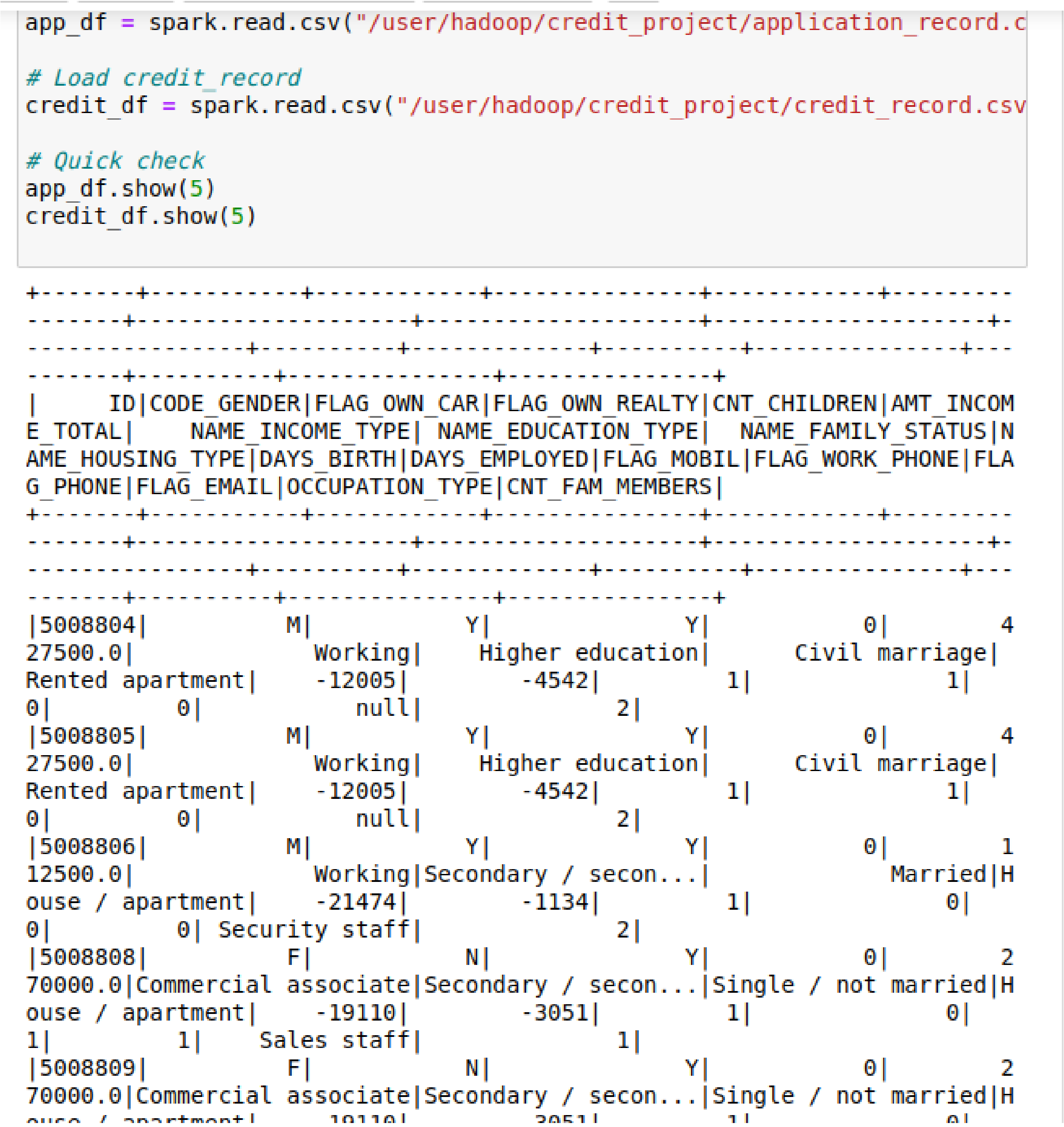
Now that our data is transformed and stored, we will create materialized views for efficient querying.

- **Define Views: Identify key metrics and insights that the business needs. Define SQL queries to create materialized views based on these requirements.**
- **Optimization: Optimize these views for performance, ensuring they are quick to query and provide accurate results.**

- **Use insightful queries to find good insights combining all data together.**
- **Calculate bad customer rate**
- **Survival Analysis**
- **Cumulative % of Bad Customers Analysis**
- **Observe Window Analysis**

Tools – Pyspark, Spark SQL

Outputs:



```
app_df = spark.read.csv("/user/hadoop/credit_project/application_record.c

# Load credit_record
credit_df = spark.read.csv("/user/hadoop/credit_project/credit_record.csv

# Quick check
app_df.show(5)
credit_df.show(5)
```

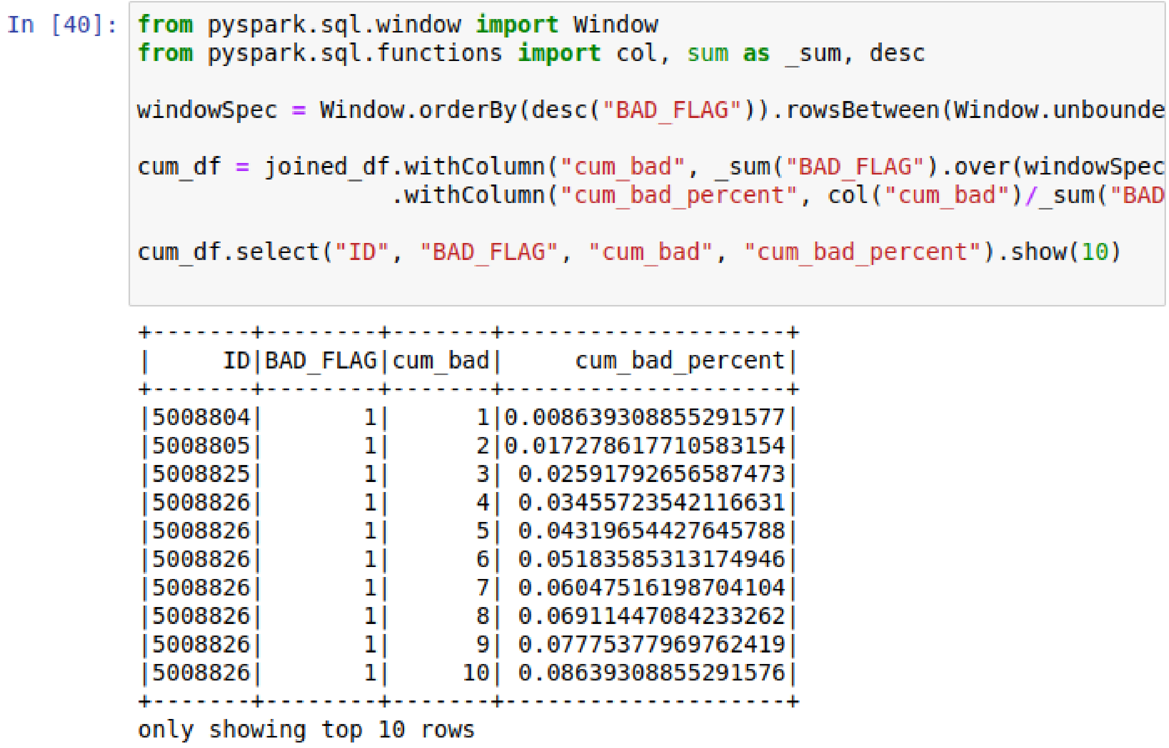
```
+-------+-----------+------------+---------------+------------+---------
-------+--------------------+--------------------+--------------------+-
----------------+----------+-------------+----------+---------------+---
-------+----------+---------------+---------------+
|     ID|CODE_GENDER|FLAG_OWN_CAR|FLAG_OWN_REALTY|CNT_CHILDREN|AMT_INCOM
E_TOTAL|    NAME_INCOME_TYPE| NAME_EDUCATION_TYPE|  NAME_FAMILY_STATUS|N
AME_HOUSING_TYPE|DAYS_BIRTH|DAYS_EMPLOYED|FLAG_MOBIL|FLAG_WORK_PHONE|FLA
G_PHONE|FLAG_EMAIL|OCCUPATION_TYPE|CNT_FAM_MEMBERS|
+-------+-----------+------------+---------------+------------+---------
-------+--------------------+--------------------+--------------------+-
----------------+----------+-------------+----------+---------------+---
-------+----------+---------------+---------------+
|5008804|          M|           Y|              Y|           0|        4
27500.0|             Working|    Higher education|      Civil marriage|
Rented apartment|    -12005|        -4542|        1|               1|
0|         0|          null|              2|
|5008805|          M|           Y|              Y|           0|        4
27500.0|             Working|    Higher education|      Civil marriage|
Rented apartment|    -12005|        -4542|        1|               1|
0|         0|          null|              2|
|5008806|          M|           Y|              Y|           0|        1
12500.0|             Working|Secondary / secon...|             Married|H
ouse / apartment|    -21474|        -1134|        1|               0|
0|         0| Security staff|              2|
|5008808|          F|           N|              Y|           0|        2
70000.0|Commercial associate|Secondary / secon...|Single / not married|H
ouse / apartment|    -19110|        -3051|        1|               0|
1|         1|    Sales staff|              1|
|5008809|          F|           N|              Y|           0|        2
70000.0|Commercial associate|Secondary / secon...|Single / not married|H
ouse / apartment|    -19110|        -3051|        1|               0|
1|         1|   Sales staff|             1|
+-------+-----------+-----------+--------------+------------+--------
-------+-------------------+-------------------+-------------------+-
----------------+---------+------------+----------+--------------+---
-------+----------+--------------+--------------+
only showing top 5 rows

+-------+-------------+------+
|     ID|MONTHS_BALANCE|STATUS|
+-------+-------------+------+
|5001711|            0|     X|
|5001711|           -1|     0|
|5001711|           -2|     0|
|5001711|           -3|     0|
|5001712|            0|     C|
+-------+-------------+------+
only showing top 5 rows
```

In [39]:

```
spark.sql("""
SELECT
    COUNT(*) AS total_customers,
    SUM(BAD_FLAG) AS bad_customers,
    ROUND(SUM(BAD_FLAG)/COUNT(*)*100,2) AS bad_customer_rate
FROM credit_analysis_view
""").show()
```

```
+---------------+-------------+-----------------+
|total_customers|bad_customers|bad_customer_rate|
+---------------+-------------+-----------------+
|         777715|        11575|             1.49|
+---------------+-------------+-----------------+
```

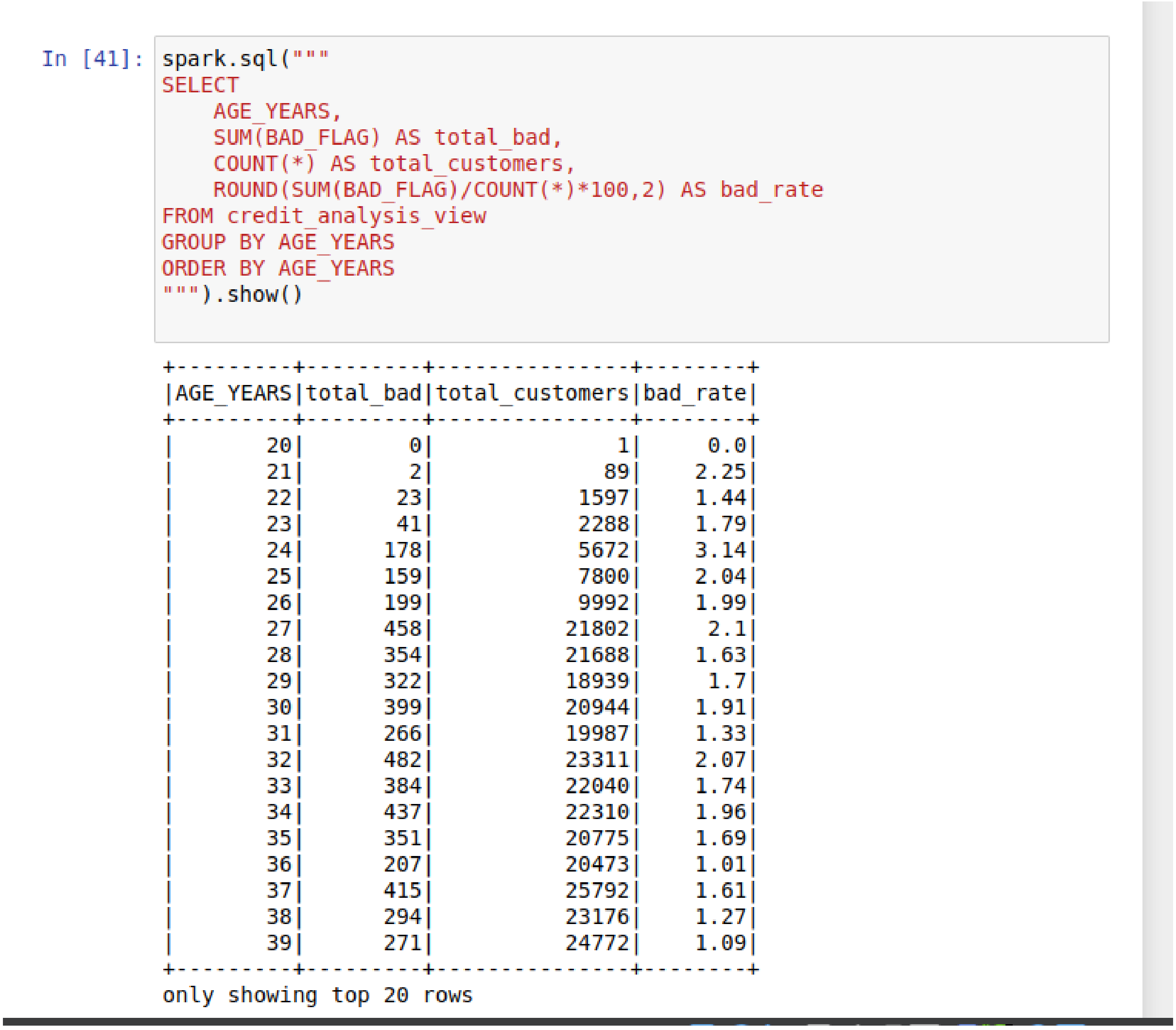
Bad Customer



In [40]:

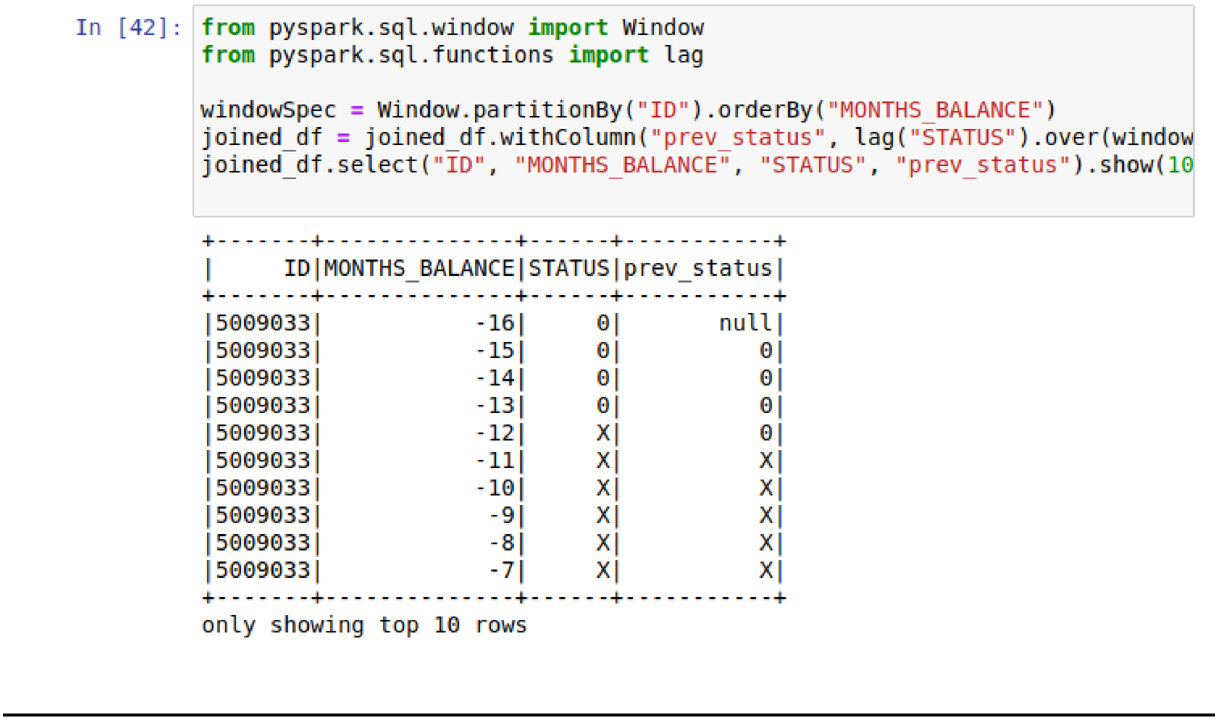
```
from pyspark.sql.window import Window
from pyspark.sql.functions import col, sum as _sum, desc

windowSpec = Window.orderBy(desc("BAD_FLAG")).rowsBetween(Window.unbounde

cum_df = joined_df.withColumn("cum_bad", _sum("BAD_FLAG").over(windowSpec
                  .withColumn("cum_bad_percent", col("cum_bad")/_sum("BAD

cum_df.select("ID", "BAD_FLAG", "cum_bad", "cum_bad_percent").show(10)
```

```
+-------+--------+-------+--------------------+
|     ID|BAD_FLAG|cum_bad|     cum_bad_percent|
+-------+--------+-------+--------------------+
|5008804|       1|      1|0.008639308855291577|
|5008805|       1|      2|0.017278617710583154|
|5008825|       1|      3| 0.02591792656587473|
|5008826|       1|      4| 0.03455723542116631|
|5008826|       1|      5| 0.04319654427645788|
|5008826|       1|      6| 0.05183585313174946|
|5008826|       1|      7| 0.06047516198704104|
|5008826|       1|      8| 0.06911447084233262|
|5008826|       1|      9| 0.07775377969762419|
|5008826|       1|     10| 0.08639308855291576|
+-------+--------+-------+--------------------+
only showing top 10 rows
```



```
In [41]: spark.sql("""
         SELECT
             AGE_YEARS,
             SUM(BAD_FLAG) AS total_bad,
             COUNT(*) AS total_customers,
             ROUND(SUM(BAD_FLAG)/COUNT(*)*100,2) AS bad_rate
         FROM credit_analysis_view
         GROUP BY AGE_YEARS
         ORDER BY AGE_YEARS
         """).show()
```

```
+---------+---------+---------------+--------+
|AGE_YEARS|total_bad|total_customers|bad_rate|
+---------+---------+---------------+--------+
|       20|        0|              1|     0.0|
|       21|        2|             89|    2.25|
|       22|       23|           1597|    1.44|
|       23|       41|           2288|    1.79|
|       24|      178|           5672|    3.14|
|       25|      159|           7800|    2.04|
|       26|      199|           9992|    1.99|
|       27|      458|          21802|     2.1|
|       28|      354|          21688|    1.63|
|       29|      322|          18939|     1.7|
|       30|      399|          20944|    1.91|
|       31|      266|          19987|    1.33|
|       32|      482|          23311|    2.07|
|       33|      384|          22040|    1.74|
|       34|      437|          22310|    1.96|
|       35|      351|          20775|    1.69|
|       36|      207|          20473|    1.01|
|       37|      415|          25792|    1.61|
|       38|      294|          23176|    1.27|
|       39|      271|          24772|    1.09|
+---------+---------+---------------+--------+
only showing top 20 rows
```



```
In [42]: from pyspark.sql.window import Window
         from pyspark.sql.functions import lag

         windowSpec = Window.partitionBy("ID").orderBy("MONTHS_BALANCE")
         joined_df = joined_df.withColumn("prev_status", lag("STATUS").over(window
         joined_df.select("ID", "MONTHS_BALANCE", "STATUS", "prev_status").show(10
```

```
+-------+--------------+------+-----------+
|     ID|MONTHS_BALANCE|STATUS|prev_status|
+-------+--------------+------+-----------+
|5009033|           -16|     0|       null|
|5009033|           -15|     0|          0|
|5009033|           -14|     0|          0|
|5009033|           -13|     0|          0|
|5009033|           -12|     X|          0|
|5009033|           -11|     X|          X|
|5009033|           -10|     X|          X|
|5009033|            -9|     X|          X|
|5009033|            -8|     X|          X|
|5009033|            -7|     X|          X|
+-------+--------------+------+-----------+
only showing top 10 rows
```

4. Visualization Layer (Plotly or Power BI)

Goal:

Upload Data in Mysql or NoSQL Read Dataset using Python or Pyspark and Build a visualization layer for business stakeholders to monitor key KPIs and trends using Python, Pyspark, Plotly or PowerBI (Optional).

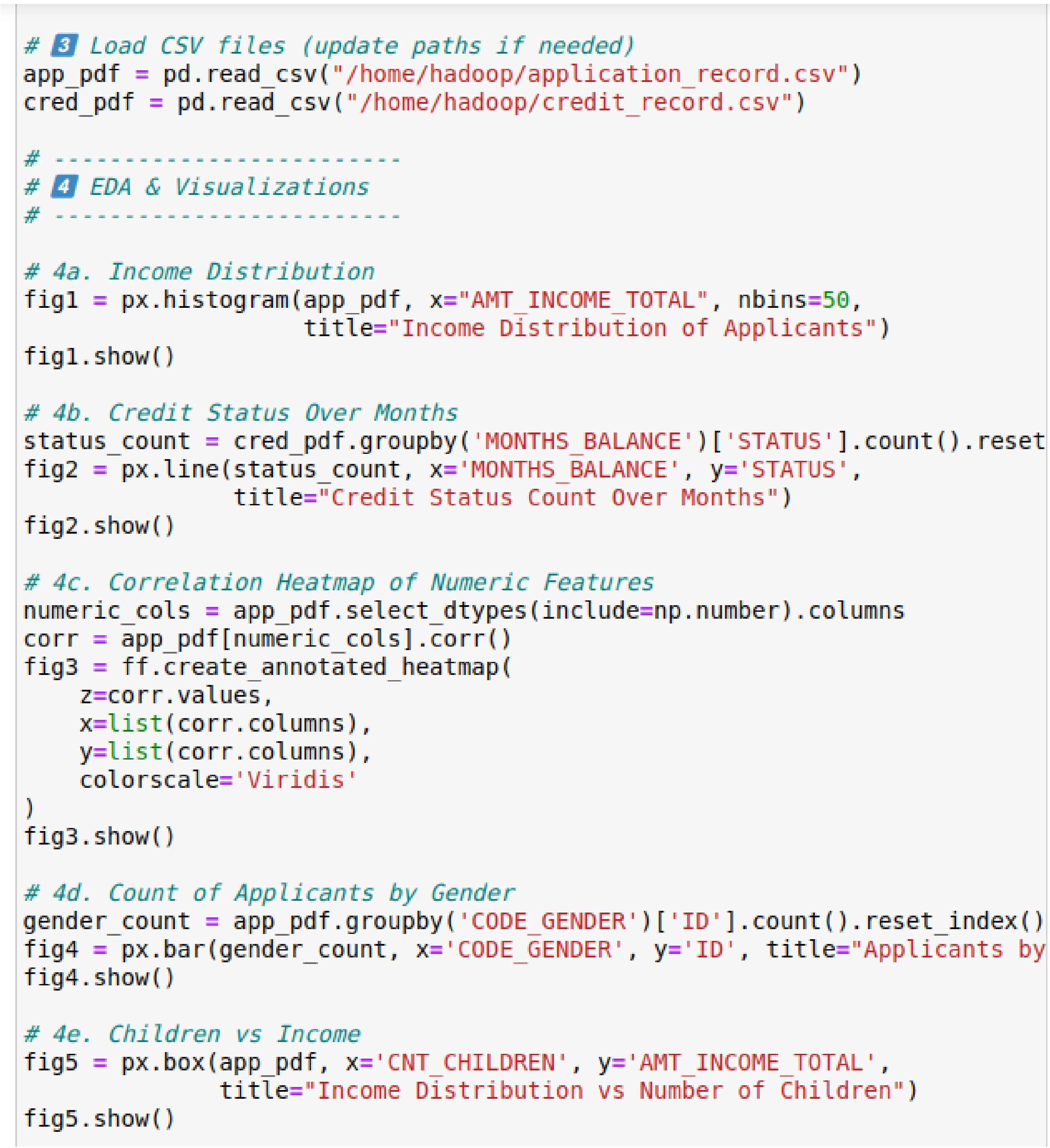
Steps:

- **Analyse the data and perform EDA to find hidden insights.**

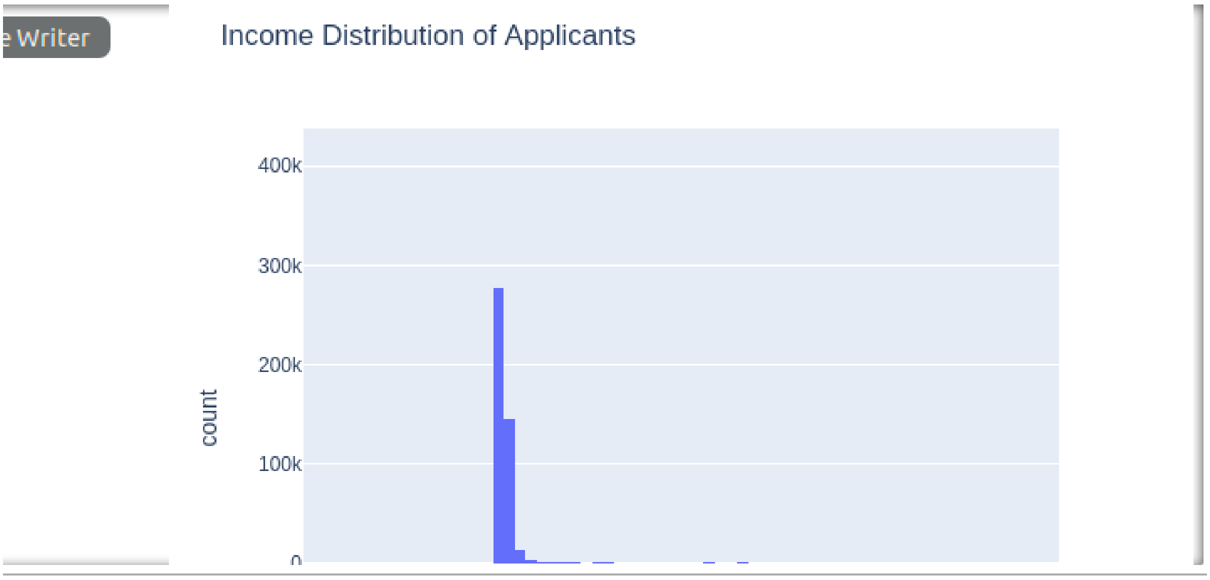
Technologies:

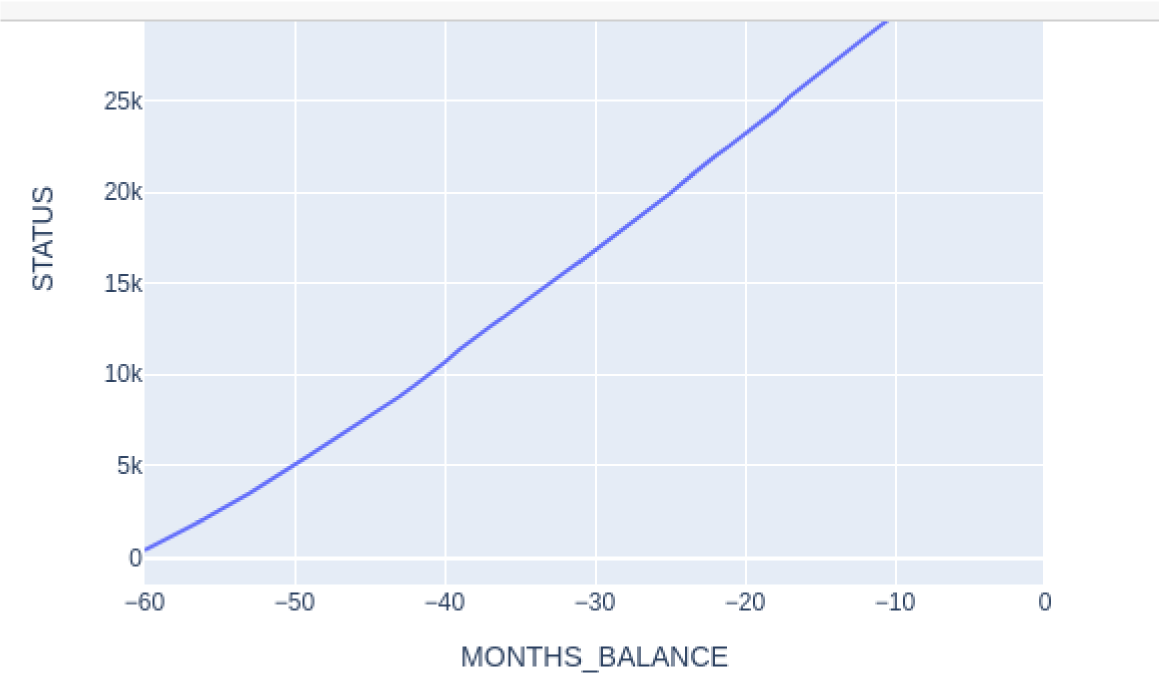
Plotly / Power BI (Mysql or MongoDB for Storage)

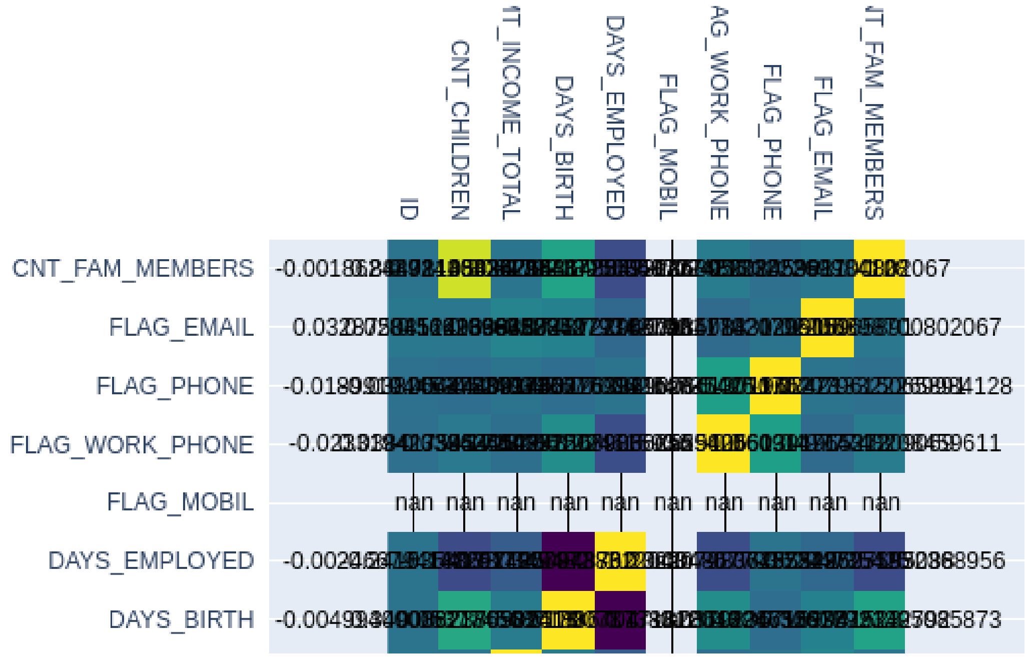
Outputs:

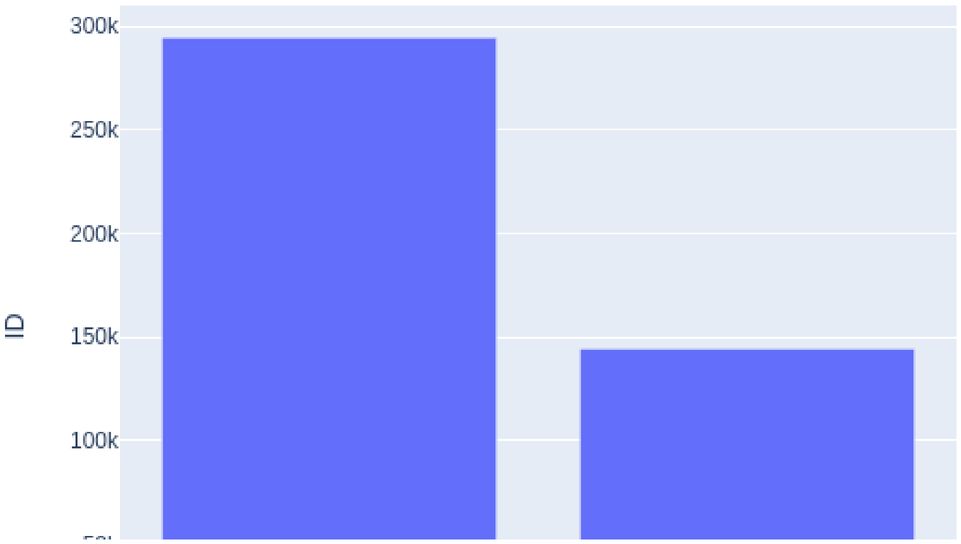


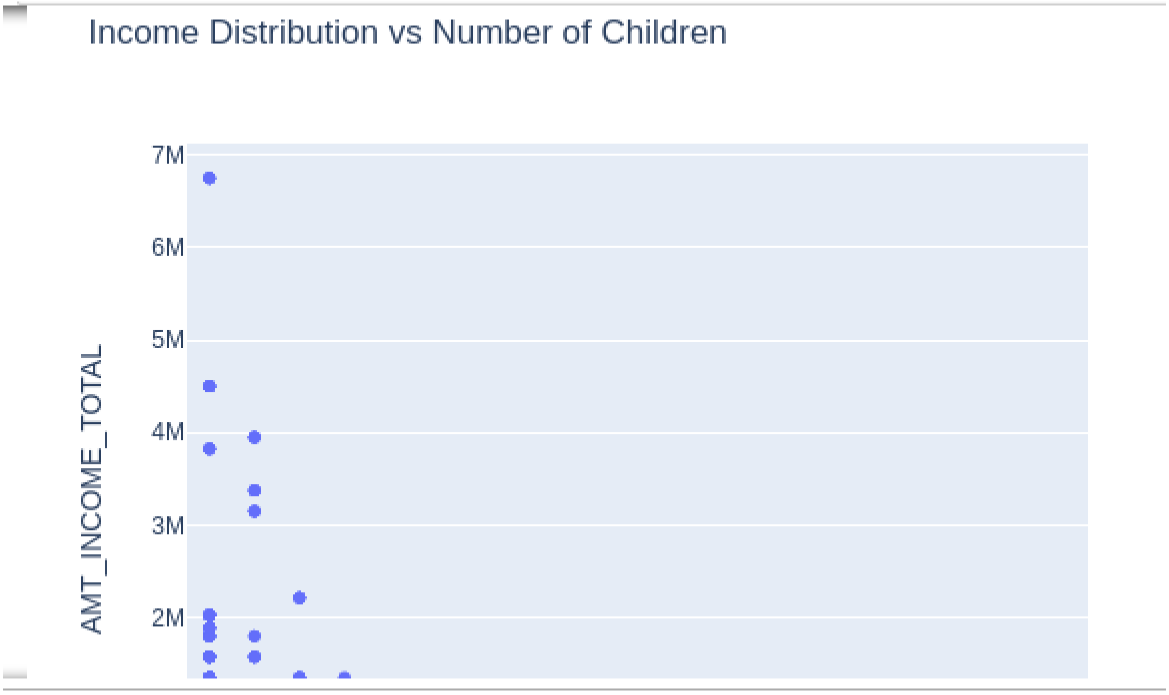
```python
# 3️⃣ Load CSV files (update paths if needed)
app_pdf = pd.read_csv("/home/hadoop/application_record.csv")
cred_pdf = pd.read_csv("/home/hadoop/credit_record.csv")

# ------------------------
# 4️⃣ EDA & Visualizations
# ------------------------

# 4a. Income Distribution
fig1 = px.histogram(app_pdf, x="AMT_INCOME_TOTAL", nbins=50,
                    title="Income Distribution of Applicants")
fig1.show()

# 4b. Credit Status Over Months
status_count = cred_pdf.groupby('MONTHS_BALANCE')['STATUS'].count().reset
fig2 = px.line(status_count, x='MONTHS_BALANCE', y='STATUS',
               title="Credit Status Count Over Months")
fig2.show()

# 4c. Correlation Heatmap of Numeric Features
numeric_cols = app_pdf.select_dtypes(include=np.number).columns
corr = app_pdf[numeric_cols].corr()
fig3 = ff.create_annotated_heatmap(
    z=corr.values,
    x=list(corr.columns),
    y=list(corr.columns),
    colorscale='Viridis'
)
fig3.show()

# 4d. Count of Applicants by Gender
gender_count = app_pdf.groupby('CODE_GENDER')['ID'].count().reset_index()
fig4 = px.bar(gender_count, x='CODE_GENDER', y='ID', title="Applicants by
fig4.show()

# 4e. Children vs Income
fig5 = px.box(app_pdf, x='CNT_CHILDREN', y='AMT_INCOME_TOTAL',
              title="Income Distribution vs Number of Children")
fig5.show()
```


e Writer
Income Distribution of Applicants
count
400k
300k
200k
100k
0


STATUS
25k
20k
15k
10k
5k
0
−60
−50
−40
−30
−20
−10
0
MONTHS_BALANCE


ID
CNT_CHILDREN
IT_INCOME_TOTAL
DAYS_BIRTH
DAYS_EMPLOYED
FLAG_MOBIL
AG_WORK_PHONE
FLAG_PHONE
FLAG_EMAIL
IT_FAM_MEMBERS
CNT_FAM_MEMBERS
FLAG_EMAIL
FLAG_PHONE
FLAG_WORK_PHONE
FLAG_MOBIL
DAYS_EMPLOYED
DAYS_BIRTH
nan nan nan nan nan nan nan nan nan nan


300k
250k
200k
150k
100k
ID



5. Orchestration Layer (Apache Airflow)

Goal:

Automate and monitor the pipeline stages using a Directed Acyclic Graph (DAG).

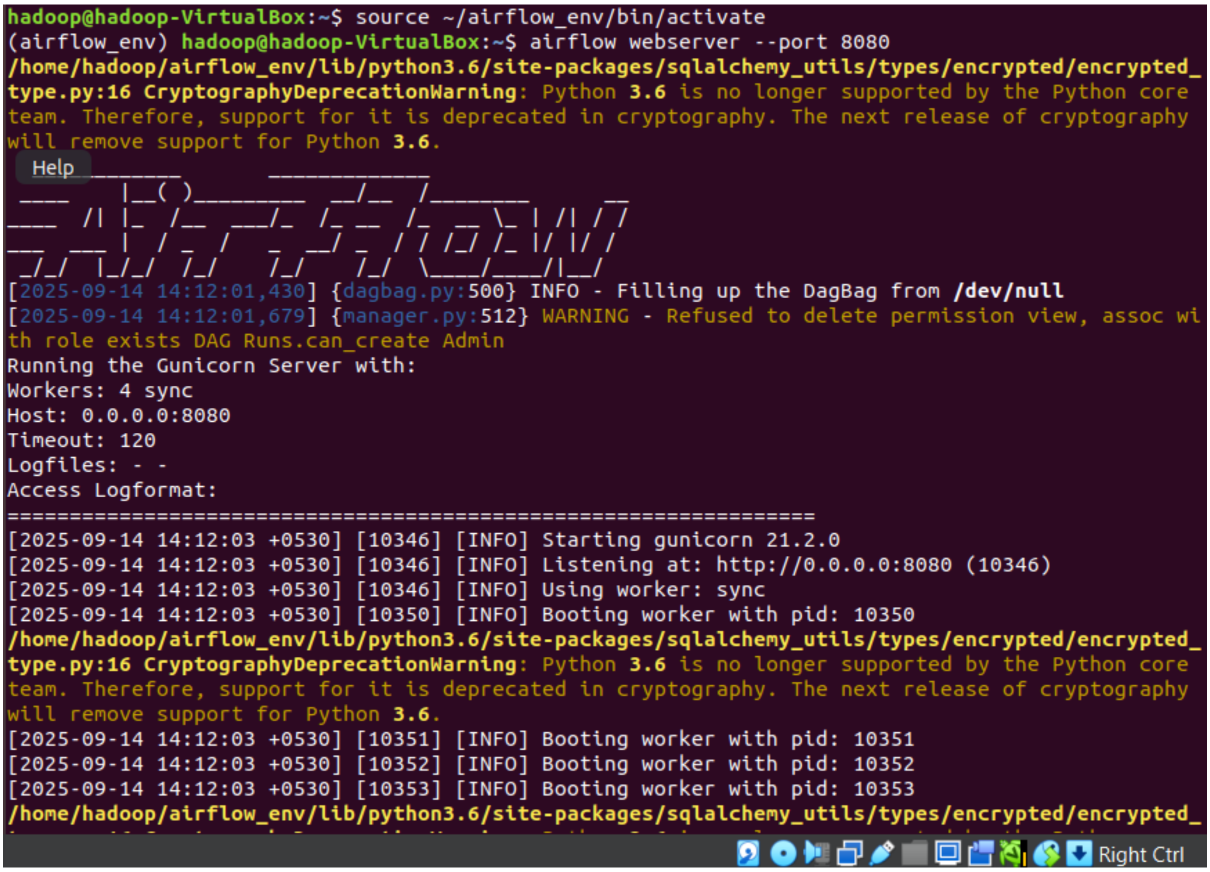
Steps:

- **Set up an Airflow DAG to manage the following tasks:**
  - **Simulate/trigger Kafka ingestion (PythonOperator / BashOperator)**
  - **Submit Pyspark batch job**
- **Schedule the pipeline to run daily/hourly based on business need.**
- **Add dependency handling and branching for fault tolerance.**

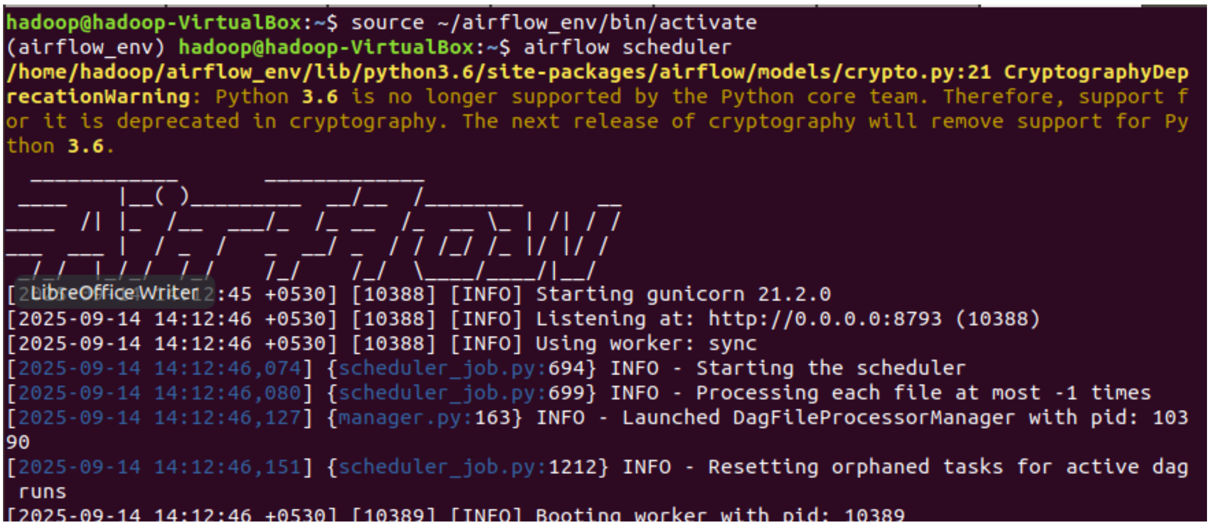
Technologies:

Apache Airflow, PythonOperator, BashOperator

Outputs:



```
hadoop@hadoop-VirtualBox:~$ source ~/airflow_env/bin/activate
(airflow_env) hadoop@hadoop-VirtualBox:~$ airflow webserver --port 8080
/home/hadoop/airflow_env/lib/python3.6/site-packages/sqlalchemy_utils/types/encrypted/encrypted_
type.py:16 CryptographyDeprecationWarning: Python 3.6 is no longer supported by the Python core
team. Therefore, support for it is deprecated in cryptography. The next release of cryptography
will remove support for Python 3.6.
  ____________       _____________
 ____    |__( )_________  __/__  /________      __
____  /| |_  /__  ___/_  /_ __  /_  __ \_ | /| / /
___  ___ |  / _  /   _  __/ _  / / /_/ /_ |/ |/ /
 _/_/  |_/_/  /_/    /_/    /_/  \____/____/|__/
[2025-09-14 14:12:01,430] {dagbag.py:500} INFO - Filling up the DagBag from /dev/null
[2025-09-14 14:12:01,679] {manager.py:512} WARNING - Refused to delete permission view, assoc wi
th role exists DAG Runs.can_create Admin
Running the Gunicorn Server with:
Workers: 4 sync
Host: 0.0.0.0:8080
Timeout: 120
Logfiles: - -
Access Logformat:
=================================================================
[2025-09-14 14:12:03 +0530] [10346] [INFO] Starting gunicorn 21.2.0
[2025-09-14 14:12:03 +0530] [10346] [INFO] Listening at: http://0.0.0.0:8080 (10346)
[2025-09-14 14:12:03 +0530] [10346] [INFO] Using worker: sync
[2025-09-14 14:12:03 +0530] [10350] [INFO] Booting worker with pid: 10350
/home/hadoop/airflow_env/lib/python3.6/site-packages/sqlalchemy_utils/types/encrypted/encrypted_
type.py:16 CryptographyDeprecationWarning: Python 3.6 is no longer supported by the Python core
team. Therefore, support for it is deprecated in cryptography. The next release of cryptography
will remove support for Python 3.6.
[2025-09-14 14:12:03 +0530] [10351] [INFO] Booting worker with pid: 10351
[2025-09-14 14:12:03 +0530] [10352] [INFO] Booting worker with pid: 10352
[2025-09-14 14:12:03 +0530] [10353] [INFO] Booting worker with pid: 10353
/home/hadoop/airflow_env/lib/python3.6/site-packages/sqlalchemy_utils/types/encrypted/encrypted_
```



```
hadoop@hadoop-VirtualBox:~$ source ~/airflow_env/bin/activate
(airflow_env) hadoop@hadoop-VirtualBox:~$ airflow scheduler
/home/hadoop/airflow_env/lib/python3.6/site-packages/airflow/models/crypto.py:21 CryptographyDep
recationWarning: Python 3.6 is no longer supported by the Python core team. Therefore, support f
or it is deprecated in cryptography. The next release of cryptography will remove support for Py
thon 3.6.
  ____________       _____________
 ____    |__( )_________  __/__  /________      __
____  /| |_  /__  ___/_  /_ __  /_  __ \_ | /| / /
___  ___ |  / _  /   _  __/ _  / / /_/ /_ |/ |/ /
 _/_/  |_/_/  /_/    /_/    /_/  \____/____/|__/
[2025-09-14 14:12:45 +0530] [10388] [INFO] Starting gunicorn 21.2.0
[2025-09-14 14:12:46 +0530] [10388] [INFO] Listening at: http://0.0.0.0:8793 (10388)
[2025-09-14 14:12:46 +0530] [10388] [INFO] Using worker: sync
[2025-09-14 14:12:46,074] {scheduler_job.py:694} INFO - Starting the scheduler
[2025-09-14 14:12:46,080] {scheduler_job.py:699} INFO - Processing each file at most -1 times
[2025-09-14 14:12:46,127] {manager.py:163} INFO - Launched DagFileProcessorManager with pid: 103
90
[2025-09-14 14:12:46,151] {scheduler_job.py:1212} INFO - Resetting orphaned tasks for active dag
 runs
[2025-09-14 14:12:46 +0530] [10389] [INFO] Booting worker with pid: 10389
```