INSTITUTE OF BUSINESS ADMINISTRATION
KARACHI

# PROJECT REPORT

(Big Data Analytics)

[Big Data Analysis of English Tweets

& Comparison of Big Data Tools]

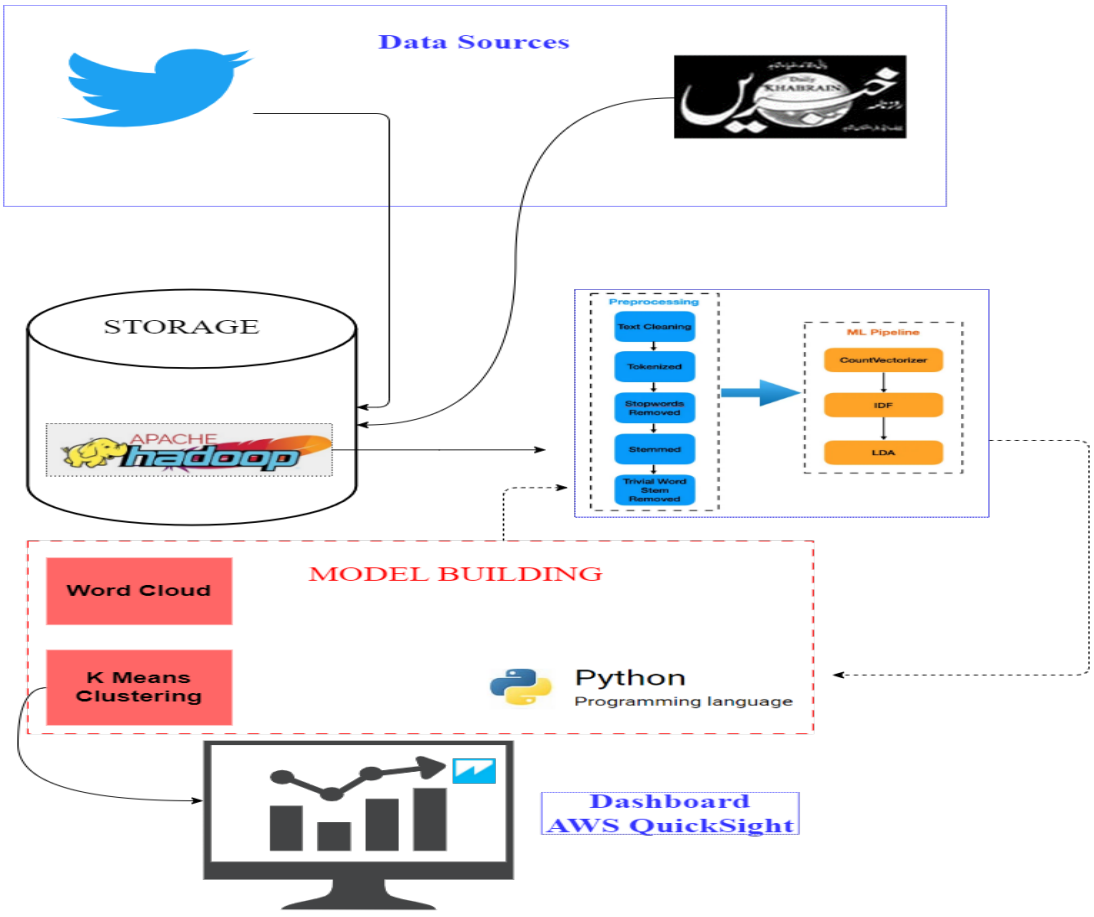

| S.NO | Name | ERP |
|---|---|---|
| 1 | Bilawal Ali | 17278 |
| 2 | Muhammed Imran Magsi | 17670 |
| 3 | Muhammad Omar Qureshi | 07133 |

# TABLE OF CONTENTS

# 1 PROJECT # 01:

# 2 PRIMARY PROJECT: URDU TWEET ANALYSIS

## 2.1 PIPELINE




## 2.2 PROJECT ABSTRACT:

We plan to build a dockerized application that first fetches live tweets from Twitter and RSS feeds of different news articles/sources. We are saving all these tweets in .csv file format. Once we have the data, we start Pre-processing files. During Pre-processing we are removing punctuations, English texts, numbers and special characters then we start removing Stop words. Once we are done with Pre-processing, we do lemmatization and tokenization of Urdu words. Then we apply Term-Frequency and Inverse document frequency to evaluate how relevant a word is to a document in a collection of documents. Now comes the modelling, we are applying K-means Algorithm to cluster the tweets on the basis of TF-IDF vectors, here

Knee Locator finds the optimal value of K. Then comes the Principal Component Analysis (PCA) that we use for denoising and data compression, and finally we get the clusters. We are creating Tag Clouds for all clusters formed during K-means also showing Tweets of all clusters on a separate page. There are three ways a user can cluster or fetch data. First, they can see the default clustering the application does by itself (Live data fetching) secondly users can upload the file, thirdly they can also search a keyword and all tweets containing the keyword will be fetched live and all the modeling will start from the beginning.

## 2.3 PREPROCESSING NEWS RAW TEXT

We have used Urduhack library to preprocess our Urdu text data, and have removed any emojis, numbers, English alphabets, links and stop-words before using the data for any visual analysis. Here are some functions which we have used from urduhack.preprocessing module while processing data.

- urduhack_normalize
- normalize_whitespace
- remove_punctuation
- replace_urls
- replace_emails
- replace_numbers
- remove_english_alphabets
- remove_diacritics

Besides, we have used python regular expressions and lemmatization techniques from Urduhack library to convert the words to their basic form.

## 2.4 TF-IDF

We have utilized Term Frequency-Inverse Document Frequency as a text vectorizer to transform the text into a usable vector. This vector is then used to generate clusters.

## 2.5 K-MEANS CLUSTERING

We used the K-means clustering algorithm for generating clusters. K-Means aims to partition a set of observations into several clusters.

## 2.6 KNEE POINT

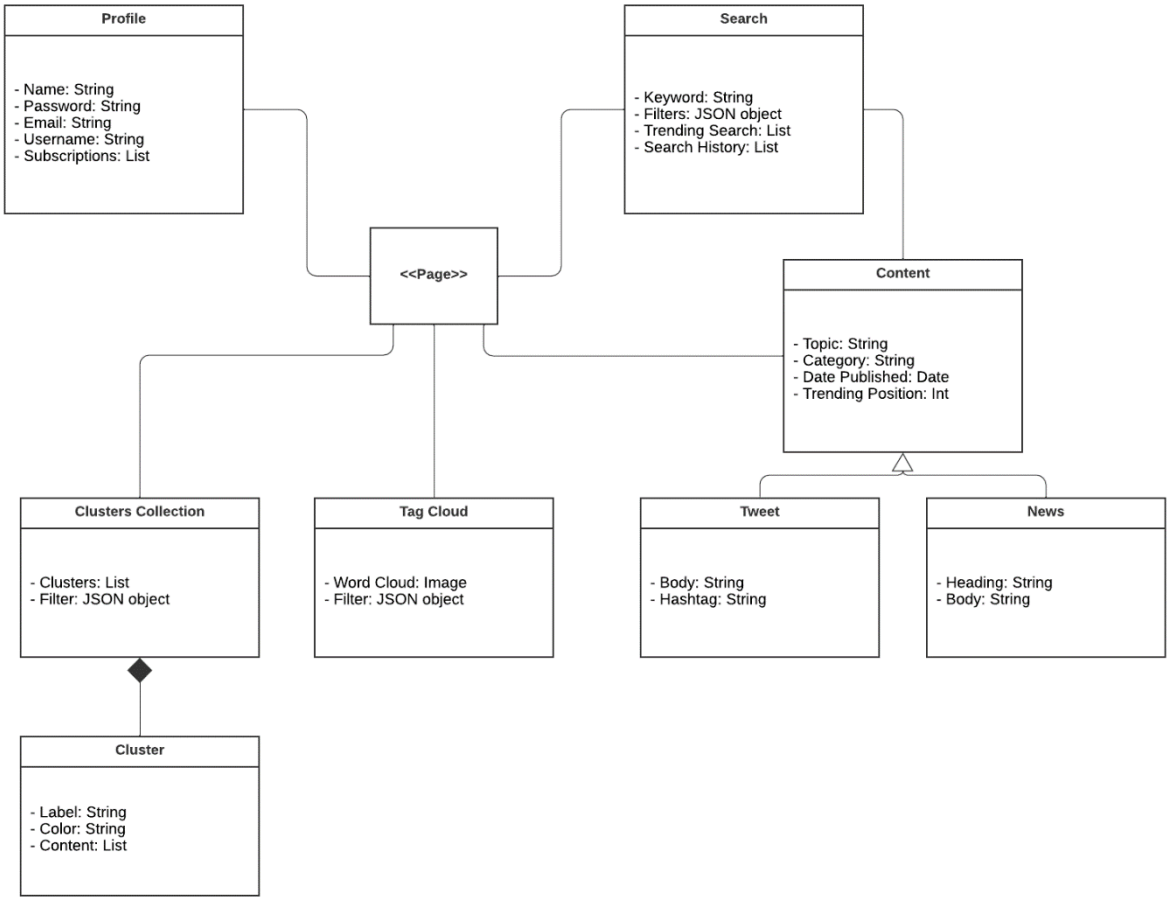
To find the optimal number of clusters, we are using Knee Point Locator which determines the suitable number of clusters using SSE intertia values.

## 2.7 WORD CLOUD

Given a textual data, the word-cloud function returns a dictionary of objects with unique words as keys, and their font size (directly proportional to the frequency) in word-cloud as value. This dictionary object is used in node.js to create and visualize the cluster.

Note: **Other details of this project is in .ipynb notebook**

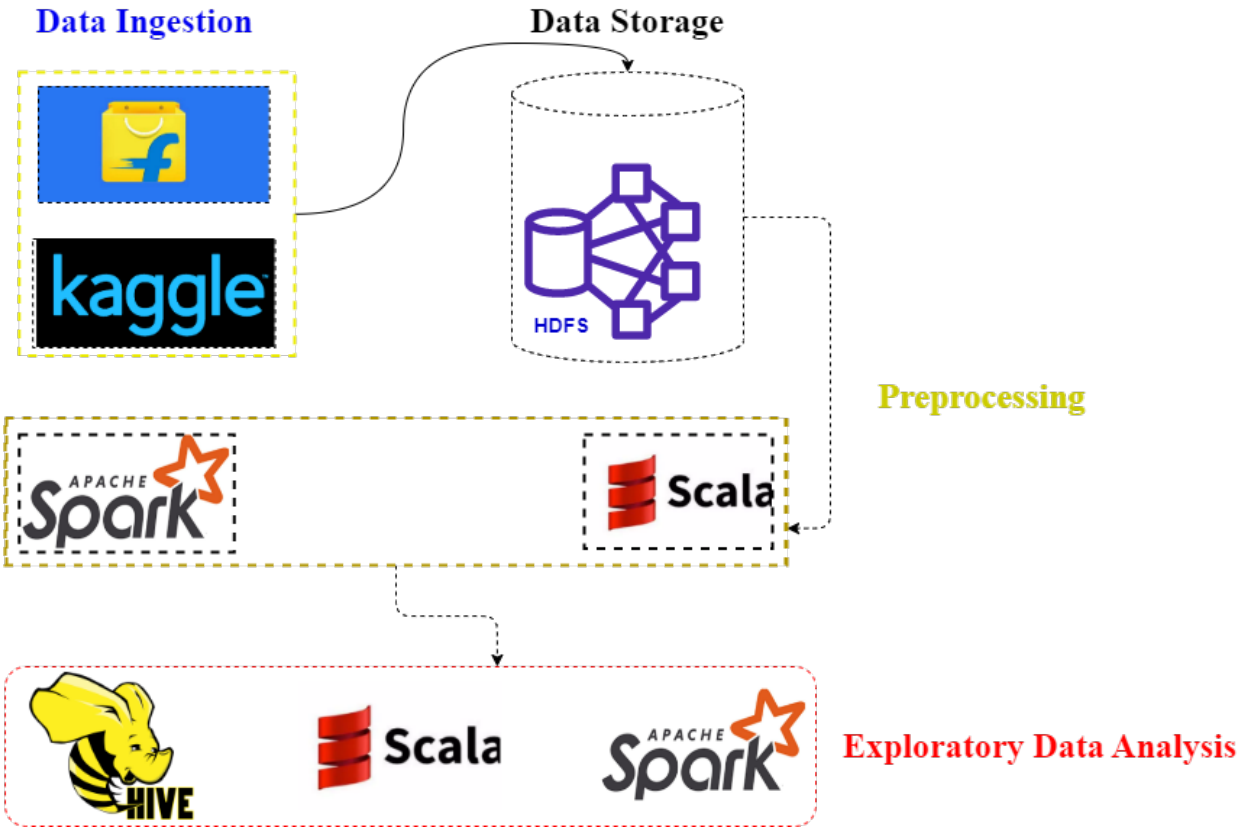
# INTERFACE:


Profile
- Name: String
- Password: String
- Email: String
- Username: String
- Subscriptions: List
Search
- Keyword: String
- Filters: JSON object
- Trending Search: List
- Search History: List
<<Page>>
Content
- Topic: String
- Category: String
- Date Published: Date
- Trending Position: Int
Clusters Collection
- Clusters: List
- Filter: JSON object
Tag Cloud
- Word Cloud: Image
- Filter: JSON object
Tweet
- Body: String
- Hashtag: String
News
- Heading: String
- Body: String
Cluster
- Label: String
- Color: String
- Content: List

# 3 PROJECT #02:

# 4 SECONDARY PROJECT: APACHE HIVE VS APACHE SPARK VS APACHE DRILL

## 4.1 DATA PIPELINE




## 4.2 DATASET INTRODUCTION

Size: 3GB

Mobile prices extracted from flipkart a day before big billion days.

Categories

Mobiles, Laptops, Tvs, Tablets

Columns

- name - Name of the item
- offer_price - Current price with offer
- original_price - Orginal price of the item without any offer
- off_now - % of off on item

- total_ratings - Number of ratings given
- total_reviews - Number of reviews written
- rating - Final rating
- description - Description of the product
- created_at - Dataset created time
- u_id - Unique id
- item_link - link of the particular item

## 4.3 SPARK INTRODUCTION

Spark is an open-source distributed computing system designed to process large-scale data sets and perform data analysis at high speeds. It provides a unified and flexible framework for data processing, enabling developers to efficiently process data across various data sources and execute complex data transformations and computations.

One of the key features of Spark is its ability to handle data processing tasks in-memory, which significantly enhances the overall performance compared to traditional disk-based processing systems. Spark's in-memory computing allows for iterative and interactive data analysis, making it well-suited for real-time analytics, machine learning, and graph processing.

Spark offers a rich set of APIs and libraries that support various programming languages, including Scala, Java, Python, and R. This versatility enables developers to work with Spark using their preferred programming language and leverage the vast ecosystem of tools and libraries available in the Spark ecosystem.

With its distributed computing model, Spark enables parallel processing of data across clusters of machines, making it highly scalable and capable of handling large-scale data processing tasks. It provides fault tolerance and resilience by automatically recovering from failures, ensuring the reliability of data processing workflows.

Furthermore, Spark provides a wide range of high-level abstractions, such as DataFrames and Datasets, which simplify the development of complex data processing pipelines. These abstractions offer a more intuitive and declarative programming model, allowing developers to focus on the logic of their data analysis tasks rather than dealing with low-level details.

Overall, Spark has revolutionized big data processing and analytics, empowering organizations to efficiently extract valuable insights from massive datasets. Its speed, scalability, and rich feature set make it a popular choice for data scientists, engineers, and analysts seeking to harness the power of distributed computing for their data-driven applications.

## 4.4 SPARK ON DOCKER

1. Install Docker for Desktop from (*https://docs.docker.com/desktop/windows/install/)*
2. Clone the Spark using the command line: *git clone https://github.com/big-data-europe/docker-spark.git*
3. Navigate to the cloned folder docker-spark and Execute: docker-compose up

4. Copy the directory (here tvs) from local machine to the Spark using : *docker cp D:\big.data\docker-hadoop\spark\docker-spark\tvs 76a4c7ff984b:home* (where **76a4c7ff984b** is the Spark container's address running on the Docker)
5. Bash into the spark-master: *docker exec -it spark-master /bin/bash*
6. Use the Spark shell *(Scala)* by executing : */spark/bin/spark-shell –master=local spark://spark-master:7077*

```
D:\big.data\docker-hadoop\spark\docker-spark>docker cp D:\big.data\docker-hadoop\spark\docker-spark\tvs 76a4c7ff984b:hom
e

D:\big.data\docker-hadoop\spark\docker-spark>docker cp D:\big.data\docker-hadoop\spark\docker-spark\mobiles 76a4c7ff984b
:home

D:\big.data\docker-hadoop\spark\docker-spark>docker cp D:\big.data\docker-hadoop\spark\docker-spark\other_electronics 76
a4c7ff984b:home

D:\big.data\docker-hadoop\spark\docker-spark> docker exec -it spark-master /bin/bash
bash-5.0# cd home
bash-5.0# ls
laptops  mobiles  other_electronics  tablet  tvs
bash-5.0#
```

Dataset directory : tablets

```
bash-5.0#  /spark/bin/spark-shell –master=local spark://spark-master:7077
```

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.3.0
      /_/

Using Scala version 2.12.15 (OpenJDK 64-Bit Server VM, Java 1.8.0_275)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val laptopsData = spark.read.format("csv").option("header", "true").load("/home/laptops")
laptopsData: org.apache.spark.sql.DataFrame = [u_id: string, name: string ... 9 more fields]

scala>  val tvsData = spark.read.format("csv").option("header", "true").load("/home/tvs")
tvsData: org.apache.spark.sql.DataFrame = [u_id: string, name: string ... 9 more fields]
```

```
scala>  val tabletData = spark.read.format("csv").option("header", "true").load("/home/tablet")
tabletData: org.apache.spark.sql.DataFrame = [u_id: string, name: string ... 9 more fields]

scala>  val laptopsData = spark.read.format("csv").option("header", "true").load("/home/laptops")
laptopsData: org.apache.spark.sql.DataFrame = [u_id: string, name: string ... 9 more fields]
```

## 4.5 SPARK COMMANDS & ANALYTICS

```
scala> val data = spark.read.format("csv")
data: org.apache.spark.sql.DataFrameReader = org.apache.spark.sql.DataFrameReader@5b33446d

scala> val data = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load("/home/tablet")
data: org.apache.spark.sql.DataFrame = [u_id: string, name: string ... 9 more fields]

scala> data.printSchema()
root
 |-- u_id: string (nullable = true)
 |-- name: string (nullable = true)
 |-- offer_price: integer (nullable = true)
 |-- original_price: integer (nullable = true)
 |-- off_now: string (nullable = true)
 |-- total_ratings: integer (nullable = true)
 |-- total_reviews: integer (nullable = true)
 |-- rating: double (nullable = true)
 |-- description: string (nullable = true)
 |-- item_link: string (nullable = true)
 |-- created_at: string (nullable = true)
```

The detailed description of the dataset

```
scala> data.show()
+--------+--------------------+-----------+--------------+-------+-------------+-------------+------+--------------------+--------------------+----------
------+
    u_id|                name|offer_price|original_price|off_now|total_ratings|total_reviews|rating|         description|           item_link|         cre
ted_at|
+--------+--------------------+-----------+--------------+-------+-------------+-------------+------+--------------------+--------------------+----------
------+
SEYTLMU5|Oppo Pad Air 4 GB...|      16999|         29999|43% off|         1908|          238|   4.4|['4 GB RAM | 64 G...|https://www.flipk...|2022-11-03 22
33:...|
OWW4N9FG|Oppo Pad Air 4 GB...|      19999|         34999|42% off|         1908|          238|   4.4|['4 GB RAM | 128 ...|https://www.flipk...|2022-11-03 22
33:...|
2BE7C106|Redmi Pad 6 GB RA...|      19999|         33999|41% off|          503|           80|   4.4|['6 GB RAM | 128 ...|https://www.flipk...|2022-11-03 22
33:...|
A97WTC95|Redmi Pad 4 GB RA...|      17999|         28999|37% off|          503|           80|   4.4|['4 GB RAM | 128 ...|https://www.flipk...|2022-11-03 22
33:...|
KJYUQU9B|Redmi Pad 4 GB RA...|      17999|         28999|37% off|          503|           80|   4.4|['4 GB RAM | 128 ...|https://www.flipk...|2022-11-03 22
33:...|
9F39HUEC|realme Pad X 6 GB...|      27999|         44999|37% off|         3263|          348|   4.4|['Compatible with...|https://www.flipk...|2022-11-03 22
33:...|
ETTF771O|realme Pad X 6 GB...|      27999|         44999|37% off|         3263|          348|   4.4|['Compatible with...|https://www.flipk...|2022-11-03 22
33:...|
VKX7NOZS|SAMSUNG Galaxy Ta...|      24999|         30999|19% off|        10126|          838|   4.5|['4 GB RAM | 64 G...|https://www.flipk...|2022-11-03 22
33:...|
U5RX4I8A|realme Pad 4 GB R...|      16849|         29999|43% off|        44194|         4830|   4.4|['4 GB RAM | 64 G...|https://www.flipk...|2022-11-03 22
33:...|
U9FKMK61|realme Pad 4 GB R...|      16899|         29999|43% off|        44194|         4830|   4.4|['4 GB RAM | 64 G...|https://www.flipk...|2022-11-03 22
33:...|
9N7OM0RV|SAMSUNG Galaxy Ta...|      13999|         21599|35% off|         3513|          339|   4.3|['3 GB RAM | 32 G...|https://www.flipk...|2022-11-03 22
33:...|
V72CGWE3|realme Pad X 4 GB...|      19999|         29999|33% off|         3263|          348|   4.4|['Compatible with...|https://www.flipk...|2022-11-03 22
33:...|
J8IKHIYJ|realme Pad Mini 4...|      14999|         23999|37% off|         4970|          484|   4.3|['18W Fast Chargi...|https://www.flipk...|2022-11-03 22
```

To show the number of rows in the dataset file

```
scala> val numRows = data.count()
numRows: Long = 7464

scala> println(s"Number of rows: $numRows")
Number of rows: 7464
```

Average ratings received for the tablets

```
scala> val avgRating = data.select(avg("rating")).first().getDouble(0)
avgRating: Double = 3.7867229367631134

scala> println(s"Average rating: $avgRating")
Average rating: 3.7867229367631134
```

High rated items: rating > 4.5

```
scala> val highRatedItems = data.filter($"rating" > 4.5)
highRatedItems: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [u_id: string, name: string ... 9 more fields]

scala> highRatedItems.show()
+--------+--------------------+-----------+-------------+-------+-------------+-------------+------+--------------------+--------------------+--------------------
--------+
|    u_id|                name|offer_price|original_price|off_now|total_ratings|total_reviews|rating|         description|           item_link|          cre
ated_at|
+--------+--------------------+-----------+-------------+-------+-------------+-------------+------+--------------------+--------------------+--------------------
-------+
|UWXIYHQQ|APPLE iPad (9th G...|      30900|        30900|     0%|        10196|          939|   4.6|['64 GB ROM', '25...|https://www.flipk...|2022-11-03 22
:33:...|
|RVCVFUPK|SAMSUNG Galaxy Ta...|      51999|        76999|32% off|         3220|          344|   4.6|['6 GB RAM | 128 ...|https://www.flipk...|2022-11-03 22
:33:...|
|TA2F3ZI1|SAMSUNG Galaxy Ta...|      66999|        85999|22% off|          160|           10|   4.7|['8 GB RAM | 128 ...|https://www.flipk...|2022-11-03 22
:33:...|
|V0MO08MQ|APPLE iPad (9th G...|      44900|        44900|     0%|        10196|          939|   4.6|['256 GB ROM', '2...|https://www.flipk...|2022-11-03 22
:33:...|
|KSN55S1H|APPLE iPad Pro 20...|      70900|        71900| 1% off|         1961|          167|   4.7|['Apple M1 chip f...|https://www.flipk...|2022-11-03 22
:33:...|
|8VCZR1SQ|SAMSUNG Galaxy Ta...|      66999|        85999|22% off|          160|           10|   4.7|['8 GB RAM | 128 ...|https://www.flipk...|2022-11-03 22
:33:...|
|PSEWFZ2X|APPLE iPad Pro 20...|      79900|        80900| 1% off|         1961|          167|   4.7|['Apple M1 chip f...|https://www.flipk...|2022-11-03 22
:33:...|
|9KH0YDSA|APPLE iPad mini (...|      60900|        60900|     0%|          409|           49|   4.6|['256 GB ROM', '2...|https://www.flipk...|2022-11-03 22
:33:...|
|4TCZI9TN|APPLE iPad (9th G...|      30900|        30900|     0%|        10196|          939|   4.6|['64 GB ROM', '25...|https://www.flipk...|2022-11-03 22
:33:...|
|128P0388|APPLE iPad mini (...|      60900|        60900|     0%|          409|           49|   4.6|['64 GB ROM', '21...|https://www.flipk...|2022-11-03 22
```

Compute the average rating and total reviews for each tablet:

```
scala> tabletAnalysis.show
+--------------------+------------------+-------------+
|                name|        avg_rating|total_reviews|
+--------------------+------------------+-------------+
|acer Spin 3 Core ...|               5.0|          2.0|
|acer Spin 5 Core ...|               5.0|          2.0|
|ASUS ROG Flow X13...|               5.0|          0.0|
|HP Spectre x360 C...|               5.0|          7.0|
|Lenovo Yoga 6 Ryz...|4.971428571428571|          5.0|
|APPLE iPad Pro (2...|               4.8|        352.0|
|APPLE iPad Pro (2...|               4.8|        352.0|
|APPLE iPad Pro (2...|               4.8|        308.0|
|APPLE iPad Pro (2...|               4.8|        308.0|
|APPLE iPad Pro (2...|               4.8|        396.0|
|APPLE iPad Pro (2...|               4.8|        308.0|
|APPLE iPad Pro (2...|               4.8|        352.0|
|Lenovo Yoga Slim ...|               4.8|         48.0|
|APPLE iPad Pro (2...|               4.8|        352.0|
|APPLE iPad Pro (2...|               4.8|        308.0|
|HP Envy 13 x360 I...|               4.8|         16.0|
|Lenovo Celeron Du...|               4.8|          0.0|
|APPLE iPad Pro (2...|               4.8|        308.0|
|APPLE iPad Pro (2...|               4.8|        352.0|
|APPLE iPad Pro (2...|               4.8|        396.0|
+--------------------+------------------+-------------+
only showing top 20 rows
```

Analyze the distribution of tablet ratings:

```
scala> ratingDistribution.show
+------+-----+
|rating|count|
+------+-----+
|     0|  735|
|   1.5|    4|
|   2.2|    4|
|   2.4|    7|
|   2.5|   21|
|   2.6|   22|
|   2.7|  104|
|   2.8|   88|
|   2.9|   31|
|     3|  131|
|   3.1|  178|
|   3.2|   92|
|   3.3|   93|
|   3.4|   78|
|   3.5|  115|
|   3.6|  209|
|   3.7|  214|
|   3.8|  240|
|   3.9|  131|
|     4|  225|
+------+-----+
only showing top 20 rows
```

Lowest and highest rated product

```
scala> val highestRatedTablet = tabletsByRating.first()
highestRatedTablet: org.apache.spark.sql.Row = [HP Spectre x360 Core i5 10th Gen - (8 GB/512 GB SSD/Windows 10 Pro) 13-aw0211TU 2 in 1 Laptop,5.0]

scala> val lowestRatedTablet = tabletsByRating.orderBy("avg_rating").first()
lowestRatedTablet: org.apache.spark.sql.Row = [DOMO Slate SLP9 OS11 4 GB RAM 64 GB ROM 10.1 inch with Wi-Fi+4G Tablet (Grey),0.0]
```

Analyze the distribution of ratings across different price ranges:

```
scala> val priceRangeAnalysis = tabletData.withColumn("price_range", when(col("offer_price") <= 10000, "Low").when(col("
offer_price") <= 20000, "Medium").otherwise("High")).groupBy("price_range").agg(avg("rating").alias("avg_rating"), count
("u_id").alias("count"))
priceRangeAnalysis: org.apache.spark.sql.DataFrame = [price_range: string, avg_rating: double ... 1 more field]

scala> priceRangeAnalysis.show
+-----------+------------------+-----+
|price_range|        avg_rating|count|
+-----------+------------------+-----+
|       High| 3.9509261325597706| 4481|
|        Low|3.2346350832266335| 1562|
|     Medium|3.8757916959887404| 1421|
+-----------+------------------+-----+
```

Determine the most popular brands based on the total number of ratings and reviews:

```
scala> brandAnalysis.show
+---------+-------------+-------------+
|    brand|total_ratings|total_reviews|
+---------+-------------+-------------+
|    APPLE|  1.0337189E7|    1032695.0|
|   realme|    2486790.0|     268583.0|
|  SAMSUNG|    2132809.0|     200068.0|
|   Lenovo|    1941314.0|     245920.0|
|    Honor|     541250.0|      67698.0|
|  Alcatel|     528099.0|      94703.0|
|        I|     489911.0|      62894.0|
|    Swipe|     165660.0|      35173.0|
|   Huawei|     151229.0|      20857.0|
|       HP|     138174.0|      16051.0|
|    iball|     111902.0|      15651.0|
| MOTOROLA|      64304.0|       8161.0|
| Micromax|      58243.0|       8093.0|
|     acer|      51631.0|       5774.0|
|    Apple|      50967.0|       7693.0|
|    Nokia|      44925.0|       5829.0|
|Panasonic|      33410.0|       4466.0|
|      TCL|      29940.0|       5547.0|
|     Oppo|      26596.0|       3322.0|
|    Redmi|      22376.0|       3596.0|
+---------+-------------+-------------+
```

Analyze the distribution of tablets across different brands:

```
scala> brandDistribution.show
+----------+-----+
|     brand|count|
+----------+-----+
|     APPLE| 2460|
|    Lenovo|  738|
|   SAMSUNG|  735|
|        HP|  483|
|         I|  407|
|      ASUS|  296|
|   Alcatel|  243|
|     Swipe|  232|
|      DELL|  190|
|     iball|  170|
|    realme|  155|
|    Maplin|  110|
|      DOMO|  108|
|   Wishtel|  106|
|     Honor|   85|
|     Redmi|   76|
|     Apple|   76|
|      LAVA|   67|
|Smartbeats|   55|
|      IQOO|   54|
+----------+-----+
```

Determine the top 5 tablets with the highest discount percentage:

```
scala> topDiscountTablets.show
+--------+--------------------+-----------+--------------+-------+-------------+-------------+------+--------------------+--------------------+-------------------
-------+
|    u_id|                name|offer_price|original_price|off_now|total_ratings|total_reviews|rating|         description|           item_link|          cre
ated_at|
+--------+--------------------+-----------+--------------+-------+-------------+-------------+------+--------------------+--------------------+-------------------
-------+
|3SNS9Z4N|HP Chromebook Med...|      22499|         24845| 9% off|         5812|          668|   3.8|['MediaTek MediaT...|https://www.flipk...|2022-11-10 22
:58:...|
|ZCSF0I4B|APPLE iPad Pro 51...|      72990|         80500| 9% off|         3215|          392|   4.7|"['10.5 Retina di...|        '512 GB ROM'| '26.67 cm (1
0.5 ...|
|KPQJ2QSJ|APPLE iPad Pro 51...|      72990|         80500| 9% off|         3215|          392|   4.7|"['10.5 Retina di...|        '512 GB ROM'| '26.67 cm (1
0.5 ...|
|HWURVWNG|MOTOROLA e32s (Mi...|       7990|          8799| 9% off|           95|           18|   3.7|['3 GB RAM | 32 G...|https://www.flipk...|2022-11-10 22
:57:...|
|J9HUMNUX|APPLE iPad Pro 20...|      89900|         98900| 9% off|          819|           84|   4.7|['6 GB RAM | 512 ...|https://www.flipk...|2022-11-10 22
:59:...|
+--------+--------------------+-----------+--------------+-------+-------------+-------------+------+--------------------+--------------------+-------------------
```

Maximum discount by each brand and its rating

```
scala> maxDiscountBrand.show
+--------+------------+------+
|   brand|max_discount|rating|
+--------+------------+------+
|   APPLE|      9% off|   4.6|
|   Swipe|      9% off|   3.7|
|  Lenovo|      9% off|   4.3|
|ankikrit|      9% off|   3.8|
|   iball|      9% off|     4|
|      HP|      9% off|   3.8|
|MOTOROLA|      9% off|   4.1|
| SAMSUNG|      9% off|   4.5|
| Wishtel|      8% off|   3.8|
|    DELL|      8% off|   4.2|
|    ASUS|      8% off|     0|
|Datawind|      8% off|   3.1|
| Coolpad|      7% off|   3.4|
|   Elevn|     62% off|   3.7|
| Contixo|     62% off|   2.7|
|    DOMO|     61% off|   3.7|
|     TCL|     60% off|   3.8|
| FUSION5|     60% off|   3.3|
|   VIZIO|     51% off|     0|
|  nuveck|     51% off|   2.4|
+--------+------------+------+
```

AvgOriginalPriceByOfferPrice

```
scala> val avgOriginalPriceByOfferPrice = data.groupBy("offer_price").agg(avg("original_price").alias("avg_original_price"))
avgOriginalPriceByOfferPrice: org.apache.spark.sql.DataFrame = [offer_price: int, avg_original_price: double]

scala> avgOriginalPriceByOfferPrice.show()
+-----------+------------------+
|offer_price|avg_original_price|
+-----------+------------------+
|      81900|           84098.0|
|      59990|           66964.0|
|      40900|           40900.0|
|      47900|           57900.0|
|      22990| 28132.85714285714|
|       2999|3306.6923076923076|
|     162900|          162900.0|
|      91990|          112990.0|
|      85900|         90390.625|
|      80999|           99999.0|
|      89900|           92900.0|
|      11146|           14999.0|
|      16999|           23974.4|
|      63191|           71900.0|
|      82500|           98200.0|
|       4799|            7999.0|
|     217999|          229599.0|
|      38600|           38600.0|
|     130000|          140990.0|
|     134990|          152990.0|
+-----------+------------------+
```

Summary of the datasets

```
scala> summaryData.show()
+--------+-----+------------------+
|Category|Count|     AverageRating|
+--------+-----+------------------+
| Laptops|16848| 3.214693732193721|
| Mobiles|17760| 4.090196917808223|
|     TVs|16752|3.0908491107286307|
| Tablets| 7464|3.7867229367631134|
+--------+-----+------------------+
```

# 5 Apache Hive: Bilawal

## 5.1 What is Hive

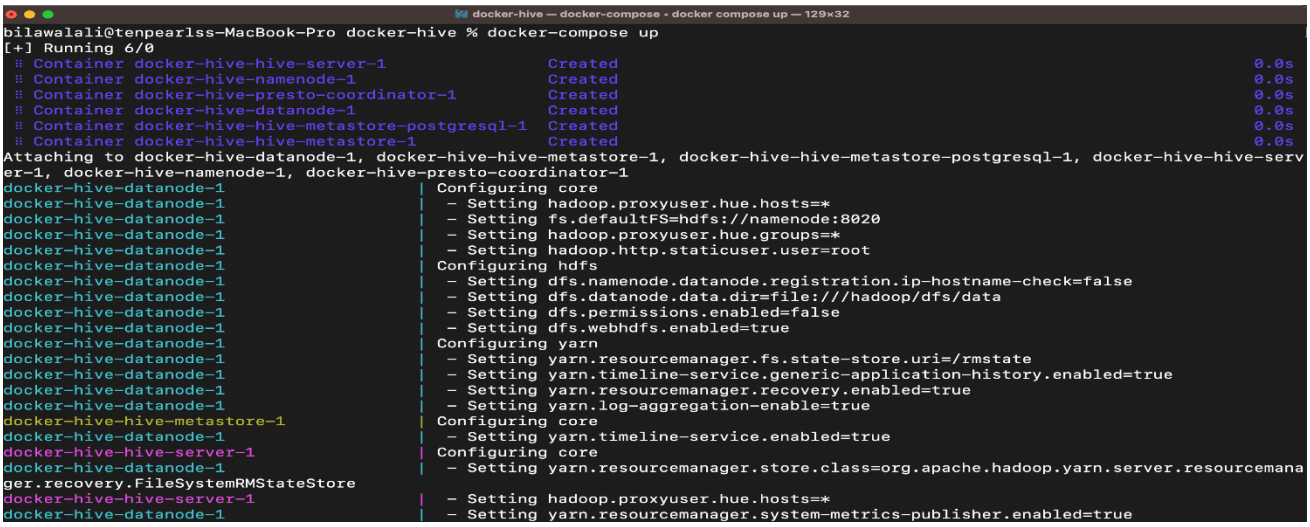
Apache Hive is a data analysis tool that runs on top of Hadoop and serves as a data warehousing solution. It is designed for users familiar with SQL, providing a SQL-like language called HiveQL for managing and querying structured data. By using Hive, we can simplify the complexities associated with Hadoop, making it easier to work with and analyze data.

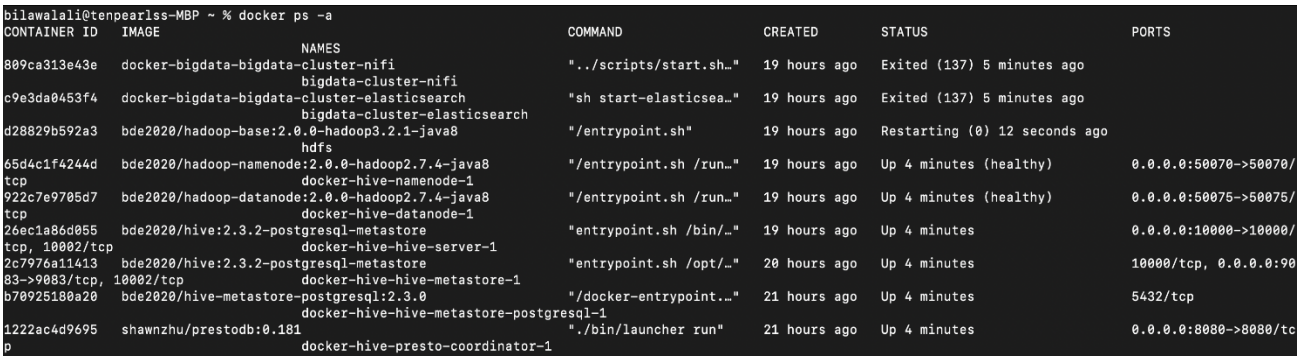
Our Working:

In our docker-compose.yml file, we have included important components like Hadoop and Hive as Docker images. These components are essential for our project as they allow us to perform queries on the Apache Hive platform. We will use Hadoop to store our large dataset, and then we will load this data into Apache Hive from a hdfs path.

## 5.2 STEPS.

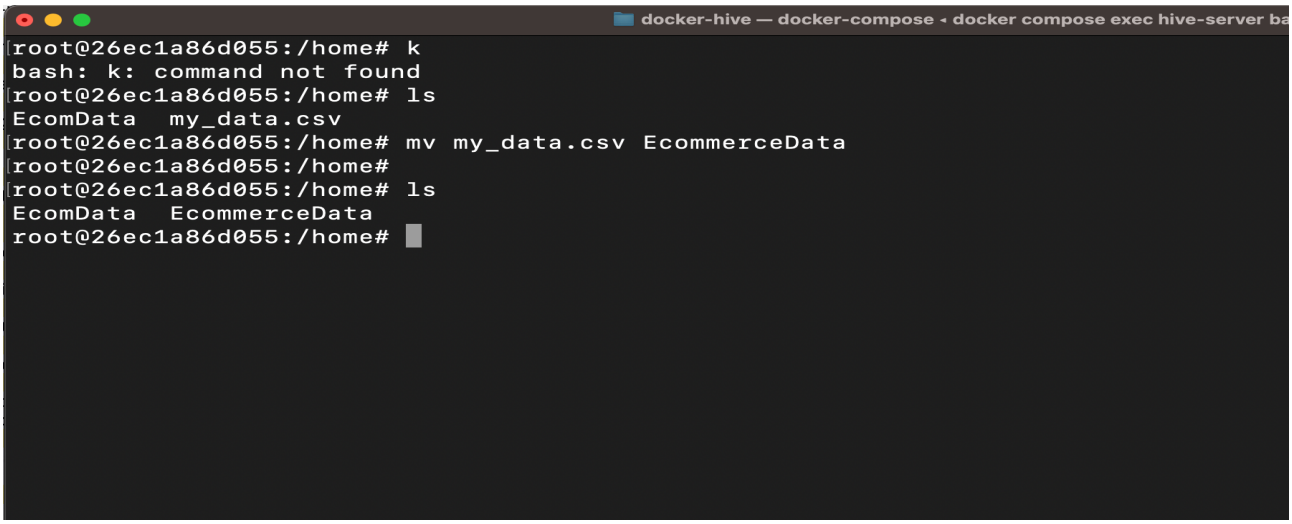
1. Run docker file and check if containers are running



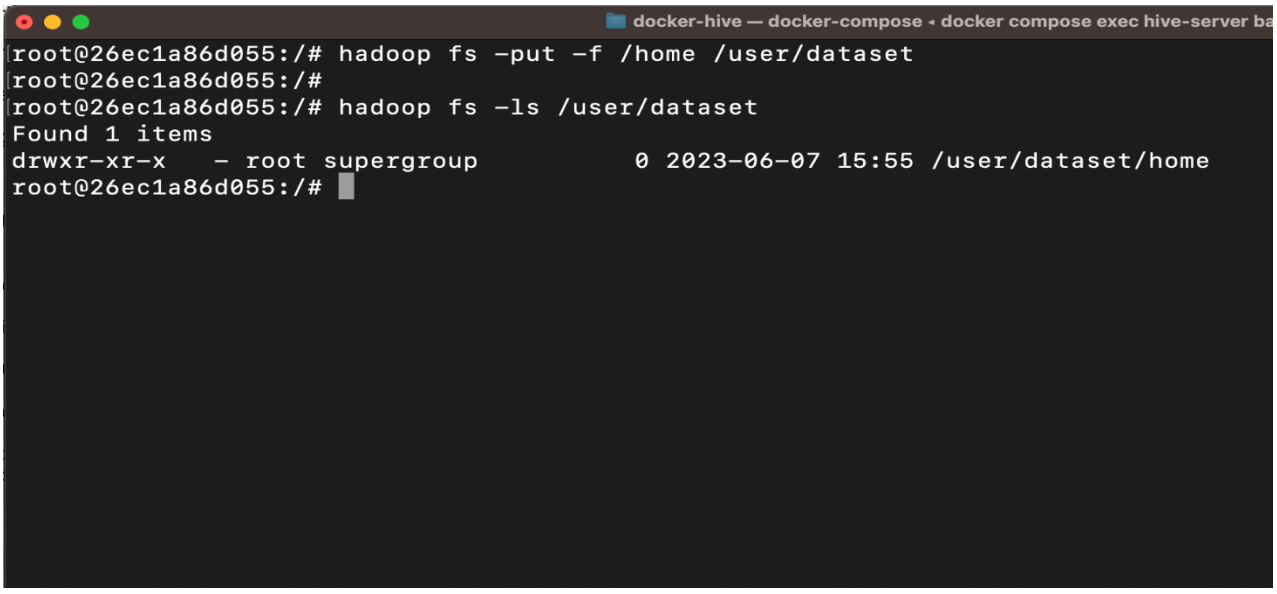




2. Copy data to container and bash into the container to see the data

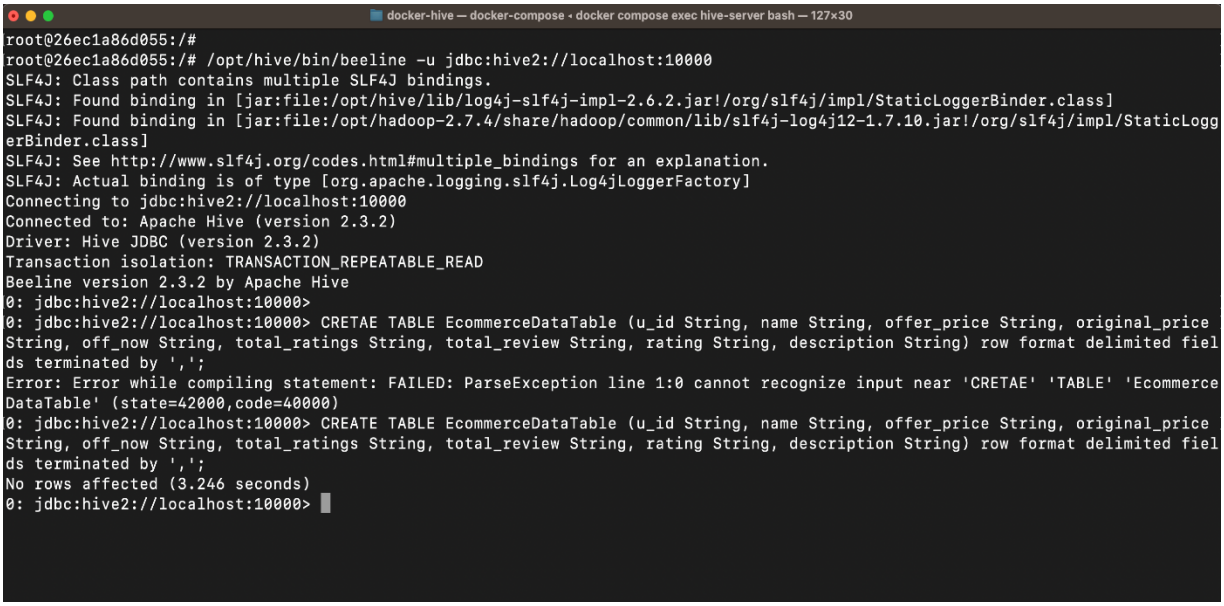
3. Renaming the data so that we can read it properly

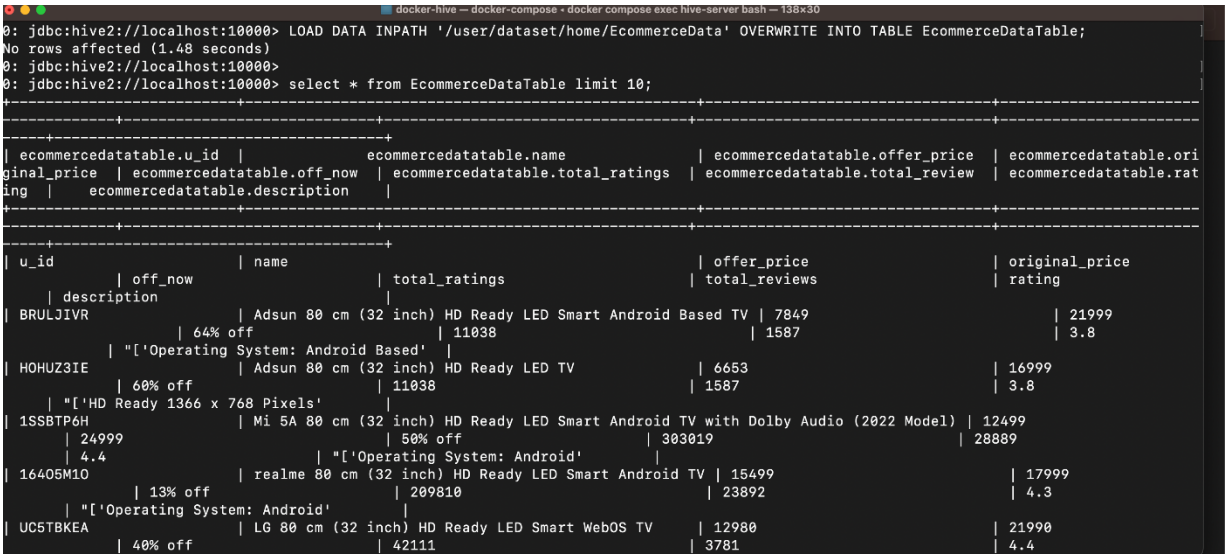



4. Saving Data to Hadoop



5. Create the table and copy the values into it from our Data

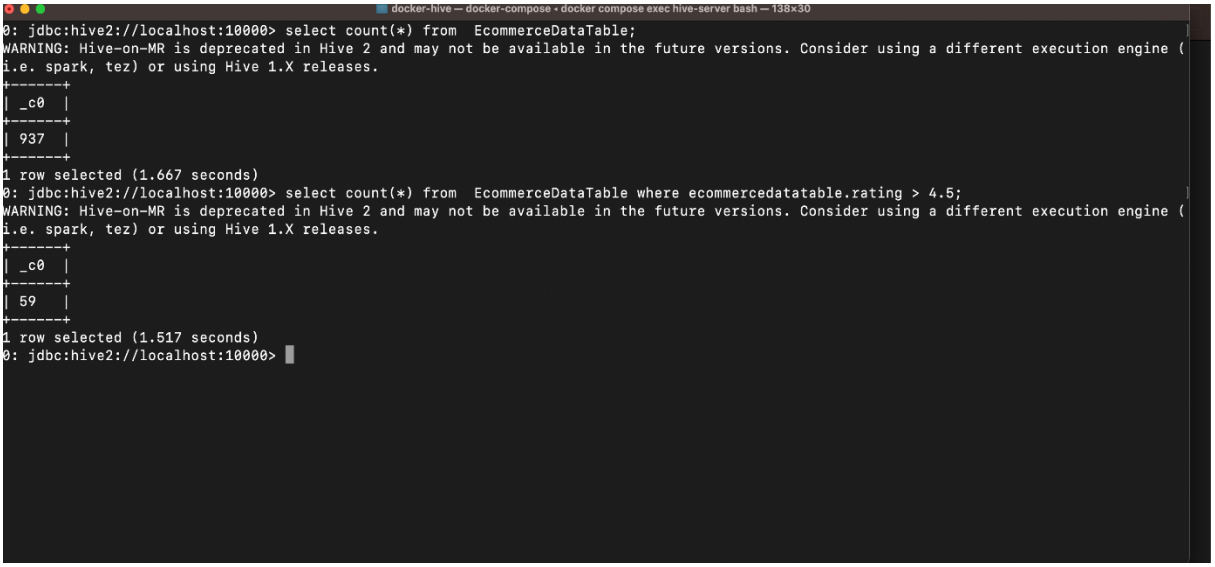


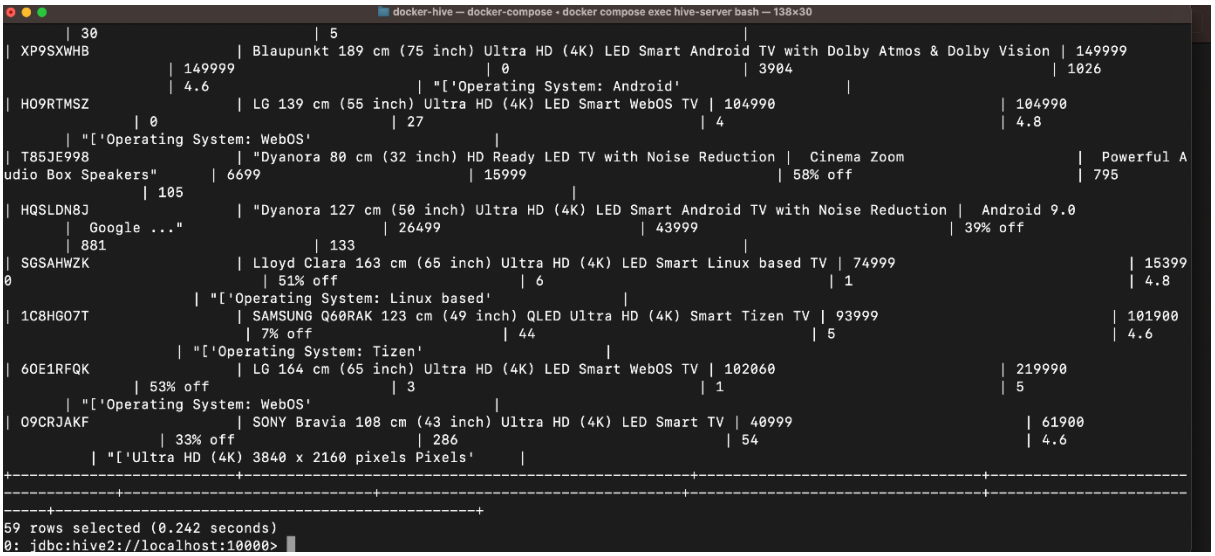



We have created the table by fetching the data from hadoop, now it's time to run the Hive Queries.

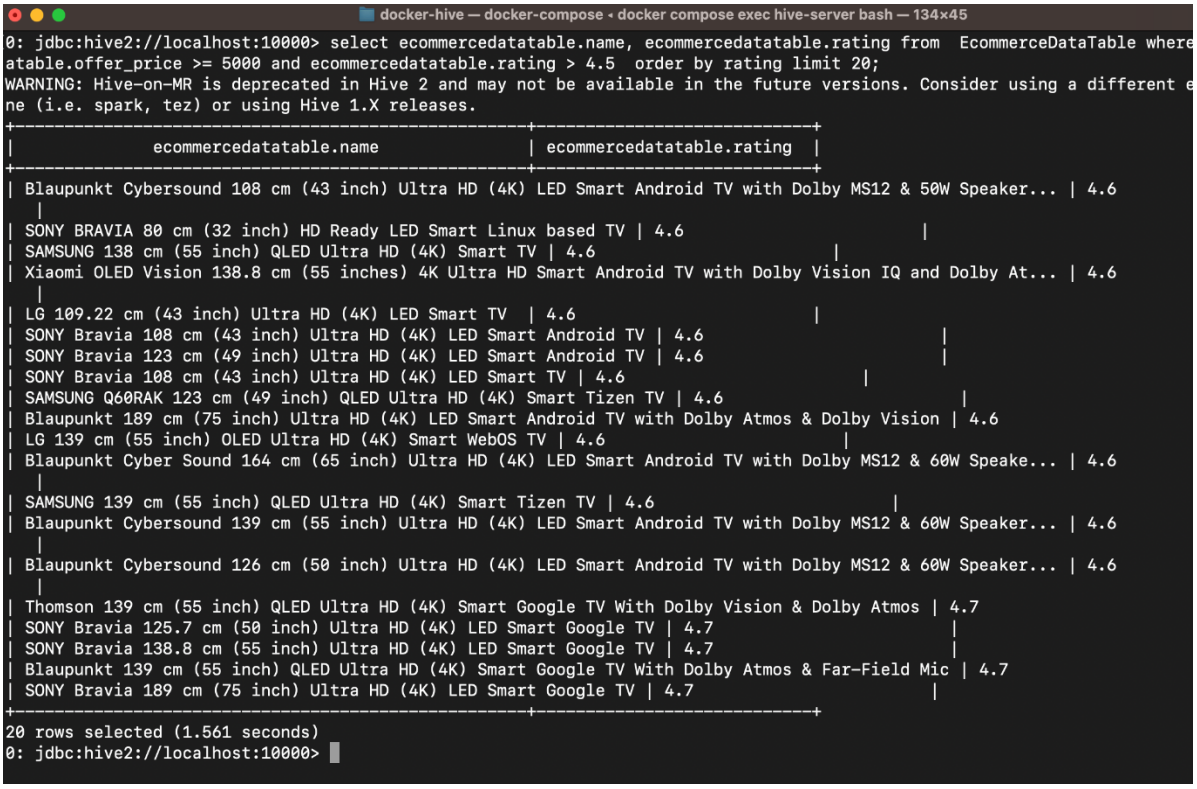
Check if queries are running fine

Query: Get count and data of all records where rating is greater than 4.5



Query: get 20 records where offer_price >= 5000 and rating > 4.5 and order by rating




Query: get 20 records where total reviews are less than 100 and also total ratings is less than 10000.