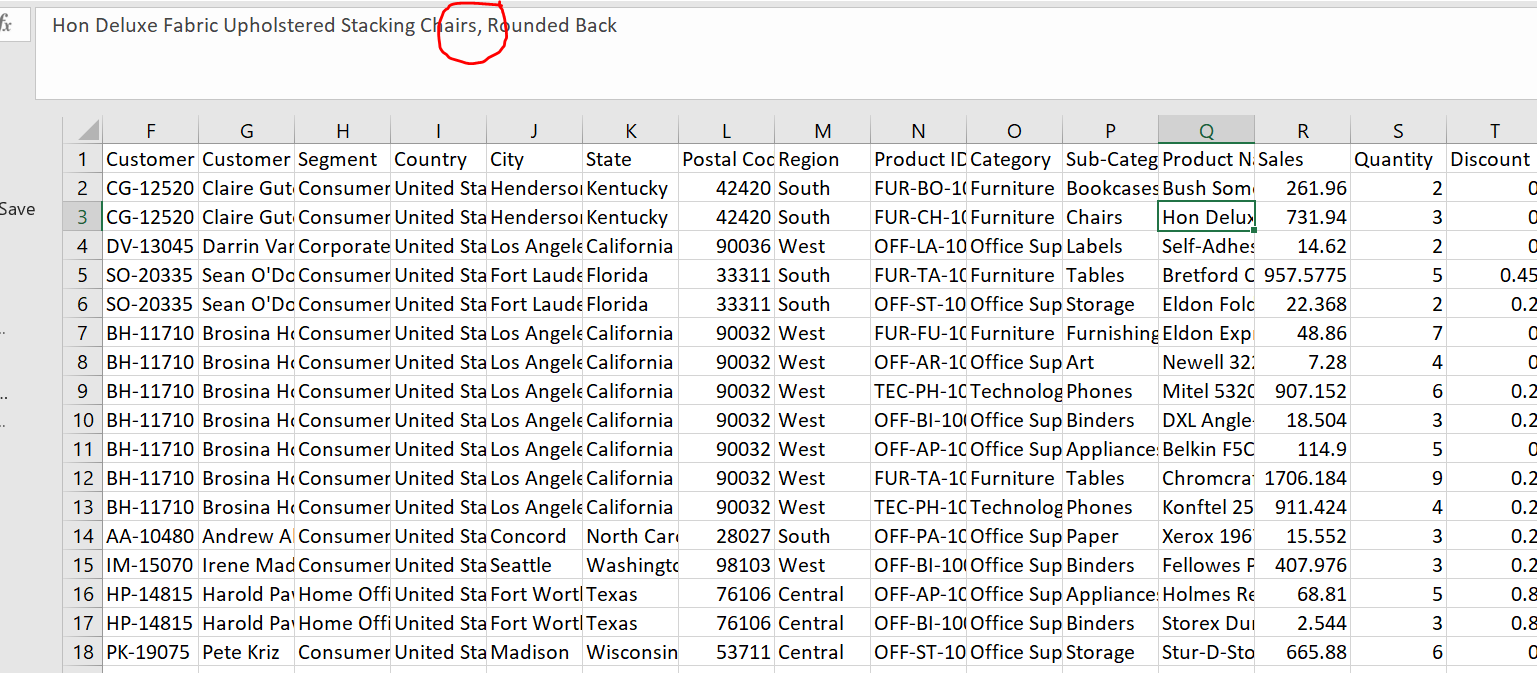
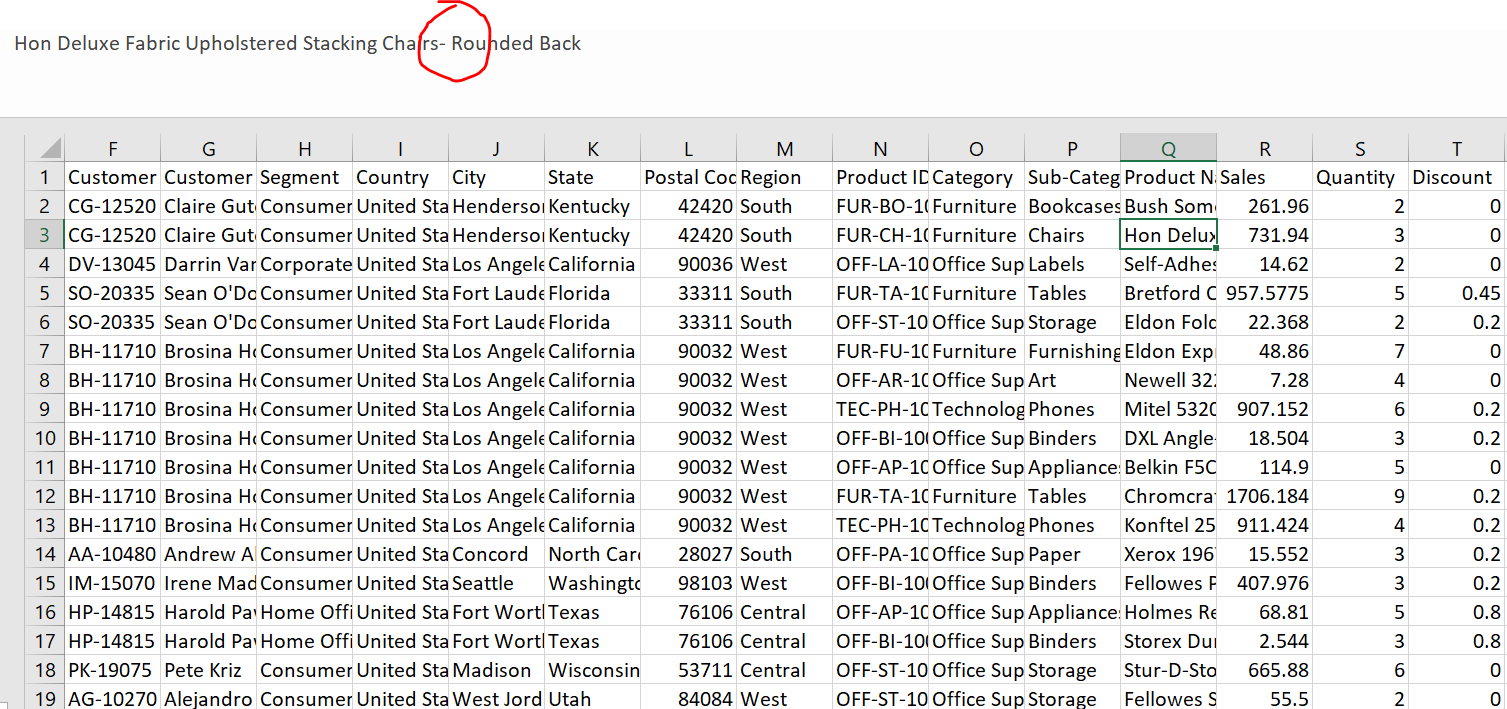
**Speed Comparison of Big Data processing engines on Google cloud platform**

**PART I: Data Preprocessing**

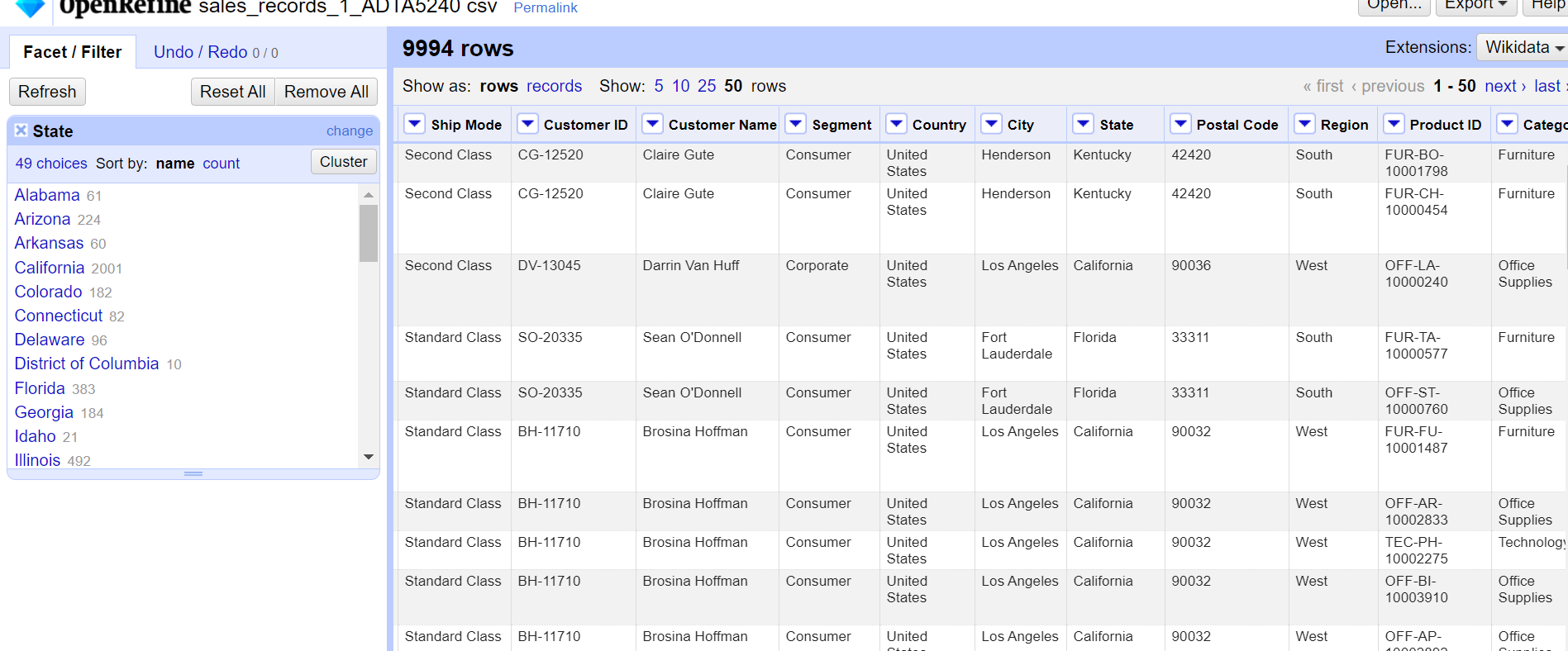
As said in ‘Data analytics life cycle’, Data preprocessing is the most important step that should be done before performing an analysis or else it could affect the decision making of an organization. Below screen shot shows the presence of ‘,’ in the ‘Product Name’ column:



Using the Excel ‘replace’ option I have replaced it with a hyphen. Below is the screenshot showing the result.



I can see that names of all the states are correct without any Typo error. (Used Facet Option available in Open Refine Tool)



**PART II: Apache Hadoop Ecosystem with Hive**

* Uploaded the cleaned dataset onto the cloud storage. The storage bucket and its subdirectories have already been created and used in ADTA 5240 course as a part of class assignments. Below screenshot shows us the file copied from bucket storage to a folder on VM.

**Command Used:**

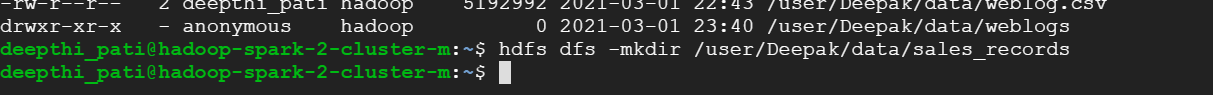
gsutil cp gs://adta-5240-bucket deepak/Data/sales\_records\_1\_ADTA5240.csv sales\_records.csv



* A folder names ‘sales\_records’ is created in Hadoop.

**Command Used:**

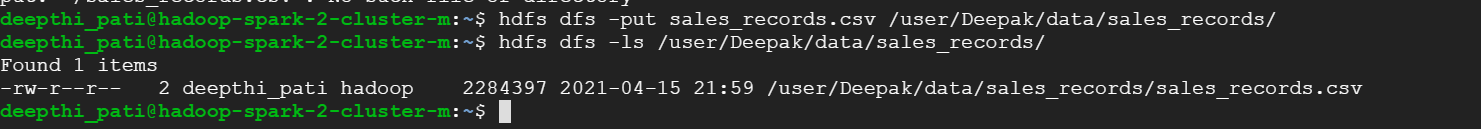
hdfs dfs -mkdir /user/Deepak/data/sales\_records



* Transferred data (sales-records) from VM to Hadoop ‘sales\_records’ directory.

**Command Used:**

1. hdfs dfs -put sales\_records.csv /user/Deepak/data/sales\_records/
2. hdfs dfs -ls /user/Deepak/data/sales\_records/



**(A) The top five states where the company has the most number of customers.**

**Commands Used:**

**Hive Table Creation:**

CREATE EXTERNAL TABLE IF NOT EXISTS sales\_records\_3

(RowID int,

OrderID string,

OrderDate string,

ShipDate string,

ShipMode string,

CustomerID string,

CustomerName string,

Segment String,

Country string,

City string,

State string,

PostalCode string,

Region string,

ProductID string,

Category string,

Subcategory string,

ProductName string,

Sales decimal,

Quantity int,

Discount decimal,

Profit decimal

)

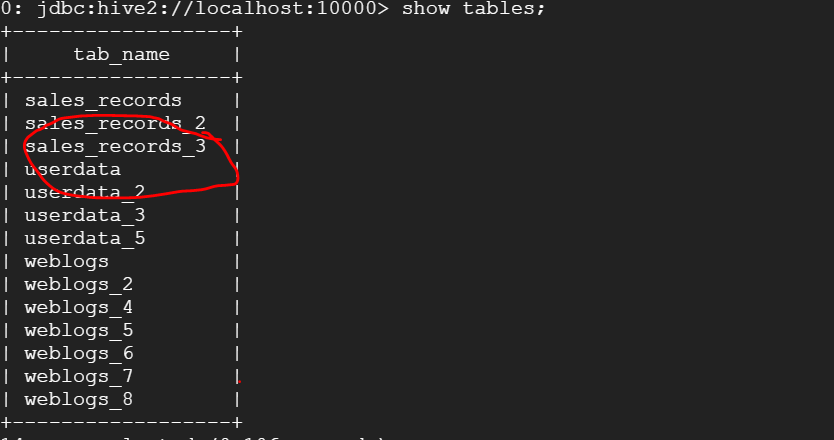
ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

LOCATION '/user/Deepak/data/sales\_records/'

tblproperties ("skip.header.line.count"="1");



**Querying the Hive table:**

**(A) The top five states where the company has the most number of customers.**

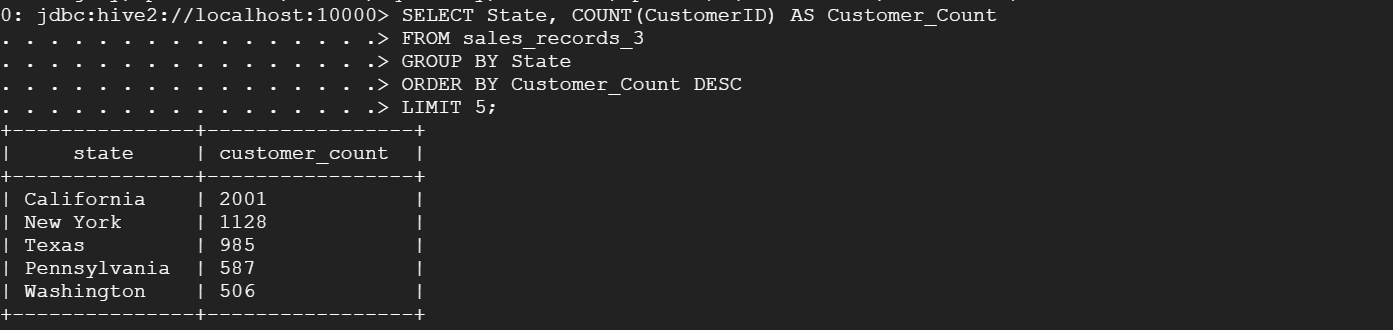
SELECT State, COUNT(CustomerID) AS Customer\_Count

FROM sales\_records\_3

GROUP BY State

ORDER BY Customer\_Count DESC

LIMIT 5;



Above picture shows us the top five states with highest number of customers.

California got the highest number of customers with ‘2001’ count, New York- 1128, Texas- 985, Pennsylvania- 587, Washington- 506.

**(b) The top ten zip-code areas where the company gets the best sale amounts, i.e., the highest sale figures**

**Command used:**

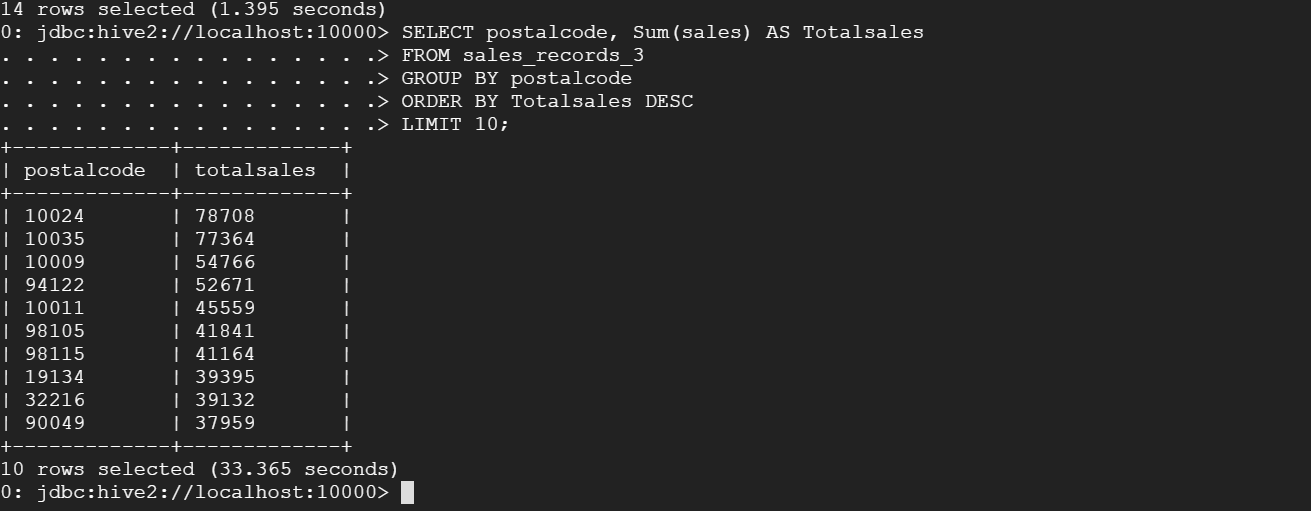
SELECT postalcode, Sum(sales) AS Totalsales

FROM sales\_records\_3

GROUP BY postalcode

ORDER BY Totalsales DESC

LIMIT 10;

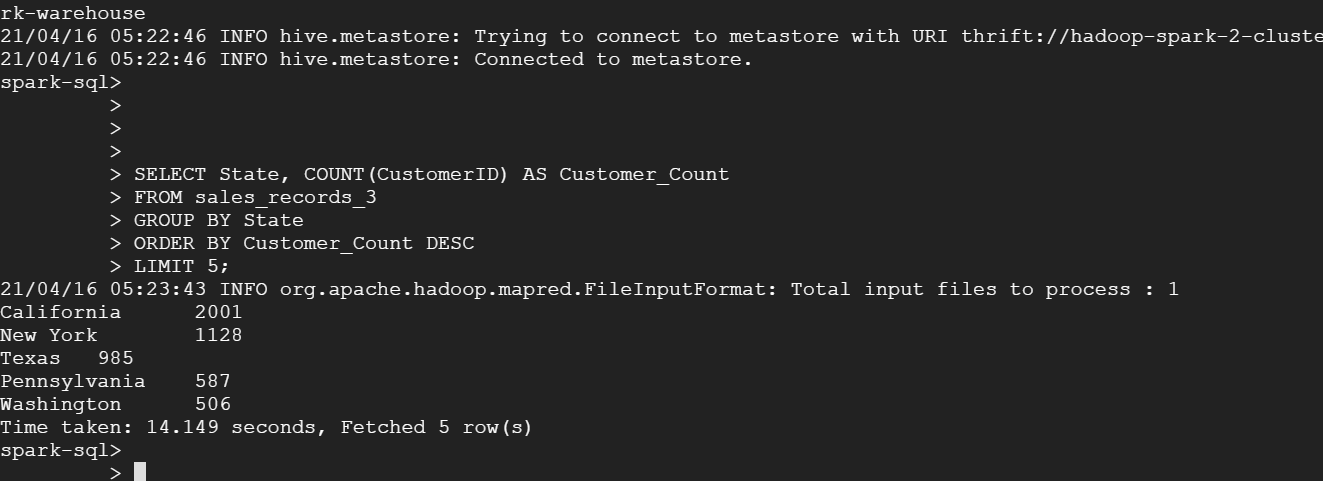


Above picture shows top 10 postal codes considering its sum of total sale amounts.

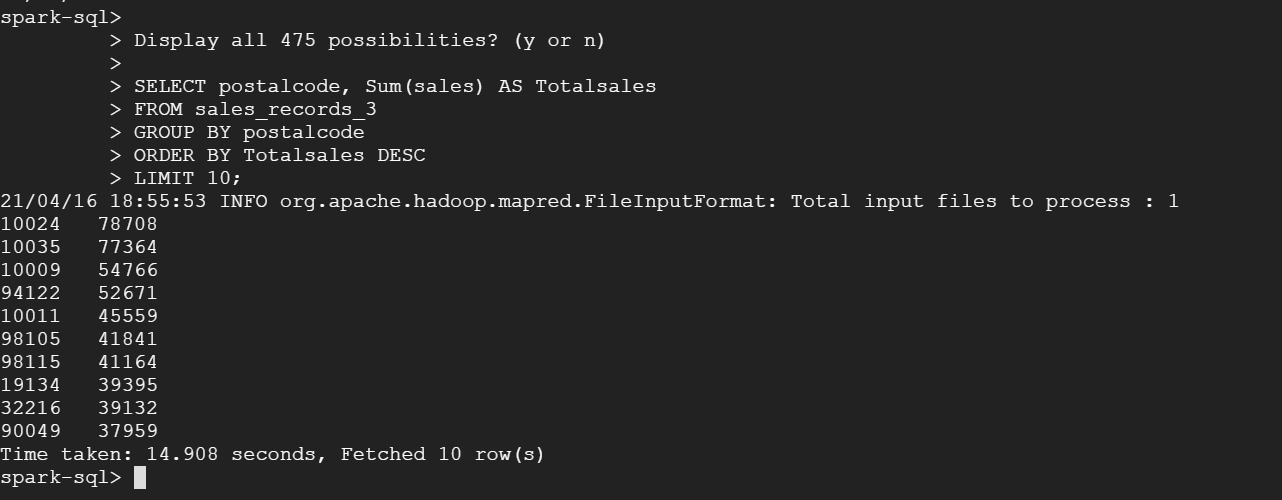
**PART 3: Apache Hadoop Ecosystem with Spark**

Here Spark is using the Hive meta store which contains all the table structure.

1. **The top five states where the company has the most number of customers. (Using Spark)**

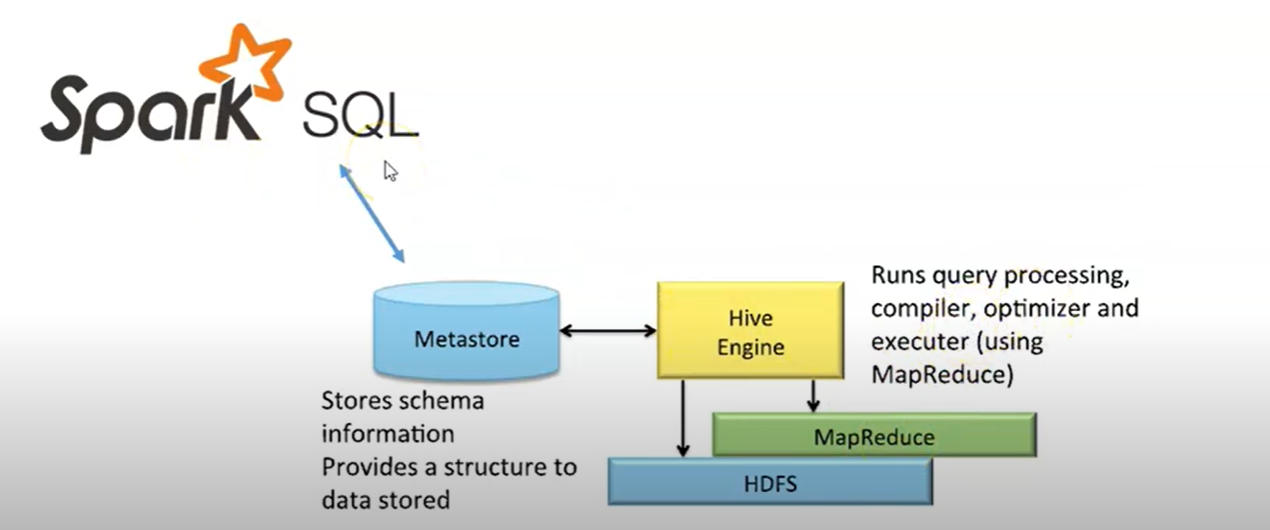


**(b) The top ten zip-code areas where the company gets the best sale amounts, i.e., the highest sale figures (Using Spark)**



I can see by using both Spark & Hive similar results were observed.

**PART 4: Compare Speed and Performance of Hive/MapReduce and Spark**



Yes, there is a large gap between the executions. Below is the reasons and explanations supporting the gap.

1. **Hive and Map reduce:**

* Initially HIVE driver collects the query from UI and sends it to the compiler.
* Compiler collects the metadata from Hive ‘Meta store’
* Compiler sends the plan to driver & driver sends the execution plan to execution engine.
* Execution jobs are requested on Hadoop (map reduce) by the execution engine of Hive.
* Once the job is done Map reduce returns the job results to execution engine again.
* Execution engine passes the results to driver & it displays to UI.

1. **Spark SQL:**

* Hive works with Mapreduce to execute a query; However, Spark SQL does not need Map reduce.
* Spark SQL directly works meta store to execute its queries.

As we can see in the above execution procedures, Hive takes a long time to execute a query as it need to perform Map reduce operation, but in case of Spark SQL the execution time was faster.

Let’s see few more differences of HIVE and spark SQL,

1. Hive provides schema flexibility, portioning and bucketing the tables whereas Spark SQL performs SQL querying it is only possible to read data from existing Hive installation.
2. Hive provides access rights for users, roles as well as groups whereas no facility to provide access rights to a user is provided by Spark SQL
3. Hive provides the facility of selective replication factor for redundant storage of data whereas spark SQL, on the other hand, does not provide any replication factor for storing data
4. As JDBC, ODBC and thrift drivers are available in Hive, we can use them to generate results whereas in case of Apache Spark SQL we can retrieve results in the form of Datasets and [Data Frame](https://www.educba.com/spark-dataframe/) APIs if Spark SQL is run with another programming language
5. There are several limitations:

* Row-level updates and real-time OLTP querying is not possible using Apache Hive whereas row-level updates and real-time online transaction processing is possible using Spark SQL.
* Provides acceptable high latency for interactive data browsing whereas in Spark SQL the latency provided is up to the minimum to enhance performance.
* Hive, like SQL statements and queries, supports UNION type whereas Spark SQL is incapable of supporting UNION type.

**PART 5: Final Presentation Videos: YouTube Links (Final Project Presentation)**

YouTube Video Link:

<https://youtu.be/6wLXFD0pOyk>