An Experimental Study On Different Data Models In Apache Hive

Jisha Mariam[#1]

[#]*Dept of Computer Science, New Horizon College of Engineering*

[1]jmjisha@gmail.com

Abstract— **Apache Hive is an open-source data warehousing solution built on top of Hadoop that has been used for performing various analysis by organization from different sectors. Hive support SQL similar language for querying called as Hive Query language (HQL). The Hive compiler converts the code written in HiveQL to MapReduce Programs automatically. Hive also includes a database or metastore which is used for storing schemas and thus used in query optimization. In addition, HiveQL can be run in both interactive mode and as Hive scripts. It also supports various data types including primitive types, collection of arrays, structures,maps. In this paper, I have tried to perform analysis using a dataset on different data models in Hive. A comparison study is performed on Total query execution time and total CPU time spent as parameters.**

Keywords— **Hive, HiveQL, Data Models, Metastore, Hive Analysis, Hive Partition, Hive Bucketing**

I. Introduction

Hive is a data warehousing infrastructure tool in Hadoop Ecosystem which provides SQL similar language HiveQL for querying and analysing Big Data. The motivation behind the development of Hive is the friction-less learning path for SQL developers & analyst. Hive has helped the programmers by saving the time spend on writing long Java MapReduce Programmers. Apache Hive is a data warehouse system built on top of Hadoop and is used for analysing structured and semi-structured data.

A. *Page Layout*

Before 2008, all the data processing infrastructure in Facebook was built around a data warehouse based on commercial RDBMS. At that time RDBMS was sufficient enough to store and analyse the data. Later it became a challenge for RDBMS to handle such large volume, velocity and variety of data for storage and perform SQL analysis. According to a Facebook article, the data scaled from a 15 TB data set in 2007 to a 2 PB data in 2009. And the infrastructure available at that time was not sufficient to cope up with the speed of data got generated and the daily data analysis process was getting delayed. So, they needed a scalable and economical solution to cope up with this very problem and, therefore started using the Hadoop framework. But the challenge with Hadoop MapReduce was to write thousand lines of MapReduce code in Java for simple analysis and all programmers were not having good knowledge in Java unlike RDBMS SQL.

Later Facebook developers thought of building a tool on top of Hadoop that can make Hadoop accessible to users with SQL background and thus they build Hive in January 2007. The vision was to bring the familiar concepts of tables, columns, partitions and a subset of SQL to the unstructured world of Hadoop, while still maintaining the extensibility and flexibility that Hadoop enjoyed [1]. In August 2008, Hive was made available as open source and it became very popular among all users within Facebook. This helped in running thousands of jobson the Hadoop/Hive cluster with hundreds of users for a wide variety of applications starting from simple summarization jobs to business intelligence, machine learning applications and to also support Facebook product features [1].

B. *Featuers*

Hive became very popular among non-programmers also as it eliminates the need for writing complex MapReduce programs. Some of it features are:

- It provides simple SQL similar language called HiveQL
- It is good for Online Analytic Processing(OLAP)
- It can be scalable to accommodate the growing volume and variety of data without affecting performance of the system.
- It works efficiently as ETL (Extract, Transform, Load) tool
- Hive supports any client application written in Java, PHP, Python, C++ or Ruby by exposing its Thrift server
- It stores the metadata information in metastore that helps to perform semantic checks during query execution and the data to be processed in Hadoop Distributed Filesystem(HDFS).

Apache Hive is used as data warehousing and for ad-hoc analysis. Hive has some limitation too like it cannot be used for real time queries and Online Transaction Processing. Therefore, Hive is coupled with other tools to make it useful in other domains too. For example, Tableau along with Apache Hive can be used for Data Visualization, Apache Tez integration with Hive provide real time processing capabilities, etc.

II. HIVE ARCHITECTURE

The main components of Hive architecture are User interfaces, Driver Metastore, Hive Query Compiler, Execution Engine, HDFS or HBASE.

Interfaces: Hive provides many user interfaces like Hive Web UI, Hive command line, and Hive HD Insight (In Windows server), application programming interfaces (API) like JDBC and ODBC for interaction between user and HDFS [2].

MetaStore: Hive stores the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping in database. The default database is Derby database. Metastore plays a very important role in imposing structure on Hadoop files. Any metadata that is needed by the mapper or the reducer is passed through xml plan files that are generated by the compiler and that contain any information that is needed at the run time [1].

The Driver manages: It manages the life cycle of a HiveQL statement during compilation, optimization and execution. On receiving the HiveQL statement, from the thrift server or other interfaces, it creates a session handle which is used to keep track of statistics like execution time, number of output rows, etc [2].

Query Compiler: The compiler processes the HiveQL statements to generate execution plan with the help of metadata stored in Metastore. At first, parser generates abstract syntax tree (AST) for the query. During this phase, type checking and semantics analysis are done by fetching the information of all input and output tables from the Metastore. This information used for building the logic plan or DAG(Direct Acyclic Graph). Later a chain of transformations is applied as a part of optimization logic where the operator DAG resulting from one transformation is passed as input to next transformation. The logical plan generated at the end of the optimization phase is then split into multiple map/reduce and HDFS tasks. At the end of this stage the physical plan is generated that looks like a DAG of tasks with each task encapsulating a part of the plan.

Execution Engine: Finally, this engine executes the execution plan created by the compiler. It manages the dependencies between different stages of DAG by executing the tasks in order of their dependencies. Each dependent task is executed only after the execution of all of its prerequisites. Execution Engine is the vital component as it directly interacts with Job Tracker, *Name Node* and *Data nodes and* produce a series of Map Reduce Jobs[4].

Hadoop MapReduce: In the execution of a Map/Reduce task, the operators inside this task are first initialized and then they will process the rows fetched by the MapReduce engine in a pipelined fashion. To perform read/write operation of a particular file format, Hive assigns the corresponding file reader/writer to the tasks that perform reading/writing of the table. For a file format, a serialization-deserialization library (called SerDe) is used to serialize and deserialize data. After all MapReduce jobs have finished, the Driver will fetch the results of the query to the user who submitted the query [5].

Hive processes the data not only stored in HDFS but also stored in other storage systems like HBase. In such case, a corresponding storage handler like HBase storage handler is used for reading and writing data stored on HBase.

III. HIVE DATA MODELLING

Databases: Namespaces function to avoid naming conflicts for tables, views, partitions, columns, and so on. Databases can also be used to enforce security for a user or group of users.

Tables*:* Homogeneous units of data which have the same schema. An example of a table could be **‘student’** table, where each row could comprise of the following columns (schema):

- USN —which is of BIGINT type
- SNAME—which is of STRING type
- AGE-which is of INT type
- CITY—which is of STRING type
- DOJ-which is of TIMESTAMP

Managed Table: As the name suggests (managed table), Hive is responsible for managing the data of a managed table. In other words, if you load the data from a file present in HDFS into a Hive Managed Table and issue a DROP command on it, the table along with its metadata will be deleted. So, the data belonging to the dropped managed_table no longer exist anywhere in HDFS and you can't retrieve it by any means.

External Table: For external table, Hive is not responsible for managing the data. In this case, when you issue the LOAD command, Hive moves the data into its warehouse directory. Then, Hive creates the metadata information for the external table. Now, if you issue a DROP command on the external table, only metadata information regarding the external table will be deleted. Therefore, you can still retrieve the data of that external table from the warehouse directory using HDFS commands.

Partitions*:* Each Table can have one or more partition Keys which determines the data storage. Partitions, apart from being storage units, also allow the user to efficiently identify the rows that satisfy a specified criterion; for example, a date_partition of type STRING. Each unique value of the partition keys defines a partition of the Table. For example, all students are partitioned year wise, then analysis on students of a particular batch can be done by referring only to that particular year partition.

Buckets *(or* ***Clusters****):* Data in each partition may in turn be divided into Buckets based on the value of a hash function of some column of the Table. For example, the "student" table may be bucketed by "city", which is one of the columns, other than the partitions columns, of the "student" table. These can be used to efficiently sample the data.

IV. IMPLEMENTATATION

A. Experimental Setup:

A real-time data collected by Govt of India from the field instruments directly without human intervention from CPCB is used in this paper. This data has been converted to .csv files and loaded to HDFS. It contains Real time National Air Quality Index values from different monitoring stations across India. The pollutants monitored are Sulphur Dioxide (SO2), Nitrogen Dioxide (NO2), Particulate Matter (PM10 and PM2.5), Carbon Monoxide (CO), Ozone(O3) etc. The attributes are Country, State, City, Place, last Update Time, Average Quality Index, Maximum Quality Index, Minimum Quality Index, Pollutant.

(i) Creation of simple managed table

```
create table air(country string,state string,city string,place string,lastup string,av int,ma int,mi int,pollutant string) row format delimited fields terminated by ',' lines terminated by '\n';
```

Loading of data from HDFS to Hive Managed table

```
load data inpath '/user/cloudera/AirQuality.csv' into table air;
```

(ii) Creation of Partitioned table

```
create table part_air(country string, city string, place string, lastup string, av int, ma int, mi int, pollutant string)
PARTITIONED BY (state string);
```

- The column used for partitioning should have low cardinality i.e. low distinct values for that column. Because if we use the column with high cardinality, then we will end up with many sub directories or files. Since the number of mappers is dependent on input size and block size, creating many partitions would end up using many mappers and it would lead to wastage of resources in most cases.
- If there are many partitions, then the Name Node which keeps track of meta data of file system in memory will also have unnecessary overhead since it has to track many partitions now.

Setting up property for partition

```
SET hive.exec.dynamic.partition=true;
SET hive.exec.dynamic.partition.mode=non-strict;
```

Loading data into Partition table

```
Insert overwrite table part_air partition(state) select country,city,place,lastup,av,ma,mi,pollutant,state from air;
```

Hive Warehouse ‘part_air’ directory is split into many sub directories based on the partition column (state) as shown below:

```
$hadoop fs -ls /user/hive/warehouse/part_air
    drwxrwxrwt  - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=Andhra_Pradesh
    drwxrwxrwt   - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=Bihar
    drwxrwxrwt   - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=Delhi
    drwxrwxrwt   - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=Gujarat
    drwxrwxrwt   - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=Haryana
    drwxrwxrwt   - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=Jharkhand
    drwxrwxrwt   - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=Karnataka
    drwxrwxrwt   - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=Kerala
    drwxr-xr-x   - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=Madhya Pradesh
    drwxrwxrwt   - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=Maharashtra
    drwxrwxrwt   - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=Odisha
    drwxrwxrwt   - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=Punjab
    drwxrwxrwt   - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=Rajasthan
    drwxrwxrwt   - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=TamilNadu
    drwxrwxrwt   - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=Telangana
    drwxrwxrwt   - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=Uttar_Pradesh
    drwxrwxrwt   - cloudera supergroup        0 2019-06-27 03:04 /user/hive/warehouse/part_air/state=West_Bengal
```

We can also filter out a particular partitioned sub directory directly from HDFS as shown below:

```
$hadoop fs -ls /user/hive/warehouse/part_air/state=Punjab
    -rw-rw-rw-3 cloudera supergroup3874 2019-06-27 03:04 /user/hive/warehouse/part_air/state=Punjab/000000_0
```

(iii) Creation of table with bucketing

Create Table buckt_air(country string, state string, city string, place string, lastup string, av int, ma int, mi int, pollutant string) clustered by (city) into 4 buckets;

Bucketing concept is based on hashing function on bucketed column. The records which generate same hash will always be in the same bucket. To divide a table into buckets we use Clustered by clause. Each bucket is just like a file in directory and all files are equally distributed.

The advantages of bucketing are:

- Map-Side joins are faster on bucketed tables because they are of similar size.
- We can keep the records sorted in each bucket
- When the data is sorted by a column in buckets and they are used for join, Map-side joins are even faster
- They also offer efficient sampling over non-bucketed tables
- With bucketing we can always define number of buckets to be formed which is not the case in partitioning
- Bucketing can be used with or without partitioning

Set the properties for bucketing

SET hive.enforce.bucketing =true;

Loading the content into table with bucketing

insert overwrite table buckt_air select country, state, city, place, lastup, av, ma, mi, pollutant from air;

The entire content is divided into 4 buckets as shown below:

```
$hadoop fs -ls /user/hive/warehouse/buckt_air
   -rw-rw-rw-   3 cloudera supergroup      36475 2019-06-27 03:31 /user/hive/warehouse/buckt_air/000000_0
   -rw-rw-rw-   3 cloudera supergroup      19548 2019-06-27 03:31 /user/hive/warehouse/buckt_air/000001_0
   -rw-rw-rw-   3 cloudera supergroup       5963 2019-06-27 03:31 /user/hive/warehouse/buckt_air/000002_0
   -rw-rw-rw-   3 cloudera supergroup       9181 2019-06-27 03:31 /user/hive/warehouse/buckt_air/000003_0
```

(iv) Creation of table with partitionbucketing

```
create table partcluster_air(country string, city string, place string, lastup string, av int, ma int, mi int, pollutant string)
PARTITIONED BY (state string) clustered by (city) into 4 buckets;
```

Set the properties for partition and bucketing

```
SET hive.exec.dynamic.partition=true;
SET hive.exec.dynamic.partition.mode=non-strict;
SET hive.enforce.bucketing =true;
```

Loading the data into table with Partition bucketing

```
insert overwrite table partcluster_air partition(state) select country,city,place,lastup,av,ma,mi,pollutant,state from air;
```

The table partcluster_air is partitioned to many sub-directories based on the partition column(state) as follows:

```
$hadoop fs -ls /user/hive/warehouse/partcluster_air
    drwxrwxrwt-cloudera supergroup 0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Andhra_Pradesh
    drwxrwxrwt   - cloudera supergroup          0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Bihar
    drwxrwxrwt   - cloudera supergroup          0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Delhi
    drwxrwxrwt   - cloudera supergroup          0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Gujarat
    drwxrwxrwt   - cloudera supergroup          0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Haryana
    drwxrwxrwt   - cloudera supergroup          0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Jharkhand
    drwxrwxrwt   - cloudera supergroup          0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Karnataka
    drwxrwxrwt   - cloudera supergroup          0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Kerala
    drwxr-xr-x   - cloudera supergroup          0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Madhya Pradesh
    drwxrwxrwt   - cloudera supergroup          0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Maharashtra
    drwxrwxrwt   - cloudera supergroup          0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Odisha
    drwxrwxrwt   - cloudera supergroup          0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Punjab
    drwxrwxrwt   - cloudera supergroup          0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Rajasthan
    drwxrwxrwt   - cloudera supergroup          0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=TamilNadu
    drwxrwxrwt   - cloudera supergroup          0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Telangana
    drwxrwxrwt   - cloudera supergroup          0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Uttar_Pradesh
    drwxrwxrwt   - cloudera supergroup          0 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=West_Bengal
```

One of the partitioned sub-directories named as “Punjab” is further divided into four files or buckets based on the clustering condition as shown below:

```
$hadoop fs -ls /user/hive/warehouse/partcluster_air/state=Punjab
    -rw-rw-rw-   3 cloudera supergroup       1431 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Punjab/000000_0
    -rw-rw-rw-   3 cloudera supergroup        868 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Punjab/000001_0
    -rw-rw-rw-   3 cloudera supergroup        665 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Punjab/000002_0
    -rw-rw-rw-   3 cloudera supergroup        910 2019-06-27 03:42 /user/hive/warehouse/partcluster_air/state=Punjab/000003_0
```

B. Analysis

An analysis has been performed using different data models of Hive for different sets of queries. Total five queries have been executed on Hive managed table, table with partitioning, table with bucketing and table with partition bucketing concept. Total CPU time spent and query execution time have been compared for each of the query among different data model tables [6].

TABLE I
LIST OF QUEIES

Query. No	Query Description
Query 1	Compare the maximum and minimum level of pollution for each pollutant in the city “Bengaluru”
Query 2	Analyse the count of pollutant for each city of “Uttar_Pradesh”
Query 3	Compare the average, maximum, minimum pollution level for various pollutant in each city of “Rajasthan”
Query 4	Analyse the average pollutant level in each city of “Andhra Pradesh” in ascending order of cities.
Query 5	Compare the average, maximum, minimum “ozone” level in

	each major capital cities in ascending order of city names.

Query 1: Compare the maximum and minimum level of pollution for each pollutant in the city "Bengaluru"**.**
Using Simple Hive Managed table
select pollutant, MAX(ma),MIN(mi) from air where city='Bengaluru' group by pollutant;
Using Hive Partitioned table
select pollutant, MAX(ma),MIN(mi) from part_air where city='Bengaluru' group by pollutant;

Using Hive Bucketing table
select pollutant, MAX(ma),MIN(mi) from buckt_air where city='Bengaluru' group by pollutant;

Using Hive Partition Bucketing table
select pollutant, MAX(ma),MIN(mi) from partcluster_air where city='Bengaluru' group by pollutant;

Query 2: Analyse the count of pollutant for each city of "Uttar_Pradesh"
Using Simple Hive Managed table
select city, COUNT(pollutant) from air where state='Uttar_Pradesh' group by city;

Similarly, the query is executed on Hive Partitioned table, Hive Bucketing table, Hive Partition Bucketing table

Query 3: Compare the average, maximum, minimum pollution level for various pollutant in each city of "Rajasthan"
Using Simple Hive Managed table
select city, pollutant,av,ma,mi from air where state='Rajasthan' order by city;

Similarly, the query is executed on Hive Partitioned table, Hive Bucketing table, Hive Partition Bucketing table

Query 4: Analyse the average pollutant level in each city of "Andhra Pradesh" in ascending order of cities.
Using Simple Hive Managed table
select city, AVG(av) from air where state='Andhra_Pradesh' group by city order by city;

Similarly, the query is executed on Hive Partitioned table, Hive Bucketing table, Hive Partition Bucketing table

Query 5: Compare the average, maximum, minimum "ozone" level in each major capital cities in ascending order of city names.
Using Simple Hive Managed table
select AVG(av),MAX(ma),MIN(mi),city from air where city in ('Amaravati','Patna','Ahmedabad','Bengaluru','Mumbai','Jaipur','Chennai', 'Hyderabad','Lucknow','Kolkata','Delhi') and pollutant='OZONE' group by city;

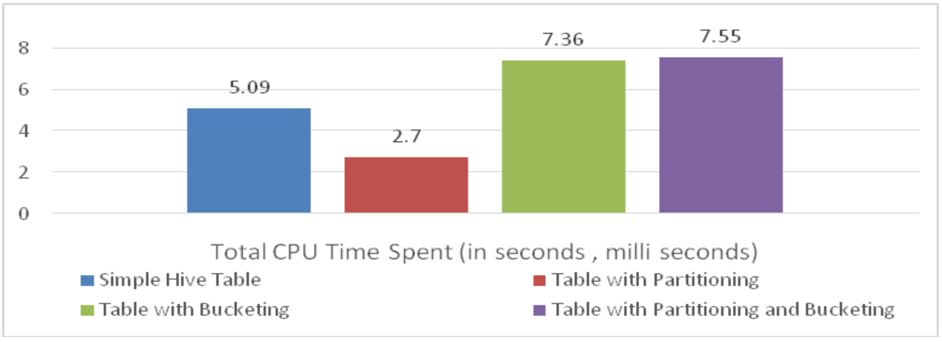
Similarly, the query is executed on Hive Partitioned table, Hive Bucketing table, Hive Partition Bucketing table.

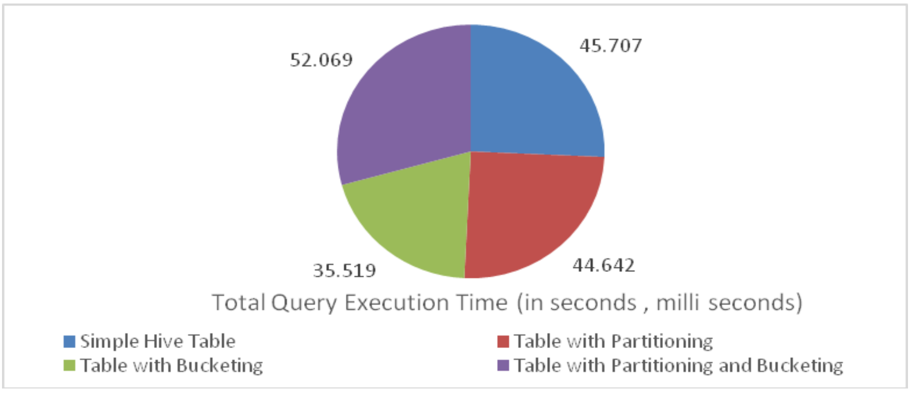
V. RESULT AND DISCUSSION

The study on various data model in Hive is accomplished by executing the queries on each of the following tables:
- Hive Managed table
- Hive Partitioned table
- Hive Bucketing table
- Hive Partition Bucketing table

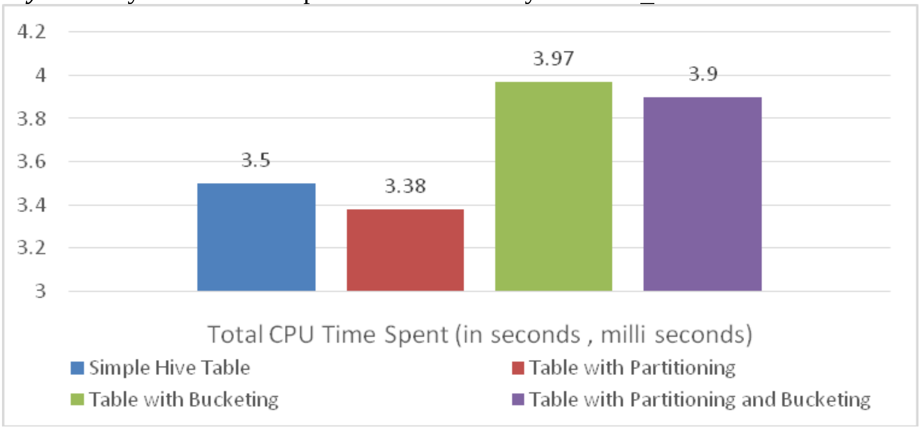
The total CPU Time Spent and Total Query Execution Time are recorded in seconds and milli-seconds for each of the query. These are plotted as charts as shown below:

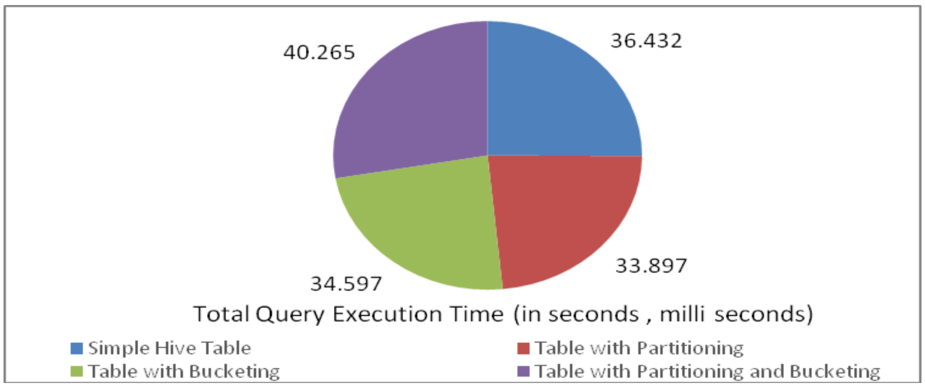
Query 1: Compare the maximum and minimum level of pollution for each pollutant in the city "Bengaluru"**.**



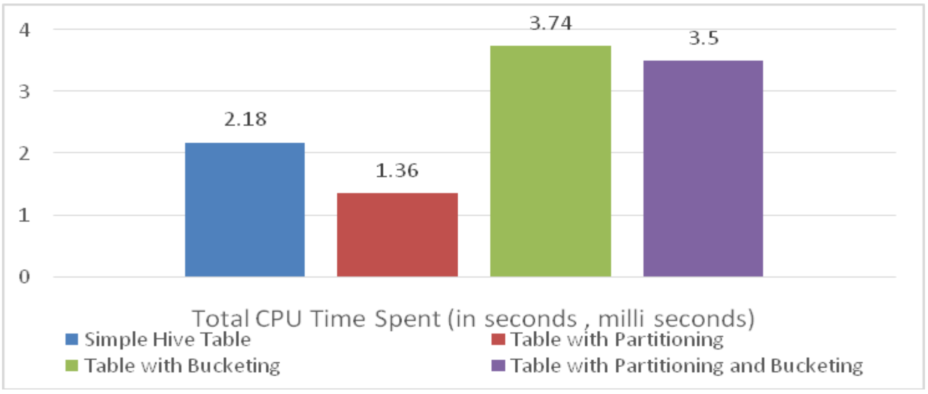


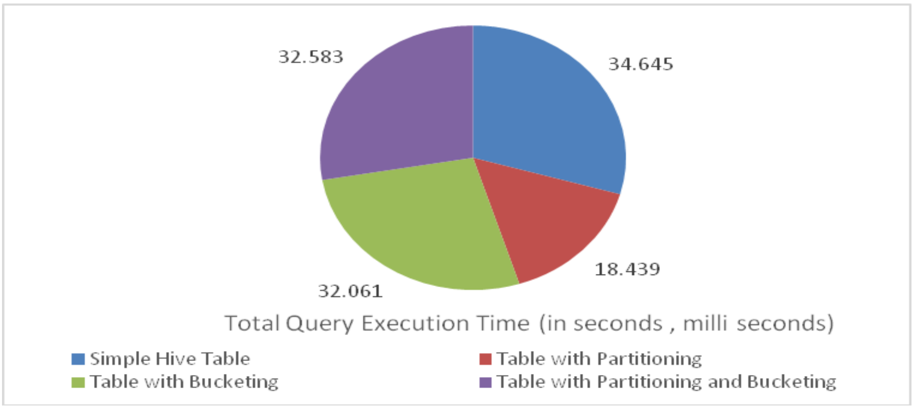
Query 2: Analyse the count of pollutant for each city of "Uttar_Pradesh"



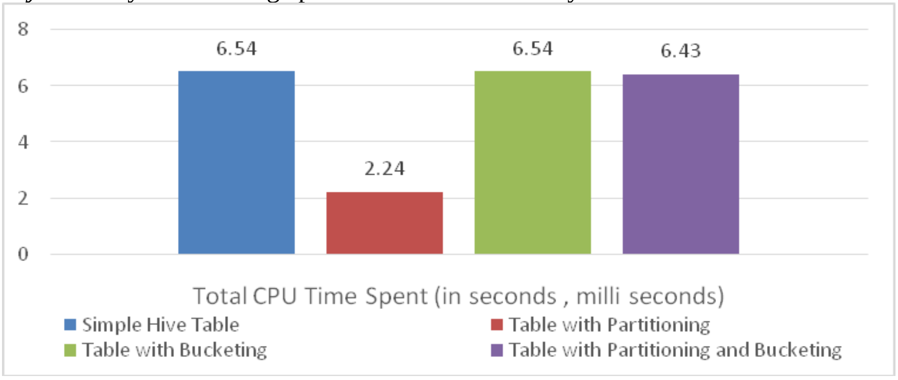


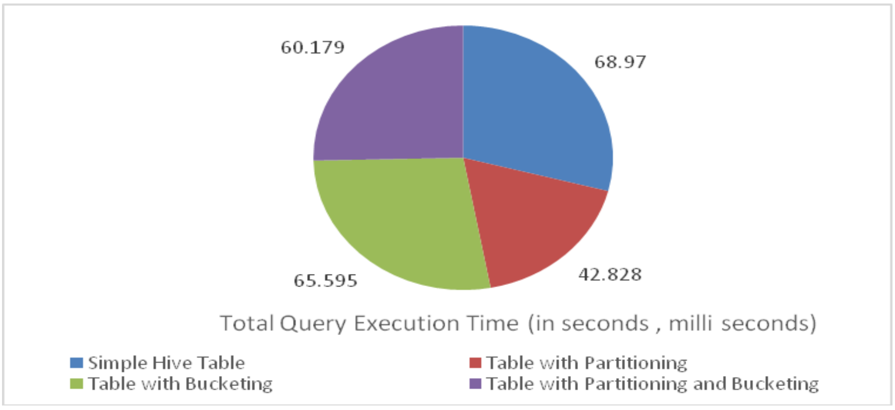
Query 3: Compare the average, maximum, minimum pollution level for various pollutant in each city of "Rajasthan"



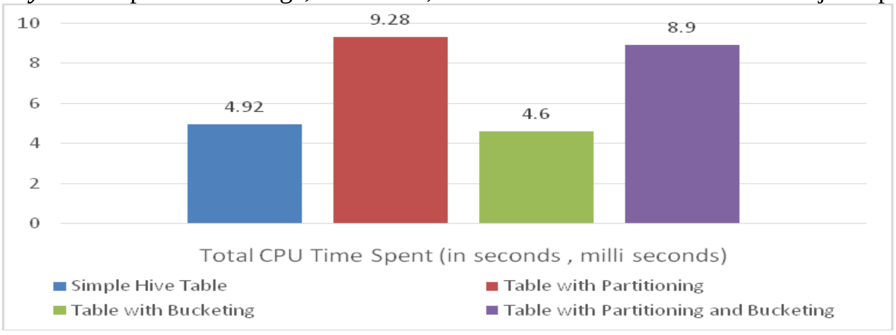


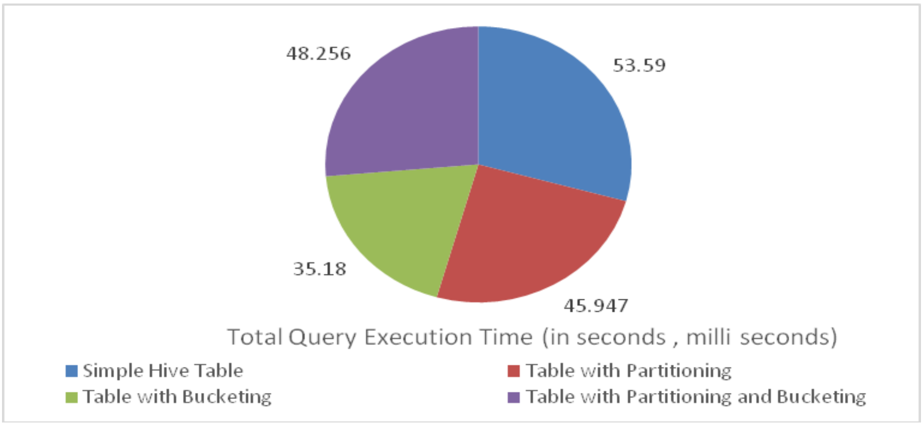
Query 4: Analyse the average pollutant level in each city of "Andhra Pradesh" in ascending order of cities.





Query 5: Compare the average, maximum, minimum "ozone" level in each major capital cities in ascending order of city names.





VI. CONCLUSION

Hive has become an easy learner tool for all programmers and non-programmers to perform various analysis on Hadoop. Anyone with little SQL knowledge can discover various insights on different sectors by using Hive's various features like complex data types, different file format support, various data models etc. This paper will help the Hive users to gain deeper knowledge in various data models in Hive by making them understand how to analyse the datasets using these concepts. Comparison of each query in terms of total CPU time spent and total query execution time helps the user to conclude that simple managed table gives better performance than the table with partition bucketing for some analysis whereas for some other analysis simple partitioned table will give better performance.

REFERENCES

[1] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, et.al , Hive – A Petabyte Scale Data Warehouse Using Hadoop, *Facebook Data Infrastructure Team*

[2] Anish Gupta, Manish K. Gupta, HIVE- Processing Structured Data in HADOOP, International Journal of Scientific & Engineering Research Vol. 8, Issue 6, June2017

[3] Kadhar Basha J, Dr. M. Balamurugan, A Review on Hive and Pig, International Journal of Advanced Research in Basic Engineering Sciences and Technology, Vol. 3, Special Issue 39, May 2017

[4] N. Puspalatha, P.Sudheer, Data Processing in Big Data by using Hive Interface, International Journal of Advance Research in Computer Science and Management Studies, Vol.3, Issue 4, April 2015.

[5] Yin Huai, Ashutosh Chauhan, Alan Gates, Gunther Hagleitner,et. al, Major Technical Advancements in Apache Hive, IGMOD'14, June 2014

[6] J.Ramsingh, Dr.V.Bhuvaneswari, An Insight on Big Data Analytics Using PigScript, *International Journal of Emerging Trends & Technology in Computer Science,* Vol. 4, Issue 6, December 2015