Airline Dataset Analysis using Hadoop, Hive, Pig and Impala

Illinois Institute of Technology, Chicago
by Ayushi Patel and Bhargavi Deshpande

Abstract — This paper is about the analysis of the airline data set which is performed using Cloudera, it delivers the modern platform for analytics optimised for the cloud. The project is performing big data analysis on airline dataset using Hadoop and tools like Pig, Hive and Impala. At the end, analysis of delay will be visualised using Excel spreadsheet.
Keywords : *Hadoop, HDFS, Pig, Hive and Impala, Data Analysis*

INTRODUCTION

There is a lot of excitement that exists with the term Big Data. In simple words, Big Data can be large-scale data which does not have a well-defined structure. The size of the data is so huge that it is not practically easy for a single computer to store and process all the data. In traditional computing approach there are many problems in a different way and the focus was always to increase the processing speed and power of the computer. As the data grows exponentially, the processing power of the single computer becomes a bottleneck and thus a new approach was needed to address the issue at hand[3]. A new way was developed wherein many non-expensive commodity computers were working together in harmony with each other, in order to store and process this big data in parallel. This allows us to extract meaningful information from a large data set. In addition, by using the cloud technology, it is easy to create cluster, compute and release the computing resources when it is not needed. So, from the cloud technology we get the computing power of the cluster of computers with minimal investment[10]. The draft mainly contains details about Hadoop, Hive, Pig and Impala and gives vague idea of the project flow.

TECHNOLOGY USED

- **Understanding of Infrastructure**

Infrastructure is the cornerstone of Big Data architecture. Possessing the right tools for storing, processing and analysing your data is crucial in any Big Data project.

Two major Users of data infrastructure : Human and System. Ways that data can be ingested/served in data infrastructure : Messaging Layer(message oriented middleware) , Data Warehousing(Batch) and Backend Services.

- **Hadoop and Big Data**

Hadoop is one of the tools designed to handle big data. Hadoop and other software products work to interpret or parse the results of big data searches through specific proprietary algorithms and methods. Hadoop is an open-source program under the Apache license that is maintained by a global community of users. Apache Hadoop is 100% open source, and pioneered to be a fundamentally new way of storing and processing data instead of relying on expensive, proprietary hardware and different systems to store and process data.

HDFS : HDFS is the primary distributed storage on Hadoop for managing pools of big data, that spans across large clusters of commodity servers. HDFS is regarded as the bucket of the hadoop ecosystem, where data is dumped and sits there until the user wants to export it to another tool, for running analysis on the stored data. Any machine that supports Java programming language can run HDFS[10].

Working of HDFS : HDFS uses a master slave architecture where each cluster has a NameNode for managing the file system operations and supporting DataNodes for managing data storage on individual computing nodes (usually there exists one DataNode per node in the Hadoop cluster). HDFS stores data in files which are divided into one or more segments and are stored in particular DataNodes. These small segments of files are referred to as block and the minimum amount of data that can be read or written is referred to as a single block. By default the size of a block is 64MB which can be changed in the configuration file.

- **Apache Pig**

Apache Pig is a high level procedural dataflow language on top of Hadoop for processing and analysing big data without having to write Java based MapReduce code. Apache Pig has RDBMS like features- joins, distinct clause, union, etc. For crunching large files containing semi-structured or unstructured data[10]. **Apache pig components:** 1. Pig Latin: It is a SQL like data flow language to join, group and aggregate distributed data sets with ease. 2. Pig Engine: Pig engine takes the Pig Latin scripts written by users, parses them, optimizes them and then executes them as a series of MapReduce jobs on a Hadoop Cluster.

- **Apache Hive: A Data Warehousing Solution for Big Data on Hadoop**

Hive is a data warehousing solution developed on top of Hadoop to meet the big data challenges of storing, managing and processing large data sets without having to write complex Java based MapReduce programs. Hive is a familiar programming model for big data professionals who know SQL but do not have a good grip in programming. Hive is not a relational database or an architecture for online transaction processing[11]. It is particularly designed for online analytical processing systems (OLAP). Hive compiler converts the queries written in HiveQL into MapReduce jobs so that Hadoop developers need not worry much about the complex programming code beyond the processing and they can focus on the business problem. The three important functions performed by Hive include - data summarization, data querying and data analysis.

Apache Hive is extensively used by data scientists and data analysts for data exploration,building data pipelines and for processing ad-hoc queries.

Hive Components :

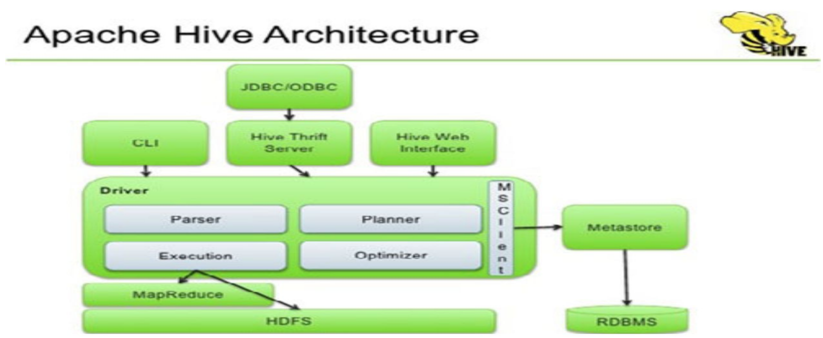


Fig.1 Apache Hive Architecture

=>. CLI, JDBC, ODBC or any other Web GUI form the external interfaces to the Hive framework interface for creating interaction between user and HDFS.
=>. Metastore Thrift API keeps track of what data is stored in which part of the HDFS .It is like a system catalog.
=>. Driver is heart of the Hive architecture responsible for compilation, optimization and execution of HiveQL statements.
=>. Thrift Server is a client side API for executing HiveQL statements.
How a HiveQL query is executed in Apache Hive? : Whenever a user submits a HiveQL query, it is first compiled. The compiled query is then executed by an execution engine like Hadoop MapReduce or Apache Tez. Data in diverse formats like ORC, AVRO, Parquet, or Text reside in HDFS on which the query is to be executed. YARN then allocates desired resources across the Hadoop cluster for execution. The results of the query execution are sent over a JDBC or ODBC connection.

- **Cloudera quick-start and Impala**

Cloudera provides a scalable, flexible, integrated platform that makes it easy to manage rapidly increasing volumes and varieties of data in your enterprise. Cloudera products and solutions enable you to deploy and manage Apache Hadoop and related projects, manipulate and analyze your data, and keep that data secure and protected[4]. Cloudera provides many products and tools. We are using CDH and Apache Impala.
CDH: The most complete, tested and popular distribution of Apache Hadoop and other related open-source projects, including Apache Impala and Cloudera Search. CDH also provides flexibility,security and integration with numerous hardware and software solutions[5].

Apache Impala: A massively parallel processing SQL engine for interactive analytics and business intelligence. Its highly optimized architecture makes it ideally suited for traditional BI-style queries with joins, aggregations, and subqueries. It can query Hadoop data files from a variety of sources, including those produced by MapReduce jobs overloaded into Hive tables[11]. The YARN resource management component lets Impala coexist on clusters running batch workloads concurrently with Impala SQL queries. You can manage
Impala alongside other Hadoop components through the Cloudera Manager user interface and secure its data through the Sentry authorization framework.

How Impala Works with CDH.

The following graphic illustrates how Impala is positioned in the broader Cloudera environment[11]:

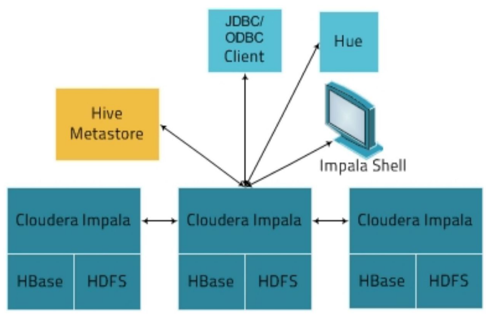


Fig.2 Working of Impala

PRACTICUM

Working of the Project :

1. Datasets

The benchmarking web resource is available at http://stat-computing.org/dataexpo/2009/.Dataset is freely available for download from this website . According to the website, the data consists of flight arrival and departure details for all commercial flights within the USA, from October 1987 to April 2008[2]. However, we are using dataset from 2003 to 2008 for our project, and all csv files are around 100 MB to 125 MB. The dataset used mainly is flights.csv which has a total of 29 variables with dimension of 31 columns and 58,19,079 rows. We have drawn conclusion by mainly focusing on variables arrival delay and departure delay. When there was a delay in the arrival of flights,then there was a delay in departure too for most flights except for few.

2. Data Preparation

Data preparation is the key to big data success. One of the primary barriers to big data success is the lack of a data preparation strategy[10]. Data preparation includes all the steps necessary to acquire, prepare, accurate, and manage the data assets of the organization. Some most important steps are given for the project.

=>**.** Data Pre-processing and Extraction and loading :

Purpose: In general the purpose of the data preprocessing to give some structure to unstructured data, Integrated data with values from external system and preprocess binary content; Writing directly to the data warehouse.

=>**.** Data storage is less of a problem than efficient data retrieval.

=>**.** In this step we download all required datasets in local machine and extracted all datasets.

=>**.** Pig cannot write to a HCatalog table in parquet format but Spark can write to Hive Tables and also write parquet files to hdfs directories

=>**.** Setting up the Data-warehouse :

Managing files on hdfs; Queries for that are given in Fig.3 and Fig.4.

Fig.3 Managing files on Hadoop

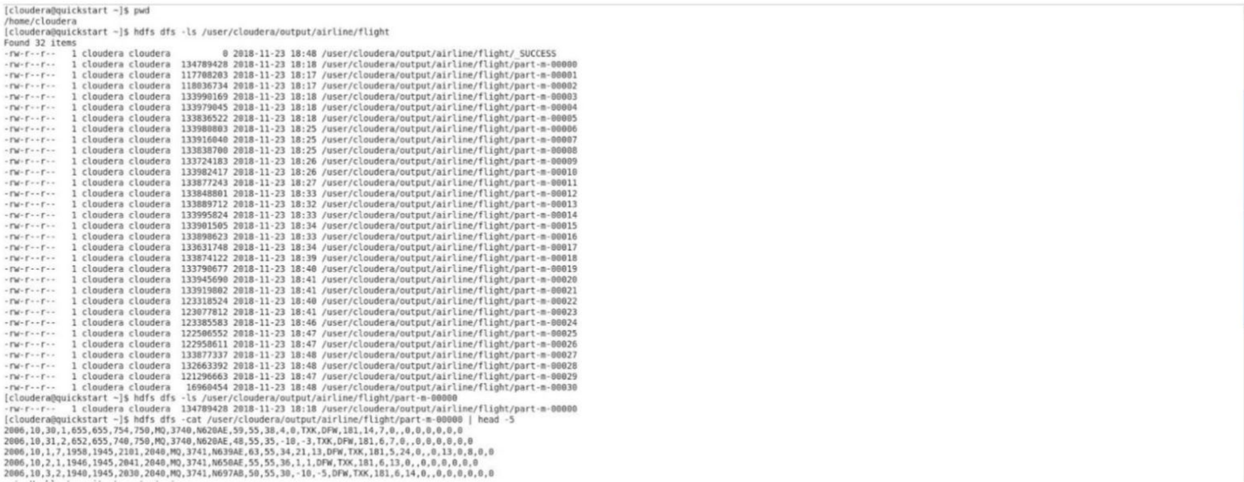


Fig.4 Managing files on Hadoop

Set all data on Quickstart Hue as shown in below figure.

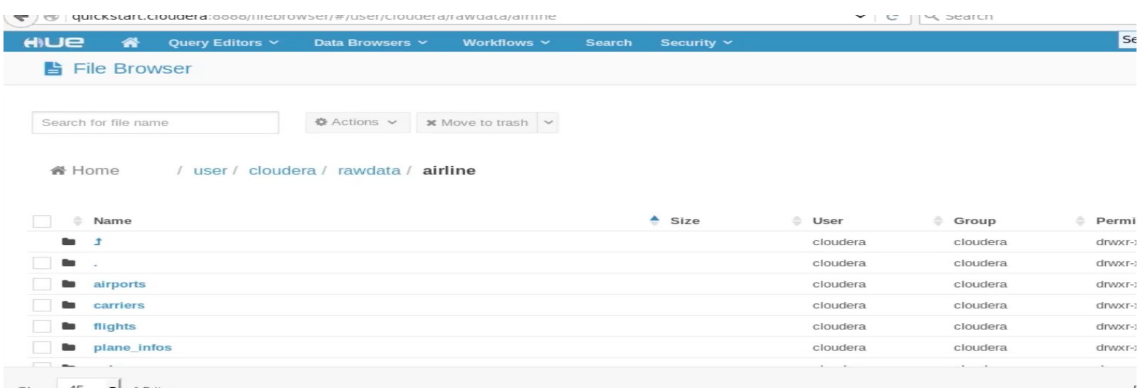


Fig.5 Data setup on Cloudera

=>. Creating Data Table :

To analyze all datasets we create a table format for Datasets of airports, carriers and plane-data and analyze different queries. The queries for same are given in Fig.6 and example output for year 2006 is given in Fig.7.

Fig.6 Creating a Table

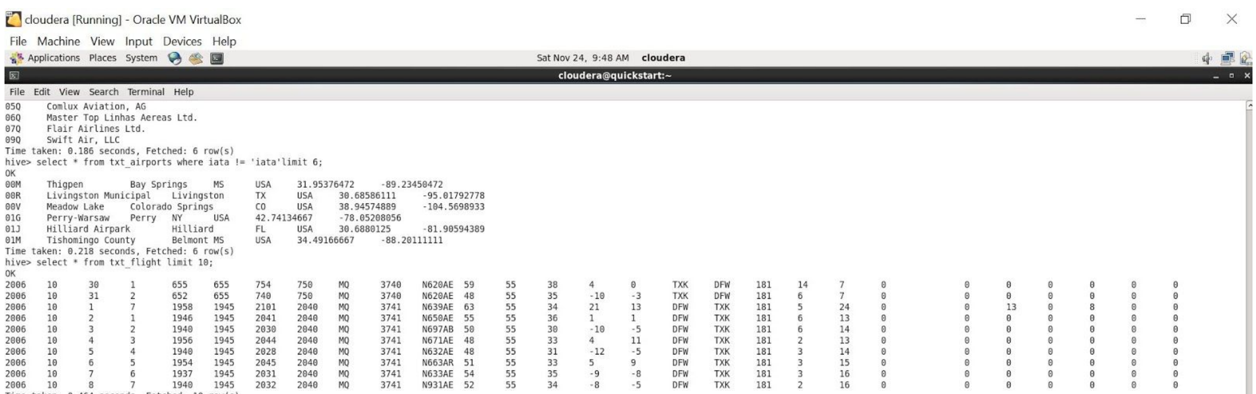


Fig.7 Example output of flight of year 2006

3. Working with Hive vs. Impala or both

Impala server is a SQL query execution engine of Hadoop. Some of the features of Impala architecture are: A massively parallel processing or MPP engine for distributed clustering environment. It is open source and uses data on HDFS[6]. Consists of various daemon processes that run on specific hosts within your Hadoop cluster. The three main components of Impala are Impala daemon, Impala statestore and Impala catalog service, represented by the daemons impalad , statestored and catalog respectively. This architecture is shown in Fig.2. However core component of Impala is the daemon process running on each Impala cluster node.

Functions of Impala are as follows:

=>. The impalad process reads and writes to data files

=>. It logically divides a query into smaller parallel queries and distributes them to different nodes in the Impala cluster. When you submit a query to the Impala daemon running on any node, the node serves as the coordinator node for that query.

=>. Impala transmits intermediate query results back to the central coordinator node. The coordinator constructs the final query output. When you run an experiment using the Impala-shell command, it may connect you to the same Impala daemon process for convenience[12].

When we submit a Hive query,It maps the context to MAPReduce job which in turn jumps into resource manager .Resource manager works on data localization. Resource Manager assigns node manager to do a job. In Impala any resource manager or node manager can be connected to any Impala Daemon, just that all should have one daemon available for it. Impala was written in c++.But it failed as it doesn't know what serdes in any database object can be viewed in hive with external libraries as they are very hive centric libraries which cannot be used in impala, shown in Fig.8.

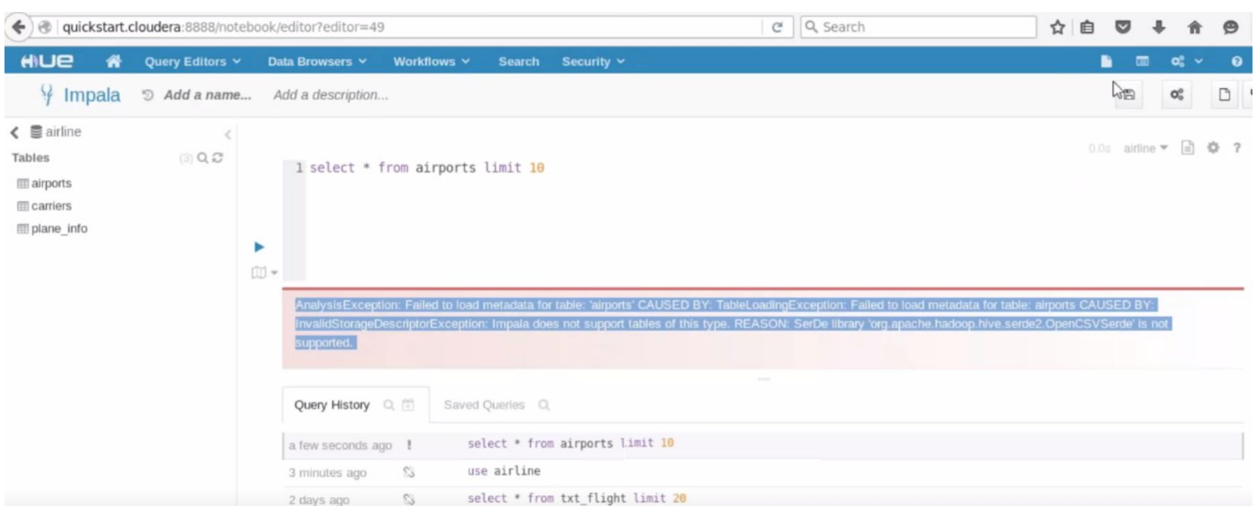


Fig.8 Error we're getting

Custom udf is created in hive.Impala supports Text,RC file Sequence file,Avro and Parquet formats. We are mainly using Parquet files. Impala helps you to create, manage, and query Parquet tables. Parquet is a column-oriented binary file format intended to be highly efficient for the types of large-scale queries that Impala is best at. Parquet is good to perform aggregation operations such as SUM() and AVG() that need to process most or all of the values from a column.There will be tremendous improvement in queries when we use parquet[12]. Time comparison with hive and Impala for same query: we get huge difference between impala and hive. Hive takes around 3 minutes to execute while Impala around 19 seconds.Impala gives a faster response. This time can even be reduced by using compression and query optimization. It is easy to query parquet version then serdes version. Main motto is to make data available, accessible and efficient in a friendly way. Morovere, second pass of parquet is less than the second pass of txt file. Parquet Data Compression for text and parquet files : With Parquet it is 17 times reduction in speed. Every Impala queries with an output are shown in below figures.

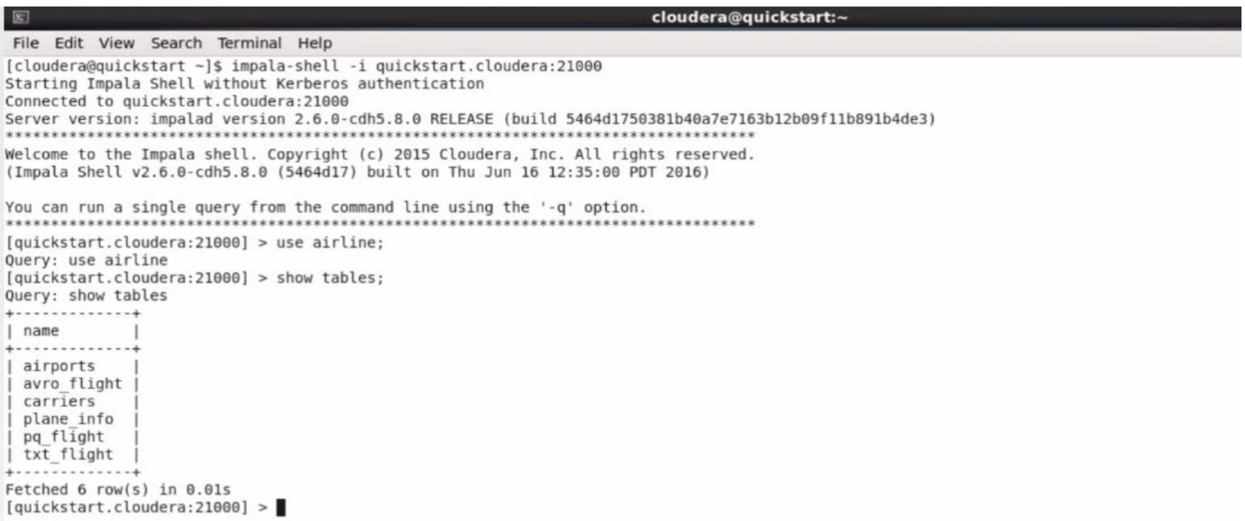


Fig.9 Showing table using Impala

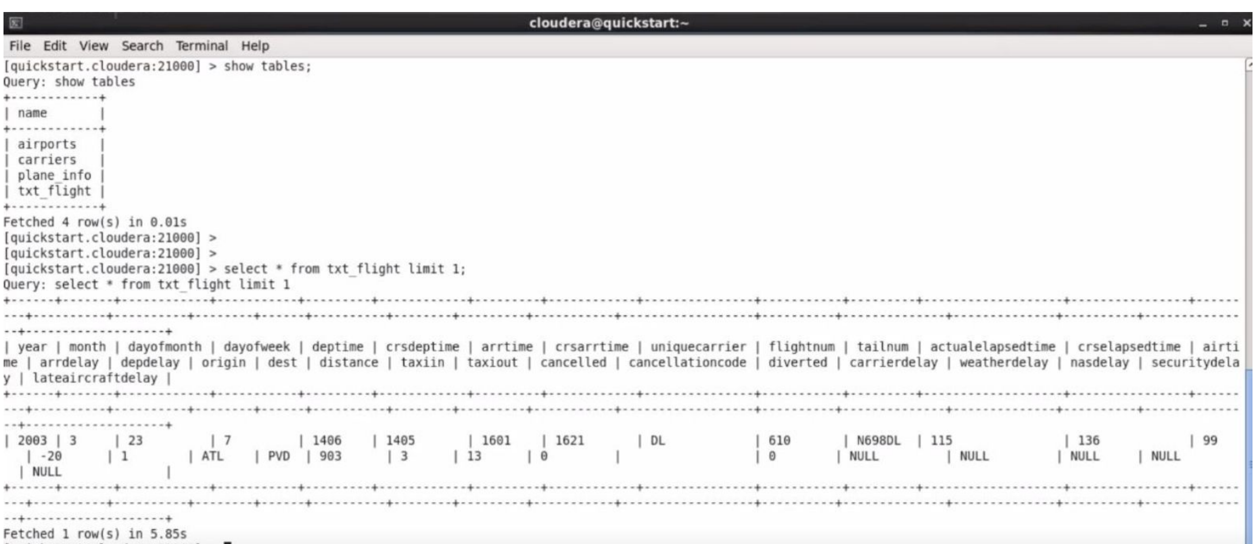


Fig.10 txt_flight output

```
[quickstart.cloudera:21000] > select year, count(1) no_of_flights from txt_flight group by year;
Query: select year, count(1) no_of_flights from txt_flight group by year
+------+---------------+
| year | no_of_flights |
+------+---------------+
| 2008 | 7009728       |
| 2007 | 7453215       |
| 2005 | 7140596       |
| 2003 | 6488540       |
| 2006 | 7141922       |
| 2004 | 7129270       |
+------+---------------+
Fetched 6 row(s) in 18.95s
[quickstart.cloudera:21000] > select year, count(1) no_of_flights from txt_flight group by year;
Query: select year, count(1) no_of_flights from txt_flight group by year
```

Fig.11 Impala taking around 19 s to run

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
| 2004 | 7129270       |
+------+---------------+
Fetched 6 row(s) in 57.16s
[quickstart.cloudera:21000] > select year, count(1) no_of_flights from txt_flight group by year;
Query: select year, count(1) no_of_flights from txt_flight group by year
+------+---------------+
| year | no_of_flights |
+------+---------------+
| 2008 | 7009728       |
| 2007 | 7453215       |
| 2005 | 7140596       |
| 2003 | 6488540       |
| 2006 | 7141922       |
| 2004 | 7129270       |
+------+---------------+
Fetched 6 row(s) in 10.09s
[quickstart.cloudera:21000] > select year, count(1) no_of_flights from txt_flight group by year;
Query: select year, count(1) no_of_flights from txt_flight group by year
+------+---------------+
| year | no_of_flights |
+------+---------------+
| 2008 | 7009728       |
| 2007 | 7453215       |
| 2005 | 7140596       |
| 2003 | 6488540       |
| 2006 | 7141922       |
| 2004 | 7129270       |
+------+---------------+
Fetched 6 row(s) in 8.89s
[quickstart.cloudera:21000] > select year, count(1) no_of_flights from pq_flight group by year;
Query: select year, count(1) no_of_flights from pq_flight group by year
```

Fig.12 Output of group by

Running the query with impala version created example of Parquet files and with Parquet files time taken is sufficiently less.

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
| 2003 | 6488540       |
| 2006 | 7141922       |
| 2004 | 7129270       |
+------+---------------+
Fetched 6 row(s) in 10.09s
[quickstart.cloudera:21000] > select year, count(1) no_of_flights from txt_flight group by year;
Query: select year, count(1) no_of_flights from txt_flight group by year
+------+---------------+
| year | no_of_flights |
+------+---------------+
| 2008 | 7009728       |
| 2007 | 7453215       |
| 2005 | 7140596       |
| 2003 | 6488540       |
| 2006 | 7141922       |
| 2004 | 7129270       |
+------+---------------+
Fetched 6 row(s) in 8.89s
[quickstart.cloudera:21000] > select year, count(1) no_of_flights from pq_flight group by year;
Query: select year, count(1) no_of_flights from pq_flight group by year
+------+---------------+
| year | no_of_flights |
+------+---------------+
| 2008 | 7009728       |
| 2007 | 7453215       |
| 2005 | 7140596       |
| 2003 | 6488540       |
| 2006 | 7141922       |
| 2004 | 7129270       |
+------+---------------+
Fetched 6 row(s) in 6.93s
[quickstart.cloudera:21000] >
```

Fig.13 Comparison of txt and Parquet fight file running time

The second pass of parquet is less than the second pass of the text file. Like it becomes 2.23 s and its 17 times reduction.

Parquet Data Compression for text and parquet files : With Parquet it is 17 times reduction in speed.

```
Fetched 6 row(s) in 6.93s
[quickstart.cloudera:21000] > select year, count(1) no_of_flights from pq_flight group by year;
Query: select year, count(1) no_of_flights from pq_flight group by year
+------+---------------+
| year | no_of_flights |
+------+---------------+
| 2008 | 7009728       |
| 2007 | 7453215       |
| 2005 | 7140596       |
| 2003 | 6488540       |
| 2006 | 7141922       |
| 2004 | 7129270       |
+------+---------------+
Fetched 6 row(s) in 2.23s
[quickstart.cloudera:21000] >
```

Fig.14 Output of second time of the Parquet fight file (too less)

4. Hive/Impala partitioning and clustering : Hive vs Impala Theory

Partitioning Basics :
Purpose : To get efficient spread of the data so that data does not get skewed by randomness.
By default, all the data files for a table are located in a single directory. Partitioning is a technique for physically dividing the data during loading, based on values from one or more columns, to speed up queries that test those columns[12]. Hive is used for partitioning. Two ways of partitioning dynamic and static partitioning both are being implemented on dataset to know the difference[4]. We are using using Static partitioning as we get data year by year. Dynamic partitioning is also shown for the
dataset as with dynamic partitioning we can add any number of partitions with single SQL execution[6].

Static partition : Used when user wants to specify partition name while inserting data in table and distinct values in portioned column are very few and more when data is incrementally loaded portioned on specific time. Loading : one portion is loaded at a time; Good for continuous operation; Not suitable for initial loads[5].
Dynamic partition : Used when you are bulk loading data in table and want to automatically deduce column values and used when distinct values in partitioned column are very high. Loading : Data is distributed between partitions dynamically[6].
Where to do data processing : Hive or Impala? : Normally we can do any of them Hive or Impala. But, we analyse that Impala does not have as much integration with order to Hadoop ecosystem as much as Hive.

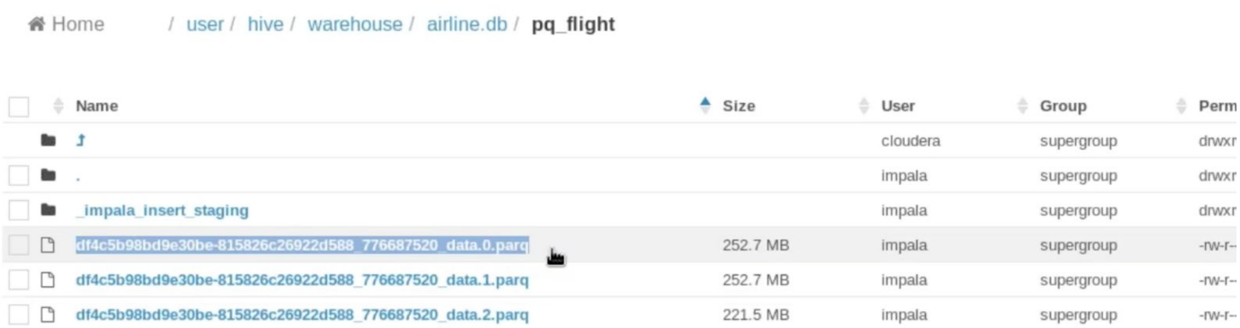
Impala Partitioning :
Analysis : Impala is faster than Apache Hive but that does not mean that it is the one stop SQL solution for all big data problems. Impala is memory intensive and having a low latency and does not run effectively for heavy data operations like joins because it is not possible to push everything into the memory. This is when Hive comes to the rescue[12]. If an application has batch processing kind of needs over big data then organizations must opt for Hive. If they need real time processing of ad-hoc queries on subset of data then Impala is a better choice.Impala does support for Hadoop Distributed File System (HDFS) and Apache HBase.

Following features are different for Hive and Impala :

Fig.15 Impala features name different from Hive

Partitioning Calculation :

For partitioning, need efficient utilisation of name node resources and so in the cluster this selected file have to be replicated three times as shown in the picture and the size of each files


Home / user / hive / warehouse / airline.db / pq_flight

Name	Size	User	Group	Perm
↥		cloudera	supergroup	drwxr
.		impala	supergroup	drwxr
_impala_insert_staging		impala	supergroup	drwxr
df4c5b98bd9e30be-815826c26922d588_776687520_data.0.parq	252.7 MB	impala	supergroup	-rw-r--
df4c5b98bd9e30be-815826c26922d588_776687520_data.1.parq	252.7 MB	impala	supergroup	-rw-r--
df4c5b98bd9e30be-815826c26922d588_776687520_data.2.parq	221.5 MB	impala	supergroup	-rw-r--

Fig.16 Impala files have to replicated

```
[quickstart.cloudera:21000] > use airline;
Query: use airline
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] > select year, count(1) from txt_flight group by year;
Query: select year, count(1) from txt_flight group by year
+------+----------+
| year | count(1) |
+------+----------+
| 2008 | 7009728  |
| 2007 | 7453215  |
| 2005 | 7140596  |
| 2003 | 6488540  |
| 2006 | 7141922  |
| 2004 | 7129270  |
+------+----------+
Fetched 6 row(s) in 105.77s
[quickstart.cloudera:21000] > select year, count(1) from pq_flight group by year;
Query: select year, count(1) from pq_flight group by year
+------+----------+
| year | count(1) |
+------+----------+
| 2008 | 7009728  |
| 2007 | 7453215  |
| 2005 | 7140596  |
| 2003 | 6488540  |
| 2006 | 7141922  |
| 2004 | 7129270  |
+------+----------+
Fetched 6 row(s) in 6.86s
```

Fig.17 Total 7 million flights

are very large. Each of these files are around 252.7 MB and so total it is around 760 MB. Now from the below figure. It can be noticed that every year total flights are 7 million and if we divide by 12, to calculate as per month then at last we are getting a data about 800 MB and it is too large so we won't be able to run query and so we are assuming that assumption for every block for each month is not good and so Hive is better choice for the project. Hive does not utilise more space, as shown in Fig.18. Moreover to reduce these three files to one, command is "set mapred.reduce.tasks=1".

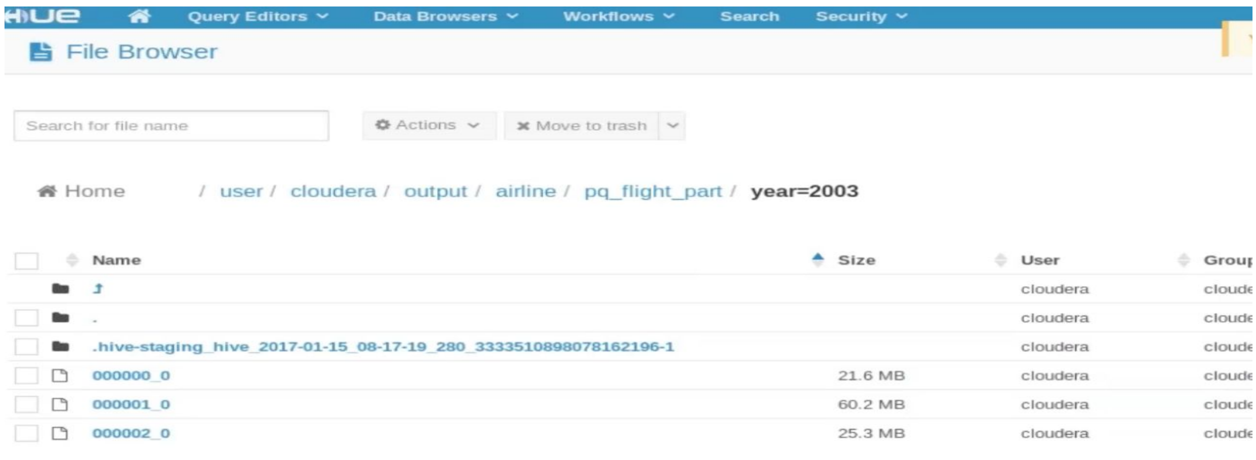


Fig.18 Hive files : less size than Impala

Hive Implementation for partitioning : Output are given in below pictures for the year 2003, 2006, 2007 and 2008.

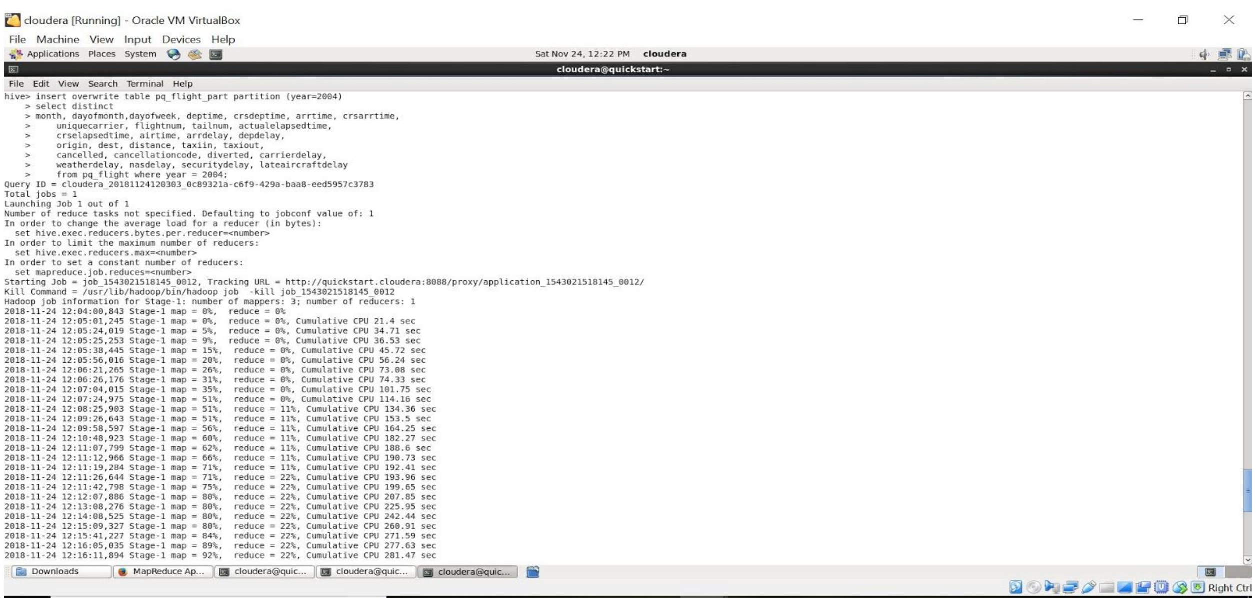


Fig.19 Query to reduce three files to one for data of year 2003

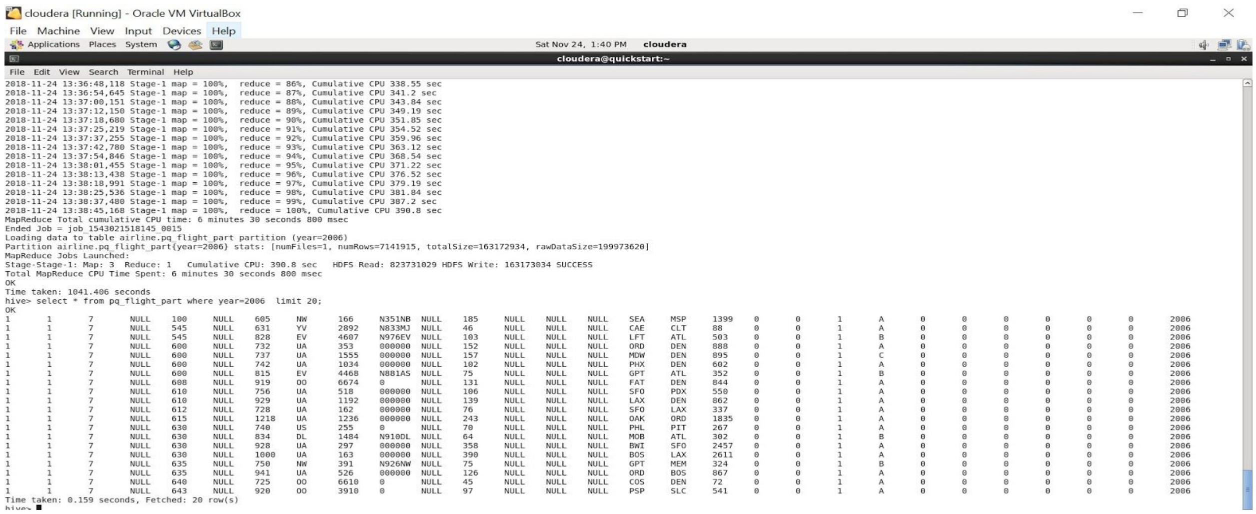


Fig.20 Output for year 2006 - Mapreduce

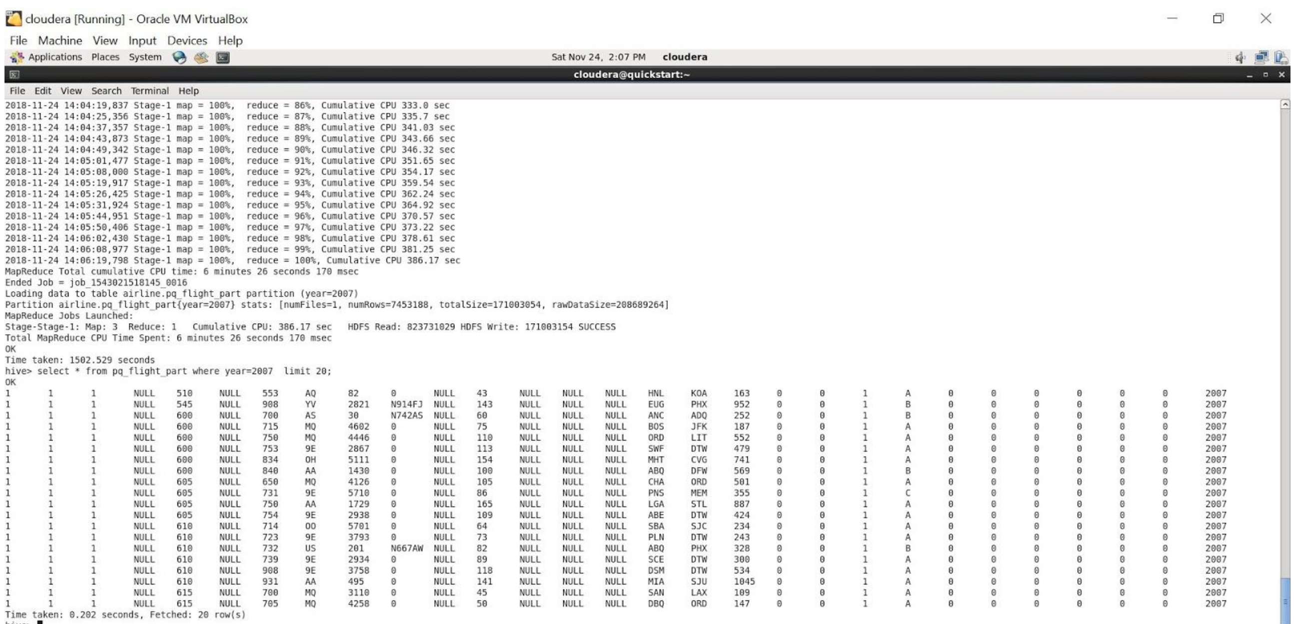


Fig.21 Output for year 2007 - Mapreduce

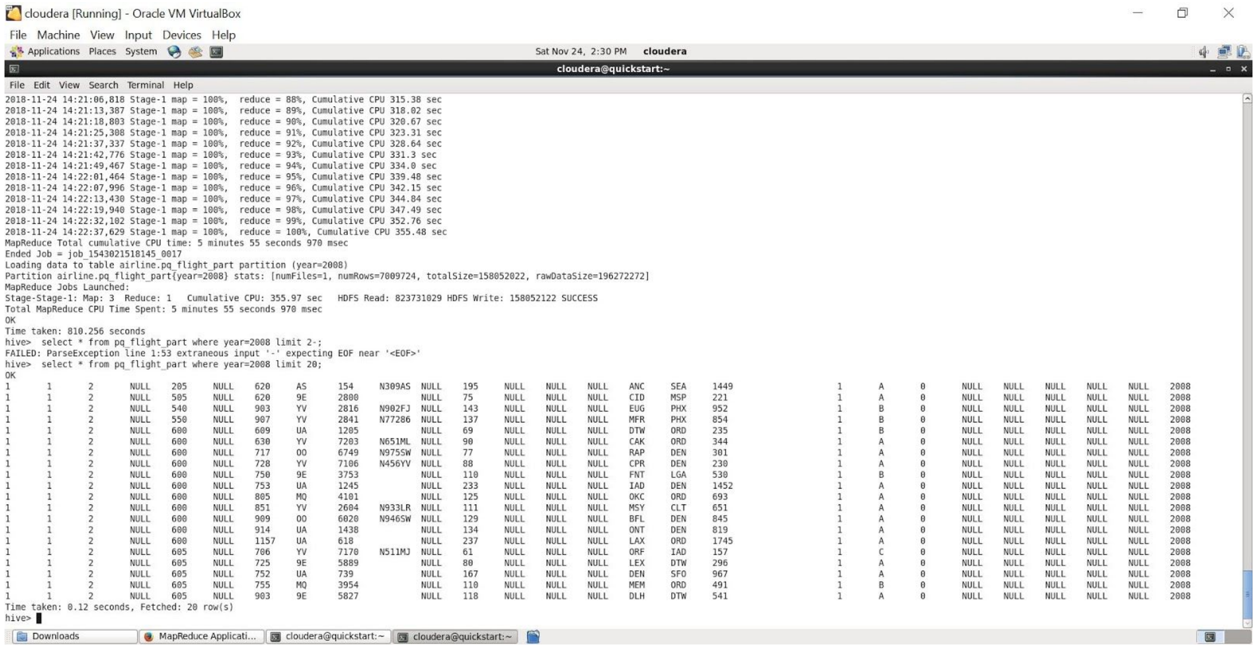


Fig.22 Output for year 2008 - Mapreduce

Clustering:

Hive is used for clustering. Clustering is used to do sampling and for bucket sort/join. Data sampling is done on carrier, origin and time using bucketed tables. Bucketed tables are fantastic in that they allow much more efficient sampling than do non-bucketed tables, and they may allow for time saving operations such as map side joins. Flow of this method is to create first small buckets and put some values in those buckets. Those values are called as samples and from each sample by particular value we can make cluster for the data sets. If we don't do bucketing then, it will be go with random sampling and clustering.

To know a reason of delay flight, in our project can do sample by year and carrier and for that query is given and if we do sampling by country we can get efficient sampling.

> clustered by (uniquecarrier, year) into 6 buckets

Impala does not support sampling and Clustering.

5. Data Compression, tuning and query optimization

=>. Hive :

Data Compression : Hive uses MapReduce and so data is compressed using Hive to save space on disk and network. After the Hive finishes the query execution, the result is submitted to the JobTracker, which resides on YARN. The JobTracker consists of Map/Reduce tasks which runs the mapper and reducer job to store the final result in the HDFS. The Map task deserializes(reading) the data from the HDFS and the Reduce task serializes(writing) the data as the result of the Hive query[8]. Map Reduce is the framework used to process the data which is stored in the HDFS, here java native language is used to writing Mapreduce programs. Hive is a batch processing framework. This component process the data using a language called Hive Query Language(HQL). Hive prevents writing MapReduce programs in Java. Instead one can use SQL like language to do their daily tasks. For HIVE there is no process to communicate Map/Reduce tasks directly. It communicates with Job tracker(Application Master in YARN) only for job processing related things once it got scheduled.

```
cloudera@quickstart:~
File Edit View Search Terminal Help
[quickstart.cloudera:21000] > select year, month, avg(arrdelay) arrdelay, avg(depdelay) depdelay from pq_flight_part where year between 2003 and 2005
                            > group by year, month ;
Query: select year, month, avg(arrdelay) arrdelay, avg(depdelay) depdelay from pq_flight_part where year between 2003 and 2005
group by year, month
+------+-------+--------------------+-------------------+
| year | month | arrdelay           | depdelay          |
+------+-------+--------------------+-------------------+
| 2003 | 4     | -0.3893058035156633 | 2.946050532757227 |
| 2005 | 8     | 8.99297482469548   | 10.14962944487993 |
| 2004 | 11    | 5.882511271462635  | 7.71968202219747  |
| 2003 | 2     | 6.008475602373169  | 6.855551274976312 |
| 2003 | 6     | 3.897317949972709  | 5.443029295462444 |
| 2005 | 10    | 4.585049689344864  | 6.455797209727311 |
| 2005 | 12    | 11.19576616680197  | 12.18423161298594 |
| 2004 | 9     | 1.035358914979271  | 4.088488881365973 |
| 2005 | 2     | 6.184215986645706  | 7.702224785238519 |
| 2004 | 7     | 9.310870723525321  | 10.36401262180606 |
| 2003 | 8     | 6.815474469120979  | 7.597835272644308 |
| 2005 | 4     | 2.594826836171204  | 4.863568215892054 |
| 2004 | 1     | 7.877412838876399  | 8.39583598175064  |
| 2004 | 5     | 7.828580601850501  | 8.552097672220738 |
| 2003 | 12    | 7.644256000883905  | 8.38653116685636  |
| 2004 | 3     | 3.854648946755941  | 5.844603231005256 |
| 2003 | 10    | 0.7331523220731888 | 2.977092123030483 |
| 2005 | 6     | 9.813724375100358  | 10.82578565242567 |
| 2003 | 7     | 6.511102255026132  | 7.578567109652045 |
| 2005 | 11    | 5.188486933214206  | 7.082841556180337 |
| 2004 | 8     | 7.133166699906167  | 8.344086898640006 |
| 2003 | 1     | 2.184339894754486  | 4.236297465095408 |
| 2003 | 5     | 1.606879216527369  | 3.831899506864425 |
| 2005 | 9     | 2.645287014708892  | 4.868672041740045 |
| 2004 | 12    | 10.89224066121997  | 12.26127026069023 |
| 2003 | 3     | 2.499213800667951  | 4.551245207243519 |
| 2004 | 10    | 4.051113899594674  | 5.615549721385218 |
```

Fig.23 Output of flight delay from year 2003 to 2005

=>. Impala :

COMPUTE STATS to improve query performance while performing joins.

File Formats : Statistic is computed on various impala file formats to compare performance. We compute on six files, we create a portion and add data to the partition. All queries and output are shown in pictures below where when you compute a statistics we can know about the numbers of columns and rows. Moreover, by HDFS caching we can also improve performance[12].

Output for Impala execution are given in below pictures including compute stats and describe.

```
alter table pq_flight_part add partition(year=2003)
alter table pq_flight_part add partition(year=2004)
alter table pq_flight_part add partition(year=2005)
alter table pq_flight_part add partition(year=2006)
alter table pq_flight_part add partition(year=2007)
alter table pq_flight_part add partition(year=2008)

set mapred.reduce.tasks=1
insert overwrite table pq_flight_part partition (year=2004)
select distinct
month, dayofmonth,dayofweek, deptime, crsdeptime, arrtime, crsarrtime,
    uniquecarrier, flightnum, tailnum, actualelapsedtime,
    crselapsedtime, airtime, arrdelay, depdelay,
    origin, dest, distance, taxiin, taxiout,
    cancelled, cancellationcode, diverted, carrierdelay,
    weatherdelay, nasdelay, securitydelay, lateaircraftdelay
    from pq_flight where year = 2004


insert overwrite table pq_flight_part partition (year)
select
month, dayofmonth,dayofweek, deptime, crsdeptime, arrtime, crsarrtime,
    uniquecarrier, flightnum, tailnum, actualelapsedtime,
    crselapsedtime, airtime, arrdelay, depdelay,
    origin, dest, distance, taxiin, taxiout,
    cancelled, cancellationcode, diverted, carrierdelay,
    weatherdelay, nasdelay, securitydelay, lateaircraftdelay, year
    from pq_flight where year != 2003
```

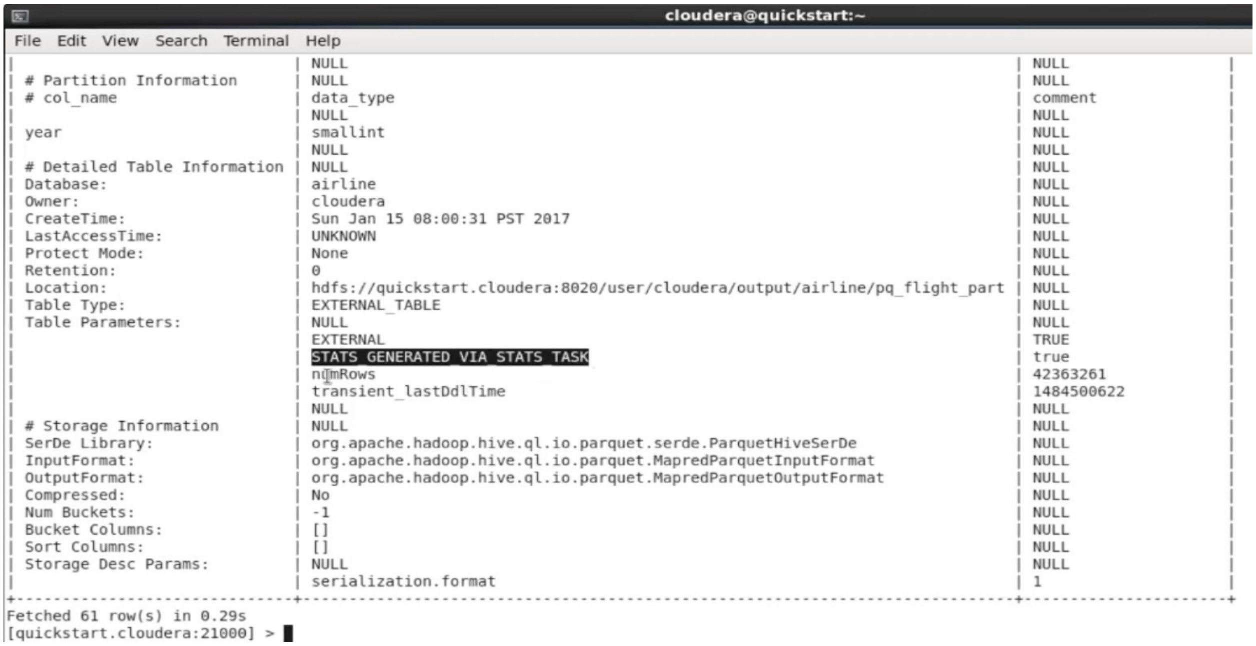
Fig.24 Creating and Adding all six files

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
| 2003 | 11    | 4.716544401284412   | 5.758203332027559 |
| 2005 | 7     | 13.85383111122728   | 14.29684095683184 |
| 2004 | 2     | 5.953804922643387   | 7.049601851596726 |
| 2004 | 6     | 11.3532444130866    | 11.48183952992779 |
| 2005 | 3     | 7.621900809412867   | 9.150015394215398 |
| 2003 | 9     | 0.9105020298429279  | 2.761916299855968 |
| 2005 | 5     | 3.039095726310181   | 5.274917272242066 |
+------+-------+---------------------+-------------------+
Fetched 36 row(s) in 1.69s
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] > compute stats pq_flight_part;
Query: compute stats pq_flight_part
WARNINGS:
Failed to open HDFS file hdfs://quickstart.cloudera:8020/user/cloudera/output/airline/pq_flight_part/year=2004/000001_0
Error(2): No such file or directory


[quickstart.cloudera:21000] > invalidate metadata;
Query: invalidate metadata

Fetched 0 row(s) in 4.03s
[quickstart.cloudera:21000] > compute stats pq_flight_part;
Query: compute stats pq_flight_part
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 6 partition(s) and 28 column(s). |
+-----------------------------------------+
Fetched 1 row(s) in 33.78s
[quickstart.cloudera:21000] > describe formatted pq_flight_part;
```

Fig.25 Execution Part



```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
|                              | NULL                                                                | NULL      |
| # Partition Information      | NULL                                                                | NULL      |
| # col_name                   | data_type                                                           | comment   |
|                              | NULL                                                                | NULL      |
| year                         | smallint                                                            | NULL      |
|                              | NULL                                                                | NULL      |
| # Detailed Table Information | NULL                                                                | NULL      |
| Database:                    | airline                                                             | NULL      |
| Owner:                       | cloudera                                                            | NULL      |
| CreateTime:                  | Sun Jan 15 08:00:31 PST 2017                                        | NULL      |
| LastAccessTime:              | UNKNOWN                                                             | NULL      |
| Protect Mode:                | None                                                                | NULL      |
| Retention:                   | 0                                                                   | NULL      |
| Location:                    | hdfs://quickstart.cloudera:8020/user/cloudera/output/airline/pq_flight_part | NULL |
| Table Type:                  | EXTERNAL_TABLE                                                      | NULL      |
| Table Parameters:            | NULL                                                                | NULL      |
|                              | EXTERNAL                                                            | TRUE      |
|                              | STATS_GENERATED_VIA_STATS_TASK                                      | true      |
|                              | numRows                                                             | 42363261  |
|                              | transient_lastDdlTime                                               | 1484500622 |
|                              | NULL                                                                | NULL      |
| # Storage Information        | NULL                                                                | NULL      |
| SerDe Library:               | org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe         | NULL      |
| InputFormat:                 | org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat       | NULL      |
| OutputFormat:                | org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat      | NULL      |
| Compressed:                  | No                                                                  | NULL      |
| Num Buckets:                 | -1                                                                  | NULL      |
| Bucket Columns:              | []                                                                  | NULL      |
| Sort Columns:                | []                                                                  | NULL      |
| Storage Desc Params:         | NULL                                                                | NULL      |
|                              | serialization.format                                                | 1         |
+------------------------------+---------------------------------------------------------------------+-----------+
Fetched 61 row(s) in 0.29s
[quickstart.cloudera:21000] >
```

Fig.26 Impala output describe all - stats part (selected one)

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
                        |
+-------+---------+-------+---------+-------------+------------------+---------+-----------------+----------------------
----------------------+
Fetched 7 row(s) in 0.06s
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] > show table stats pq_flight;
Query: show table stats pq_flight
+-------+--------+----------+--------------+-------------------+---------+-------------------+----------------------
+
| #Rows | #Files | Size     | Bytes Cached | Cache Replication | Format  | Incremental stats | Location
|
+-------+--------+----------+--------------+-------------------+---------+-------------------+----------------------
+
| -1    | 3      | 726.85MB | NOT CACHED   | NOT CACHED        | PARQUET | false             | hdfs://quickstart.cloudera:8020/user/hive/warehouse/airline.
|
+-------+--------+----------+--------------+-------------------+---------+-------------------+----------------------
+
Fetched 1 row(s) in 4.80s
[quickstart.cloudera:21000] > compute stats pq_flights;
Query: compute stats pq_flights
ERROR: AnalysisException: Table does not exist: airline.pq_flights

[quickstart.cloudera:21000] > compute stats pq_flight;
Query: compute stats pq_flight
+-----------------------------------------+
| summary                                 |
+-----------------------------------------+
| Updated 1 partition(s) and 29 column(s). |
+-----------------------------------------+
Fetched 1 row(s) in 27.13s
[quickstart.cloudera:21000] >
```

Fig.27 Show table stats and compute stats including size of file (selected one)

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
  |
+----------+--------+---------+-------------+------------------+--------+-------------------+------------------------------------------
--+
| 42363271 | 3      | 726.85MB | NOT CACHED   | NOT CACHED        | PARQUET | false             | hdfs://quickstart.cloudera:8020/user/hive/wa
it |
+----------+--------+---------+-------------+------------------+--------+-------------------+------------------------------------------
--+
Fetched 1 row(s) in 0.01s
[quickstart.cloudera:21000] > show table stats pq_flight_part;
Query: show table stats pq_flight_part
+-------+----------+--------+----------+--------------+-------------------+---------+-------------------+----------------------------
-----------------------+
| year  | #Rows    | #Files | Size     | Bytes Cached | Cache Replication | Format  | Incremental stats | Location
                       |
+-------+----------+--------+----------+--------------+-------------------+---------+-------------------+----------------------------
-----------------------+
| 2003  | 6488540  | 3      | 107.12MB | NOT CACHED   | NOT CACHED        | PARQUET | false             | hdfs://quickstart.cloudera:8020/user
q_flight_part/year=2003 |
| 2004  | 7129260  | 1      | 155.95MB | NOT CACHED   | NOT CACHED        | PARQUET | false             | hdfs://quickstart.cloudera:8020/user
q_flight_part/year=2004 |
| 2005  | 7140596  | 3      | 133.18MB | NOT CACHED   | NOT CACHED        | PARQUET | false             | hdfs://quickstart.cloudera:8020/user
q_flight_part/year=2005 |
| 2006  | 7141922  | 3      | 136.93MB | NOT CACHED   | NOT CACHED        | PARQUET | false             | hdfs://quickstart.cloudera:8020/user
q_flight_part/year=2006 |
| 2007  | 7453215  | 3      | 143.88MB | NOT CACHED   | NOT CACHED        | PARQUET | false             | hdfs://quickstart.cloudera:8020/user
q_flight_part/year=2007 |
| 2008  | 7009728  | 3      | 131.34MB | NOT CACHED   | NOT CACHED        | PARQUET | false             | hdfs://quickstart.cloudera:8020/user
q_flight_part/year=2008 |
| Total | 42363261 | 16     | 808.40MB | 0B           |                   |         |                   |
                       |
+-------+----------+--------+----------+--------------+-------------------+---------+-------------------+----------------------------
-----------------------+
Fetched 7 row(s) in 0.01s
[quickstart.cloudera:21000] >
```

Fig.28 Including number of rows

6. Using database views to represent data

Purpose : Purpose of viewing data is to project and to hide complexities and security. Security because of computing some airplane accident. All command for viewing a data and outputs for that are given in below figures.

```
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] >
[quickstart.cloudera:21000] > show tables;
Query: show tables
+----------------+
| name           |
+----------------+
| airports       |
| carriers       |
| plane_info     |
| pq_airports    |
| pq_carriers    |
| pq_flight      |
| pq_flight_part |
| pq_plane_info  |
| txt_flight     |
+----------------+
Fetched 9 row(s) in 0.01s
[quickstart.cloudera:21000] >
```

Fig.29 Showing Tables

```
+----------+---------+-----------+-----------+-----------+-----------+
Fetched 1 row(s) in 5.22s
[quickstart.cloudera:21000] > compute stats pq_airports
                            > ;
Query: compute stats pq_airports
comput+---------------------------------------+
| summary                               |
+---------------------------------------+
| Updated 1 partition(s) and 7 column(s). |
+---------------------------------------+
Fetched 1 row(s) in 1.32s
[quickstart.cloudera:21000] > compute stats pq_carriers;
Query: compute stats pq_carriers
+---------------------------------------+
| summary                               |
+---------------------------------------+
| Updated 1 partition(s) and 2 column(s). |
+---------------------------------------+
Fetched 1 row(s) in 5.19s
[quickstart.cloudera:21000] > compute stats pq_plane_info;
Query: compute stats pq_plane_info
```

Fig.30 Computing stats of tables

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
+---------------------------------------+
| Updated 1 partition(s) and 2 column(s). |
+---------------------------------------+
etched 1 row(s) in 5.19s
quickstart.cloudera:21000] > compute stats pq_plane_info;
uery: compute stats pq_plane_info
+---------------------------------------+
| summary                               |
+---------------------------------------+
| Updated 1 partition(s) and 8 column(s). |
+---------------------------------------+
etched 1 row(s) in 5.58s
quickstart.cloudera:21000] >
quickstart.cloudera:21000] >
quickstart.cloudera:21000] >
quickstart.cloudera:21000] >
quickstart.cloudera:21000] >
quickstart.cloudera:21000] > desc pq_airports;
uery: describe pq_airports
+---------+--------+---------+
| name    | type   | comment |
+---------+--------+---------+
| iata    | string |         |
| airport | string |         |
| city    | string |         |
| state   | string |         |
| country | string |         |
| geolat  | float  |         |
| geolong | float  |         |
+---------+--------+---------+
etched 7 row(s) in 0.03s
quickstart.cloudera:21000] >
quickstart.cloudera:21000] >
quickstart.cloudera:21000] > desc pq_carriers
```

Fig.31 Describing Table

From above table we can write any query to view data differently. Query example is shown in Fig.32, which will add more table and that's shown in Fig.33. After that need to compute all tables again and this query to view is for security purpose that you can identify airplane accident immediately.

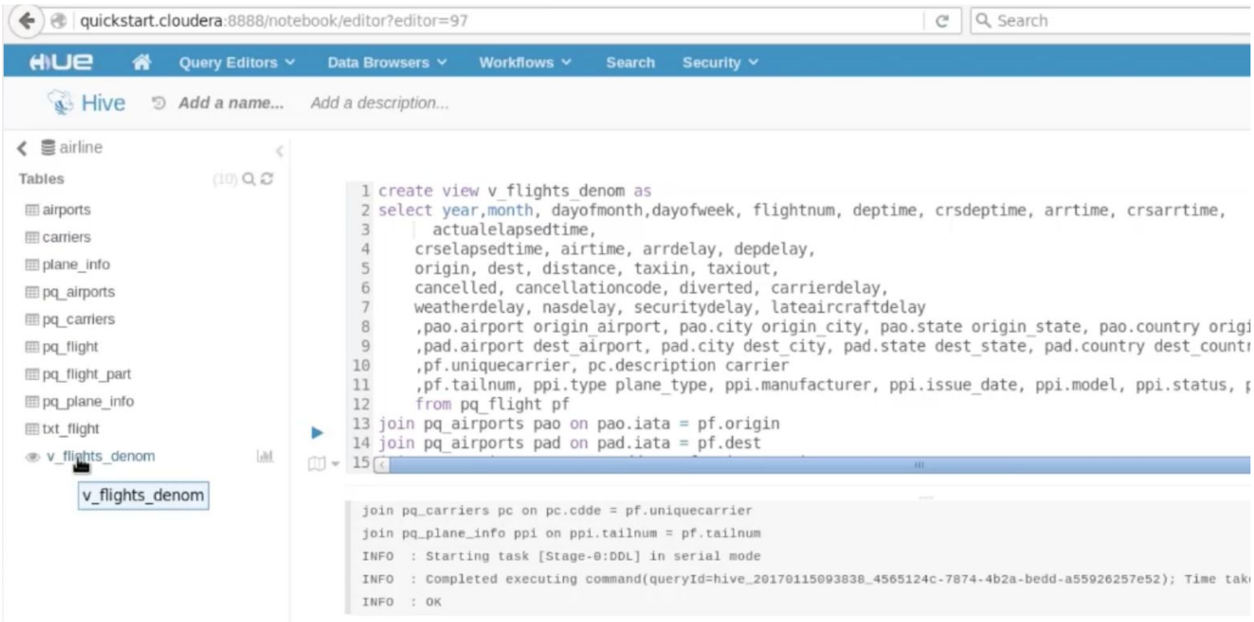


Fig.32 Query to join tables differently

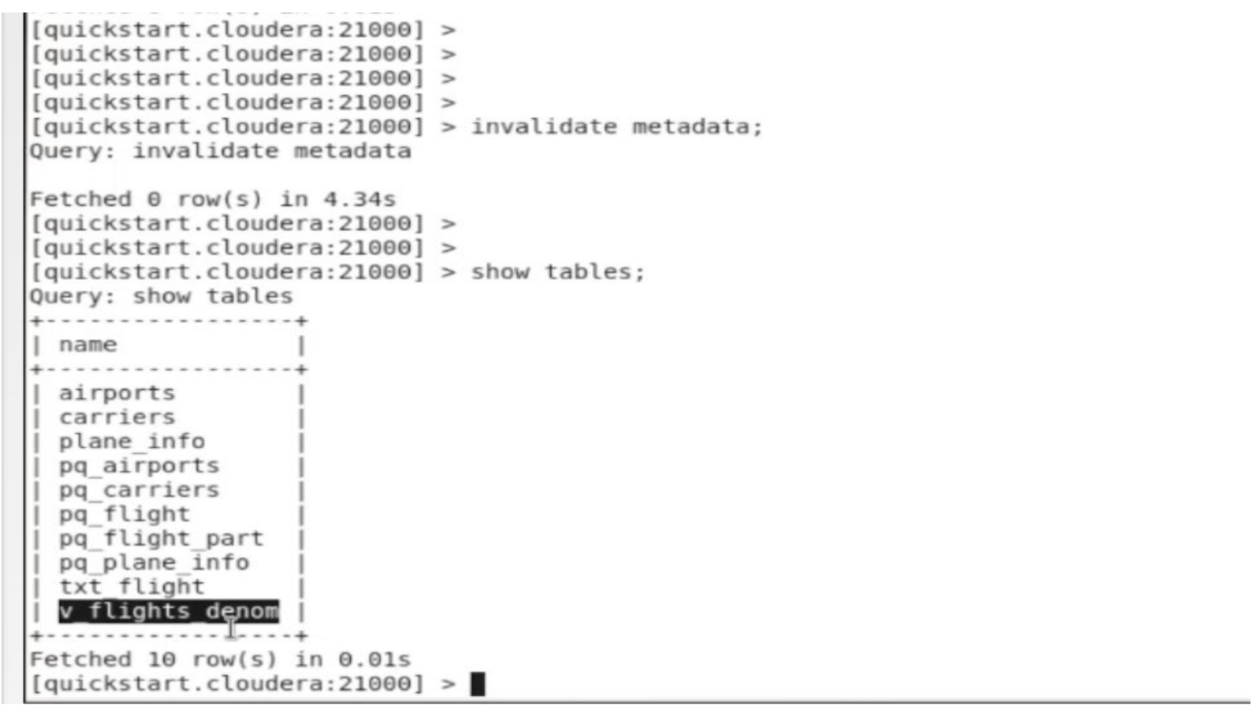


Fig.33 One more table add afte

7. Visualizing data using Microsoft Excel

In this project we visualized details about airplane delay in Microsoft Excel graph which is shown below. And from this visualization we analyzed that normally for every arrival delay, departure was also delayed. So in our graph the number of flights for arrival and departure

delay were same for most of the points according to data. As per dataset considered ,we also observed that it was not the case for one point wherein the number of flights for both delay were different and it is for some number of flights where arrival delay happened but departure was not delayed and vice versa which can be seen in Fig.35.

Year	Arr Delay	Dep Delay
2003	-0.38305	2.460505
2005	8.99297	10.149629
2004	5.8825112	7.7196820
2003	6.008475	6.8555555
2003	3.8973179	5.4430292
2005	4.5858496	6.4555797
2005	11.195766	12.184231
2004	1.0353589	4.0884888
2005	6.1842159	7.7022247
2004	9.3108707	10.364012
2003	6.8154744	7.5978352
2005	2.5948268	4.8635682
2004	7.8774128	8.3958359
2004	7.8285806	80552097(
2003	7.644256	8.3865311
2004	3.8546489	5.8446032
2003	0.7331523	2.9770921
2005	9.8137243	10.825785
2003	6.5111022	7.5785671
2005	5.1884869	7.8828415
2004	7.1331666	8.3448868
2003	2.1843398	4.2362974
2003	1.6068792	3.8318995
2005	2.6452870	4.8686720
2004	18.892240	12.261270
2003	2.4992138	4.5512452
2004	4.0511113:	5.6155497

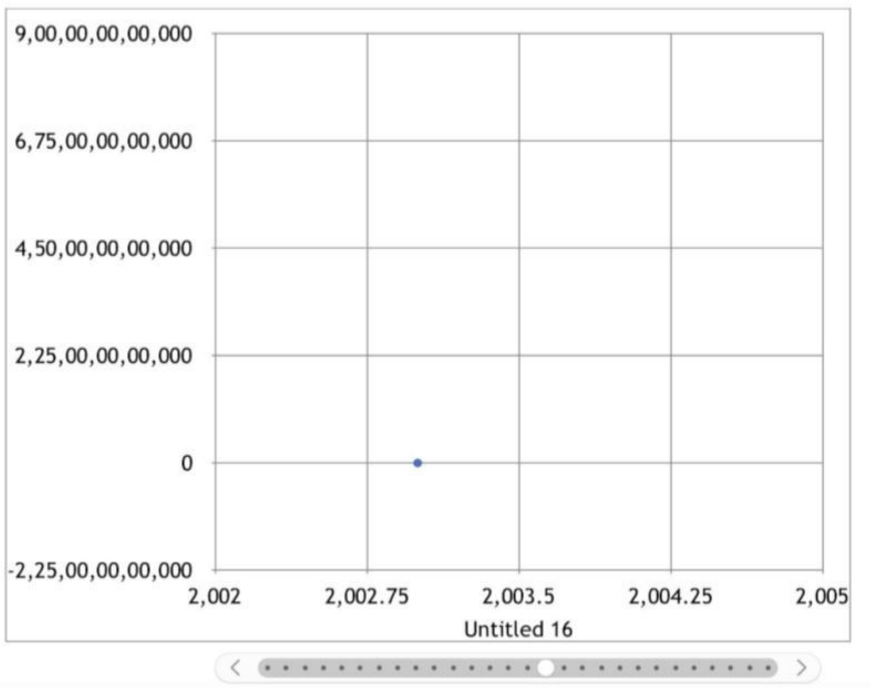


Fig.34 Arrival and Departure delays most of the time at same point

Year	Arr Delay	Dep Delay
2003	-0.38305	2.460505
2005	8.99297	10.149629
2004	5.8825112	7.7196820
2003	6.008475	6.8555555
2003	3.8973179	5.4430292
2005	4.5858496	6.4555797
2005	11.195766	12.184231
2004	1.0353589	4.0884888
2005	6.1842159	7.7022247
2004	9.3108707	10.364012
2003	6.8154744	7.5978352
2005	2.5948268	4.8635682
2004	7.8774128	8.3958359
2004	7.8285806	80552097(
2003	7.644256	8.3865311
2004	3.8546489	5.8446032
2003	0.7331523	2.9770921
2005	9.8137243	10.825785
2003	6.5111022	7.5785671
2005	5.1884869	7.8828415
2004	7.1331666	8.3448868
2003	2.1843398	4.2362974
2003	1.6068792	3.8318995
2005	2.6452870	4.8686720
2004	18.892240	12.261270
2003	2.4992138	4.5512452
2004	4.0511113:	5.6155497

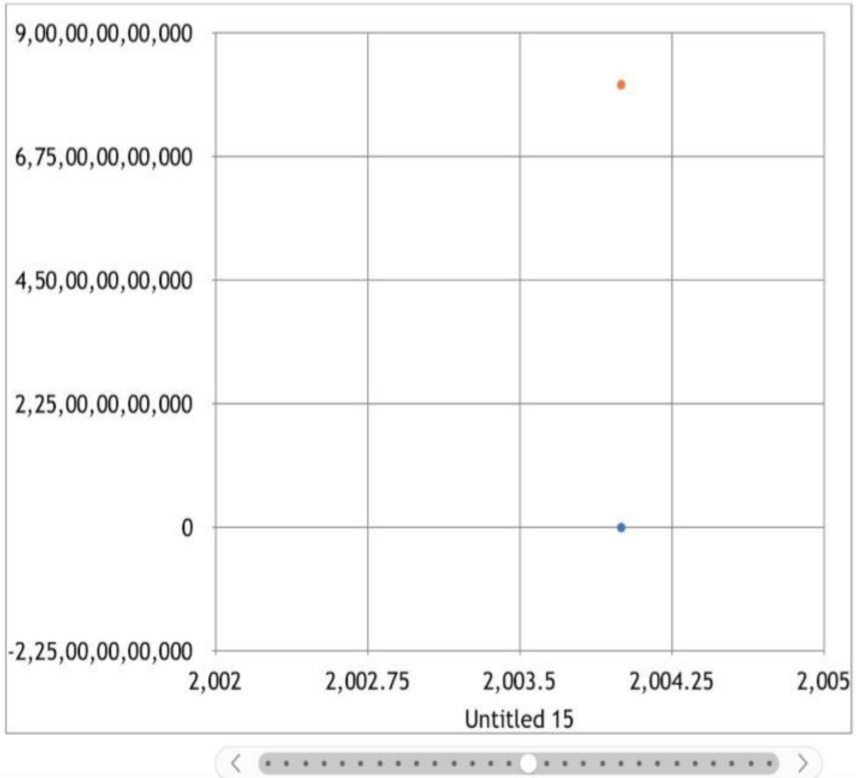


Fig.35 Arrival and Departure delays are for different num of flight

CONCLUSION

This practicum is all about the analysis of airline dataset. The main purpose of the practicum was - after preprocessing the data sets we summarized all tables and also viewed data based on various problem statements differently. Moreover, we compared queries and output from Hive and Impala and analyzed delays of flights. In conclusion, we analyzed that Impala does not have as much integration with order to Hadoop ecosystem as much as Hive and we assume that Impala does not support HDFS and Hbase. But, we have seen that Impala's - compute stats feature was used to improve performance; an execution time for Impala is very less than Hive execution time. From the visualization part,we finally concluded that for most of the time whenever arrival delays happened , departure was delayed too except some number of flights, when it was not the case as per the dataset considered.

REFERENCES

[1]https://www.cloudera.com/

[2]http://stat-computing.org/dataexpo/2009/

[3]https://www.ijcsmc.com/docs/papers/June2017/V6I6201764.pdf

[4]https://www.cloudera.com/documentation/enterprise/latest/PDF/cloudera-quickstart.pdf

[5]https://www.cloudera.com/documentation/enterprise/5-9-x/PDF/cloudera-introduction.pdf

[6]https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#Language_Manual DDL-Dynamic Partitions

[7]https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL+BucketedTables

[8]https://www.cloudera.com/documentation/enterprise/5-9-x/topics/introduction_compression.html

[9]https://www.cloudera.com/documentation/enterprise/5-9-x/topics/impala_compute_stats.html

[10]http://hadoopilluminated.com/hadoop_illuminated/hadoop-illuminated.pdf

[11]http://cidrdb.org/cidr2015/Papers/CIDR15_Paper28.pdf

[12]https://www.cloudera.com/documentation/enterprise/5-5-x/PDF/cloudera-impala.pdf