Data Warehousing with Hive

By : Amit Singh Chowdhery

Agenda:

Problem statement
Existing Solution
What are data warehouses and what is OLAP?
How does Apache hive act as a data warehouse?
What makes hive as Hive? (Architecture)
How to interact with Hive?
What are different HQL commands that are used commonly?
What are the optimization techniques in Hive ?
Use cases of hive/ where to use hive/why and why not

Objective : How hive emerged as a data warehouse engine in Airbnb?

Problem statement:

In 2014 **Airbnb** was using RDBMS(**Oracle and MySQL databases)** to keep all day to day activities inside a single cluster and the same setup was used by Sales teams to do

- **Analytics using AdHoc Queries** : Team fires query to extract information from billions of rows in order to identify customer preference, bounces from websites

- **Dashboarding and Reporting** : Sales team needs to fetch reports during business hours in order to give more insights about sales made on same day

Their existing setup(**RDBMS cluster**) started causing hiccups as data kept on growing on a day to day basis. Major Issues they faced :

- **Reporting jobs became tardy** : **Airbnb** sales and marketing needs reports to make decisions that will be pushed the next day and job started giving latency and hence decision making started falling back

- **Business critical jobs** were getting impacted : During business hours, if any reports needed to be extracted, this disrupted normal functioning of Business.

- **AdHoC queries started giving famous RTO(Request Time Out):** Internal DE teams were strictly prohibited to run any kind of ad hoc queries as they might lead to overall slowdown of normal workflow plus RTO started becoming the norm.

- **ML models** to be deployed : **Airbnb** started looking to **implement** ML pipelines in their core workflows and we know how to train a model how much data it requires. Again putting pressure on existing databases.

With all these major factors impacting their **Business As Usual, Airbnb** in 2014 laid out approach to separate all data infrastructure into two separate mirrored clusters:

- One to run all of the **business critical jobs** like creating booking,finalizing payments – things that have to be run and done on time –
- Another one for **ad hoc queries and reporting**

Solution:

Eventually **Airbnb** started multi cluster approach , where for

- **Cluster 1** : They opt for same RDBMS which will ingest and process day to day data
- **Cluster 2**: They created a data warehouse using **Hive** and also uses the Presto SQL query

Note : Presto is out of scope here

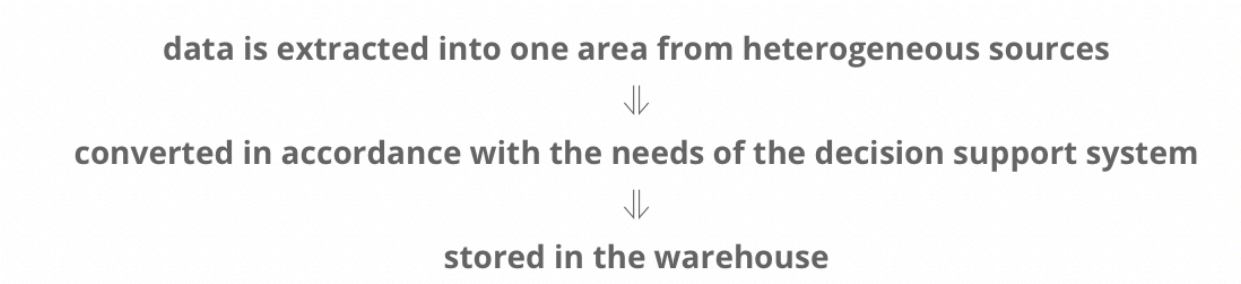
Well, for cluster 1 we don't have to think much but what about cluster 2 and in that what is Hive there?

So to get insight into Hive we have to revise the data warehouse concept first?

Cluster 2 in Airbnb was created for reporting and analytics, where processed(**Golden**) data is kept in RDBMS format so what is this Data warehouse?

A **data warehouse** is a central repository of information that can be analyzed to make more informed decisions. Data gets ingested into data warehouse from :

- **Transactional systems** : order entry, airline reservations, payroll, employee records, manufacturing, and shipping.
- **Relational databases** : Microsoft SQL Server, Oracle Database, MySQL and IBM DB2
- **Spreadsheets**: Data present in any flat files.



Usually main reason for keeping data in data warehouse are:

- **Speeding up response times :** source systems are fully optimized in order to process many small transactions, such as orders, in a short time. **Airbnb** adhoc queries were now executing in hrs after using DWh(Hive)

- **Faster and more flexible reporting :** structure of data warehouses enables end users to report in a flexible manner and to quickly perform interactive analysis. Airbnb DA/DE were able fetch reports writing similar SQL queries.
- **Improved data quality :** Data warehouse greatly improves the data quality, by correcting the data while loading or by tackling the problem at its source.

  Analytical cluster helped **Airbnb DE** to create pipelines that can easily be scheduled and orchestrated as they don't have to focus on data quality

- **Unburdening operational systems :** By transferring data to a separate computer in order to analyze it, the operational system is unburdened.

So How does Datewarehouse relate to OLAP?

OLAP stands for **online analytical Processing**. These are the systems powering Airbnb reporting and analytical verticals.

OLAP significantly differs from OLTP as :

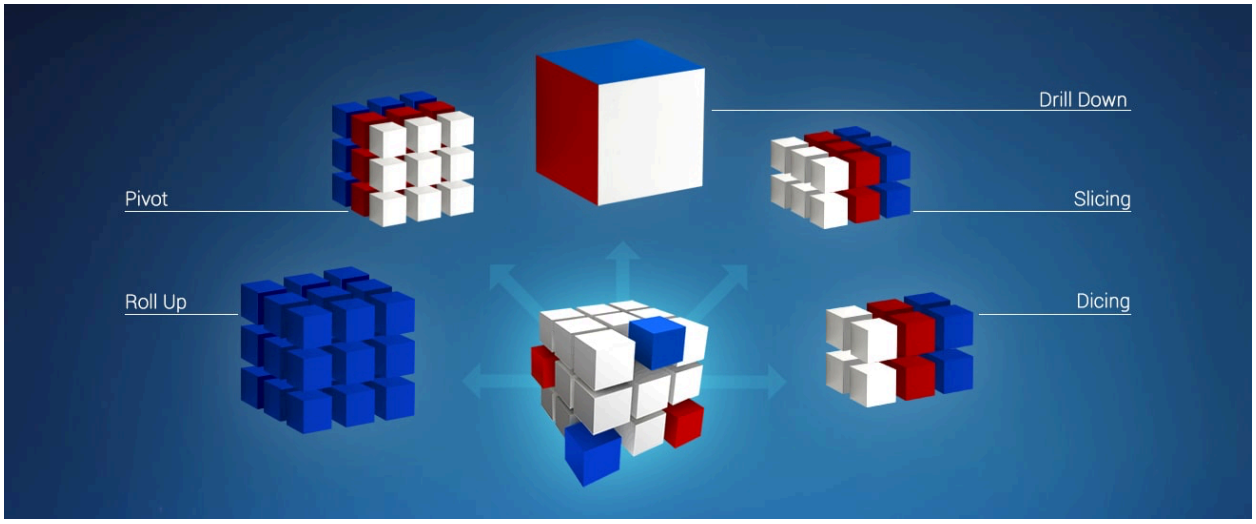
- OLAP is mostly for analytical and OLTP is for transactional
- OLAP works for historical data and OLTP is for current data
- OLAP works on TB/PB/ZB and OLTP is for MB,GB

So the question is what is the relation between data warehouse and OLAP?

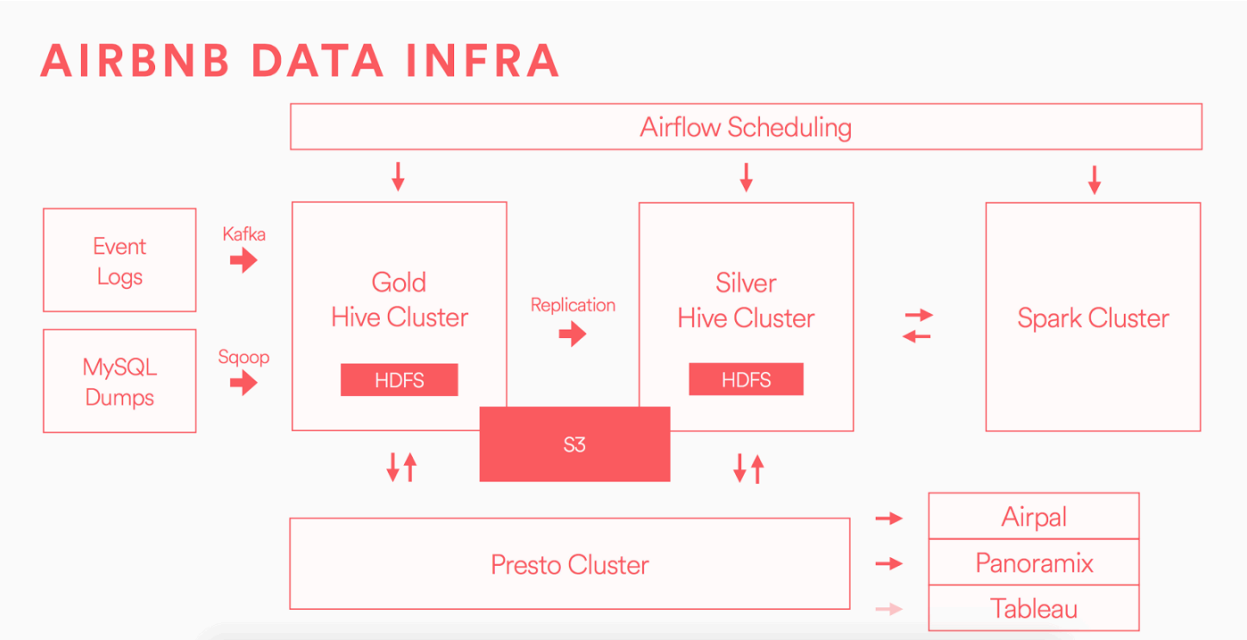
- **OLAP** in **data warehouse** as a **computing technology** that allows query data and **analyze** it from different perspectives.
- The technology is a great solution for business analysts who need to pre-aggregate and pre-calculate data for fast analysis.
- Thats where **Apache Hive** comes in, **for batch processing** TB scales of data

Basic OLAP operations include(Already Covered):

- Drill up
- Drill down
- Slice
- Dice
- Pivot



How does Apache hive act as a data warehouse?



Airbnb created a data warehouse using Hive to achieve **reporting and analysis,** over a third of all employees at the company had launched an SQL query against Hive , just after launch.

Then What is Apache Hive?

Apache **Hive** is a **distributed**, f**ault-tolerant** data **warehouse** system that enables analytics at a massive scale and that too by writing powerful SQL queries.

This was created by **facebook** , where they are storing

- Several thousand tables
- 700 terabytes of data
- more than 1000 users.

and is being used extensively for both reporting and ad-hoc analyzes.

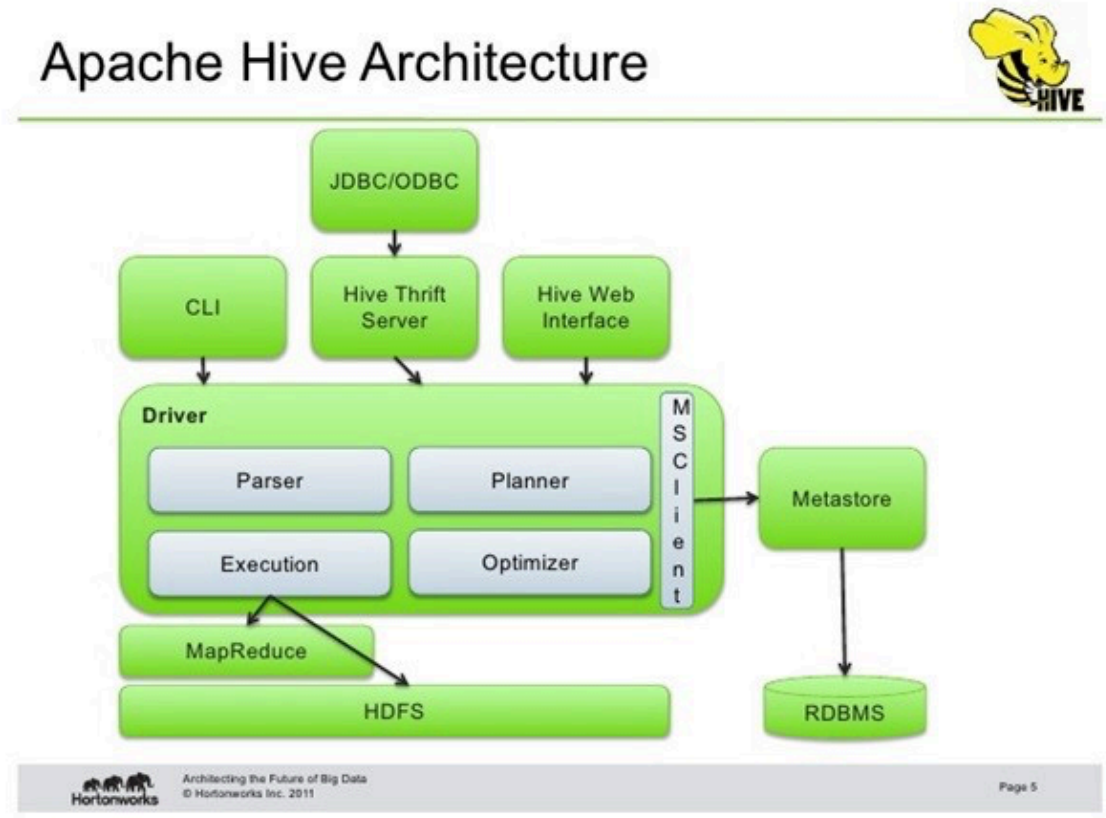
Similar to this there are giants who are **storing/processing** huge amounts of data in today's world inside Hive so what does Hive manage this?

How does Hive work under the hood? (Architecture)

All credit goes to its architecture which makes writing SQL queries possible to be executed on PB scales of data.

Main **Architecture** Components:

a. Hadoop core components(Hdfs, MapReduce)
b. Metastore
c. Hive Server
d. Driver
e. Compiler
f. Execution Engine
g. Hive Clients



a. Hadoop core components:

i. **HDFS**: When we load the data into a Hive Table it internally stores the data in HDFS path i.e by default in hive warehouse directory.

The hive default warehouse location can be found at

```
cd /etc/hive/conf
vi hive-site.xml

<property>
     <name>hive.metastore.uris</name>
     <value>thrift://nn01.jayserver.com:9083</value>
</property>

<property>
     <name>hive.metastore.warehouse.dir</name>
     <value>/apps/hive/warehouse</value>
</property>
```

ii. **MapReduce***:* By default any SQL query that we run will eventually be running a Map Reduce job by converting or compiling the query into a java class file, building a jar and executing this jar file.

```
hive (jvanchir_cards)> select count(*) from orders;

Query ID = jvanchir_20181101123144_73841f3a-3f1e-4119-8709-5c00d0bb01b9
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1540458187951_3054, Tracking URL = http://rm01.         .com:19288/proxy/application_1540458187951_3054/
Kill Command = /usr/hdp/2.6.5.0-292/hadoop/bin/hadoop job  -kill job_1540458187951_3054
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-11-01 12:31:57,583 Stage-1 map = 0%,  reduce = 0%
2018-11-01 12:32:03,975 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.02 sec
2018-11-01 12:32:09,255 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 7.81 sec
MapReduce Total cumulative CPU time: 7 seconds 810 msec
Ended Job = job_1540458187951_3054
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.81 sec   HDFS Read: 3007957 HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 810 msec
OK
68883
Time taken: 27.629 seconds, Fetched: 1 row(s)
```

b. **Metastore:** Namespace for tables.

i. This is a crucial part for the hive as all the **metadata** information related to the hive such as details related to the **table, columns, partitions**, location is present as part of it.

ii. Usually, the **Metastore** is available as part of the Relational databases eg: Derby,MySQL

```
vi hive-site.xml
    <property>
        <name>javax.jdo.option.ConnectionDriverName</name>
        <value>com.mysql.jdbc.Driver</value>
    </property>
    <property>
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>Hj*&%$</value>
    </property>
    <property>
        <name>javax.jdo.option.ConnectionURL</name>
        <value>jdbc:mysql://jay01.abc.com/metastore</value>  // The DB name in mysql is metastore
    </property>
```

Note: In real-time the access to metastore is restricted to the Admin and specific users.

c. **Hive Server:**
   i. Enables clients to execute queries against Hive
   ii. Supports multi-client concurrency and authentication
   iii. Designed to provide better support for open API clients like JDBC and ODBC

d. **Compiler:**
   i. **Parses the query**
   ii. **Does semantic analysis** on the different query blocks
   iii. **Query expressions**
   iv. Eventually generates an **execution plan** with the help of the table and partition metadata looked up from the **metastore**.

e. **Driver** : The component in architecture that:
   i. Used for direct SQL and HiveQL access to Hive, **enabling Business Intelligence** (BI), **analytics**, and **reporting** on Hive-based data.
   ii. Efficiently **transforms** an application's SQL query into the equivalent form in **HiveQL**
   iii. **Interrogates** Hive to obtain schema information to present to a SQL-based application
   iv. Overall driver is nothing but A **bunch** of **jar** files

Mostly here, programs in java/python/c++/c/scala/.NET need to be written. But inside them it's SQL only.

Consider this as a JDBC/ODBC connection used by Java to connect to RDBMS for fetching and working on data.

f. **Execution Engine:**
    i. Execution of the execution plan made by the compiler is performed in the execution engine.
    ii. The plan is a DAG of stages.
    iii. Dependencies within the various stages of the plan is managed by execution engine
    iv. it executes these stages on the suitable system components.
g. **Hive Clients:** It is the interface through which we can submit the hive queries. Example :
    i. **Terminal interfaces** : Hive CLI(deprecated),Beeline
    ii. **Web-interfaces** : Hue, Ambari

Note : Here the instructor should showcase demos of hive clients

What Does Job Execution Look Like?

To understand this , let's try to resolve **Airbnb** Analyst query :

Question : "**In 2019,Get the details of all travelers about their choice of stays in New York city i.e whether they book Entire home/apt or Private Room or they preferred Shared room**"

Dataset to be used: **NYC_2019.csv (https://drive.google.com/drive/folders/1I0vSzIJ-LhhZfax16-sVaKpo8BQBi6my?usp=sharing)**

Sample dataset:

id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_month	calculated_host_listings_count	availability_365
2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	2018-10-19	0.21	6	365
2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	2019-05-21	0.38	2	355
3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.9419	Private room	150	3	0			1	365
3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	2019-07-05	4.64	1	194
5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	2018-11-19	0.10	1	0
5099	Large Cozy 1 BR Apartment In Midtown East	7322	Chris	Manhattan	Murray Hill	40.74767	-73.975	Entire home/apt	200	3	74	2019-06-22	0.59	1	129
5121	BlissArtsSpace!	7356	Garon	Brooklyn	Bedford-Stuyvesant	40.68688	-73.95596	Private room	60	45	49	2017-10-05	0.40	1	0
5178	Large Furnished Room Near B'way	8967	Shunichi	Manhattan	Hell's Kitchen	40.76489	-73.98493	Private room	79	2	430	2019-06-24	3.47	1	220

Answer: we being analysts will quickly go to the hive console and will fire the below query.(Assuming table and data are already created)

```
31
32
33 select room_type,count(1) from 29june2022.airbnb_nyc_data group by room_type limit 3;
```

Output :

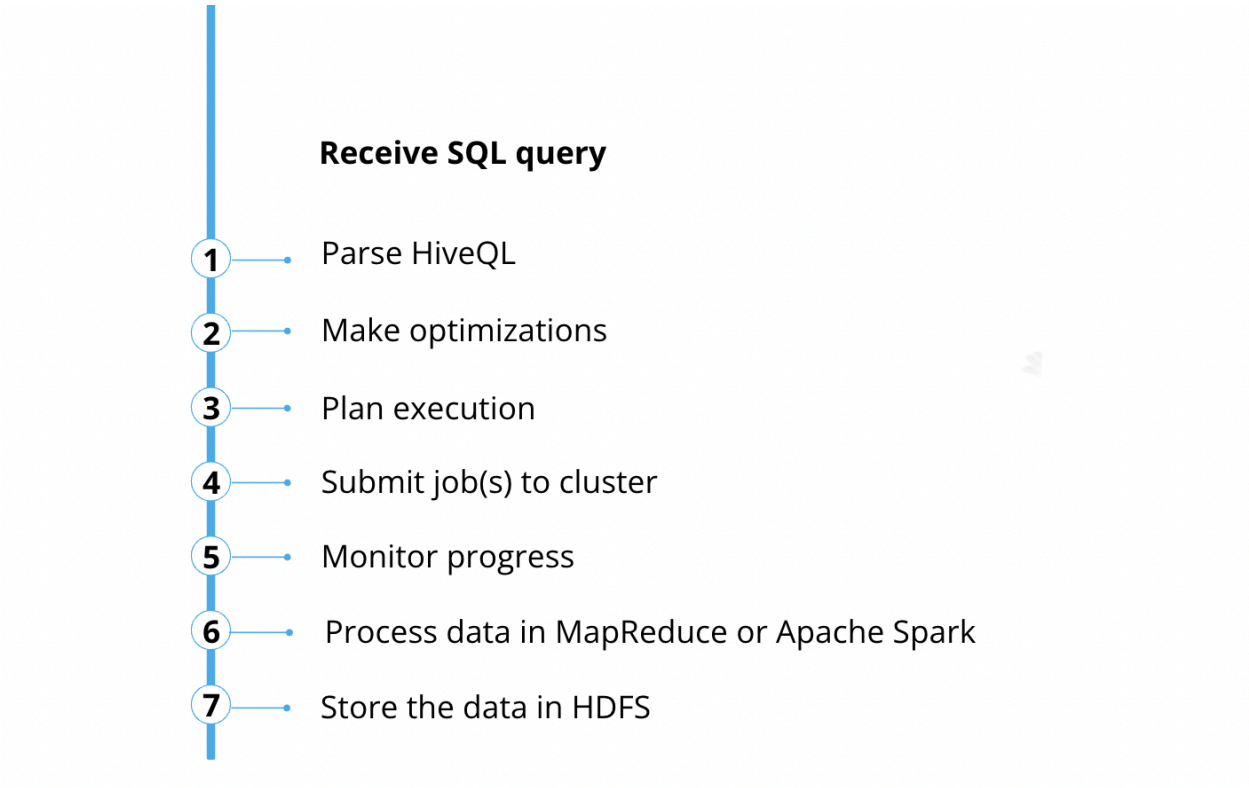
room_type	total
Entire home/apt	25409
Private Room	22326
Shared Room	1160

Awesome Job here!!!!

But what happened in the background from writing a query to getting a result in front of you. While the query was executing some logs like **mapreduce observed?**

SQL → MapReduce → Table(SQL)

1. First of all, the user **submits** their query and CLI sends that query to the Driver.
2. Then the **driver** takes the help of the query **compiler** to check syntax.
3. Then **compiler** requests for **Metadata** by sending a metadata request to Metastore.
4. In response to that request, metastore sends metadata to the compiler.
5. Then the compiler **resends the plan** to the driver after checking requirements.
6. The Driver sends the plan to the **execution engine**.
7. Execution engine query executes the **MapReduce** job. And in the meantime the execution engine executes metadata operations with Metastore.
8. Then the **execution engine** fetches the results from the Data Node and sends those results to the driver.
9. At last, the driver sends the results to the hive interface.



What are different HQL commands that are used commonly?

DDL Commands in Hive

CREATE	Database,Table
DROP	Database,Table
TRUNCATE	Table
ALTER	Database,Table

SHOW	Databases,Tables,Table Properties,Partitions,Functions,Index
DESCRIBE	Database, Table ,View

DML Commands in Hive

LOAD	`LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename`

INSERT	`INSERT OVERWRITE TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...) [IF NOT EXISTS]] select_statement1 FROM from_statement;`
	
	`INSERT INTO TABLE tablename1 [PARTITION (partcol1=val1, partcol2=val2 ...)] select_statement1 FROM from_statement;`

UPDATE	`UPDATE tablename SET column = value [, column = value ...] [WHERE expression]`
DELETE	`DELETE FROM tablename [WHERE expression]`
EXPORT	`EXPORT TABLE tablename [PARTITION (part_column="value"[, ...])]` `TO 'export_target_path' [ FOR replication('eventid') ]`

IMPORT	`IMPORT [[EXTERNAL] TABLE new_or_original_tablename [PARTITION (part_column="value"[, ...])]]`
	`FROM 'source_path'`
	`[LOCATION 'import_target_path']`

Question 2 : As data Engineer let's explore another query from sales team

“Get the trend among the visitors on how they are planning their vacations, some might be staying more than 3 days, some for 2 and maybe 1 days and show it through visualization.”

Dataset : NYC_2019.csv (same drive link)

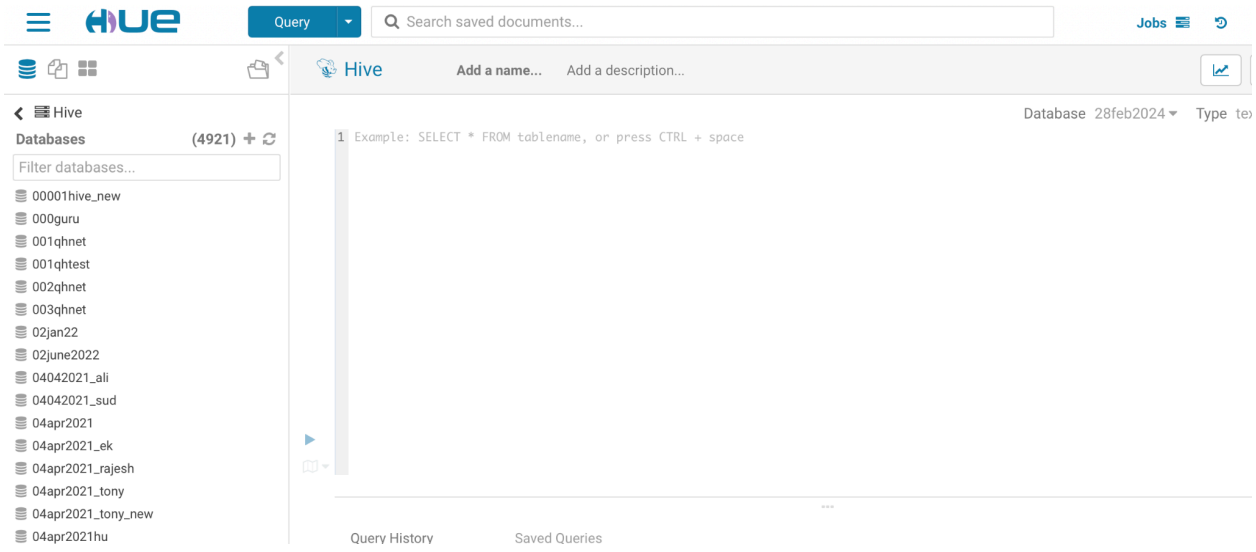
Procedure : Since we have explored DDL and DML commands we will limit our scope to them

Lets bifurcate our solutions in multiple steps:

Step 1: Log in to Hive GUI via HUE
Step 2: Create database as **airbnb_sales**
Step 3: Inside same database , create a table **trending_stays**
Step 4 : Populate table created with **NYC_2019.csv** file
Step 5: Build a query and execute
Step 6: Analyze the output via inbuilt visualization

Solution :

Step 1:



Step 2: Create database as **airbnb_sales**

3. Inside same database , create a table trending_stays

```
create table trending_stays
(
id INT,
name STRING,
host_id    INT,
host_name STRING,
neighbourhood_group STRING,
neighbourhood STRING,
latitude DOUBLE,
longitude DOUBLE,
room_type STRING,
price INT,
minimum_nights INT,
number_of_reviews INT,
last_review     STRING,
reviews_per_month DOUBLE,
calculated_host_listings_count INT,
availability_365 INT
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
```

```
WITH SERDEPROPERTIES (
   "separatorChar" = ",",
   "quoteChar"     = "\"")
TBLPROPERTIES("skip.header.line.count"="1");
```

Yuppy **table created successfully** but wait wait, what is there is last 4-5 lines

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
   "separatorChar" = ",",
   "quoteChar"     = "\"")
TBLPROPERTIES("skip.header.line.count"="1");
```

- **ROW FORMAT** : delimiters used tell how rows are terminated.By default “\n”
- **SerDe** is short for Serializer/Deserializer, allows Hive to read in data from a table, and write it back out to HDFS in any custom format.
    - ThriftSerDe → Default
    - AvroSerDe
    - OrcSerde
    - ParquetHiveSerDe
    - OpenCSVSerde

  Why use OpenCSVSerde ?

  In our case, if we see raw data in csv file provided we have name column value as **“Lovely Room 1, Garden, Best Area, Legal rental” ,** 1 single column having multiple commas and hence issue while loading so to avoid same we used:

  ```
  WITH SERDEPROPERTIES (
     "separatorChar" = ",",
     "quoteChar"     = "\"")
  ```

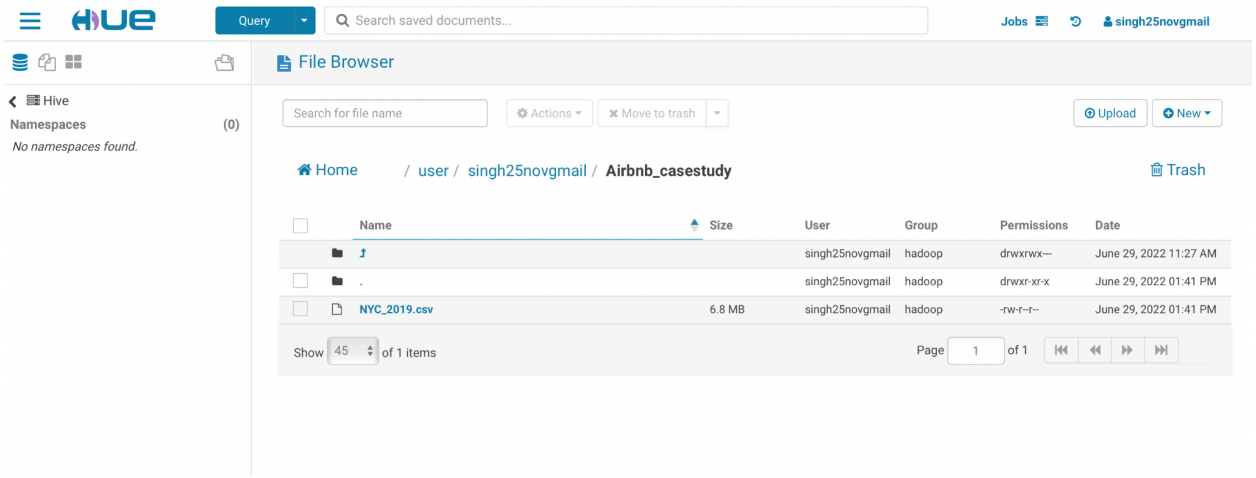
- **TBLPROPERTIES :** clause allows you to tag the table definition with your own **metadata** key/value pairs. In our case, we know that the data file has a **header** and this cannot be part of the data and hence skipping the same.

  Other predefined table properties include:

- ❖ TBLPROPERTIES ("comment"="*table_comment*")
- ❖ TBLPROPERTIES ("hbase.table.name"="*table_name*")
- ❖ TBLPROPERTIES ("immutable"="true") or ("immutable"="false")
- ❖ TBLPROPERTIES ("orc.compress"="ZLIB") or ("orc.compress"="SNAPPY") or ("orc.compress"="NONE") and other ORC properties
- ❖ TBLPROPERTIES ("transactional"="true") or ("transactional"="false")
- ❖ TBLPROPERTIES ("NO_AUTO_COMPACTION"="true") or ("NO_AUTO_COMPACTION"="false"), the default is "false"
- ❖ TBLPROPERTIES ("auto.purge"="true") or ("auto.purge"="false")

4. Populate table created with **NYC_2019.csv** file

Table has been created successfully but it is **empty,** we need to have data inside it and our csv file is present on our local disk. As we know, hive reads data from HDFS so the file **needs to be placed in HDFS dir.**



/user/singh25novgmail/Airbnb_casestudy → directory in HDFS
NYC_2019.csv → Data File inside

There are 4 ways to load data in Hive:

- Using **LOAD** Command (**preferred way**) : Hive provides us the functionality to load pre-created table entities either from our local file system or from HDFS
- Internally, Source file moves to default location: /user/hive/warehosue/airbnb_casestudy/NYC_2019.csv

Syntax:

LOAD DATA [LOCAL] INPATH '<The table data location>' [OVERWRITE] INTO TABLE <table_name>;

Note:

1. The **LOCAL** Switch specifies that the data we are loading is available in our Local File System. If the LOCAL switch is not used, the hive will consider the location as an HDFS path location.
2. The **OVERWRITE** switch allows us to overwrite the table data.

Final Command:

```
LOAD DATA INPATH
'/user/singh25novgmail/Airbnb_casestudy/NYC_2019.csv' INTO TABLE
trending_stays;
```

- Using **INSERT** command : Just like **SQL** , here hiveQL also provides a similar way to upload data but here the source table has to be there upfront.
- It calls **MapReduce** in background

Syntax:

INSERT INTO TABLE <table_name> VALUES (<add values as per column entity>);

What are the optimization techniques in Hive ?

Initially Airbnb started using **Apache Hive** as their core Data Warehouse as is. While Hadoop/hive can process nearly any amount of data, **optimizations** can lead to big savings, proportional to the amount of data, in terms of processing time and cost.

Below are the optimizations that are recommended :

a. Execution Engine in Hive

Later on with time , the Airbnb team realized that existing Hive architecture needs to be optimized in order to fully harness it and they started working on optimizing the query engine first as MR was getting stale. Below are the options:

- **Apache Tez** :**Apache Tez** is a client-side library which operates like an execution engine, an alternative to traditional **MapReduce Engine**.

  set hive.execution.engine=tez;

- **LLAP(Long Lived Analytical Processing):**a SQL-on-Hadoop processing framework, bringing the promise of low latency SQL queries on Hadoop and YARN.
- **Hive on Spark:** Latest and recommended, provides Hive with the ability to utilize Apache Spark as its execution engine. Will cover Spark in detail

—> IN spark add hive too

b. Partitioning

Problem Statement:
Airbnb marketing decided to review their targets every biweekly on every Monday(gap of 14 days) where they will strategize and bring efficient decisions to the table.

To finalize this, they need data on every 14th and 28th of every month from hive.But every time scanning billions of rows doesn't make sense. Unnecessary scanning of tables can lead to latency with eventually query failing.

Proposed Solutions :

Option 1 : Create a separate table for each date every month before the marketing team asks?
Option 2: Let's divide the table into parts based on the values of particular columns**.**

Deserving Solution: Best way to go ahead is with Option 2

So what **Option 2** talks about is **Partitioning**.

- Partitioning divides the table into parts based on the values of particular columns.
- A table can have multiple partition columns to identify a particular partition.
- Using partitions it is easy to do queries on slices of the data.
- The data of the partition columns are not saved in the files

- When we query data on a partitioned table, it will only scan the relevant partitions to be queried and skip irrelevant partitions.
- Partitioning is basically of two types: **Static** and **Dynamic**.

Static partitioning:

- Manually decide how many **partitions** tables will have and also value for those partitions
- Here tables are populated by using the **LOAD** command to create the partition.
- Usually when loading files (big files) into Hive tables static partitions are preferred.
- Enable Static partition in the hive as:
    - **set hive.mapred.mode = strict**
- Always use **where** clause to use limit in the static partition.

Dynamic partitioning:

- This provides flexibility and creates partitions automatically depending on the data that we are inserting into the table.
- INSERT with a SELECT query are used to create multiple partitions in table sales dynamically.
- If there is a requirement to partition a number of columns but you don't know **how many columns** then also **dynamic partition** is suitable.
- Here, there is no mandatory need of **where clause**.
- To enable dynamic partitioning:
    - **set hive.exec.dynamic.partition=true;**
    - **set hive.exec.dynamic.partition.mode=nonstrict;**

When to use Partitioning?

1. When reading the entire data set takes too long
2. When queries almost always filter on the partition columns
3. When there are a reasonable number of different values for partition columns

When Not to Use Partitioning

1. When columns have too many unique rows
2. When creating a dynamic partition as it can lead to high number of partitions
3. When the partition is less than 20k

Bucketing

Problem Statement:
Airbnb marketing decided to review their targets every biweekly on every Monday(gap of 14 days) and this time they added that data should be given company verticals, for example :

a. Get data from sales team
b. Get data from Infrastructure team
c. Get data from customer care team

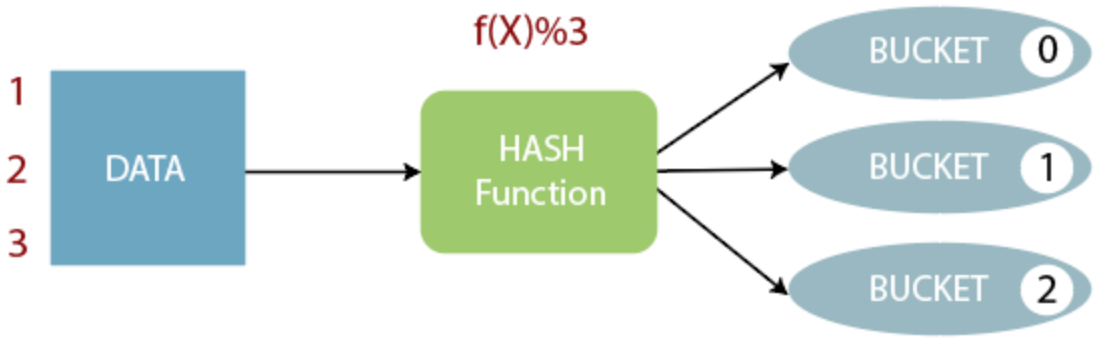
To finalize this, they need data on every 14th and 28th of every month from hive.But every time scanning billions of rows doesn't make sense. Unnecessary scanning of tables can lead to latency with eventually query failing.

Proposed Solution:
a. Partitioning can be used to group data this way
b. Groupby statement can be used while issuing query
c. Segregating hive table data into multiple files/directories.

Solution:
Option a : as we already discussed partitioning above and it clearly says that data can be distributed as partition across clusters but inside partition there is no way to group.
Option b : Group By statement is used at client side to get data segregated. Here we dont want TB/PB of data to be streamed online and then gets groped

Option c: It's better to segregate data into multiple files and this is what is called **bucketing.**

Bucketing: provides flexibility to further segregate the data into more manageable sections called buckets or clusters and this is how it works:
þ



a. The concept of bucketing is based on the hashing technique.
b. Here, modules of current column value and the number of required buckets is calculated (let say, F(x) % 3).
c. Now, based on the resulting value, the data is stored into the corresponding bucket.
d. Records which are bucketed by the same column value will always be saved in the same bucket.

Using AVRO/ORC/Parquet Format

By default, **text file**(csv,tsv) is the default format supported by hive. Yes!! This is human friendly (easy to read) and it has support for read/write in any programming language but is not recommended for big data as:

a. Consumes more space to store numeric value as string
b. Difficult to represent in binary form

Avro was born to be used in Hive and this has major improvements like:

- Embeds schema metadata in file
- Supports **JSON** format
- Serializing data in a compact binary format
- Ideal for long term storage.
- Efficient storage

SAMPLE AVRO FILE:

```
{
  "name": "cia.sad.Operative",
  "type": "record",
  "fields": [
    {
      "name": "name",
      "type": "string"
    },
    {
      "name": "birthdate",
      "type": {
        "type": "int",
        "logicalType": "date"
      }
    },
    {
      "name": "operation",
      "type": {
        "name": "cia.sad.Operation",
        "type": "enum",
        "symbols": ["SILVERLAKE", "TREADSTONE", "BLACKBRIAR"]
      }
    }
  ]
}
```

Disadvantages :

1. AVRO supports row wise storage. For ex: in below table if we need information about ID = 3, first 2 rows will be scanned unnecessarily hene more I/O

ID	Name	Department
1	emp1	d1
2	emp2	d2
3	emp3	d3

The data in a row-wise storage format will be stored as follows:

1	emp1	d1	2	emp2	d2	3	emp3	d3

2. AVRO is not that much fit for Analytical queries.
3. To read/write data, everytime schema is needed.

Parquet File Format:

And then came a columnar file format which is :

1. More efficient
2. Improves query performance
3. Store data with nested structures
4. Low storage consumption.

Example:

1	2	3	emp1	emp2	emp3	d1	d2	d3

Cost-based Optimizer

There are two major optimizers now in Hive to further optimize query performance in general,

a. Query Vectorization :
   i. Allows Hive to process a batch of rows together instead of processing one row at a time
   ii. Operations are performed on the entire column vector, which improves the instruction pipelines and cache use
   iii. To enable vectorization,
   SET hive.vectorized.execution.enabled=true; -- default false

b. Cost-Based Optimization (CBO):
   i. A cost-based optimizer (CBO) generates efficient query plans.
   ii. CBO provides two types of optimizations: logical and physical.
   iii. The CBO uses column and table statistics stored in the Hive metastore to create query plans that improve query performance.
   iv. Statistics for the CBO are objects that contain information about the columns and partitions of a table.
   v. The CBO uses these statistics to estimate the cardinality and number of distinct values, etc. These estimates then help generate highly efficient query plans.
   vi. To enable it :
      1. **SET hive.cbo.enable=true** -- default true after v0.14.0

Where to use Apache hive ?

Hive is a data warehousing solution so it is mostly used for Reporting,Analytics , don't ever use this for storing day to day(transactional) data.

Certain big names using Hive for analytical purposes:

a. **Tiktok** → Tiktok used Hive for storing data to be consumed in its own recommendation engine.
b. **Walmart** → Walmart has transformed decision making by incorporating Hive in the business world resulting in repeated sales. Walmart observed a significant 10% to 15% increase in online sales for $1 billion in incremental revenue
c. **Netflix** → Hive is being used at **Netflix** for ad hoc queries and analytics.On a daily basis, they run **hundreds** of Reporting jobs on hive from their visualization tools plus **thousands of Hive and Pig-based ETL jobs.**
d. **Facebook** → FB warehouses stores upwards of **300 PB of data in Hive**, with an incoming daily rate of about **600 TB.**The most widely used system at Facebook for large data transformations on raw logs is Hive.
e. **UnitedHealthGroup** → UHG uses hive to **analyze**,**gathered** customer feedback through direct marketing channels and through social media in order to make intuitive decision , target relevant customers.