DATA ENGINEERING ON YELP DATASET USING HADOOP

1. BIGDATA

Big Data Definition

Big Data is the collection of huge datasets of semi-structured and unstructured data, generated by the high-performance heterogeneous group of devices ranging from social networks to scientific computing applications.

- Evolution of Data

The size of this data ranges from a few hundred gigabytes to zetabytes that surpasses the data
capturing, storing, processing capabilities of traditional database management tools.

- Sources of Big Data

Traditional Database's - Databases including a variety of data sources, such as MS Access, DB2, Oracle, SQL, and Amazon RDS etc contribute to big data
Social Media - Data generated is in the form of images, videos and text.
Cloud - Data from commodity, finance,various industries are stored on the cloud. AWS, Azure, Gsuite etc
IoT - IoT is an ecosystem of smart devices connected through the internet, capable of collecting, exchanging user data.

- Characteristics of Big Data

Volume
Velocity
Variety
Veracity

- Some applications of big data

Retail
Financial Services
Healthcare
Manufacturing

- Types of data

structured

semi-structured
unstructured.

- Job Roles in Big Data Domain:

Big data consultant
Architect
Big data engineers
Data Analyst

2. HDFS & MR

Hadoop

- Hadoop is an open source, Java based programming framework that manages data processing and storage for big data applications in a distributed computing system.

- It has two main components HDFS and the YARN.

Features of Hadoop

- Flexibility in data processing

  Hadoop provides flexibility regarding data processing as it processes unstructured data as well.

- Scalability

  Hadoop is easily scalable as it is an open-source platform that runs on a distributed
  system involving thousands of nodes.

- Fault-proof- Hadoop is a fault-proof platform as the data is stored in HDFS.

- Faster processing speed-

  Hadoop does batch processing at a faster rate as it supports parallel processing.

- Cost-effective

Hadoop is quite cost-effective as it utilises commodity hardware to store data.

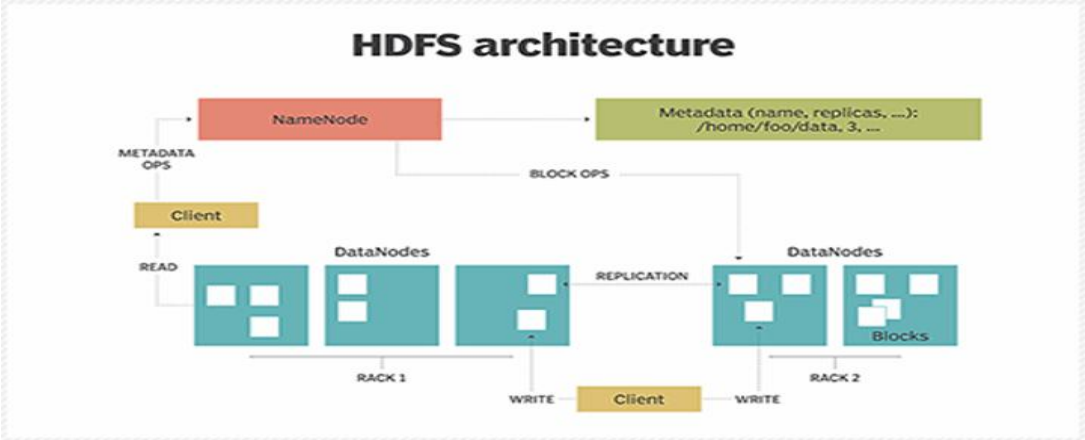
Hadoop vs RDBMS

- In RDBMS, query response time is immediate. In Hadoop, due to batch processing there is a time gap.

- Hadoop doesn't support real time data processing (OLTP), it is designed to support large scale batch processing work loads (OLAP) where as RDBMS supports real-time data processing.

Hadoop Architecture

Hadoop also follows the master-slave architecture. The master is known as the NameNode and the slaves are referred to as DataNodes.



Key Components
Hadoop distributed file system (HDFS)
MapReduce

Main components of HDFS

NAME NODE
DATA NODE
Secondary NameNode
STANDBY Namenode

YARN - Yet Another Resource Negotitator

YARN also follows the Master-Slave architecture.
Master - ResourceManager
Slave - NodeManager

Key aspects of HDFS

REPLICATION IN HDFS
SPLITTING OF DATA

Read/Write Operation in the HDFS:

To read a file from the HDFS, a client needs to interact with the NameNode, as it stores all the metadata. NameNodes check for the required privileges. It also provides the address of the data nodes where a file is stored. After that, the client will interact directly with the respective data nodes and read the data blocks. The read operation in the HDFS is distributed, and the client reads the data parallelly from the DataNodes.

MAP-REDUCE

MAP - Definition

Data parallel model is used in MAP phase.

MAP can be interpreted as one line of code for any data-parallel algorithm, where one function gets repeatedly executed on data distributed across multiple processors.
Ideally all the cleaning, Filtering and data transformation services without any aggregate services can be achieved through Map Construct.

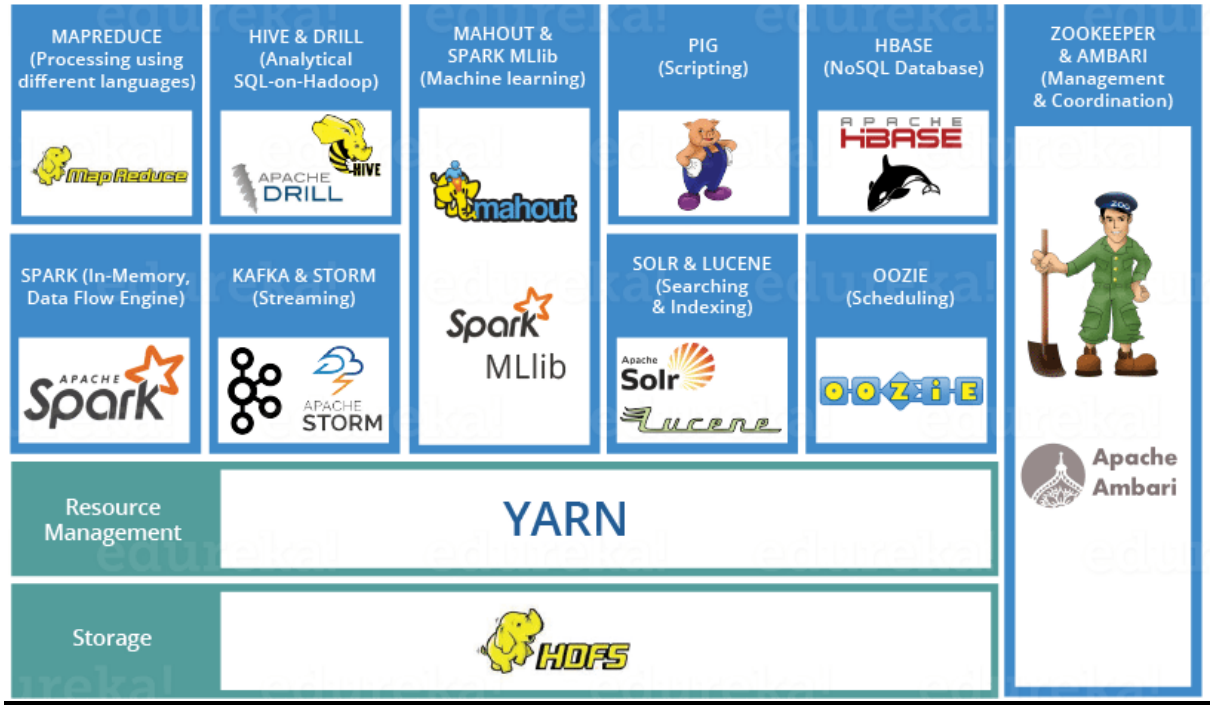
Reduce - Definition

Inverse-tree parallel model for aggregating results in Reduce phase

Reduce, as the name suggests, is the act of performing any valid operation parallelly on a chunk of input data and getting an output that is much smaller in size than the input.

Finding max in a list of numbers, Finding sum in a list of numbers, Finding a unique set of numbers from a list of numbers, Find average of a list of numbers are few sample operations that can be performed using ReduceConstruct.

HADOOP ECO SYSTEM



3. HIVE

Hive is a data warehouse software

Hive vs RDBMS

Hive - Write once, Read many times but RDBMS - Read and Write many times.

Table Operations

OLAP vs OLTP

Refer Link : http://hadooptutorial.info/hive-vs-rdbms/

Uses of Hive-Metastore

HIVE Data Types :

Primitive Data type - String,Int,float,double,Timestamp,Binary

Complex Data type – STRUCT(Object) , MAP(Key-Value), ARRAY(Indices)

P_numbers=Struct{areacode INT,number INT}
P_number.areacode

P_number=MAP(“statecode”,”044” “number”,”123456”)
P_number[“number”]

V_id array(‘id1’,’id2’,’id3’)
V_id [0]

Internal vs External tables

4. YELP Dataset and Students.dat

Display file << STUDENT.DAT>>

```
CREATE TABLE IF NOT EXISTS students (
name STRING ,
id INT ,
subjects ARRAY < STRING >,
feeDetails MAP < STRING , FLOAT >,
phoneNumber STRUCT <areacode: INT , number : INT > )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
COLLECTION ITEMS TERMINATED BY '#'
MAP KEYS TERMINATED BY '|'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

Syntax:
load data inpath 'add path to your file here' overwrite into table <table_Name>;

Query:
load data '/root/hive/student.dat' overwrite into table students;

To check the output

Select * FROM students;

The Yelp Dataset

Link : https://www.yelp.com/dataset_challange

About the Review dataset

The Challange Dataset:

2.7M reviews and 649K tips by users for 86K business

566K business attributes, e.g hours,parking,availability,ambience
Social network of 687K users for a total of 4.2M social edges
Agregated check-ins over time for each of the 86K businesses
200,000 pictures from the included businesses

Data Engineering is driven by goal

Yelp Dataset & Schema
Intrested areas of research

Cultural Trends
By adding a dicerse set of cities,we want participants to compare and contrast what makes a particular cty different. For example : are prople in international cities less concerned about driving in to a business, indicated by their lack of mention about parking ? What cuisines are Yelpers raving about in these different countries ? Do Americans tend to eat out late compared to the Germans and English ? In which countries are Yelpers sticklers for service quality ? In international cities such as Montreal are french speakers reviewing places differently than English speakers ?

Location Mining and Urban panning
How much of a business success is really just location ? Do you see reviewers behaviour change when they travel ?

Seasonal Trends
What about seasonal effects: Are HVAC contractors being reviewed just at onset of winter, and manicure salons at onset of summer? Are there more reviews for sports bars on major game days and if so. Could you predict that?

Infer Categories
Do you see any non-intutive correlations between business categories eg: how many karaoke bars also offer Korean food and vice versa? What business deserve their own subcategory(ie) Szechuan or Hunan versus just "Chinese restaurants")and can you learn this from the review text ?

Natural language ProcessingNLP

How well can you guess a review's rating from its text alone? What are the most common positive and negatives words used in our reviews ? Are Yelpers a sarcastic bunch? And what kinds of correlations do you see between tips and reviews. could you extract tips from reviews?

Changepoints and Events
Can you detect when things change suddenly(ie) a business coming under new management)? Can you see when a city starts going nuts over cronuts ?

Social Graph Mining

Can you figure out who the trend setters are and who found the best waffle joint before waffles were cool ? How much influence does my social circle have on my business choices and my ratings ?

Domain
Business
Review
User
check-in
tip
photo

5. YELP Data modelling, table creations

<< Display JSON files here >>

{"user_id": "oMy_rEb0UBEmMlu-zcxnoQ", "name": "Johnny",
"review_count": 8, "yelping_since": "2014-11-03", "friends":
["cvVMmlU1ouS3I5fhutaryQ", "nj6UZ8tdGo8YJ9lUMTVWNw",
"RTtdEVhAmeWqCSp0IgJ99w", "t3UKA1sl4e6LY_xsjuvI0A",
"s057_BvOfnKNvQquJf7VNg", "VYrdepCgdzJ4WaxP7dBGpg",
"XXLSk6sQQDyr3dZ4zE-O0g", "Py8ThfExQaXF2Woqr7kWUw",
"233YNvzVtZ1ObkaNkUzNIw", "L6iE9NpmHHJQTk0JQlRlSA",
"Y7XTMgZ_q5Bj5f9KhK1R4Q"], "useful": 0, "funny": 0, "cool": 0, "fans": 0,
"elite": [], "average_stars": 4.67, "compliment_hot": 0, "compliment_more": 0,
"compliment_profile": 0, "compliment_cute": 0, "compliment_list": 0,
"compliment_note": 0, "compliment_plain": 1, "compliment_cool": 0,
"compliment_funny": 0, "compliment_writer": 0, "compliment_photos": 0}

Structure of user.Json

user
{
' user_id’: (encrypted user id),
' name': (user Name),
' review_count': (count of the number of reviews given)'
yelping_since': (date,formatted like '2021-03-14'),
' friends': (list of encrypted user id’s)
‘Useful‘ (Ratings)
‘Funny‘(Ratings)
‘Cool‘(Ratings)
‘Fans‘(Ratings)

‘elite array<string>
‘average_stars (Average stars given by user),
‘compliment_hot‘: 0,
‘compliment_more‘: 0,
‘compliment_profile‘: 0,
‘compliment_cute‘: 0,
‘compliment_list‘: 0,
‘compliment_note‘: 0,
‘compliment_plain‘: 1,
‘compliment_cool‘: 0,
‘compliment_funny‘: 0,
‘compliment_writer‘: 0,
‘compliment_photos‘: 0}
}

Structure of review.Json

{"review_id":"v0i_UHJMo_hPBq9bxWvW4w","user_id":"bv2nCi5Qv5vroFiqKGopiw","business_id":"0W4lkclzZThpx3V65bVgig","stars":5,"date":"2016-05-28","text":"Love the staff, love the meat, love the place. Prepare for a long line around lunch or dinner hours. \n\nThey ask you how you want you meat, lean or something maybe, I can't remember. Just say you don't want it too fatty. \n\nGet a half sour pickle and a hot pepper. Hand cut french fries too.","useful":0,"funny":0,"cool":0}

review

‘review_id’: (encrypted user id),
'user_id': (encrypted user id),
'business_id': (encrypted business id),
'stars': (star rating, rounded to half-stars),
'date': (date,formatted like '2021-03-14'),
'text': (review text)
Useful: (votes)
Funny: (votes)
cool: (votes)

YELP Table creation Syntax :

```
CREATE TABLE yelp_user_hdfs (
user_id string,
name string,
review_count bigint,
```

```
yelping_since string,
friends array<string>,
useful_rating double,
Funny_rating double,
Cool_rating double,
Fans_rating double,
elite array<string>,
average_stars float
)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
with serdeproperties ( 'paths' = '' );

load data local inpath '/home/hadoop/users_250.json' overwrite into table
yelp_user_hdfs;

CREATE TABLE yelp_review_hdfs (
review_id string,
user_id string,
business_id string,
stars bigint,
r_date string,
text string,
useful double,
funny double,
 cool double
)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
with serdeproperties ( 'paths' = '' );

load data local inpath '/home/hadoop/review_250.json' overwrite into table
yelp_review_hdfs;
```

6. Basic Queries in Hive and few UDF explanation

Queries on students.dat

Execute few basic queries on user and review tables.

```
Select * from yelp_user_hdfs limit 10;
Select * from yelp_review_hdfs limit 10;
```

Workings :

Query Workings :

```
-- 1. Number of reviews
select count (*) as review_count
from yelp_user_hdfs;

- 2. check the size of review text
select text,size(split(text,' ')) as words_count
from yelp_review_hdfs limit 10 ;

-- 3. Gives the total number of records, total number of unique user, min
and avg star values

select count (*) as Total_recs_count, count (distinct user_id) as
user_count,
min (average_stars) as min_star,
avg (average_stars) as avg_star
from yelp_user_hdfs;

=== Analyse the data
What is the avg number of review each reviwer given

select avg (review_cnt)
from (
select user_id, count (*) as review_cnt
from yelp_user_hdfs
group by user_id
)a;

2. Average reviews count by each user
avg - is one of the common used UDF it will tell the growth pace

select avg (stars_count)
from (
select review_id, sum (stars) as stars_count
from yelp_review_hdfs
group by review_id
)a;
```

Datafile : students.dat

NESTED QUERIES IN HIVE :

SUBquery is not supported in HIVE, we can utilize nested query column aliais cannot be used in where clause

1. Query to determine the set of students having lab fees greater than 1000

```
Select name , feeDetails[ 'labfee' ] from students where
feeDetails[ 'labfee' ] > 1000 ;
```

2. Assume that the labfees are getting hiked by 30% and you would like to see the
projected values in case it is greater than 1800.

Method : 01

```
Select name , feeDetails[ 'labfee' ]* 1.3 as projected_fee from students
where feeDetails[ 'labfee' ]* 1.3 > 1800 ;
```

TIME TAKEN : 0.127 seconds, Fetched: 2 row(s)

// when computation is complex this method will not work

Method : 02

NESTED SELECT STATEMENT

```
FROM
(SELECT name , feeDetails[ 'labfee' ]* 1.3 as projected_fee from
students) newfee
SELECT newfee.name,newfee.projected_fee where newfee.projected_fee
> 1800 ;
```

TIME TAKEN : 0.099 seconds, Fetched: 2 row(s)

Useful Links :

SUB Queries in SELECT clause Help document :

https://cwiki.apache.org/confluence/display/Hive/Subqueries+in+SELECT

SUB Queries in FROM clause Help document :

https://cwiki.apache.org/confluence/display/Hive/LanguageManual+SubQueries

Joins

```
select yelp_user_hdfs.user_id,yelp_review_hdfs.review_id
from yelp_user_hdfs JOIN yelp_review_hdfs
ON yelp_user_hdfs.user_id=yelp_review_hdfs.user_id limit 10;
```

Advance functions in HIVE

1. Explode() :
//Different values are displayed in single column
// HIVE UDTF's are not supported in nested expressions (select name, explode(0) from table;)

```
Select explode(feedetails) FROM students;
Select explode(subjects) FROM students;
Select explode(feedetails) FROM students WHERE name = "Alexa" ;
```

2. Upper() :
```
Select upper ( name ) from students;
```

3. Regex_Replace()
```
Select regexp_replace(concat(upper(name),id ),' ','') as username from students;
```

Three Things acheived
Name is converted in Uppercase
concatenate name and ID
To remove mid space reg exp replace replace space with null

Hint:

count(*) - returns the total no. of retrieved rows, including rows containing null values
count(column) - returns the no. of rows for which the given column is not null
sum(column) - returns the sum of values in the given column

avg(column) - returns the average of values in the given column (= sum (column) /count(column))

Useful Links :

SUB Queries in SELECT clause Help document :

https://cwiki.apache.org/confluence/display/Hive/Subqueries+in+SELECT

SUB Queries in FROM clause Help document :

https://cwiki.apache.org/confluence/display/Hive/LanguageManual+SubQueries

7. Partitioning

Partitions are used to make queries faster by dividing the tables into smaller parts using partition key columns.

if in a table with details of employees in a company(name, salary, department, designation), you can create a partition for each department. Hive will store each partition separately and will scan only the partition that is needed, thereby making the query faster.

Static and Dynamic Partitioning

Helpful Link : https://community.cloudera.com/t5/Support-Questions/what-is-the-difference-between-partition-Static-and-Dynamic/td-p/21028

STATIC PARTIONING

Insert input data files individually into a partition table is Static Partition
Usually when loading files (big files) into Hive Tables static partitions are preferred
Static Partition saves your time in loading data compared to dynamic partition
You “statically” add a partition in table and move the file into the partition of the table.
We can alter the partition in static partition

The Data should contain data only for the mentioned path
Partition value to be given in query

STATIC PARTIONING - Example

Datafile : customer.dat

```
CREATE TABLE IF NOT EXISTS customer_partitioned(
customer_fname varchar ( 64 ),
customer_lname varchar ( 64 ),
customer_addr string ,
city varchar ( 64 ))
PARTITIONED BY (country VARCHAR (64),state VARCHAR (64))
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

REPEAT the steps to load all datafiles

```
LOAD DATA INPATH 'yourfilepath'
INTO TABLE customer_partitioned
PARTITION (country= 'usa' , state= 'az' );

LOAD DATA INPATH 'yourfilepath'
INTO TABLE customer_partitioned
PARTITION (country= 'india' , state= 'mh' );
```

Use commands

```
show partitions customer_partitioned
```

To check in HADOOP :

```
hadoop fs -ls /user/hive/warehouse
hadoop fs -ls /user/hive/warehouse/customer_partioned
hadoop fs -ls /user/hive/warehouse/customer_partioned/country=india
```

DYNAMIC PARTIONING

Single insert to partition table is known as dynamic partition
Usually dynamic partition load the data from non partitioned table
Dynamic Partition takes more time in loading data compared to static partition
When you have large data stored in a table then Dynamic partition is suitable.
If you want to partition number of column but you don't know how many columns then also dynamic partition is suitable

IN HDFS prompt :

```
hadoop fs -mkdir /user/root/hive/customer_addr
```

GRANT PERMISSION

```
hadoop fs -put customer.dat
```

IN HIVE PROMPT :

```
CREATE TABLE IF NOT EXISTS customer(
customer_fname varchar ( 64 ),
customer_lname varchar ( 64 ),
customer_addr string ,
city varchar ( 64 ),
state VARCHAR (64),
country VARCHAR (64))
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

LOAD DATA INPATH '' INTO TABLE customer;

select * from customer;

set hive.exec.dynamic.partition=true;
set hive.exec.dynamic.partition.mode = nonstrict;

CREATE TABLE partitioned_customer(
customer_fname varchar(64),
customer_lname varchar(64),
customer_addr string,
city varchar(64))
PARTITIONED BY (country VARCHAR(64),state VARCHAR(64));
```

Load data into the partitioned table using the previously created normal table
This query will fire a MR job to do the insertion
Included partition column in the last in the required order

```
Insert into table partitioned_customer partition(country,state)
select
customer_fname,customer_lname,customer_addr,city,country,state from
customer;
```

-- Below query will display partitions created
show partitions partitioned_customer;

Select * from partitioned_customer where state='az' and country='usa';

If you want to drop any partitions, it can be done as:
ALTER Table partitioned_customer DROP IF EXISTS
PARTITION(country='usa', state='nw');

Too much granularity then metastore gets over loaded.

Purpose of Partitioning and Bucketing

This helps breaking big file to smaller chunks
HIVE can skip reading unwanted portions

YELP – dataset partioning

There is not straight forward column available for partition , so we extract year from yelping_since and use that as a partition key.

```
CREATE TABLE yelp_user_year (
user_id string,
name string,
review_count bigint,
yelping_since string,
friends array<string>,
useful_rating double,
Funny_rating double,
Cool_rating double,
Fans_rating double,
elite array<string>,
average_stars float,
yr string
);

insert overwrite table yelp_user_year
select
user_id,name,review_count,yelping_since,friends,useful_rating,funny_rating,co
ol_rating,fans_rating,elite,average_stars,substr(yelping_since,1,4)  from
yelp_user_hdfs;
```

```
select distinct(yr) from yelp_user_year;
select min(yr), max(yr) from yelp_user_year;


set hive.exec.dynamic.partition=true;
set hive.exec.dynamic.partition.mode = nonstrict;

create external table if not exists
yelp_user_hdfs_partioned (
user_id string,
name string,
review_count bigint,
yelping_since string,
useful_rating double,
Funny_rating double,
Cool_rating double,
Fans_rating double,
average_stars float
)
partitioned by
(yr string);

insert overwrite table yelp_user_hdfs_partioned partition (yr)
select user_id,
name,
review_count,
yelping_since,
useful_rating,
Funny_rating,
Cool_rating,
Fans_rating ,
average_stars,
yr
from yelp_user_year;

show partitions yelp_user_hdfs_partioned;
```

Run query for partition and non partition table

a. Partition Table

```
select count(*) as user_count
from yelp_user_hdfs_partioned;
```

b. For normal table

```
select count(*) as user_count
from yelp_user_hdfs
where yr = 2010;
```

Partitioning is one of the ways to improve query performance
Date and Time is most popular columns
try to be logical

If there is an analysis on a country level, so partition by country
helpful if we run a sentiment analysis

8. Bucketing , Few practices

If file size is huge, dictate hive how to create multiple files can be consolidated (partioning with a order)

Partitioning allows you to organise data into different directories.

Partitioning is one step above bucketing

Bucketing allows you to organise data into different files.

Primary Difference

Partitioning is always done on a column

Partitioning does not guarantee equal amount of distribution

Partitioning is always done on a column,

bucket uses a hash function hive knows through the hash function which file that record should go based on the table bucket scheme.

Because we use hash function mostly results in files of equal size.

```
set hive.enforce.bucketing=true ;
```

YELP example

create external table if not exists

```
yelp_user_hdfs_partioned_yr_bucket (
user_id string,
name string,
review_count bigint,
yelping_since string,
useful_rating double,
Funny_rating double,
Cool_rating double,
Fans_rating double,
average_stars float
)
partitioned by (yr string)
clustered by (user_id) into 4 buckets
location ' ';


insert overwrite table yelp_user_hdfs_partioned_yr_bucket partition (yr)
select user_id,
name,
review_count,
yelping_since,
useful_rating,
Funny_rating,
Cool_rating,
Fans_rating ,
average_stars,
yr
from yelp_user_hdfs_partioned;
```

So far we have studied the following:

Advanced fucntions in Hive
Creating and Querying Partitions
Creating and Querying using buckets
Advantages of partitioning & bucketing

Understand check

If you consider GST records of India,
Which of the following columns should you NOT use as a partition key?
How many unique partitions will be created if you use “State” as partition key?
Bucket using District

Excercise : TRY applying partitioning and bucketing for review dataset.

9. Compression and advanced queries n_grams, split,size,sentence UDF

File Formats

ORC

Optimized Row Columnar FORMAT
Stores data in Column format
Offer better performance
By default files are compressed
Good data homegenity - good compression ratio(size of the data is greatly reduced)
Avoid unwanted data seek using strip (collection of rows) details (250 MB, Index data, file footer)
Contents stored as binary, not human readable (orc reader or serdes used by hive only can read)

In column major order i.e. if there are three columns in a table then column 1 is written first in contiguous memory units of the disk, then column2 and so on. So, data of a single column is always present together in disk.

Syntax:
stored as orc location ''
tblproperties("orc.compress"="SNAPPY")

YELP ORC creation

```
CREATE EXTERNAL TABLE yelp_user_hdfs_ORC (
user_id string,
name string,
review_count bigint,
yelping_since string,
friends array<string>,
useful_rating double,
Funny_rating double,
```

```
Cool_rating double,
Fans_rating double,
elite array<string>,
average_stars float,
yr string
)
STORED AS ORC
LOCATION ''
TBLPROPERTIES ('orc.compress'='SNAPPY');

insert overwrite table yelp_user_hdfs_ORC
select * from yelp_user_hdfs;
```

Parquet

Developed by cloudera and twitter, another file format in hadoop ecosystem

Built for complex data types

Ability to choose compression scheme

All the meatadata written at the end of parquest file.

Seperates meta data from column data.

Parquet works best with Apache spark. Because of vectorized execution
Presto works well with ORC file

Based on nature of data we need to experiment and conclude.

```
stored as parquet
 location ''
tblproperties("parquet.compress"="SNAPPY")
```

COMPLEX QUERIES - statistical analysis

UDF - Date objects ,timestamp to date
UDAF - aggregate function - bunch of rows and one value result
UDTF - table generating function - take one row and give multiple outputs

```
select * from yelp_review_hdfs_partioned where yr='2017'

// check the size of review text
select text,size(split(text,' ')) as words_count
from yelp_review_hdfs_partioned where text is not null limit 10 ;
```

If you spitted from date value and portioned using year then the above query can run for particular year and retrieval will be faster.

```
// sentences UDF
select explode(sentences(lower(text))) as words_length
from yelp_review_hdfs_partioned where text is not null limit 10 ;

-- N Grams of length 2 , list top 6 words of length 2

select explode(ngrams(sentences(lower(text)),2,6)) as words_length_of_2
from yelp_review_hdfs_partioned;.
```

Thank You !

Additional Practice

1Additional NYC parking assignments available

https://data.cityofnewyork.us/browse?q=parking+tickets

Data File:

https://data.cityofnewyork.us/City-Government/Parking-Violations-Issued-Fiscal-Year-2022/pvqr-7yc4

Data Dictionary:

https://data.cityofnewyork.us/City-Government/Parking-Violations-Issued-Fiscal-Year-2017/2bnn-yakx

Useful Links :

SUB Queries in SELECT clause Help document :

https://cwiki.apache.org/confluence/display/Hive/Subqueries+in+SELECT

SUB Queries in FROM clause Help document :

https://cwiki.apache.org/confluence/display/Hive/LanguageManual+SubQueries

Joins

Link :
https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Joins

Built-in Functions

To Know more:
https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF#LanguageManualUDF-Built-inFunctions