Part IV: Hive (remove header from csv before copying file from local Downloads folder to your Linux file system, data is loaded from hdfs to hive)

.....

1) sudo service ssh start **(run this command in the beginning just in case)**

2) create database wqd7007;

3) show databases;

4) hdfs dfs -mkdir /user/hdfs/batting

5) hdfs dfs -put Batting.csv /user/hdfs/batting/

6) hdfs dfs -ls /user/hdfs/batting/

7) use wqd7007;

8) CREATE EXTERNAL TABLE IF NOT EXISTS batting(

playerID STRING, yearID INT, stint INT, teamID STRING, lgID STRING, G INT, G_batting INT,

AB INT, R INT, H INT, B2 INT, B3 INT, HR INT, RBI INT, SB INT, CS INT, BB INT, SO INT, IBB INT,

HBP INT, SH INT, SF INT, GIDP INT, G_old INT)

COMMENT 'Batting stats'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

LOCATION 'hdfs://localhost:9000/user/hdfs/batting';

9) CREATE TABLE dummycars2(

Car STRING, MPG Double, Cylinders INT, Displacement DOUBLE, Horsepower DOUBLE, Weight DOUBLE, Acceleration DOUBLE, Model INT, Origin STRING)

COMMENT 'Dummy Car Details 2' **(it's optional to add a comment)**

CLUSTERED BY (Origin) into 3 buckets

ROW FORMAT DELIMITED

FIELDS TERMINATED BY '\073' LINES TERMINATED BY '\n' **(\073 is semicolon, ; )**

STORED AS orc tblproperties('transactional'='true'); **(this creates an ACID table which allows for UPDATE and DELETE)**

10) CREATE EXTERNAL TABLE IF NOT EXISTS batting3(

playerID STRING, yearID INT, stint INT, teamID STRING, lgID STRING, G INT, G_batting INT,

AB INT, R INT, H INT, B2 INT, B3 INT, HR INT, RBI INT, SB INT, CS INT, BB INT, SO INT, IBB INT,

HBP INT, SH INT, SF INT, GIDP INT, G_old INT)

COMMENT 'Batting Details 3'

CLUSTERED BY (lgID) into 3 buckets

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS orc

LOCATION 'hdfs://localhost:9000/user/hdfs/batting'

tblproperties('transactional'='true'); **(this did not work unfortunately)**

11) SET hive.txn.manager=org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;

SET hive.support.concurrency=true;

SET hive.enforce.bucketing=true;

SET hive.exec.dynamic.partition.mode=nonstrict;

SET hive.compactor.initiator.on=true;

SET hive.compactor.worker.threads=1; **(run these 5 lines after creating ACID table but before INSERT)**

11) INSERT INTO TABLE dummycars2 VALUES ('Chevrolet Chevelle Malibu',18.0,8,307.0,130.0,3504.,12.0,70,'US');

12) select * from batting;

13) select * from batting limit 5; **(select first 5 rows from batting table)**

14) select distinct teamid from batting; **(select unique values in teamid column)**

15) SELECT sum(R) FROM batting; **(sum of R column)**

16) SELECT sum(R) as sum_of_r FROM batting; **(sum of R column with alias, but the alias is useful only if there are follow-up operations)**

17) SELECT AVG(R) FROM batting; **(average of R column)**

18) SELECT ROUND(AVG(R),2) FROM batting; **(average of R column, rounded to 2 decimal places)**

19) SELECT COUNT(*) FROM batting; **(count total number of rows in batting table)**

20) SELECT COUNT(AB) FROM batting; **(count number of non-null/empty rows in AB column)**

21) SELECT SUM(CASE WHEN (AB IS NULL OR AB = '') THEN 1 ELSE 0 END) FROM batting; **(count the number of null/empty rows in AB column)**

22) SELECT COUNT(DISTINCT teamid) FROM batting; **(count the number of unique values in teamid column)**

23) SELECT AB

FROM batting

ORDER BY AB DESC

LIMIT 5; **(select top 5 highest values in AB column)**

24) SELECT G, G_batting

FROM batting

ORDER BY G DESC, G_batting ASC

LIMIT 5; **(select 5 records with primary descending sorting in G column, then for each values in G column, secondary ascending sorting is performed for G_batting column)**

25) SELECT teamid, COUNT(*)

FROM batting

GROUP BY teamid; **(count the number of non null/empty values for each teamid. GROUP BY often paired with aggregate functions such as COUNT,SUM,AVG,MAX,MIN, stddev, etc.)**

26) SELECT teamid, AVG(G)

FROM batting

GROUP BY teamid; **(count the average G values for each team id)**

27) SELECT yearID, max(R) FROM batting GROUP BY yearID;

28) SELECT a.yearID, a.playerID, a.R FROM batting a

JOIN (SELECT yearID year_ID, max(R) max_r FROM batting GROUP BY yearID) b

ON (a.yearID = b.year_ID AND a.R = b.max_r); **(this JOIN refers to an INNER JOIN, where it selects only matching rows in both table 1 and table 2)**

29) SELECT yearID, playerID

FROM batting

WHERE G > 150

UNION

SELECT yearID, playerID

FROM batting

WHERE B2 > 35; **(union will join two tables without duplicating the rows. Make sure the columns chosen in both tables are the same)**

30) SELECT stddev(R) FROM batting; **(standard deviation of R column)**

31) SELECT yearID, stddev(R) FROM batting GROUP BY yearID;

32) SELECT min(R) FROM batting; **(minimum of R column)**

33) SELECT max(R) FROM batting; **(maximum of R column)**

34) UPDATE dummycars2

SET horsepower = 120

WHERE horsepower = 150; **(update command, only on ACID tables)**

35) DELETE FROM dummycars2

WHERE weight = 3433; **(delete command, only on ACID tables)**

36) SELECT * FROM dummycars2

WHERE Displacement < 320 AND Origin = 'US';

37) SET hivevar:target_horsepower=150;

SET hivevar:updated_horsepower=120;

38) INSERT OVERWRITE TABLE dummycars

SELECT Car, mpg, Cylinders, Displacement,

CASE

WHEN Horsepower = ${hivevar:target_horsepower}

THEN ${hivevar:updated_horsepower}

ELSE Horsepower

END AS Horsepower,

Weight, Acceleration, Model, Origin

FROM dummycars; **(non-ACID table way of doing update, can refer to chatgpt)**

39) SET hivevar:target_weight=3433;

40) INSERT OVERWRITE TABLE dummycars

SELECT * FROM dummycars

WHERE Weight <> ${hivevar:target_weight}; **(non-ACID table way of doing delete, can refer to chatgpt)**

41) exit; **(exit hive shell)**