**HIVE Assignment**

Connect the MapR Sandbox via Putty.

Using Hive Command Line Interactive (CLI).

|  |
| --- |
| hive |

Press Enter Key and you will Enter into the HIVE CLI.

1. Create database ‘inueron\_db’ if it doesn’t exist

|  |
| --- |
| CREATE DATABASE IF NOT EXISTS ineuron\_db ; |

1. In Hive Command Line Interface, show Current database name.

|  |
| --- |
| set hive.cli.print.current.db=true; |

Refer - <https://stackoverflow.com/questions/17986436/how-to-identify-which-database-the-user-is-using-in-hive-cli>

1. Change the database to inueron\_db

|  |
| --- |
| USE ineuron\_db; |

1. List all the database available

|  |
| --- |
| SHOW DATABASES; |

Filters on database

|  |
| --- |
| SHOW DATABASE LIKE ‘i.\*’ ; |

Graphical user interface, text, application

Description automatically generated

1. Show the HDFS File Path of Database

|  |
| --- |
| DESCRIBE DATABASE EXTENDED ineuron\_db; |

1. Drop the database

|  |
| --- |
| DROP DATABASE ineuron\_db; |

Drop database with Cascade option

|  |
| --- |
| DROP DATABASE IF EXISTS ineuron\_db CASCADE; |

<https://www.tutorialspoint.com/hive/hive_drop_database.htm>

1. Recreate the database and save datafile to a specific location.

|  |
| --- |
| CREATE DATABASE ineuron\_db LOCATION '/user/ineuron/mydb'; |

1. Show the HDFS File Path of Database

|  |
| --- |
| DESCRIBE DATABASE ineuron\_db; |

Text

Description automatically generated

1. Create Table ‘emp\_details1’ with 4 columns and respective data type.

|  |
| --- |
| create table emp\_details1  (  emp\_name int,  unit string,  exp int,  location string  )  row format delimited  fields terminated by ','; |

1. Describe the table ‘emp\_details1’

|  |
| --- |
| DESCRIBE emp\_details1; |



1. Describe Storage format for the table ‘emp\_details1’

|  |
| --- |
| DESCRIBE FORMATTED emp\_details1; |

1. List all the tables having name like ‘emp’.

|  |
| --- |
| SHOW TABLES LIKE '\*emp\*'; |

Table

Description automatically generated with low confidence

1. Create a local file data - Open new Putty session

|  |
| --- |
| mkdir -p hive |
| vi hive/emp\_details.txt |
| Press i and Copy paste following data |
| abc,ops,10,blr |
| Press Esc and :wq! Press enter |

Text

Description automatically generated

1. In HIVE CLI, Load data from local File System into New table and Verify data load

|  |
| --- |
| create table if not exists emp\_details  (  emp\_name string,  unit string,  exp int,  location string  )  row format delimited  fields terminated by ','; |
| LOAD DATA LOCAL INPATH '/home/mapr/hive/emp\_details.txt' INTO TABLE emp\_details; |
| select count(0) from emp\_details ; |

Text, application, table

Description automatically generated

1. In HIVE CLI - Export the table into the Local file System

|  |
| --- |
| INSERT OVERWRITE LOCAL DIRECTORY '/home/mapr/hive/output'  ROW FORMAT DELIMITED  FIELDS TERMINATED BY '|'  SELECT \* FROM emp\_details; |

Check the exported file into the Local System – In the Putty Session

|  |
| --- |
| cat /home/mapr/hive/output/00000\_0 |



1. Load Data from HDFS to HIVE Table

Step i: Load data into HDFS File System via Local File System.

In Putty Session,

|  |
| --- |
| hadoop fs -mkdir -p /user/ineuron/hive |
| hadoop fs -ls -R /user/ineuron/ |
| hadoop fs -put /home/mapr/hive/emp\_details.txt /user/ineuron/hive |
| hadoop fs -ls -R /user/ineuron/ |



Step ii: Create External table and Verify the Count.

|  |
| --- |
| create external table emp\_details\_ext  (  emp\_name string,  unit string,  exp int,  location string  )  row format delimited  fields terminated by ','  location '/user/ineuron/hive/'; |
| select count(0) from emp\_details\_ext ; |

Table

Description automatically generated with medium confidence

1. HIVE – Complex Data Type Table

Step i: Save the complex\_data\_type.txt to the local file system. In Putty

|  |
| --- |
| cd /home/mapr/hive |
| vi complex\_data\_type.txt |
| emp\_id name location skill\_set  101 first:Amit,last:Mishra bbsr,751024 Hadoop,Hive  102 first:Aditya,last:Kulkarni bnglr,123412 Hadoop,Hive,Oracle |

Step ii: Create table of complex\_data\_type, Load the local file system and query it.

|  |
| --- |
| create external table complex\_data\_type\_new  (  emp\_id int,  name map<string, string>,  location struct<city:string, pin:int>,  skill\_set array<string>  )  row format delimited fields terminated by '\t'  collection items terminated by ','  map keys terminated by ':'  LOCATION '/user/ineuron/hive'  tblproperties ("skip.header.line.count"="1"); |
| LOAD DATA LOCAL INPATH '/home/mapr/hive/complex\_data\_type.txt'  OVERWRITE INTO TABLE complex\_data\_type\_new; |
| SELECT emp\_id, name['first'], name['last'],location.city,skill\_set[0],skill\_set[1] FROM complex\_data\_type\_new; |

Text, letter

Description automatically generated

1. HIVE SQL
2. Select all columns from table ‘emp\_details’ ;

|  |
| --- |
| SELECT \* FROM emp\_details; |

1. View only single column emp\_name from table ‘emp\_details’

|  |
| --- |
| SELECT emp\_name FROM emp\_details; |
| SELECT e.emp\_name FROM emp\_details e; |

1. Records Details of Employee having experience of more than 2 / 2+ Years

|  |
| --- |
| SELECT \* FROM emp\_details WHERE exp >= 2; |

1. Records details of Employee having unit like ‘op’

|  |
| --- |
| SELECT \* FROM emp\_details WHERE unit like "%op%"; |

1. Finding out details of employees whose name contains either 'mi' or 'ni'.

|  |
| --- |
| SELECT \* FROM emp\_details WHERE emp\_name RLIKE '.\*(mi|ni).\*'; |

1. No of Employees by Location

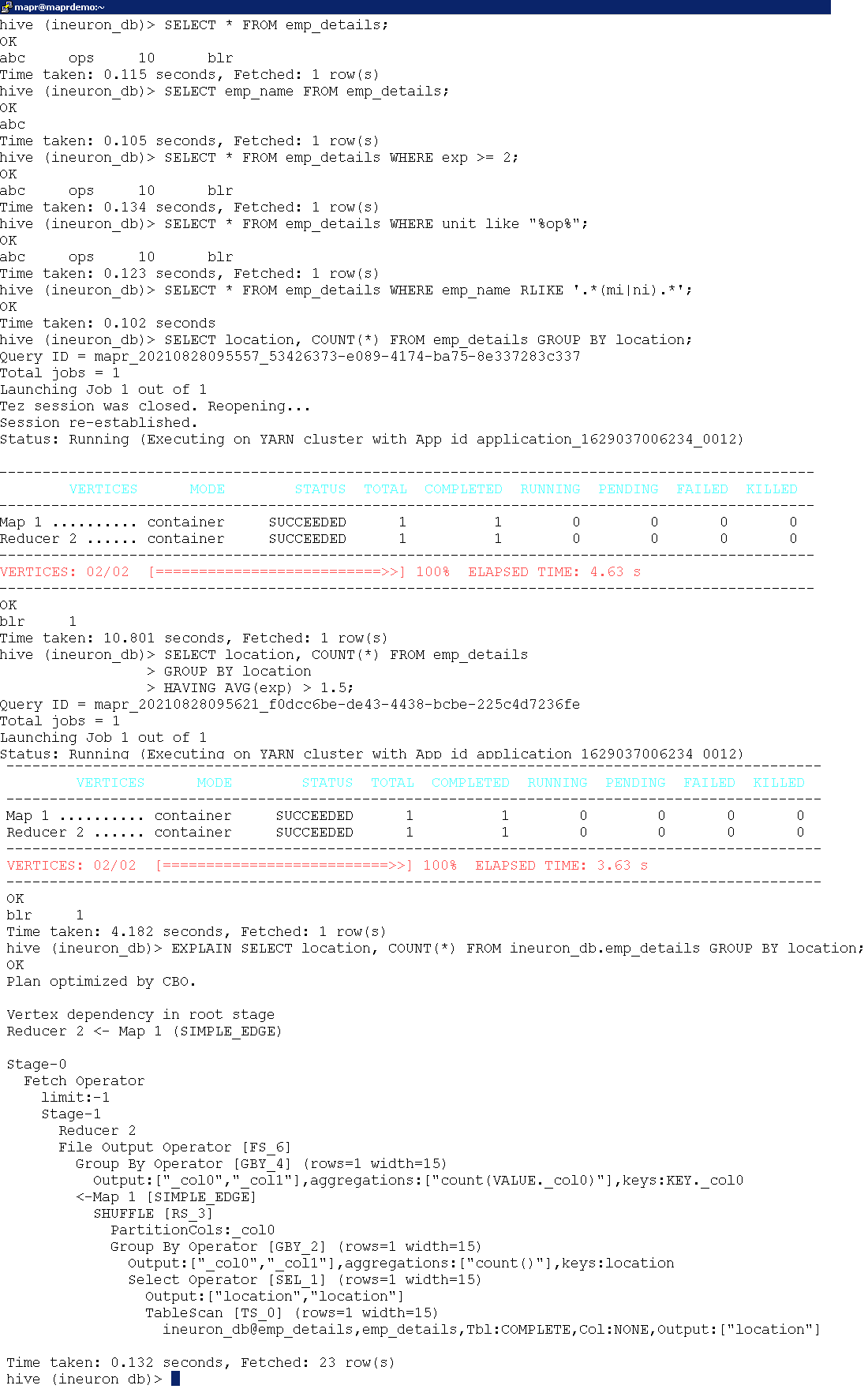
|  |
| --- |
| SELECT location, COUNT(\*) FROM emp\_details GROUP BY location; |

1. Getting number of employees present in the locations where average experience of employees is greater than 1.5 years

|  |
| --- |
| SELECT location, COUNT(\*) FROM emp\_details  GROUP BY location  HAVING AVG(exp) > 1.5; |

1. Explain Plan of Hive SQL

|  |
| --- |
| EXPLAIN SELECT location, COUNT(\*) FROM ineuron\_db.emp\_details GROUP BY location; |



1. CSV Ser De Storage Table

Step i: Create csv on local file system using Putty session

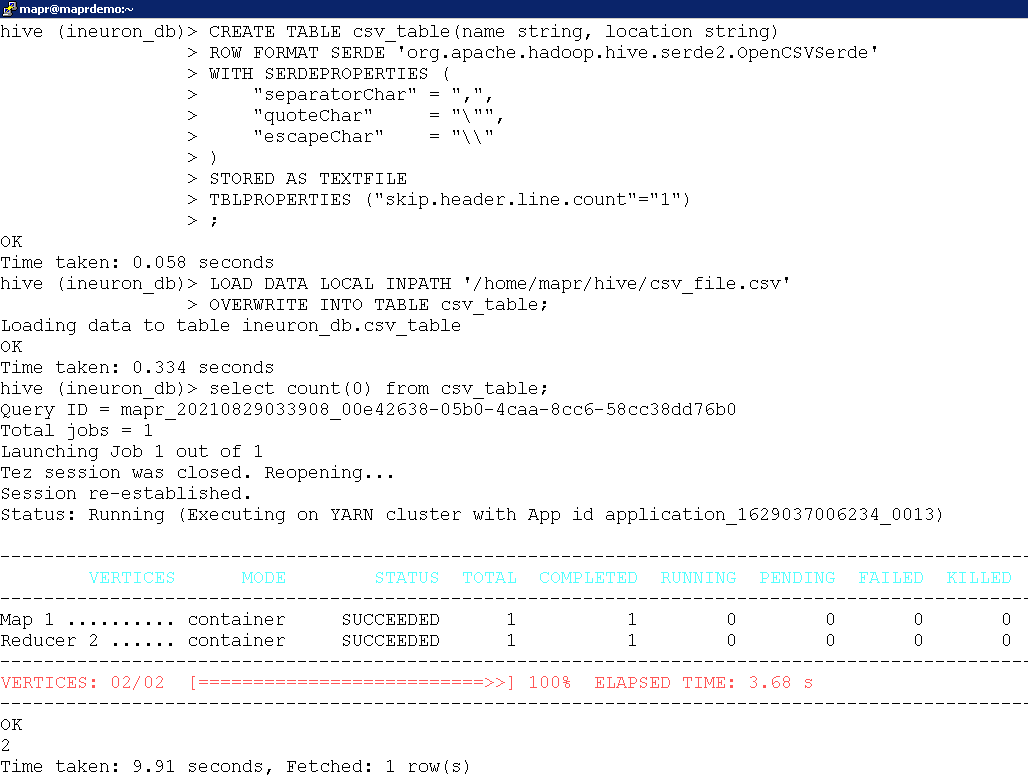
|  |
| --- |
| cd /home/mapr/hive/ |
| vi csv\_file.csv |
| Press i and Copy paste following data |
| name,location  Amit, "BBSR,India"  Sumit, "MUM,India" |
| Press Esc and :wq! And Press Enter |
| cat /home/mapr/hive/csv\_file.csv |

Text

Description automatically generated

Step ii: In HIVE CLI, create table, Load the data from Local file system, Check count of table and check storage of table.

|  |
| --- |
| CREATE TABLE csv\_table(name string, location string)  ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'  WITH SERDEPROPERTIES (  "separatorChar" = ",",  "quoteChar" = "\"",  "escapeChar" = "\\"  )  STORED AS TEXTFILE  TBLPROPERTIES ("skip.header.line.count"="1")  ; |
| LOAD DATA LOCAL INPATH '/home/mapr/hive/csv\_file.csv'  OVERWRITE INTO TABLE csv\_table; |
| select count(0) from csv\_table; |
| DESCRIBE FORMATTED csv\_table; |





1. REGEX Ser De Storage Format Table

Step i: Save sample file on Local File System using Putty

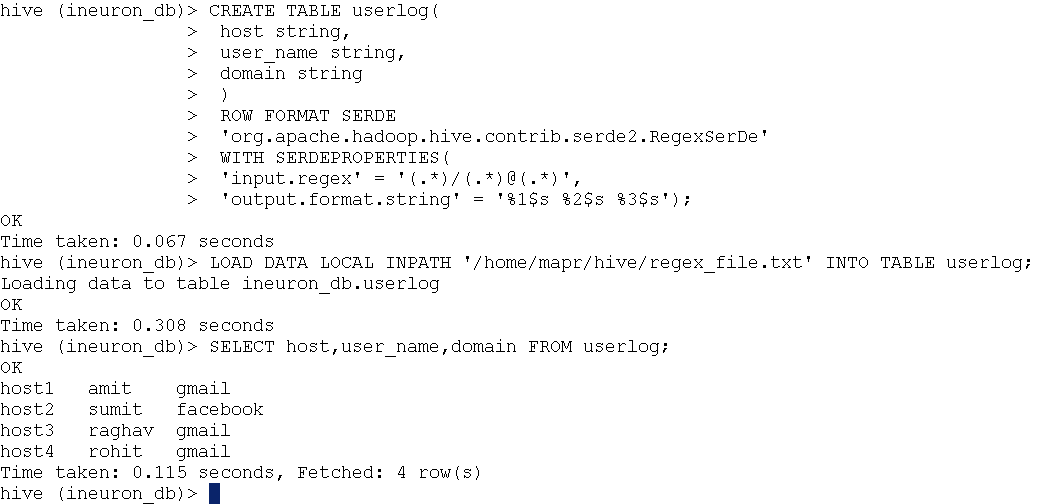
|  |
| --- |
| cd /home/mapr/hive/ |
| vi regex\_file.txt |
| Press i and copy paste following lines |
| host1/amit@gmail  host2/sumit@facebook  host3/raghav@gmail  host4/rohit@gmail |
| Press Esc and :wq! And Press Enter |
| cat /home/mapr/hive/regex\_file.txt |

Text

Description automatically generated

Step ii: Create RegEx Storage Table, Load the data from Flat file, Check the data.

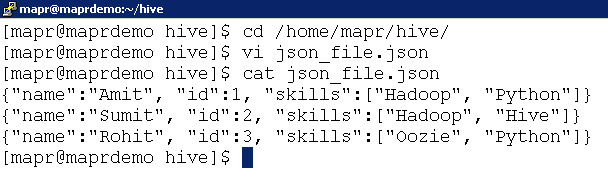
|  |
| --- |
| CREATE TABLE userlog(  host string,  user\_name string,  domain string  )  ROW FORMAT SERDE  'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'  WITH SERDEPROPERTIES(  'input.regex' = '(.\*)/(.\*)@(.\*)',  'output.format.string' = '%1$s %2$s %3$s'); |
| LOAD DATA LOCAL INPATH '/home/mapr/hive/regex\_file.txt' INTO TABLE userlog; |
| SELECT host,user\_name,domain FROM userlog; |



1. JSON Ser Storage Format Table

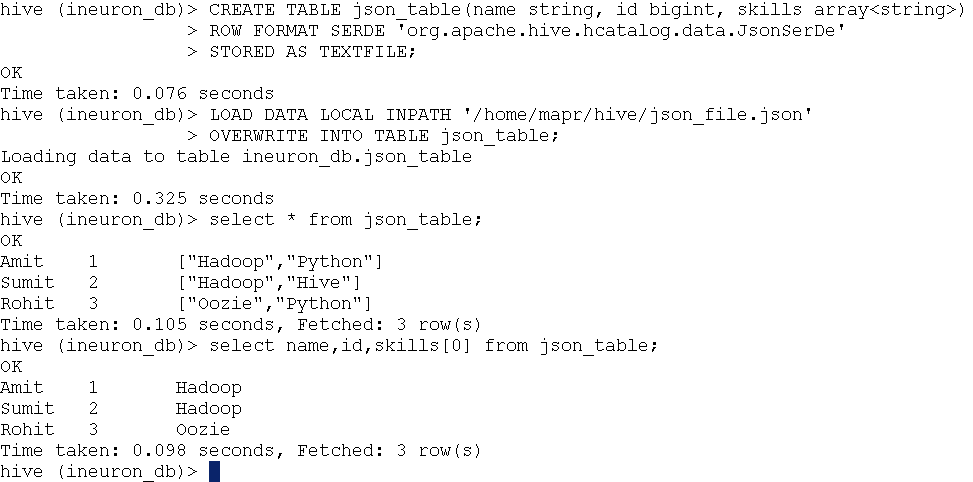
Step i: Save locally json file system using Putty session

|  |
| --- |
| cd /home/mapr/hive/ |
| vi json\_file.json |
| Press i and copy paste following contents |
| {"name":"Amit", "id":1, "skills":["Hadoop", "Python"]}  {"name":"Sumit", "id":2, "skills":["Hadoop", "Hive"]}  {"name":"Rohit", "id":3, "skills":["Oozie", "Python"]} |
| Press Esc and :wq! And Press enter |
| cat /home/mapr/hive/json\_file.json |



Step ii: In HIVE CLI, Create Table with Json Ser de, Load data and access the data

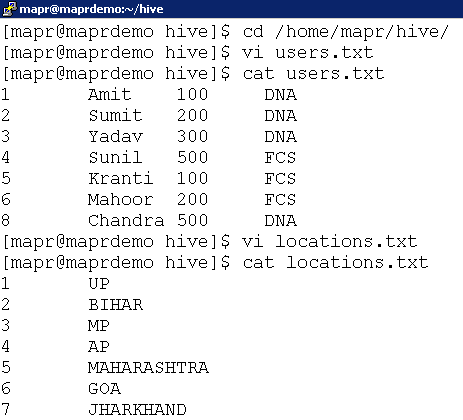
|  |
| --- |
| CREATE TABLE json\_table(name string, id bigint, skills array<string>)  ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'  STORED AS TEXTFILE; |
| LOAD DATA LOCAL INPATH '/home/mapr/hive/json\_file.json'  OVERWRITE INTO TABLE json\_table; |
| select \* from json\_table; |
| select name,id,skills[0] from json\_table; |



1. Bucketing

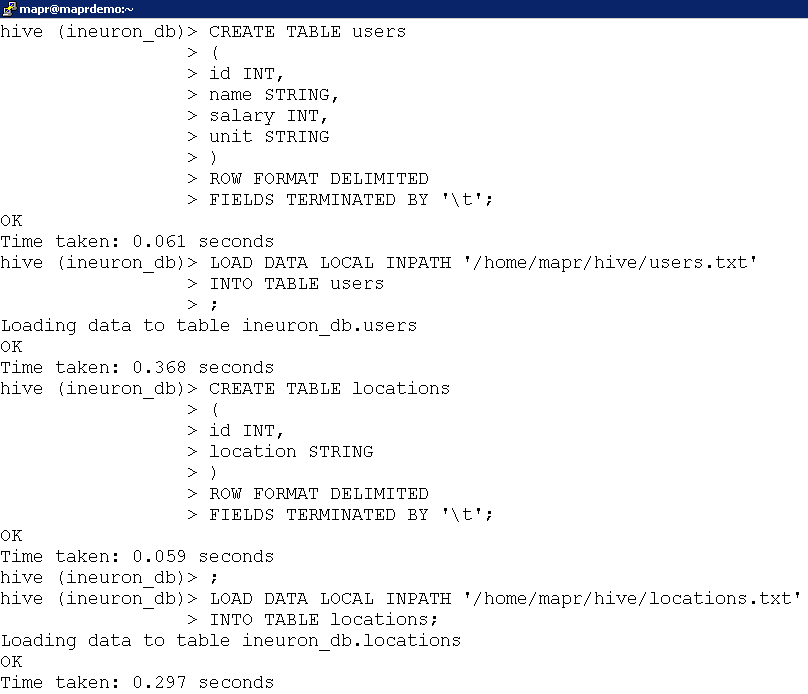
Step i: Load 2 data files user.txt and locations.txt on Local File System using Putty Session

|  |
| --- |
| cd /home/mapr/hive/ |
| vi users.txt |
| 1 Amit 100 DNA  2 Sumit 200 DNA  3 Yadav 300 DNA  4 Sunil 500 FCS  5 Kranti 100 FCS  6 Mahoor 200 FCS  8 Chandra 500 DNA |
| cat user.txt |
| vi locations.txt |
| 1 UP  2 BIHAR  3 MP  4 AP  5 MAHARASHTRA  6 GOA  7 JHARKHAND |
| cat locations.txt |



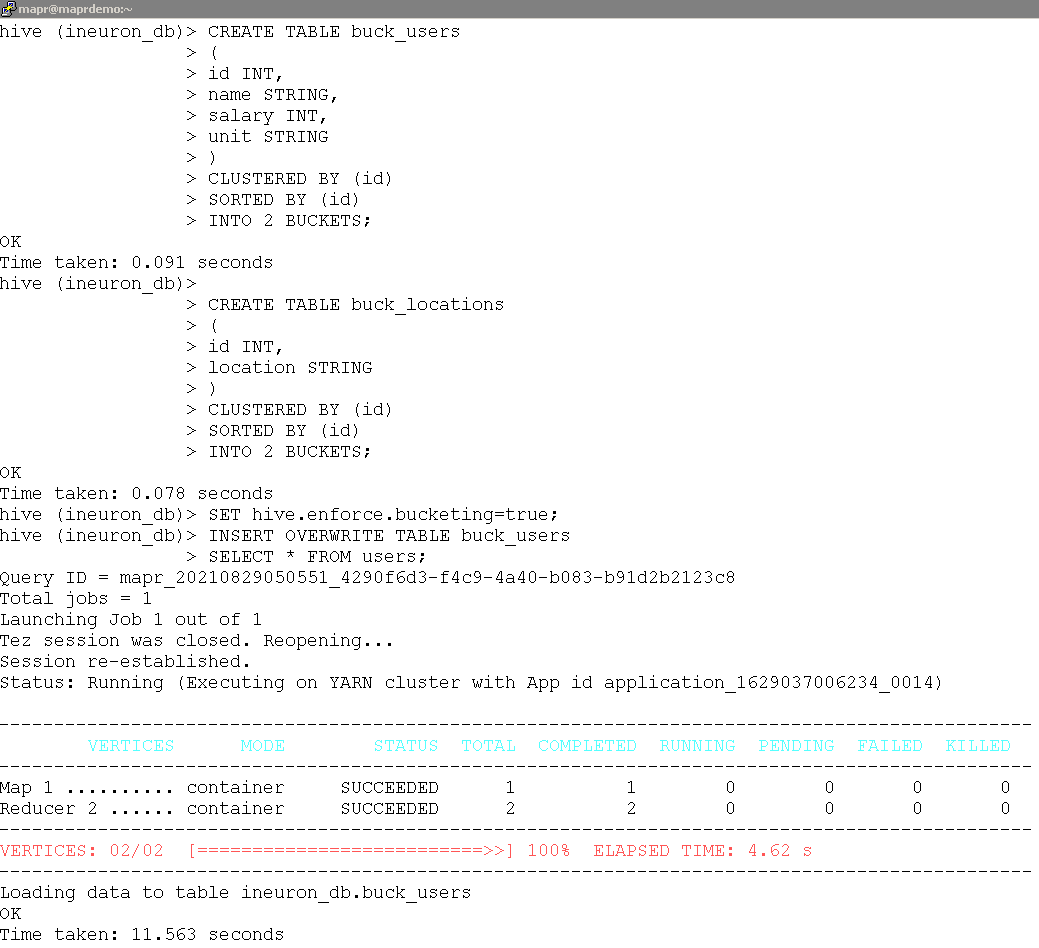
Step ii: In HIVE CLI, create two tables users and locations and load the data

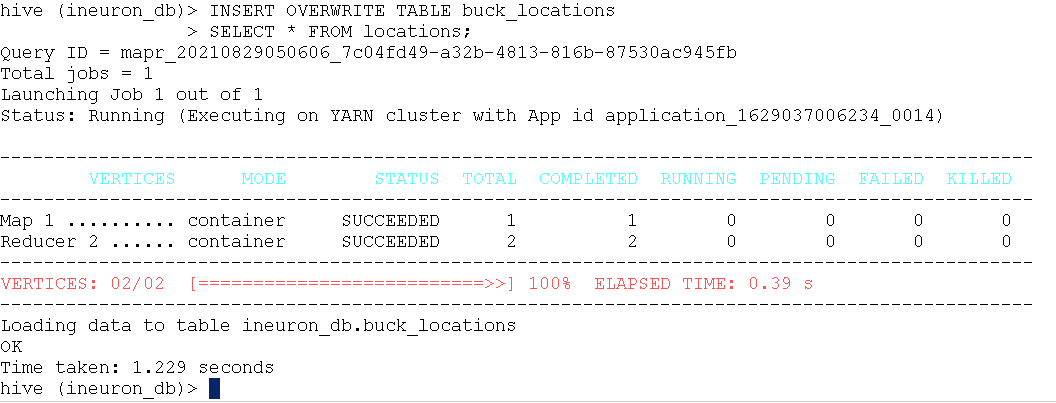
|  |
| --- |
| CREATE TABLE users  (  id INT,  name STRING,  salary INT,  unit STRING  )  ROW FORMAT DELIMITED  FIELDS TERMINATED BY '\t'; |
| LOAD DATA LOCAL INPATH '/home/mapr/hive/users.txt'  INTO TABLE users; |
| CREATE TABLE locations  (  id INT,  location STRING  )  ROW FORMAT DELIMITED  FIELDS TERMINATED BY '\t'; |
| LOAD DATA LOCAL INPATH '/home/mapr/hive/locations.txt'  INTO TABLE locations; |



Step iii: Create table with bucket i.e buck\_users and buck\_locations and load the data from existing tables user and locations and enforce bucketing.

|  |
| --- |
| CREATE TABLE buck\_users  (  id INT,  name STRING,  salary INT,  unit STRING  )  CLUSTERED BY (id)  SORTED BY (id)  INTO 2 BUCKETS; |
| CREATE TABLE buck\_locations  (  id INT,  location STRING  )  CLUSTERED BY (id)  SORTED BY (id)  INTO 2 BUCKETS; |
| SET hive.enforce.bucketing=true; |
| INSERT OVERWRITE TABLE buck\_users  SELECT \* FROM users; |
| INSERT OVERWRITE TABLE buck\_locations  SELECT \* FROM locations; |





Step iv: Verify 2 buckets / 2 files of buck\_users and buck\_locations are created into the HIVE Storage locations

|  |
| --- |
| hadoop fs -ls -R /user/ineuron/mydb/ |
| hadoop fs -ls -R /user/ineuron/mydb/buck\_locations |
| hadoop fs -ls -R /user/ineuron/mydb/buck\_users |

Text

Description automatically generated

1. Partitioning

Step i: Load the data file into the Local File System using Putty Session

|  |
| --- |
| cd /home/mapr/hive |
| vi emp\_details2.txt |
| ABC,ops,10,BBSR  DEF,ops,5,BBSR  GHI,ops,2,YSR  PQR,ops,7,MDR  XYZ,ops,1,MDR |
| cat emp\_details2.txt |

Text

Description automatically generated

Step ii: Create table emp\_details2 and load the local file system.

|  |
| --- |
| create table emp\_details2  (  emp\_name string,  unit string,  exp int,  location string  )  row format delimited  fields terminated by ','; |
| load data local inpath '/home/mapr/hive/emp\_details2.txt'  into table emp\_details2; |
| describe formatted emp\_details2; |

Text

Description automatically generated

Text

Description automatically generated with low confidence

Step iii: Create new table with partition enabled ‘emp\_details\_partitioned’ and load the data via static partition

|  |
| --- |
| create table emp\_details\_partitioned  (  emp\_name string,  unit string,  exp int  )  partitioned by (location string); |
| insert overwrite table emp\_details\_partitioned  partition(location = 'BBSR')  select emp\_name, unit, exp from emp\_details2  where location = 'BBSR'; |

Graphical user interface, text, application

Description automatically generated

Step iv: Check the HDFS File System of above 2 tables

|  |
| --- |
| hadoop fs -ls /user/ineuron/mydb/emp\_details2 |
| hadoop fs -ls /user/ineuron/mydb/emp\_details\_partitioned |

Text

Description automatically generated

Step v: Reload the data via dynamic partition.

|  |
| --- |
| set hive.exec.dynamic.partition.mode=nonstrict; |
| insert overwrite table emp\_details\_partitioned  partition (location)  select \* from emp\_details2; |

Graphical user interface, text, application

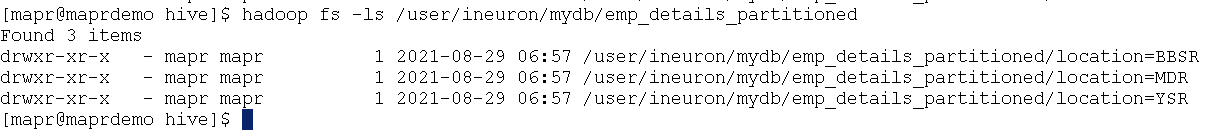
Description automatically generated

Graphical user interface, text, application

Description automatically generated

Step vi: Check the new partition on the HDFS File System

|  |
| --- |
| hadoop fs -ls /user/ineuron/mydb/emp\_details\_partitioned |



Step vii: Drop the partition, Check the data in table

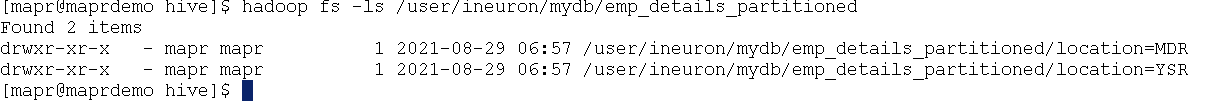
|  |
| --- |
| alter table emp\_details\_partitioned drop partition(location='BBSR'); |
| select \* From emp\_details\_partitioned where location='BBSR'; |

Text

Description automatically generated

Step viii: Recheck the HDFS File System after dropping the partition

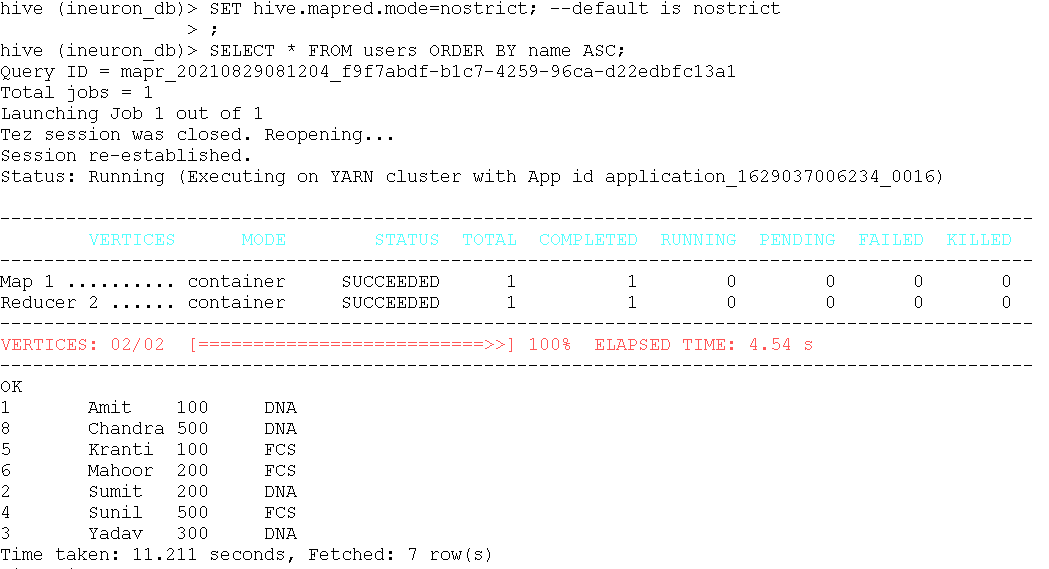
|  |
| --- |
| hadoop fs -ls /user/ineuron/mydb/emp\_details\_partitioned |



1. ORDER BY, SORT BY, DISTRIBUTE BY, CLUSTER BY

<https://stackoverflow.com/questions/13715044/hive-cluster-by-vs-order-by-vs-sort-by>

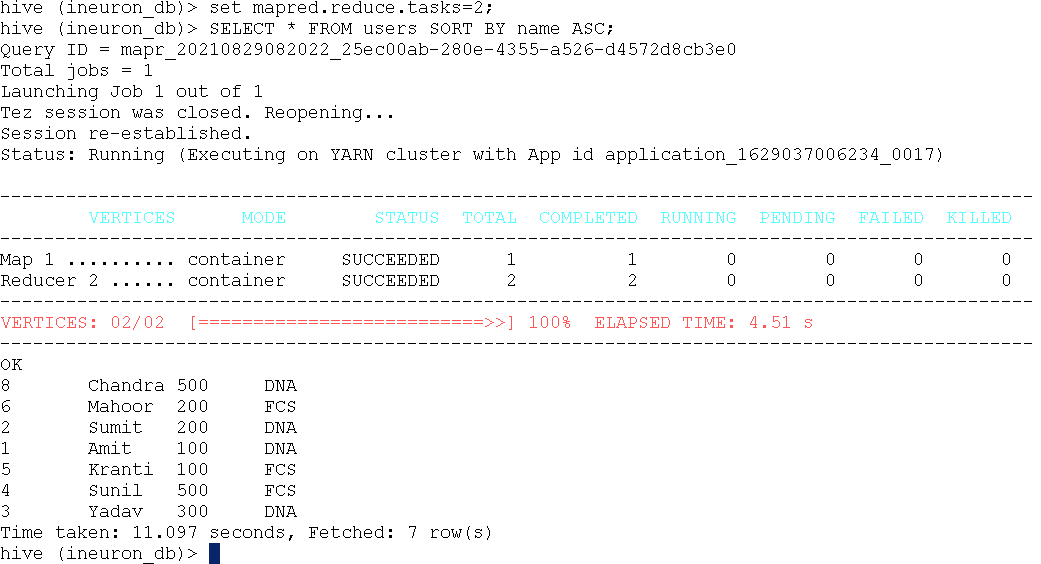
|  |
| --- |
| --default is nostrict  SET hive.mapred.mode=nostrict; |
| SELECT \* FROM users ORDER BY name ASC; |
| SELECT \* FROM users SORT BY name ASC; |



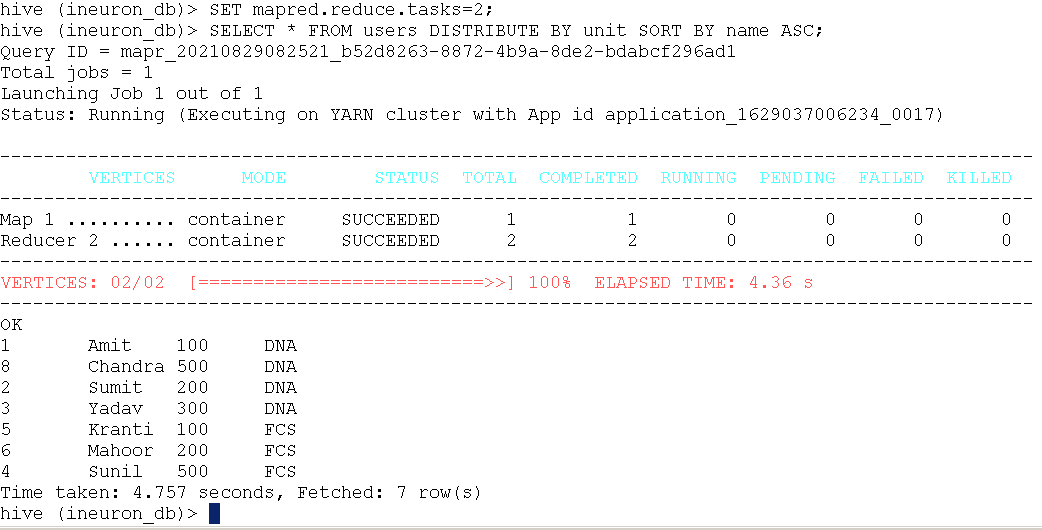
Graphical user interface, application

Description automatically generated

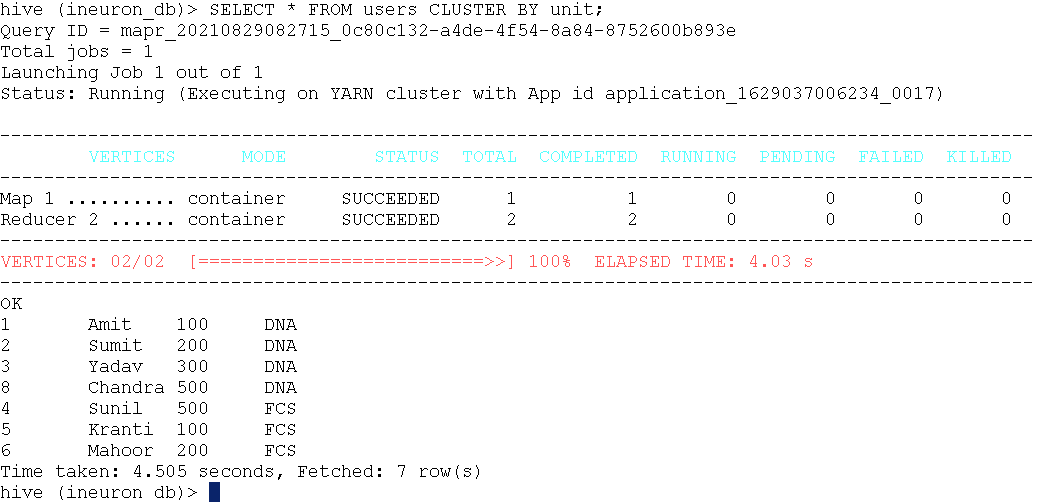
|  |
| --- |
| set mapred.reduce.tasks=2; |
| SELECT \* FROM users SORT BY name ASC; |



|  |
| --- |
| SET mapred.reduce.tasks=2; |
| SELECT \* FROM users DISTRIBUTE BY unit SORT BY name ASC; |



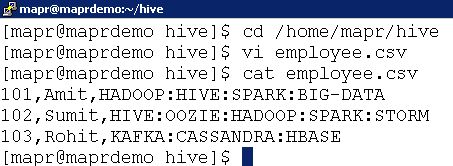
|  |
| --- |
| SELECT \* FROM users CLUSTER BY unit; |



1. Dealing with ARRAY, LATERAL VIEW, EXPLODE, SORT ARRAY, CONCAT

Step i: Load the data into Local file system

|  |
| --- |
| cd /home/mapr/hive |
| vi employee.csv |
| 101,Amit,HADOOP:HIVE:SPARK:BIG-DATA  102,Sumit,HIVE:OOZIE:HADOOP:SPARK:STORM  103,Rohit,KAFKA:CASSANDRA:HBASE |
| cat employee.csv |



Step ii: Create table, load data, try out concat, explode, lateral view queries

|  |
| --- |
| CREATE TABLE employee  (  id INT,  name STRING,  skills ARRAY<STRING>  )  ROW FORMAT DELIMITED  FIELDS TERMINATED BY ','  COLLECTION ITEMS TERMINATED BY ':'; |
| LOAD DATA LOCAL INPATH '/home/mapr/hive/employee.csv'  INTO TABLE employee; |
| SELECT  size(skills),  array\_contains(skills, 'HADOOP'),  sort\_array(skills),  concat\_ws("|", skills)  FROM employee; |
| SELECT explode(skills) AS skills FROM employee; |
| SELECT id, name, skill  FROM employee LATERAL VIEW explode(skills) skill\_set  AS skill; |

Text

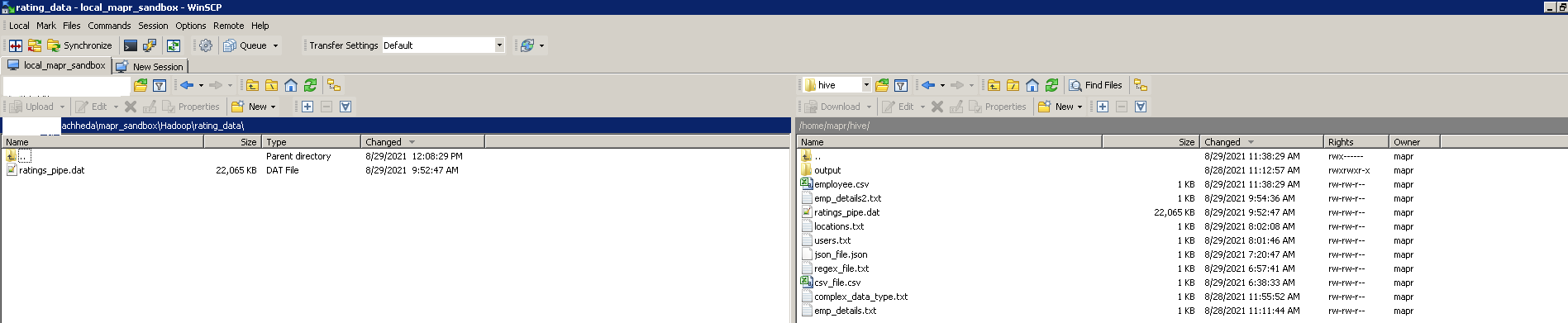
Description automatically generated

Text

Description automatically generated

1. Storage (ORC, Parquet, Sequence, RC)

Step i: Upload [ratings\_pipe.dat](https://github.com/arpitchheda/learn/blob/main/bigdata/hive/sampledata/ratings_pipe.dat) from Local machine to MapR Sandbox via tool like WinScp or FileZilla at the location /home/mapr/hive/



Step ii: Create table for different storage types and load the data

|  |
| --- |
| create table text\_table  (  c1 int,  c2 int,  c3 int,  c4 int  )  row format delimited  fields terminated by '|'; |
| load data local inpath '/home/mapr/hive/ratings\_pipe.dat'  into table text\_table; |
| create table seq\_table  (  c1 int,  c2 int,  c3 int,  c4 int  )  stored as SEQUENCEFILE; |
| create table rc\_table  (  c1 int,  c2 int,  c3 int,  c4 int  )  stored as RCFILE; |
| create table prq\_table  (  c1 int,  c2 int,  c3 int,  c4 int  )  stored as PARQUET; |
| create table orc\_table  (  c1 int,  c2 int,  c3 int,  c4 int  )  stored as ORC; |
| FROM text\_table  INSERT OVERWRITE TABLE seq\_table SELECT \*  INSERT OVERWRITE TABLE rc\_table SELECT \*  INSERT OVERWRITE TABLE prq\_table SELECT \*  INSERT OVERWRITE TABLE orc\_table SELECT \*; |

Table

Description automatically generated

Text

Description automatically generated

Step iii: View format of ORC table

|  |
| --- |
| describe formatted orc\_table; |

Table

Description automatically generated with medium confidence

Step iv: View the storage space

|  |
| --- |
| hadoop fs -ls -h -R /user/ineuron/mydb/ |

Table

Description automatically generated

Step v: Compress the tables

|  |
| --- |
| SET hive.exec.compress.output=true; |
| SET mapred.max.split.size=256000000; |
| -- block compression for sequence file  SET mapred.output.compression.type=BLOCK; |
| SET mapred.output.compression.codec=org.apache.hadoop.io.compress.SnappyCodec; |
| FROM text\_table  INSERT OVERWRITE TABLE seq\_table SELECT \*  INSERT OVERWRITE TABLE rc\_table SELECT \*  INSERT OVERWRITE TABLE prq\_table SELECT \*  INSERT OVERWRITE TABLE orc\_table SELECT \*; |

Text, application

Description automatically generated

Step vi: Recheck file size in HDFS after compression

|  |
| --- |
| hadoop fs -ls -h -R /user/ineuron/mydb/ |

Graphical user interface, table

Description automatically generated

1. HUE Editor – SQL Queries Joins

Open HUE Editor - <http://localhost:8888>

Username – mapr

Password – mapr

Select the appropriate database ‘inueron\_db’

|  |
| --- |
| SELECT \* FROM buck\_users u INNER JOIN buck\_locations l  ON u.id = l.id; |

Graphical user interface, application, table

Description automatically generated

|  |
| --- |
| SELECT \* FROM buck\_users u LEFT OUTER JOIN buck\_locations l  ON u.id = l.id; |

Graphical user interface, table

Description automatically generated

|  |
| --- |
| SELECT \* FROM buck\_users u RIGHT OUTER JOIN buck\_locations l  ON u.id = l.id; |

Table

Description automatically generated

|  |
| --- |
| SELECT \* FROM buck\_users u FULL OUTER JOIN buck\_locations l  ON u.id = l.id; |

Graphical user interface, table

Description automatically generated