Chapter 5. Big Data Analytics

Exercise Workbook

Contents

Lab 1: Start with MariaDB

In this lab, you will use DDL and DML commands from MariaDB. Actually, many RDBMS system exist, and not all commands are the same, but most of them support standard SQL. There is no problem with command compatibility. Here, we use the commands based on MariaDB.

1. Creating Database & Table

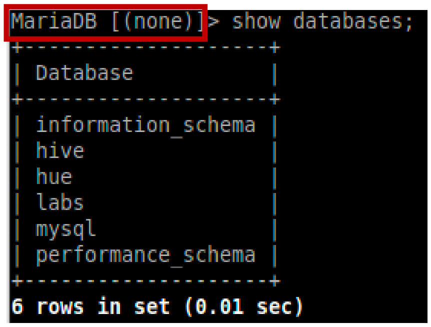
1.1. In a terminal window, log in to MariaDB and show the existing database lists.

```
$ mysql --user=student --password=student
```

Note: If you do not enter anything after the password, you will be prompted for the password:

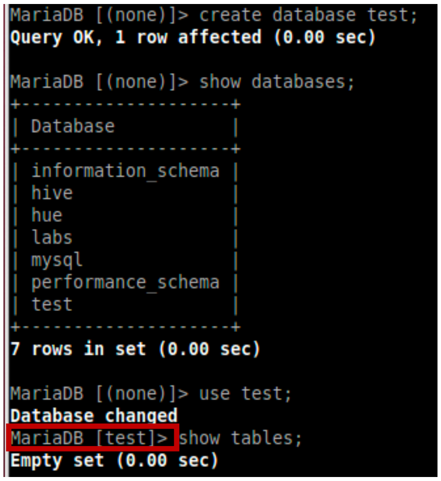
1.2. If the login is successful, the "MariaDB [(none)]>" prompt appears and a screen waiting for commands is displayed. Enter a command to check which database exists here.

```
MariaDB [(none)]> show databases;
```



```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| hive               |
| hue                |
| labs               |
| mysql              |
| performance_schema |
+--------------------+
6 rows in set (0.01 sec)
```

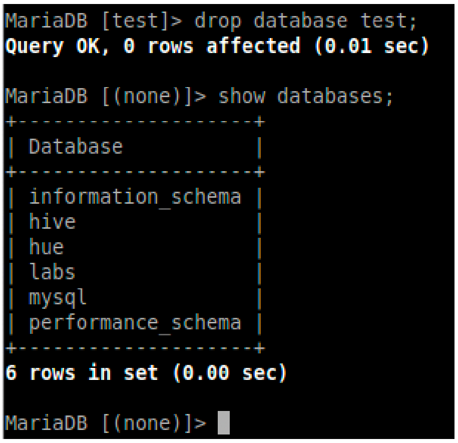
1.3. Next, enter the command to create a new test database and review the table in test DB.



```
MariaDB [(none)]> create database test;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| hive               |
| hue                |
| labs               |
| mysql              |
| performance_schema |
| test               |
+--------------------+
7 rows in set (0.00 sec)

MariaDB [(none)]> use test;
Database changed
MariaDB [test]> show tables;
Empty set (0.00 sec)
```

1.4. Drop the test database;

```
MariaDB [test]> drop database test;
```



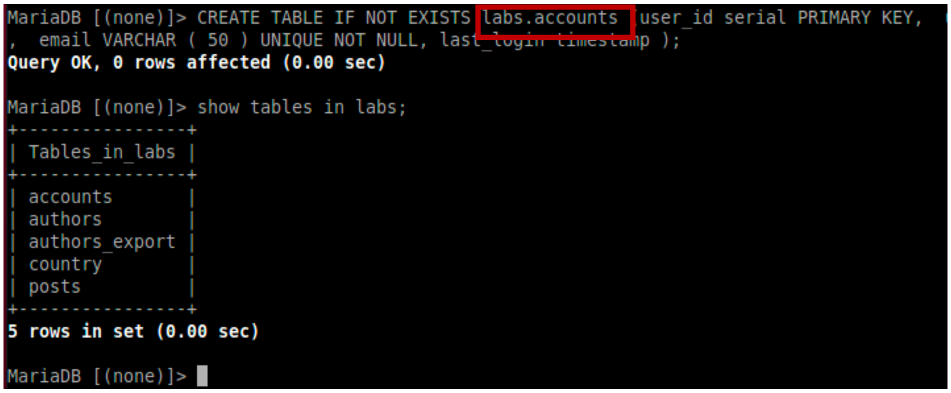
```
MariaDB [test]> drop database test;
Query OK, 0 rows affected (0.01 sec)

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| hive               |
| hue                |
| labs               |
| mysql              |
| performance_schema |
+--------------------+
6 rows in set (0.00 sec)

MariaDB [(none)]>
```

1.5. Using the labs database, create the accounts table as follows and verify the accounts table.

Column Name	Column Type	Constraints
user_id	serial	Primary key
username	varchar(30)	UNIQUE, NOT NULL
password	varchar(20)	NOT NULL
email	Varchar(50)	UNIQUE, NOT NULL
last_login	timestamp	

1.5.1. How to create directly without changing the database.

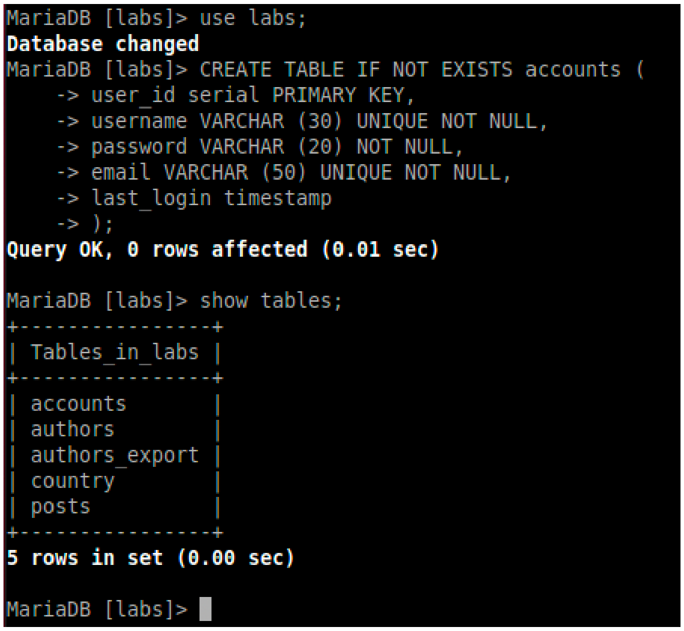
```
MariaDB [(none)]> CREATE TABLE IF NOT EXISTS labs.accounts (
                  user_id serial PRIMARY KEY,
                  username VARCHAR (30) UNIQUE NOT NULL,
                  password VARCHAR (20) NOT NULL,
                  email VARCHAR ( 50 ) UNIQUE NOT NULL,
                  last_login timestamp );
```



```
MariaDB [(none)]> CREATE TABLE IF NOT EXISTS labs.accounts (user_id serial PRIMARY KEY,
,  email VARCHAR ( 50 ) UNIQUE NOT NULL, last_login timestamp );
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> show tables in labs;
+----------------+
| Tables_in_labs |
+----------------+
| accounts       |
| authors        |
| authors_export |
| country        |
| posts          |
+----------------+
5 rows in set (0.00 sec)

MariaDB [(none)]>
```

1.5.2. How to change the database used to labs and create a table.

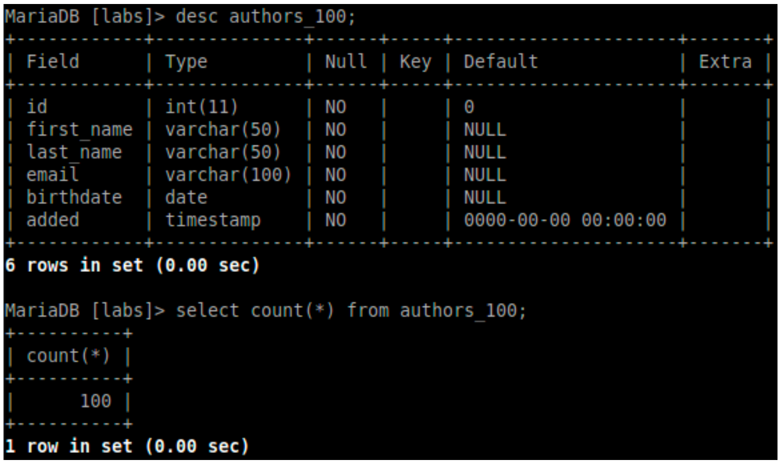
```
MariaDB [(none)]> use labs;
```



```
MariaDB [labs]> use labs;
Database changed
MariaDB [labs]> CREATE TABLE IF NOT EXISTS accounts (
    -> user_id serial PRIMARY KEY,
    -> username VARCHAR (30) UNIQUE NOT NULL,
    -> password VARCHAR (20) NOT NULL,
    -> email VARCHAR (50) UNIQUE NOT NULL,
    -> last_login timestamp
    -> );
Query OK, 0 rows affected (0.01 sec)

MariaDB [labs]> show tables;
+----------------+
| Tables_in_labs |
+----------------+
| accounts       |
| authors        |
| authors_export |
| country        |
| posts          |
+----------------+
5 rows in set (0.00 sec)

MariaDB [labs]>
```

1.6. Use CTAS(Create Table As Select) to create authors_100 that only stores 100 records from existing authors.

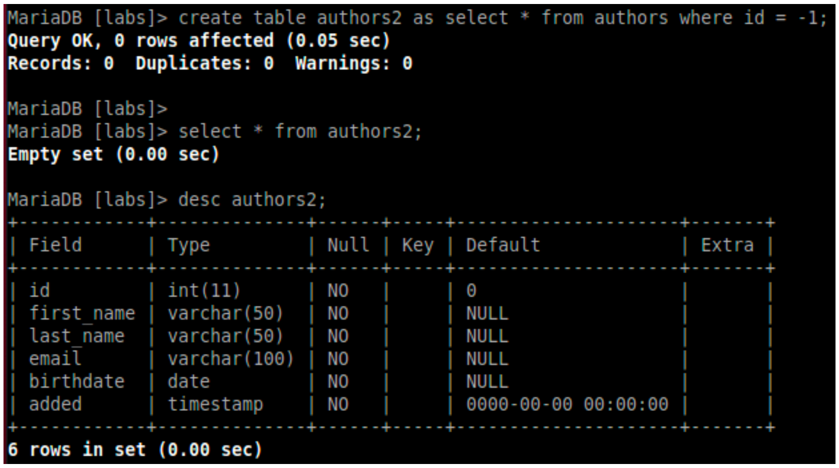
```
MariaDB [labs]> create table authors_100 as select * from authors where id <= 100;
```



```
MariaDB [labs]> desc authors_100;
+------------+--------------+------+-----+---------------------+-------+
| Field      | Type         | Null | Key | Default             | Extra |
+------------+--------------+------+-----+---------------------+-------+
| id         | int(11)      | NO   |     | 0                   |       |
| first_name | varchar(50)  | NO   |     | NULL                |       |
| last_name  | varchar(50)  | NO   |     | NULL                |       |
| email      | varchar(100) | NO   |     | NULL                |       |
| birthdate  | date         | NO   |     | NULL                |       |
| added      | timestamp    | NO   |     | 0000-00-00 00:00:00 |       |
+------------+--------------+------+-----+---------------------+-------+
6 rows in set (0.00 sec)

MariaDB [labs]> select count(*) from authors_100;
+----------+
| count(*) |
+----------+
|      100 |
+----------+
1 row in set (0.00 sec)
```

1.7. Use CTAS to create authors2 with the same schema without records.

```
MariaDB [labs]> create table authors2 as select * from authors where id = -1;
```

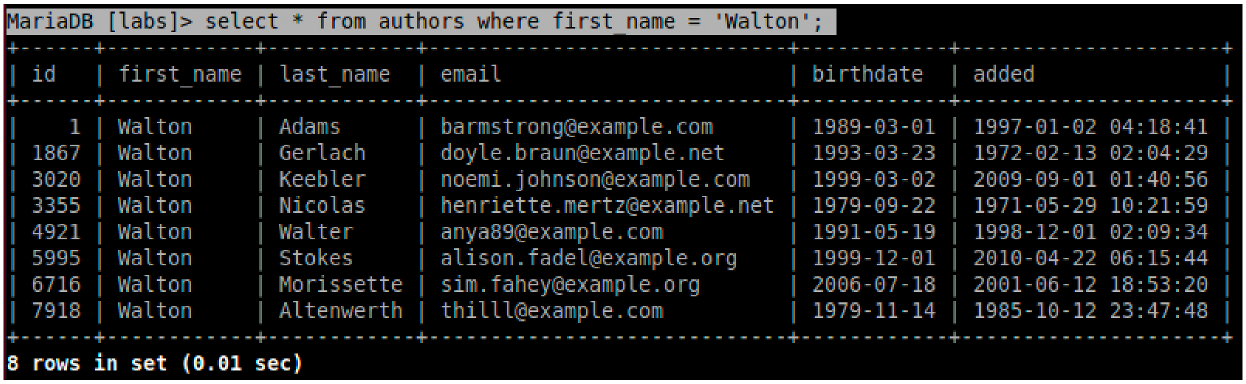


```
MariaDB [labs]> create table authors2 as select * from authors where id = -1;
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [labs]>
MariaDB [labs]> select * from authors2;
Empty set (0.00 sec)

MariaDB [labs]> desc authors2;
+------------+--------------+------+-----+---------------------+-------+
| Field      | Type         | Null | Key | Default             | Extra |
+------------+--------------+------+-----+---------------------+-------+
| id         | int(11)      | NO   |     | 0                   |       |
| first_name | varchar(50)  | NO   |     | NULL                |       |
| last_name  | varchar(50)  | NO   |     | NULL                |       |
| email      | varchar(100) | NO   |     | NULL                |       |
| birthdate  | date         | NO   |     | NULL                |       |
| added      | timestamp    | NO   |     | 0000-00-00 00:00:00 |       |
+------------+--------------+------+-----+---------------------+-------+
6 rows in set (0.00 sec)
```

1.8. Use the authors table to query records with the following conditions, and show the results.

1.8.1. Find all Walton in first name.

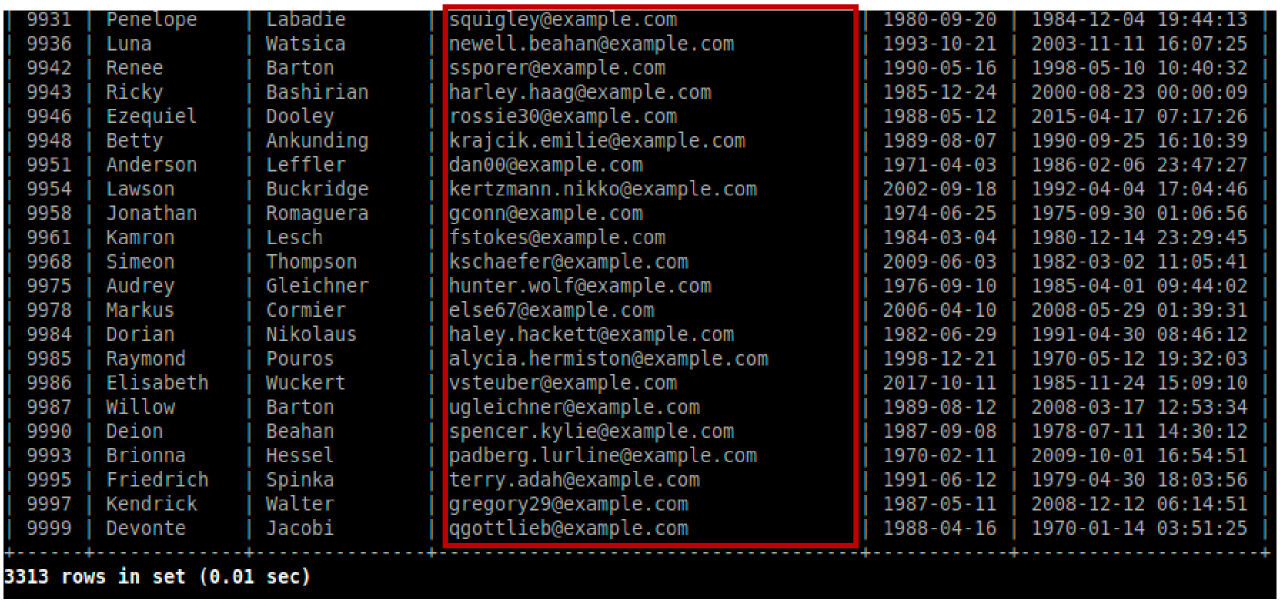
```
MariaDB [labs]> select * from authors where first_name = 'Walton';
```



```
MariaDB [labs]> select * from authors where first_name = 'Walton';
+------+------------+------------+---------------------------+------------+---------------------+
| id   | first_name | last_name  | email                     | birthdate  | added               |
+------+------------+------------+---------------------------+------------+---------------------+
|    1 | Walton     | Adams      | barmstrong@example.com    | 1989-03-01 | 1997-01-02 04:18:41 |
| 1867 | Walton     | Gerlach    | doyle.braun@example.net   | 1993-03-23 | 1972-02-13 02:04:29 |
| 3020 | Walton     | Keebler    | noemi.johnson@example.com | 1999-03-02 | 2009-09-01 01:40:56 |
| 3355 | Walton     | Nicolas    | henriette.mertz@example.net | 1979-09-22 | 1971-05-29 10:21:59 |
| 4921 | Walton     | Walter     | anya89@example.com        | 1991-05-19 | 1998-12-01 02:09:34 |
| 5995 | Walton     | Stokes     | alison.fadel@example.org  | 1999-12-01 | 2010-04-22 06:15:44 |
| 6716 | Walton     | Morissette | sim.fahey@example.org     | 2006-07-18 | 2001-06-12 18:53:20 |
| 7918 | Walton     | Altenwerth | thilll@example.com        | 1979-11-14 | 1985-10-12 23:47:48 |
+------+------------+------------+---------------------------+------------+---------------------+
8 rows in set (0.01 sec)
```

1.8.2. Find everyone who uses example.com for their email address.

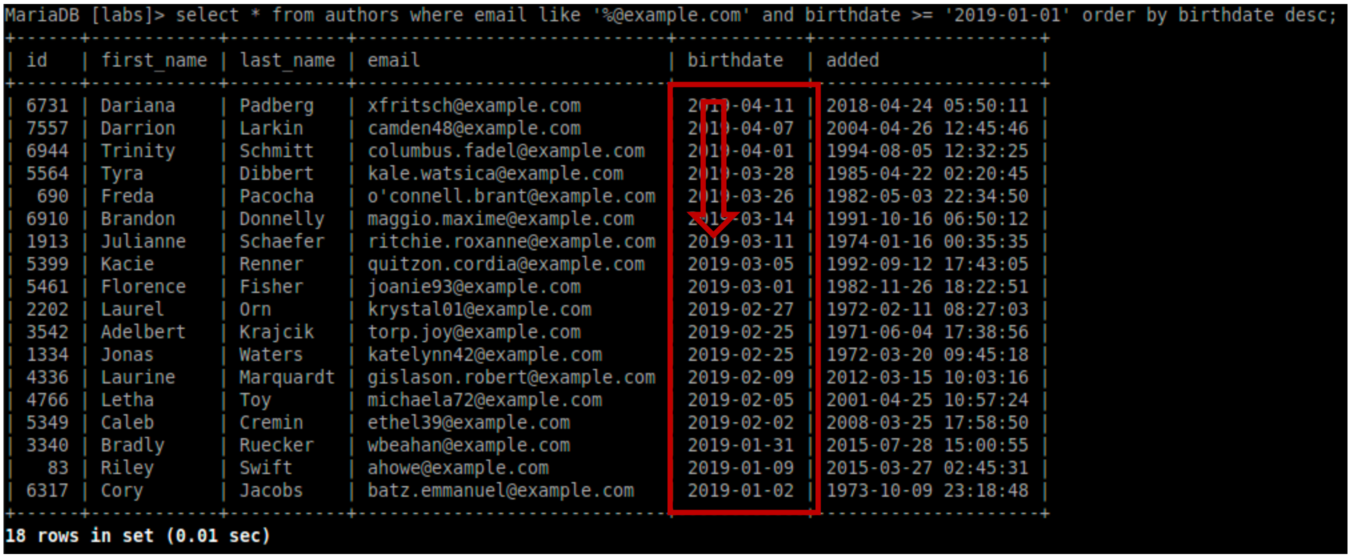
```
MariaDB [labs]> select * from authors where email like '%@example.com';
```



```
| 9931 | Penelope   | Labadie    | squigley@example.com           | 1980-09-20 | 1984-12-04 19:44:13 |
| 9936 | Luna       | Watsica    | newell.beahan@example.com      | 1993-10-21 | 2003-11-11 16:07:25 |
| 9942 | Renee      | Barton     | ssporer@example.com            | 1990-05-16 | 1998-05-10 10:40:32 |
| 9943 | Ricky      | Bashirian  | harley.haag@example.com        | 1985-12-24 | 2000-08-23 00:00:09 |
| 9946 | Ezequiel   | Dooley     | rossie30@example.com           | 1988-05-12 | 2015-04-17 07:17:26 |
| 9948 | Betty      | Ankunding  | krajcik.emilie@example.com     | 1989-08-07 | 1990-09-25 16:10:39 |
| 9951 | Anderson   | Leffler    | dan00@example.com              | 1971-04-03 | 1986-02-06 23:47:27 |
| 9954 | Lawson     | Buckridge  | kertzmann.nikko@example.com    | 2002-09-18 | 1992-04-04 17:04:46 |
| 9958 | Jonathan   | Romaguera  | gconn@example.com              | 1974-06-25 | 1975-09-30 01:06:56 |
| 9961 | Kamron     | Lesch      | fstokes@example.com            | 1984-03-04 | 1980-12-14 23:29:45 |
| 9968 | Simeon     | Thompson   | kschaefer@example.com          | 2009-06-03 | 1982-03-02 11:05:41 |
| 9975 | Audrey     | Gleichner  | hunter.wolf@example.com        | 1976-09-10 | 1985-04-01 09:44:02 |
| 9978 | Markus     | Cormier    | else67@example.com             | 2006-04-10 | 2008-05-29 01:39:31 |
| 9984 | Dorian     | Nikolaus   | haley.hackett@example.com      | 1982-06-29 | 1991-04-30 08:46:12 |
| 9985 | Raymond    | Pouros     | alycia.hermiston@example.com   | 1998-12-21 | 1970-05-12 19:32:03 |
| 9986 | Elisabeth  | Wuckert    | vsteuber@example.com           | 2017-10-11 | 1985-11-24 15:09:10 |
| 9987 | Willow     | Barton     | ugleichner@example.com         | 1989-08-12 | 2008-03-17 12:53:34 |
| 9990 | Deion      | Beahan     | spencer.kylie@example.com      | 1987-09-08 | 1978-07-11 14:30:12 |
| 9993 | Brionna    | Hessel     | padberg.lurline@example.com    | 1970-02-11 | 2009-10-01 16:54:51 |
| 9995 | Friedrich  | Spinka     | terry.adah@example.com         | 1991-06-12 | 1979-04-30 18:03:56 |
| 9997 | Kendrick   | Walter     | gregory29@example.com          | 1987-05-11 | 2008-12-12 06:14:51 |
| 9999 | Devonte    | Jacobi     | qgottlieb@example.com          | 1988-04-16 | 1970-01-14 03:51:25 |
+------+------------+------------+--------------------------------+------------+---------------------+
3313 rows in set (0.01 sec)
```

1.8.3. Among them in 1.8.2, list only those who have a birthday after 2019 in descending order.

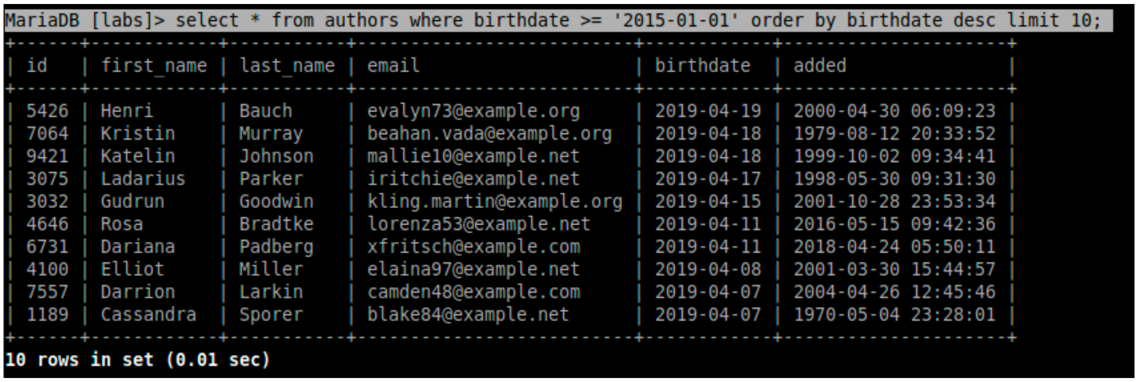
```
MariaDB [labs]> select * from authors where email like '%@example.com' and birthdate >= '2019-01-01' order by birthdate desc;
```



```
MariaDB [labs]> select * from authors where email like '%@example.com' and birthdate >= '2019-01-01' order by birthdate desc;
+------+------------+-----------+-----------------------------+------------+---------------------+
| id   | first_name | last_name | email                       | birthdate  | added               |
+------+------------+-----------+-----------------------------+------------+---------------------+
| 6731 | Dariana    | Padberg   | xfritsch@example.com        | 2019-04-11 | 2018-04-24 05:50:11 |
| 7557 | Darrion    | Larkin    | camden48@example.com        | 2019-04-07 | 2004-04-26 12:45:46 |
| 6944 | Trinity    | Schmitt   | columbus.fadel@example.com  | 2019-04-01 | 1994-08-05 12:32:25 |
| 5564 | Tyra       | Dibbert   | kale.watsica@example.com    | 2019-03-28 | 1985-04-22 02:20:45 |
|  690 | Freda      | Pacocha   | o'connell.brant@example.com | 2019-03-26 | 1982-05-03 22:34:50 |
| 6910 | Brandon    | Donnelly  | maggio.maxime@example.com   | 2019-03-14 | 1991-10-16 06:50:12 |
| 1913 | Julianne   | Schaefer  | ritchie.roxanne@example.com | 2019-03-11 | 1974-01-16 00:35:35 |
| 5399 | Kacie      | Renner    | quitzon.cordia@example.com  | 2019-03-05 | 1992-09-12 17:43:05 |
| 5461 | Florence   | Fisher    | joanie93@example.com        | 2019-03-01 | 1982-11-26 18:22:51 |
| 2202 | Laurel     | Orn       | krystal01@example.com       | 2019-02-27 | 1972-02-11 08:27:03 |
| 3542 | Adelbert   | Krajcik   | torp.joy@example.com        | 2019-02-25 | 1971-06-04 17:38:56 |
| 1334 | Jonas      | Waters    | katelynn42@example.com      | 2019-02-25 | 1972-03-20 09:45:18 |
| 4336 | Laurine    | Marquardt | gislason.robert@example.com | 2019-02-09 | 2012-03-15 10:03:16 |
| 4766 | Letha      | Toy       | michaela72@example.com      | 2019-02-05 | 2001-04-25 10:57:24 |
| 5349 | Caleb      | Cremin    | ethel39@example.com         | 2019-02-02 | 2008-03-25 17:58:50 |
| 3340 | Bradly     | Ruecker   | wbeahan@example.com         | 2019-01-31 | 2015-07-28 15:00:55 |
|   83 | Riley      | Swift     | ahowe@example.com           | 2019-01-09 | 2015-03-27 02:45:31 |
| 6317 | Cory       | Jacobs    | batz.emmanuel@example.com   | 2019-01-02 | 1973-10-09 23:18:48 |
+------+------------+-----------+-----------------------------+------------+---------------------+
18 rows in set (0.01 sec)
```

1.8.4. List only the top 10 people born after 2015.

```
MariaDB [labs]> select * from authors where birthdate >= '2015-01-01' order by
birthdate desc limit 10;
```

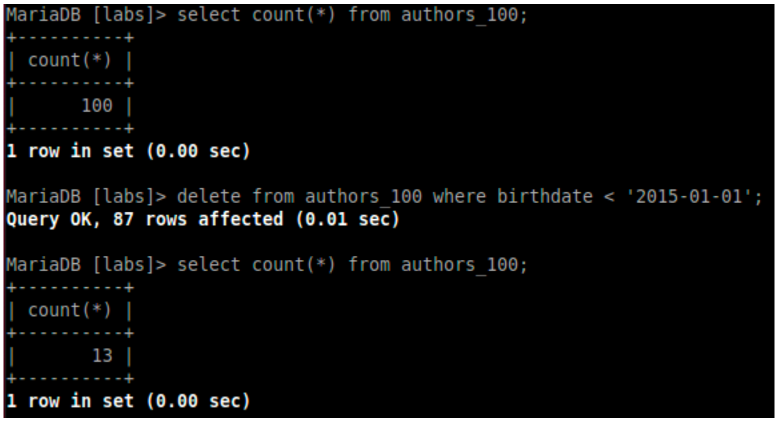


```
MariaDB [labs]> select * from authors where birthdate >= '2015-01-01' order by birthdate desc limit 10;
+------+------------+-----------+---------------------------+------------+---------------------+
| id   | first_name | last_name | email                     | birthdate  | added               |
+------+------------+-----------+---------------------------+------------+---------------------+
| 5426 | Henri      | Bauch     | evalyn73@example.org      | 2019-04-19 | 2000-04-30 06:09:23 |
| 7064 | Kristin    | Murray    | beahan.vada@example.org   | 2019-04-18 | 1979-08-12 20:33:52 |
| 9421 | Katelin    | Johnson   | mallie10@example.net      | 2019-04-18 | 1999-10-02 09:34:41 |
| 3075 | Ladarius   | Parker    | iritchie@example.net      | 2019-04-17 | 1998-05-30 09:31:30 |
| 3032 | Gudrun     | Goodwin   | kling.martin@example.org  | 2019-04-15 | 2001-10-28 23:53:34 |
| 4646 | Rosa       | Bradtke   | lorenza53@example.net     | 2019-04-11 | 2016-05-15 09:42:36 |
| 6731 | Dariana    | Padberg   | xfritsch@example.com      | 2019-04-11 | 2018-04-24 05:50:11 |
| 4100 | Elliot     | Miller    | elaina97@example.net      | 2019-04-08 | 2001-03-30 15:44:57 |
| 7557 | Darrion    | Larkin    | camden48@example.com      | 2019-04-07 | 2004-04-26 12:45:46 |
| 1189 | Cassandra  | Sporer    | blake84@example.net       | 2019-04-07 | 1970-05-04 23:28:01 |
+------+------------+-----------+---------------------------+------------+---------------------+
10 rows in set (0.01 sec)
```

1.9. Use the authors_100 table to delete and records with the following conditions, and show the results.

1.9.1. Display the current number of records, and verify that all records with a birth date before 2015 have been deleted.

```
MariaDB [labs]> delete from authors_100 where birthdate < '2015-01-01';
```

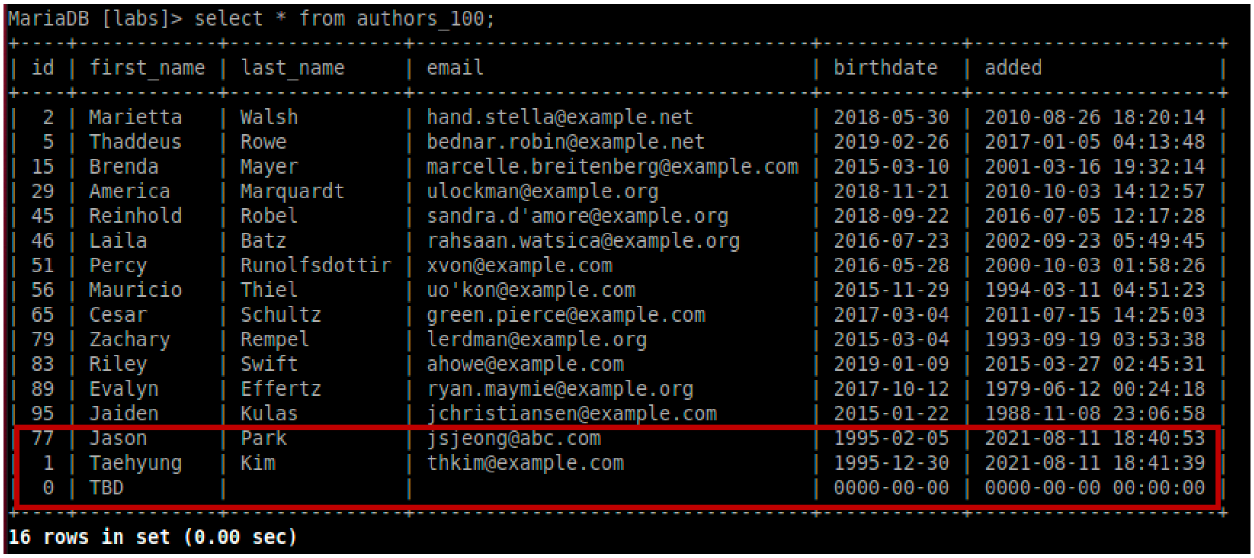


```
MariaDB [labs]> select count(*) from authors_100;
+----------+
| count(*) |
+----------+
|      100 |
+----------+
1 row in set (0.00 sec)

MariaDB [labs]> delete from authors_100 where birthdate < '2015-01-01';
Query OK, 87 rows affected (0.01 sec)

MariaDB [labs]> select count(*) from authors_100;
+----------+
| count(*) |
+----------+
|       13 |
+----------+
1 row in set (0.00 sec)
```

1.9.2. Confirm that line 87 has been deleted, and add the following data to the table. Validate this record. Add to 3 rows like as :

id	first_name	last_name	email	birthdate	added
77	Jason	Park	jsjeong@abc.com	1995-02-05	now
1	Taehyung	Kim	thkim@example.com	1995-12-31	now
	TBD				

```
MariaDB [labs]> insert into authors_100  values (77, 'Jason', 'Park', 'jsjeong@abc.com', '1995-02-05', now());

MariaDB [labs]> insert into authors_100  values (1, 'Taehyung', 'Kim', 'thkim@example.com', '1995-12-30', now());

MariaDB [labs]> insert into authors_100 (first_name) values ('TBD');
```

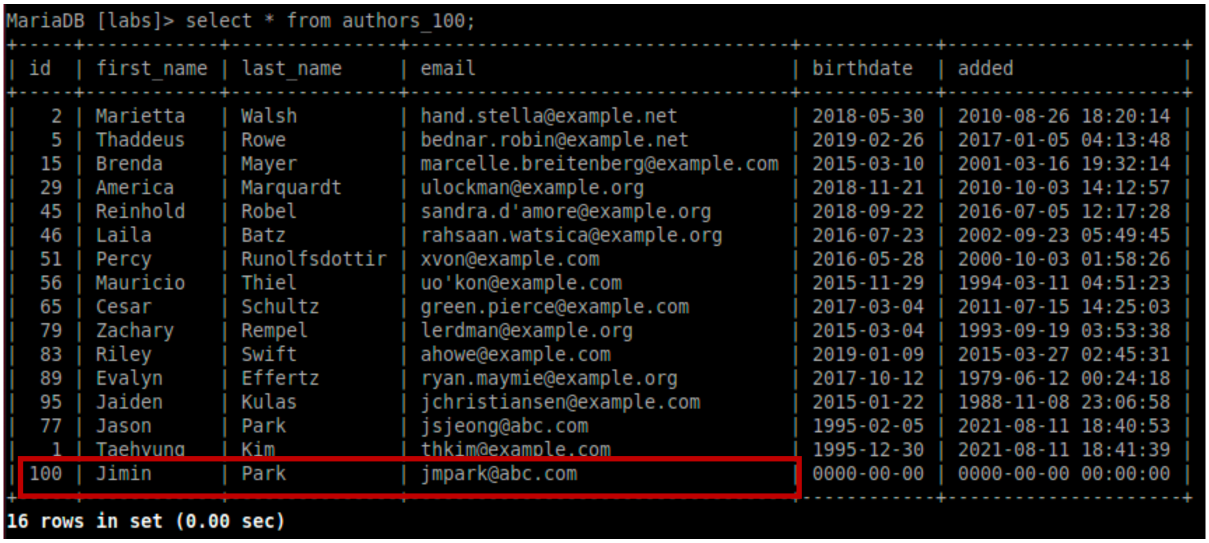


```
MariaDB [labs]> select * from authors_100;
+----+------------+---------------+----------------------------------+------------+---------------------+
| id | first_name | last_name     | email                            | birthdate  | added               |
+----+------------+---------------+----------------------------------+------------+---------------------+
|  2 | Marietta   | Walsh         | hand.stella@example.net          | 2018-05-30 | 2010-08-26 18:20:14 |
|  5 | Thaddeus   | Rowe          | bednar.robin@example.net         | 2019-02-26 | 2017-01-05 04:13:48 |
| 15 | Brenda     | Mayer         | marcelle.breitenberg@example.com | 2015-03-10 | 2001-03-16 19:32:14 |
| 29 | America    | Marquardt     | ulockman@example.org             | 2018-11-21 | 2010-10-03 14:12:57 |
| 45 | Reinhold   | Robel         | sandra.d'amore@example.org       | 2018-09-22 | 2016-07-05 12:17:28 |
| 46 | Laila      | Batz          | rahsaan.watsica@example.org      | 2016-07-23 | 2002-09-23 05:49:45 |
| 51 | Percy      | Runolfsdottir | xvon@example.com                 | 2016-05-28 | 2000-10-03 01:58:26 |
| 56 | Mauricio   | Thiel         | uo'kon@example.com               | 2015-11-29 | 1994-03-11 04:51:23 |
| 65 | Cesar      | Schultz       | green.pierce@example.com         | 2017-03-04 | 2011-07-15 14:25:03 |
| 79 | Zachary    | Rempel        | lerdman@example.org              | 2015-03-04 | 1993-09-19 03:53:38 |
| 83 | Riley      | Swift         | ahowe@example.com                | 2019-01-09 | 2015-03-27 02:45:31 |
| 89 | Evalyn     | Effertz       | ryan.maymie@example.org          | 2017-10-12 | 1979-06-12 00:24:18 |
| 95 | Jaiden     | Kulas         | jchristiansen@example.com        | 2015-01-22 | 1988-11-08 23:06:58 |
| 77 | Jason      | Park          | jsjeong@abc.com                  | 1995-02-05 | 2021-08-11 18:40:53 |
|  1 | Taehyung   | Kim           | thkim@example.com                | 1995-12-30 | 2021-08-11 18:41:39 |
|  0 | TBD        |               |                                  | 0000-00-00 | 0000-00-00 00:00:00 |
+----+------------+---------------+----------------------------------+------------+---------------------+
16 rows in set (0.00 sec)
```

1.10. Update table data. Modify the record whose id value is 0 with the following information.

1.10.1. First_name:Jimin, last_name: Park, email:jmpark@abc.com

```
MariaDB [labs]> update authors_100 set id = 100, first_name = 'Jimin', last_name = 'Park', email = 'jmpark@abc.com' where id = 0;
```



```
MariaDB [labs]> select * from authors_100;
+-----+------------+---------------+----------------------------------+------------+---------------------+
| id  | first_name | last_name     | email                            | birthdate  | added               |
+-----+------------+---------------+----------------------------------+------------+---------------------+
|   2 | Marietta   | Walsh         | hand.stella@example.net          | 2018-05-30 | 2010-08-26 18:20:14 |
|   5 | Thaddeus   | Rowe          | bednar.robin@example.net         | 2019-02-26 | 2017-01-05 04:13:48 |
|  15 | Brenda     | Mayer         | marcelle.breitenberg@example.com | 2015-03-10 | 2001-03-16 19:32:14 |
|  29 | America    | Marquardt     | ulockman@example.org             | 2018-11-21 | 2010-10-03 14:12:57 |
|  45 | Reinhold   | Robel         | sandra.d'amore@example.org       | 2018-09-22 | 2016-07-05 12:17:28 |
|  46 | Laila      | Batz          | rahsaan.watsica@example.org      | 2016-07-23 | 2002-09-23 05:49:45 |
|  51 | Percy      | Runolfsdottir | xvon@example.com                 | 2016-05-28 | 2000-10-03 01:58:26 |
|  56 | Mauricio   | Thiel         | uo'kon@example.com               | 2015-11-29 | 1994-03-11 04:51:23 |
|  65 | Cesar      | Schultz       | green.pierce@example.com         | 2017-03-04 | 2011-07-15 14:25:03 |
|  79 | Zachary    | Rempel        | lerdman@example.org              | 2015-03-04 | 1993-09-19 03:53:38 |
|  83 | Riley      | Swift         | ahowe@example.com                | 2019-01-09 | 2015-03-27 02:45:31 |
|  89 | Evalyn     | Effertz       | ryan.maymie@example.org          | 2017-10-12 | 1979-06-12 00:24:18 |
|  95 | Jaiden     | Kulas         | jchristiansen@example.com        | 2015-01-22 | 1988-11-08 23:06:58 |
|  77 | Jason      | Park          | jsjeong@abc.com                  | 1995-02-05 | 2021-08-11 18:40:53 |
|   1 | Taehyung   | Kim           | thkim@example.com                | 1995-12-30 | 2021-08-11 18:41:39 |
| 100 | Jimin      | Park          | jmpark@abc.com                   | 0000-00-00 | 0000-00-00 00:00:00 |
+-----+------------+---------------+----------------------------------+------------+---------------------+
16 rows in set (0.00 sec)
```

1.11.Drop the authors2 table and exit MariaDB.

```
MariaDB [labs]> drop table authors2;
MariaDB [labs]> exit
```

Lab 2: Working with SQL Tables

1. Create a table named countries. Use the following schema:

Column Name	Column Type	Constraints
id	serial	
name	varchar(50)	UNIQUE, NOT NULL
city	varchar(50)	NOT NULL
population	integer	
latitude	decimal(10,8)	
longitude	decimal(11,8)	

1.1. Create a table named **famous_people**. Use the following conditions:

Column Name	Conditions	Contraints
uid	auto-incrementing value	
name	string up to 100 characters	Cannot be NULL
occupation	string up to 150 characters	
birthday	string up to 50 characters	
existence	contains either true or false	If unknown, should be false

1.2. Create a table named **products** The table must contain the following sample data and contain an auto incrementing **pid** column :

Product name	Model	Max Weight	Max service years	Parts name
DC motor	D9I40GBH	750	15	DCX
BLDC motor	S7J30GHE	3,800	25	SNP
AC motor	G8I50BHE	10,000	30	GDE

1.3. Other considerations:

- Product name and Model have maximum 75 characters
- Product name and Model cannot be empty
- The Weights can be in the range of 0.001 kg to 40,000 kg
- Numbers should be formatted to use commas every 1000 places and periods for decimal places. Decimals should not be formatted smaller than the 100th place.
- Conservation status is a 3 letter code

1.4. Create a table named **orders**. Use the following conditions:

Column Name	Conditions	Contraints
id	auto-incrementing value	
customer_name	string up to 100 characters	cannot be empty
burger	string up to 50 characters	
side	string up to 50 characters	
drink	string up to 50 characters	
order_total	dollars and cents numeric	cannot be empty. all orders are less than $100

2. Modifying tables

2.1. Rename the famous_people table to celebrities.

2.2. Change the **celebrities** table with the following:

2.2.1. Change **name** column to **first_name**

Change its data type to varchar(80)

2.2.2. Add a new column named **last_name.**
The column can store strings up 100 characters. This column cannot be empty

2.2.3. Change the data type of **date_of_birth** to a date type instead of a string

date_of_birth column cannot be empty

2.3. Change the products table with the following:

2.3.1. The maximum weight can now be in the range of 0.0001 kg to 200,000 kg

2.3.2. The model cannot contain duplicates

2.4. Change the **orders** table with the following:

2.4.1. Add a new column to store customer email addresses. Make it a string with maximum 50 characters

2.4.2. Add a new column named **customer_loyalty_points**. It holds an integer value. The default value for the column should be 0.

2.4.3. Add a new column called **burger_cost**. It contains dollars and cents. The maximum value is $100. The default value is 0.

2.4.4. Add a new column called **side_cost**. It contains dollars and cents. The maximum value is $100. The default value is 0.

2.4.5. Add a new column called **drink_cost**. It contains dollars and cents. The maximum value is $100. The default value is 0.

2.4.6. Remove the **order_total** column.

Lab 3: Working with Tables

1. Adding data to tables

1.1. Add to **countries** table:

Name	Capital	Population
USA	Washington D.C.	333,098,437
Germany	Berlin	84,073,352
France	Paris	65,426,179
Korea	Seoul	51,305,186

1.2. Add to **celebrities** table

1.2.1. Add birthdays for the members of BTS. BTS members are Singer and Song Writers.

Name	Date of Birth
Namjoon Kim	September 12, 1994
Jeongguk Jeon	September 1, 1997
Yoongi Min	March 9, 1993
Hoseok Jung	February 18, 1994
Taehyun Kim	December 30, 1995
Jimin Park	October 13, 1995
Seokjin Kim	December 4, 1992

1.2.2. Use the INSERT command to add the following:

First Name	Last Name	Occupation	Date of Birth
Yong	Cho	Singer, Actor	December 12, 1915
SY	Lee	Actor	July 3, 1962

Also, entered the deceases status for the two entries above. Frank Sinatra is deceased. Tom Cruise is alive but use the default setting to enter that status.

1.2.3. Enter the following data. What happens?

First Name	Last Name	Occupation	Date of Birth	Deceased
Jason		Singer, Actress	'08/15/1968'	false
Henry		Singer, Songwriter, Actor	'11/06/1961'	true

Last name is a required field.

2. Updating tables

2.1. Update the **celebrities** table

2.1.1. Change the last name column so that we can add the entries from 1.2.3.

2.1.2. Review the schema for the celebrities table. What would happen if we tried to enter the following:

First Name	Last Name	Occupation	Date of Birth	Deceased
Alice	Perry	Singer, Actor	'01/08/1945'	NULL

2.2. Enter the following into the **orders** table: You will have to examine the schema and use INSERT statements appropriately.

2.2.1. Customer information:
Henry Kim, Shaun Silverman, and Jason Gomez.

2.2.2. Customer email:
Henry's email address is henry@lab.com.
Shaun's email address is shaun@lab.com.
Jason doesn't have an email address.

2.2.3. Order information:
Henry orders a cheeseburger, fries and a soda
Shaun orders a cheeseburger, onion ring and chocolate shake
He also orders a chicken burger, fries, soda
Jason orders a side of onion rings with a strawberry shake.

The item costs and redeem points are listed below:

Item	Cost	Redeem points
Burger	3.00	5
Cheeseburger	4.00	7
Chicken Burger	3.50	6
Double Cheeseburger	6.00	9
Fries	1.50	2

Onion Ring	2.00	2
Soda	0.99	1
Shake	2.00	2

2.3. Query the **countries** table for the following information:

2.3.1. Population of Korea

2.3.2. Population and capital of all entries in the **countries** table

2.3.3. Names of countries in alphabetical order

2.3.4. Country, capital and population, ordered by population, descending

2.3.5. Same as above but ordered ascending

2.3.6. Countries with a population less than 100,000,000.

2.3.7. Countries with a population between 50 and 100 million

2.4. Query the **products** table for the following information:

2.4.1. Product name, model, max weight, and max service years, ordered by max service years in ascending order

2.4.2. List the name of products in descending order to maximum weight

2.5. Query the **celebrities** table or the following information:

2.5.1. First and last name of all celebrities still alive

2.5.2. All celebrities born after 1990, in ascending order by age

2.5.3. List first name of all celebrities who are both singers and actors(actress)

2.5.4. List first and last name of all singers sorted by age, descending order

2.6. Query the **orders** table for the following information:

2.6.1. List the customer with the highest single expensive order

2.6.2. List all customers email address so that we can send them coupons. Do not include those that do not have an email address

2.6.3. List names of all customers who ordered fries

2.6.4. List all orders that included a shake

2.6.5. List any order that did not include a main sandwich.

Lab 4: Working with Queries

1. Creating queries

   1.1. Create queries on the **countries** table

   - 1.1.1. Name the country with the smallest population
   - 1.1.2. Name the country with the second largest population
   - 1.1.3. List the first row in the countries table
   - 1.1.4. How many countries are listed in the **countries** table?

   1.2. Create queries on the **products** table

   - 1.2.1. List all unique model name
   - 1.2.2. Count number of products with a parts name of DCX
   - 1.2.3. What is the maximum service years and which model is?
   - 1.2.4. What is the average weight of all the products

   1.3. Create queries on the **celebrities** table

   - 1.3.1. How many celebrities have multiple occupations?
   - 1.3.2. Which celebrities have more than 3 occupations?
   - 1.3.3. Who is the oldest celebrity?
   - 1.3.4. Who is the youngest?

   1.4. Create queries on the **orders** table

   - 1.4.1. What is the total amount of all the orders
   - 1.4.2. How many customer have multiple orders and who are they?

2. Updating data in tables

   2.1. Modify the **countries** table:

   - 2.1.1. Add continent column as string with maximum 100 characters
   - 2.1.2. Change the continent to the following:
     USA = North America
     Germany = Europe

France = Europe
Korea = Asia

2.2. Modify celebrities table:

2.2.1. Remove anyone who is not a singer

2.2.2. Remove the column occupation

2.2.3. Change the name of the table to singers

2.2.4. Remove the members of BTS that we entered in Lab 3:, section 1.2.1. We will replace the individual members with the group.

2.2.5. Add a new singer with name = BTS.

Lab 5: Using Pig for ETL Processing (Pre-processing)

In this lab, you will use the Pig to explore, correct, and pre-processing data in files. The two data files to be used in the lab are stored in different formats, so in order to analyze them together, the data format must be unified. For this work, Pig is used to perform pre-processing.

The pre-processed data is used for analysis using hive in the next unit.

1. Working in the Grunt shell

1.1. Create the working directory for this lab and change to the directory. In lab, we'll make works directory here.

```
$ mkdir  ~/works
$ cd  ~/works
```

1.2. Copy pig_data1.txt, pig_data2.txt files in the Data folder under the home directory to this working directory.

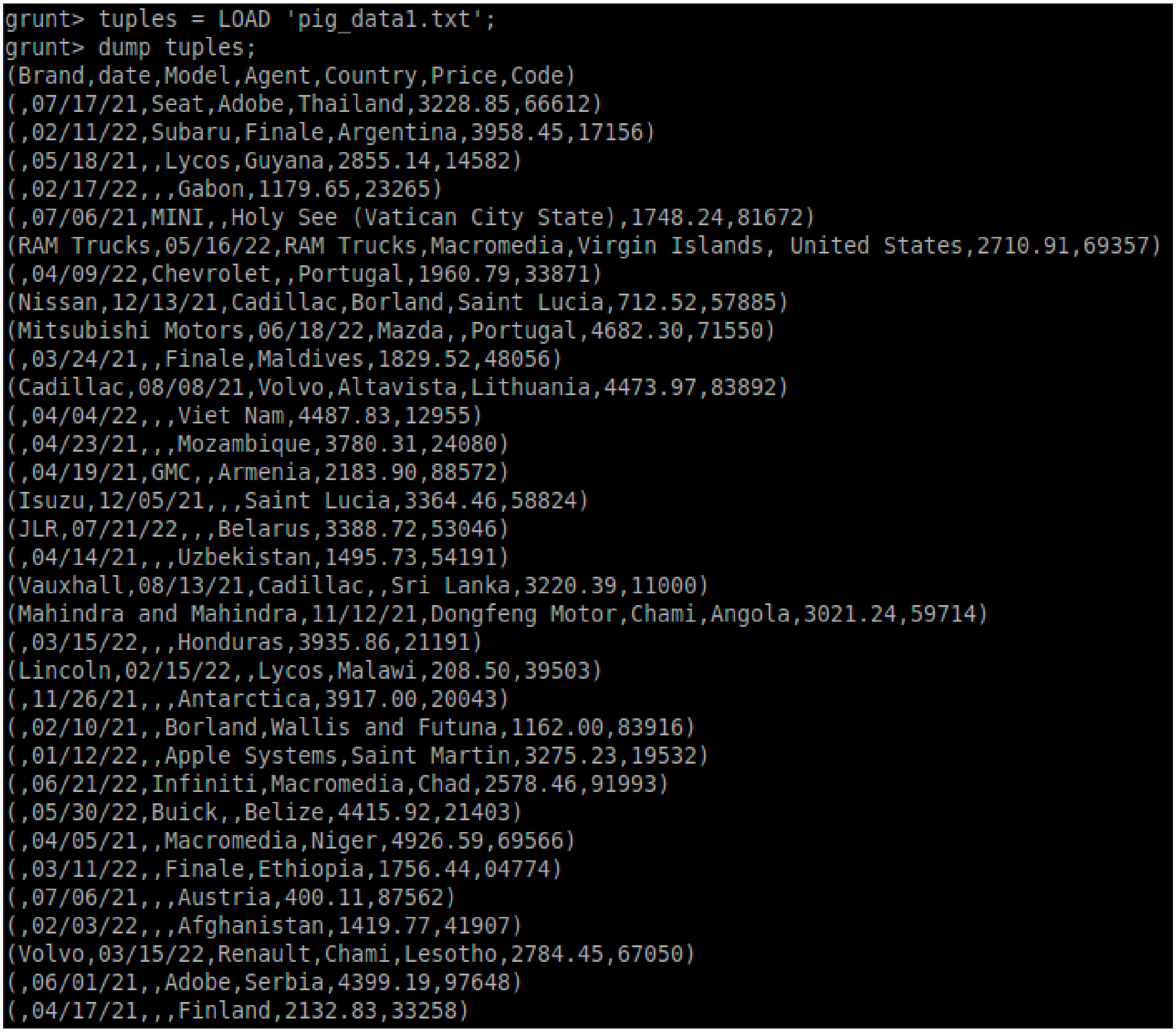
```
$ cp  ~/Data/pig_data* .
```

1.3. Start the Grunt shell in local mode.

```
$ pig -x local
```

1.4. Load the data in the data_pig1.txt file into shell and show the contents.

```
grunt> tuples = LOAD 'pig_data1.txt';
grunt> dump tuples;
```



```
grunt> tuples = LOAD 'pig_data1.txt';
grunt> dump tuples;
(Brand,date,Model,Agent,Country,Price,Code)
(,07/17/21,Seat,Adobe,Thailand,3228.85,66612)
(,02/11/22,Subaru,Finale,Argentina,3958.45,17156)
(,05/18/21,,Lycos,Guyana,2855.14,14582)
(,02/17/22,,,Gabon,1179.65,23265)
(,07/06/21,MINI,,Holy See (Vatican City State),1748.24,81672)
(RAM Trucks,05/16/22,RAM Trucks,Macromedia,Virgin Islands, United States,2710.91,69357)
(,04/09/22,Chevrolet,,Portugal,1960.79,33871)
(Nissan,12/13/21,Cadillac,Borland,Saint Lucia,712.52,57885)
(Mitsubishi Motors,06/18/22,Mazda,,Portugal,4682.30,71550)
(,03/24/21,,Finale,Maldives,1829.52,48056)
(Cadillac,08/08/21,Volvo,Altavista,Lithuania,4473.97,83892)
(,04/04/22,,,Viet Nam,4487.83,12955)
(,04/23/21,,,Mozambique,3780.31,24080)
(,04/19/21,GMC,,Armenia,2183.90,88572)
(Isuzu,12/05/21,,,Saint Lucia,3364.46,58824)
(JLR,07/21/22,,,Belarus,3388.72,53046)
(,04/14/21,,,Uzbekistan,1495.73,54191)
(Vauxhall,08/13/21,Cadillac,,Sri Lanka,3220.39,11000)
(Mahindra and Mahindra,11/12/21,Dongfeng Motor,Chami,Angola,3021.24,59714)
(,03/15/22,,,Honduras,3935.86,21191)
(Lincoln,02/15/22,,Lycos,Malawi,208.50,39503)
(,11/26/21,,,Antarctica,3917.00,20043)
(,02/10/21,,Borland,Wallis and Futuna,1162.00,83916)
(,01/12/22,,Apple Systems,Saint Martin,3275.23,19532)
(,06/21/22,Infiniti,Macromedia,Chad,2578.46,91993)
(,05/30/22,Buick,,Belize,4415.92,21403)
(,04/05/21,,Macromedia,Niger,4926.59,69566)
(,03/11/22,,Finale,Ethiopia,1756.44,04774)
(,07/06/21,,,Austria,400.11,87562)
(,02/03/22,,,Afghanistan,1419.77,41907)
(Volvo,03/15/22,Renault,Chami,Lesotho,2784.45,67050)
(,06/01/21,,Adobe,Serbia,4399.19,97648)
(,04/17/21,,,Finland,2132.83,33258)
```

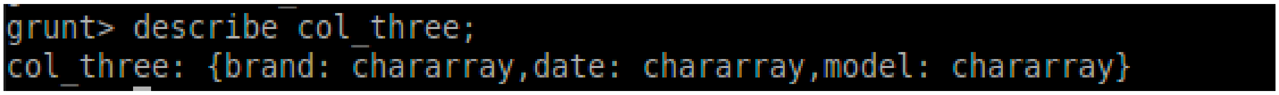
Note: You can see some records. You may discover records that are missing some data or have errors.

1.5. Load the first three field values and display them on the screen.

```
grunt> col_three = load 'pig_data1.txt' as (brand:chararray, date:chararray,
model:chararray);

grunt> dump col_three;
```

1.6. Use the DESCRIBE command to see the schema of col_three;

```
grunt> describe col_three;
```



```
grunt> describe col_three;
col_three: {brand: chararray,date: chararray,model: chararray}
```

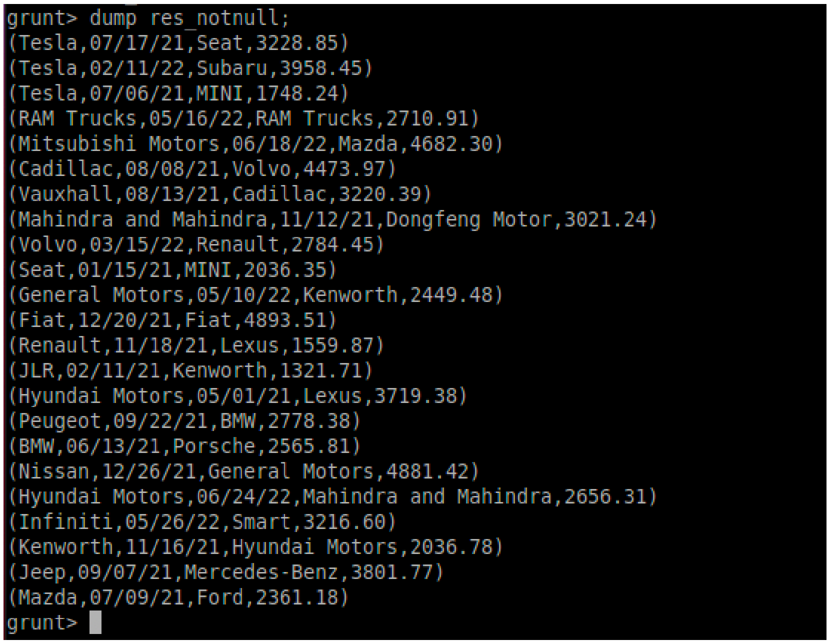
1.7. Next, create a relation that meets the following conditions using the default data type and field order.

1.7.1. Read pig_data1.txt and find only the records with the 6th field value greater than 1000, and print only the 1st, 2nd, 3rd, and 6th field values on the result screen.

```
grunt> data = load 'pig_data1.txt';
grunt> hi_price = filter data by $5 > 1000;
grunt> res = foreach hi_price generate $0, $1, $2, $5;
grunt> dump res;
```

1.7.2. Display the previous results except that the first and third field values are null.

```
grunt> res_notnull = filter res by $0 is not null and $2 is not null;
grunt> dump res_notnull;
grunt> quit;
```



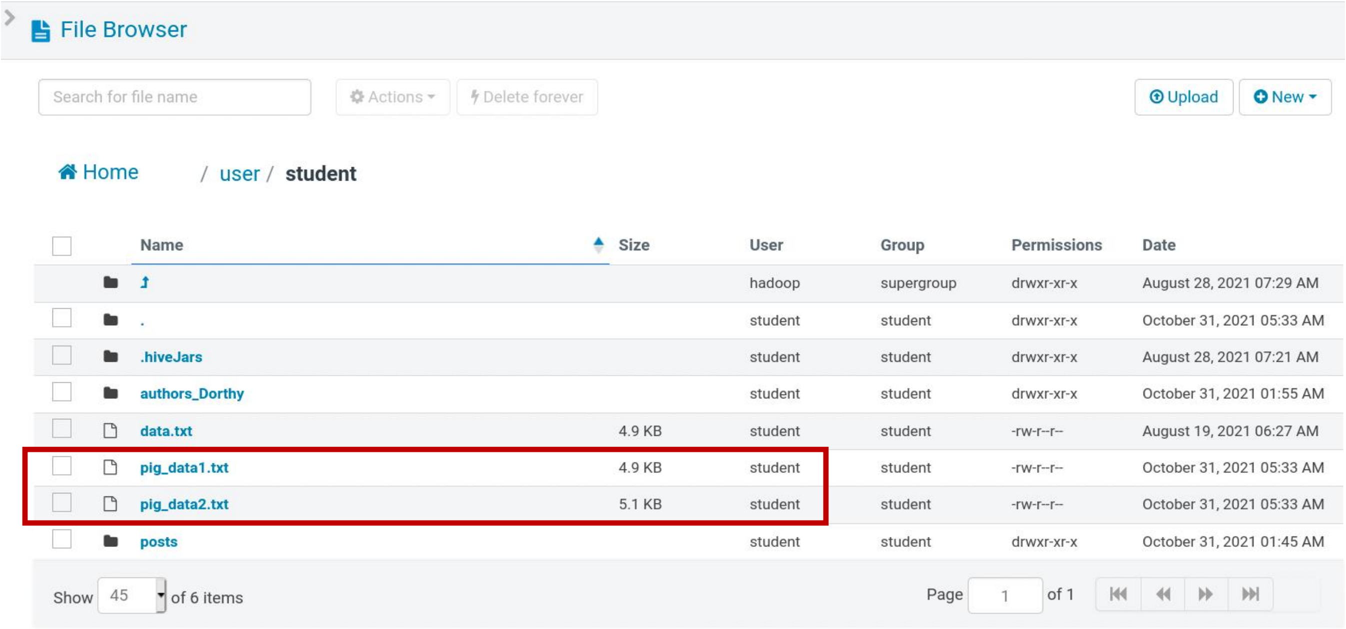
```
grunt> dump res_notnull;
(Tesla,07/17/21,Seat,3228.85)
(Tesla,02/11/22,Subaru,3958.45)
(Tesla,07/06/21,MINI,1748.24)
(RAM Trucks,05/16/22,RAM Trucks,2710.91)
(Mitsubishi Motors,06/18/22,Mazda,4682.30)
(Cadillac,08/08/21,Volvo,4473.97)
(Vauxhall,08/13/21,Cadillac,3220.39)
(Mahindra and Mahindra,11/12/21,Dongfeng Motor,3021.24)
(Volvo,03/15/22,Renault,2784.45)
(Seat,01/15/21,MINI,2036.35)
(General Motors,05/10/22,Kenworth,2449.48)
(Fiat,12/20/21,Fiat,4893.51)
(Renault,11/18/21,Lexus,1559.87)
(JLR,02/11/21,Kenworth,1321.71)
(Hyundai Motors,05/01/21,Lexus,3719.38)
(Peugeot,09/22/21,BMW,2778.38)
(BMW,06/13/21,Porsche,2565.81)
(Nissan,12/26/21,General Motors,4881.42)
(Hyundai Motors,06/24/22,Mahindra and Mahindra,2656.31)
(Infiniti,05/26/22,Smart,3216.60)
(Kenworth,11/16/21,Hyundai Motors,2036.78)
(Jeep,09/07/21,Mercedes-Benz,3801.77)
(Mazda,07/09/21,Ford,2361.18)
grunt> 
```

2. Processing Input Data from a Pig script

 2.1. Make the script file, etl_1.pig, running your command for validation in grunt shell.

  2.1.1. First, copy the data pig_data1.txt and pig_data2.txt to home directory folder in hdfs .

```
$ hdfs dfs -put pig_data1.txt pig_data2.txt .
```



Home / user / student

□		Name	Size	User	Group	Permissions	Date
		↑		hadoop	supergroup	drwxr-xr-x	August 28, 2021 07:29 AM
□		.		student	student	drwxr-xr-x	October 31, 2021 05:33 AM
□		.hiveJars		student	student	drwxr-xr-x	August 28, 2021 07:21 AM
□		authors_Dorthy		student	student	drwxr-xr-x	October 31, 2021 01:55 AM
□		data.txt	4.9 KB	student	student	-rw-r--r--	August 19, 2021 06:27 AM
□		pig_data1.txt	4.9 KB	student	student	-rw-r--r--	October 31, 2021 05:33 AM
□		pig_data2.txt	5.1 KB	student	student	-rw-r--r--	October 31, 2021 05:33 AM
□		posts		student	student	drwxr-xr-x	October 31, 2021 01:45 AM

Show 45 of 6 items Page 1 of 1

  2.1.2. Load the data file pig_data1.txt using the data schema below.

Index	Field Name	Data type
0	brand	chararray
1	date	chararray
2	model	chararray
3	agent	chararray
4	country	chararray
5	price	int
6	code	chararray

2.1.3. Filter out all fields with null values for brand name, model, and agent.

2.1.4. Select only values with a price of 1000 or more, and use the foreach command to create relations in the following order:

Index	Field Name	Data type
0	code	chararray
1	brand	chararray
2	model	chararray
3	date	chararray
4	country	chararray
5	price	int

2.1.5. Change the brand field to uppercase and remove any leading or trailing whitespace. (Hint: built-in function TRIM)

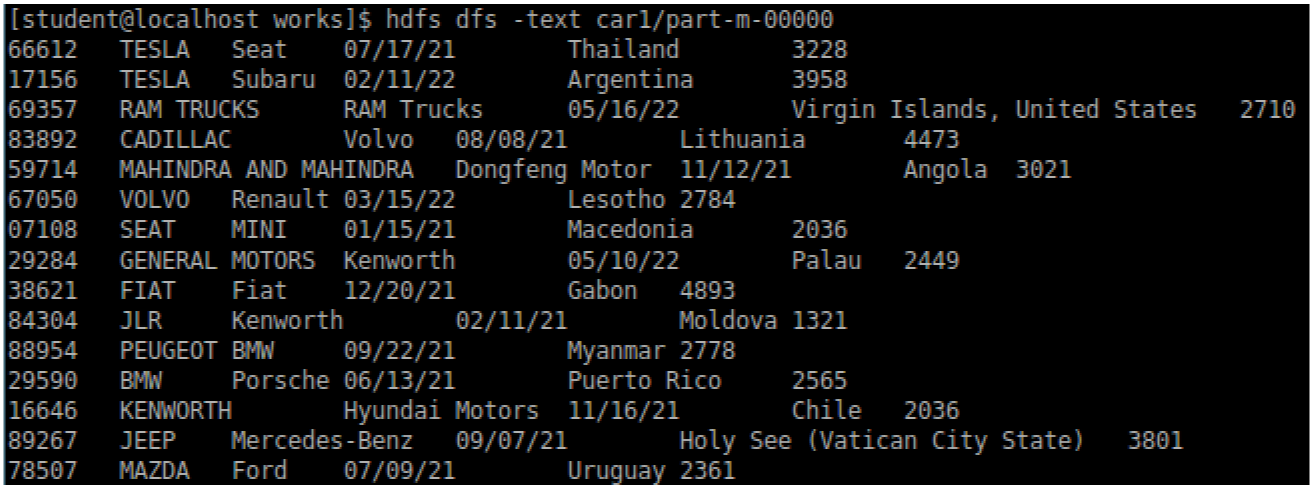
2.1.6. Save the result relation as car1.

```
$ vim etl_1.pig
data = LOAD 'pig_data1.txt' AS (brand:chararray,
          date:chararray,
          model:chararray,
          agent:chararray,
          country:chararray,
          price:int,
          code:chararray);


data_notnull = FILTER data BY brand is not null and model is not null and agent is not null;
price_up = filter data_notnull by price > 1000;
```

```
reordered = FOREACH price_up GENERATE code,
        UPPER(TRIM(brand)),
        model,
        date,
        country,
        price;

STORE reordered INTO 'car1';
```

2.1.7. Run the etl_1.pig

```
$ pig etl_1.pig
```



```
[student@localhost works]$ hdfs dfs -text car1/part-m-00000
66612   TESLA   Seat    07/17/21        Thailand        3228
17156   TESLA   Subaru  02/11/22        Argentina       3958
69357   RAM TRUCKS      RAM Trucks      05/16/22        Virgin Islands, United States   2710
83892   CADILLAC        Volvo   08/08/21        Lithuania       4473
59714   MAHINDRA AND MAHINDRA   Dongfeng Motor  11/12/21        Angola  3021
67050   VOLVO   Renault 03/15/22        Lesotho 2784
07108   SEAT    MINI    01/15/21        Macedonia       2036
29284   GENERAL MOTORS  Kenworth        05/10/22        Palau   2449
38621   FIAT    Fiat    12/20/21        Gabon   4893
84304   JLR     Kenworth        02/11/21        Moldova 1321
88954   PEUGEOT BMW     09/22/21        Myanmar 2778
29590   BMW     Porsche 06/13/21        Puerto Rico     2565
16646   KENWORTH        Hyundai Motors  11/16/21        Chile   2036
89267   JEEP    Mercedes-Benz   09/07/21        Holy See (Vatican City State)   3801
78507   MAZDA   Ford    07/09/21        Uruguay 2361
```

3. Create the script file for pre-processing

3.1. Make the script file, etl_2.pig using pig_data2.txt.

3.1.1. First, load the data file pig_data2.txt using the data schema below.

Index	Field Name	Data type
0	brand	chararray
1	date	chararray
2	model	chararray

3	agent	chararray
4	country	chararray
5	price	int
6	code	chararray

3.1.2. Filter out all fields with null values for brand name, model, and agent.

3.1.3. Duplicate records with the same value are deleted.

3.1.4. Change the brand field to uppercase and remove any leading or trailing whitespace.

3.1.5. Remove the $sign in the cost field, create a relation in the following field order:

Index	Field Name	Data type
0	code	chararray
1	brand	chararray
2	model	chararray
3	date	chararray
4	country	chararray
5	price	chararray

3.1.6. Save the result in the same format as etl_1.pig to car2 folder in hdfs.

Hint: In the pig_data2 file, since each field is separated by ",", you must use the command USING PigStorage to separate it.

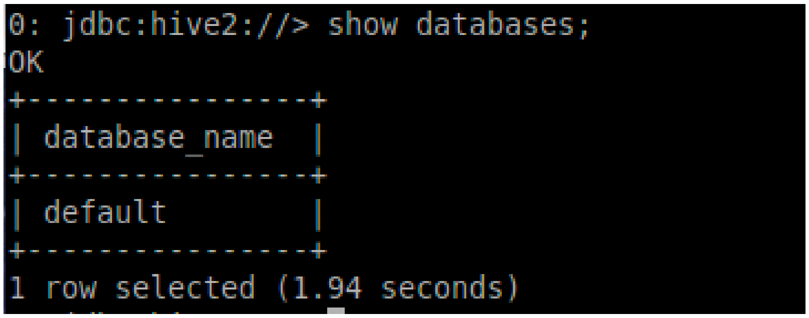
```
data = LOAD 'pig_data2.txt' USING PigStorage(',')

    AS (brand:chararray,

      date:chararray,

      model:chararray,

      agent:chararray,

      country:chararray,

      price:chararray,

      code:chararray);
```

Lab 6: Running Basic Queries with Hive QL

1. In terminal window, run the Beeline.

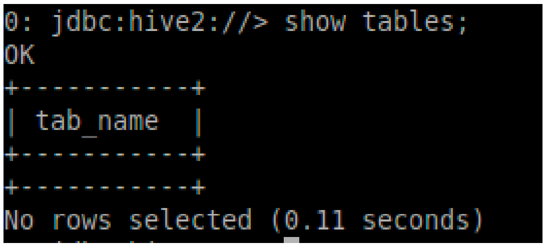
```
$ beeline -u jdbc:hive2://
```

1.1. If the beeline is running, the "0: jdbc:hive2://> " prompt appears and a screen waiting for commands is displayed. Enter a command to check which database exists here.



```
0: jdbc:hive2://> show databases;
OK
+----------------+
| database_name  |
+----------------+
| default        |
+----------------+
1 row selected (1.94 seconds)
```

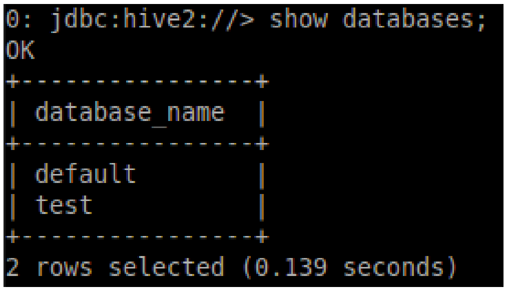
1.2. Next, enter the command to show the table in default DB.



```
0: jdbc:hive2://> show tables;
OK
+-----------+
| tab_name  |
+-----------+
+-----------+
No rows selected (0.11 seconds)
```

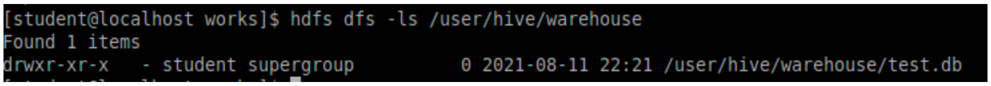
1.3. Create the test database and show the database list.

```
0: jdbc:hive2://> create database test;

0: jdbc:hive2://> show databases;
```



```
0: jdbc:hive2://> show databases;
OK
+----------------+
| database_name  |
+----------------+
| default        |
| test           |
+----------------+
2 rows selected (0.139 seconds)
```

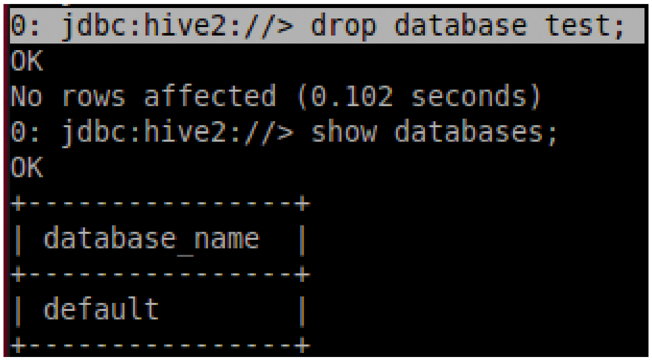
Note: The created database is saved as a folder in the hdfs directory.



```
[student@localhost works]$ hdfs dfs -ls /user/hive/warehouse
Found 1 items
drwxr-xr-x   - student supergroup          0 2021-08-11 22:21 /user/hive/warehouse/test.db
```

1.4. Delete the created test database and verify the result.

```
0: jdbc:hive2://> drop database test;

0: jdbc:hive2://> show databases;
```



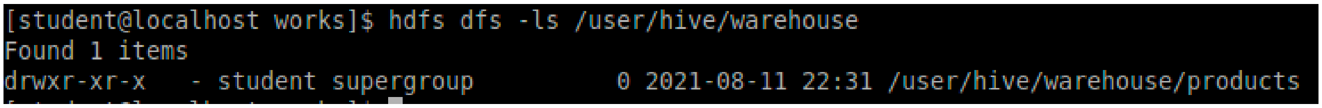
```
0: jdbc:hive2://> drop database test;
OK
No rows affected (0.102 seconds)
0: jdbc:hive2://> show databases;
OK
+----------------+
| database_name  |
+----------------+
| default        |
+----------------+
```

1.5. Create a table named products. Use the following schema:

Column Name	Column Type
Code	Int
brand	string
l_date	string
model	string
country	string
price	int

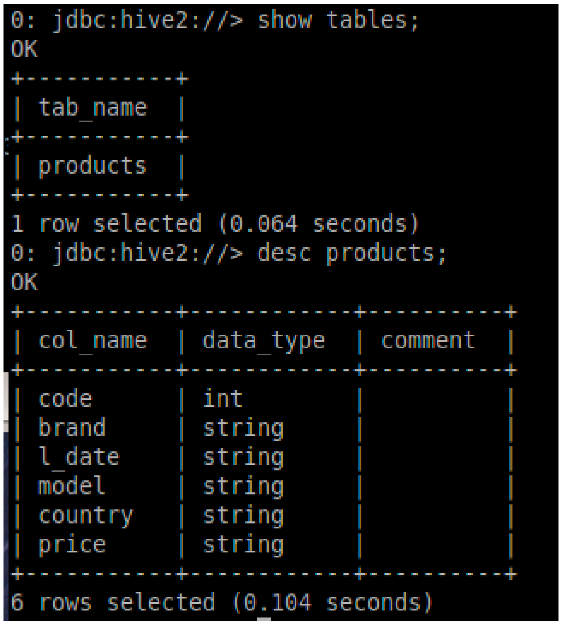
```
0: jdbc:hive2://> create table if not exists products (

code int,

brand string,

l_date string,

model string,

country string,

price string

)

row format delimited fields terminated by '\t';
```

1.6. Verify the products table using desc command and check the hdfs file system.



```
[student@localhost works]$ hdfs dfs -ls /user/hive/warehouse
Found 1 items
drwxr-xr-x   - student supergroup          0 2021-08-11 22:31 /user/hive/warehouse/products
```

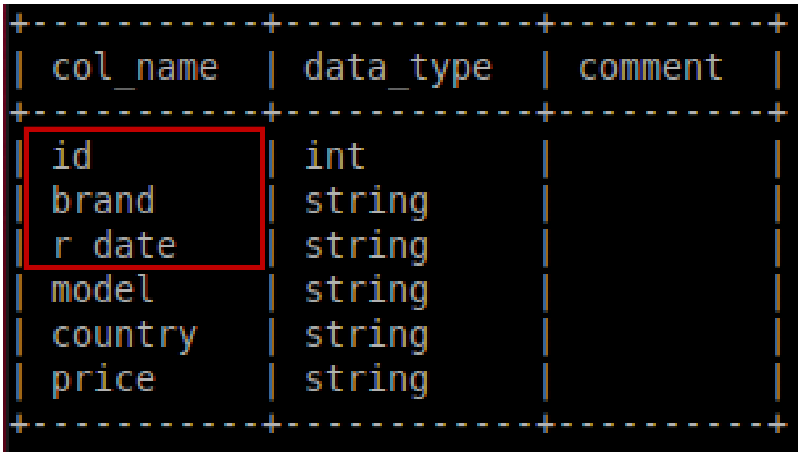
```
0: jdbc:hive2://> show tables;
0: jdbc:hive2://> desc products;
```



```
0: jdbc:hive2://> show tables;
OK
+-----------+
| tab_name  |
+-----------+
| products  |
+-----------+
1 row selected (0.064 seconds)
0: jdbc:hive2://> desc products;
OK
+-----------+------------+----------+
| col_name  | data_type  | comment  |
+-----------+------------+----------+
| code      | int        |          |
| brand     | string     |          |
| l_date    | string     |          |
| model     | string     |          |
| country   | string     |          |
| price     | string     |          |
+-----------+------------+----------+
6 rows selected (0.104 seconds)
```

1.7. Change the field l_date to the r_date column name, and change the code to id.

```
0: jdbc:hive2://> alter table products change code id int;
0: jdbc:hive2://> alter table products change l_date r_date string;
```

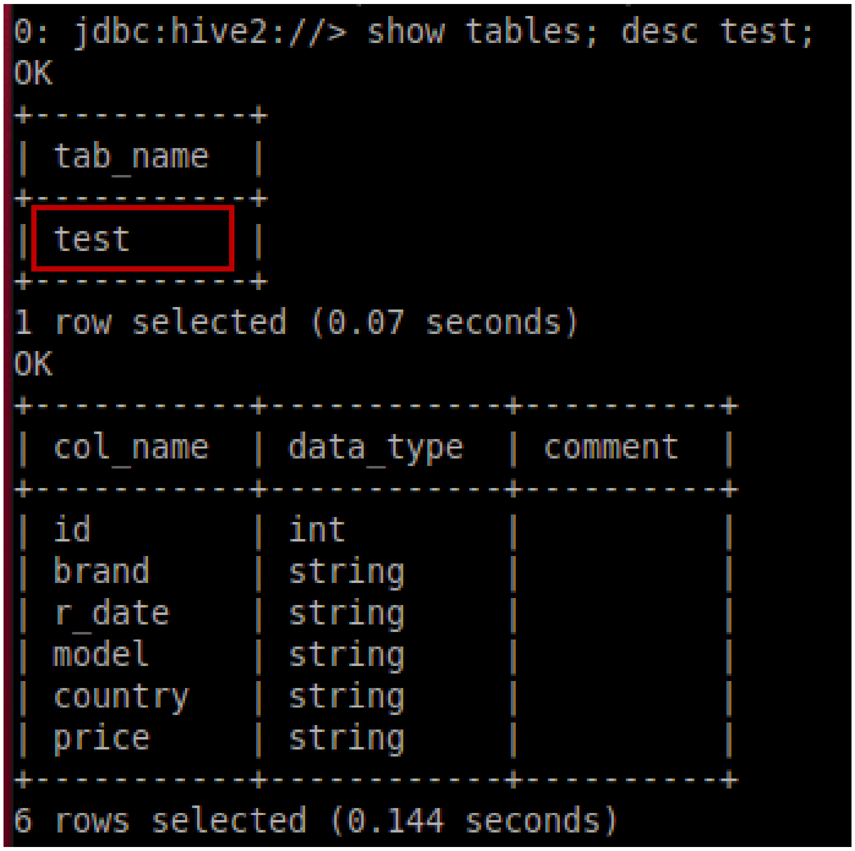


```
+-----------+------------+----------+
| col_name  | data_type  | comment  |
+-----------+------------+----------+
| id        | int        |          |
| brand     | string     |          |
| r date    | string     |          |
| model     | string     |          |
| country   | string     |          |
| price     | string     |          |
+-----------+------------+----------+
```

1.8. Rename the products table to test.

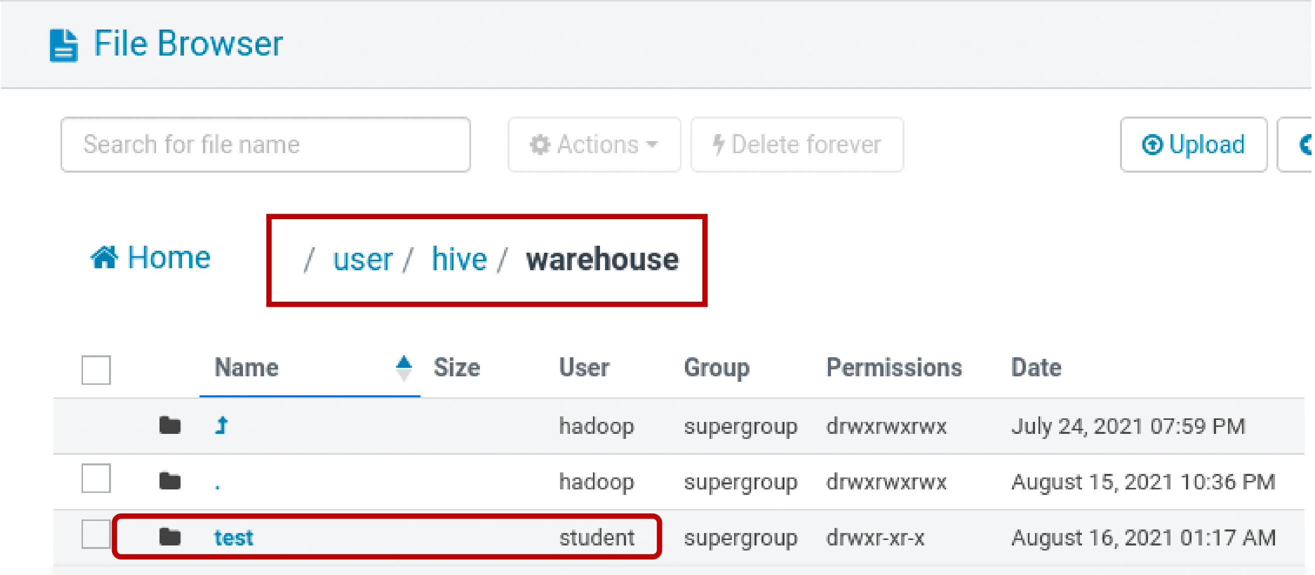
```
0: jdbc:hive2://> alter table products rename to test;
```

Note: When the alter command is executed, the hdfs folder name is also changed from products to test.



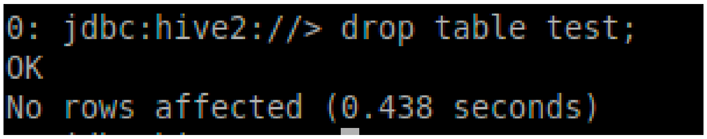
```
0: jdbc:hive2://> show tables; desc test;
OK
+-----------+
| tab_name  |
+-----------+
| test      |
+-----------+
1 row selected (0.07 seconds)
OK
+-----------+------------+----------+
| col_name  | data_type  | comment  |
+-----------+------------+----------+
| id        | int        |          |
| brand     | string     |          |
| r_date    | string     |          |
| model     | string     |          |
| country   | string     |          |
| price     | string     |          |
+-----------+------------+----------+
6 rows selected (0.144 seconds)
```

1.9. Drop the test table and verify the hdfs directory in /user/hive/warehouse.

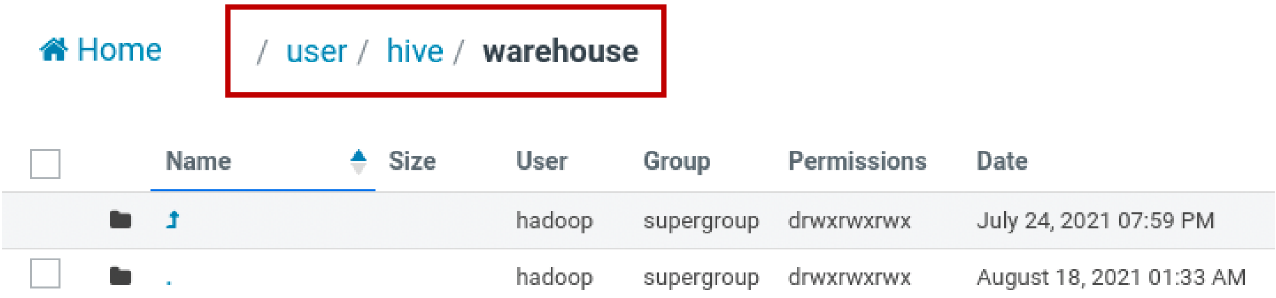


	Name	Size	User	Group	Permissions	Date
	↥		hadoop	supergroup	drwxrwxrwx	July 24, 2021 07:59 PM
□	.		hadoop	supergroup	drwxrwxrwx	August 15, 2021 10:36 PM
□	test		student	supergroup	drwxr-xr-x	August 16, 2021 01:17 AM

```
0: jdbc:hive2://> drop table test;
```



```
0: jdbc:hive2://> drop table test;
OK
No rows affected (0.438 seconds)
```



Home / user / hive / warehouse

□		Name	Size	User	Group	Permissions	Date
		↥		hadoop	supergroup	drwxrwxrwx	July 24, 2021 07:59 PM
□		.		hadoop	supergroup	drwxrwxrwx	August 18, 2021 01:33 AM

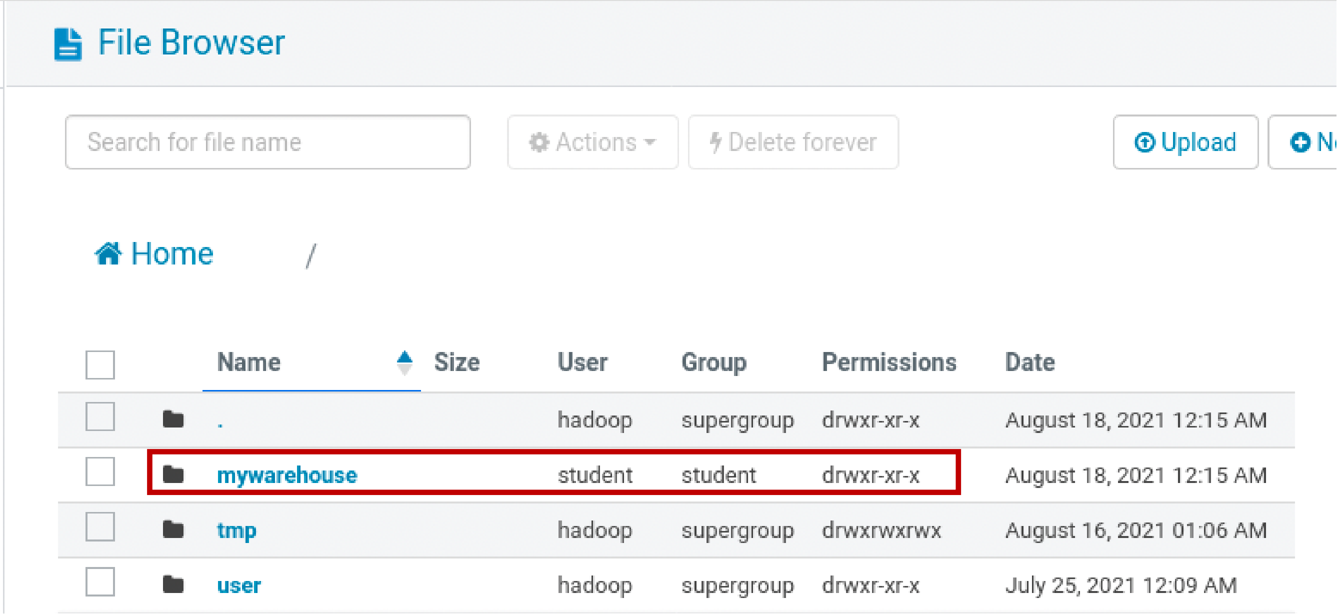
Note: When the managed table is deleted, the folder in the file system of hdfs is also deleted.

2. Create and load a table using Shell

2.1. Before using the shell, create a directory to be used in the lab as /mywarehouse in the hdfs file system and change the permission to student.

2.2. If there is /mywarehouse in your HDFS, skip these phases.

```
$ sudo -u hadoop hdfs dfs -mkdir /mywarehouse
$ sudo -u hadoop hdfs dfs -chown student:student /mywarehouse
```



□		Name	Size	User	Group	Permissions	Date
□		.		hadoop	supergroup	drwxr-xr-x	August 18, 2021 12:15 AM
□		mywarehouse		student	student	drwxr-xr-x	August 18, 2021 12:15 AM
□		tmp		hadoop	supergroup	drwxrwxrwx	August 16, 2021 01:06 AM
□		user		hadoop	supergroup	drwxr-xr-x	July 25, 2021 12:09 AM

2.3. And the data preprocessed using pig in the previous lab is saved here.

```
$ hdfs dfs -mv car1 car2 /mywarehouse
```

2.4. Create and load a table using editor and execute the script the shell -f option.

2.4.1. Create an external table using CREATE TABLE. In the last unit, you practiced creating file using Pig for data pre-processing. The schema is shown below:

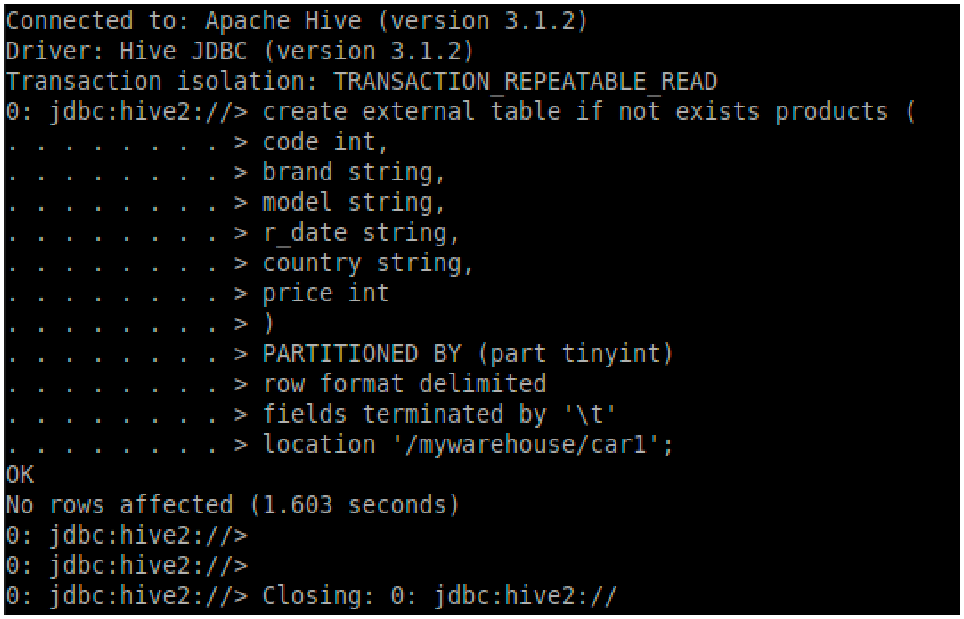
Column Name	Column Type
Code	Int
brand	string
model	string
r_date	string
country	string
price	int

2.4.2. The data are separated by '\t' for each record and are stored in the /mywarehouse/car1 directory in the hdfs file system.

```
$vi products.sql
create external table if not exists products (
code int,
brand string,
model string,
r_date string,
country string,
price int
)
row format delimited
fields terminated by '\t'
location '/mywarehouse/car1';
```

2.4.3. Run the products.sql with -f option.

```
$ beeline -u jdbc:hive2:// -f products.sql
```

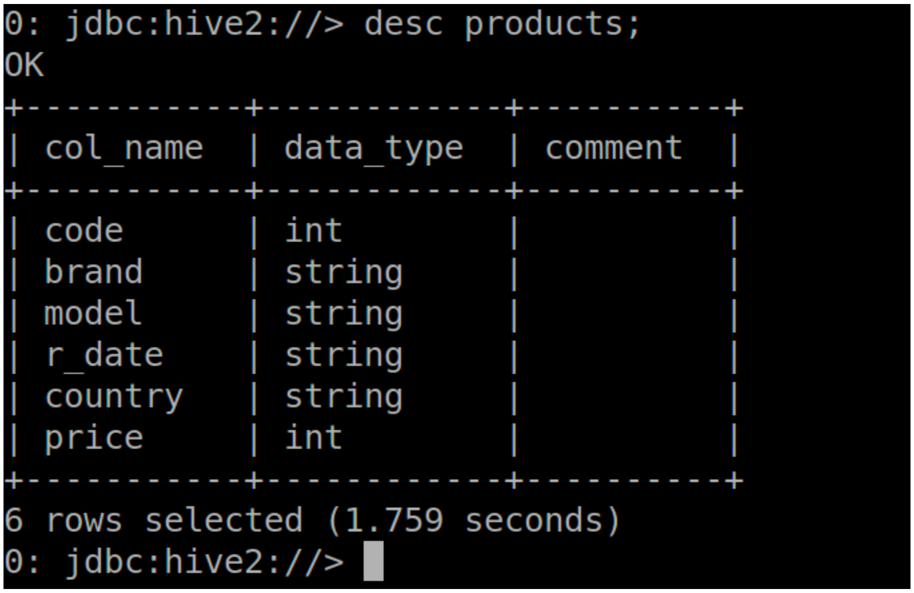


```
Connected to: Apache Hive (version 3.1.2)
Driver: Hive JDBC (version 3.1.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://> create external table if not exists products (
. . . . . . . . > code int,
. . . . . . . . > brand string,
. . . . . . . . > model string,
. . . . . . . . > r_date string,
. . . . . . . . > country string,
. . . . . . . . > price int
. . . . . . . . > )
. . . . . . . . > PARTITIONED BY (part tinyint)
. . . . . . . . > row format delimited
. . . . . . . . > fields terminated by '\t'
. . . . . . . . > location '/mywarehouse/car1';
OK
No rows affected (1.603 seconds)
0: jdbc:hive2://>
0: jdbc:hive2://>
0: jdbc:hive2://> Closing: 0: jdbc:hive2://
```

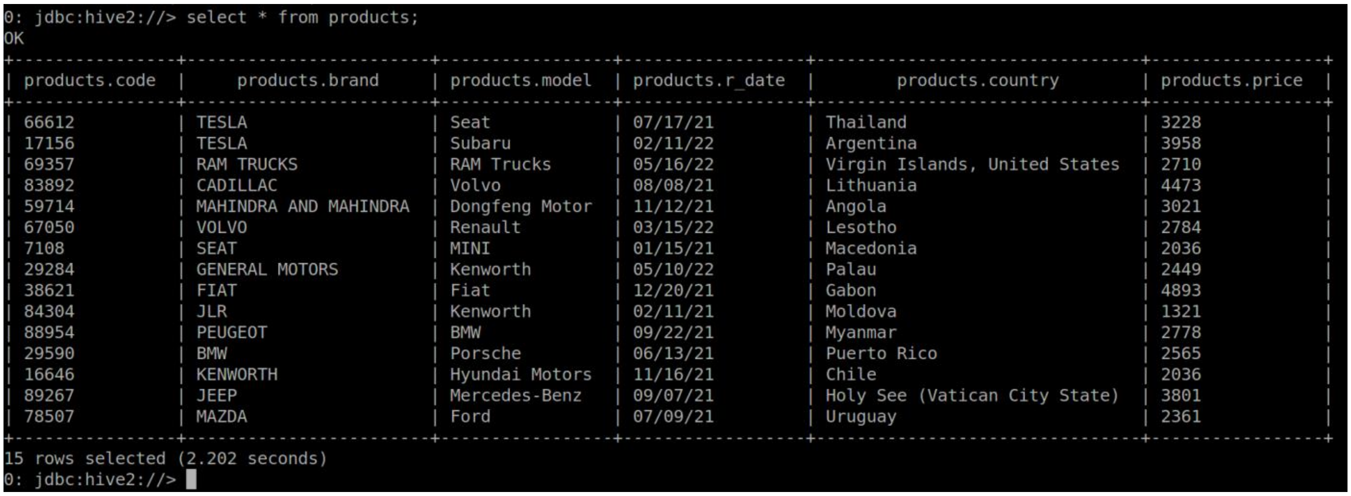
2.5. Verify and delete the products table created by script. Terminate the beeline.

```
$ beeline -u jdbc:hive2://
```

```
0: jdbc:hive2://> desc products;

0: jdbc:hive2://> select * from products;

0: jdbc:hive2://> drop table products;

0: jdbc:hive2://> !quit

(exit the beeline)

$
```

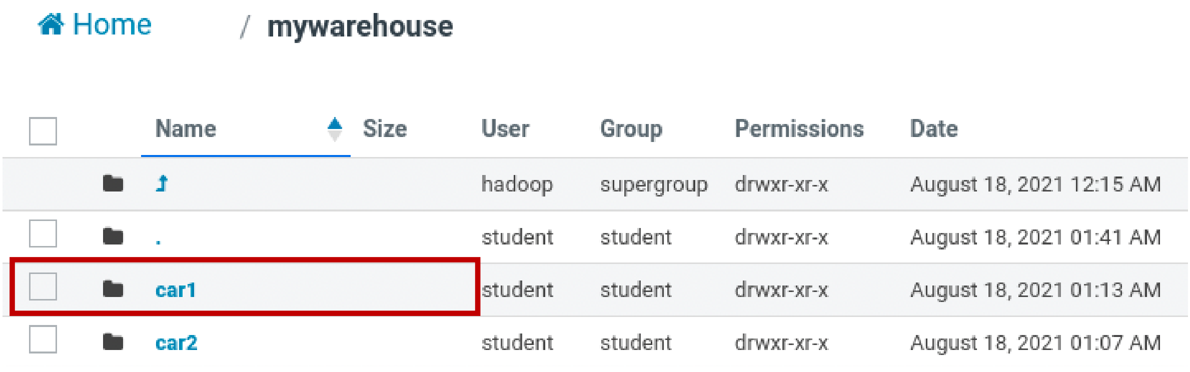


```
0: jdbc:hive2://> desc products;
OK
+-----------+------------+----------+
| col_name  | data_type  | comment  |
+-----------+------------+----------+
| code      | int        |          |
| brand     | string     |          |
| model     | string     |          |
| r_date    | string     |          |
| country   | string     |          |
| price     | int        |          |
+-----------+------------+----------+
6 rows selected (1.759 seconds)
0: jdbc:hive2://>
```



```
0: jdbc:hive2://> select * from products;
OK
+-----------------+------------------------+----------------+------------------+-------------------------------+-----------------+
| products.code   |     products.brand     | products.model | products.r_date  |       products.country        | products.price  |
+-----------------+------------------------+----------------+------------------+-------------------------------+-----------------+
| 66612           | TESLA                  | Seat           | 07/17/21         | Thailand                      | 3228            |
| 17156           | TESLA                  | Subaru         | 02/11/22         | Argentina                     | 3958            |
| 69357           | RAM TRUCKS             | RAM Trucks     | 05/16/22         | Virgin Islands, United States | 2710            |
| 83892           | CADILLAC               | Volvo          | 08/08/21         | Lithuania                     | 4473            |
| 59714           | MAHINDRA AND MAHINDRA  | Dongfeng Motor | 11/12/21         | Angola                        | 3021            |
| 67050           | VOLVO                  | Renault        | 03/15/22         | Lesotho                       | 2784            |
| 7108            | SEAT                   | MINI           | 01/15/21         | Macedonia                     | 2036            |
| 29284           | GENERAL MOTORS         | Kenworth       | 05/10/22         | Palau                         | 2449            |
| 38621           | FIAT                   | Fiat           | 12/20/21         | Gabon                         | 4893            |
| 84304           | JLR                    | Kenworth       | 02/11/21         | Moldova                       | 1321            |
| 88954           | PEUGEOT                | BMW            | 09/22/21         | Myanmar                       | 2778            |
| 29590           | BMW                    | Porsche        | 06/13/21         | Puerto Rico                   | 2565            |
| 16646           | KENWORTH               | Hyundai Motors | 11/16/21         | Chile                         | 2036            |
| 89267           | JEEP                   | Mercedes-Benz  | 09/07/21         | Holy See (Vatican City State) | 3801            |
| 78507           | MAZDA                  | Ford           | 07/09/21         | Uruguay                       | 2361            |
+-----------------+------------------------+----------------+------------------+-------------------------------+-----------------+
15 rows selected (2.202 seconds)
0: jdbc:hive2://>
```

Note: It was saved in the folder car1 where the data was pre-processed in pig. After creating this as an external table, select the table contents.



Home / mywarehouse

□	Name	Size	User	Group	Permissions	Date
	↑		hadoop	supergroup	drwxr-xr-x	August 18, 2021 12:15 AM
□	.		student	student	drwxr-xr-x	August 18, 2021 01:41 AM
□	car1		student	student	drwxr-xr-x	August 18, 2021 01:13 AM
□	car2		student	student	drwxr-xr-x	August 18, 2021 01:07 AM

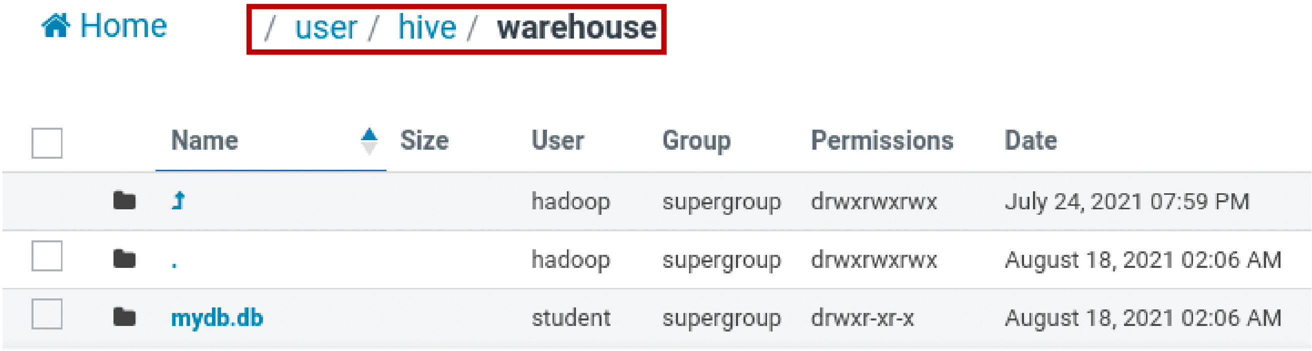
Note: When a table created as external is deleted, table information is deleted from the metastore, but car1's data remains.

3. Create a table using Sqoop's Hive import option

   3.1. In a terminal, run the following command to import the country table from MariaDB using hive-import option.

      3.1.1. First, login the beeline, and create the mydb database for lab.

```
0: jdbc:hive2://> create database mydb;
```



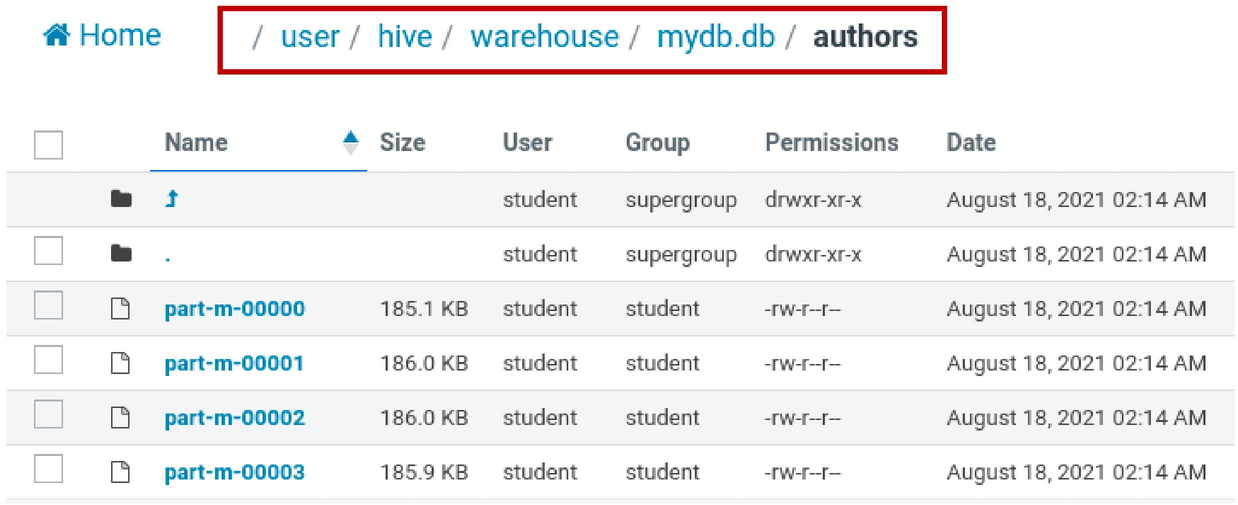
Home / user / hive / warehouse

□	Name	Size	User	Group	Permissions	Date
	↑		hadoop	supergroup	drwxrwxrwx	July 24, 2021 07:59 PM
□	.		hadoop	supergroup	drwxrwxrwx	August 18, 2021 02:06 AM
□	mydb.db		student	supergroup	drwxr-xr-x	August 18, 2021 02:06 AM

Note: The created database mydb is created in /user/hive/warehouse with the name mydb.db and is distinguished from general tables.

      3.1.2. In another terminal, run sqoop command like this:

```
$ hdfs dfs -rm -r authors

$ sqoop import --connect jdbc:mysql://localhost/labs --username student --password
student --fields-terminated-by '\t' --table authors --hive-import --hive-database
'mydb' --hive-table 'authors' --split-by id
```

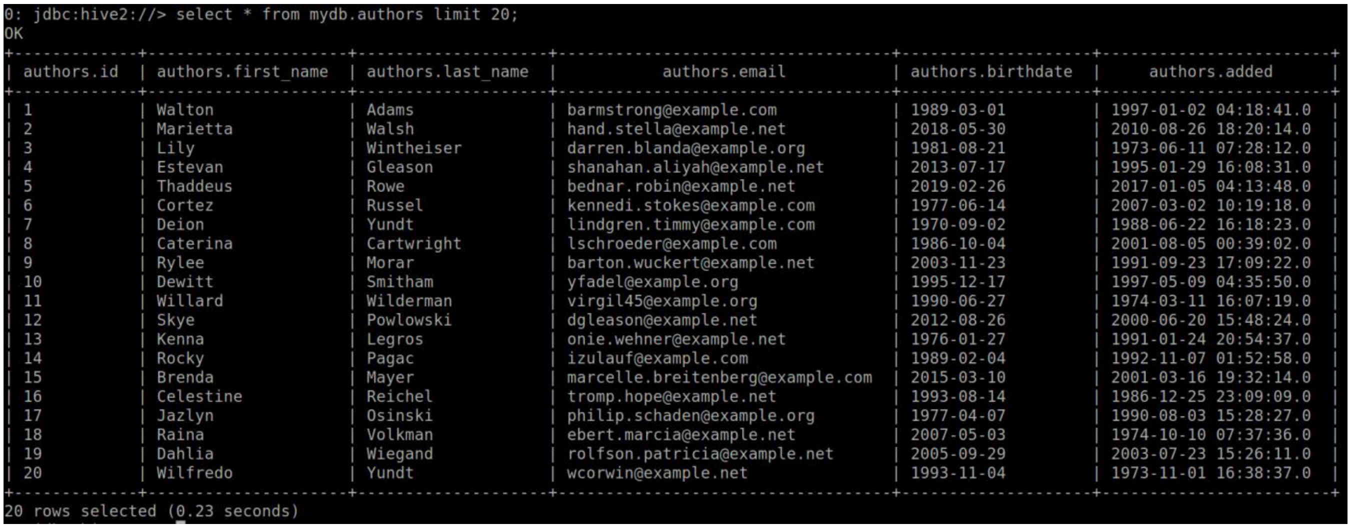


Home / user / hive / warehouse / mydb.db / authors

□		Name	Size	User	Group	Permissions	Date
				student	supergroup	drwxr-xr-x	August 18, 2021 02:14 AM
□		.		student	supergroup	drwxr-xr-x	August 18, 2021 02:14 AM
□		part-m-00000	185.1 KB	student	student	-rw-r--r--	August 18, 2021 02:14 AM
□		part-m-00001	186.0 KB	student	student	-rw-r--r--	August 18, 2021 02:14 AM
□		part-m-00002	186.0 KB	student	student	-rw-r--r--	August 18, 2021 02:14 AM
□		part-m-00003	185.9 KB	student	student	-rw-r--r--	August 18, 2021 02:14 AM

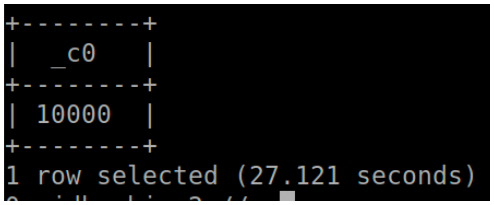
3.2. Verify that the table imported from beeline is saved correctly. 10000 records checked.

```
0: jdbc:hive2://> select * from mydb.authors;
0: jdbc:hive2://> use mydb;
0: jdbc:hive2://> select count(*) from authors;
```



```
0: jdbc:hive2://> select * from mydb.authors limit 20;
OK
+--------------+---------------------+--------------------+-----------------------------------+-------------------+------------------------+
| authors.id   | authors.first_name  | authors.last_name  |           authors.email           | authors.birthdate |     authors.added      |
+--------------+---------------------+--------------------+-----------------------------------+-------------------+------------------------+
| 1            | Walton              | Adams              | barmstrong@example.com            | 1989-03-01        | 1997-01-02 04:18:41.0  |
| 2            | Marietta            | Walsh              | hand.stella@example.net           | 2018-05-30        | 2010-08-26 18:20:14.0  |
| 3            | Lily                | Wintheiser         | darren.blanda@example.org         | 1981-08-21        | 1973-06-11 07:28:12.0  |
| 4            | Estevan             | Gleason            | shanahan.aliyah@example.net       | 2013-07-17        | 1995-01-29 16:08:31.0  |
| 5            | Thaddeus            | Rowe               | bednar.robin@example.net          | 2019-02-26        | 2017-01-05 04:13:48.0  |
| 6            | Cortez              | Russel             | kennedi.stokes@example.com        | 1977-06-14        | 2007-03-02 10:19:18.0  |
| 7            | Deion               | Yundt              | lindgren.timmy@example.com        | 1970-09-02        | 1988-06-22 16:18:23.0  |
| 8            | Caterina            | Cartwright         | lschroeder@example.com            | 1986-10-04        | 2001-08-05 00:39:02.0  |
| 9            | Rylee               | Morar              | barton.wuckert@example.net        | 2003-11-23        | 1991-09-23 17:09:22.0  |
| 10           | Dewitt              | Smitham            | yfadel@example.org                | 1995-12-17        | 1997-05-09 04:35:50.0  |
| 11           | Willard             | Wilderman          | virgil45@example.org              | 1990-06-27        | 1974-03-11 16:07:19.0  |
| 12           | Skye                | Powlowski          | dgleason@example.net              | 2012-08-26        | 2000-06-20 15:48:24.0  |
| 13           | Kenna               | Legros             | onie.wehner@example.net           | 1976-01-27        | 1991-01-24 20:54:37.0  |
| 14           | Rocky               | Pagac              | izulauf@example.com               | 1989-02-04        | 1992-11-07 01:52:58.0  |
| 15           | Brenda              | Mayer              | marcelle.breitenberg@example.com  | 2015-03-10        | 2001-03-16 19:32:14.0  |
| 16           | Celestine           | Reichel            | tromp.hope@example.net            | 1993-08-14        | 1986-12-25 23:09:09.0  |
| 17           | Jazlyn              | Osinski            | philip.schaden@example.org        | 1977-04-07        | 1990-08-03 15:28:27.0  |
| 18           | Raina               | Volkman            | ebert.marcia@example.net          | 2007-05-03        | 1974-10-10 07:37:36.0  |
| 19           | Dahlia              | Wiegand            | rolfson.patricia@example.net      | 2005-09-29        | 2003-07-23 15:26:11.0  |
| 20           | Wilfredo            | Yundt              | wcorwin@example.net               | 1993-11-04        | 1973-11-01 16:38:37.0  |
+--------------+---------------------+--------------------+-----------------------------------+-------------------+------------------------+
20 rows selected (0.23 seconds)
```

Note: This is the result of searching 20 records of authors. This is the case when the database name and table are used together.



```
+---------+
|   _c0   |
+---------+
|  10000  |
+---------+
1 row selected (27.121 seconds)
```

Note: This checks the total number of authors records.

4. Write a hive QL commands that execute the following conditions.

   4.1. Save the posts table using sqoop in mydb database, and find the total number of records.
   Hint: The --null-string and --null-non-string arguments are optional. If not specified, then the string "null" will be used.
   The --hive-drop-import-delims argument drops \n, \r, and \01 from string fields when importing to Hive.

   4.2. Import the only first_name, last_name, email and save the authors_parquet folder in parquet format in /mywarehouse of hdfs, and create an authors_parquet external table using this folder.
   Hint: sqoop import --connect jdbc:mysql://localhost/labs --username student --password student --table authors --target-dir /mywarehouse/authors_parquet --as-parquetfile

   4.3. Use alter table to rename the “first_name” column to “fname” in authors_parquet. Verify the change with DESCRIBE.

   4.4. Rename the entire table to authors_parquet2. And verify the change the folder name in /mywarehouse.

Lab 7: Handling partitioned table for performance

In this lab, you will create a table for car data that is partitioned by part number. You already built the data using pig service in previous lab. We can query the car data in the same table, but distinguish them by part numbers. It does this by creating a partition for each record.

1. Creating and Loading a static partition to table

In terminal windows,

1.1. Run the Beeline.

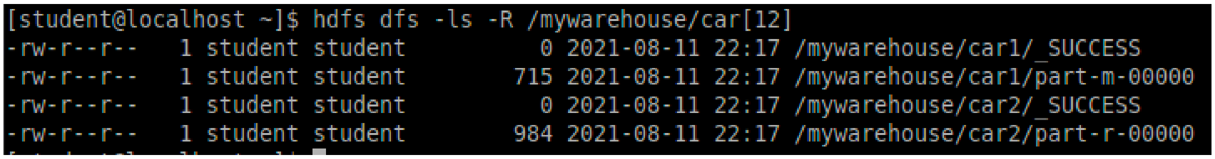
```
$ beeline -u jdbc:hive2://
```

1.2. Use the mydb database for lab.

```
0: jdbc:hive2://> use mydb;
```

1.3. Make sure that the processed data is in the following HDFS.

```
$ hdfs dfs -ls -R /mywarehouse/car[12]
```



```
[student@localhost ~]$ hdfs dfs -ls -R /mywarehouse/car[12]
-rw-r--r--   1 student student          0 2021-08-11 22:17 /mywarehouse/car1/_SUCCESS
-rw-r--r--   1 student student        715 2021-08-11 22:17 /mywarehouse/car1/part-m-00000
-rw-r--r--   1 student student          0 2021-08-11 22:17 /mywarehouse/car2/_SUCCESS
-rw-r--r--   1 student student        984 2021-08-11 22:17 /mywarehouse/car2/part-r-00000
```

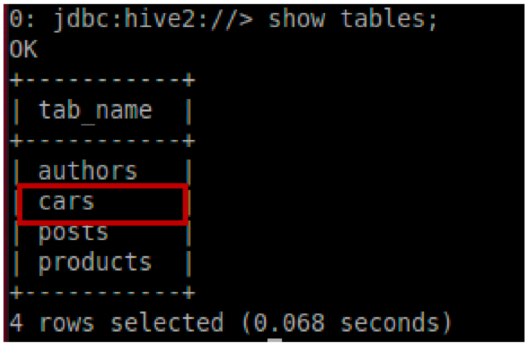
1.4. Create the table called cars with the following schema:

Column Name	Column Type
Code	Int
brand	string
l_date	string
model	string
country	string
price	int

1.4.1. The partition column is part_num (type tinyint), and field delimiter is '\t'

1.4.2. The table location is /mywarehouse/car with managed type.

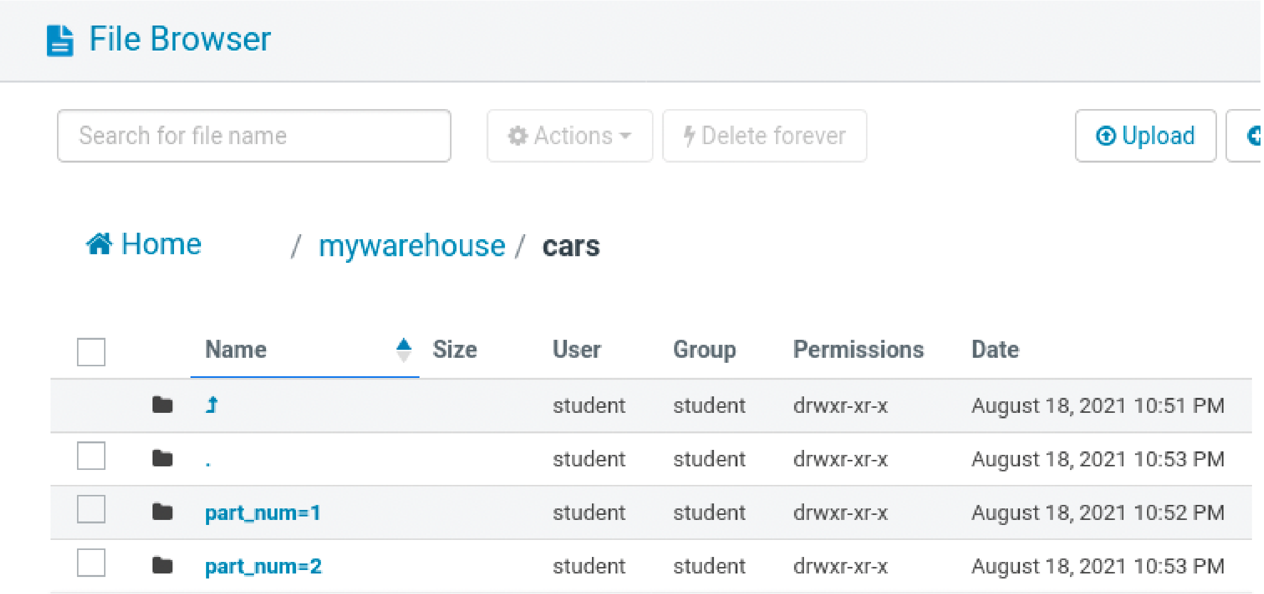
```
0: jdbc:hive2://> create table if not exists cars (
code int,
brand string,
model string,
r_date string,
country string,
price int
)
PARTITIONED BY (part_num tinyint)
row format delimited
fields terminated by '\t'
location '/mywarehouse/cars';
```



```
0: jdbc:hive2://> show tables;
OK
+-----------+
| tab_name  |
+-----------+
| authors   |
| cars      |
| posts     |
| products  |
+-----------+
4 rows selected (0.068 seconds)
```

1.4.3. Change the cars table to add two partitions. Partitions are one for part 1 and one for part 2.

```
0: jdbc:hive2://> alter table cars add partition (part_num=1);
0: jdbc:hive2://> alter table cars add partition (part_num=2);
```



Home / mywarehouse / cars

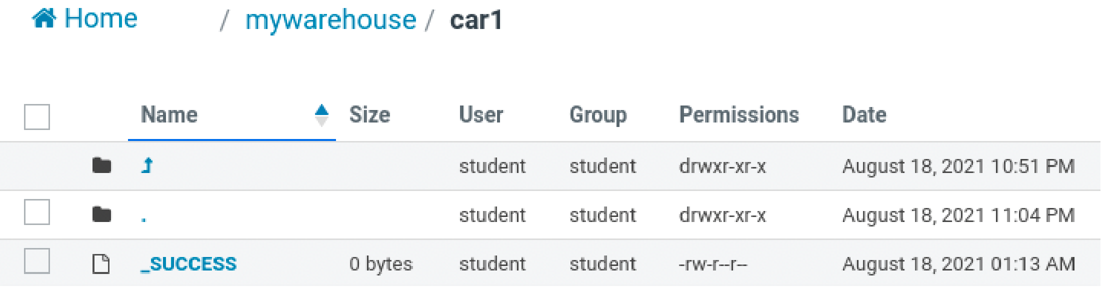
□		Name	Size	User	Group	Permissions	Date
		↥		student	student	drwxr-xr-x	August 18, 2021 10:51 PM
□		.		student	student	drwxr-xr-x	August 18, 2021 10:53 PM
□		part_num=1		student	student	drwxr-xr-x	August 18, 2021 10:52 PM
□		part_num=2		student	student	drwxr-xr-x	August 18, 2021 10:53 PM

1.4.4. Load the data from /mywarehouse/car1 into the part 1 partition, and load the data from /mywarehouse/car2 into the part 2 partition.

```
0: jdbc:hive2://> load data inpath '/mywarehouse/car1' into table cars
partition(part_num=1);
```

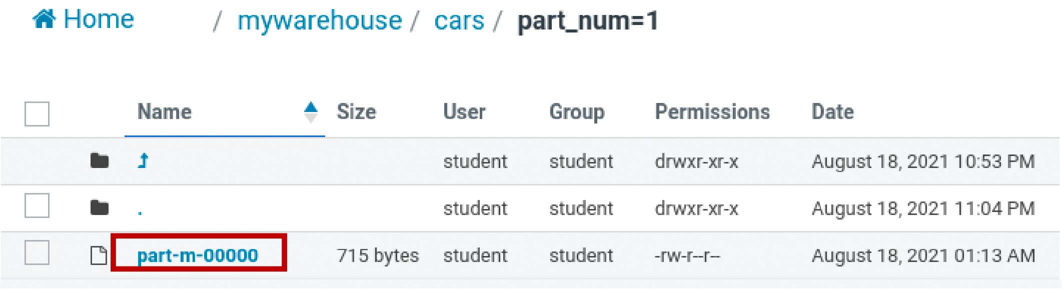
When the load command is executed, the file in the existing car1 is moved to the cars/part_num=1 folder.

1.4.5. Verify the car1 and cars folder.



Home / mywarehouse / car1

□		Name	Size	User	Group	Permissions	Date
		↥		student	student	drwxr-xr-x	August 18, 2021 10:51 PM
□		.		student	student	drwxr-xr-x	August 18, 2021 11:04 PM
□		_SUCCESS	0 bytes	student	student	-rw-r--r--	August 18, 2021 01:13 AM

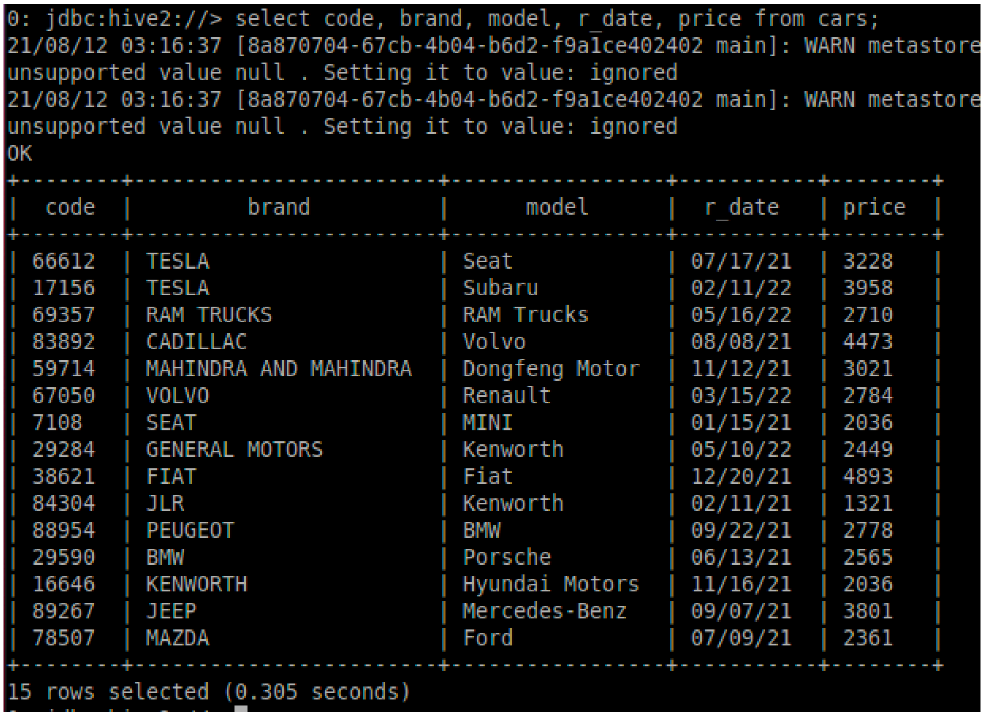


🏠 Home / mywarehouse / cars / part_num=1

□		Name ▲	Size	User	Group	Permissions	Date
	📁	↥		student	student	drwxr-xr-x	August 18, 2021 10:53 PM
□	📁	.		student	student	drwxr-xr-x	August 18, 2021 11:04 PM
□	🗋	part-m-00000	715 bytes	student	student	-rw-r--r--	August 18, 2021 01:13 AM

1.4.6. Show the data of car1 using select statement

```
0: jdbc:hive2://> select code, brand, model, r_date, price from cars;
```

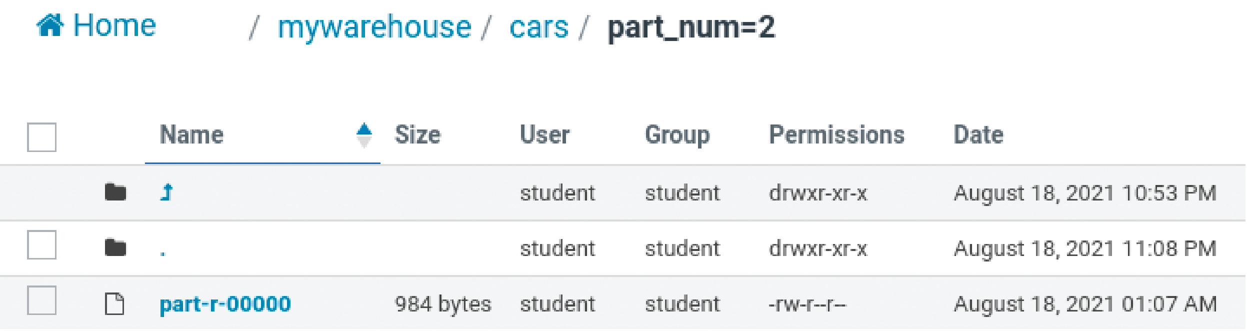


```
0: jdbc:hive2://> select code, brand, model, r_date, price from cars;
21/08/12 03:16:37 [8a870704-67cb-4b04-b6d2-f9a1ce402402 main]: WARN metastore
unsupported value null . Setting it to value: ignored
21/08/12 03:16:37 [8a870704-67cb-4b04-b6d2-f9a1ce402402 main]: WARN metastore
unsupported value null . Setting it to value: ignored
OK
+--------+------------------------+-----------------+-----------+--------+
|  code  |         brand          |      model      |  r_date   | price  |
+--------+------------------------+-----------------+-----------+--------+
| 66612  | TESLA                  | Seat            | 07/17/21  | 3228   |
| 17156  | TESLA                  | Subaru          | 02/11/22  | 3958   |
| 69357  | RAM TRUCKS             | RAM Trucks      | 05/16/22  | 2710   |
| 83892  | CADILLAC               | Volvo           | 08/08/21  | 4473   |
| 59714  | MAHINDRA AND MAHINDRA  | Dongfeng Motor  | 11/12/21  | 3021   |
| 67050  | VOLVO                  | Renault         | 03/15/22  | 2784   |
| 7108   | SEAT                   | MINI            | 01/15/21  | 2036   |
| 29284  | GENERAL MOTORS         | Kenworth        | 05/10/22  | 2449   |
| 38621  | FIAT                   | Fiat            | 12/20/21  | 4893   |
| 84304  | JLR                    | Kenworth        | 02/11/21  | 1321   |
| 88954  | PEUGEOT                | BMW             | 09/22/21  | 2778   |
| 29590  | BMW                    | Porsche         | 06/13/21  | 2565   |
| 16646  | KENWORTH               | Hyundai Motors  | 11/16/21  | 2036   |
| 89267  | JEEP                   | Mercedes-Benz   | 09/07/21  | 3801   |
| 78507  | MAZDA                  | Ford            | 07/09/21  | 2361   |
+--------+------------------------+-----------------+-----------+--------+
15 rows selected (0.305 seconds)
```

Note: The warning message here can be ignored.

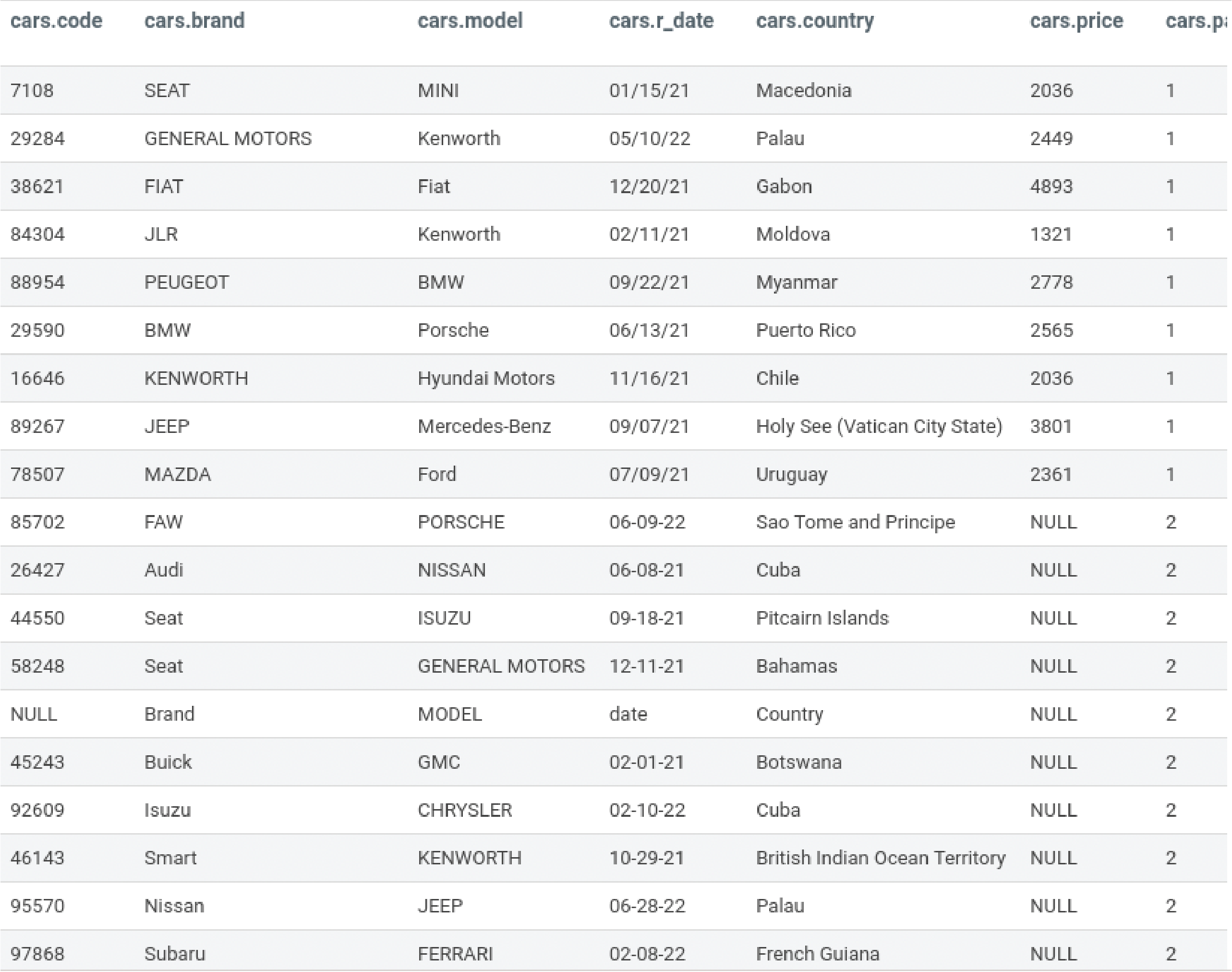
1.4.7. Move the data in car2 to part_num=2 using hdfs mv instead of the load command and verify that the data looks the same as 1.4.6.

```
0: jdbc:hive2://> load data inpath '/mywarehouse/car2' into table cars
partition(part_num=2)
```



Home / mywarehouse / cars / part_num=2

☐		Name	Size	User	Group	Permissions	Date
	📁	↑		student	student	drwxr-xr-x	August 18, 2021 10:53 PM
☐	📁	.		student	student	drwxr-xr-x	August 18, 2021 11:08 PM
☐	🗋	part-r-00000	984 bytes	student	student	-rw-r--r--	August 18, 2021 01:07 AM



cars.code	cars.brand	cars.model	cars.r_date	cars.country	cars.price	cars.pa
7108	SEAT	MINI	01/15/21	Macedonia	2036	1
29284	GENERAL MOTORS	Kenworth	05/10/22	Palau	2449	1
38621	FIAT	Fiat	12/20/21	Gabon	4893	1
84304	JLR	Kenworth	02/11/21	Moldova	1321	1
88954	PEUGEOT	BMW	09/22/21	Myanmar	2778	1
29590	BMW	Porsche	06/13/21	Puerto Rico	2565	1
16646	KENWORTH	Hyundai Motors	11/16/21	Chile	2036	1
89267	JEEP	Mercedes-Benz	09/07/21	Holy See (Vatican City State)	3801	1
78507	MAZDA	Ford	07/09/21	Uruguay	2361	1
85702	FAW	PORSCHE	06-09-22	Sao Tome and Principe	NULL	2
26427	Audi	NISSAN	06-08-21	Cuba	NULL	2
44550	Seat	ISUZU	09-18-21	Pitcairn Islands	NULL	2
58248	Seat	GENERAL MOTORS	12-11-21	Bahamas	NULL	2
NULL	Brand	MODEL	date	Country	NULL	2
45243	Buick	GMC	02-01-21	Botswana	NULL	2
92609	Isuzu	CHRYSLER	02-10-22	Cuba	NULL	2
46143	Smart	KENWORTH	10-29-21	British Indian Ocean Territory	NULL	2
95570	Nissan	JEEP	06-28-22	Palau	NULL	2
97868	Subaru	FERRARI	02-08-22	French Guiana	NULL	2

Note: We checked the entire data including part_num=2 as above.

1.4.8. Verify the data for both number of part were correctly loaded by counting the records for each:

```
0: jdbc:hive2://> select count(*) from cars where part_num=1;
```

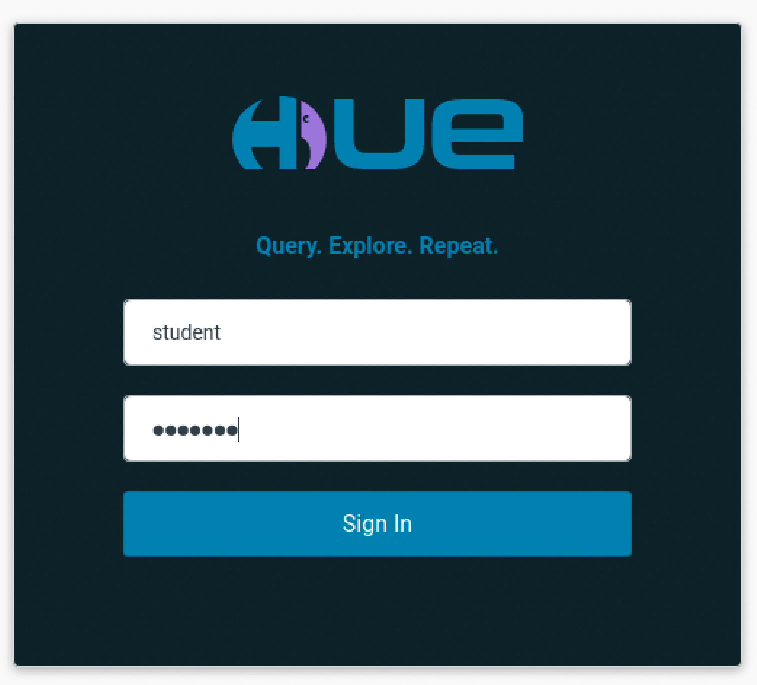
```
0: jdbc:hive2://> select count(*) from cars where part_num=2;
```

Question: What is the number of records in each partition?

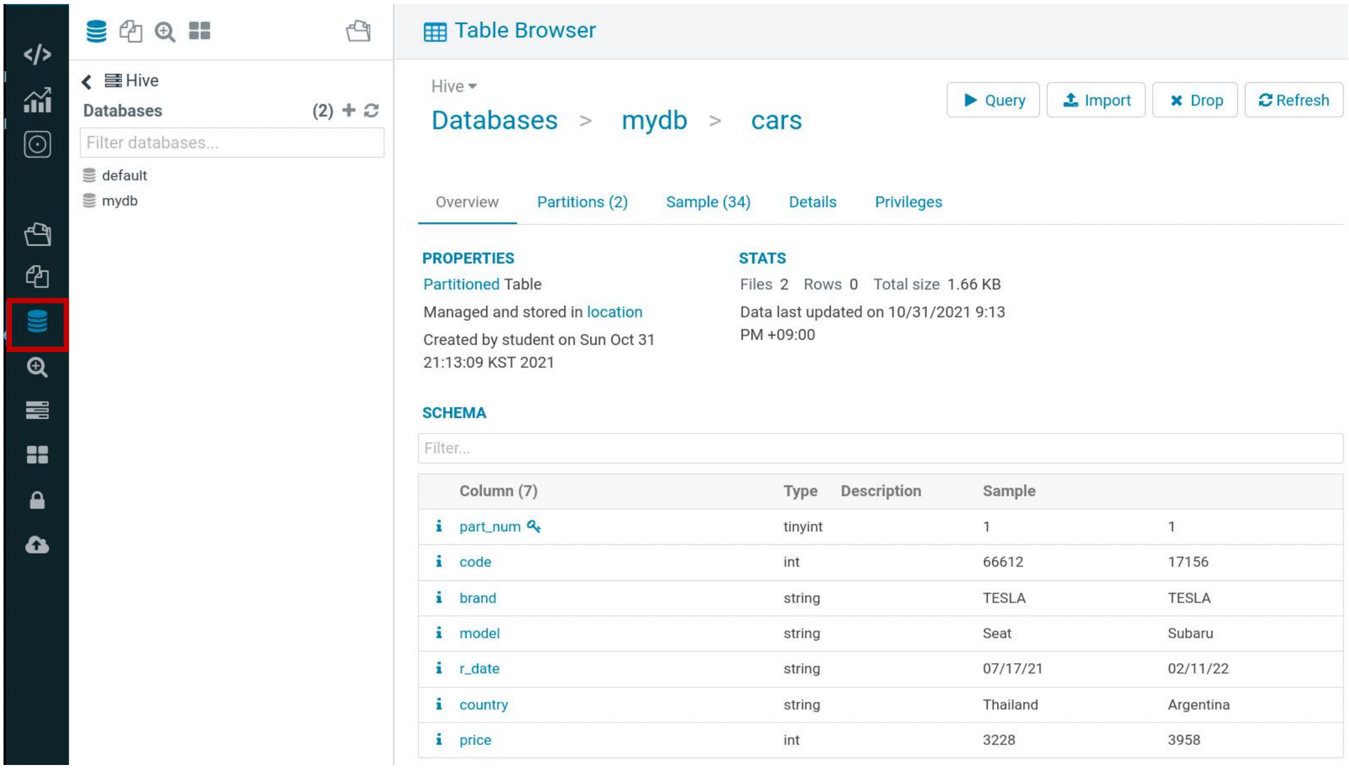
2. Working with Table in hue.

In this lab, you will practice using Hue query to execute queries. And verify the databases and tables using the Table browser in Hue, and practice creating a new database and table using it.

2.1. In firebox url, type the Hue page http://localhost:8888. The id is student, and password is student. Clink the "sign in" to enter hue.



2.2. Click the Table browser in the left toolbar.



PROPERTIES

Partitioned Table

Managed and stored in location

Created by student on Sun Oct 31 21:13:09 KST 2021

STATS

Files 2 Rows 0 Total size 1.66 KB

Data last updated on 10/31/2021 9:13 PM +09:00

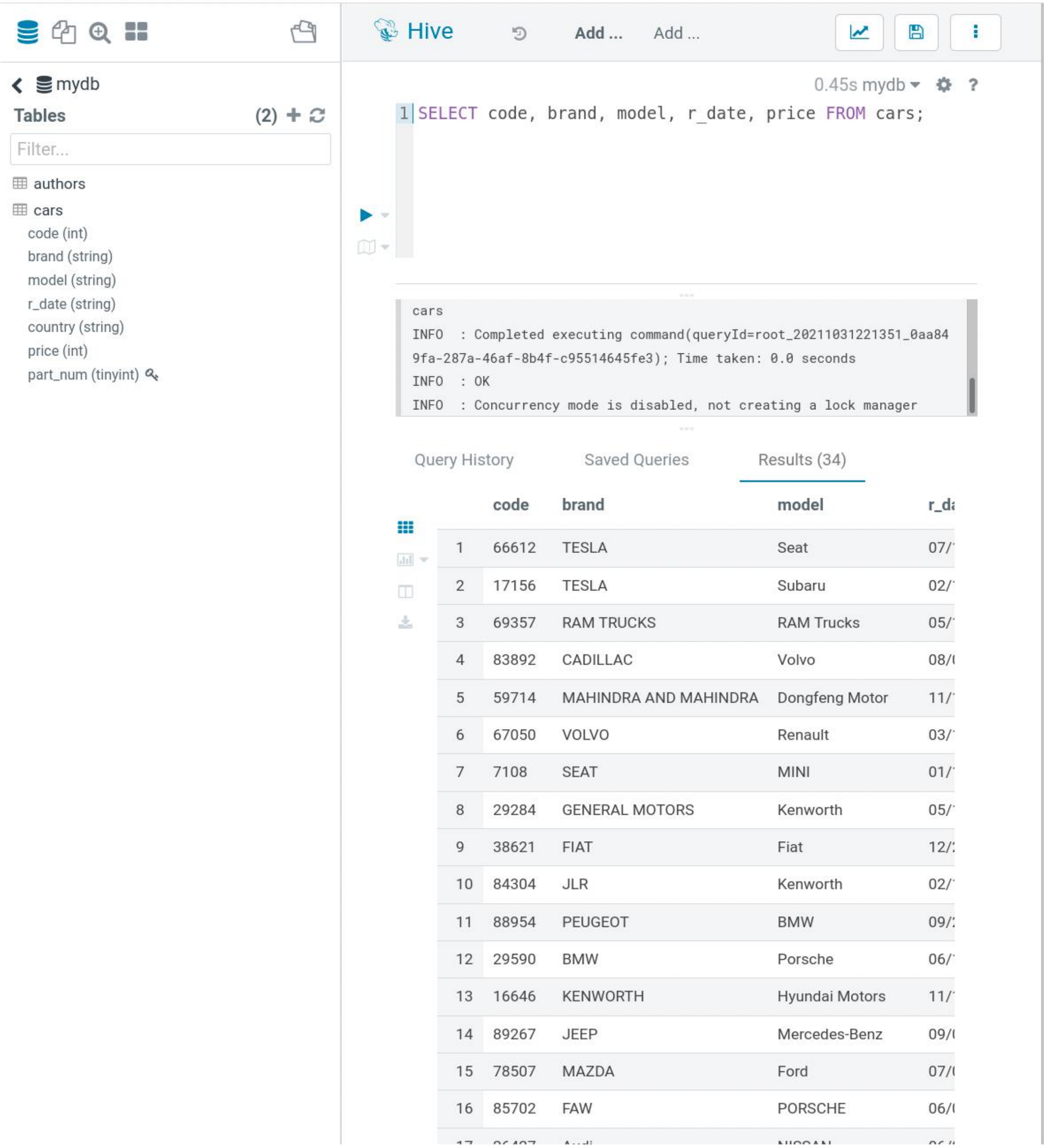
SCHEMA

Column (7)	Type	Description	Sample	
part_num	tinyint		1	1
code	int		66612	17156
brand	string		TESLA	TESLA
model	string		Seat	Subaru
r_date	string		07/17/21	02/11/22
country	string		Thailand	Argentina
price	int		3228	3958

Note: The table browser displays the mydb database name and a list of tables.

2.3. Execute the command executed in beeline in the hive editor and verify the result.

2.3.1. select code, brand, model, r_date, price from cars;



authors

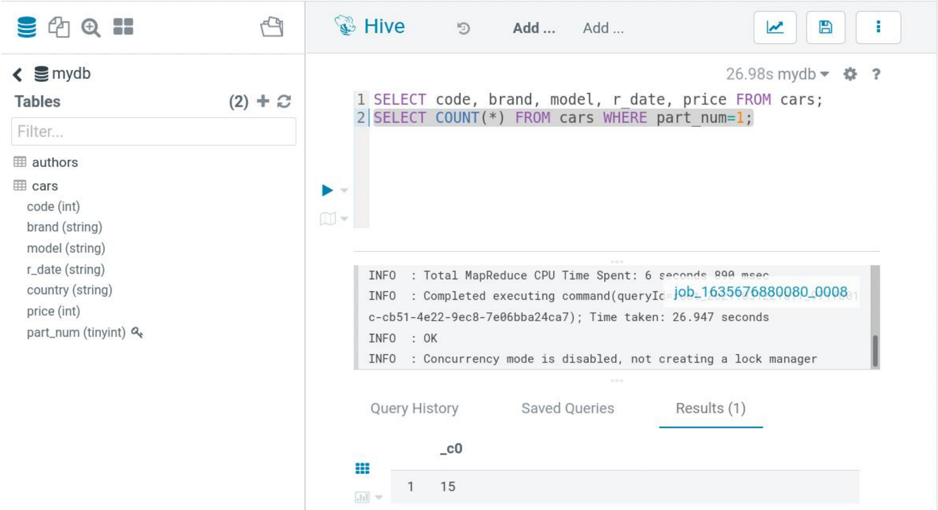
cars
- code (int)
- brand (string)
- model (string)
- r_date (string)
- country (string)
- price (int)
- part_num (tinyint)

```
SELECT code, brand, model, r_date, price FROM cars;
```

```
cars
INFO  : Completed executing command(queryId=root_20211031221351_0aa84
9fa-287a-46af-8b4f-c95514645fe3); Time taken: 0.0 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
```

	code	brand	model	r_da
1	66612	TESLA	Seat	07/
2	17156	TESLA	Subaru	02/
3	69357	RAM TRUCKS	RAM Trucks	05/
4	83892	CADILLAC	Volvo	08/
5	59714	MAHINDRA AND MAHINDRA	Dongfeng Motor	11/
6	67050	VOLVO	Renault	03/
7	7108	SEAT	MINI	01/
8	29284	GENERAL MOTORS	Kenworth	05/
9	38621	FIAT	Fiat	12/
10	84304	JLR	Kenworth	02/
11	88954	PEUGEOT	BMW	09/
12	29590	BMW	Porsche	06/
13	16646	KENWORTH	Hyundai Motors	11/
14	89267	JEEP	Mercedes-Benz	09/
15	78507	MAZDA	Ford	07/
16	85702	FAW	PORSCHE	06/
17	[illegible]	Audi	NISSAN	06/

2.3.2. select count(*) from cars where part_num=1;

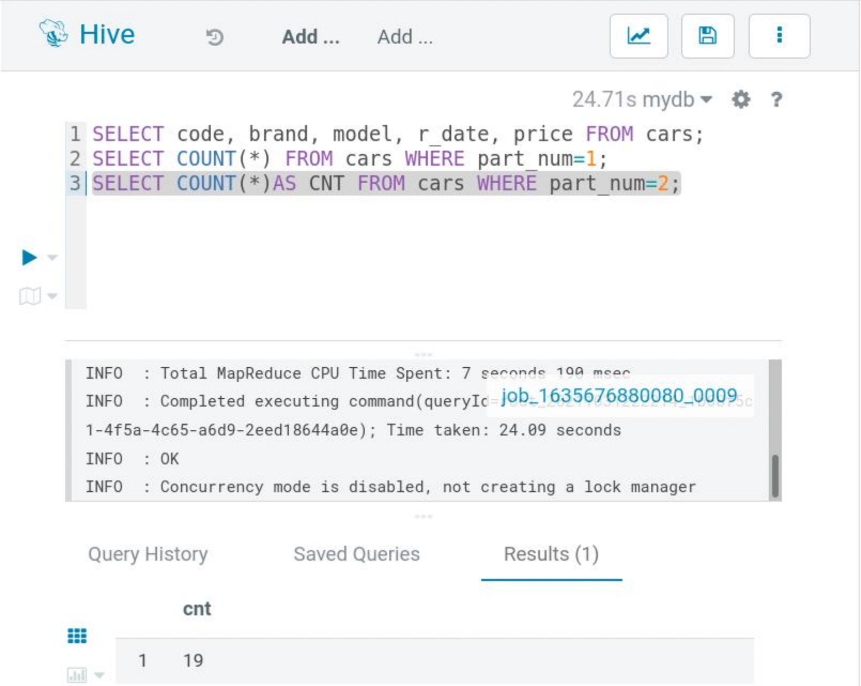


```
SELECT code, brand, model, r_date, price FROM cars;
SELECT COUNT(*) FROM cars WHERE part_num=1;
```

```
INFO  : Total MapReduce CPU Time Spent: 6 seconds 890 msec
INFO  : Completed executing command(queryIc job_1635676880080_0008
c-cb51-4e22-9ec8-7e06bba24ca7); Time taken: 26.947 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
```

	_c0
1	15

2.3.3. select count(*) as CNT from cars where part_num=2;

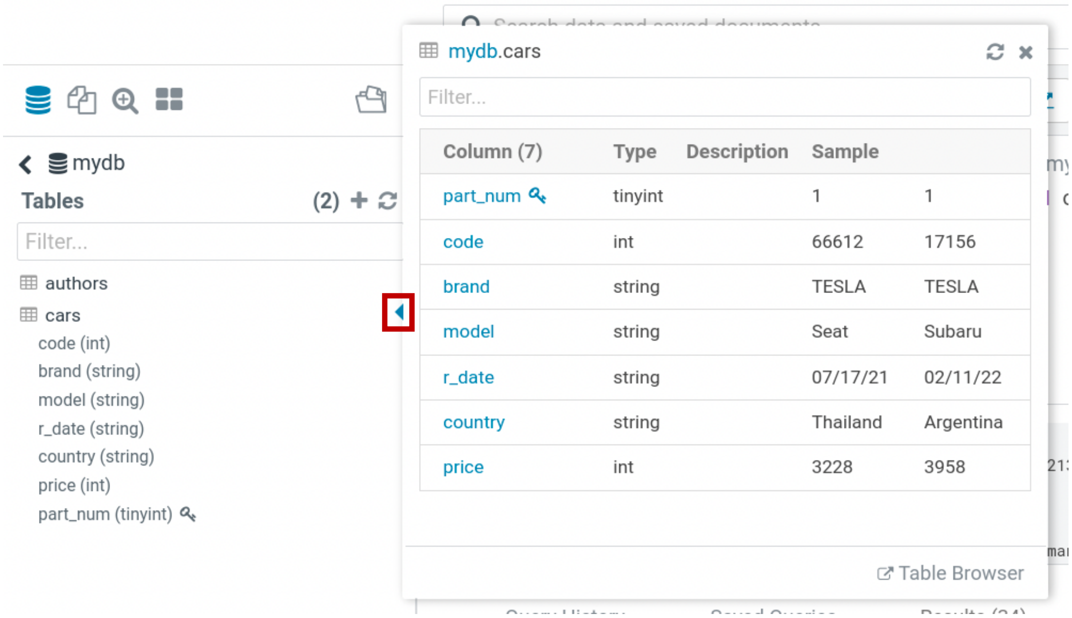


```
SELECT code, brand, model, r_date, price FROM cars;
SELECT COUNT(*) FROM cars WHERE part_num=1;
SELECT COUNT(*)AS CNT FROM cars WHERE part_num=2;
```

```
INFO  : Total MapReduce CPU Time Spent: 7 seconds 190 msec
INFO  : Completed executing command(queryIc job_1635676880080_0009
1-4f5a-4c65-a6d9-2eed18644a0e); Time taken: 24.09 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
```

	cnt
1	19

2.3.4. You can check the schema information and sample data contents by clicking "i" next to the cars table in the left menu.



Column (7)	Type	Description	Sample	
part_num	tinyint		1	1
code	int		66612	17156
brand	string		TESLA	TESLA
model	string		Seat	Subaru
r_date	string		07/17/21	02/11/22
country	string		Thailand	Argentina
price	int		3228	3958

2.4. There are two problems with the data in cars table. One is that the price is null in part_num=2, another is that the sub-title is included in the middle of the data such as “Brand, MODEL…”. Writing a query that solves these problems.

2.5. Creating queries on cars for the following information.

2.5.1. Model of TESLA

2.5.2. Code, Model and Brand of all entries in the cars

2.5.3. Brand name in alphabetical order

2.5.4. Brand, model, r_date, and price, ordered by price, descending

2.5.5. Same as above but ordered ascending

2.5.6. Brand with a price less than $2000.00

2.5.7. Brand with a price between $3000.00 and $4000.00

2.5.8. All records with model names MINI, BWM, Porsche.

2.5.9. Name the Brand with the cheapest price

2.5.10. Name the Brand with the second expensive price

2.5.11. How many records do not have overlapping brands?

2.5.12. What is the average price per brand?

Lab 8: Working with complex Data type

In this lab, you will run a complex data type and join operation.

1. Working with Relation Database

   1.1. First, import all tables in the MariaDB labs database. The table imported as an external table is created as a hive table.

   1.1.1. Create id column and set it as unique Key

   1.1.2. Create continent_name to hold the name of the continents

   1.1.3. Add the following continents in the order shown:
   Africa
   Asia
   Europe
   North America
   South America

   1.2. Modify the countries table

   1.2.1. Remove all the items from the table

   1.2.2. Remove the continents table we added in the previous lab

   1.2.3. Add the following information to the countries table

name	capital	population	continent_id
USA	Washington D.C.	333,098,437	match to North America
Germany	Berlin	84,073,352	match to Europe
France	Paris	65,426,179	match to Europe
Korea	Seoul	51,305,186	match to Asia
Mexico	Mexico City	130,262,216	match to South America
Egypt	Cairo	104,258,327	match to Africa

2. Creating multiple join tables

Currently, the orders table has many repeated information. If you examine the table carefully, there are four sections of information that can be identified. They are:

- Customer Information
  - Customer Name
  - Customer email

- Order information
    - Customer who ordered
    - Status of order (completed, in progress, placed)
- Order items information
    - Which order is this for?
    - List of products ordered
- Products information
    - name of product
    - cost
    - product type (sandwich, side, drink)
    - redeem_points

We have added the status of the order so that system can use to help the management keep track of all the orders.

2.1. Create customer table

2.1.1. Needs to have a unique Key

2.1.2. Customer name

2.1.3. Customer email

2.2. Create orders table

2.2.1. Needs to have a unique Key

2.2.2. Needs to link to the customer unique Key

2.2.3. An order status column

2.3. Create order_items table

2.3.1. Needs to have a unique Key

2.3.2. Needs to link to the order unique Key

2.3.3. Needs to link to the products unique Key for each product ordered

2.4. Create a products table

2.4.1. Needs to have a unique Key

2.4.2. Product name

2.4.3. Product cost

2.4.4. Amount of redeem point earned for this product

Lab 9: Complex Queries

1. Create complex queries by joining multiple tables

   1.1. Create joined queries from countries and continent to return the following information:

   1.1.1. List of country names and the associated continent

   1.1.2. List of all countries in Asia, with names of their capital

   1.2. Create joined queries from customer, order, order_item and product to return the following information:

   1.2.1. All orders, including the customer who ordered, and all the products purchased

   1.2.2. Any order that does not include a sandwich in it. Show the order id.

   1.2.3. Who was the customer who made the order without the sandwich?

   1.2.4. Total cost of all of Shaun Silverman's orders in our records

   1.2.5. For every product, total number of times that product has been ordered..

Lab 10: Working with Apache Impala

1. Setting up the lab environment

Impala and Kudu are resource intensive. Further, Kudu cannot be easily setup in a standalone Hadoop cluster such as in our Virtual Box environment. To overcome this limitation, we shall follow the Apache Kudu Quickstart guidelines (https://kudu.apache.org/docs/quickstart.html) adapted for Windows environment along with a Docker image of Impala to connect to the Kudu storage.

1.1. Install Docker for Windows

1.1.1. Either search for "install docker desktop for windows" on your browser or go directly to https://docs.docker.com/docker-for-windows/install/

1.1.2. Follow the instructions to install Docker Desktop for Windows.

1.1.3. At the end of the installation, you will be asked to install WSL 2 backend. Please install this.

1.1.4. Start Docker Desktop for Windows

1.2. Install Git for Windows

Note: You have previously installed Git in the AWS S3 Storage lab in Chapter 4 Unit 1. Unless you have removed and uninstalled the program, you may skip this step.

1.2.1. Either search for "install git for windows " or go directly to https://git-scm.com/download/win

1.2.2. You may choose either the 32-bit or 64-bit version. Whenever possible, it is better to choose the 64-bit version but this will depend on your Windows version.

1.3. Install Apache Impala Quickstart

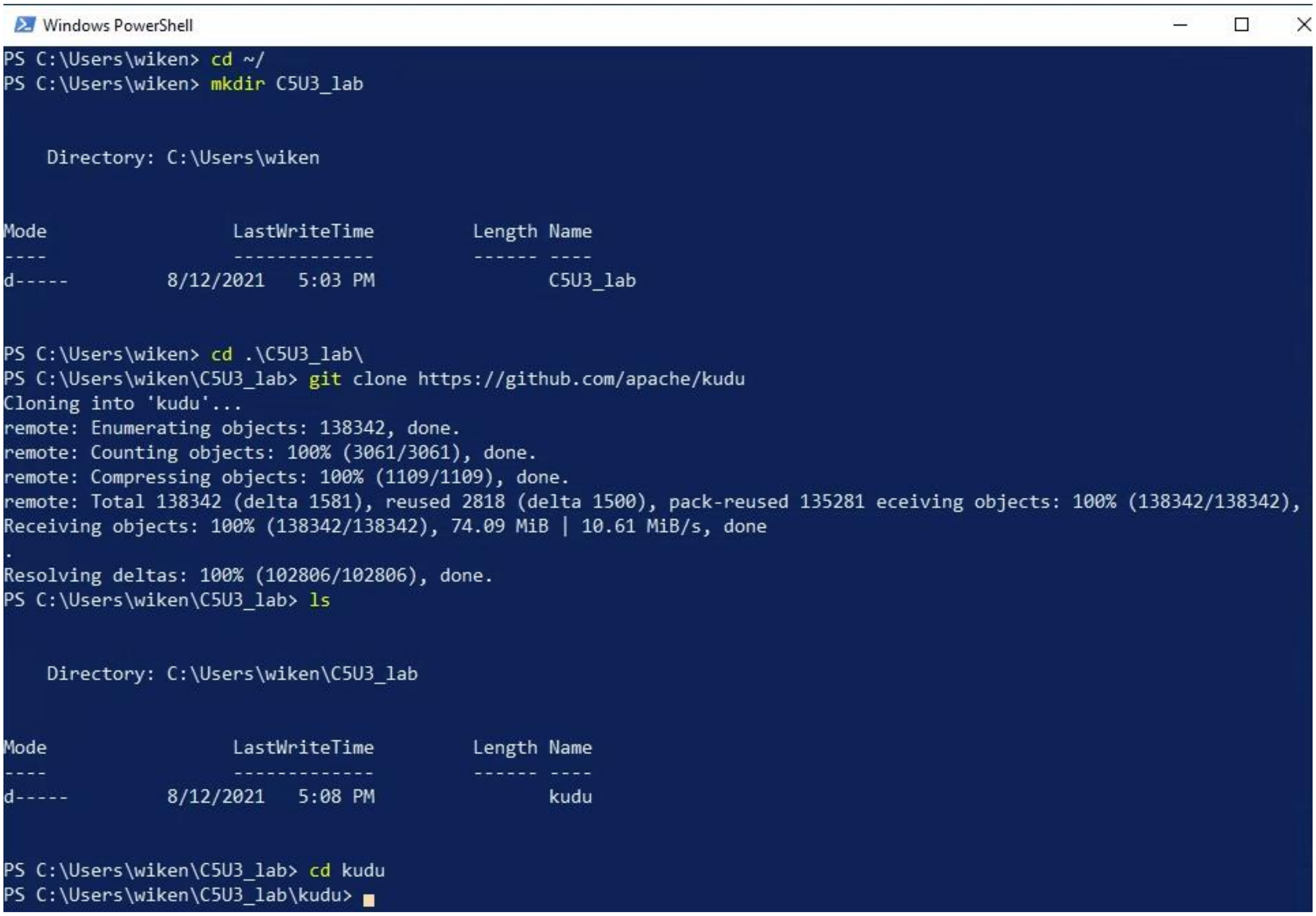
1.3.1. From Windows search, type "powershell" and start a new Windows PowerShell session

1.3.2. From the home directory on your Windows computer, create and navigate into C5U3_lab folder

1.3.3. Clone the https://github.com/apache/kudu git project

1.3.4. A new "kudu" directory will be created. Navigate to that directory

```
cd ~
mkdir C5U3_lab
cd C5U3_lab
git clone https://github.com/apache/kudu
ls
cd kudu›
```



```
PS C:\Users\wiken> cd ~/
PS C:\Users\wiken> mkdir C5U3_lab


    Directory: C:\Users\wiken


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         8/12/2021   5:03 PM                C5U3_lab


PS C:\Users\wiken> cd .\C5U3_lab\
PS C:\Users\wiken\C5U3_lab> git clone https://github.com/apache/kudu
Cloning into 'kudu'...
remote: Enumerating objects: 138342, done.
remote: Counting objects: 100% (3061/3061), done.
remote: Compressing objects: 100% (1109/1109), done.
remote: Total 138342 (delta 1581), reused 2818 (delta 1500), pack-reused 135281 eceiving objects: 100% (138342/138342),
Receiving objects: 100% (138342/138342), 74.09 MiB | 10.61 MiB/s, done
.
Resolving deltas: 100% (102806/102806), done.
PS C:\Users\wiken\C5U3_lab> ls


    Directory: C:\Users\wiken\C5U3_lab


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         8/12/2021   5:08 PM                kudu


PS C:\Users\wiken\C5U3_lab> cd kudu
PS C:\Users\wiken\C5U3_lab\kudu>
```

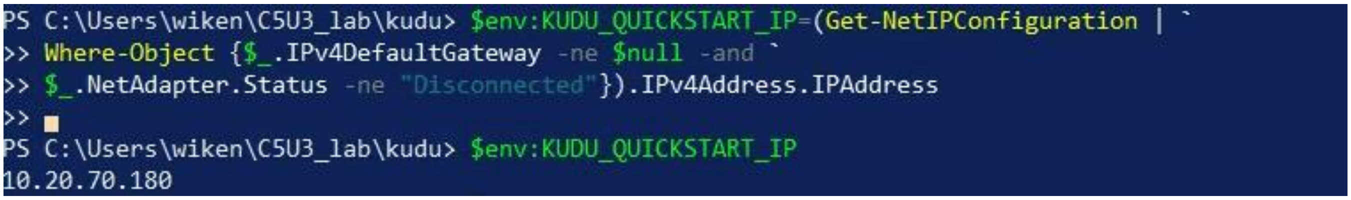
1.4. Start the Kudu docker containers

1.4.1. Setup the KUDU_QUICKSTART_IP environmental variable with the following command. In PowerShell the back tick (`) is used to continue the command on the next line.

```
$env:KUDU_QUICKSTART_IP=(Get-NetIPConfiguration | `
Where-Object {$_.IPv4DefaultGateway -ne $null -and `
$_.NetAdapter.Status -ne "Disconnected"}).IPv4Address.IPAddress
```

1.4.2. Check to make sure the variable has been set properly

```
$env:KUDU_QUICKSTART_IP
```



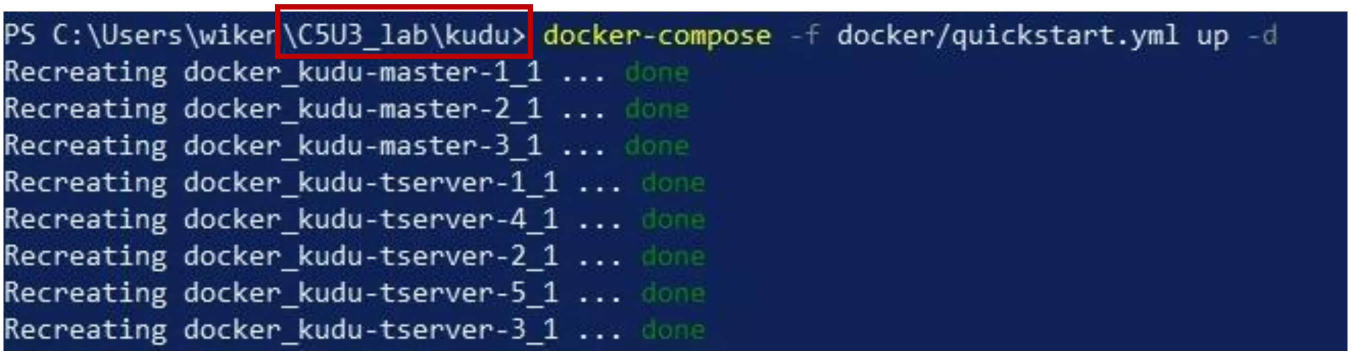
```
PS C:\Users\wiken\C5U3_lab\kudu> $env:KUDU_QUICKSTART_IP=(Get-NetIPConfiguration | `
>> Where-Object {$_.IPv4DefaultGateway -ne $null -and `
>> $_.NetAdapter.Status -ne "Disconnected"}).IPv4Address.IPAddress
>>
PS C:\Users\wiken\C5U3_lab\kudu> $env:KUDU_QUICKSTART_IP
10.20.70.180
```

1.5. Bring up the docker Kudu cluster using the provided YAML configuration file

CAUTION: Docker desktop for Windows must be running and you must be in the kudu directory before executing the following command.

1.5.1. Run the following command from PowerShell:

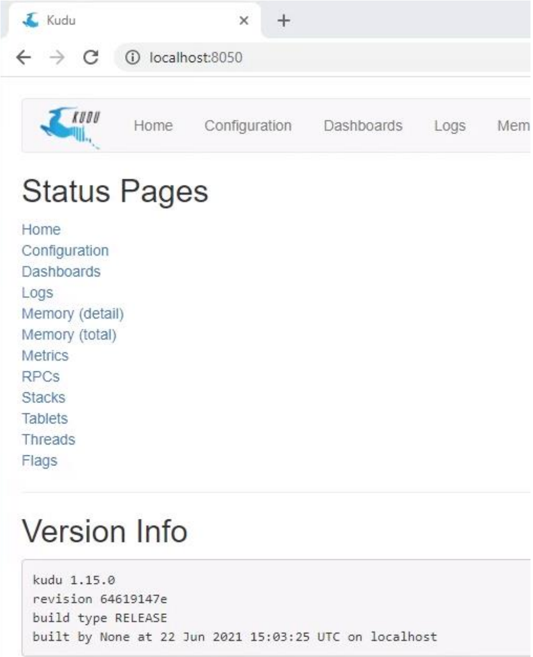
```
docker-compose -f docker/quickstart.yml up -d
```



```
PS C:\Users\wiken\C5U3_lab\kudu> docker-compose -f docker/quickstart.yml up -d
Recreating docker_kudu-master-1_1 ... done
Recreating docker_kudu-master-2_1 ... done
Recreating docker_kudu-master-3_1 ... done
Recreating docker_kudu-tserver-1_1 ... done
Recreating docker_kudu-tserver-4_1 ... done
Recreating docker_kudu-tserver-2_1 ... done
Recreating docker_kudu-tserver-5_1 ... done
Recreating docker_kudu-tserver-3_1 ... done
```

NOTE: When you first create the Kudu cluster, the displayed text will say "Creating" instead of "Recreating"

1.5.2. Check to make sure the Kudu service is up by browsing to localhost:8050



Status Pages

Home
Configuration
Dashboards
Logs
Memory (detail)
Memory (total)
Metrics
RPCs
Stacks
Tablets
Threads
Flags

Version Info

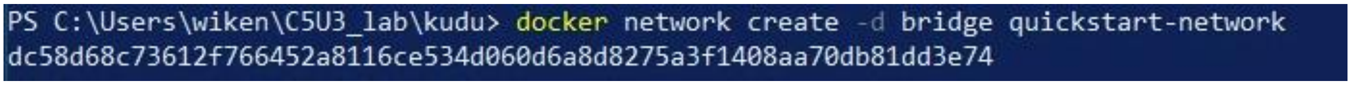
```
kudu 1.15.0
revision 64619147e
build type RELEASE
built by None at 22 Jun 2021 15:03:25 UTC on localhost
```

1.6. Create a network for Impala and Kudu to communicate

1.6.1. Use docker network command to create a bridge network

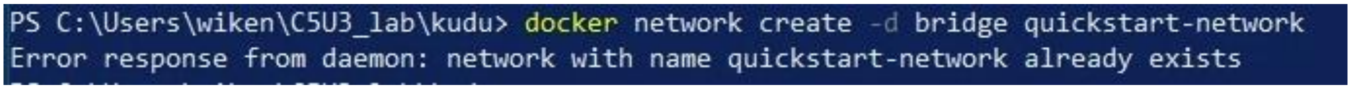
NOTE: The Network id generated will be different on your machine. If you have already attempted this lab before, you may get a message that the quickstart-network already exists. You may ignore the message and continue

```
docker network create -d bridge quickstart-network
```



```
PS C:\Users\wiken\C5U3_lab\kudu> docker network create -d bridge quickstart-network
dc58d68c73612f766452a8116ce534d060d6a8d8275a3f1408aa70db81dd3e74
```

OR



```
PS C:\Users\wiken\C5U3_lab\kudu> docker network create -d bridge quickstart-network
Error response from daemon: network with name quickstart-network already exists
```

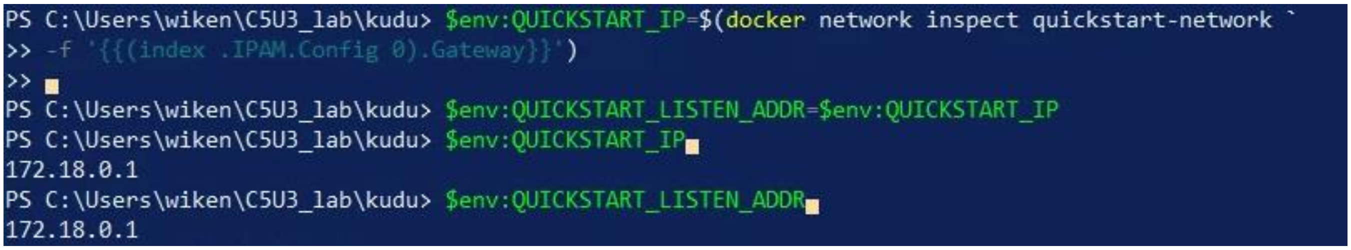
1.7. Install Apache Impala docker image

1.7.1. Set up environmental variables for Impala to use with the following two commands.

```
$env:QUICKSTART_IP=$(docker network inspect quickstart-network `
-f '{{(index .IPAM.Config 0).Gateway}}')
```

```
$env:QUICKSTART_LISTEN_ADDR=$env:QUICKSTART_IP
```

1.7.2. Check that the environmental variables have been created



```
PS C:\Users\wiken\C5U3_lab\kudu> $env:QUICKSTART_IP=$(docker network inspect quickstart-network `
>> -f '{{(index .IPAM.Config 0).Gateway}}')
>>
PS C:\Users\wiken\C5U3_lab\kudu> $env:QUICKSTART_LISTEN_ADDR=$env:QUICKSTART_IP
PS C:\Users\wiken\C5U3_lab\kudu> $env:QUICKSTART_IP
172.18.0.1
PS C:\Users\wiken\C5U3_lab\kudu> $env:QUICKSTART_LISTEN_ADDR
172.18.0.1
```

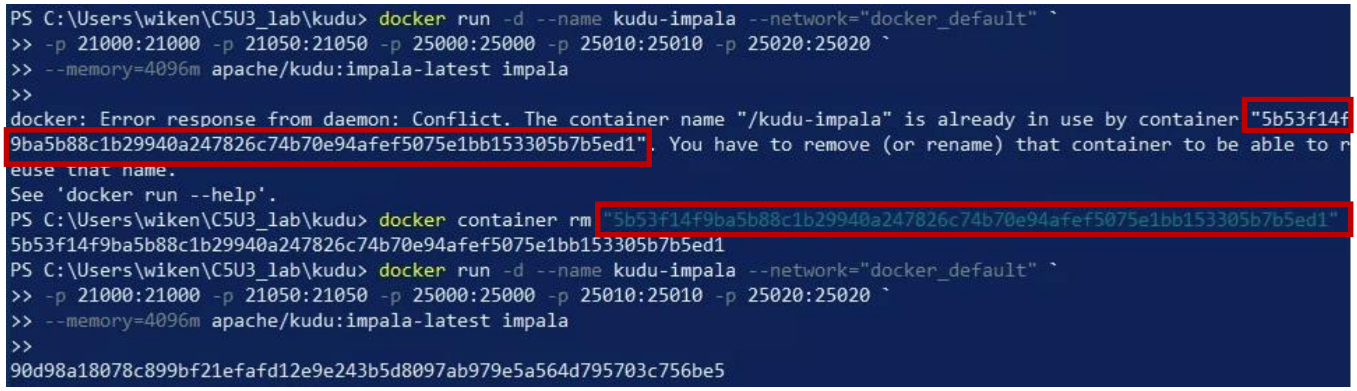
1.7.3. Start the Impala cluster with the following command:

```
docker run -d --name kudu-impala --network="docker_default" `
-p 21000:21000 -p 21050:21050 -p 25000:25000 -p 25010:25010 -p 25020:25020 `
--memory=4096m apache/kudu:impala-latest impala
```

CAUTION: If you have previously attempted this lab, there may already exist a container with the name "/kudu-impala." In this case, it is best to remove the container and run a fresh new kudu-impala container Only follow the instruction to remove the existing container if a container already exists. If not, skip to step 1.7.5

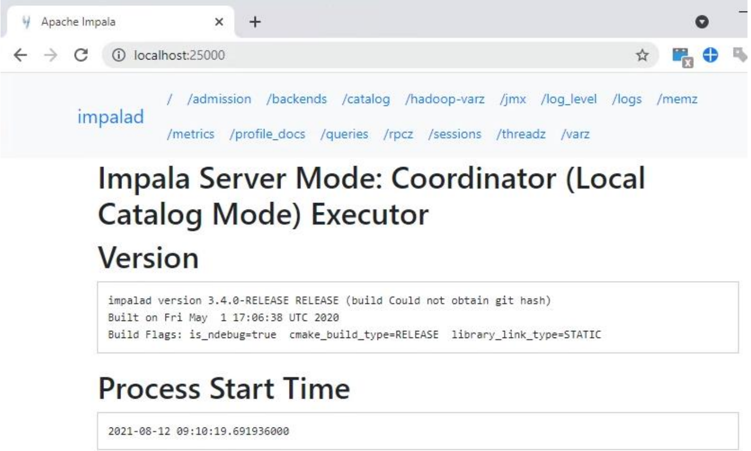
1.7.4. **CAUTION**: Follow only if container already exists. Use docker container rm to remove the existing container. Use the container id displayed in the error message when deleting

```
docker container rm <"container id">
```



```
PS C:\Users\wiken\C5U3_lab\kudu> docker run -d --name kudu-impala --network="docker_default" `
>> -p 21000:21000 -p 21050:21050 -p 25000:25000 -p 25010:25010 -p 25020:25020 `
>> --memory=4096m apache/kudu:impala-latest impala
>>
docker: Error response from daemon: Conflict. The container name "/kudu-impala" is already in use by container "5b53f14f
9ba5b88c1b29940a247826c74b70e94afef5075e1bb153305b7b5ed1". You have to remove (or rename) that container to be able to r
euse that name.
See 'docker run --help'.
PS C:\Users\wiken\C5U3_lab\kudu> docker container rm "5b53f14f9ba5b88c1b29940a247826c74b70e94afef5075e1bb153305b7b5ed1"
5b53f14f9ba5b88c1b29940a247826c74b70e94afef5075e1bb153305b7b5ed1
PS C:\Users\wiken\C5U3_lab\kudu> docker run -d --name kudu-impala --network="docker_default" `
>> -p 21000:21000 -p 21050:21050 -p 25000:25000 -p 25010:25010 -p 25020:25020 `
>> --memory=4096m apache/kudu:impala-latest impala
>>
90d98a18078c899bf21efafd12e9e243b5d8097ab979e5a564d795703c756be5
```

1.7.5. Check to make sure Impala services are up. Use your browser and navigate to localhost:25000



Apache Impala
localhost:25000
impalad
/ /admission /backends /catalog /hadoop-varz /jmx /log_level /logs /memz
/metrics /profile_docs /queries /rpcz /sessions /threadz /varz

Impala Server Mode: Coordinator (Local Catalog Mode) Executor

Version

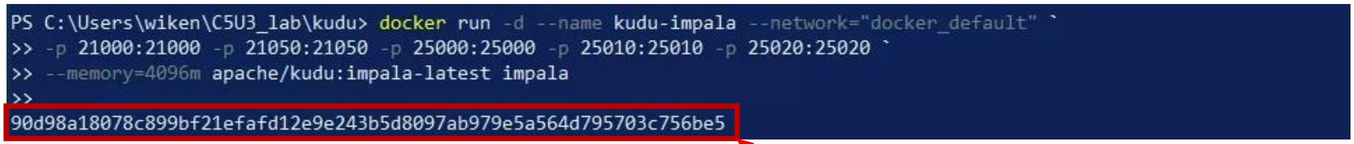
```
impalad version 3.4.0-RELEASE RELEASE (build Could not obtain git hash)
Built on Fri May  1 17:06:38 UTC 2020
Build Flags: is_ndebug=true  cmake_build_type=RELEASE  library_link_type=STATIC
```

Process Start Time

```
2021-08-12 09:10:19.691936000
```

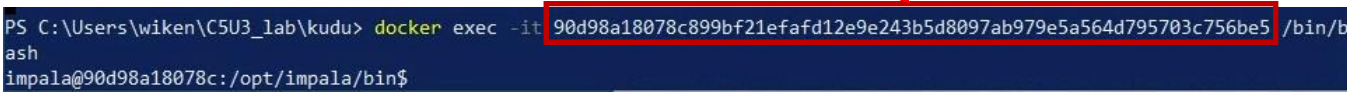
1.8. Start the Impala Shell

1.8.1. Open a command line to the Impala docker container with the following command. The container id is displayed when you created the container.



```
PS C:\Users\wiken\C5U3_lab\kudu> docker run -d --name kudu-impala --network="docker_default" `
>> -p 21000:21000 -p 21050:21050 -p 25000:25000 -p 25010:25010 -p 25020:25020 `
>> --memory=4096m apache/kudu:impala-latest impala
>>
90d98a18078c899bf21efafd12e9e243b5d8097ab979e5a564d795703c756be5
```

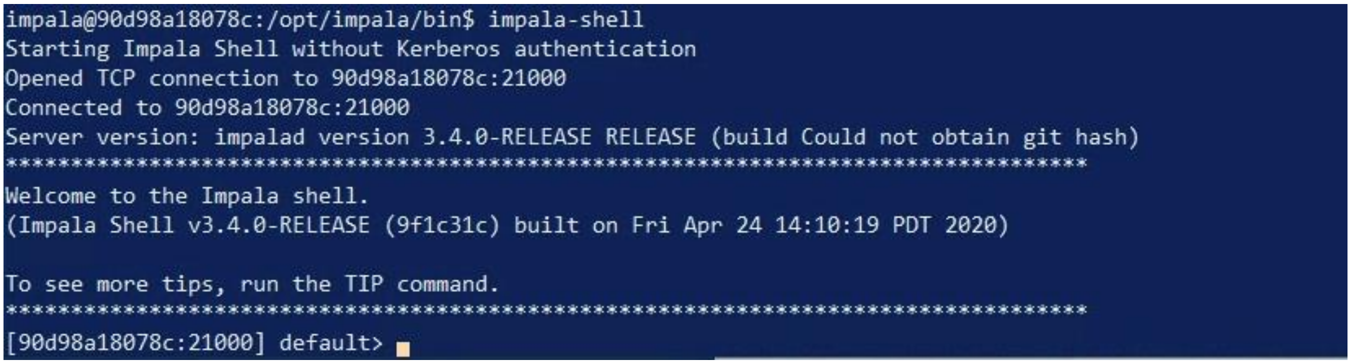
```
docker exec -it <container id> /bin/bash
```



```
PS C:\Users\wiken\C5U3_lab\kudu> docker exec -it 90d98a18078c899bf21efafd12e9e243b5d8097ab979e5a564d795703c756be5 /bin/b
ash
impala@90d98a18078c:/opt/impala/bin$
```

1.8.2. Once inside the docker container, start the Impala shell.

```
impala-shell
```



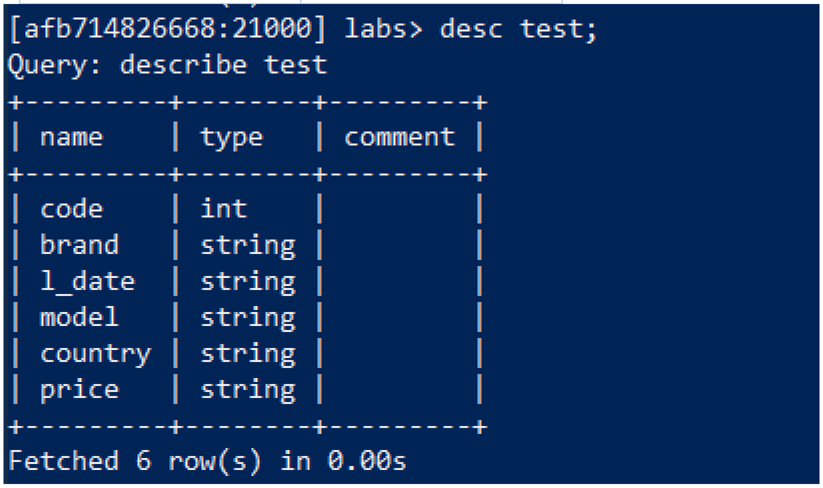
```
impala@90d98a18078c:/opt/impala/bin$ impala-shell
Starting Impala Shell without Kerberos authentication
Opened TCP connection to 90d98a18078c:21000
Connected to 90d98a18078c:21000
Server version: impalad version 3.4.0-RELEASE RELEASE (build Could not obtain git hash)
***********************************************************************************
Welcome to the Impala shell.
(Impala Shell v3.4.0-RELEASE (9f1c31c) built on Fri Apr 24 14:10:19 PDT 2020)

To see more tips, run the TIP command.
***********************************************************************************
[90d98a18078c:21000] default>
```

2. Create the labs database and show the database list.

```
0: jdbc:hive2://> create database labs;
0: jdbc:hive2://> show databases;
```

2.1. Create a table named test. Use the following schema:

Column Name	Column Type
Code	Int
brand	string
l_date	string
model	string
country	string
price	int



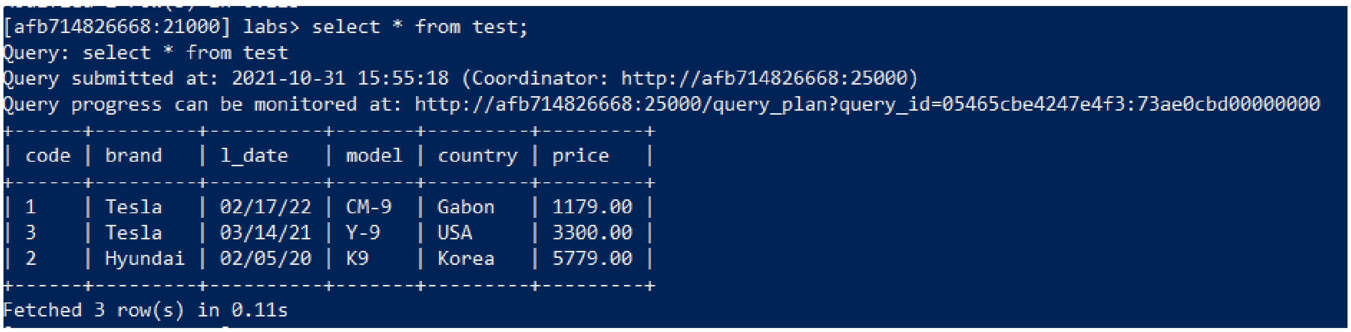
```
[afb714826668:21000] labs> desc test;
Query: describe test
+---------+--------+---------+
| name    | type   | comment |
+---------+--------+---------+
| code    | int    |         |
| brand   | string |         |
| l_date  | string |         |
| model   | string |         |
| country | string |         |
| price   | string |         |
+---------+--------+---------+
Fetched 6 row(s) in 0.00s
```

```
>Insert the following values and verify the result by executing the select statement.
```

2.1.1. INSERT INTO test VALUES (1, 'Tesla', '02/17/22', ‘CM-9', 'Gabon', '1179.00');

2.1.2. INSERT INTO test VALUES (2, 'Hyundai', '02/05/20', ‘K9', 'Korea', '5779.00');

2.1.3. INSERT INTO test VALUES (3, 'Tesla', '03/14/21', ‘Y-9', 'USA', '3300.00');



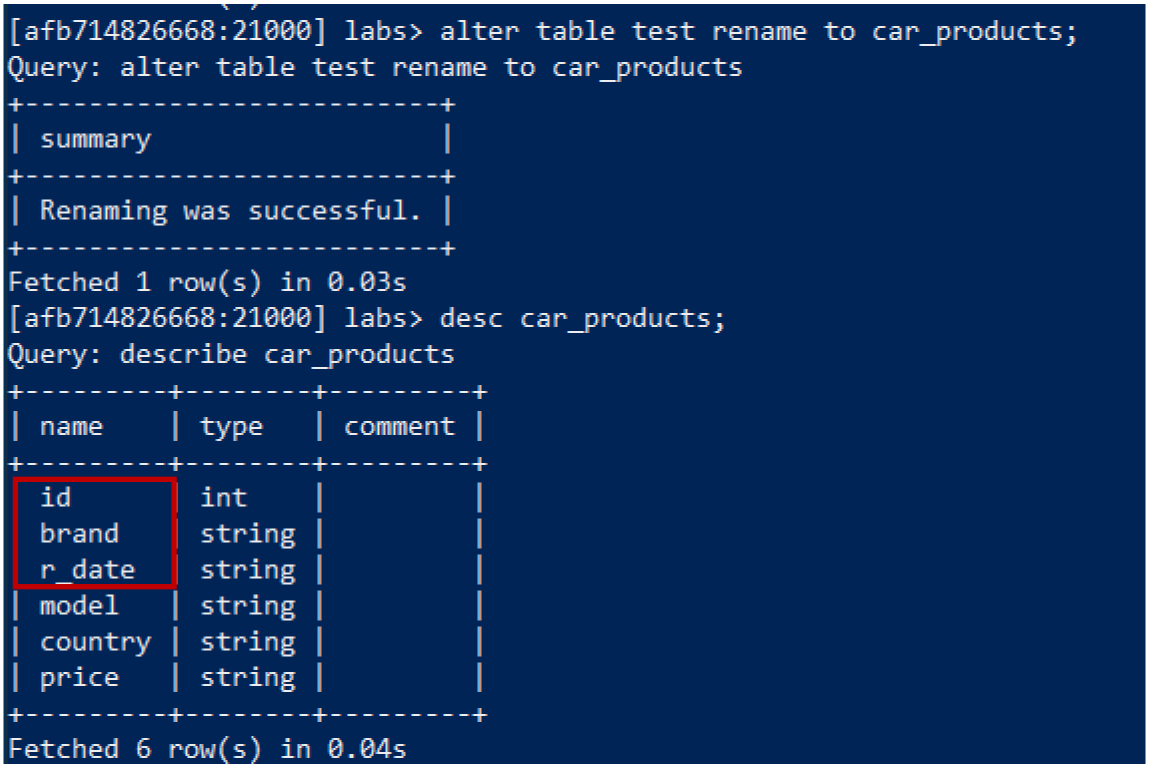
```
[afb714826668:21000] labs> select * from test;
Query: select * from test
Query submitted at: 2021-10-31 15:55:18 (Coordinator: http://afb714826668:25000)
Query progress can be monitored at: http://afb714826668:25000/query_plan?query_id=05465cbe4247e4f3:73ae0cbd00000000
+------+---------+----------+-------+---------+---------+
| code | brand   | l_date   | model | country | price   |
+------+---------+----------+-------+---------+---------+
| 1    | Tesla   | 02/17/22 | CM-9  | Gabon   | 1179.00 |
| 3    | Tesla   | 03/14/21 | Y-9   | USA     | 3300.00 |
| 2    | Hyundai | 02/05/20 | K9    | Korea   | 5779.00 |
+------+---------+----------+-------+---------+---------+
Fetched 3 row(s) in 0.11s
```

2.2. Change the field l_date to the r_date column name, and change the code to id.

```
> alter table test change code id int;
> alter table test change l_date r_date string;
```

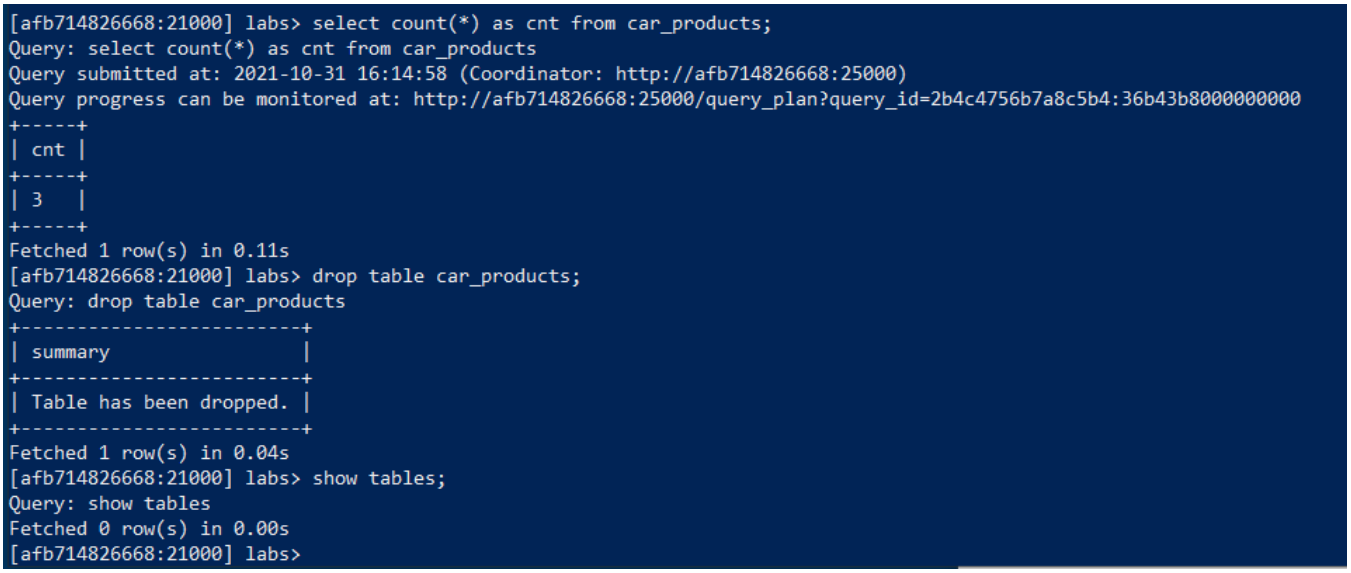
2.3. Rename the test car_products;

```
> alter table test rename to car_products;
```



```
[afb714826668:21000] labs> alter table test rename to car_products;
Query: alter table test rename to car_products
+--------------------------+
| summary                  |
+--------------------------+
| Renaming was successful. |
+--------------------------+
Fetched 1 row(s) in 0.03s
[afb714826668:21000] labs> desc car_products;
Query: describe car_products
+---------+--------+---------+
| name    | type   | comment |
+---------+--------+---------+
  id        int    |         |
  brand     string |         |
  r_date    string |         |
| model   | string |         |
| country | string |         |
| price   | string |         |
+---------+--------+---------+
Fetched 6 row(s) in 0.04s
```

2.4. Verify the number of records in car_products. And drop the car_products;



```
[afb714826668:21000] labs> select count(*) as cnt from car_products;
Query: select count(*) as cnt from car_products
Query submitted at: 2021-10-31 16:14:58 (Coordinator: http://afb714826668:25000)
Query progress can be monitored at: http://afb714826668:25000/query_plan?query_id=2b4c4756b7a8c5b4:36b43b8000000000
+-----+
| cnt |
+-----+
| 3   |
+-----+
Fetched 1 row(s) in 0.11s
[afb714826668:21000] labs> drop table car_products;
Query: drop table car_products
+-------------------------+
| summary                 |
+-------------------------+
| Table has been dropped. |
+-------------------------+
Fetched 1 row(s) in 0.04s
[afb714826668:21000] labs> show tables;
Query: show tables
Fetched 0 row(s) in 0.00s
[afb714826668:21000] labs>
```

Lab 11: Create a batch view

Creating an actual Lambda architecture is a very difficult and time-consuming project. Many enterprises are trying to move away from this architecture whenever possible due to its high maintenance and development costs. However, there are many use cases where Lambda architecture is the only viable choice.

Creating an actual lambda architecture would be beyond the scope of this lab. An actual proof-of-concept (POC) project will typically involve many moving pieces and actual code development. In this lab, we will create a data pipeline that will have many of the working parts without the real complexity of an actual working POC.

1. Create an Apache Flume data pipeline

Create a Flume configuration file with the following single source, two channels and two sinks

1.1. A single netcat source

1.1.1. Refer to https://flume.apache.org/releases/content/1.9.0/FlumeUserGuide.html#netcat-tcp-source

1.1.2. The hostname will be localhost

1.1.3. It will read from port 44444

1.2. First channel

1.2.1. A file channel

1.2.2. Refer to https://flume.apache.org/releases/content/1.9.0/FlumeUserGuide.html#file-channel

1.2.3. Set the checkpoint directory to /tmp/flume/checkpoint

1.2.4. Set the data directory to /tmp/flume/data

1.3. Second channel

1.3.1. A memory channel

1.3.2. Refer to https://flume.apache.org/releases/content/1.9.0/FlumeUserGuide.html#memory-channel

1.3.3. Maximum events store in the channel should be 10000

1.3.4. Maximum number of events per transaction

1.4. First Sink

1.4.1. A HDFS sink

1.4.2. Refer to https://flume.apache.org/releases/content/1.9.0/FlumeUserGuide.html#hdfs-sink

1.4.3. Save the data in /user/student/labc5u4

1.4.4. We will never roll based on time interval

1.4.5. We will roll when the file size reaches 16384

1.4.6. We will never roll based on number of events

1.4.7. The file type will be a DataStream

1.5. Second Sink

1.5.1. A logger sink

1.5.2. Refer to https://flume.apache.org/releases/content/1.9.0/FlumeUserGuide.html#logger-sink

1.6. Connecting the data pipelines

1.6.1. The single source will connect to both channels

1.6.2. The file channel will connect with the HDFS sink

1.6.3. The memory channel will connect with the logger sink

1.7. Run the flume agent

1.7.1. Assume that the name of the agent is a1

```
flume-ng agent --conf $FLUME_HOME/conf \
-conf-file net2hdfsandlog.conf  \
--name a1 \
-Dflume.root.logger=INFO,console
```

1.8. Start the streaming source simulator

1.8.1. Navigate the /home/student/Scripts

1.8.2. Execute the stream2.py Python script

```
cd /home/student/Scripts
python stream2.py ~/Data/anonymous-msweb.data
```

2. Create an external Hive table linked to HDFS

2.1. Start the beeline console

2.2. If a labs database does not already exits, create one

2.3. Create an external table and name it weblog

2.3.1. Add string column and name it line_type

2.3.2. Add string column and name it id1

2.3.3. Add string column and name it id2

2.3.4. Add string column and name it title

2.3.5. Add string column and name it url

2.3.6. Set the delimiter to a comma

2.3.7. Store as a textfile

2.3.8. Set the location to where Flume has saved to HDFS in above step 1.4.3

2.4. Confirm that the new Hive table is reading from the HDFS directory. Run the following query:

```
SELECT * FROM weblog LIMIT 10;
```

Lab 12: Create a speed view

We will simulate ingesting real-time streaming data .

1. Setup Apache Kafka

In order to reduce resource demand to our virtual machine, we shall stop hbase and run only Apache Kafka

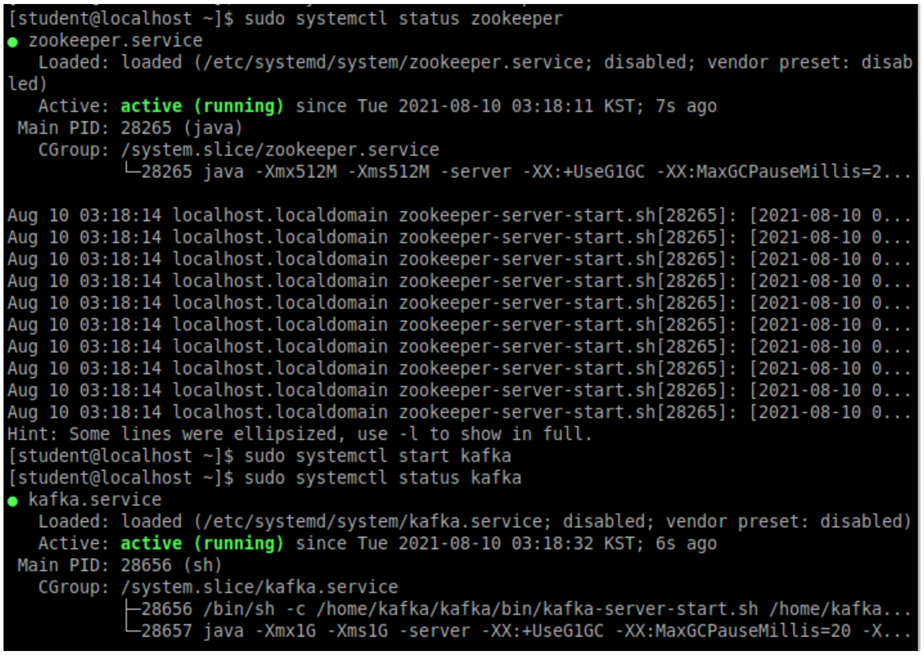
1.1. Stop HBase services

```
sudo stop-hbase.sh
```

1.2. Restart Kafka and Zookeeper

```
sudo systemctl stop kafka
sudo systemctl stop zookeeper
sudo systemctl start zookeeper
sudo systemctl status zookeeper
sudo systemctl start kafka
sudo systemctl status kafka
```

1.3. Make sure that both zookeeper and kaka is running. If not repeat above step.



```
[student@localhost ~]$ sudo systemctl status zookeeper
● zookeeper.service
   Loaded: loaded (/etc/systemd/system/zookeeper.service; disabled; vendor preset: disab
led)
   Active: active (running) since Tue 2021-08-10 03:18:11 KST; 7s ago
 Main PID: 28265 (java)
   CGroup: /system.slice/zookeeper.service
           └─28265 java -Xmx512M -Xms512M -server -XX:+UseG1GC -XX:MaxGCPauseMillis=2...

Aug 10 03:18:14 localhost.localdomain zookeeper-server-start.sh[28265]: [2021-08-10 0...
Aug 10 03:18:14 localhost.localdomain zookeeper-server-start.sh[28265]: [2021-08-10 0...
Aug 10 03:18:14 localhost.localdomain zookeeper-server-start.sh[28265]: [2021-08-10 0...
Aug 10 03:18:14 localhost.localdomain zookeeper-server-start.sh[28265]: [2021-08-10 0...
Aug 10 03:18:14 localhost.localdomain zookeeper-server-start.sh[28265]: [2021-08-10 0...
Aug 10 03:18:14 localhost.localdomain zookeeper-server-start.sh[28265]: [2021-08-10 0...
Aug 10 03:18:14 localhost.localdomain zookeeper-server-start.sh[28265]: [2021-08-10 0...
Aug 10 03:18:14 localhost.localdomain zookeeper-server-start.sh[28265]: [2021-08-10 0...
Aug 10 03:18:14 localhost.localdomain zookeeper-server-start.sh[28265]: [2021-08-10 0...
Aug 10 03:18:14 localhost.localdomain zookeeper-server-start.sh[28265]: [2021-08-10 0...
Hint: Some lines were ellipsized, use -l to show in full.
[student@localhost ~]$ sudo systemctl start kafka
[student@localhost ~]$ sudo systemctl status kafka
● kafka.service
   Loaded: loaded (/etc/systemd/system/kafka.service; disabled; vendor preset: disabled)
   Active: active (running) since Tue 2021-08-10 03:18:32 KST; 6s ago
 Main PID: 28656 (sh)
   CGroup: /system.slice/kafka.service
           ├─28656 /bin/sh -c /home/kafka/kafka/bin/kafka-server-start.sh /home/kafka...
           └─28657 java -Xmx1G -Xms1G -server -XX:+UseG1GC -XX:MaxGCPauseMillis=20 -X...
```

2. Create an Apache NiFi dataflow

   2.1. Open Apache NiFI from Firefox browser

   2.2. Create a new processor group

      2.2.1. Name the processor group, Lab C5U4

      2.2.2. Double click on processor group labC5U4 to enter its canvas

   2.3. Add ListenTCP processor to canvas

      2.3.1. Set Port to 44444 in Properties tab

      2.3.2. Change Max Size of Socket Buffer to 200000 B

      2.3.3. Set Client Auth to NONE

   2.4. Add PublishKafka_2_6 processor to canvas

      2.4.1. Automatically terminate failure relationship

      2.4.2. Set topic name to weblogs

      2.4.3. Keep Delivery Guarantee to Best Effort

      2.4.4. Change Use Transactions to false

   2.5. Add a funnel

   2.6. Connect the data sources

2.6.1. Connect ListenTCP to PublishKafka_2_6 on success relationship

2.6.2. Connect PublishKafka_2_6 to a funnel

2.7. Add ConsumeKafka_2_6 to canvas

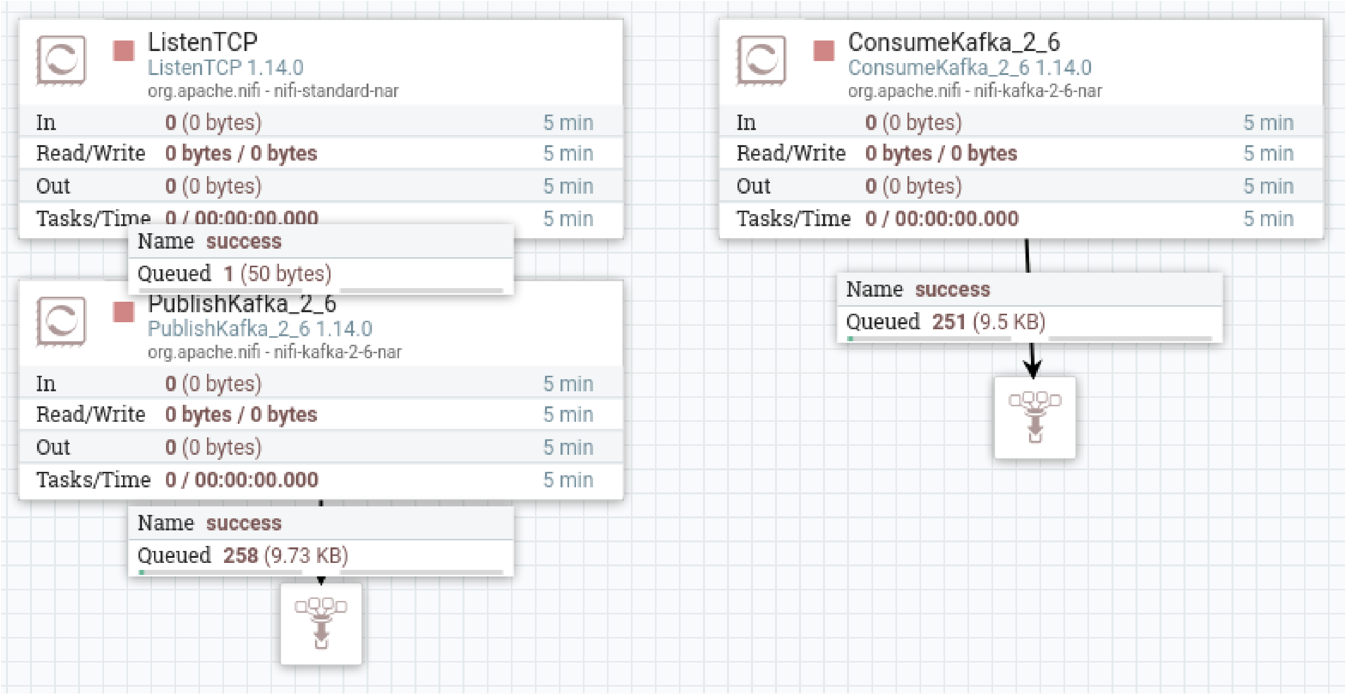
2.7.1. Change topic name to weblogs

2.7.2. Set honor transactions to false

2.7.3. Set Group ID to 1

2.7.4. Leave offset reset to latest

2.7.5. Add another funnel and connect the output of ConsumeKafka_2_6 to the funnel on the success relationship



ListenTCP
ListenTCP 1.14.0
org.apache.nifi - nifi-standard-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Name success
Queued 1 (50 bytes)

PublishKafka_2_6
PublishKafka_2_6 1.14.0
org.apache.nifi - nifi-kafka-2-6-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Name success
Queued 258 (9.73 KB)

ConsumeKafka_2_6
ConsumeKafka_2_6 1.14.0
org.apache.nifi - nifi-kafka-2-6-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Name success
Queued 251 (9.5 KB)

3. Test the speed view dataflow

3.1. Select all the components and run them

3.2. Start the streaming source simulator

3.2.1. Navigate the /home/student/Scripts

3.2.2. Execute the stream2.py Python script

```
cd /home/student/Scripts
```

```
python stream2.py ~/Data/anonymous-msweb.data
```

3.3. Wait until you see flowfiles flowing through the data flow. Stop all the processor and inspect the queues.

3.3.1. Inspect the queue between ListenTCP and PublishKafka_2_6. Make sure you view the content of the flowfile

3.3.2. Inspect the queue between PublishKafka_2_6 and the funnel

3.3.3. Inspect the queue between ConsumeKafka_2_6 and the funnel

3.4. Start all the components again

3.5. From another terminal, use kafka-console-consumer to verify that the streaming data has been properly ingested

3.5.1. Run the following command from another terminal

```
kafka-console-consumer \
 --bootstrap-server localhost:9092 \
--topic weblogs \
 --from-beginning
```

3.5.2. After making sure, everything is flowing properly, stop all components in NiFi

3.5.3. Terminate the console consumer

END OF LAB