MIDS W205 : Lab 5

Lab	5	*Lab Title*	Working with Relational Databases
Related Modules(s)	5	*Goal*	Get you introduced to a RDBMS (PostgreSQL)
Last Updated	1/7/17	*Expected Duration*	60 minutes

Introduction, Resources

While our initial investigations have dealt with Hive and SparkSQL, often as a Data Scientist, you will encounter relational databases like PostgreSQL. In this lab, we will learn about the following:

1. How to create a database in PostgreSQL
2. How to load data into PostgreSQL
3. How to run queries on PostgreSQL
4. How queries are transformed into plans for DAGs in PostgreSQL

Resource	What
http://www.postgresql.org/docs/9.5/static/index.html	PostgreSQL Documentation
http://www.postgresql.org/docs/9.5/static/sql.html	The SQL Language

Step 1. Setup the environment

We need to setup an EC2 instance and make sure that PostgreSQL is up and running. Do the following:

1. Launch an instance of UCB W205 Spring 2016

2. Attach your EBS volume from Lab 2. Note that PostgreSQL should be installed after you finish

step 3.4 of Lab 2

3. Check whether PostgreSQL is up and running:

```
ps auxw | grep postgres
```

4. If not, change your current path to /data directory

```
cd /data
```

and start Postgres

```
start_postgres.sh
```

Step 2. Getting the Data

1. We need some data in order to create a database, schema and, ultimately, query. The data we'll consider is a toy dataset called "DVD rental".

2. Navigate to the /data directory on your AWS instance and download the Pagila data as follows:

```
wget -O pangila.zip "https://ftp.postgresql.org/pub/projects/pgFoundry/dbsamples/pagila/pagila/pagila-0.10.1.zip"
```

3. Unzip the data:

```
unzip pagila.zip
```

4. Connecting to the PostgreSQL instance: Log into postgres as the postgres user:

```
psql -U postgres
```

5. Create a database:

```
create database dvdrental;
```

6. Connect to the database:

```
\c dvdrental
```

7. Import the data: Load the data using the `\i` command. Starting a line with `\i` runs `.sql` scripts in Postgres.

   ```
   \i pagila-0.10.1/pagila-schema.sql
   \i pagila-0.10.1/pagila-insert-data.sql
   \i pagila-0.10.1/pagila-data.sql
   ```

   Note: *This data set may contain some duplicate keys. For purposes of this lab, you can safely ignore any primary key violations during import.*

8. Check to see if it worked:

   At this point the data should be loaded. Examine the database schema by typing:

   ```
   \dt
   ```

   Examine the schema of a table using the `\d` command

   ```
   \d <table name>
   ```

> ***Question 1: What is the output of \dt?***
>
> ***Question 2: What is the schema for the customer table?***

Step 3. Running Queries and Understanding EXPLAIN plans

We want to understand not only what queries we can issue against data, but also how that query maps to an execution plan. For each of the following sections, run the queries provided, and generate their explain plans using:

```
  EXPLAIN <sql query here>
```

Run the following simple queries, then generate their explain plans.

1. **Projection:**

```
SELECT customer_id, first_name, last_name FROM customer;
```

2. **Projection and Selection #1:**

```
SELECT customer_id,
     amount,
     payment_date
FROM payment
WHERE amount <= 1
OR amount >= 8;
```

3. **Projection and Selection #2:**

```
SELECT customer_id,
     payment_id,
     amount
FROM payment
WHERE amount BETWEEN 5 AND 9;
```

> ***Question 3: What similarities do you see in the explain plains for these 3 queries?***

4. **Merging Data: UNIONs:** Run the following 2 statements:

   Union of 2 tables:

```
SELECT u.customer_id,
     sum(u.amount)
FROM
( SELECT *
 FROM payment_p2007_01
 UNION SELECT *
 FROM payment_p2007_02) u
WHERE u.payment_date <= '2007-02-01 00:00:00'::TIMESTAMP WITHOUT time ZON
E
GROUP BY u.customer_id ;
```

   Partition a Table:

```
SELECT customer_id,
sum(amount)
FROM payment
WHERE payment_date <= '2007-02-01 00:00:00'::TIMESTAMP WITHOUT time ZONE
GROUP BY customer_id ;
```

> ***Question 4: What is the difference between the plans for the Partitioned table and the union query? Why do you think this difference exists?***

5. **Merging Data: JOINs:**

```
SELECT customer.customer_id,
     first_name,
     last_name,
     email,
     amount,
     payment_date
FROM customer
INNER JOIN payment ON payment.customer_id = customer.customer_id;
```

> ***Question 5: What join algorithm is used for the inner join?***

6. Finally, disconnect from postgres: `\q`

Submissions

Submit your answers to the questions through ISVC as a text file, docx file, or PDF.