MIDS W205

|  |  |  |  |
| --- | --- | --- | --- |
| **Lab #** | 5 | **Lab Title** | Working with Relational Databases |
| **Related Module(s)** | 5 | **Goal** | Get you introduced to a RDBMS (PostgreSQL) |
| **Last Updated** | 2/14/16 | **Expected duration** | 40 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
   1. Attach your EBS volume from Lab 2. Note that PostgreSQL should be installed after you finish step 3.4 of Lab 2
   2. Check whether PostgreSQL is up and running:

ps auxwww | grep postgres

* 1. If not, change your current path to /data :
     1. cd /data
     2. Start Postgres: /data/start\_postgres.sh

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

wget -O pagila.zip <http://pgfoundry.org/frs/download.php/1719/pagila-0.10.1.zip>

1. Unzip the data

unzip pagila.zip

1. Connecting to the PostgreSQL instance, creating a database, and importing the data:
   1. Log into postgres as the postgres user:

psql –U postgres

* 1. Create the database:

create database dvdrental;

* 1. Connect to the database using \c

\c dvdrental

* 1. Load the data using the \i command. \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

At this point the data is loaded. Examine the database schema using the \dt command. Examine the schema of a table using the \d <table name> command

### Question 1: What is the output of \dt?

### Question 2: What is the schema for the customer table?

# Step 2. 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>

## Projection and Selection

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

**Projection:**

SELECT customer\_id, first\_name, last\_name FROM customer;

**Projection and Selection #1:**

SELECT customer\_id,

amount,

payment\_date

FROM payment

WHERE amount <= 1 OR amount >= 8;

**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?

## Merging Data: JOINs and UNIONs:

Run the following statements:

Union 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 zone

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?

Join 2 tables:

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?

Finally, disconnect from postgres, using \q

# Submissions

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