# Lab 7: Working with Relational Databases

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, you'll explore the basics of loading data into Postgres, creating running queries and understanding how those queries are transformed into plans for DAGs. Submit your answers through the LMS as a text file, docx file, or PDF.

## Getting the Data

Navigate to the /data directory on your AWS instance or the $HOME directory on your Vagrant install. Download the Pagila data as follows:

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

unzip pagila.zip
```

Next, we're going to log into postgresql and import the data.

Log into postgres as the postgres user:

```
psql –U postgres
```

Now create the database:

```
create database dvdrental;
```

Connect to the database using \c

```
\c dvdrental
```

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-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?

## 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
;
```

Partitioned 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