L1 E1 - Step 1 & 2

January 10, 2023

1 Exercise 1 - Sakila Star Schema & ETL

All the database tables in this demo are based on public database samples and transformations - `Sakila` is a sample database created by `MySql` Link - The postgresql version of it is called `Pagila` Link - The facts and dimension tables design is based on O'Reilly's public dimensional modelling tutorial schema Link

2 STEP0: Using ipython-sql

- Load ipython-sql: `%load_ext sql`
- To execute SQL queries you write one of the following atop of your cell:
  - `%sql`
    - For a one-liner SQL query
    - You can access a python var using `$`
  - `%%sql`
    - For a multi-line SQL query
    - You can **NOT** access a python var using `$`
- Running a connection string like: `postgresql://postgres:postgres@db:5432/pagila` connects to the database

3 STEP1 : Connect to the local database where Pagila is loaded

3.1 1.1 Create the pagila db and fill it with data

- Adding `"!"` at the beginning of a jupyter cell runs a command in a shell, i.e. we are not running python code but we are running the `createdb` and `psql` postgresql commmand-line utilities

```
In [1]: !PGPASSWORD=student createdb -h 127.0.0.1 -U student pagila
        !PGPASSWORD=student psql -q -h 127.0.0.1 -U student -d pagila -f Data/pagila-schema.sql
        !PGPASSWORD=student psql -q -h 127.0.0.1 -U student -d pagila -f Data/pagila-data.sql
```

```
 setval
--------
    200
(1 row)

 setval
--------
    605
(1 row)

 setval
--------
     16
(1 row)

 setval
--------
    600
(1 row)

 setval
--------
    109
(1 row)

 setval
--------
    599
(1 row)

 setval
--------
   1000
(1 row)

 setval
--------
   4581
(1 row)

 setval
--------
      6
(1 row)

 setval
--------
  32098
```

```
(1 row)

 setval
--------
  16049
(1 row)

 setval
--------
      2
(1 row)

 setval
--------
      2
(1 row)
```

3.2 1.2 Connect to the newly created db

```
In [2]: %load_ext sql
```

```
In [3]: DB_ENDPOINT = "127.0.0.1"
        DB = 'pagila'
        DB_USER = 'student'
        DB_PASSWORD = 'student'
        DB_PORT = '5432'

        # postgresql://username:password@host:port/database
        conn_string = "postgresql://{}:{}@{}:{}/{}" \
                                .format(DB_USER, DB_PASSWORD, DB_ENDPOINT, DB_PORT, DB)

        print(conn_string)
```

```
postgresql://student:student@127.0.0.1:5432/pagila
```

```
In [4]: %sql $conn_string
```

```
Out[4]: 'Connected: student@pagila'
```

4 STEP2 : Explore the 3NF Schema

4.1 2.1 How much? What data sizes are we looking at?

```
In [5]: nStores = %sql select count(*) from store;
        nFilms = %sql select count(*) from film;
        nCustomers = %sql select count(*) from customer;
```

```
        nRentals = %sql select count(*) from rental;
        nPayment = %sql select count(*) from payment;
        nStaff = %sql select count(*) from staff;
        nCity = %sql select count(*) from city;
        nCountry = %sql select count(*) from country;

        print("nFilms\t\t=", nFilms[0][0])
        print("nCustomers\t=", nCustomers[0][0])
        print("nRentals\t=", nRentals[0][0])
        print("nPayment\t=", nPayment[0][0])
        print("nStaff\t\t=", nStaff[0][0])
        print("nStores\t\t=", nStores[0][0])
        print("nCities\t\t=", nCity[0][0])
        print("nCountry\t\t=", nCountry[0][0])
```

```
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
nFilms                = 1000
nCustomers        = 599
nRentals        = 16044
nPayment        = 16049
nStaff                = 2
nStores                = 2
nCities                = 600
nCountry                = 109
```

4.2 2.2 When? What time period are we talking about?

```
In [6]: %%sql
        select min(payment_date) as start, max(payment_date) as end from payment;
```

```
 * postgresql://student:***@127.0.0.1:5432/pagila
1 rows affected.
```

```
Out[6]: [(datetime.datetime(2017, 1, 24, 21, 21, 56, 996577, tzinfo=psycopg2.tz.FixedOffsetTimez
```

4.3 2.3 Where? Where do events in this database occur?

TODO: Write a query that displays the number of addresses by district in the address table. Limit the table to the top 10 districts. Your results should match the table below.

```
In [12]: %%sql
         select district, COUNT(*) as n
         from address
         GROUP BY district
         ORDER BY COUNT(*) DESC
         LIMIT 10

 * postgresql://student:***@127.0.0.1:5432/pagila
10 rows affected.


Out[12]: [('Buenos Aires', 10),
          ('California', 9),
          ('Shandong', 9),
          ('West Bengali', 9),
          ('So Paulo', 8),
          ('Uttar Pradesh', 8),
          ('Maharashtra', 7),
          ('England', 7),
          ('Southern Tagalog', 6),
          ('Punjab', 5)]

<tbody><tr>
    <th>district</th>
    <th>n</th>
</tr>
<tr>
    <td>Buenos Aires</td>
    <td>10</td>
</tr>
<tr>
    <td>California</td>
    <td>9</td>
</tr>
<tr>
    <td>Shandong</td>
    <td>9</td>
</tr>
<tr>
    <td>West Bengali</td>
    <td>9</td>
</tr>
```

```
<tr>
    <td>So Paulo</td>
    <td>8</td>
</tr>
<tr>
    <td>Uttar Pradesh</td>
    <td>8</td>
</tr>
<tr>
    <td>Maharashtra</td>
    <td>7</td>
</tr>
<tr>
    <td>England</td>
    <td>7</td>
</tr>
<tr>
    <td>Southern Tagalog</td>
    <td>6</td>
</tr>
<tr>
    <td>Punjab</td>
    <td>5</td>
</tr>
```