Basics of SQL SELECT Statement

Here we will learn one of the most commonly used statements of SQL (Structured Query Language), the SELECT statement. The SELECT statement is used to select data from a database.

How does the syntax of a SELECT statement look?

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

What do the keywords / clauses of a SQL statement shown above do?

- **FROM**: Specifies from which table to get the data. The clause can include optional JOIN subclauses to specify the rules for joining tables.
- [Optional Clause] **WHERE** : Specifies which rows to retrieve.

Why is there a semicolon after the SQL statements?

- Some database systems require a semicolon at the end of each SQL statement for execution. It is a standard way to separate one SQL statement from another which allows more than one SQL statement to be executed in the same call to the server. So, it is good practice to use a semicolon at the end of each SQL statement.

Software Used in this Lab

In this lab, we will use Datasette , an open source multi-tool for exploring and publishing data.

Database Used in this Lab

The database used in this lab comes from the following dataset source: Film Locations in San Francisco under a PDDL: Public Domain Dedication and License.

Objectives

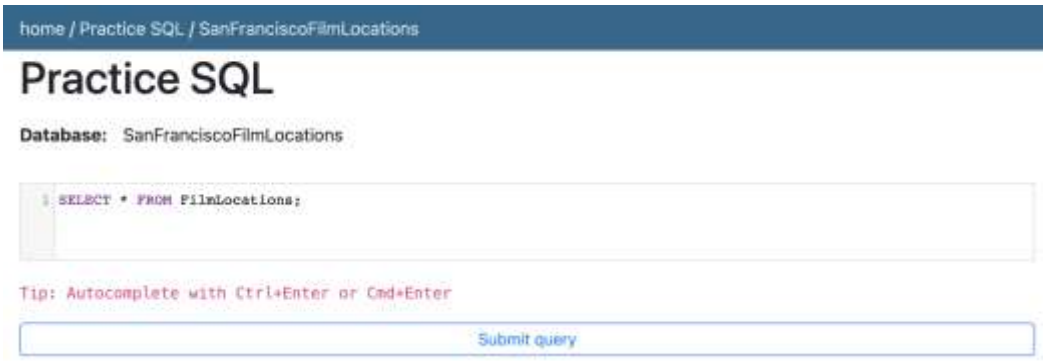
- Querying a database.
- Retrieving data records from one or more tables of a database as result set according to the criteria specified.

Task A: Exploring the Database

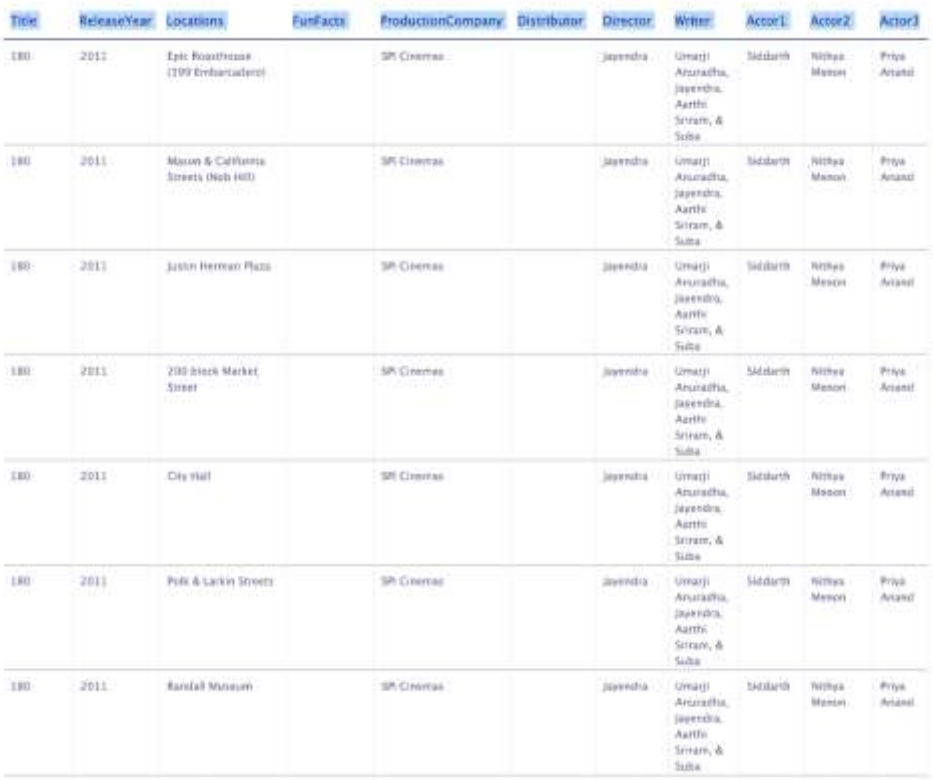
Let us first explore the **SanFranciscoFilmLocations** database using the **Datasette** tool:

1. If the first statement listed below is not already in the Datasette textbox on the right, then copy the code below by clicking on the little copy button on the bottom right of the codeblock below and then paste it into the textbox of the Datasette tool using either **Ctrl+V** or right-click in the text box and choose **Paste**.

```
SELECT * FROM FilmLocations;
```



2. Click **Submit Query**.
3. Now you can scroll down the table and explore all the columns and rows of the **FilmLocations** table to get an overall idea of the table contents.



4. These are the column attribute descriptions from the **FilmLocations** table:

```
FilmLocations(
    Title:              titles of the films,
    ReleaseYear:        time of public release of the films,
    Locations:          locations of San Francisco where the films were shot,
    FunFacts:           funny facts about the filming locations,
    ProductionCompany:  companies who produced the films,
    Distributor:        companies who distributed the films,
    Director:           people who directed the films,
    Writer:             people who wrote the films,
    Actor1:             person 1 who acted in the films,
    Actor2:             person 2 who acted in the films,
    Actor3:             person 3 who acted in the films
)
```

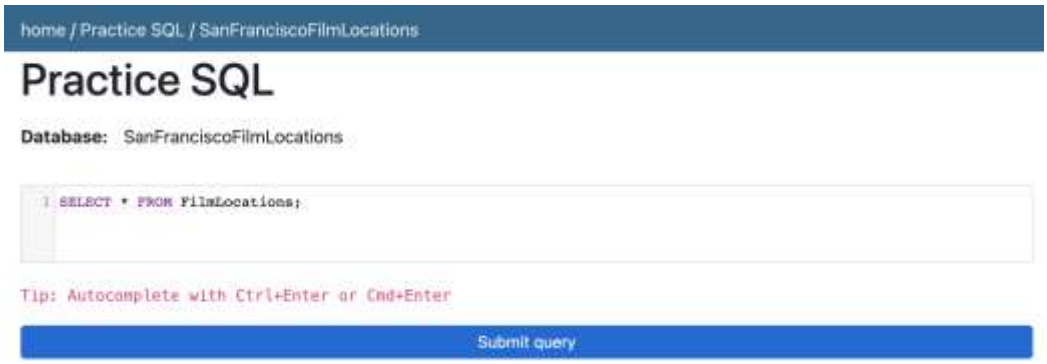
Task B: Example exercises on SELECT statement

Now let us go through some examples of SELECT queries:

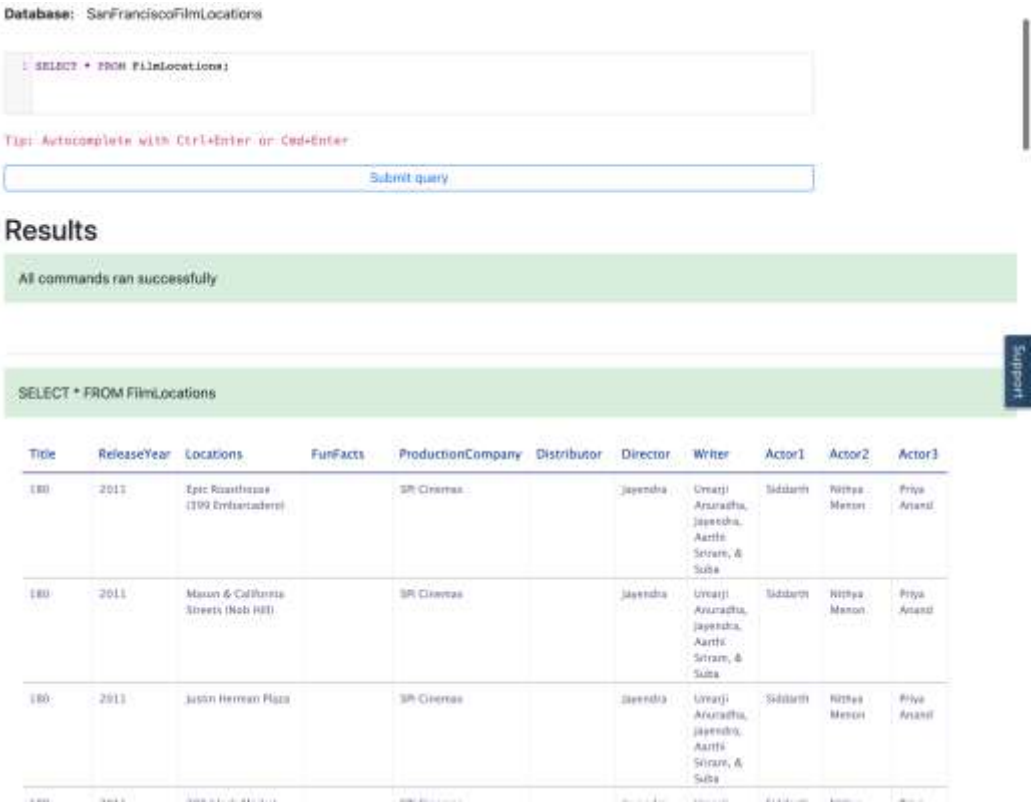
1. In this example, suppose we want to retrieve details of all the films from the "FilmLocations" table. The details of each film record should contain all the film columns.
    1. Problem:

       Retrieve all records with all columns from the "FilmLocations" table.

    2. Solution: `SELECT * FROM FilmLocations;`



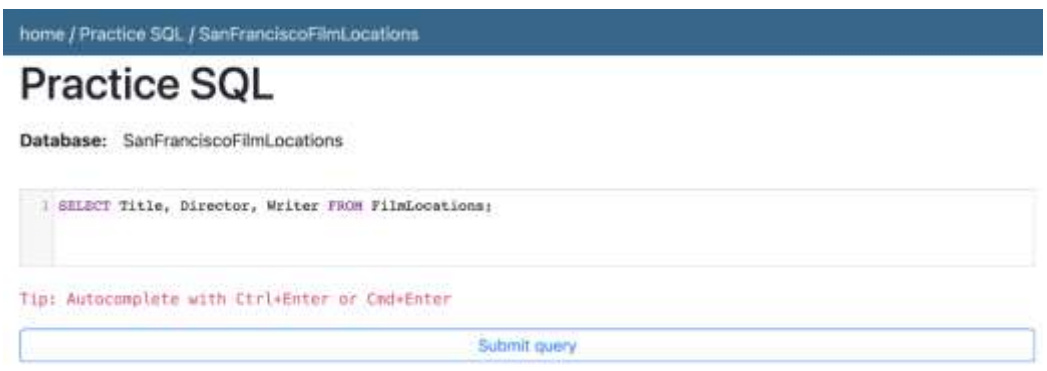
4. Output result set:



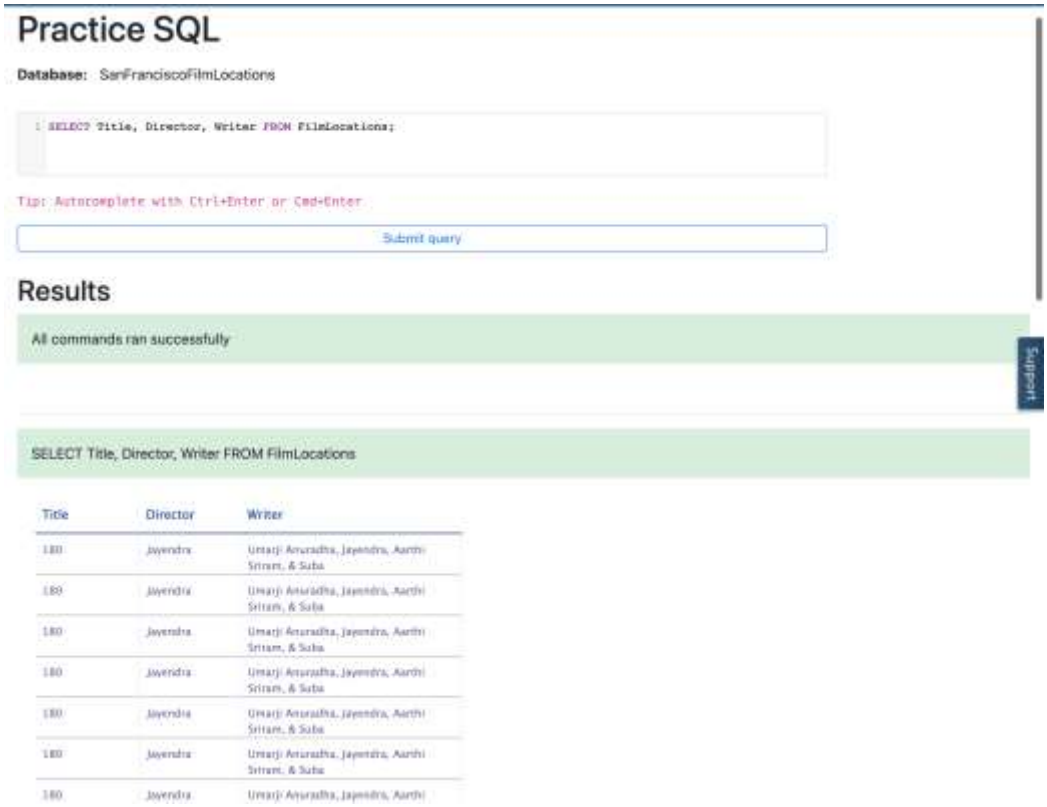
2. In this example, now we want to retrieve selective details of all the film records. Let us retrieve the names of all the films along with director names and writer names.
    1. Problem:

        Retrieve the names of all films with director names and writer names.

    2. Solution: `SELECT Title, Director, Writer FROM FilmLocations;`



4. Output result set:

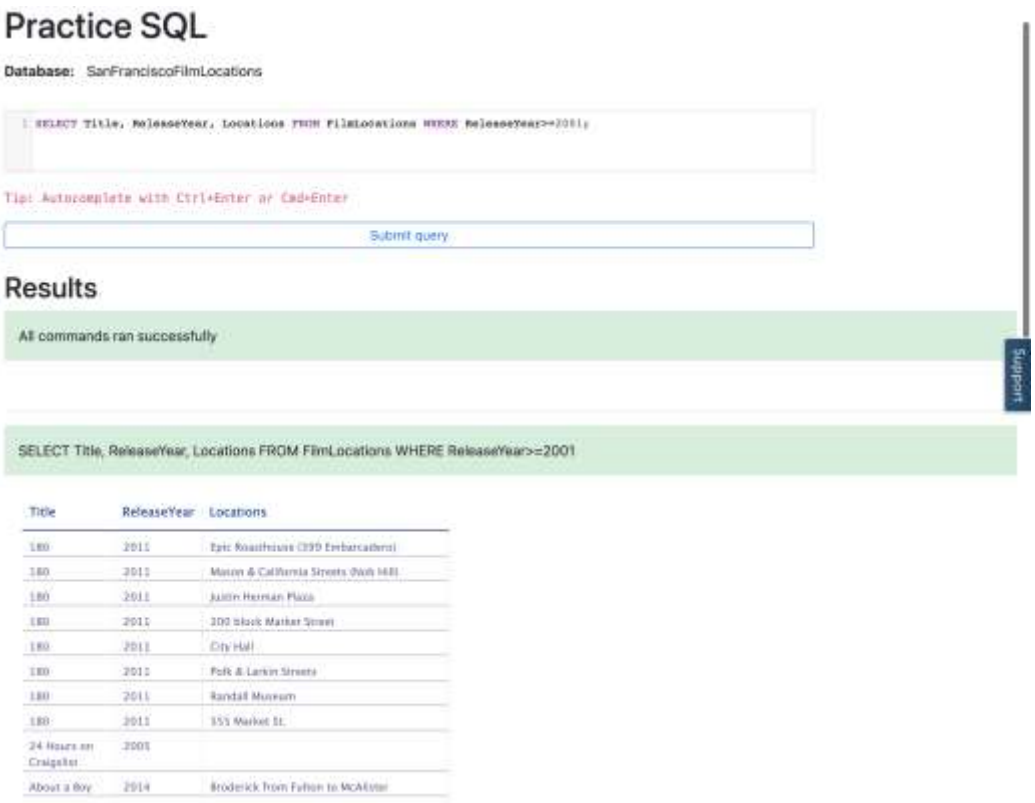


3. In this example, we want to retrieve film names along with filming locations and release years. But we also want to restrict the output resultset so that we can retrieve only the film records released in 2001 and onwards (release years after 2001 including 2001).
    1. Problem:

        Retrieve the names of all films released in the 21st century and onwards (release years after 2001 including 2001), along with filming locations and release years.

    2. Solution: `SELECT Title, ReleaseYear, Locations FROM FilmLocations WHERE ReleaseYear>=2001;`

4. Output result set:



Task C: Practice exercises on SELECT statement

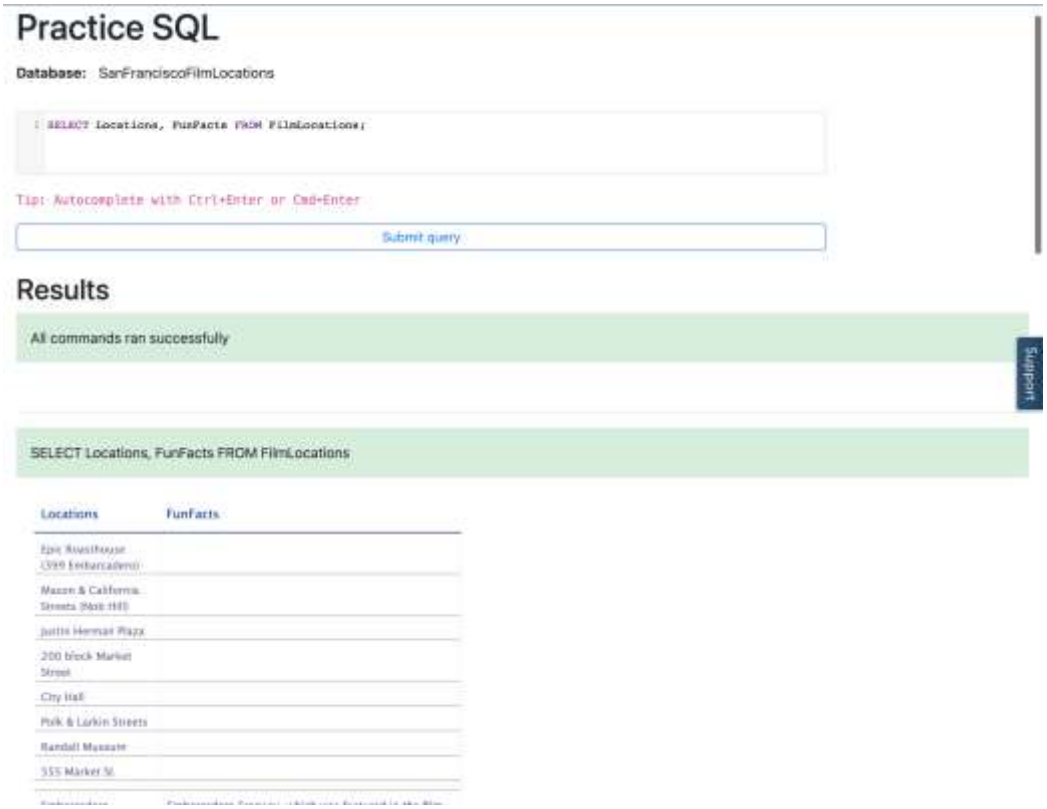
Finally, let us practice creating and running some SELECT queries.

1. Problem:

   Retrieve the fun facts and filming locations of all films.

   Solution: `SELECT Locations, FunFacts FROM FilmLocations;`

Output:

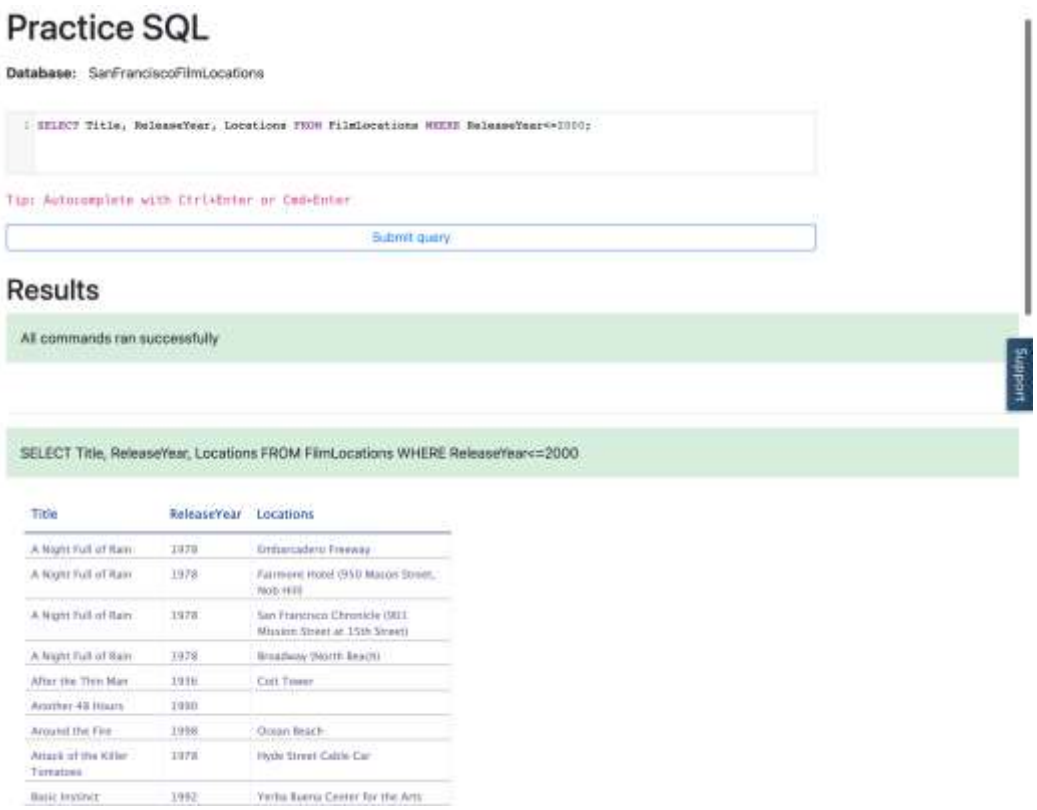


2. Problem:

Retrieve the names of all films released in the 20th century and before (release years before 2000 including 2000) that, along with filming locations and release years.

Solution: `SELECT Title, ReleaseYear, Locations FROM FilmLocations WHERE ReleaseYear<=2000;`

Output:

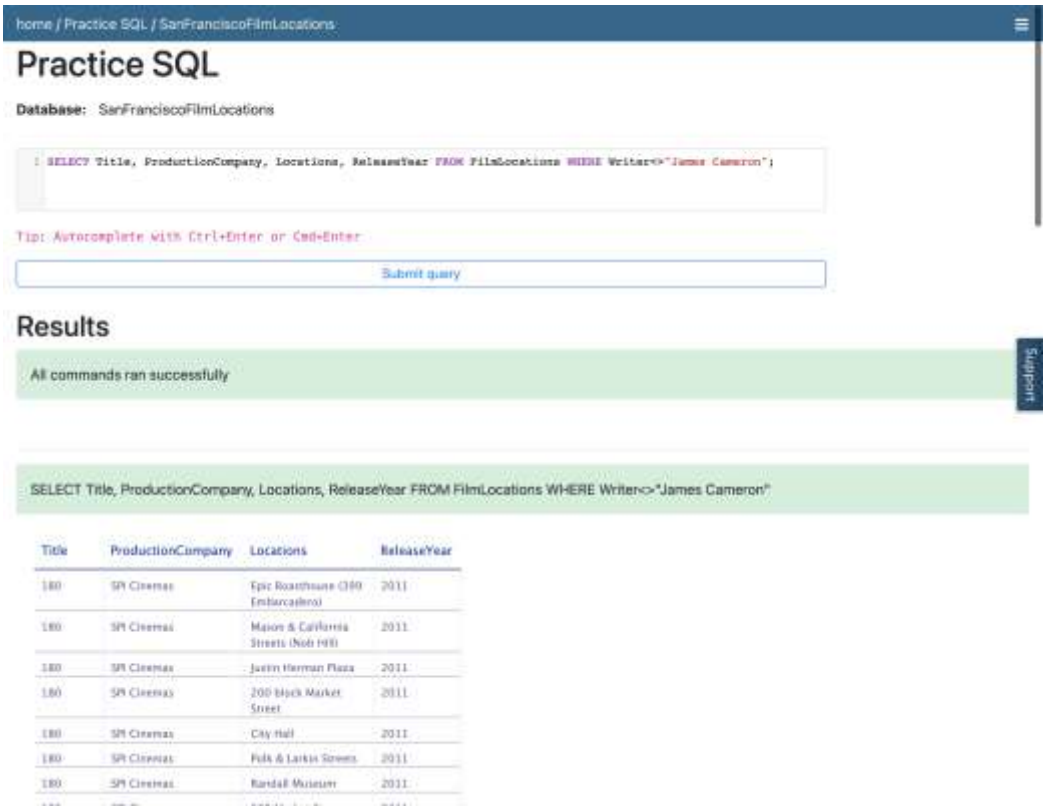


3. Problem:

Retrieve the names, production company names, filming locations, and release years of the films which are not written by James Cameron.

Solution: `SELECT Title, ProductionCompany, Locations, ReleaseYear FROM FilmLocations WHERE Writer<>"James Cameron";`

Output:



This is the end of the notebook. In this code notebook some of knowledge required for a data scientist and some of the skills used by data scientists on a daily basis were shown and applied. The code and images examples were provided by IBM, and the development of the code, solutions and outputs, as well as some notes and editions, were carried out by me, Saulo Villaseñor (https://www.linkedin.com/in/saulo-villase%C3%B1or-60669610a), so that this notebook is available to everybody and work as a reference for anyone who wishes to learn new skills.