Accessing Databases with SQL Magic

Estimated time needed: **15** minutes

Objectives

After completing this lab you will be able to:

- Perform simplified database access using SQL "magic"

To communicate with SQL Databases from within a JupyterLab notebook, we can use the SQL "magic" provided by the ipython-sql extension. "Magic" is JupyterLab's term for special commands that start with "%". Below, we'll use the *load_ext* magic to load the ipython-sql extension. In the lab environemnt provided in the course the ipython-sql extension is already installed and so is the ibm_db_sa driver.

Here you will be creating and connecting to a new SQlite database SQLiteMagic.

The syntax for connecting to magic sql using sqllite is

%sql sqlite://DatabaseName

where DatabaseName will be your **.db** file

In [10]:
```
import csv, sqlite3
con = sqlite3.connect("SQLiteMagic.db")
cur = con.cursor()
```

In [11]:
```
%load_ext sql
```
```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

In [12]:
```
%sql sqlite:///SQLiteMagic.db
```

Out[12]:
```
'Connected: @SQLiteMagic.db'
```

For convenience, we can use %%sql (two %'s instead of one) at the top of a cell to indicate we want the entire cell to be treated as SQL. Let's use this to create a table and fill it with some test data for experimenting.

In [14]:
```
%%sql

CREATE TABLE INTERNATIONAL_STUDENT_TEST_SCORES (
        country VARCHAR(50),
        first_name VARCHAR(50),
        last_name VARCHAR(50),
        test_score INT
);
INSERT INTO INTERNATIONAL_STUDENT_TEST_SCORES (country, first_name, last_name, test
VALUES
('United States', 'Marshall', 'Bernadot', 54),
('Ghana', 'Celinda', 'Malkin', 51),
('Ukraine', 'Guillermo', 'Furze', 53),
('Greece', 'Aharon', 'Tunnow', 48),
('Russia', 'Bail', 'Goodwin', 46),
('Poland', 'Cole', 'Winteringham', 49),
('Sweden', 'Emlyn', 'Erricker', 55),
('Russia', 'Cathee', 'Sivewright', 49),
('China', 'Barny', 'Ingerson', 57),
('Uganda', 'Sharla', 'Papaccio', 55),
('China', 'Stella', 'Youens', 51),
('Poland', 'Julio', 'Buesden', 48),
('United States', 'Tiffie', 'Cosely', 58),
('Poland', 'Auroora', 'Stiffell', 45),
('China', 'Clarita', 'Huet', 52),
('Poland', 'Shannon', 'Goulden', 45),
('Philippines', 'Emylee', 'Privost', 50),
('France', 'Madelina', 'Burk', 49),
('China', 'Saunderson', 'Root', 58),
('Indonesia', 'Bo', 'Waring', 55),
('China', 'Hollis', 'Domotor', 45),
('Russia', 'Robbie', 'Collip', 46),
('Philippines', 'Davon', 'Donisi', 46),
('China', 'Cristabel', 'Radeliffe', 48),
('China', 'Wallis', 'Bartleet', 58),
('Moldova', 'Arleen', 'Stailey', 38),
('Ireland', 'Mendel', 'Grumble', 58),
('China', 'Sallyann', 'Exley', 51),
('Mexico', 'Kain', 'Swaite', 46),
('Indonesia', 'Alonso', 'Bulteel', 45),
('Armenia', 'Anatol', 'Tankus', 51),
('Indonesia', 'Coralyn', 'Dawkins', 48),
('China', 'Deanne', 'Edwinson', 45),
('China', 'Georgiana', 'Epple', 51),
('Portugal', 'Bartlet', 'Breese', 56),
('Azerbaijan', 'Idalina', 'Lukash', 50),
('France', 'Livvie', 'Flory', 54),
('Malaysia', 'Nonie', 'Borit', 48),
('Indonesia', 'Clio', 'Mugg', 47),
('Brazil', 'Westley', 'Measor', 48),
('Philippines', 'Katrinka', 'Sibbert', 51),
('Poland', 'Valentia', 'Mounch', 50),
('Norway', 'Sheilah', 'Hedditch', 53),
('Papua New Guinea', 'Itch', 'Jubb', 50),
('Latvia', 'Stesha', 'Garnson', 53),
('Canada', 'Cristionna', 'Wadmore', 46),
('China', 'Lianna', 'Gatward', 43),
('Guatemala', 'Tanney', 'Vials', 48),
('France', 'Alma', 'Zavittieri', 44),
('China', 'Alvira', 'Tamas', 50),
('United States', 'Shanon', 'Peres', 45),
('Sweden', 'Maisey', 'Lynas', 53),
('Indonesia', 'Kip', 'Hothersall', 46),
('China', 'Cash', 'Landis', 48),
('Panama', 'Kennith', 'Digance', 45),
('China', 'Ulberto', 'Riggeard', 48),
('Switzerland', 'Judy', 'Gilligan', 49),
('Philippines', 'Tod', 'Trevaskus', 52),
('Brazil', 'Herold', 'Heggs', 44),
('Latvia', 'Verney', 'Note', 50),
('Poland', 'Temp', 'Ribey', 50),
('China', 'Conroy', 'Egdal', 48),
('Japan', 'Gabie', 'Alessandone', 47),
('Ukraine', 'Devlen', 'Chaperlin', 54),
('France', 'Babbette', 'Turner', 51),
('Czech Republic', 'Virgil', 'Scotney', 52),
('Tajikistan', 'Zorina', 'Bedow', 49),
('China', 'Aidan', 'Rudeyeard', 50),
('Ireland', 'Saunder', 'MacLice', 48),
('France', 'Waly', 'Brunstan', 53),
('China', 'Gisele', 'Enns', 52),
('Peru', 'Mina', 'Winchester', 48),
('Japan', 'Torie', 'MacShirrie', 50),
('Russia', 'Benjamen', 'Kenford', 51),
('China', 'Etan', 'Burn', 53),
('Russia', 'Merralee', 'Chaperlin', 38),
('Indonesia', 'Lanny', 'Malam', 49),
('Canada', 'Wilhelm', 'Deeprose', 54),
('Czech Republic', 'Lari', 'Hillhouse', 48),
('China', 'Ossie', 'Woodley', 52),
('Macedonia', 'April', 'Tyer', 50),
('Vietnam', 'Madelon', 'Dansey', 53),
('Ukraine', 'Korella', 'McNamee', 52),
('Jamaica', 'Linnea', 'Cannam', 43),
('China', 'Mart', 'Coling', 52),
('Indonesia', 'Marna', 'Causbey', 47),
('China', 'Berni', 'Daintier', 55),
('Poland', 'Cynthia', 'Hassell', 49),
('Canada', 'Carma', 'Schule', 49),
('Indonesia', 'Malia', 'Blight', 48),
('China', 'Paulo', 'Seivertsen', 47),
('Niger', 'Kaylee', 'Hearley', 54),
('Japan', 'Maure', 'Jandak', 46),
('Argentina', 'Foss', 'Feavers', 45),
('Venezuela', 'Ron', 'Leggitt', 60),
('Russia', 'Flint', 'Gokes', 40),
('China', 'Linet', 'Conelly', 52),
('Philippines', 'Nikolas', 'Birtwell', 57),
('Australia', 'Eduard', 'Leipelt', 53)
```

```
 * sqlite:///SQLiteMagic.db
(sqlite3.OperationalError) table INTERNATIONAL_STUDENT_TEST_SCORES already exists
[SQL: CREATE TABLE INTERNATIONAL_STUDENT_TEST_SCORES (
        country VARCHAR(50),
        first_name VARCHAR(50),
        last_name VARCHAR(50),
        test_score INT
);]
(Background on this error at: http://sqlalche.me/e/13/e3q8)
```

Using Python Variables in your SQL Statements

You can use python variables in your SQL statements by adding a ":" prefix to your python variable names.

For example, if I have a python variable `country` with a value of `"Canada"`, I can use this variable in a SQL query to find all the rows of students from Canada.

In [15]:
```
country = "Canada"
%sql select * from INTERNATIONAL_STUDENT_TEST_SCORES where country = :country
```

```
 * sqlite:///SQLiteMagic.db
Done.
```

Out[15]:

country	first_name	last_name	test_score
Canada	Cristionna	Wadmore	46
Canada	Wilhelm	Deeprose	54
Canada	Carma	Schule	49

Assigning the Results of Queries to Python Variables

You can use the normal python assignment syntax to assign the results of your queries to python variables.

For example, I have a SQL query to retrieve the distribution of test scores (i.e. how many students got each score). I can assign the result of this query to the variable `test_score_distribution` using the `=` operator.

In [16]:
```
test_score_distribution = %sql SELECT test_score as "Test_Score", count(*) as "Freq
test_score_distribution
```

```
 * sqlite:///SQLiteMagic.db
Done.
```

Out[16]:

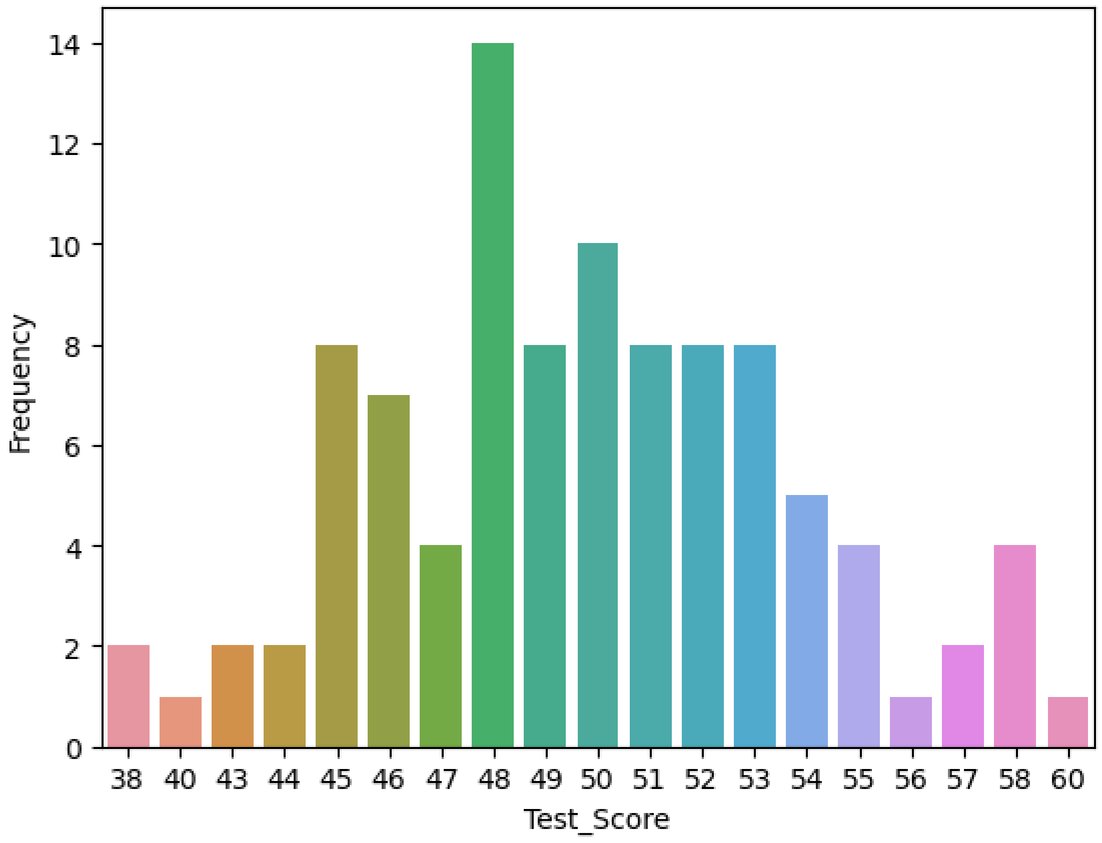
Test_Score	Frequency
38	2
40	1
43	2
44	2
45	8
46	7
47	4
48	14
49	8
50	10
51	8
52	8
53	8
54	5
55	4
56	1
57	2
58	4
60	1

Converting Query Results to DataFrames

You can easily convert a SQL query result to a pandas dataframe using the `DataFrame()` method. Dataframe objects are much more versatile than SQL query result objects. For example, we can easily graph our test score distribution after converting to a dataframe.

In [17]:
```
dataframe = test_score_distribution.DataFrame()
%matplotlib inline
# uncomment the following line if you get an module error saying seaborn not found
# !pip install seaborn==0.9.0
import seaborn

plot = seaborn.barplot(x='Test_Score',y='Frequency', data=dataframe)
```



Now you know how to work within JupyterLab notebooks using SQL "magic"!

```
In [18]: %%sql

         -- Feel free to experiment with the data set provided in this notebook for practice
         SELECT country, first_name, last_name, test_score FROM INTERNATIONAL_STUDENT_TEST_S
```

```
 * sqlite:///SQLiteMagic.db
Done.
```

Out[18]:

country	first_name	last_name	test_score
United States	Marshall	Bernadot	54
Ghana	Celinda	Malkin	51
Ukraine	Guillermo	Furze	53
Greece	Aharon	Tunnow	48
Russia	Bail	Goodwin	46
Poland	Cole	Winteringham	49
Sweden	Emlyn	Erricker	55
Russia	Cathee	Sivewright	49
China	Barny	Ingerson	57
Uganda	Sharla	Papaccio	55
China	Stella	Youens	51
Poland	Julio	Buesden	48
United States	Tiffie	Cosely	58
Poland	Auroora	Stiffell	45
China	Clarita	Huet	52
Poland	Shannon	Goulden	45
Philippines	Emylee	Privost	50
France	Madelina	Burk	49
China	Saunderson	Root	58
Indonesia	Bo	Waring	55
China	Hollis	Domotor	45
Russia	Robbie	Collip	46
Philippines	Davon	Donisi	46
China	Cristabel	Radeliffe	48
China	Wallis	Bartleet	58
Moldova	Arleen	Stailey	38
Ireland	Mendel	Grumble	58
China	Sallyann	Exley	51
Mexico	Kain	Swaite	46
Indonesia	Alonso	Bulteel	45

country	first_name	last_name	test_score
Armenia	Anatol	Tankus	51
Indonesia	Coralyn	Dawkins	48
China	Deanne	Edwinson	45
China	Georgiana	Epple	51
Portugal	Bartlet	Breese	56
Azerbaijan	Idalina	Lukash	50
France	Livvie	Flory	54
Malaysia	Nonie	Borit	48
Indonesia	Clio	Mugg	47
Brazil	Westley	Measor	48
Philippines	Katrinka	Sibbert	51
Poland	Valentia	Mounch	50
Norway	Sheilah	Hedditch	53
Papua New Guinea	Itch	Jubb	50
Latvia	Stesha	Garnson	53
Canada	Cristionna	Wadmore	46
China	Lianna	Gatward	43
Guatemala	Tanney	Vials	48
France	Alma	Zavittieri	44
China	Alvira	Tamas	50
United States	Shanon	Peres	45
Sweden	Maisey	Lynas	53
Indonesia	Kip	Hothersall	46
China	Cash	Landis	48
Panama	Kennith	Digance	45
China	Ulberto	Riggeard	48
Switzerland	Judy	Gilligan	49
Philippines	Tod	Trevaskus	52
Brazil	Herold	Heggs	44
Latvia	Verney	Note	50

country	first_name	last_name	test_score
Poland	Temp	Ribey	50
China	Conroy	Egdal	48
Japan	Gabie	Alessandone	47
Ukraine	Devlen	Chaperlin	54
France	Babbette	Turner	51
Czech Republic	Virgil	Scotney	52
Tajikistan	Zorina	Bedow	49
China	Aidan	Rudeyeard	50
Ireland	Saunder	MacLice	48
France	Waly	Brunstan	53
China	Gisele	Enns	52
Peru	Mina	Winchester	48
Japan	Torie	MacShirrie	50
Russia	Benjamen	Kenford	51
China	Etan	Burn	53
Russia	Merralee	Chaperlin	38
Indonesia	Lanny	Malam	49
Canada	Wilhelm	Deeprose	54
Czech Republic	Lari	Hillhouse	48
China	Ossie	Woodley	52
Macedonia	April	Tyer	50
Vietnam	Madelon	Dansey	53
Ukraine	Korella	McNamee	52
Jamaica	Linnea	Cannam	43
China	Mart	Coling	52
Indonesia	Marna	Causbey	47
China	Berni	Daintier	55
Poland	Cynthia	Hassell	49
Canada	Carma	Schule	49
Indonesia	Malia	Blight	48

country	first_name	last_name	test_score
China	Paulo	Seivertsen	47
Niger	Kaylee	Hearley	54
Japan	Maure	Jandak	46
Argentina	Foss	Feavers	45
Venezuela	Ron	Leggitt	60
Russia	Flint	Gokes	40
China	Linet	Conelly	52
Philippines	Nikolas	Birtwell	57
Australia	Eduard	Leipelt	53

Author

Rav Ahuja

Lakshmi Holla

Other Contributor(s)

Malika

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2022-03-04	1.0	Lakshmi Holla	Created lab using sqlite3
