**Accessing Databases with SQL Magic**[**¶**](https://jupyterlab-0-labs-prod-jupyterlab-us-east-3.labs.cognitiveclass.ai/user/akulasamson/lab/tree/labs/DB0201EN/DB0201EN-Week3-1-3-SQLmagic-v3-py.ipynb#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](https://github.com/catherinedevlin/ipython-sql" \t "_blank) 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.**

[1]:



**!**pip install sqlalchemy**==**1.3.9

**!**pip install ibm\_db\_sa

Requirement already satisfied: sqlalchemy==1.3.9 in /home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (1.3.9)

Requirement already satisfied: ibm\_db\_sa in /home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (0.3.3)

Requirement already satisfied: sqlalchemy>=0.7.3 in /home/jupyterlab/conda/envs/python/lib/python3.6/site-packages (from ibm\_db\_sa) (1.3.9)

[2]:



**%**load\_ext sql

**Now we have access to SQL magic. With our first SQL magic command, we'll connect to a Db2 database. However, in order to do that, you'll first need to retrieve or create your credentials to access your Db2 database.**



**This image shows the location of your connection string if you're using Db2 on IBM Cloud. If you're using another host the format is: username:password@hostname:port/database-name?security=SSL**

[4]:



*# Enter your Db2 credentials in the connection string below*

*# Recall you created Service Credentials in Part III of the first lab of the course in Week 1*

*# i.e. from the uri field in the Service Credentials copy everything after db2:// (but remove the double quote at the end)*

*# for example, if your credentials are as in the screenshot above, you would write:*

*# %sql ibm\_db\_sa://my-username:my-password@hostname:port/BLUDB?security=SSL*

*# Note the ibm\_db\_sa:// prefix instead of db2://*

*# This is because JupyterLab's ipython-sql extension uses sqlalchemy (a python SQL toolkit)*

*# which in turn uses IBM's sqlalchemy dialect: ibm\_db\_sa*

**%**sql ibm\_db\_sa:**//**kvn77701:INISp92TCJkh15dQ**@**54a2f15b**-**5c0f**-**46df**-**8954**-**7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733**/**BLUDB**?**security**=**SSL

[4]:

'Connected: kvn77701@BLUDB'

**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.**

[5]:



**%%**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\_score)

**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)

​

\* ibm\_db\_sa://kvn77701:\*\*\*@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB

Done.

99 rows affected.

[5]:

[]

**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.**

[6]:



country **=** "Canada"

**%**sql **select** **\*** **from** INTERNATIONAL\_STUDENT\_TEST\_SCORES **where** country **=** :country

\* ibm\_db\_sa://kvn77701:\*\*\*@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB

Done.

[6]:

| **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**

**Did you know? IBM Watson Studio lets you build and deploy an AI solution, using the best of open source and IBM software and giving your team a single environment to work in.**[**Learn more here.**](https://cocl.us/ibm_watson_studio_infobox)

**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.**

[7]:



test\_score\_distribution **=** **%**sql SELECT test\_score **as** "Test Score", count(**\***) **as** "Frequency" **from** INTERNATIONAL\_STUDENT\_TEST\_SCORES GROUP BY test\_score;

test\_score\_distribution

\* ibm\_db\_sa://kvn77701:\*\*\*@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB

Done.

[7]:

| **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.**

[8]:



dataframe **=** test\_score\_distribution.DataFrame()

​

**%**matplotlib inline

*# uncomment the following line if you get an module error saying seaborn not found*

*# !pip install seaborn*

**import** seaborn

​

plot **=** seaborn.barplot(x**=**'Test Score',y**=**'Frequency', data**=**dataframe)

Now you know how to work with Db2 from within JupyterLab notebooks using SQL "magic"!

[9]:



**%%**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\_SCORES;

\* ibm\_db\_sa://kvn77701:\*\*\*@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB

Done.

[9]:

| **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 |
| 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 |