Akram Alzaghir

Accessing Databases with SQL Magic

Estimated time needed: **15** minutes

Objectives

After completing this lab you will be able to:

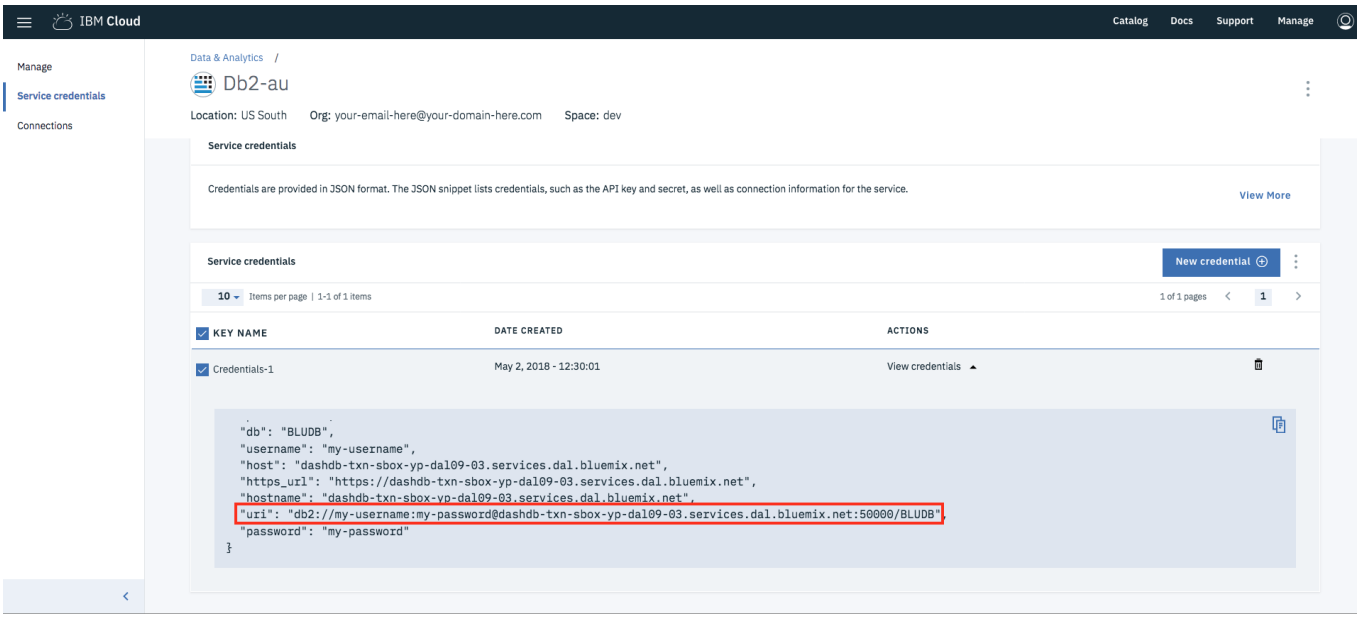
- Perform simplified database access using SQL "magic"

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) 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.

In [4]:

```
%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

In [5]:

```
# 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@dashdb-txn-sbox-yp-dal09-03.services.dal.blu
emix.net:50000/BLUDB
# Note the ibm_db_sa:// prefix instead of db2://
# This is because JupyterLab's ipython-sql extension uses sqlalchemy (a python SQL tool
kit)
# which in turn uses IBM's sqlalchemy dialect: ibm_db_sa
%sql ibm_db_sa://tnm91075:krrsj3j7zjn%40nphp@dashdb-txn-sbox-yp-dal09-12.services.dal.b
luemix.net:50000/BLUDB
```

Out[5]:

```
'Connected: tnm91075@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.

In [6]:

```
%%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_sco
re)
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://tnm91075:***@dashdb-txn-sbox-yp-dal09-12.services.dal.bluem
ix.net:50000/BLUDB
(ibm_db_dbi.ProgrammingError) ibm_db_dbi::ProgrammingError: Statement Exec
ute Failed: [IBM][CLI Driver][DB2/LINUXX8664] SQL0601N  The name of the ob
ject to be created is identical to the existing name "TNM91075.INTERNATION
AL_STUDENT_TEST_SCORES" of type "TABLE".  SQLSTATE=42710 SQLCODE=-601
[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/f405)
```

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 [105]:

```
# Cell magics: start with a double %% sign and apply to the entire cell
# Line magics: start with a single % (percent) sign and apply to a particular line in a
cell
country = "Canada"
%sql select * from INTERNATIONAL_STUDENT_TEST_SCORES where country = :country

# or you can use this command only, and ignore the first command too
#%sql select * from INTERNATIONAL_STUDENT_TEST_SCORES where country = 'Canada'
```

```
 * ibm_db_sa://tnm91075:***@dashdb-txn-sbox-yp-dal09-12.services.dal.bluem
ix.net:50000/BLUDB
Done.
```

Out[105]:

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

In [40]:

```
%sql select country, count(country)as total from INTERNATIONAL_STUDENT_TEST_SCORES grou
p by country;
# or you can use this command only, and ignore the first command too
#%sql select * from INTERNATIONAL_STUDENT_TEST_SCORES where country = 'Canada'
```

```
 * ibm_db_sa://tnm91075:***@dashdb-txn-sbox-yp-dal09-12.services.dal.bluem
ix.net:50000/BLUDB
Done.
```

Out[40]:

country	total
Argentina	1
Armenia	1
Australia	1
Azerbaijan	1
Brazil	2
Canada	3
China	23
Czech Republic	2
France	5
Ghana	1
Greece	1
Guatemala	1
Indonesia	8
Ireland	2
Jamaica	1
Japan	3
Latvia	2
Macedonia	1
Malaysia	1
Mexico	1
Moldova	1
Niger	1
Norway	1
Panama	1
Papua New Guinea	1
Peru	1
Philippines	5
Poland	7
Portugal	1
Russia	6
Sweden	2
Switzerland	1
Tajikistan	1
Uganda	1
Ukraine	3
United States	3
Venezuela	1

	country	total
	Vietnam	1

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 [109]:

```
#%%sql
test_score_distribution = %sql SELECT test_score as "Test Score", count(*) as "Frequenc
y" from INTERNATIONAL_STUDENT_TEST_SCORES GROUP BY test_score;
test_score_distribution
# also we use like this
#test_score_distribution = %sql SELECT test_score, count(test_score) as "Frequency" fro
m INTERNATIONAL_STUDENT_TEST_SCORES GROUP BY test_score;
#test_score_distribution
```

```
 * ibm_db_sa://tnm91075:***@dashdb-txn-sbox-yp-dal09-12.services.dal.bluem
ix.net:50000/BLUDB
Done.
```

Out[109]:

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

In [100]:

```
# Cell magics: start with a double %% sign and apply to the entire cell
# Line magics: start with a single % (percent) sign and apply to a particular line in a cell
%sql SELECT country, test_score from INTERNATIONAL_STUDENT_TEST_SCORES where test_score = (select max(test_score) from INTERNATIONAL_STUDENT_TEST_SCORES)
```

```
 * ibm_db_sa://tnm91075:***@dashdb-txn-sbox-yp-dal09-12.services.dal.bluemix.net:50000/BLUDB
Done.
```

Out[100]:

country	test_score
Venezuela	60

In [99]:

```
# Cell magics: start with a double %% sign and apply to the entire cell
# Line magics: start with a single % (percent) sign and apply to a particular line in a cell
%sql SELECT * from INTERNATIONAL_STUDENT_TEST_SCORES where test_score = (select max(test_score) from INTERNATIONAL_STUDENT_TEST_SCORES)
```

```
 * ibm_db_sa://tnm91075:***@dashdb-txn-sbox-yp-dal09-12.services.dal.bluemix.net:50000/BLUDB
Done.
```

Out[99]:

country	first_name	last_name	test_score
Venezuela	Ron	Leggitt	60

In [97]:

```
# Cell magics: start with a double %% sign and apply to the entire cell
# Line magics: start with a single % (percent) sign and apply to a particular line in a cell
%sql SELECT avg(test_score) as avg_score from INTERNATIONAL_STUDENT_TEST_SCORES
```

```
 * ibm_db_sa://tnm91075:***@dashdb-txn-sbox-yp-dal09-12.services.dal.bluemix.net:50000/BLUDB
Done.
```

Out[97]:

avg_score
49.737373

In [93]:

```
%%sql
-- Cell magics: start with a double %% sign and apply to the entire cell
-- Line magics: start with a single % (percent) sign and apply to a particular line in
 a cell
-- retrieves all first name, last name and test score  with average score in every row
 SELECT first_name, last_name, test_score, (select avg(test_score) from INTERNATIONAL_S
TUDENT_TEST_SCORES) as avg_score from INTERNATIONAL_STUDENT_TEST_SCORES
```

```
 * ibm_db_sa://tnm91075:***@dashdb-txn-sbox-yp-dal09-12.services.dal.bluem
ix.net:50000/BLUDB
Done.
```

Out[93]:

first_name	last_name	test_score	avg_score
Marshall	Bernadot	54	49.737373
Celinda	Malkin	51	49.737373
Guillermo	Furze	53	49.737373
Aharon	Tunnow	48	49.737373
Bail	Goodwin	46	49.737373
Cole	Winteringham	49	49.737373
Emlyn	Erricker	55	49.737373
Cathee	Sivewright	49	49.737373
Barny	Ingerson	57	49.737373
Sharla	Papaccio	55	49.737373
Stella	Youens	51	49.737373
Julio	Buesden	48	49.737373
Tiffie	Cosely	58	49.737373
Auroora	Stiffell	45	49.737373
Clarita	Huet	52	49.737373
Shannon	Goulden	45	49.737373
Emylee	Privost	50	49.737373
Madelina	Burk	49	49.737373
Saunderson	Root	58	49.737373
Bo	Waring	55	49.737373
Hollis	Domotor	45	49.737373
Robbie	Collip	46	49.737373
Davon	Donisi	46	49.737373
Cristabel	Radeliffe	48	49.737373
Wallis	Bartleet	58	49.737373
Arleen	Stailey	38	49.737373
Mendel	Grumble	58	49.737373
Sallyann	Exley	51	49.737373
Kain	Swaite	46	49.737373
Alonso	Bulteel	45	49.737373
Anatol	Tankus	51	49.737373
Coralyn	Dawkins	48	49.737373
Deanne	Edwinson	45	49.737373
Georgiana	Epple	51	49.737373
Bartlet	Breese	56	49.737373
Idalina	Lukash	50	49.737373
Livvie	Flory	54	49.737373

first_name	last_name	test_score	avg_score
Nonie	Borit	48	49.737373
Clio	Mugg	47	49.737373
Westley	Measor	48	49.737373
Katrinka	Sibbert	51	49.737373
Valentia	Mounch	50	49.737373
Sheilah	Hedditch	53	49.737373
Itch	Jubb	50	49.737373
Stesha	Garnson	53	49.737373
Cristionna	Wadmore	46	49.737373
Lianna	Gatward	43	49.737373
Tanney	Vials	48	49.737373
Alma	Zavittieri	44	49.737373
Alvira	Tamas	50	49.737373
Shanon	Peres	45	49.737373
Maisey	Lynas	53	49.737373
Kip	Hothersall	46	49.737373
Cash	Landis	48	49.737373
Kennith	Digance	45	49.737373
Ulberto	Riggeard	48	49.737373
Judy	Gilligan	49	49.737373
Tod	Trevaskus	52	49.737373
Herold	Heggs	44	49.737373
Verney	Note	50	49.737373
Temp	Ribey	50	49.737373
Conroy	Egdal	48	49.737373
Gabie	Alessandone	47	49.737373
Devlen	Chaperlin	54	49.737373
Babbette	Turner	51	49.737373
Virgil	Scotney	52	49.737373
Zorina	Bedow	49	49.737373
Aidan	Rudeyeard	50	49.737373
Saunder	MacLice	48	49.737373
Waly	Brunstan	53	49.737373
Gisele	Enns	52	49.737373
Mina	Winchester	48	49.737373
Torie	MacShirrie	50	49.737373
Benjamen	Kenford	51	49.737373
Etan	Burn	53	49.737373
Merralee	Chaperlin	38	49.737373

first_name	last_name	test_score	avg_score
Lanny	Malam	49	49.737373
Wilhelm	Deeprose	54	49.737373
Lari	Hillhouse	48	49.737373
Ossie	Woodley	52	49.737373
April	Tyer	50	49.737373
Madelon	Dansey	53	49.737373
Korella	McNamee	52	49.737373
Linnea	Cannam	43	49.737373
Mart	Coling	52	49.737373
Marna	Causbey	47	49.737373
Berni	Daintier	55	49.737373
Cynthia	Hassell	49	49.737373
Carma	Schule	49	49.737373
Malia	Blight	48	49.737373
Paulo	Seivertsen	47	49.737373
Kaylee	Hearley	54	49.737373
Maure	Jandak	46	49.737373
Foss	Feavers	45	49.737373
Ron	Leggitt	60	49.737373
Flint	Gokes	40	49.737373
Linet	Conelly	52	49.737373
Nikolas	Birtwell	57	49.737373
Eduard	Leipelt	53	49.737373

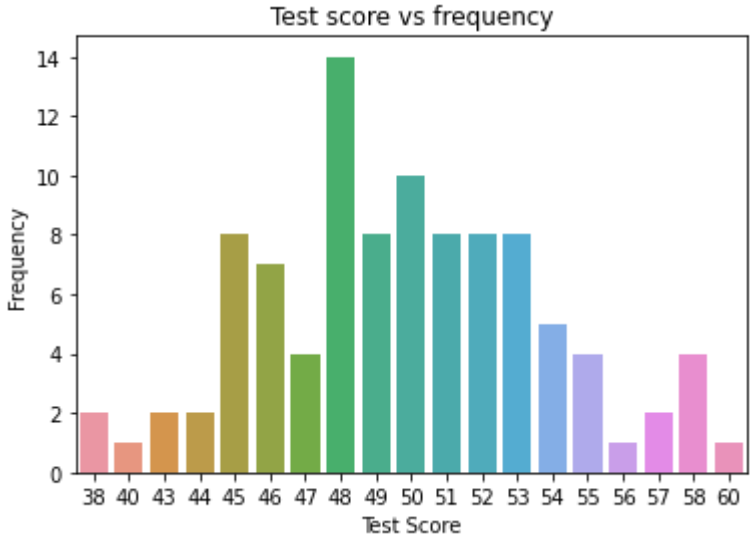
You can easily convert a SQL query result to a pandas dataframe using the `DataFrame()` method. Dataframe objects are much more versatile than SQL query result objects. For example, we can easily graph our test score distribution after converting to a dataframe.

In [82]:

```
dataframe = test_score_distribution.DataFrame()

%matplotlib inline
# uncomment the following line if you get an module error saying seaborn not found
#!pip install seaborn
# seaborn.barplot is a visualization of the group by action based on Matplotlib
import seaborn as sns

plot = sns.barplot(x='Test Score',y='Frequency', data=dataframe).set_title("Test score
 vs frequency")
```



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

In [104]:

```
%%sql
-- Cell magics: start with a double %% sign and apply to the entire cell
-- Line magics: start with a single % (percent) sign and apply to a particular line in
 a cell
-- 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_SCORE
S;
```

```
 * ibm_db_sa://tnm91075:***@dashdb-txn-sbox-yp-dal09-12.services.dal.bluem
ix.net:50000/BLUDB
Done.
```

Out[104]:

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

country	first_name	last_name	test_score
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

country	first_name	last_name	test_score
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

Author

Rav Ahuja (https://www.linkedin.com/in/ravahuja/)

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-07-17	2.0	Lavanya	Moved lab to course repo in GitLab