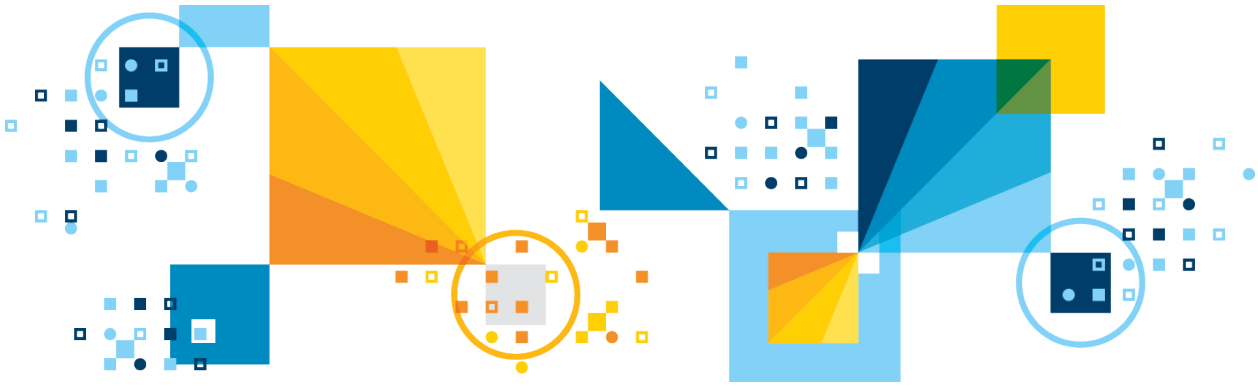


Db2 Hands On Labs

Db2 Warehouse

die einfache Art, komplexe Analysen durchzuführen

Andreas Weininger, IBM, andreas.weininger@de.ibm.com
Stefan Hummel, IBM, stefan.hummel@de.ibm.com

Table of Contents

Objectives of this lab 3
Background of this Lab 4
How to use the VMWare Image 5
Preparation: Download scripts and documentation 5
Lab 1: Getting Access to Bluemix 6
Lab 2: Initializing Db2 8
Lab 3: Creating Tables 11
Lab 4: Loading Data 15
Lab 5: Executing Queries 18
Lab 6: Remote Access to Db2 21
Lab 7: Monitoring Db2 23
Lab 8: Linear Regression in SQL 26
Lab 9: Linear Regression with R 30
Lab 10: Linear Regression with Python 34
Lab 11: Machine Learning with k-means in Db2 36
Lab 12: Accessing Db2 from Spark and Python 43
Lab 13: Analyzing data with R in RStudio 56
Appendix I: Create Bluemix and DSX account 63
Appendix II: Start vmware image and docker container 67
Acknowledgements and Disclaimers 68

Objectives of this lab

The goal of this lab is to show how developers can perform advanced analytics with data stored in Db2, a managed analytical database offering for the cloud and on-premises by IBM.

Attendees will learn:

- How can Db2 used for creating fast and scalable solutions?
- How can Spark be used with Db2 for creating applications?
- How can data science / machine learning algorithms be implemented with Db2?
- How can new Db2 features like user defined aggregates be used by developers?
- How can R and Python be used with Db2? How can Db2 be integrated in the Data Science Experience?

Since Db2 will be used on Bluemix, the Platform as a Service (PaaS) offering by IBM, the first objective of this lab is to show how to create an account on Bluemix and how to login to Bluemix.

The lab will teach you

- how to create tables in Db2,
- how to load data into these tables, and
- how to analyze the data loaded.
- how to configure remote connections to Db2,
- and how to monitor queries running on Db2

Scripts with solutions for all of the lab exercises are provided, but it is recommended that you first try to create a script on your own before using the lab script.

Background of this Lab

This lab uses three datasets:

- The first part of the lab uses the schema, the load files and the queries of the Star Schema Benchmark defined by Pat O'Neil (http://www.cs.umb.edu/~poneil/StarSchemaB.PDF). You find a copy of it on your desktop in the VMware image.

  The Star Schema Benchmark was chosen since it simulates a typical star schema data mart, and provides all the required information for our lab (schema, data, queries).

  The files for the Star Schema Benchmark are in the VMware image in the directory `/home/user/Advanced-Analytics/SSB`.

- The next part of the lab uses real data from the U.S. Geological Survey (http://www.usgs.gov/). The data used is measurements of water resources over time. There are many different kinds of measurements, but the following labs focus on the water discharge at a specific site. Site IDs are used instead of the full site names.

- The final part of the lab uses motor vehicle collision data from the New York Police Department available at https://data.cityofnewyork.us/widgets/h9gi-nx95

How to use the VMWare Image

There is a user account created for *user*. The password for this accounts is

User	Password
user	user

Commands which require root permission can be executed with the sudo command.

The browser Firefox is installed in the VMware Image. This browser should be used for working with Db2.

All exercises will be executed from the VMware Image. Therefore, the very first step is to start the VMware image and log in as *user*.

Preparation: Download scripts and documentation

1. Open a terminal
2. In your home directory `/home/user` enter:

```
git clone https://github.com/stefanhummel/Advanced-Analytics.git
```

3. The directory `/home/user/Advanced-Analytics` should look like this:

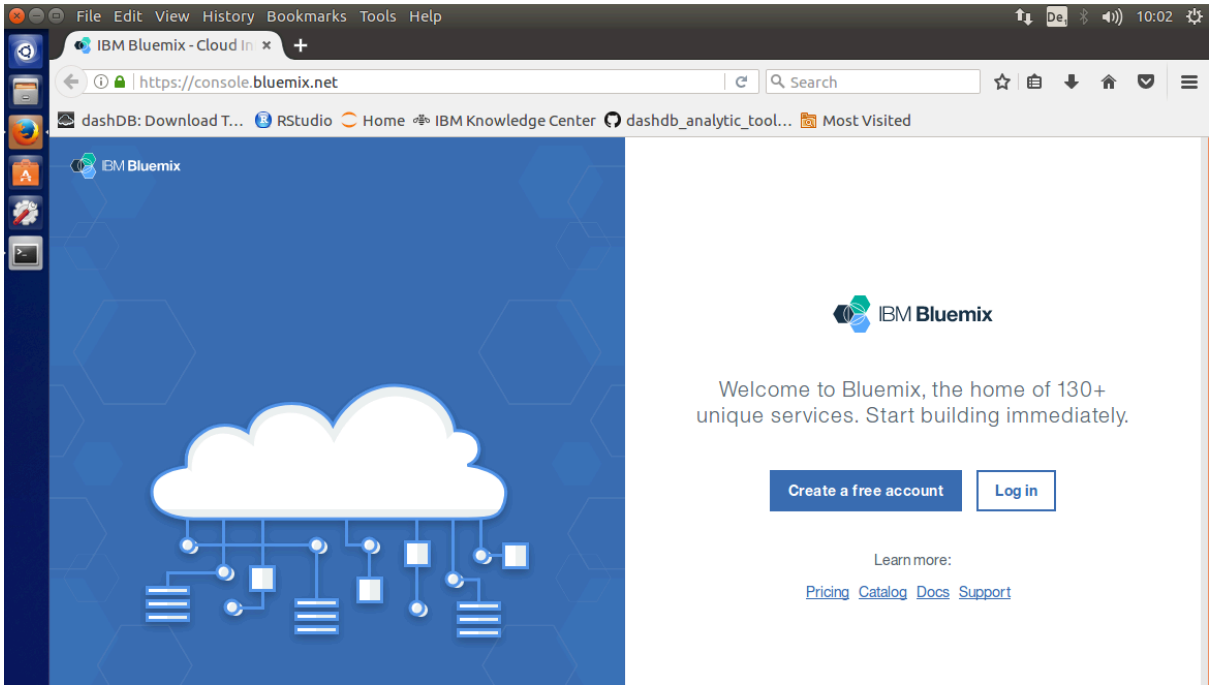
```
drwxrwxr-x  8 user user 4096 Aug 29 08:50 ./
drwxr-xr-x 24 user user 4096 Aug 29 08:49 ../
drwxrwxr-x  8 user user 4096 Aug 29 08:50 .git/
drwxrwxr-x  2 user user 4096 Aug 29 08:50 NYPD/
-rw-rw-r--  1 user user  216 Aug 29 08:50 README.md
drwxrwxr-x  5 user user 4096 Aug 29 08:50 SSB/
drwxrwxr-x  2 user user 4096 Aug 29 08:50 linear-regression/
drwxrwxr-x  4 user user 4096 Aug 29 08:50 spark/
drwxrwxr-x  2 user user 4096 Aug 29 08:50 user-defined-aggregates/
```

You find all the needed load files and scripts in this directory structure. The Documents folder contains the most current documentation of this lab.

Lab 1: Getting Access to Bluemix

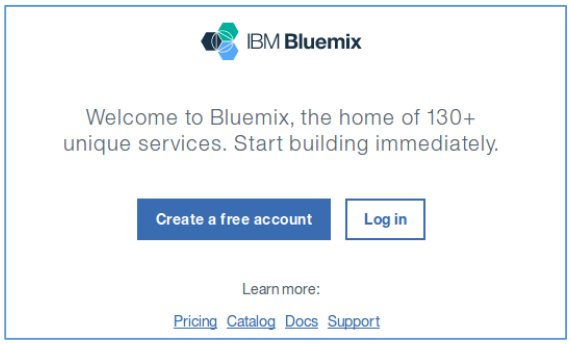
4. The next step in this lab is getting access to Bluemix, the IBM cloud platform for Platform as a Service (PaaS) offering. Start the Firebox browser by clicking the fourth icon from the top in the bar on the left.

5. Next enter the URL https://console.ng.bluemix.net/. You should get a screen like that:

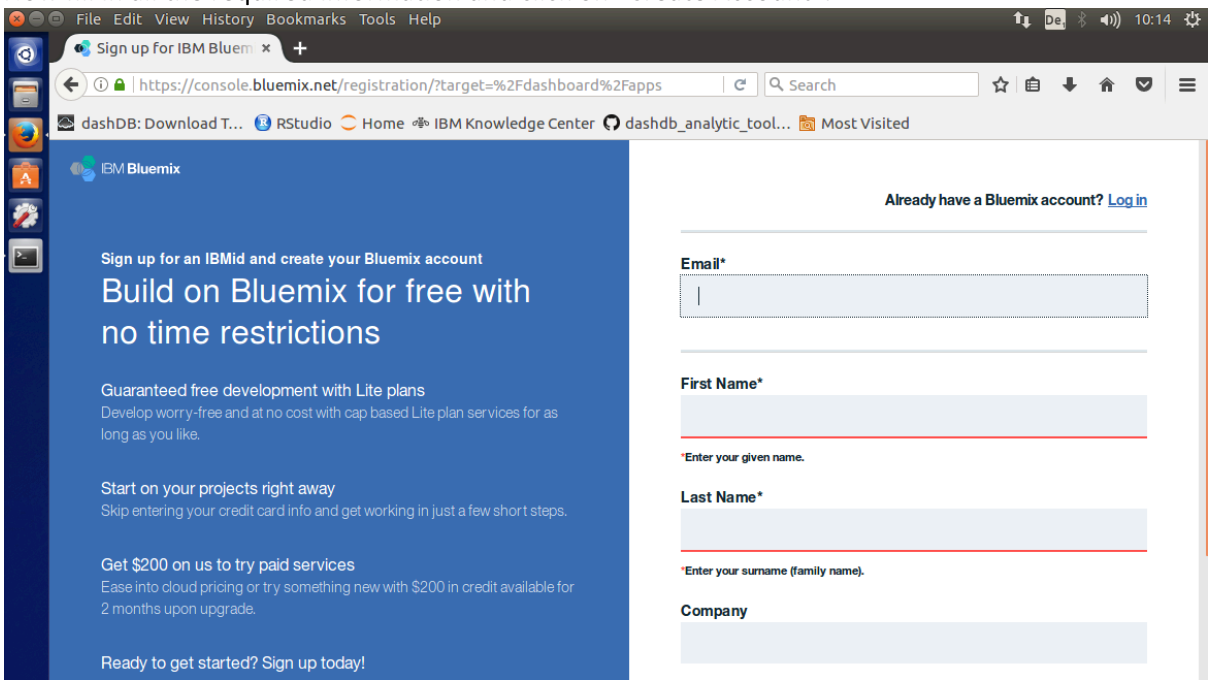


6. If you have already an account on Bluemix, you can log on directly by selecting “Log in” and continue with Lab 2. Otherwise select “Create a free account” and continue with step 7.

7. For registering select “Create a free account”:



8. Now fill in all the required information and click on “Create Account”:

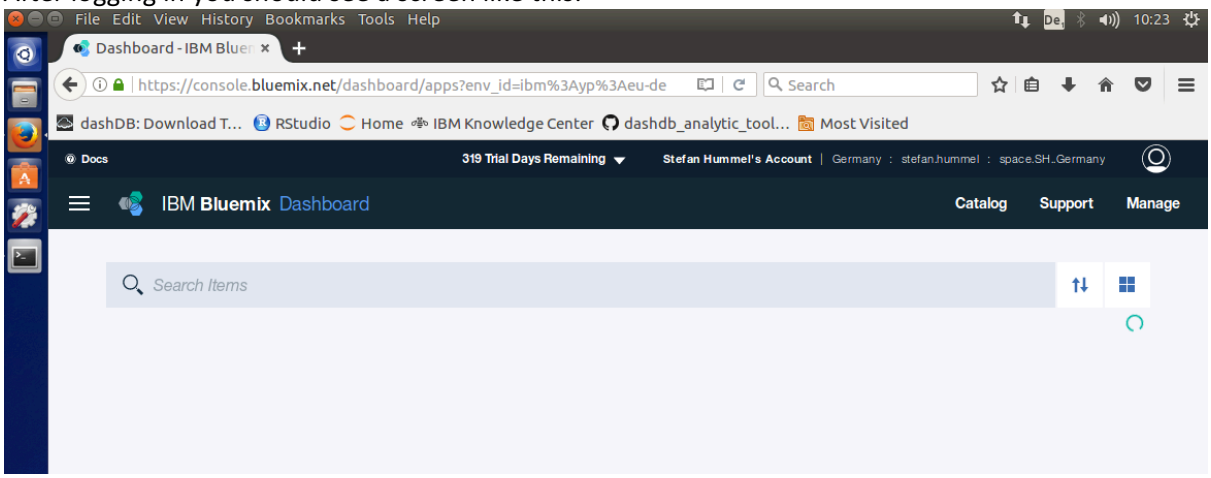


9. Welcome on IBM Bluemix. Now “Log in” and start working with.

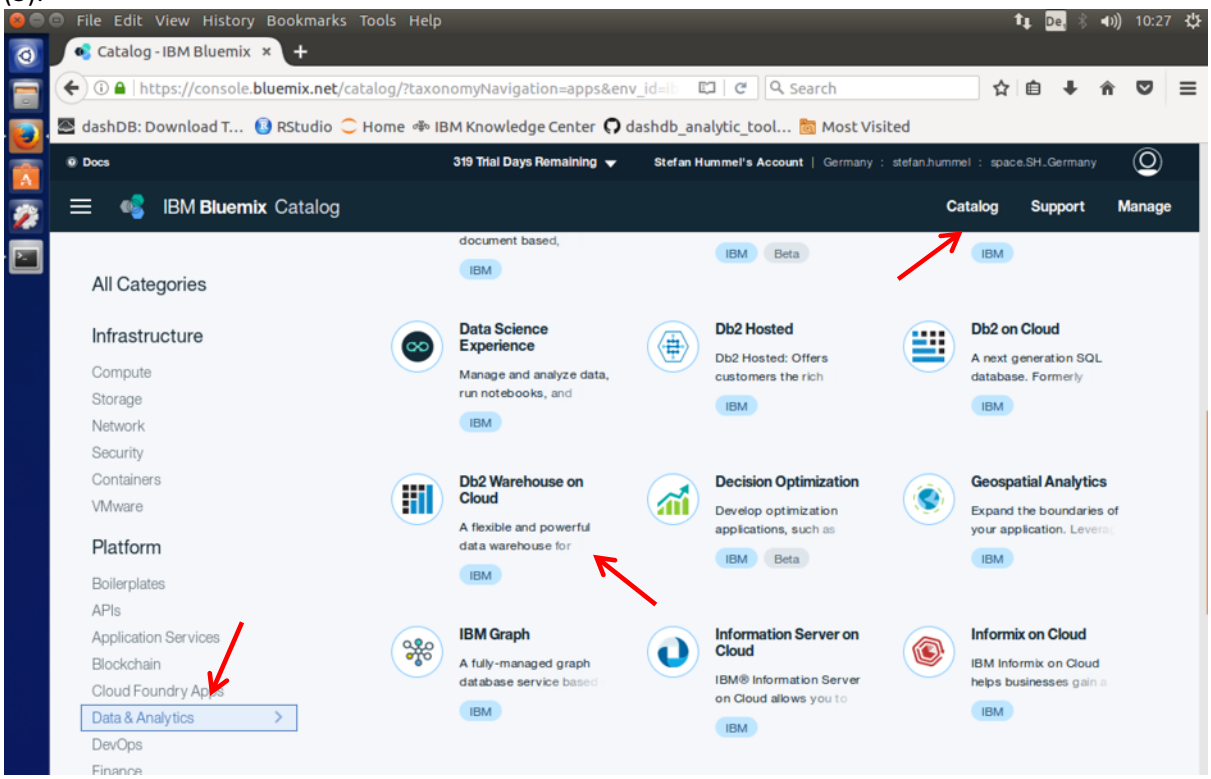
10. After you have clicked on “Log in”, you should get the following sign in screen (might be you get another screen first, where you can adjust your cookie preferences):

Lab 2: Initializing Db2

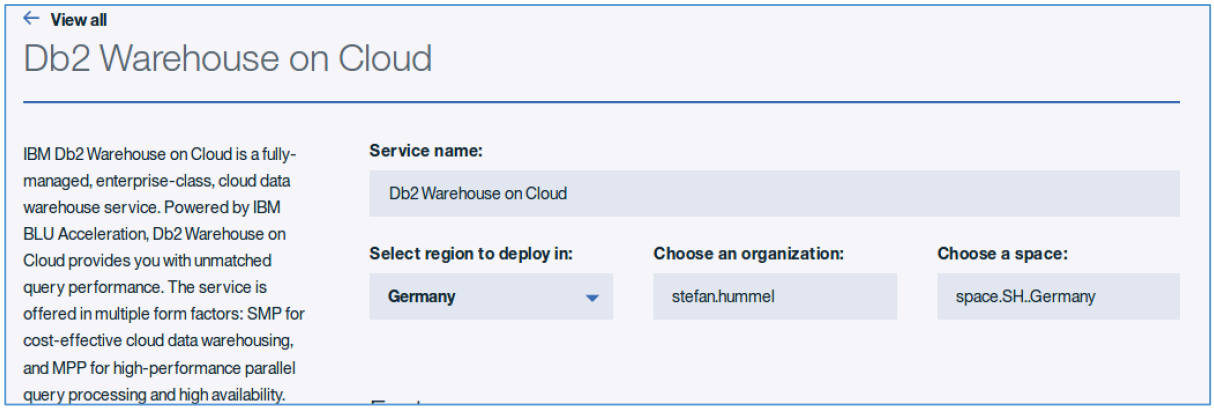
11. After logging in you should see a screen like this:



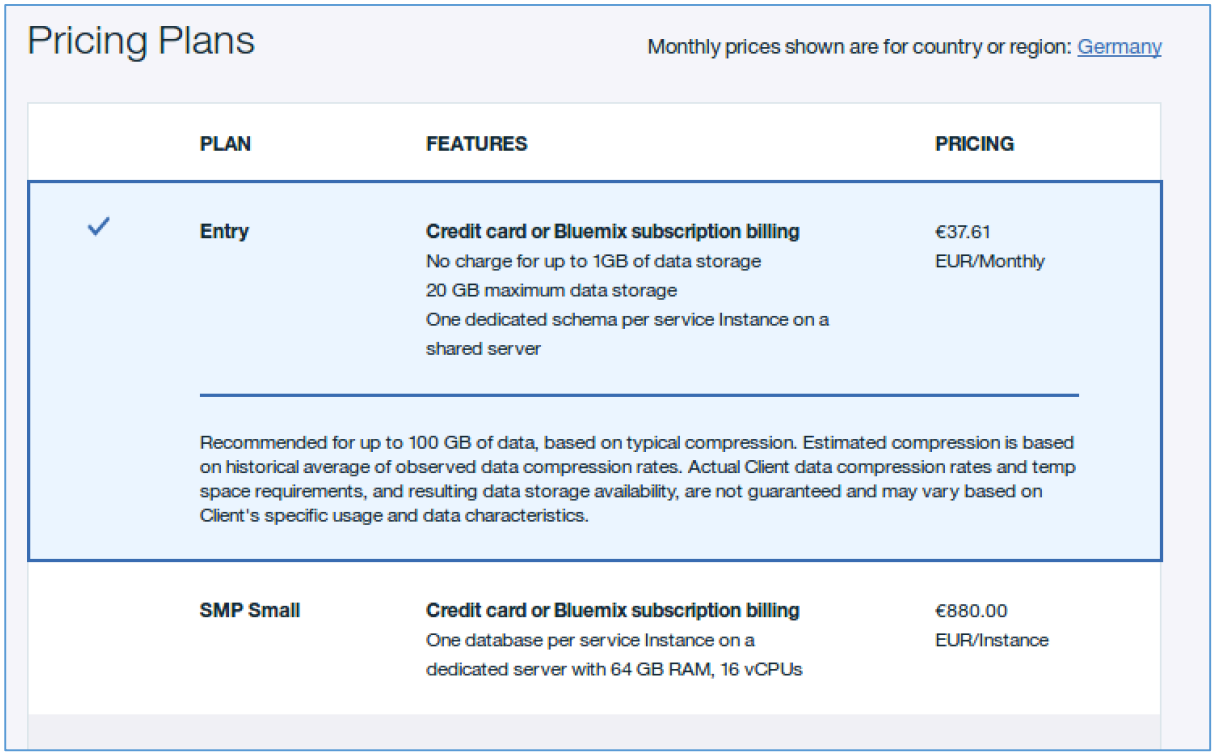
12. The next step is creating your own Db2 database. For this select “Catalog” (1) first. In the category “Platform” (2) select “Data & Analytics”. Then you will find the Bluemix service “Db2 Warehouse on Cloud” (3):



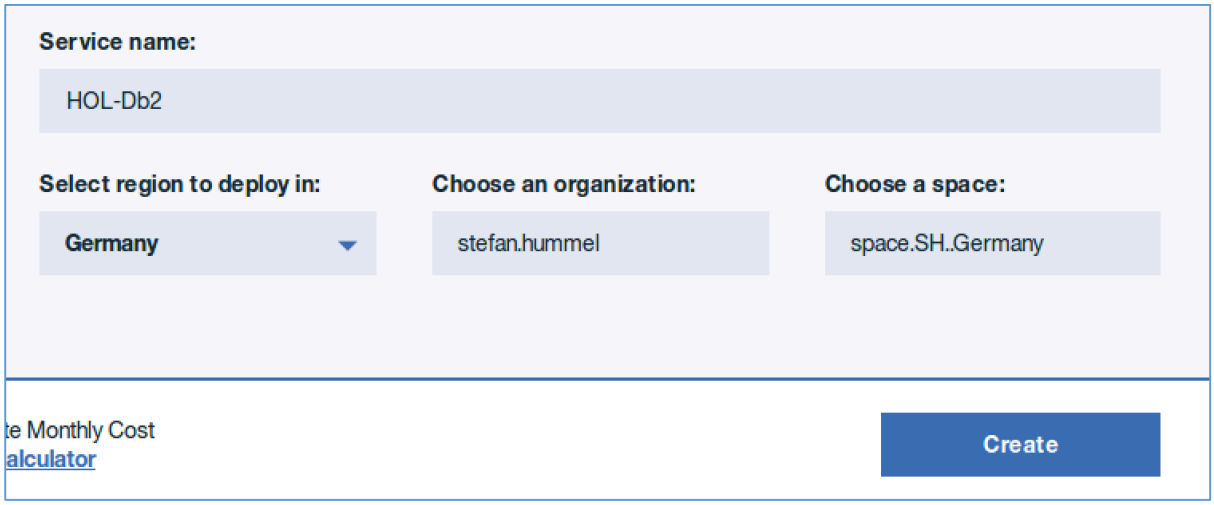
13. The following screen appears:



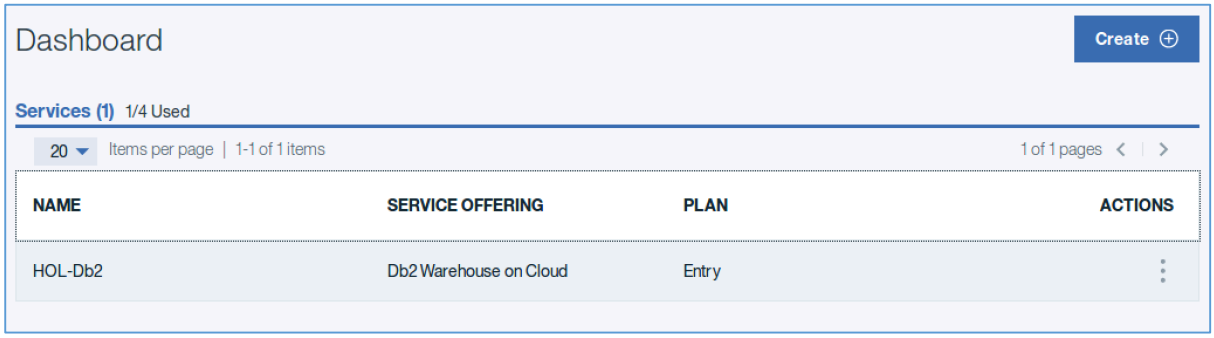
14. Scroll Down. You should see the license plans for Db2. For our lab we will use the ***Entry Plan*** which is already selected.
Kindly note that there is no charge for up to 1GB of data storage, a maximum data storage of 20 GB and one dedicated schema per service instance on a shared server.



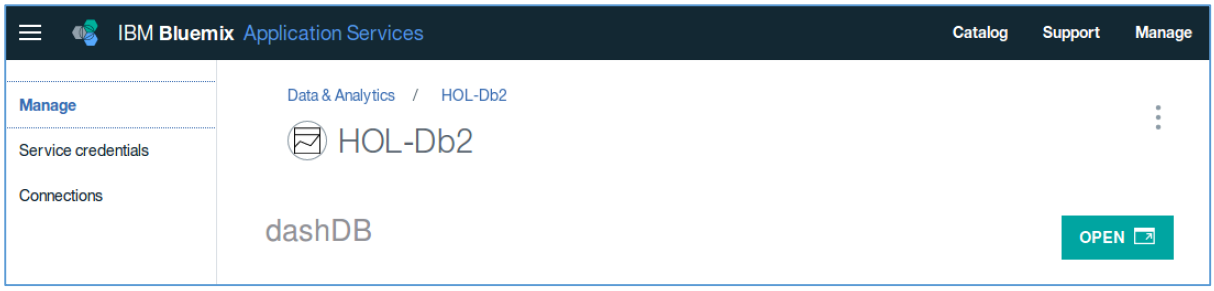
15. Scroll up again. Change the “Service name” to ***HOL -Db2*** and then click on ***Create***:



16. Now you see the initialized service *Db2 Warehouse for Cloud*. Launch the service by clicking on the corresponding line.



17. On the next screen are the service details of HOL-Db2. Here you can
- open the service (see next chapter)
- manage service credentials
- manage connections to applications

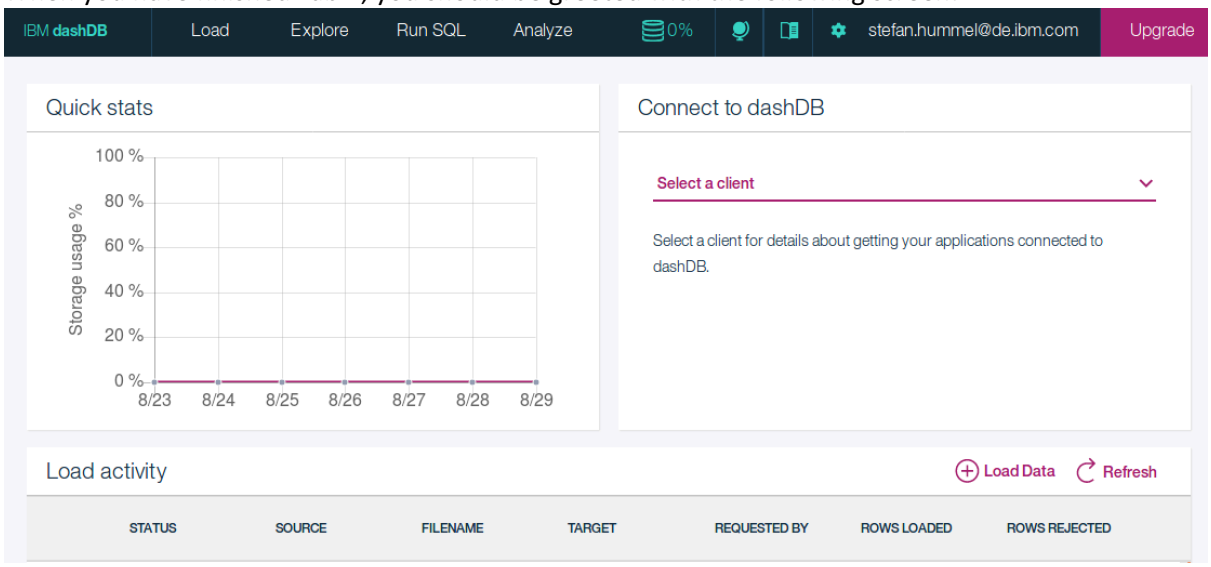


18. Continue with the next lab, in which you will learn to create tables in Db2.

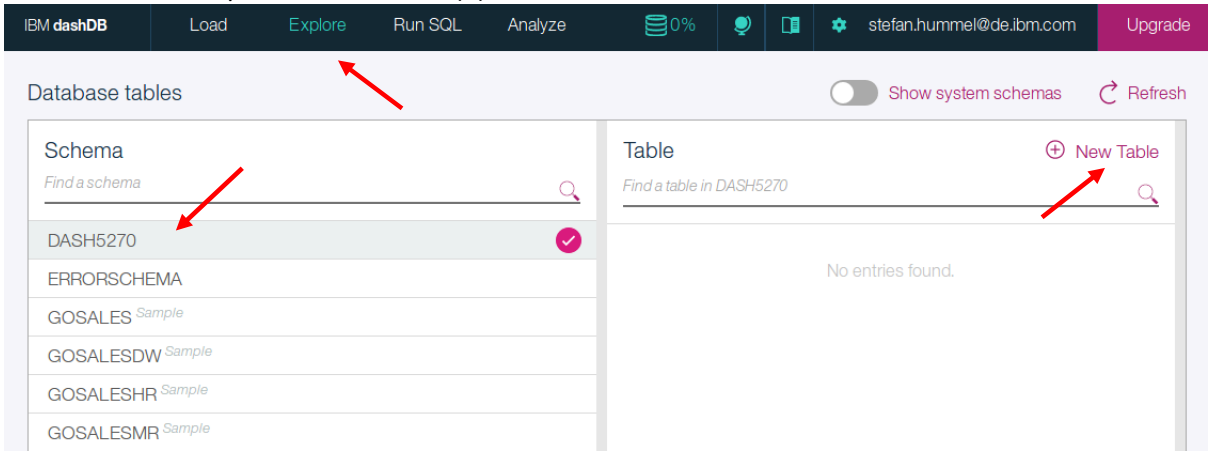
Lab 3: Creating Tables

In this exercise, you will learn several methods how to create tables.

19. When you have finished Lab 2, you should be greeted with the following screen:

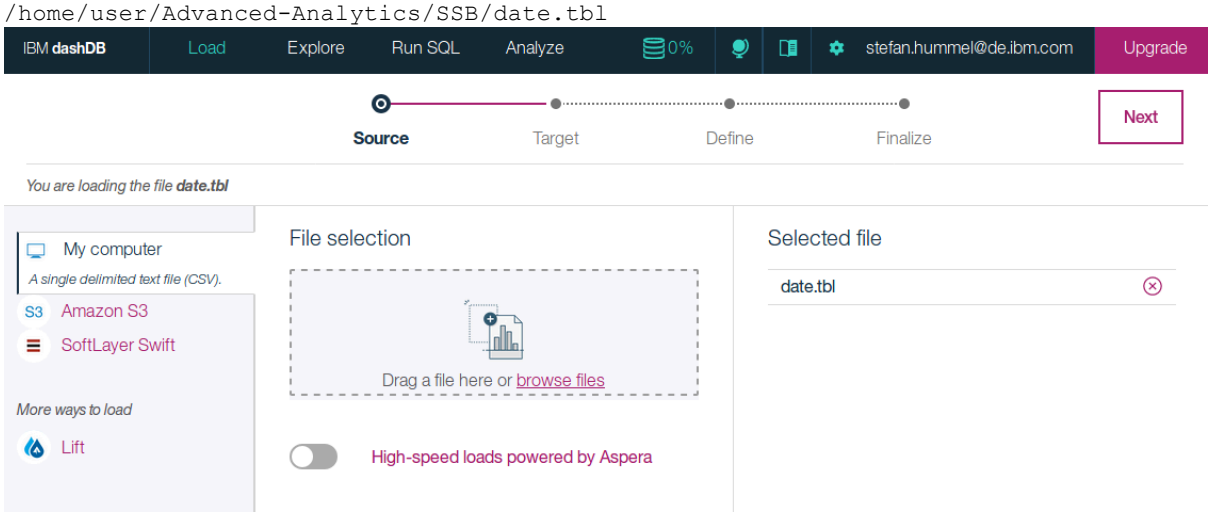


On the top you will find functions to load data, explore objects, run SQL statements and to analyze content. To create a table manually use the function “Explore” (1), select the schema beginning with “DASH…” (2) and then use the option “New Table” (3)

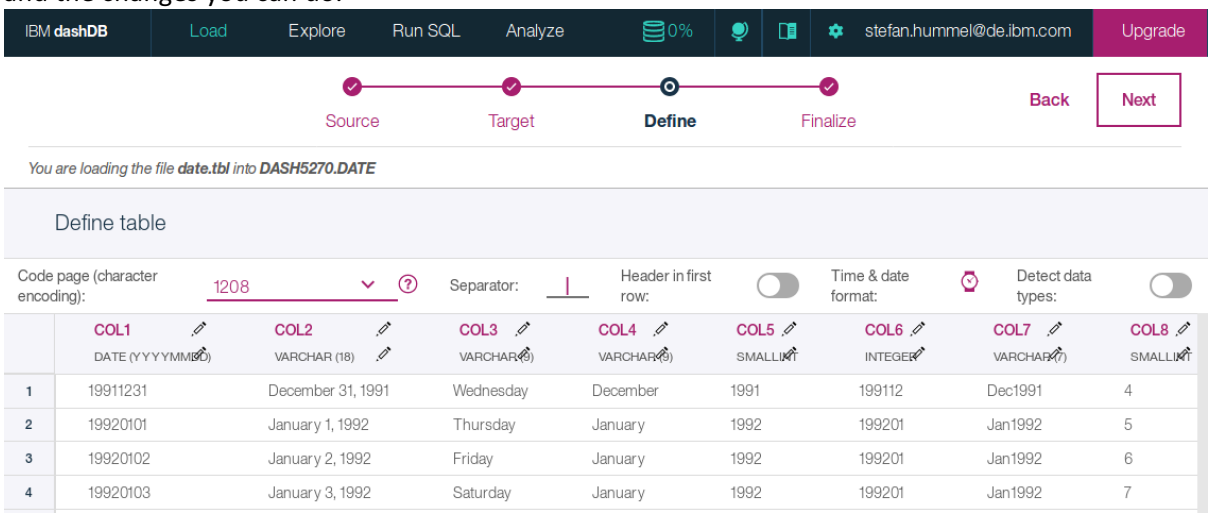


Define a table with the graphical interface as well as with the use of the function “Run SQL” and a custom DDL.

20. Now create a table and load data with the “Load” function on the top. Select the file `/home/user/Advanced-Analytics/SSB/date.tbl`



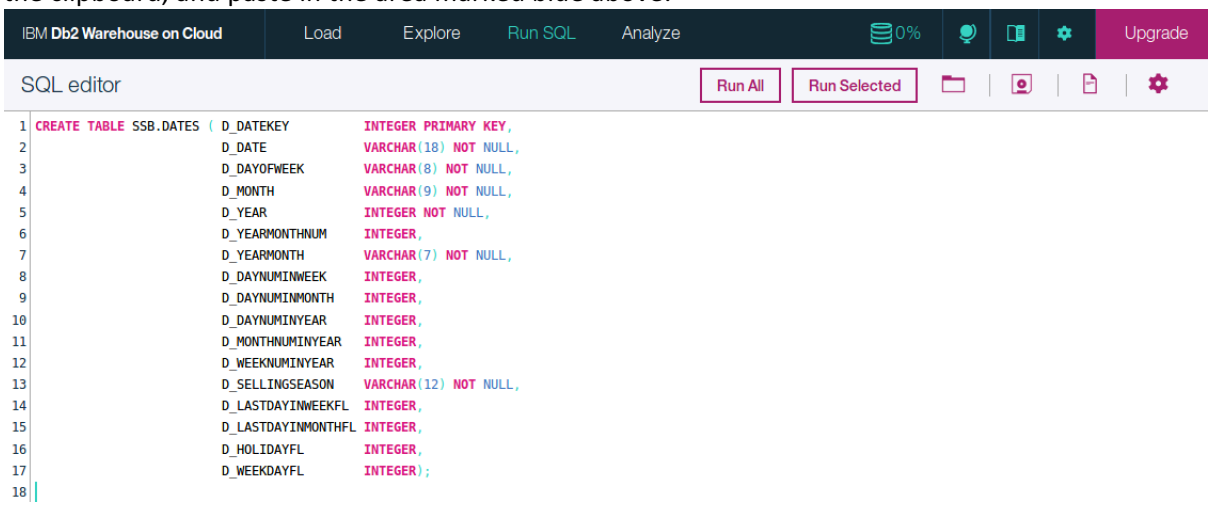
21. You see that date.tbl is selected. Make sure that “Header in first row” is disabled, and change the separator character for columns to the pipe symbol (“|”). Db2 will now generate the DDL based on the detected data and the changes you can do.



	COL1 DATE (YYYYMMDD)	COL2 VARCHAR (18)	COL3 VARCHAR(9)	COL4 VARCHAR(9)	COL5 SMALLINT	COL6 INTEGER	COL7 VARCHAR(7)	COL8 SMALLINT
1	19911231	December 31, 1991	Wednesday	December	1991	199112	Dec1991	4
2	19920101	January 1, 1992	Thursday	January	1992	199201	Jan1992	5
3	19920102	January 2, 1992	Friday	January	1992	199201	Jan1992	6
4	19920103	January 3, 1992	Saturday	January	1992	199201	Jan1992	7

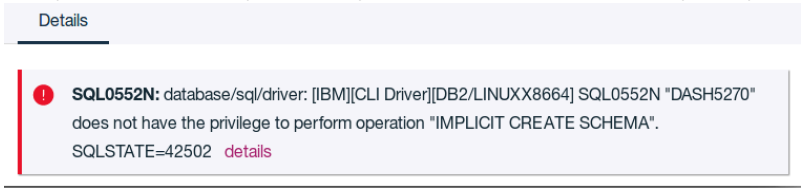
You could run this DDL and load the data in the Finalize step. But since we are expecting certain column names in our queries which we will run later, and since we have already a file with the DDL for the table “date” prepared in the file system, we will now learn the second method for creating tables.

22. Switch to the function “Run SQL” and use the DDL in `/home/user/Advanced-Analytics/SSB/ssb-dbgen-master/ssb.ddl` . Open this file with your favorite editor (vi, emacs) and copy the DDL for table `DATES` to the clipboard, and paste in the area marked blue above:



23. Unfortunately, you will get an error.
Why?

24. The problem is that you had specified the schema SSB explicitly, which isn't allowed in the Db2 entry plan.

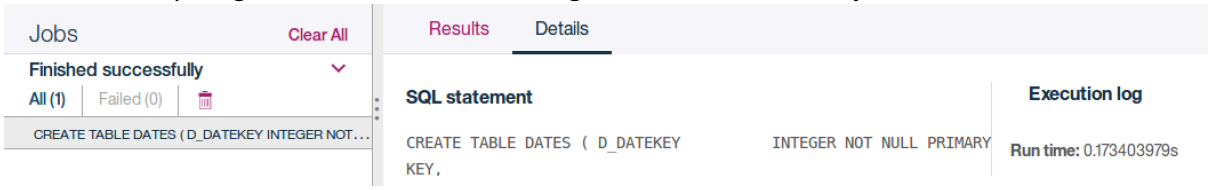


Remove the schema and try again:

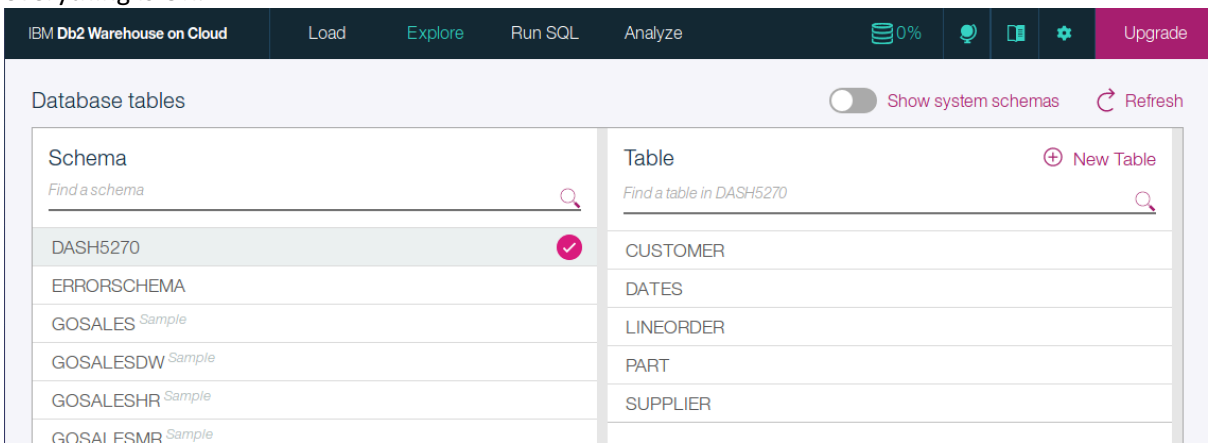
25. But we still get an error. We have D_DATEKEY defined as primary key. But Db2 requires – like Db2 – a primary key also as defined as NOT NULL, which we haven't done yet:

26. Let's insert the NOT NULL constraint and try again:

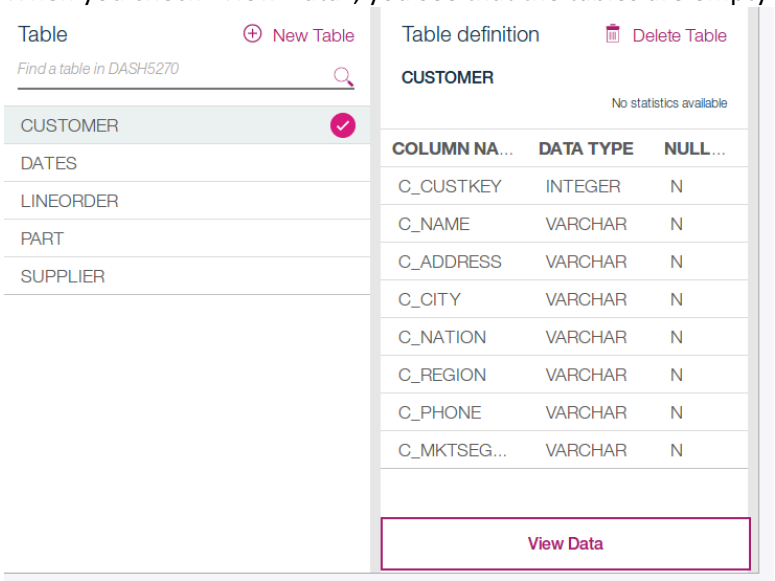
27. This time everything works and the table dates gets created. Check the job details.



28. Now create the remaining tables defined in ssb.ddl in the same way. You can specify several create table statements at once, each terminated via a semicolon (";"). Fix any problems in the DDL which you encounter in the same way as shown above. When you are done, you can check in the "Tables" screen whether everything is OK:



29. When you check "View Data", you see that the tables are empty as expected.

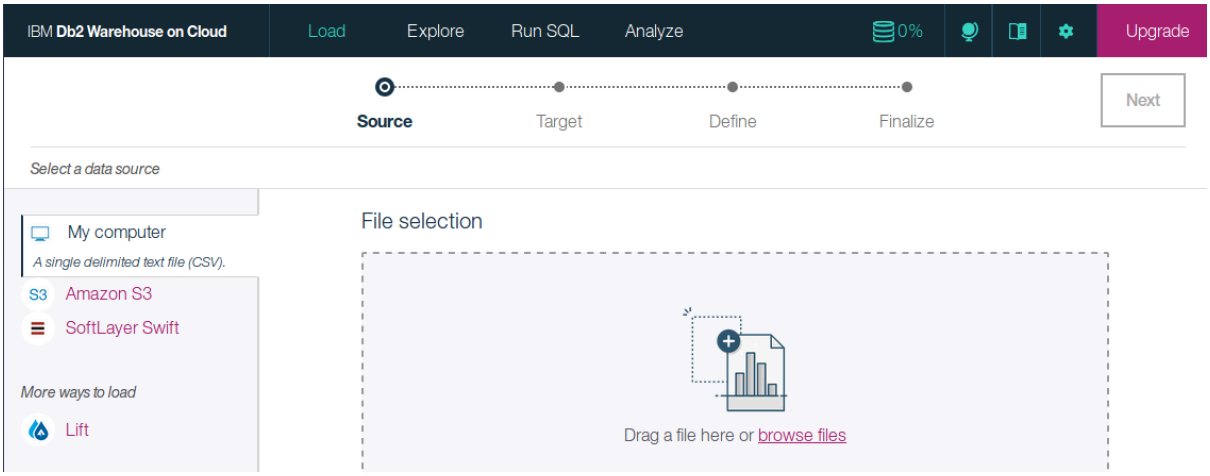


Now you are ready to load data which will be done in the next lab. When you have configured remote connections to Db2, which we will learn in a later lab, you can use those also for creating tables.

Lab 4: Loading Data

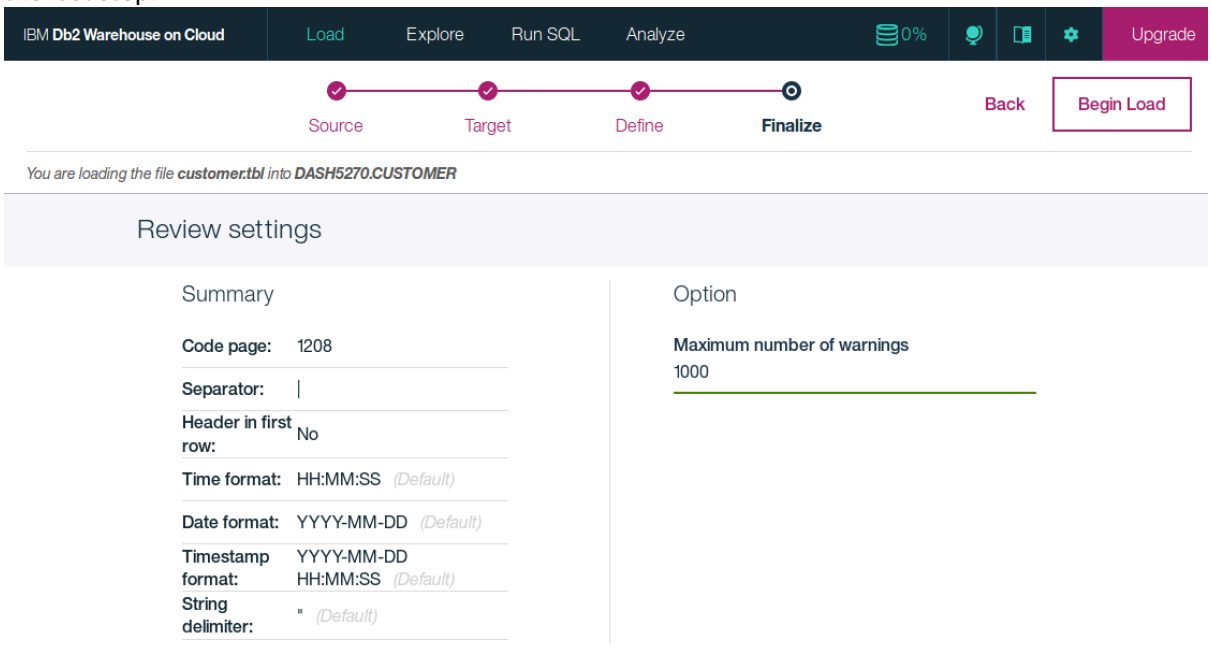
30. In this exercise, we will load data into the tables which we created in the previous lab. There are several methods for loading data:
- load from desktop
- load from cloud storage
- load with IBM Lift

If you have data already stored in the cloud, you can also load these data directly which is very fast. In this lab we load from our local virtual machine and from Softlayer Object Store.
There are two data sets prepared in our virtual machine, one with 1GB of data, the other with 100MB. Because we probably have a slow internet connection we will use the smaller data set.
The first step for loading is selecting “Load” from the toolbar, and then selecting “My computer”. You are shown the following input mask. Select “Browse Files”:



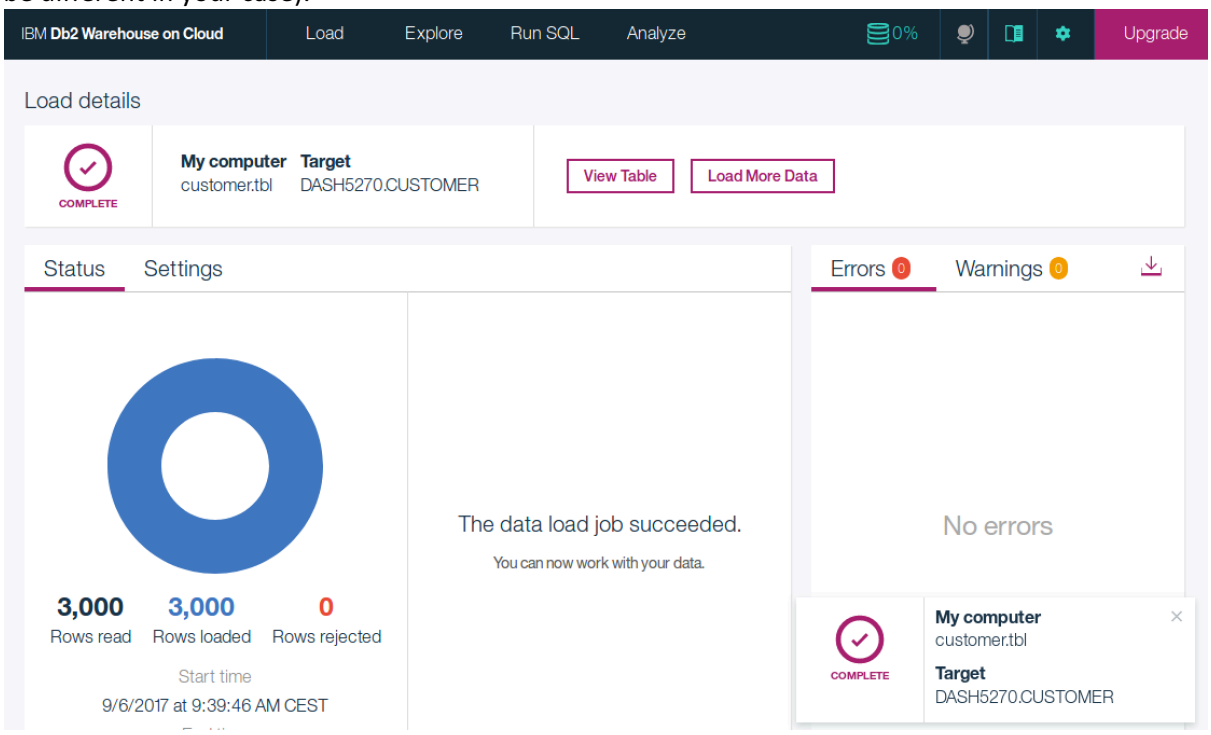
31. In the file browser select the file “customer.tbl” in the directory
`/home/user/Advanced-Analytics/SSB/loadfiles`

32. Go through the steps of the load wizard. Please notice that the first row contains no column names that there is another column separator than the default. Finally review your settings and select “Begin Load” in the last step.



33. Now the file will be uploaded to your Bluemix service. This may take some time for larger files. Let’s wait and drink a coffee until “Loading from my computer” is done.

34. When loading is done, check whether everything was successful. You should get a screen similar to the following one (Note the values correspond to 1GB of load data, therefore, the actually values display may be different in your case):

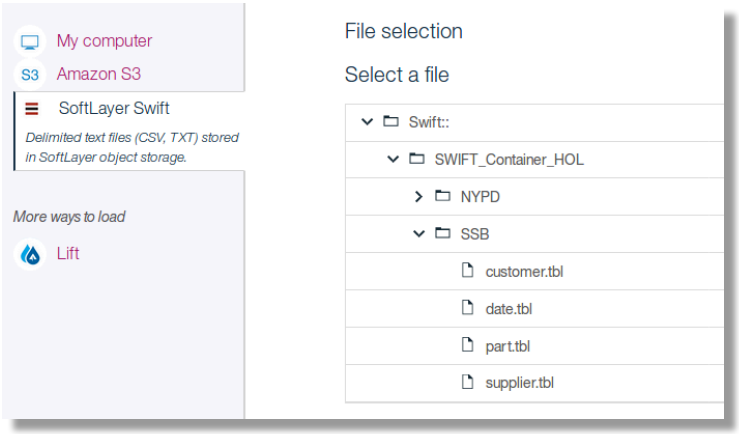


If you get errors or warnings please examine the problem or ask the instructor.

35. Load the next table PART from Softlayer Swift. Select the appropriate option on the left side and follow the wizard. Use the following credentials:

SoftLayer authentication endpoint	Frankfurt, DE
User name	DSWOS1438849-2:DSW1438849
API access key	44eaea8a64c14c427e89ebc8075f0c924b83bef42dc2809befe8f0cc0fd70717

You will find the data in the container SWIFT_Container_HOL:



36. Load all the other tables as well. At the end these tables should be filled with data:
- CUSTOMER
- *DATES* (not DATE)
- *PART*
- *SUPPLIER* and
- *LINEORDER*

Do this in the same way as in the steps before. When this is finished you can proceed to the next lab.

Lab 5: Executing Queries

In this part, you will learn how to run queries against the table which we just have created and loaded. For this, select “Run SQL”.

37. There are several buttons which help you executing queries. Let's first focus on the settings in the upper right corner. Click on it and you see, what options you can configure.

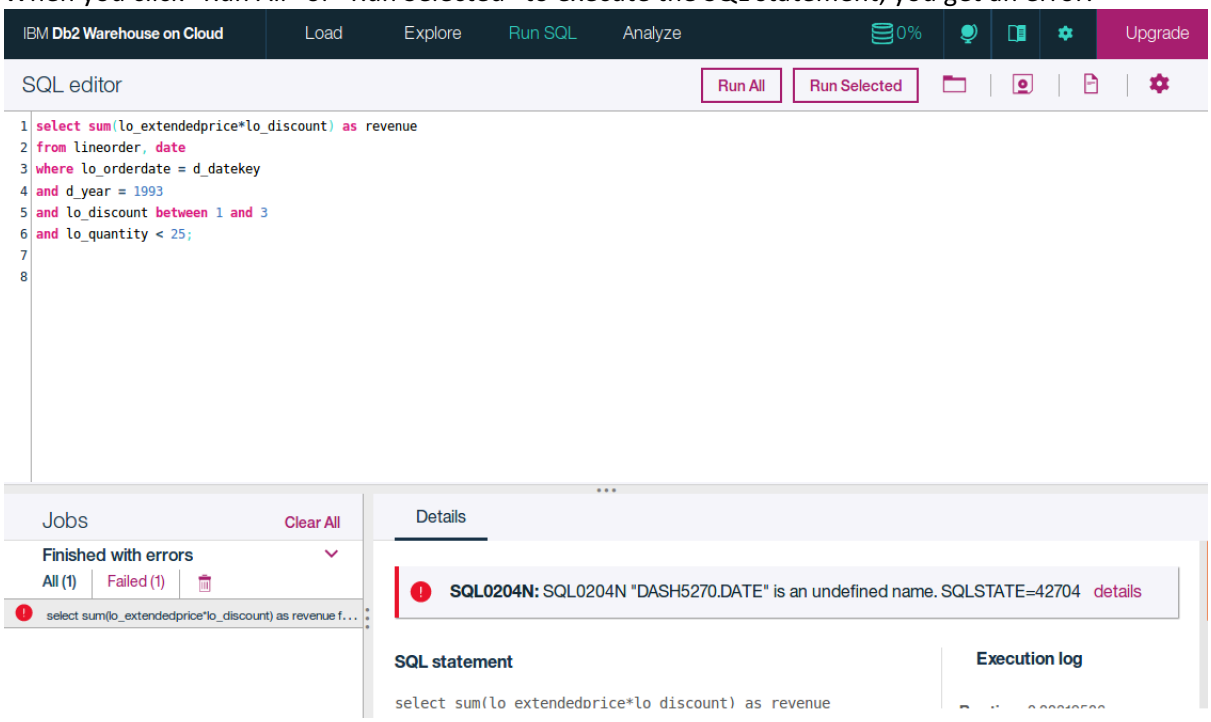
38. Now let's start running queries. The queries for the Star Schema Benchmark are in the directory `/home/user/Advanced-Analytics/SSB/queries`

    Let's look at query q1_1a.sql:

```
select sum(lo_extendedprice*lo_discount) as revenue
from lineorder, date
where lo_orderdate = d_datekey
and d_year = 1993
and lo_discount between 1 and 3
and lo_quantity < 25;
```

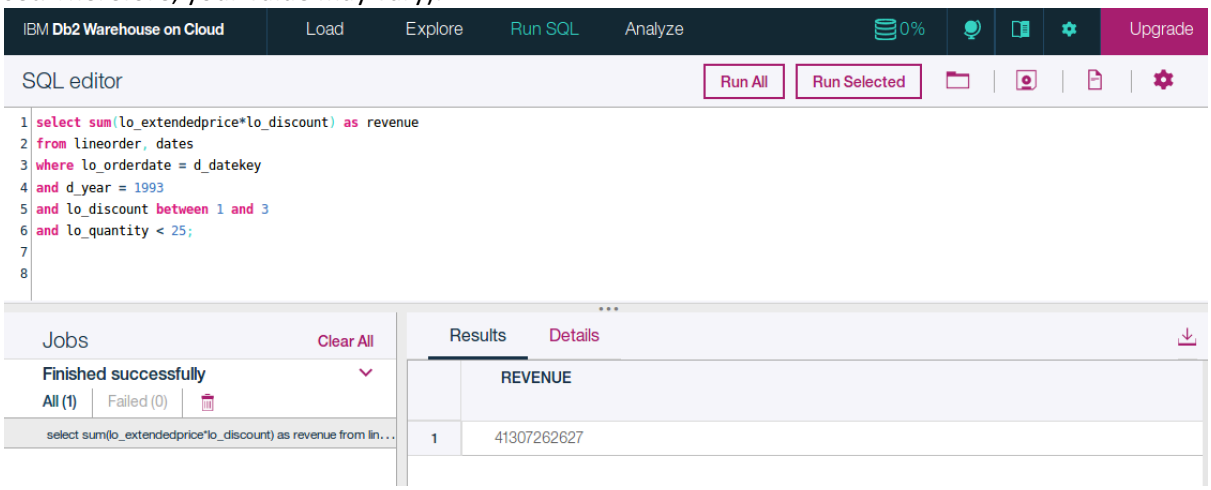
    Paste this query into the SQL.

39. When you click “Run All” or “Run Selected” to execute the SQL statement, you get an error:

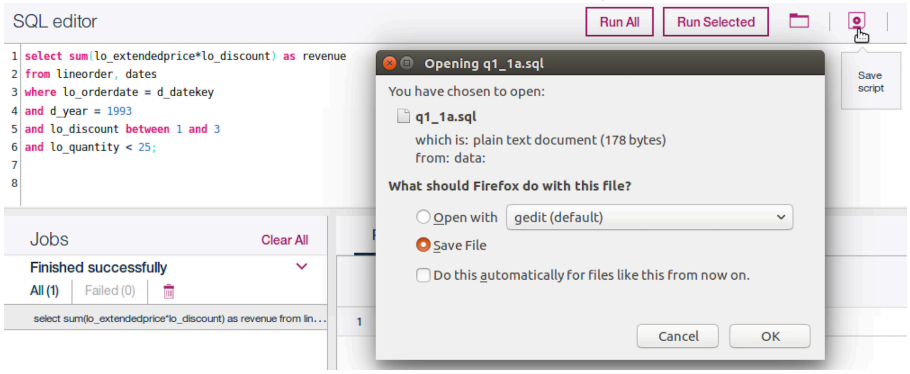


40. When you scroll down, you will notice the reason for our error. Our table is called “dates”, not “date”. Therefore, correct this and run the query again. This time the query is successfully executed.

41. When you scroll down, you will also see the result (please note that the value corresponds to a 1GB data set. Therefore, your value may vary):

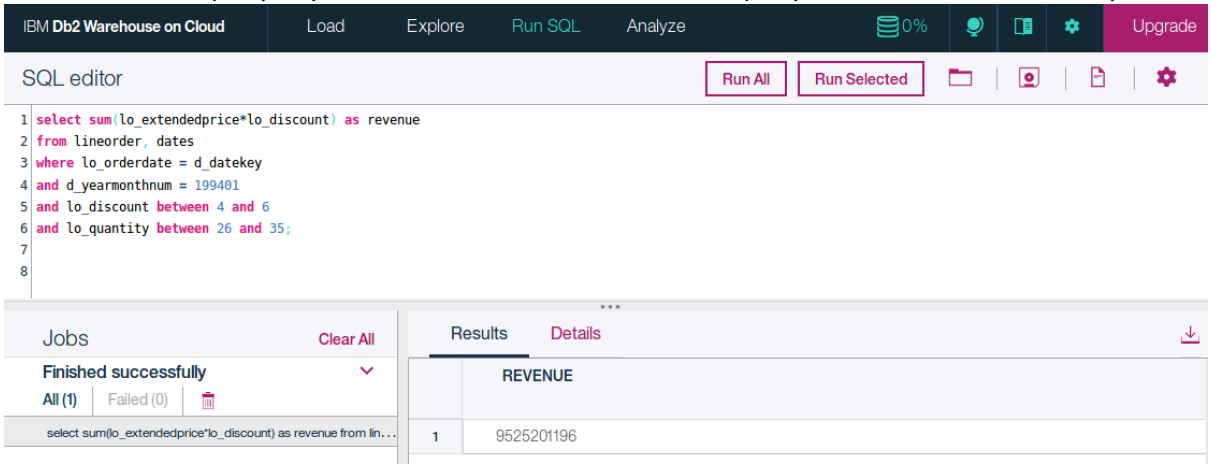


42. Now you can save the modified query by clicking the icon “Save script” and clicking “OK”. Please note that your workplace is in the cloud and “saving” means downloading the file to your local machine. The file will be stored in the download directory of your browser. In our case this is the directory ~/Downloads. Use a terminal window to check that the file was actually stored there:



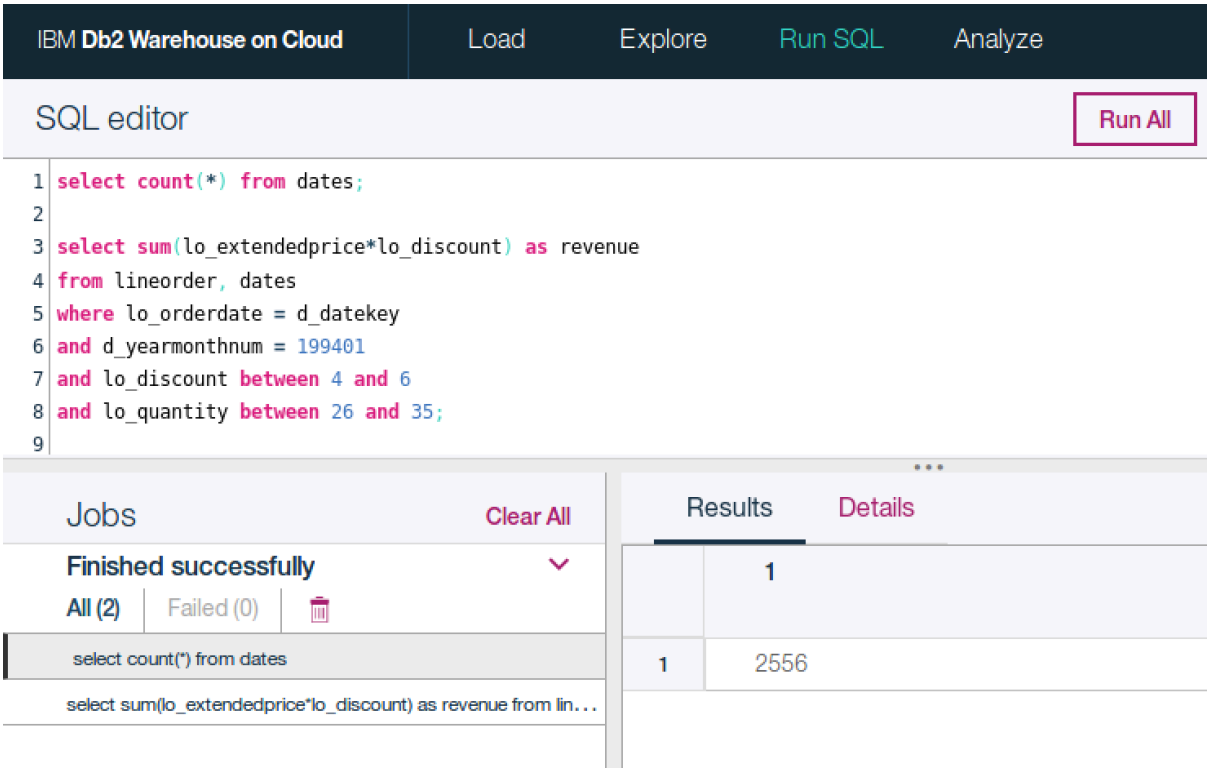
43. Let's run the next SQL statement by loading it from the local file system on our VMware. Let's use query q1_2.sql for this. Select “Open script”. Open a script from your local machine means to upload the file to the cloud.

44. Execute the SQL query. If you see this or a similar screen, the query was executed successfully:



45. Now run all the remaining queries in `/home/user/Advanced-Analytics/SSB/queries` with one of the two methods which you learned.

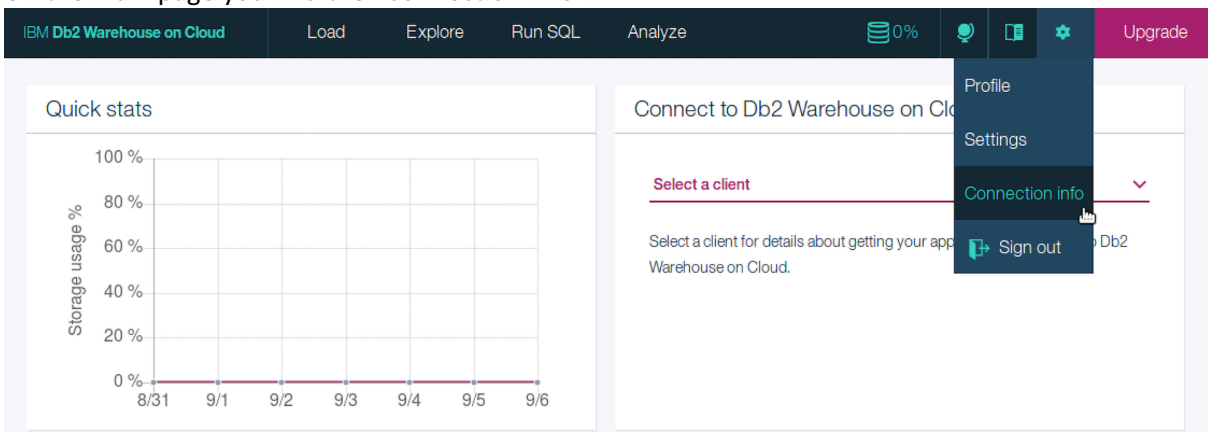
46. Paste more than one SQL statement into the SQL Editor and use the option “Run All”. Look at the lower area. You see more jobs on the left side and its corresponding results and details on the right side.



Lab 6: Remote Access to Db2

The goal of this lab is to show how Db2 cannot only be accessed via the web interface which we have seen in the previous labs but also with traditional remote Db2 clients. We will use *clpplus* as an example for such a client.

47. On the main page you find the “Connection Info”:



Follow the instructions on the left side and install the Db2 connection driver for Linux. The connection configuration data for your individual system is on the right side.
The result should look like this:

```
user@linux: ~/Downloads/dsdriver
Install Path:   /home/user/Downloads/dsdriver/.
-
- 'installDSDriver' script has started ...
-
- Unzipping and Copying each driver files to install path ...
  --> SUCCESS
- Generating db2profile & db2cshrc files ...
- Generating db2profile32 & db2cshrc32 files ...
  --> SUCCESS
- Performing post-installation clean up ...
  --> SUCCESS
-
- 'installDSDriver' completed successfully.
- Check /home/user/Downloads/dsdriver/./installDSDriver.log file for complete details.
-
user@linux:~/Downloads/dsdriver$
```

48. When you click on “Windows” as your platform you will get as well connection information for ODBS, .NET and CLPPlus.

49. Open a terminal window. Since the Data Server Driver is already installed you only have to set the environment correctly:

```
. ~/tmp/dsdriver/db2profile
db2cli writecfg add -database <database> -host <host> -port <port>
db2cli writecfg add -dsn <dsn name>  -database <database> -host <host> -port <port>
```

50. After having setup the connection in this way, change in your `/home/user/Advanced-Analytics terminal to /SSB/queries` and start clpplus:

```
user@linux:~$ clpplus -nw dash5270@dashdb
CLPPlus: Version 1.6
Copyright (c) 2009, 2011, IBM CORPORATION.  All rights reserved.

Enter password: ************

Database Connection Information :
---------------------------------
Hostname = dashdb-entry-fra-fra02-01.services.eu-de.bluemix.net
Database server = DB2/LINUXX8664  SQL11019
SQL authorization ID = dash5270
Local database alias = DASHDB
Port = 50001

SQL>
```

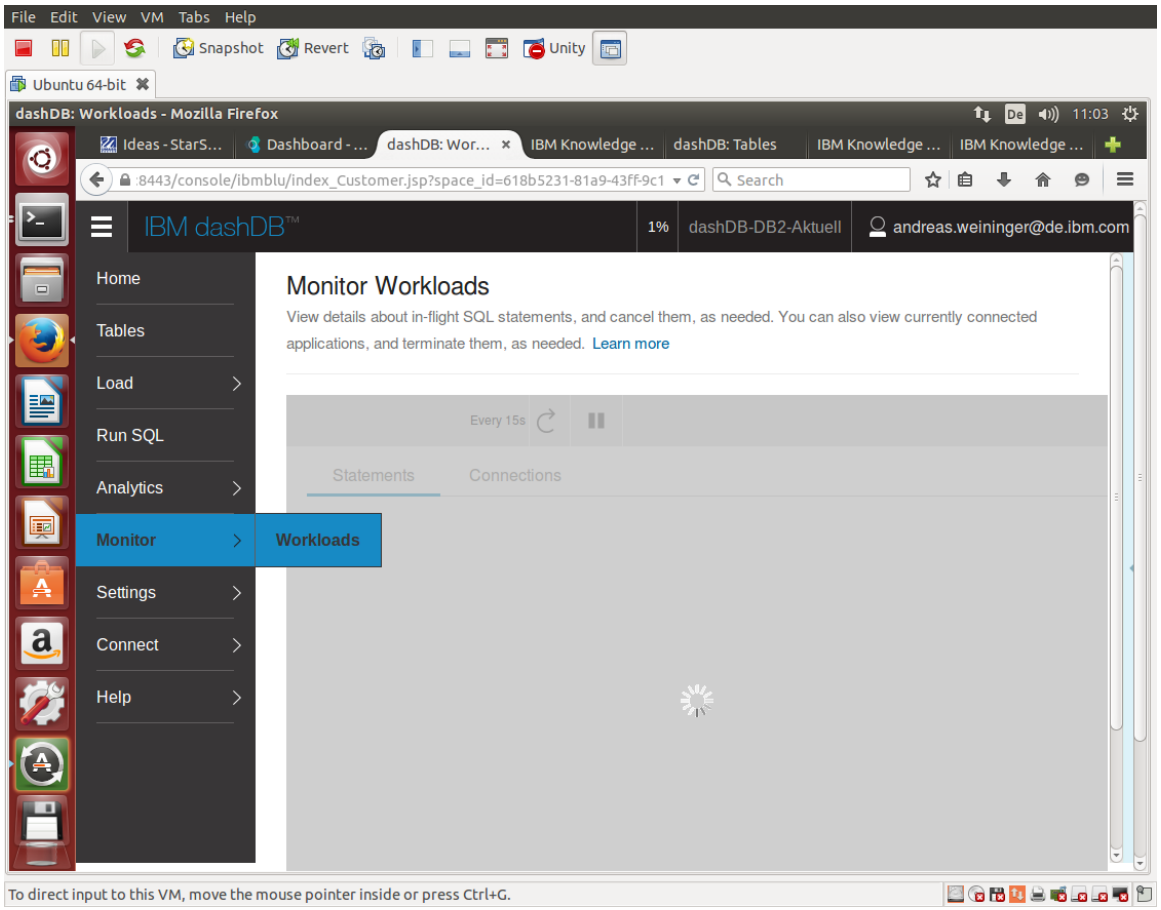
Now you are connected to your Db2 database and can run all the SSB queries.

Lab 7: Monitoring Db2

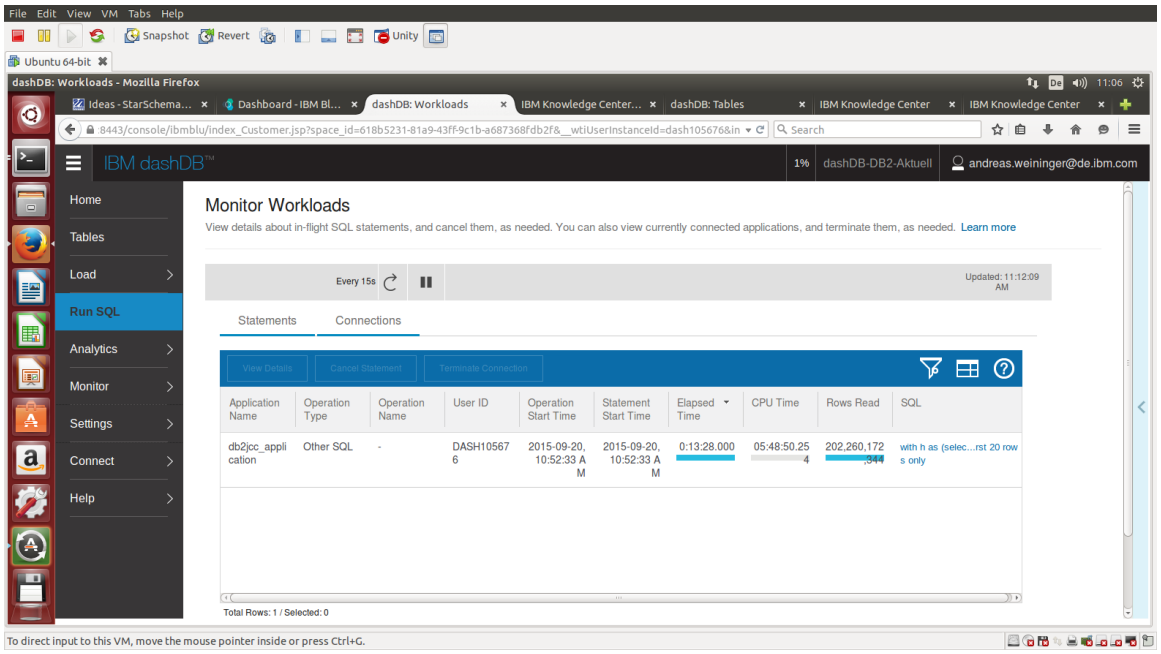
51. In this lab we will perform some monitoring of Db2. For this let us first look at some longer running query. There is a script ***long-running-query.sql*** in the directory `/home/user/Advanced-Analytics/SSB/queries`. Execute this query within the clpplus session set up in the previous lab:

```
SQL> with h as (select lo_partkey, d_year, d_weeknuminyear, sum(lo_revenue) revenue from lineorder,
dates where lo_orderdate = d_datekey group by lo_partkey, d_year, d_weeknuminyear) select
h0.lo_partkey, h1.d_year, h1. d_weeknuminyear, h0.revenue, h1.revenue, h1.revenue-h0.revenue from h
h0, h h1 where h0.lo_partkey = h1.lo_partkey and (h0.d_year = h1.d_year and h0.d_weeknuminyear + 1 =
h1.d_weeknuminyear) or (h0.d_year+1 = h1.d_year and h0.d_weeknuminyear = 53 and h1.d_weeknuminyear =
1) order by h1.revenue-h0.revenue asc fetch first 20 rows only;
```

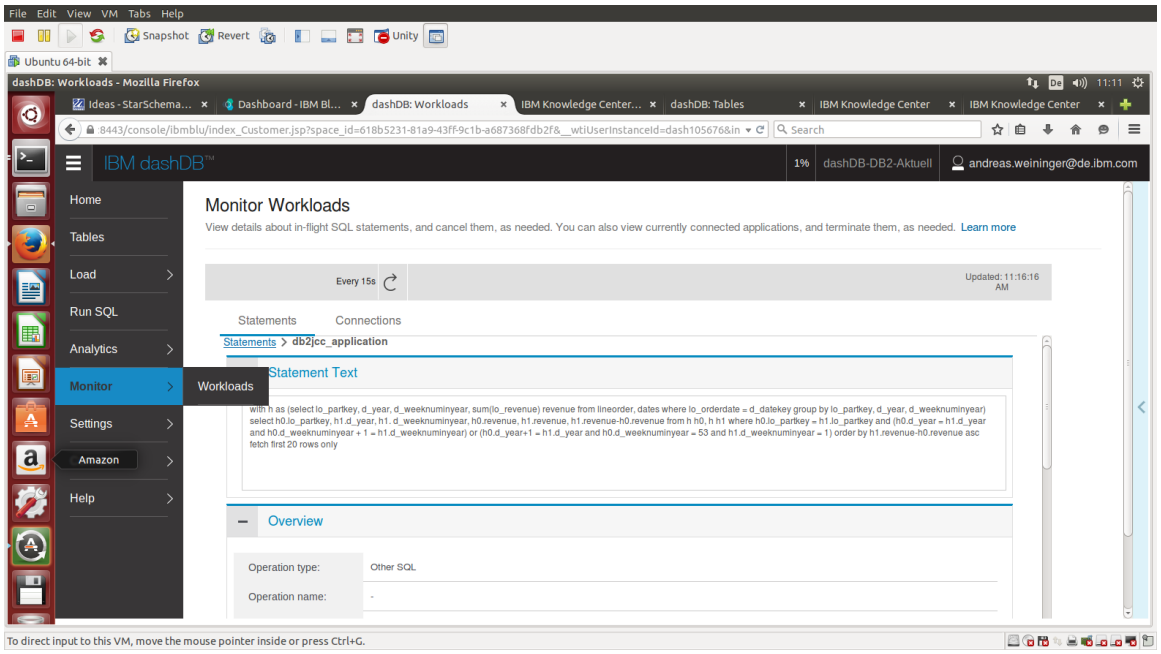
52. Now let's monitor this query. Select “Monitor” and “Workload” in the Db2 web interface:



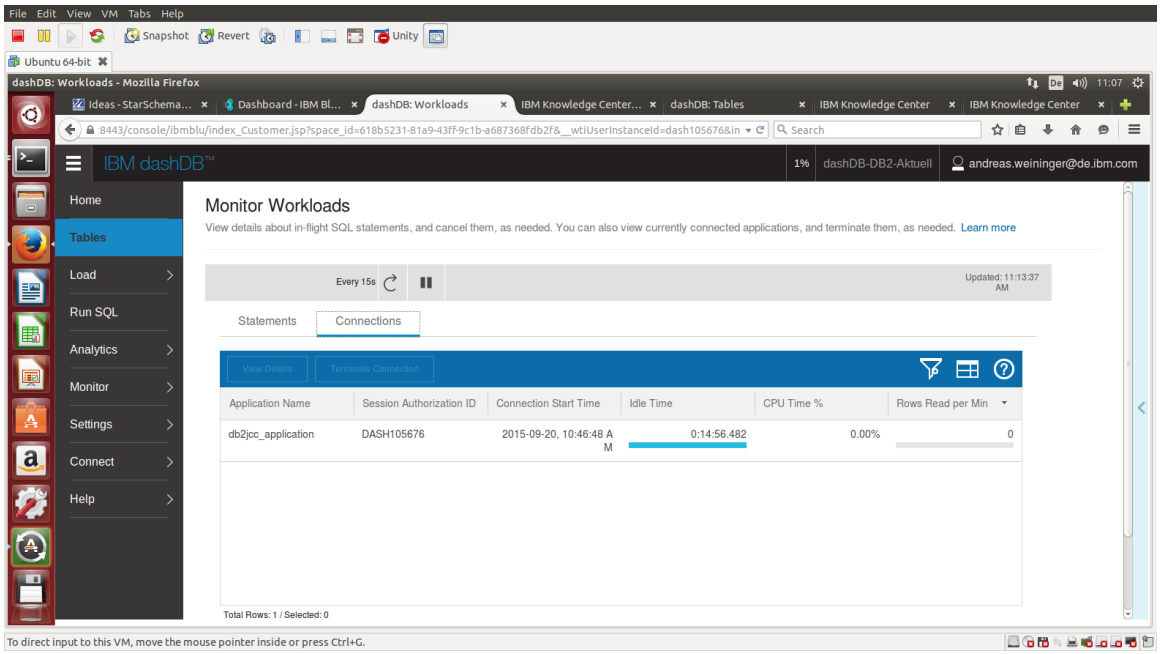
53. This will give you the following screen. If you click at the SQL statement in the last column you recognize that this is the statement which we just submitted:



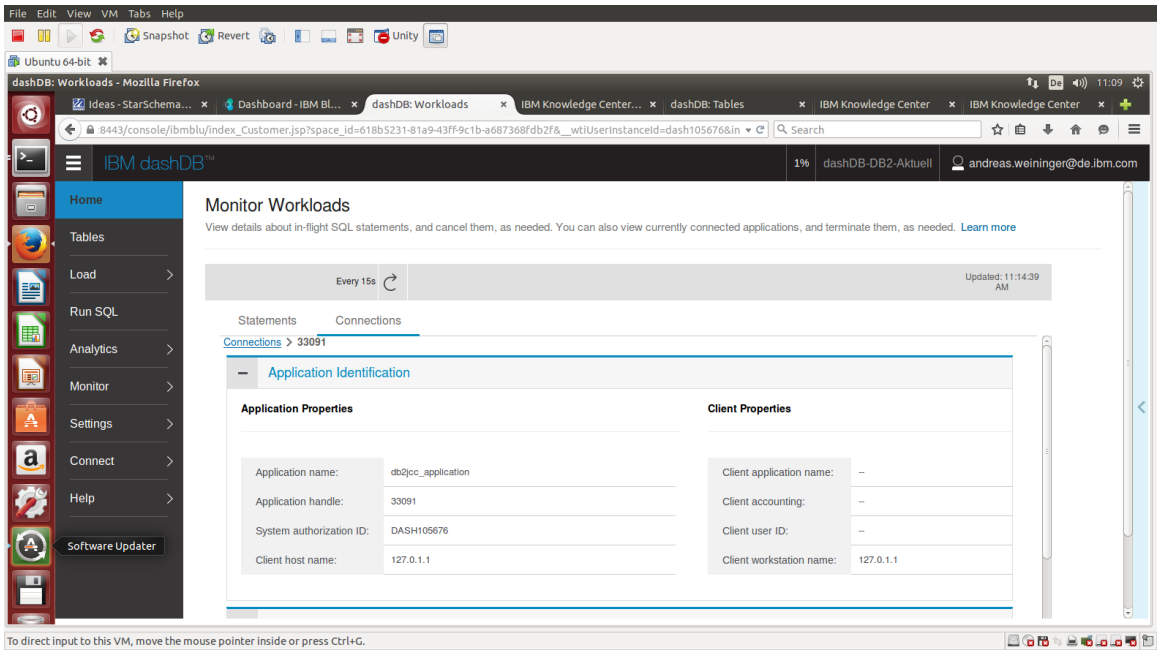
54. Click on the statement to select it and then click on “View details” to get more information on the statement:



55. You can also look at connections:



56. Again clicking on the connection and then on “View Details” gives you more information on this connection:



57. Since this statement is running so long, let us cancel it. For that select the statement and click on “Cancel Statement”.

Lab 8: Linear Regression in SQL

58. Db2 offers a rich set of in-database analytics. The first step will be implementing linear regression directly with SQL. The files necessary for this lab are located in the directory `/home/user/Advanced-Analytics/linear-regression`

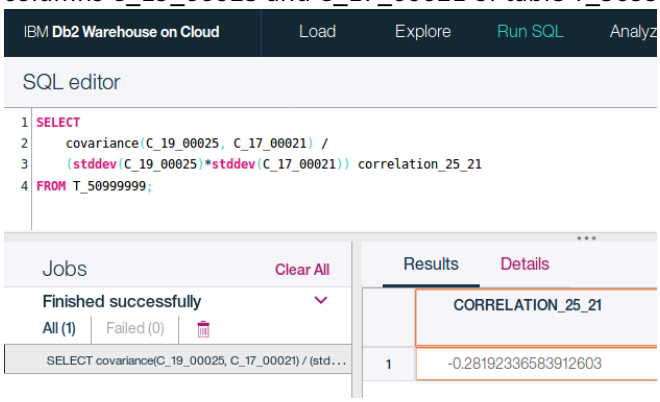
59. First the data have to be loaded in a table. For this, open the Db2 console. Next create a table T_50999999 like in “create-table-50999999.sql.

60. Load data from the file “lf_50999999.unl” into this table. Select “Load” and “My computer” for this.
Select the load file (located in `/home/user/Advanced-Analytics/linear-regression`) and specify that row one doesn't contain column names.
Scroll down and specify “|” as separator character.
The file contains *dates* and *times*, and specify “YYYY-MM-DD HH:MM” as format for columns which contain both dates and times. For that edit the field which contains “YYYY-MM-DD HH:MM:SS” and remove “:SS”.
Finally click on “Preview”:

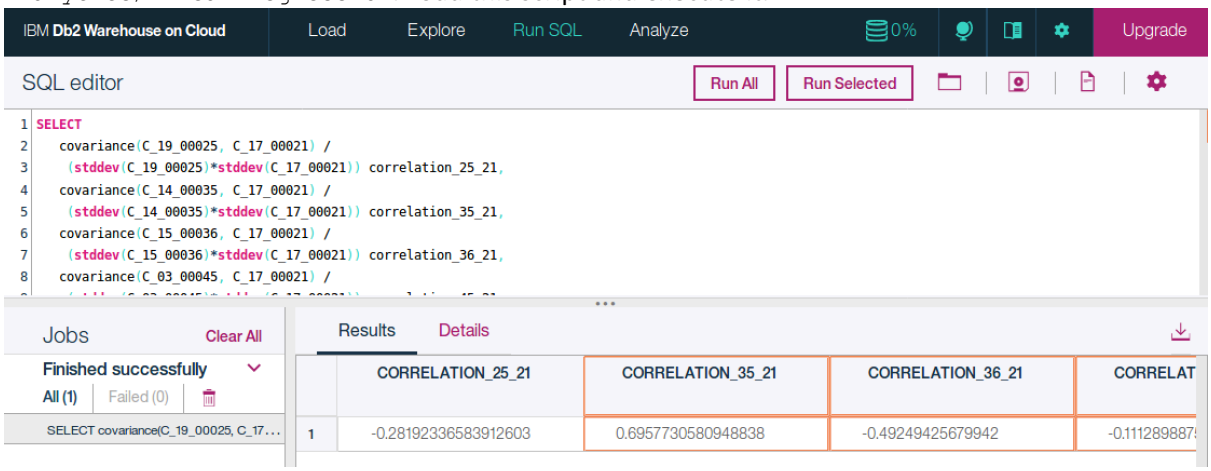
61. At next we want to find out whether any columns in our data set are correlated. We will try to do this with standard SQL. We will use the Pearson's correlation coefficient (https://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient) for determining whether two columns are correlated. The Pearson's correlation coefficient is defined in the following way:

$$\mathrm{cov}(X,Y)/(\sigma_X \sigma_Y)$$

62. Db2 contains built-in functions *covariance* and *stddev*. Write a SQL statement to check the correlation of columns C_19_00025 and C_17_00021 of table T_509999999.



63. There is already a script for checking the correlation of all columns in the directory `/home/user/Advanced-Analytics/linear-regression`. Load this script and execute it.



64. Which columns are correlated most?

65. In the previous step we have determined that the columns C_17_00021 and C_18_00052 are correlated most. There is a file `parameter_cd_query.txt` in the directory `/home/user/Advanced-Analytics/linear-regression`. In this file we can look up what the semantics of the different columns is. We find that C_17_00021 gives the air temperature in F, and C_18_00052 gives the relative humidity in percent. We want to use a linear model with C_17_00021 as the dependent variable and C_18_00052 as the independent variable. We call the parameters of the model w0 and w1. Therefore, our linear function is w0 + w1 * C_18_00052. For the minimum of the loss function we get the values for W0 and W1 with the following formulas:

```
W0 = avg(C_17_00021) - w1*avg(C_18_00052)
```

```
W1 = (avg(C_18_00052*C_17_00021)-avg(C_18_00052)*avg(C_17_00021))
    /(avg(C_18_00052*C_18_00052)-avg(C_18_00052)*avg(C_18_00052))
```

To be able to compute the parameters for our model on part of the data and test it on another part of the data, we partition our data in those rows which have C_DATETIME < 2014-07-21 10:00:00 which will be the data used for learning, and in those rows which have C_DATETIME >= 2014-07-21 10:00:00 which will be used for testing.

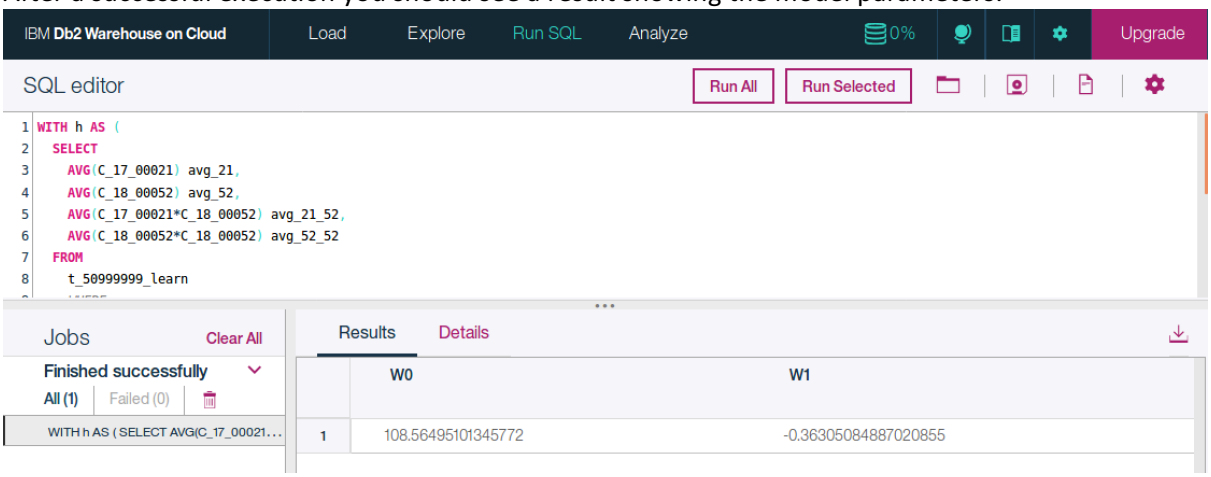
We will be doing this partitioning by defining two views *t_50999999_learn* and *t_50999999_test*. In the directory: `/home/user/Advanced-Analytics/linear-regression` there is already a script ***create-views.sql*** which creates these two views. Deploy both views.

Now write an SQL script which computes the parameters of this linear regression model. Use the formulas shown above. Execute the script and write down the parameter.

A solution for this script can be found in the file get-model-parameters.sql:

```
WITH h AS (
  SELECT
    AVG(C_17_00021) avg_21,
    AVG(C_18_00052) avg_52,
    AVG(C_17_00021*C_18_00052) avg_21_52,
    AVG(C_18_00052*C_18_00052) avg_52_52
  FROM
    t_50999999_learn
)
SELECT
  avg_21 - ((avg_21_52 - avg_52 * avg_21) /
  (avg_52_52 - avg_52 * avg_52))*avg_52 AS w0,
  (avg_21_52 - avg_52 * avg_21) / (avg_52_52 - avg_52 * avg_52) AS w1
FROM
  h;
```

After a successful execution you should see a result showing the model parameters:

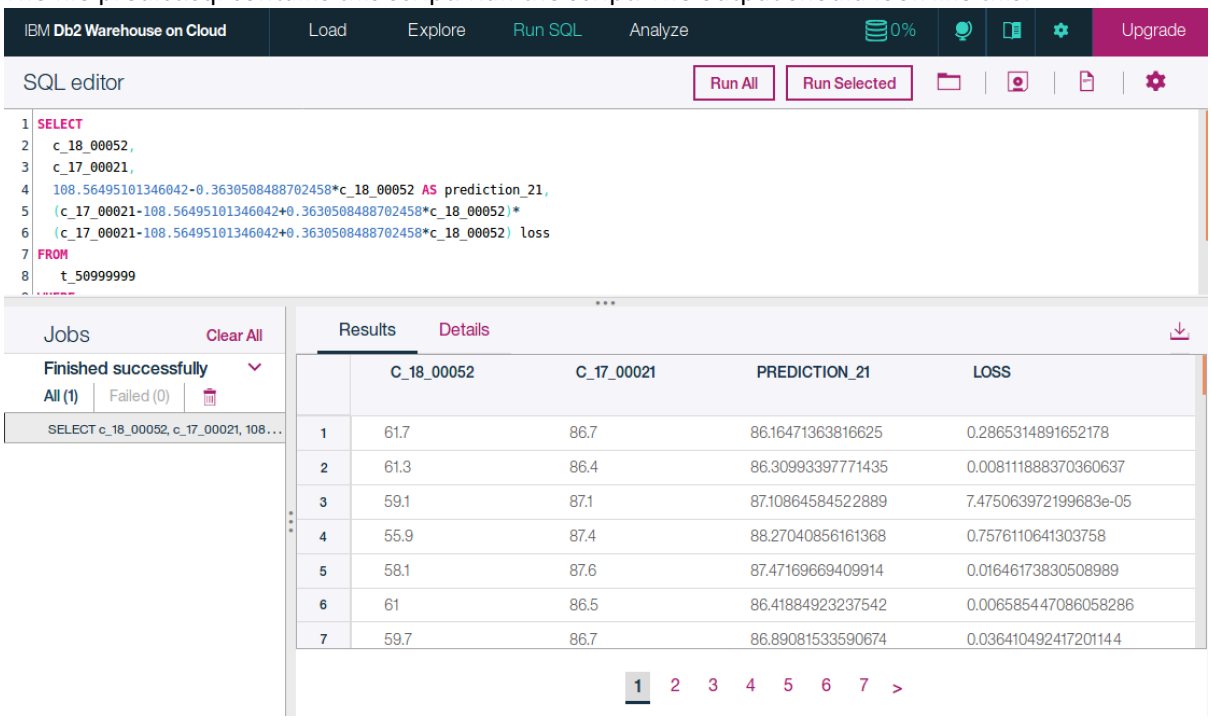


66. Now we can use these parameters to predict the air temperature for the second part of the data set. Write an SQL statement for this and execute it.

    The following SQL script determines not only the predicted values but also computes the loss to have a measure for the quality of the prediction:

```
SELECT
  c_18_00052,
  c_17_00021,
  108.56495101346-0.3630508488702458*c_18_00052 AS prediction_21,
  (c_17_00021-108.56495101346042+0.3630508488702458*c_18_00052)*
  (c_17_00021-108.56495101346042+0.3630508488702458*c_18_00052) loss
FROM
   t_50999999_test;
```

67. The file predict.sql contains this script. Run the script. The output should look like this:

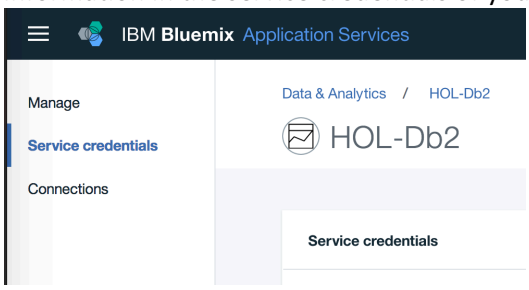


	C_18_00052	C_17_00021	PREDICTION_21	LOSS
1	61.7	86.7	86.16471363816625	0.2865314891652178
2	61.3	86.4	86.30993397771435	0.008111888370360637
3	59.1	87.1	87.10864584522889	7.475063972199683e-05
4	55.9	87.4	88.27040856161368	0.7576110641303758
5	58.1	87.6	87.47169669409914	0.01646173830508989
6	61	86.5	86.41884923237542	0.006585447086058286
7	59.7	86.7	86.89081533590674	0.036410492417201144

1 2 3 4 5 6 7 >

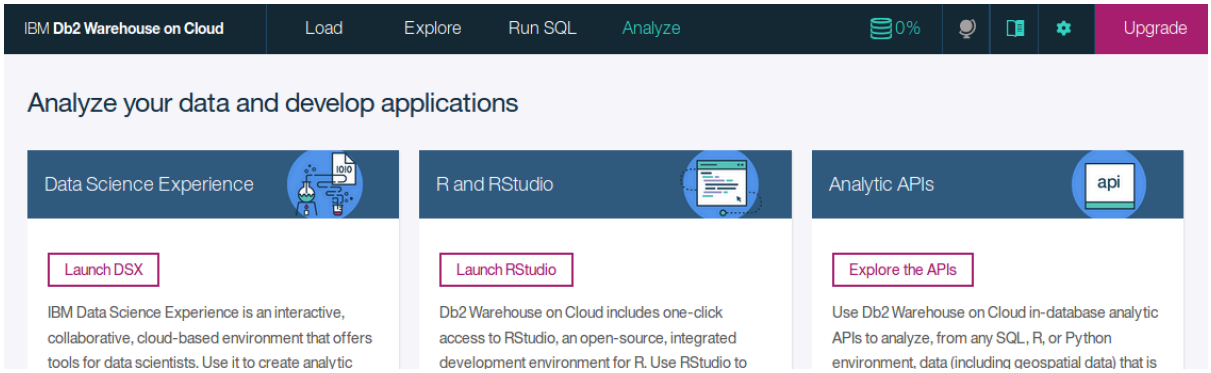
Lab 9: Linear Regression with R

68. In this lab we want to use R and in-database analytics for linear regression. We will use RStudio for submitting the R code. For starting RStudio we need the User ID and the password for Db2. You get this information in the service credentials of your Bluemix service with the name “HOL-Db2”



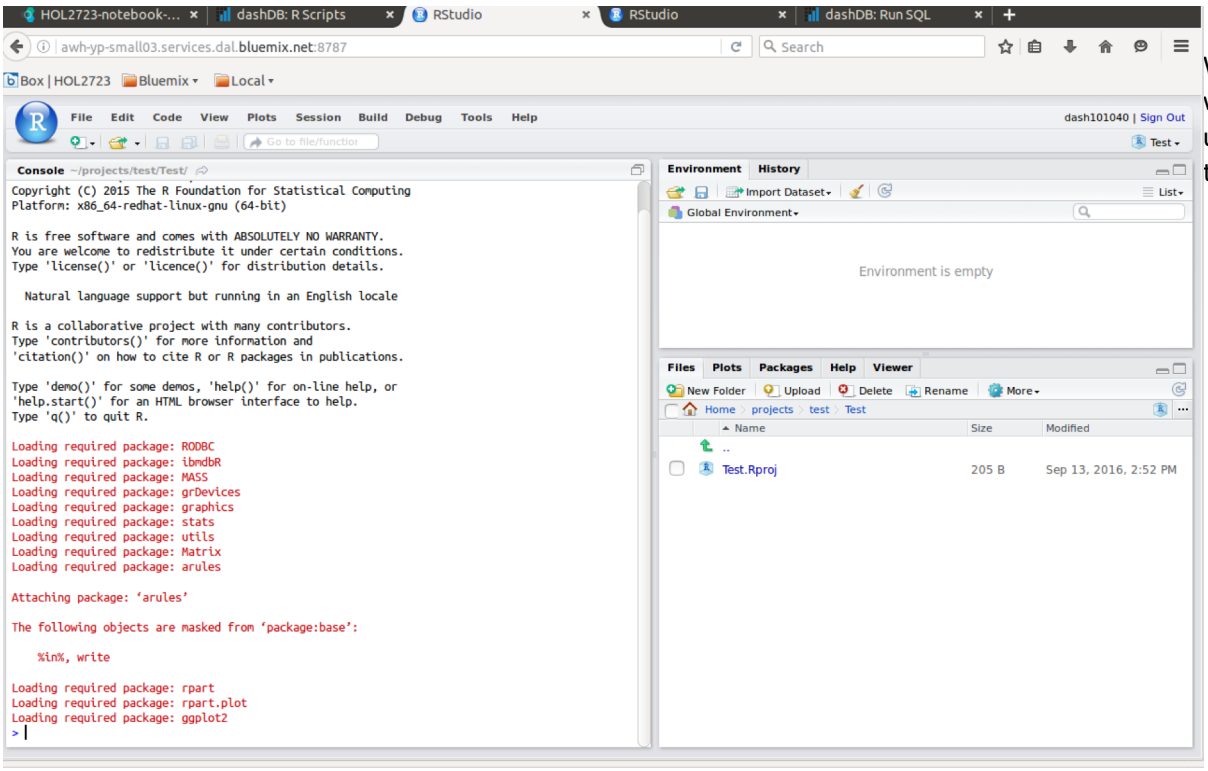
Copy or write this information down for later use:

69. Next we will start RStudio. Change to the “Analyze” options. When you get the following screen, click on “Launch RStudio”:



70. Now you see the GUI of RStudio. To warm-up with RStudio please press CTRL-L to clear the console area.

71. Now you will see the console window on the left side of your screen:



We will use the console window for typing commands. We will show the commands preceded by the prompt "> ", but you don't have to type the prompt.

72. First let's open a connection to our Db2 database (which is called bludb):

```
> con=idaConnect('bludb')
```

The documentation for the in-database analytics functions can be found here: https://cran.r-project.org/web/packages/ibmdbR/ibmdbR.pdf

73. Next we initialize the in-database analytics functions:

```
> idaInit(con)
```

74. Next we will try some simple functions of the in-database analytics package. All these functions start with the prefix ida. The function will show us the available tables. Type:

```
> idaShowTables()
```

75. This produces a list of tables available in our database. The list will look something like this (the output will depend on the tables created by other exercises):

```
       Schema                              Name      Owner Type
1  DASH101040                         ACCIDENTS DASH101040    T
2  DASH101040                                D1 DASH101040    T
3  DASH101040                                D2 DASH101040    T
4  DASH101040                                F1 DASH101040    T
5  DASH101040                         NYPD_COLL DASH101040    T
6  DASH101040                    NYPD_COLL_KM DASH101040    V
7  DASH101040                  NYPD_KM_CLUSTERS DASH101040    T
8  DASH101040                   NYPD_KM_COLUMNS DASH101040    T
9  DASH101040          NYPD_KM_COLUMN_STATISTICS DASH101040    T
10 DASH101040                    NYPD_KM_MODEL DASH101040    T
```

```
11 DASH101040                          NYPD_KM_OUT DASH101040    T
12 DASH101040       NYPD_MOTOR_VEHICLE_COLLISIONS DASH101040    T
13 DASH101040 NYPD_MOTOR_VEHICLE_COLLISIONS_STAGE DASH101040    T
14 DASH101040                    T_472727101175000 DASH101040    T
15 DASH101040                          T_50999999 DASH101040    T
```

76. A data frame is the object used by most operations. It is a pointer to the actual table. Let us create a data frame for our table T_50999999 and assign it to a variable named df:

```
> df=ida.data.frame("T_50999999")
```

77. Now we can apply functions to data frames. Nrow gives the number of rows in the table:

```
> nrow(df)
[1] 732
```

78. The output shows that there are 732 rows in this table. It also possible to define data frames for views. Let us do this for our views T_50999999_LEARN and T_50999999_TEST. Let us also check the number of rows in these views:

```
> df_learn=ida.data.frame("T_50999999_LEARN")
> df_test=ida.data.frame("T_50999999_TEST")
> nrow(df_learn)
[1] 424

> nrow(df_test)
[1] 308
```

We see that there are 424 and 308 rows in these views.

79. Next let's try to visualize the relationship between columns C_18_00052 and C_17_00021 in T_50999999_LEARN.

We load the R library ggplot2 first:

```
> library(ggplot2)
```

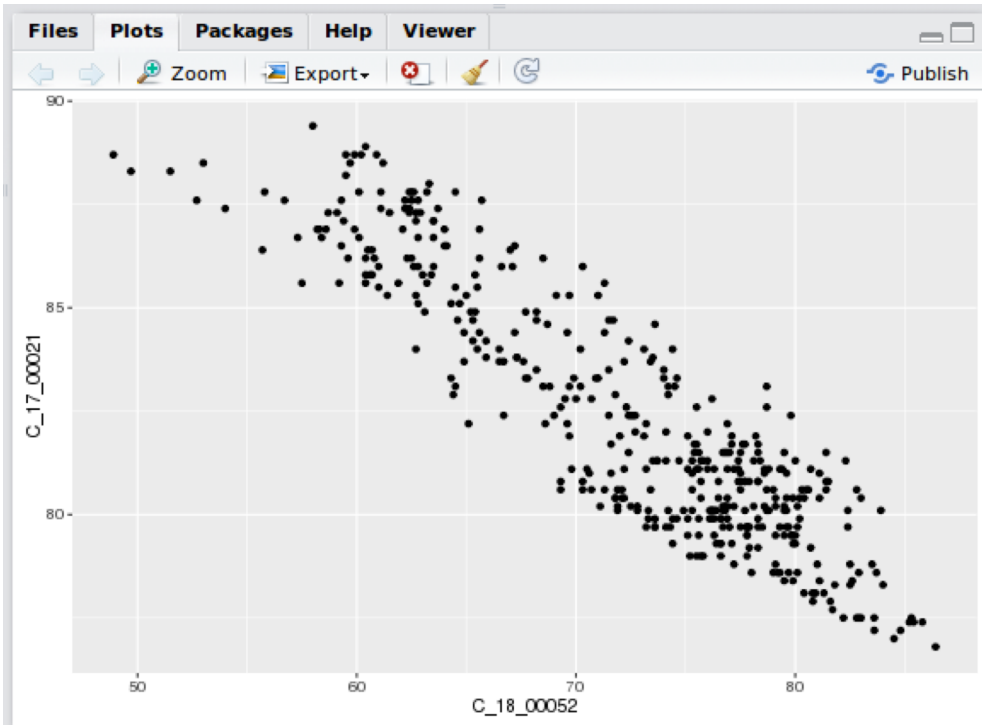
80. We have to convert the ida data frame to a regular R data frame first, since the plotting functions don't work on ida data frames directly. We do this with the following command and assign the data frame to a variable l:

```
> l=as.data.frame(df_learn)
```

81. The plotting is performed in this way:

```
> p = ggplot(l,aes(x=C_18_00052,y=C_17_00021))
> p+geom_point()
```

82. We get the following plot in RStudio:



83. Now let us also compute the linear model with R in the Db2 database. The ida function for this is idaLm. We specifiy the column and the ida data frame:

```
> lm1=idaLm(C_17_00021~.,df_learn)
```

84. Now we can information about this model:

```
> print(lm1)

Coefficients:
              Estimate   Std.Error   t.value Pr(>|t|)
C_18_00052  -0.3630508 0.008303625 -43.72197        0 ***
Intercept  108.5649510 0.603139360 179.99978        0 ***

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error:  1.298999  on  422  degrees of freedom

Log-likelihood:  -711.5433
AIC:  1429.087
BIC:  1441.236
```

85. We see that we get the same coefficients as in the previous lab when we manually coded the linear regression in SQL.

    A main advantage of using ida data frames instead of R data frames is that calculations are directly done in Db2 using the resources available to Db2.

Lab 10: Linear Regression with Python

86. In this lab, we will not compute the linear regression in Db2, but will extract the data to the client and process the data there. In addition, we will not use a GUI, but instead use the command line. As programming language, we will use Python.

    When logged in as user user, open a terminal window. In the terminal window change the directory to `/home/user/Advanced-Analytics/linear-regression`. Then type in python ("$" is the prompt which you don't have to type; blue is the output of the system, black the text which you have to type yourself, <red> is information which you have to type yourself, but replace with the concrete information for your system) to get an interactive Python shell:

```
$ python

Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

87. For accessing the Db2 database we use the SQLAlchemy toolkit (http://www.sqlalchemy.org). First, we import the create_engine function which is used for creating the database connection:

```
>>> from sqlalchemy import create_engine
```

88. Next we use the create_engine function to initialize the information for the database connection. We get the necessary information for the parameters from section 68.

```
>>> e = create_engine("db2+ibm_db://<userid>:<password>@<hostname>:<portnumber>/<databasename>")
```

89. Now we connect to the Db2 database:

```
>>> c = e.connect()
```

90. We will use the pandas library (http://pandas.pydata.org/) which provides data frames for Python. Therefore, we import the DataFrame function from this library:

```
>>> from pandas import DataFrame
```

91. On the connect variable c which we created above, we can use the execute method to execute SQL statement against our database. The result of type ResultProxy will be assigned to a variable r:

```
>>> r = c.execute("SELECT * FROM T_50999999_LEARN")
```

92. We use the fetchall method on r to initialize a pandas data frame and assign this data frame to a variable df:

```
>>> df =DataFrame(r.fetchall())
```

93. Next let us look at some methods on getting information about the content of the data frame. With the head method we can look at the first few rows. The first line of the output just numbers columns, while all the other output lines are numbered in the first column:

```
>>> df.head()
      0     1
0  65.4  85.8
1  66.6  86.0
2  69.7  85.3
3  71.7  84.7
```

```
4   71.5   84.7
```

94. With values.shape, we can get information on the number of rows and columns in the data frame:

```
>>> df.values.shape

(424, 2)
```

95. We can project individual columns with iloc:

```
>>> df.iloc[:,1]

0      85.8
1      86.0
2      85.3
3      84.7
4      84.7
5      84.6
6      84.2
7      84.0
8      84.0
9      83.8
```

...

96. Now we want to compute a linear regression model on that. For this we will use the scikit-learn machine learning library (scikit-learn.org/). First we import the linear model from the library, initialize it and pass it our values from the data frame:

```
>>> import sklearn.linear_model as sl
>>> linReg = sl.LinearRegression(fit_intercept=True)
>>> linReg.fit(y=df.iloc[:,1],X=df.iloc[:,0].values.reshape(-1,1))
      LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

97. We can print the parameter of our model with intercept_ and coef_. Obviously, we are getting the same values as in the two previous labs:

```
>>> print(linReg.intercept_)
108.564951013
>>> print(linReg.coef_)
[-0.36305085]
```

Lab 11: Machine Learning with k-means in Db2

98. CREATE TABLE

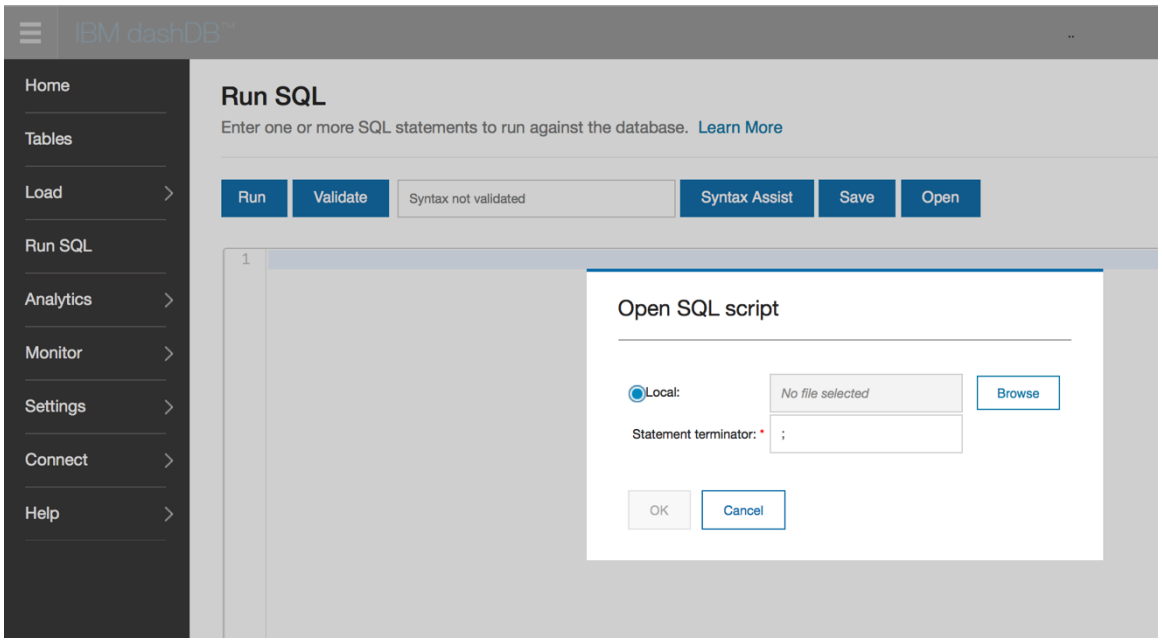
This lab is based on open data from NYPD about car collisions. In order to explore the data set a table has to be defined. It should have the following structure:

```
CREATE TABLE NYPD_COLL (
DATE                        DATE NOT NULL,
TIME                        TIME NOT NULL,
BOROUGH                     VARCHAR(65),
ZIP_CODE                    VARCHAR(10),
LATITUDE                    DECIMAL(31,10),
LONGITUDE                   DECIMAL(31,10),
LOCATION                    VARCHAR(65),
ON_STREET_NAME              VARCHAR(65),
CROSS_STREET_NAME           VARCHAR(65),
OFF_STREET_NAME             VARCHAR(65),
PERSONS_INJURED             INT,
PERSONS_KILLED              INT,
PEDESTRIANS_INJURED         INT,
PEDESTRIANS_KILLED          INT,
CYCLIST_INJURED             INT,
CYCLIST_KILLED              INT,
MOTORIST_INJURED            INT,
MOTORIST_KILLED             INT,
CONTRIBUTING_FACTOR_1       VARCHAR(65),
CONTRIBUTING_FACTOR_2       VARCHAR(65),
CONTRIBUTING_FACTOR_3       VARCHAR(65),
CONTRIBUTING_FACTOR_4       VARCHAR(65),
CONTRIBUTING_FACTOR_5       VARCHAR(65),
UNIQUE_KEY                  INT NOT NULL,
VEHICLE_TYPE_CODE_1         VARCHAR(65),
VEHICLE_TYPE_CODE_2         VARCHAR(65),
VEHICLE_TYPE_CODE_3         VARCHAR(65),
VEHICLE_TYPE_CODE_4         VARCHAR(65),
VEHICLE_TYPE_CODE_5         VARCHAR(65)
);
```

You can find the script to create this table in the path `/home/user/Advanced-Analytics/create_nypd_table.ddl`

The schema for this table is identical to the User ID of your Db2 database and cannot be changed in the free tier of the Db2 service.

99. Navigate to the Run SQL page in Db2 and load the contents of the file from your desktop. Execute the create table statement by pressing the Run button.

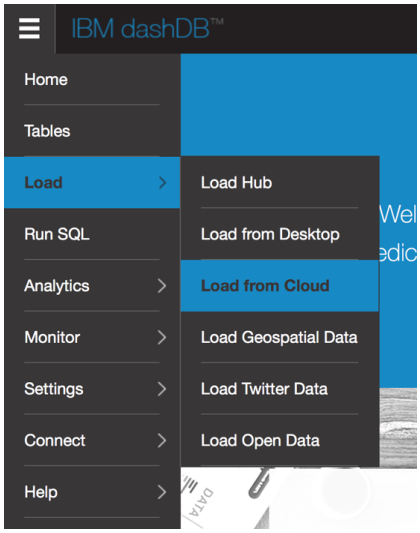


100. LOAD DATA
After the table is created you can load the data from your desktop or from Softlayer cloud.

[see below at step 106 if you want to load from desktop instead]

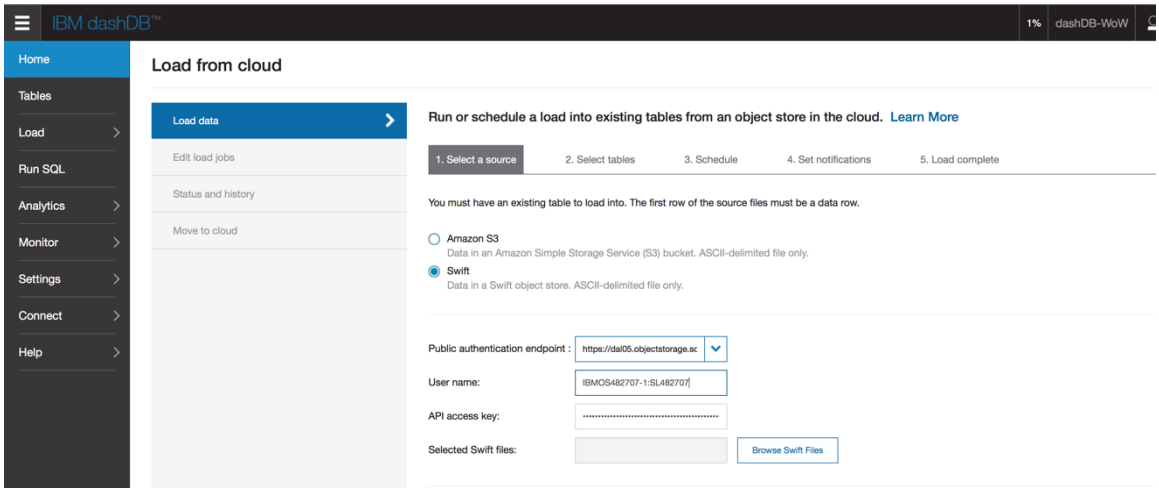
LOADING FROM CLOUD

Navigate to Load from Cloud:

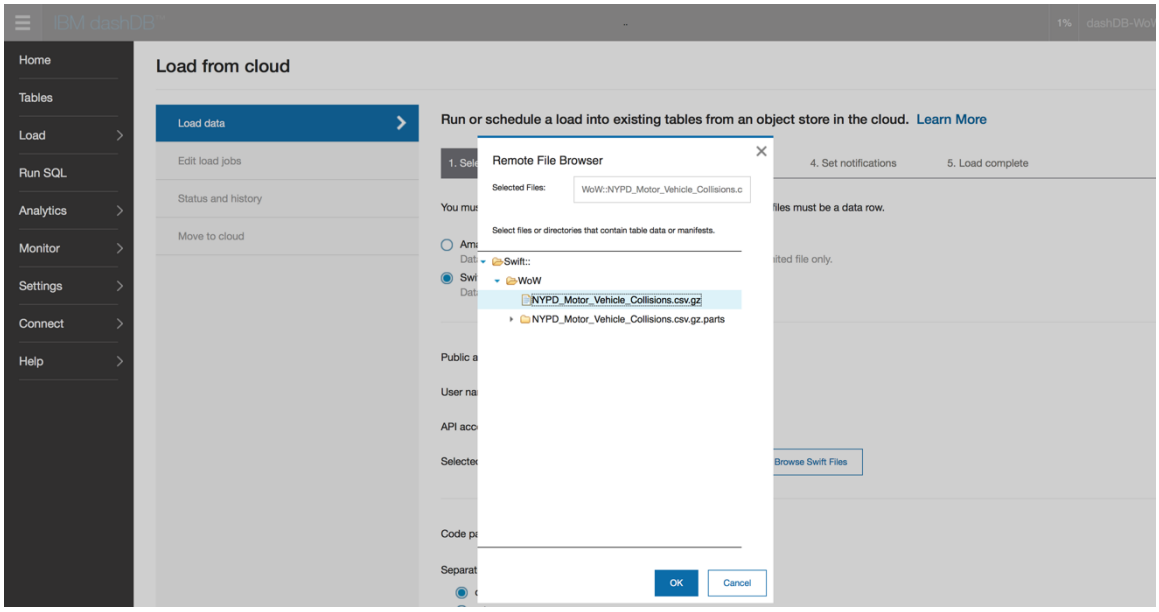


Select Swift as the data source type, dal05 as the datacenter, and use the following credentials to access the data:

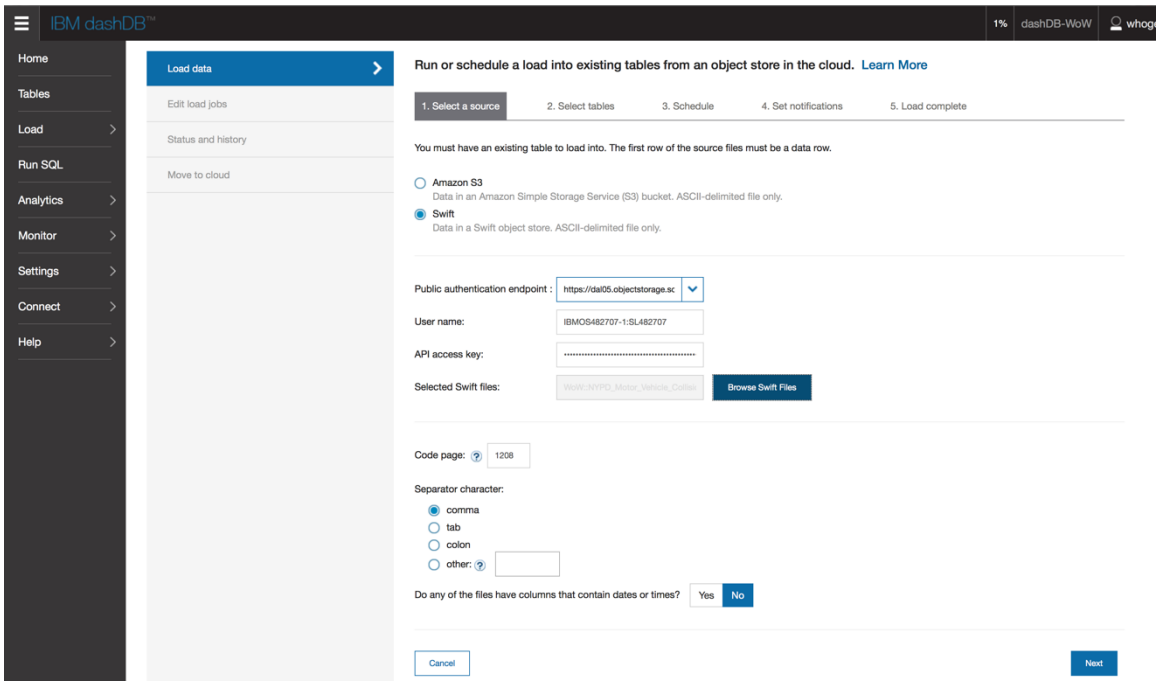
```
User name: IBMOS482707-1:SL482707
Password: 20851f8eaee8fbc877254468d97671929e0bbc870b00a7a3e6920b53938f210a
```



101. Press Browse Swift Files and select the compressed csv file NYPD_Motor_Vehicle_Collisions.csv.gz in path WoW.

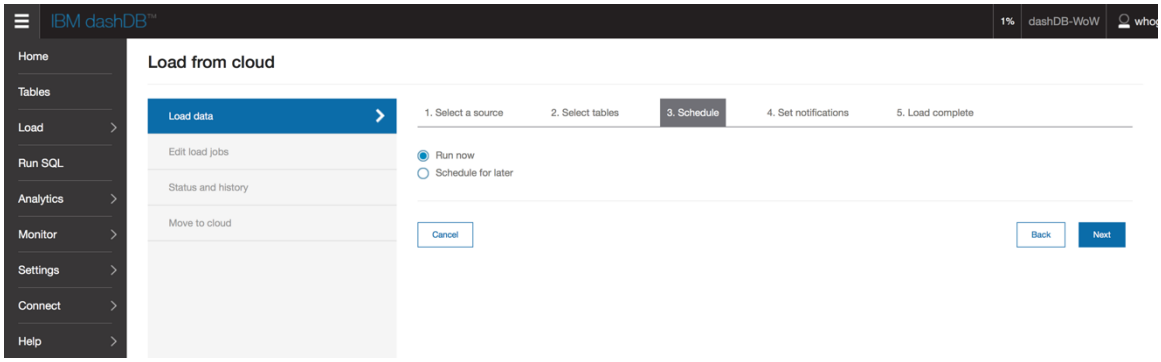


102. You can leave the other parameters on default values and press next to move on:



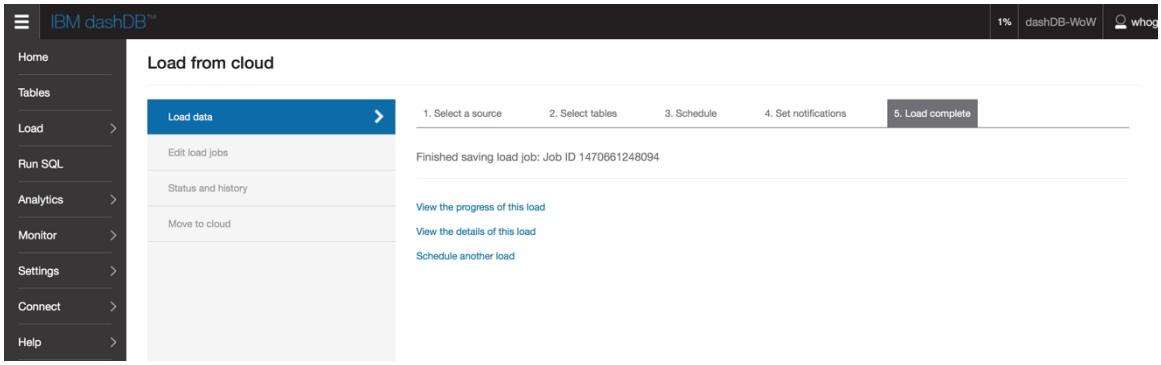
103. Select your Db2 user name as schema and the table name NYPD_COLL. Select Replace any data and press Next to move on.

Select Run now and press Next to move on:

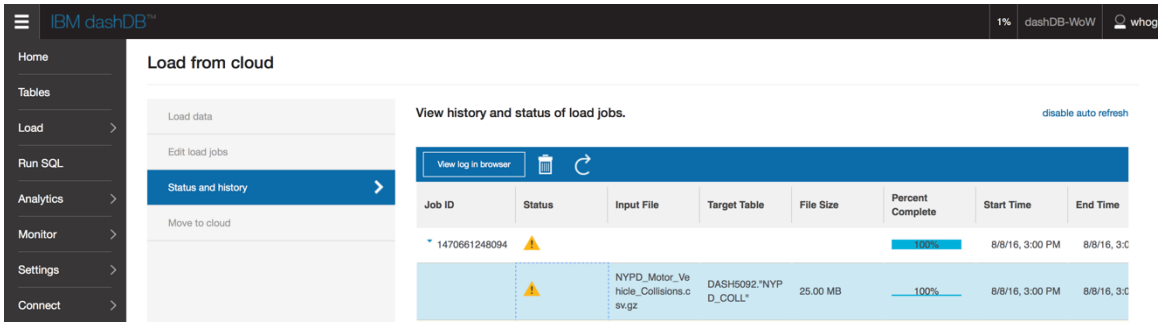


104. You can leave all the settings on default values and press Finish to start the loading process.

Press View the progress of this load:



105. When the load job has finished the status should say 100% complete. The job completes with warnings. Click on the line with the input file and press View log in browser to see the log file for this load:

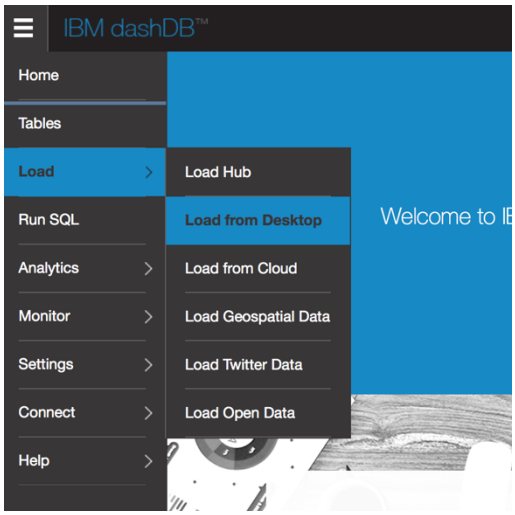


Not all data could be loaded but we can ignore any errors and move on to the next point.

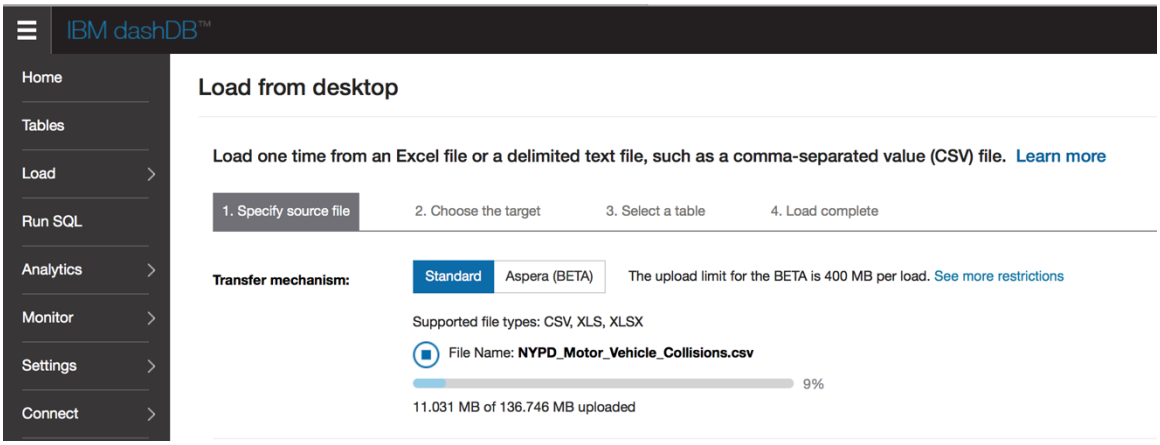
106. LOADING FROM DESKTOP

If you prefer loading from desktop, use the csv file in path /home/user/Advanced-Analytics/NYPD/NYPD_Motor_Vehicle_Collisions.csv

Navigate to Load from Desktop



107. Choose the csv file from your Desktop and wait until it is uploaded to the system.



You don't have to change the default settings for File characteristics. When the file is uploaded, just move on to Preview. You can preview the loaded records on the next page. Press Next to move on.

Select Load into existing table and press Next again.

Select your Db2 user name as Schema and the table name NYPD_COLL. Select Replace any data and press Finish to start the loading process.

Not all data could be loaded but we can ignore any errors and move on to the next point.

108. Db2 has implemented some machine learning algorithms that we will test in this lab. One of these ML algorithm is k-means. You can find a description of k-means in Wikipedia (https://en.wikipedia.org/wiki/K-means_clustering).

The list of machine learning algorithms is described in the Db2 documentation: https://www.ibm.com/support/knowledgecenter/en/SS6NHC/com.ibm.swg.im.dashdb.analytics.doc/doc/r_analytic_stored_procedures.html

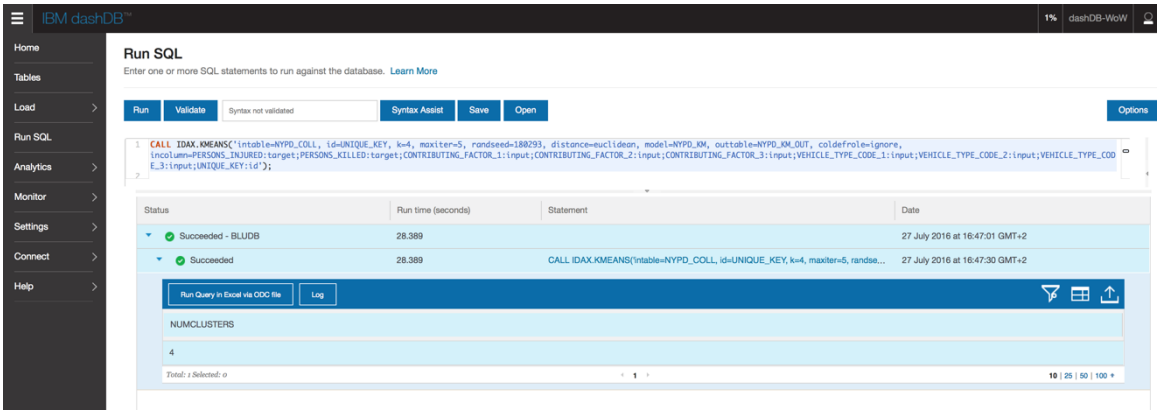
To run the k-means algorithm a stored procedure has to be called:

```
CALL IDAX.KMEANS('intable=NYPD_COLL, id=UNIQUE_KEY, k=4, maxiter=5, randseed=180293,
  distance=euclidean, model=NYPD_KM, outtable=NYPD_KM_OUT, coldefrole=ignore,
  incolumn=PERSONS_INJURED:target;PERSONS_KILLED:target;CONTRIBUTING_FACTOR_1:input;
  CONTRIBUTING_FACTOR_2:input;CONTRIBUTING_FACTOR_3:input;
  VEHICLE_TYPE_CODE_1:input;VEHICLE_TYPE_CODE_2:input;
  VEHICLE_TYPE_CODE_3:input;UNIQUE_KEY:id');
```

This call runs the k-means algorithm on the table NYPD_COLL. It uses the column UNIQUE_KEY as an identifier and a few columns to cluster the data. K-means is set to a maximum number of iterations of 5 and should create 4 clusters.

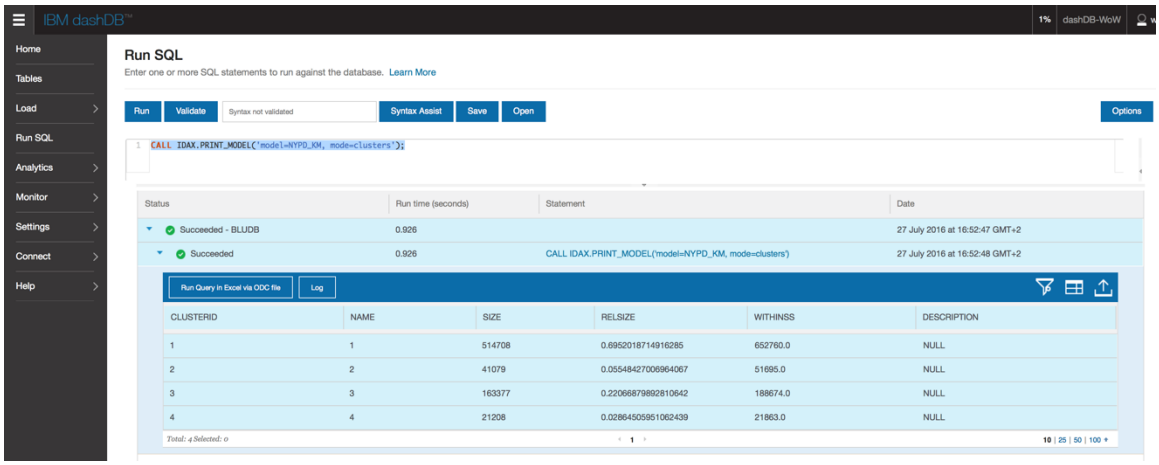
The SQL is stored in path `/home/user/Advanced-Analytics/k-means.sql`. Navigate to Run SQL on Db2, upload the SQL and run it.

The stored procedure runs for some seconds and should give the number of identified clusters as result.

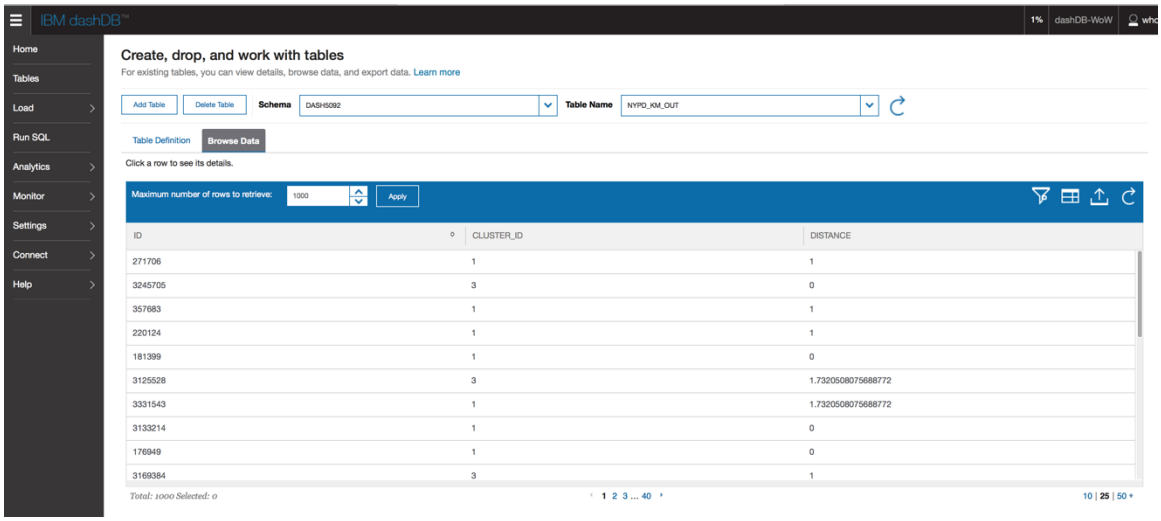


109. The k-means algorithms created some tables to store the results of the clustering. The result of the clustering is a model that represents the found clusters. To get information about this model execute the following SQL:

```
CALL IDAX.PRINT_MODEL('model=NYPD_KM, mode=clusters');
```



110. The table NYPD_KM_OUT holds the cluster information for each row of the base table NYPD_KM. Navigate to the Table view of Db2, select the NYPD_KM_OUT table and the Browse Data tab to preview the data.

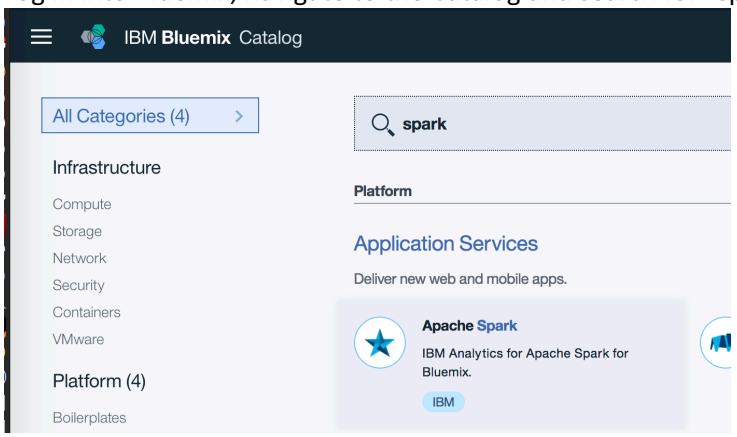


A join between NYPD_COLL and NYPD_KM_OUT would bring the cluster information together with the details of each record.

Lab 12: Accessing Db2 from Spark and Python

111. In this lab you will access the Db2 environment from Spark by using Python. In order to do so you have to create a Spark run-time environment first. Bluemix has a Spark service available and you have to create such a service as the first step.

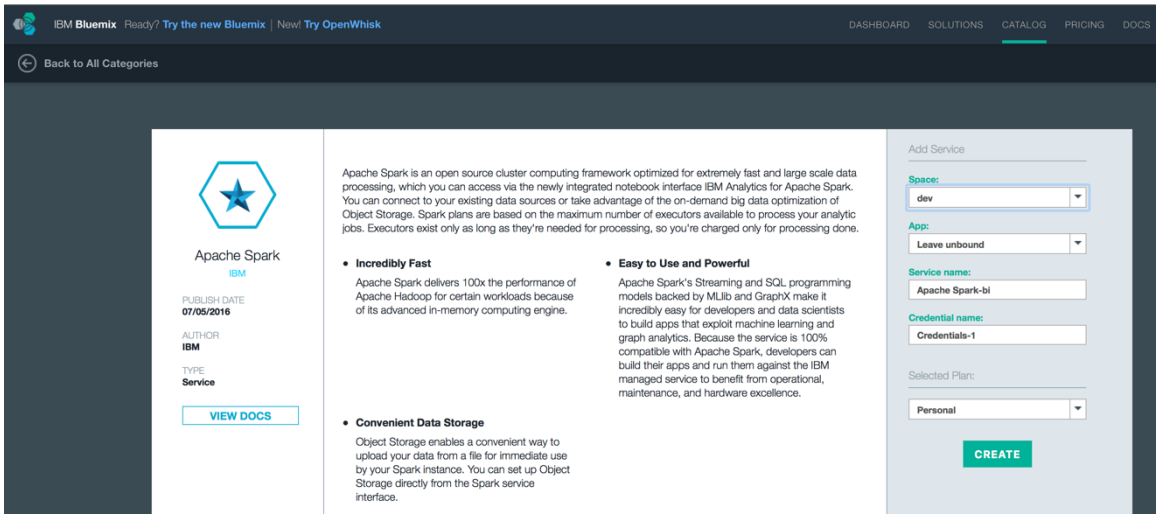
    Log in into Bluemix, navigate to the Catalog and search for “spark”.



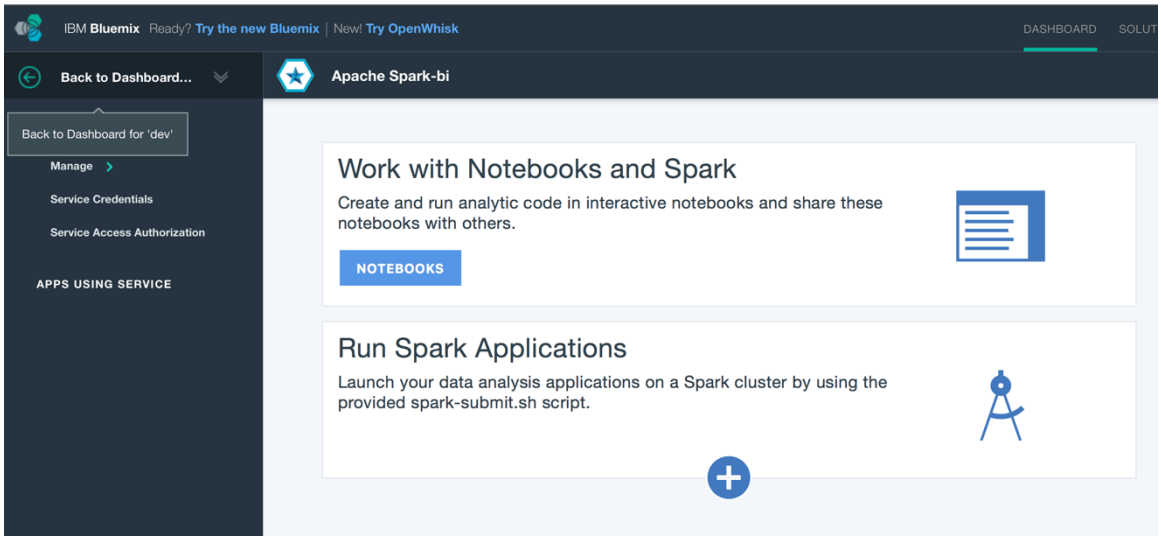
    Click on the *Apache Spark* service logo and note the Service name to identify the Spark Service later.

    Spark Service name:

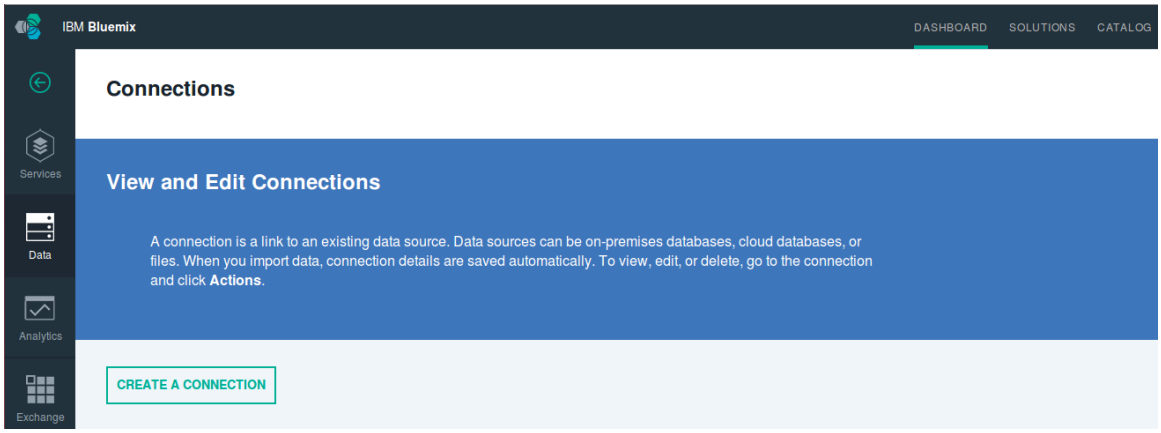
112. Create a Spark service with the Personal plan:



113. When the Spark service is started, select NOTEBOOKS to create a notebook interface for this service:

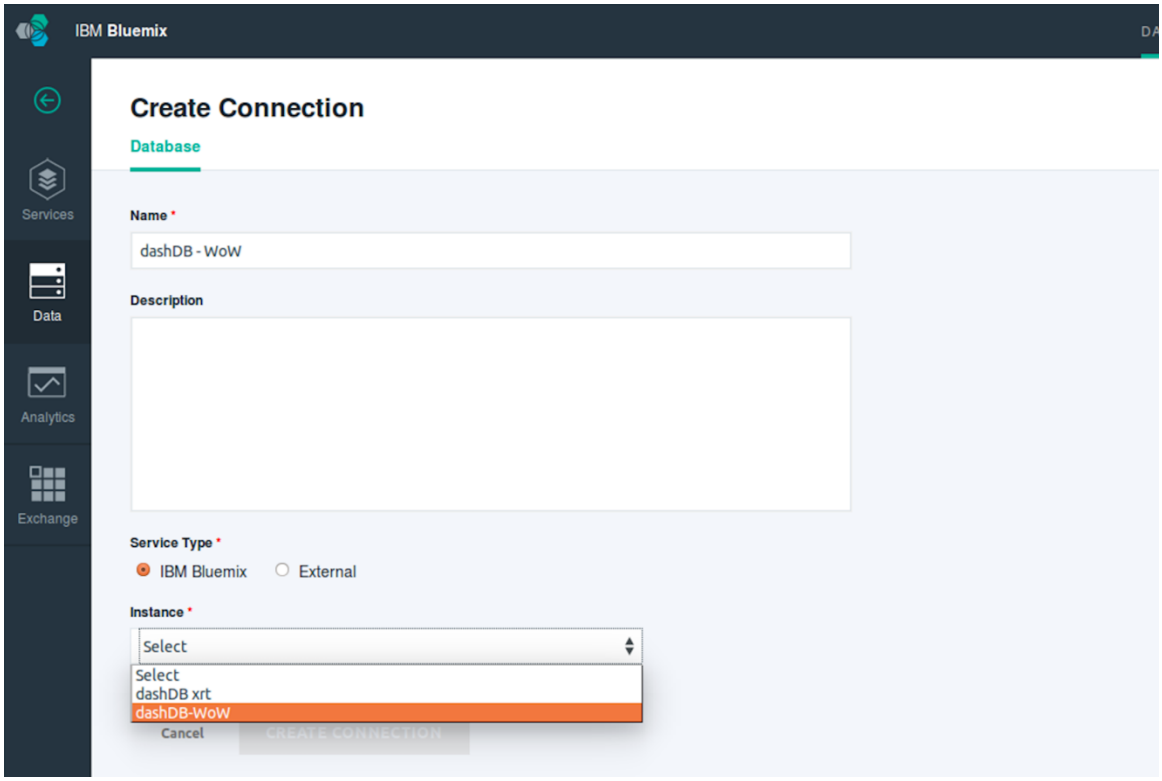


114. In the notebook we want to use data that is stored in Db2. Therefore, before creating a notebook, we have to setup a connection from Spark to the Db2 database. To configure this connection, first click on Data in the left column and then choose CREATE CONNECTION.

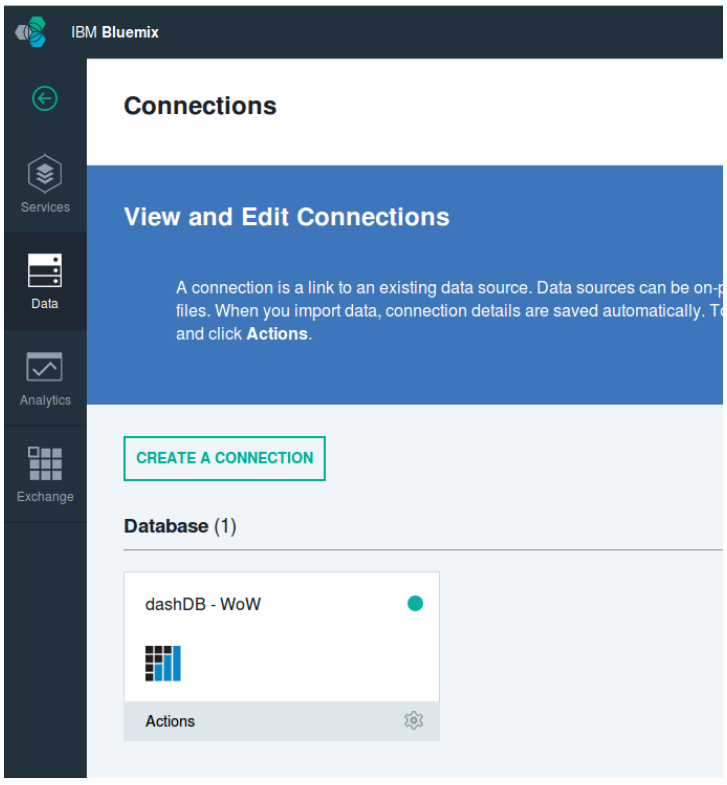


Give the database connection a name, select IBM Bluemix as Service Type and select your Db2 database as the instance. Verify, that you select the correct Db2 service by comparing it with the name noted above.

115. The database is automatically set to BLUDB. You don't have to change it. Press CREATE CONNECTION to proceed.

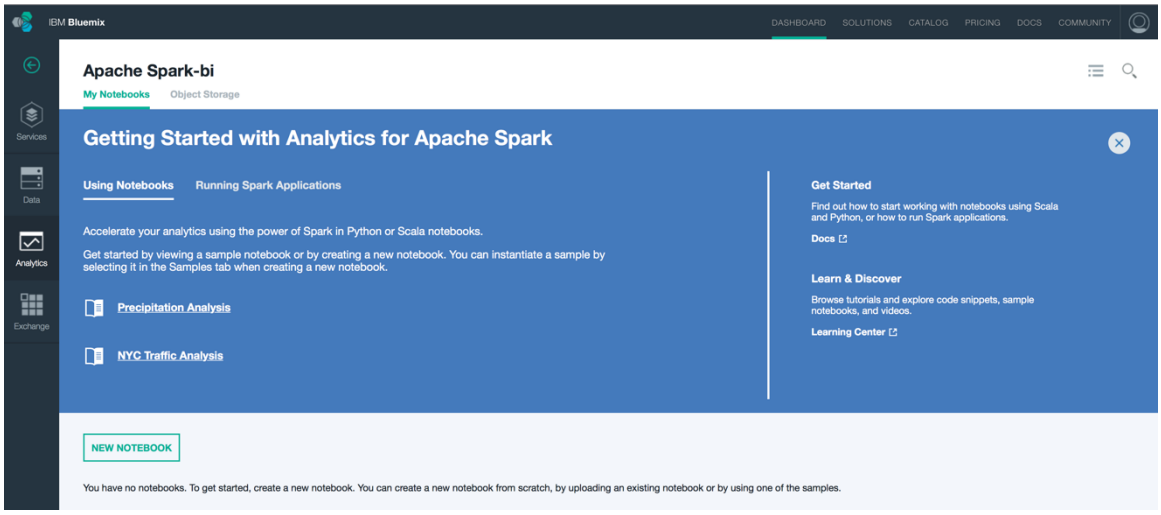


116. The connection is immediately verified and after a few seconds you should see a green marker that signals successful connection to your Db2 instance:

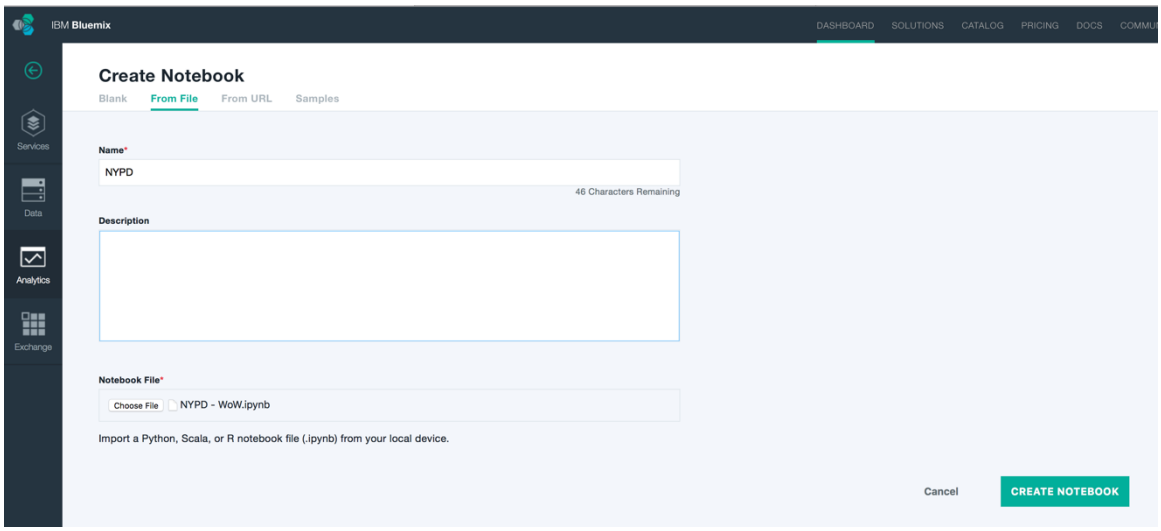


117. We now move on and create a new notebook. Verify that you are using the correct Spark Service by comparing its name in the headline of the web page with the Spark Service name noted above.

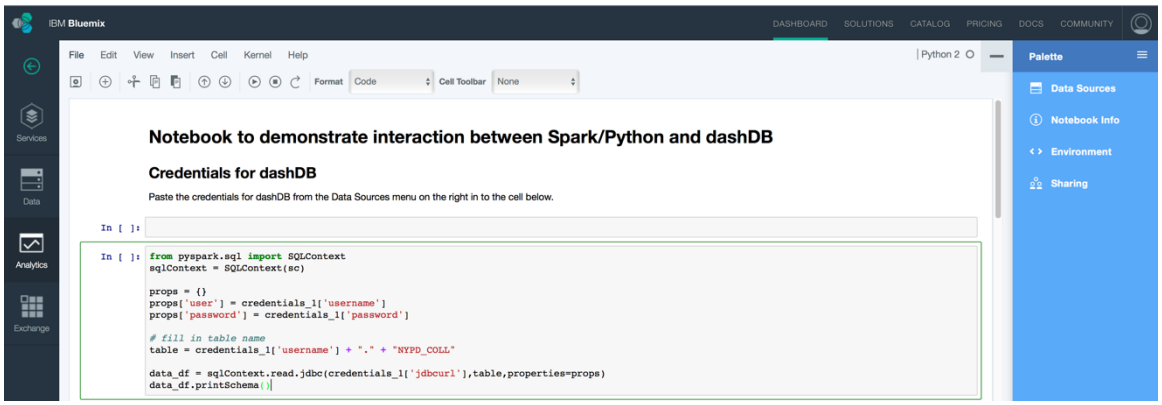
    Click on Analytics on the left and select NEW NOTEBOOK to create a new notebook:



118. A notebook to start with is stored in path `/home/user/Advanced-Analytics/NYPD-WoW.ipynb`. Select the tab From File, give the Notebook a name and choose the notebook from the path above. Press CREATE NOTEBOOK to proceed.



119. The notebook interface opens and shows the preloaded notebook. A notebook is a mix of documentation, code and results and ideally suited for ad hoc exploration of data.



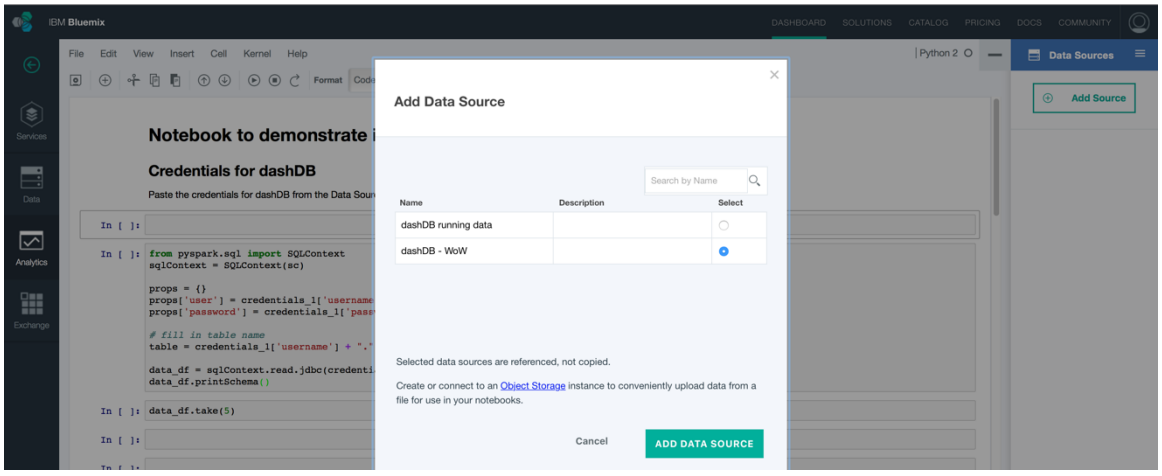
To help you using a notebook notice the following:

- You can place the cursor in any cell of the notebook and press RUN above to execute the code in that cell.
- A cell that has never been run has an empty label: `[ ]:`
- A cell that is currently executing has a star in the label: `[*]:`
- A cell that has been executed has a number in the label: `[5]:`

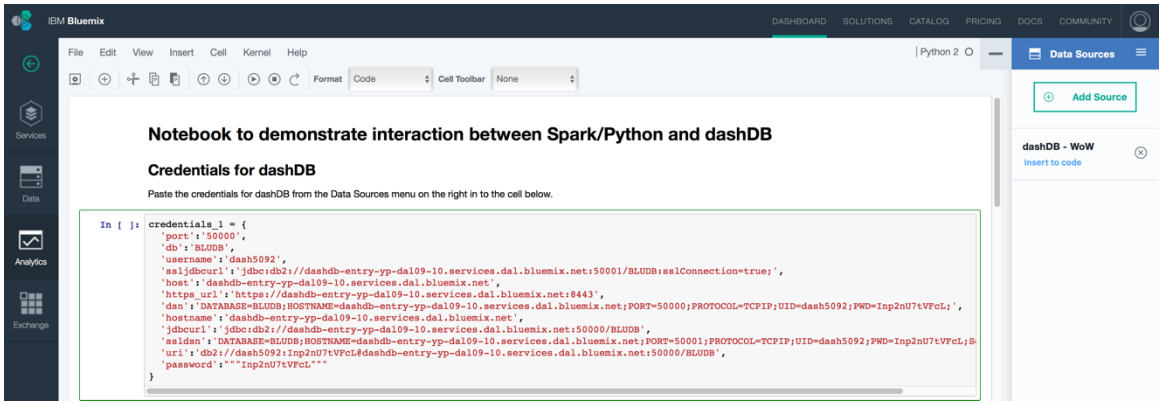
But before you execute code you have to import the credentials of the Db2 environment.

120. Place the cursor on the empty cell under “Paste the credentials …”

Select Data Sources in the column on the right and press Add Source. A popup is showing up from which you can select the Db2 connection you created before:



121. Select your Db2 instance and press ADD DATA SOURCE. After doing this the database shows up in the right column and by pressing Insert to code the credentials are copied to the empty cell:

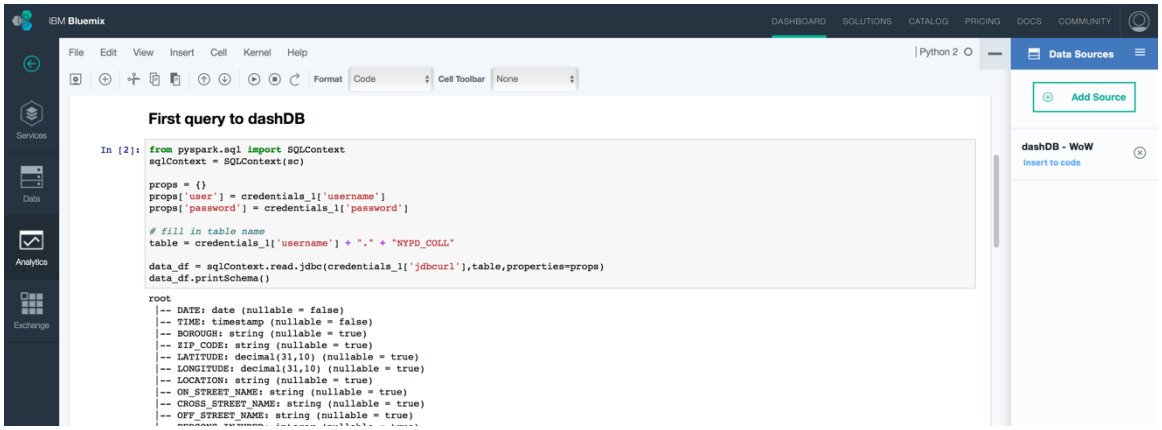


Make sure, that the structure for your credentials has the name “credentials_1”. If it is a different name, please change it to “credentials_1”. Place the cursor in that cell and press run to execute the code in this cell. This loads the credentials into the notebook environment.

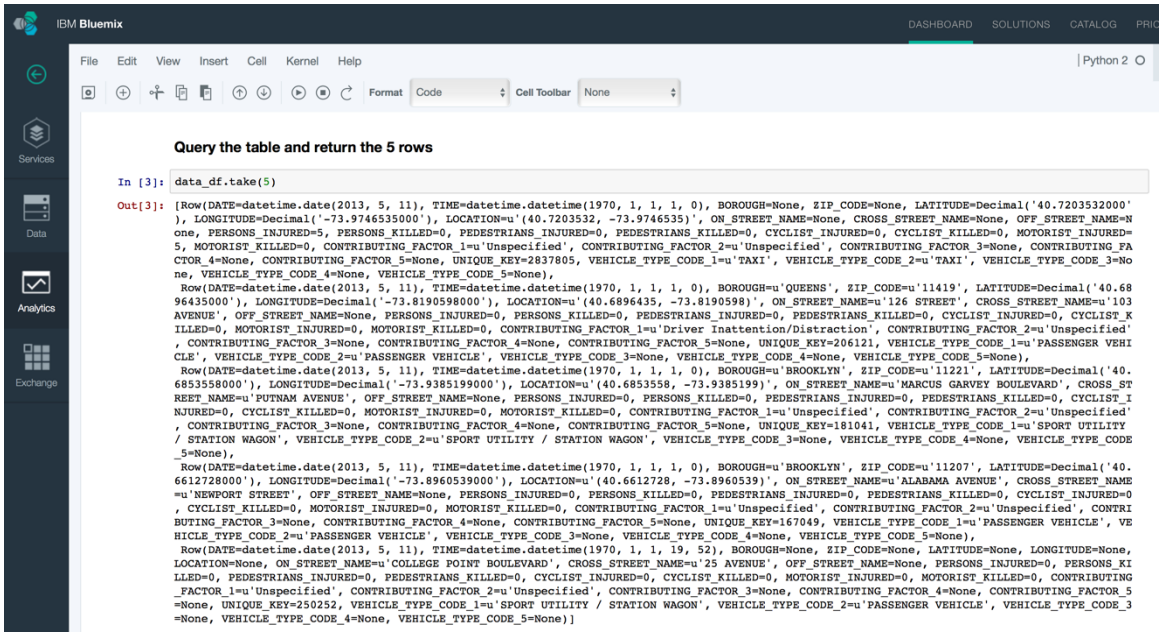
122. Now you are ready to access data in your Db2 database. You do so by placing the cursor in the next code cell and press run.

The code creates a SparkSQL context using the PySpark libraries. After setting the credential parameters and choosing the table in Db2 the database is accessed and a data frame is created as a result. The last step is printing the schema of the table.

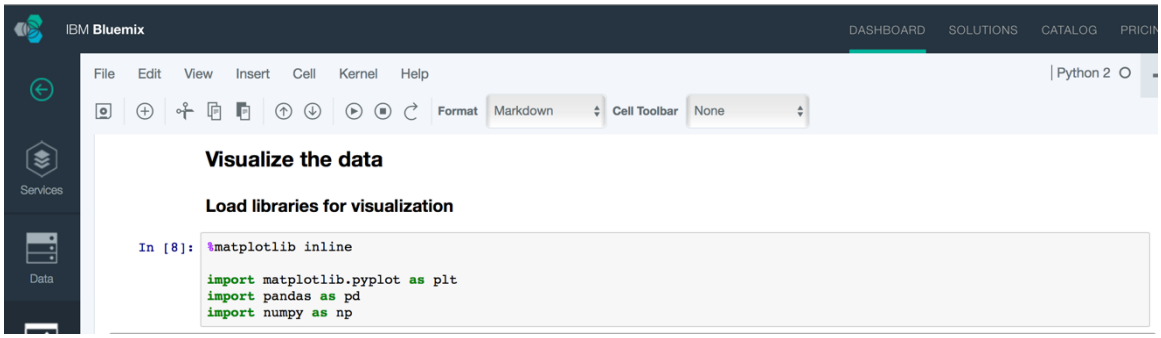
The execution of Spark is lazy. As no data from the data frame is accessed the data is not really queried from the database. That is why this statement is really fast.



123. The next code cell is printing the first 5 rows of the table. Place the cursor in the code cell and press run. It takes a moment to execute the statement as now the table is really queried and the result is loaded into the Spark environment.

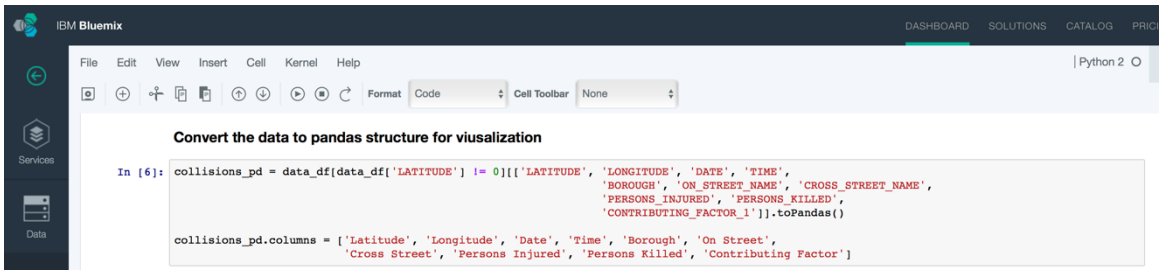


124. To visualize data, some libraries have to be loaded. The following cell is doing this for you. Place the cursor in the cell and press run.

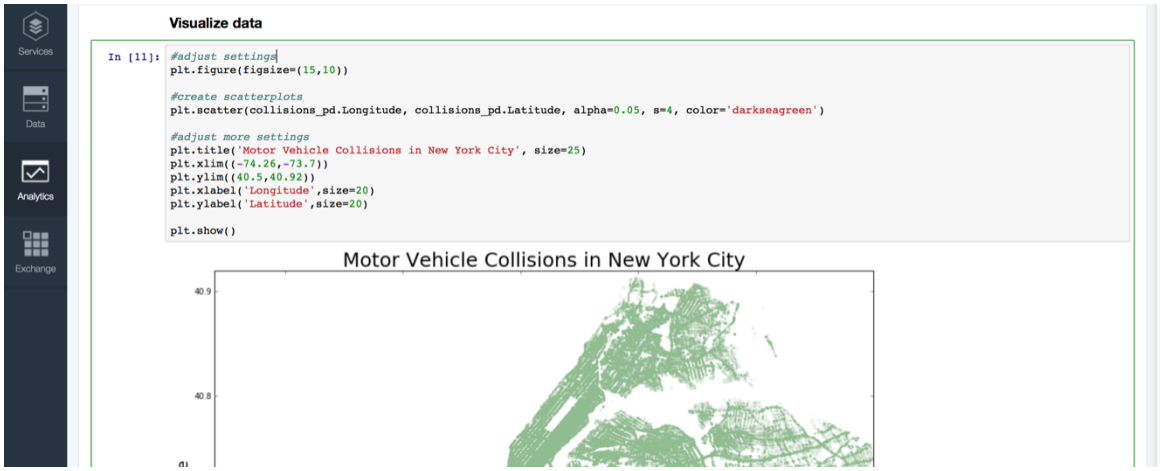


To visualize data you don't have to rely on the libraries installed with the Spark service on Bluemix. You are able to install additional packages and use them immediately.

125. To visualize the data we use the geo coordinates. Therefore, the data is filtered on rows that have a value for latitude. Run this cell. It works without Db2 involvement in the Spark environment:



The Spark kernel is executing the code in the cell until the star in the label disappears. It might take a few seconds. Finally, the visualization could be started. It is using a scatter plot and you can see the streets of New York City although no map is used underneath:

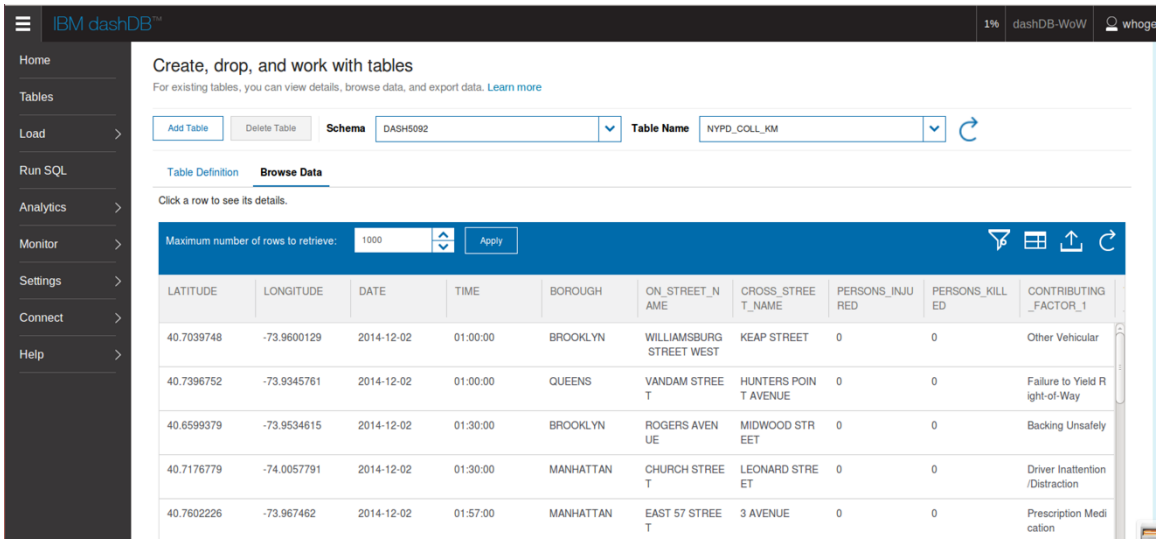


126. To access the data from the k-means clustering a view has to be created in Db2 because SparkSQL is not able to run a join on a remote database directly.

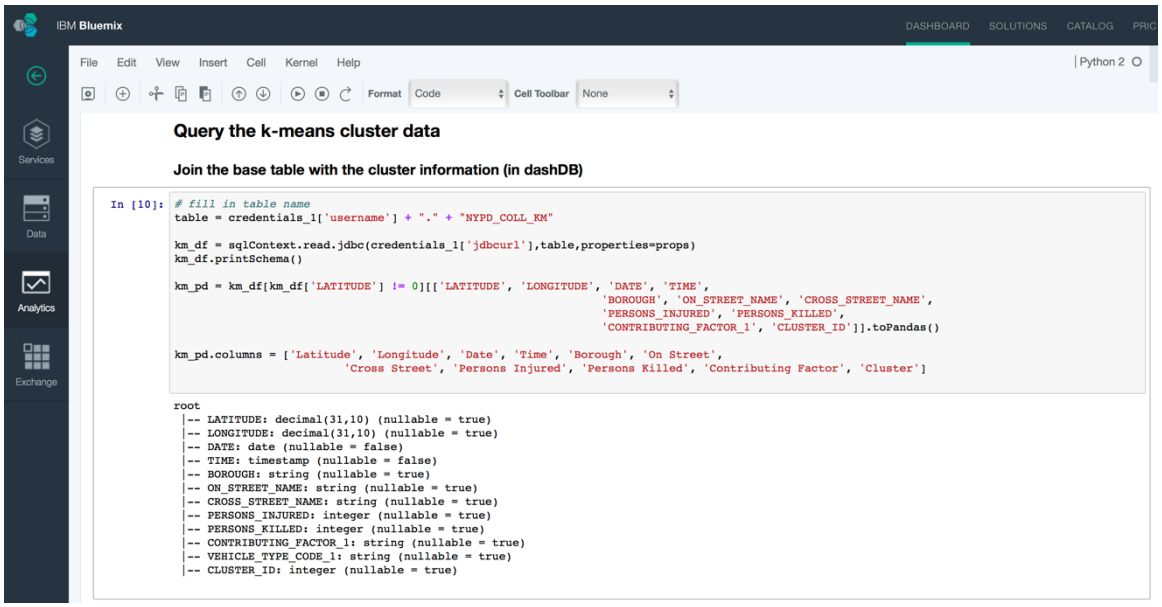
Therefore you create a view in Db2, that is doing the join instead. This join combines the cluster information that was generated by the k-means stored procedure with the base table:

```
CREATE VIEW NYPD_COLL_KM AS
  SELECT LATITUDE, LONGITUDE, DATE, TIME, BOROUGH, ON_STREET_NAME,
         CROSS_STREET_NAME,PERSONS_INJURED, PERSONS_KILLED,
         CONTRIBUTING_FACTOR_1, VEHICLE_TYPE_CODE_1, CLUSTER_ID
  FROM NYPD_COLL N, NYPD_KM_OUT K
  WHERE N.UNIQUE_KEY = K.ID;
```

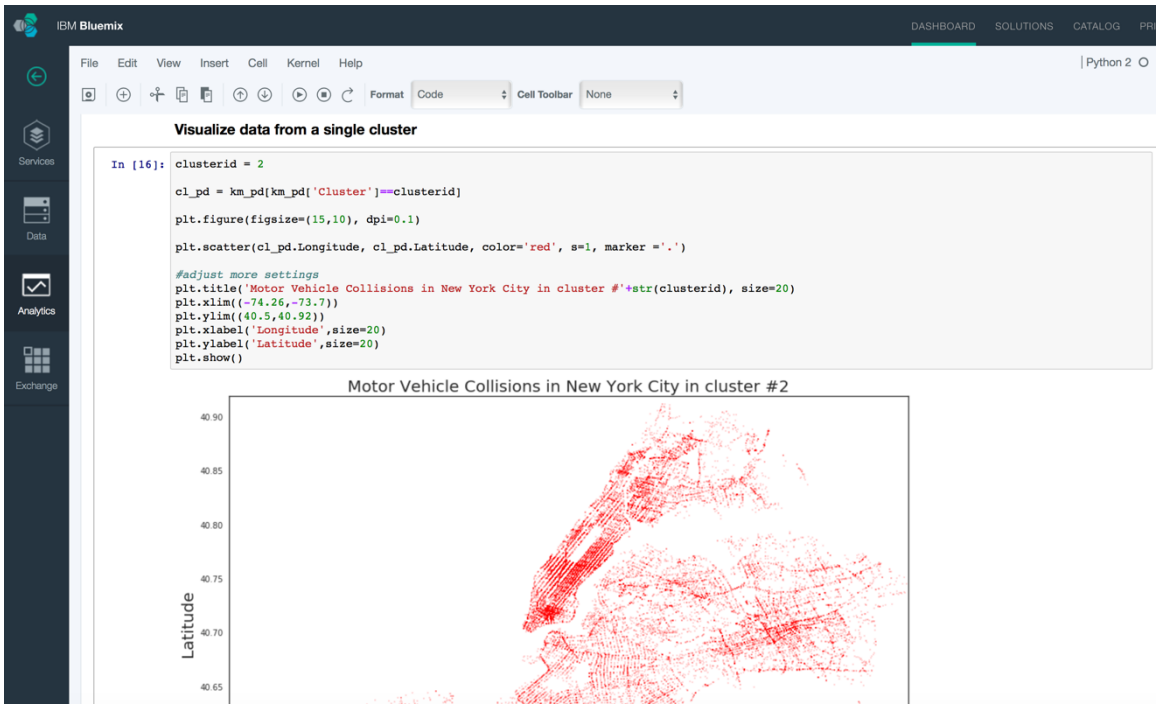
Go to the Db2 interface and select Run SQL. Press Open and select the needed DDL for the view from your local desktop. It is in `/home/user/Advanced-Analytics/NYPD/create_km_view.sql`. Execute the statement by pressing Run. After successful execution, you can preview the result of this view if you switch to the Table view of the Db2 console and select the view from the list of tables:



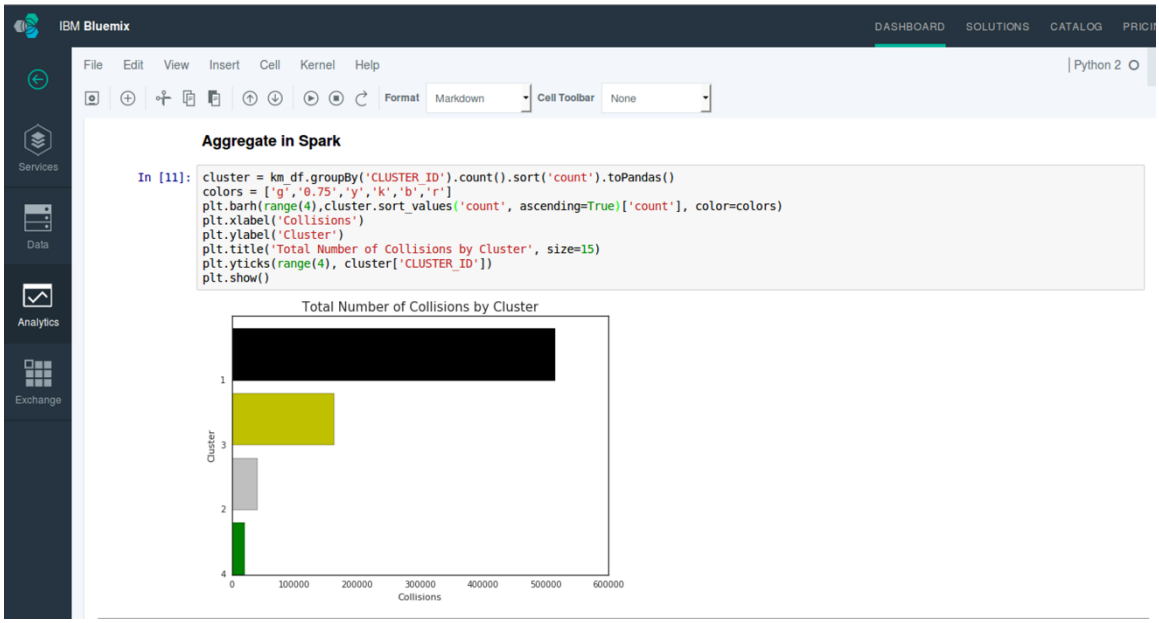
127. After the view in Db2 has created, you can execute the PySpark code to fetch the data to the Spark environment:



128. The next cell selects the rows for a specific cluster (e.g. cluster 2) and visualizes the data. You can also try different cluster ids for visualization.

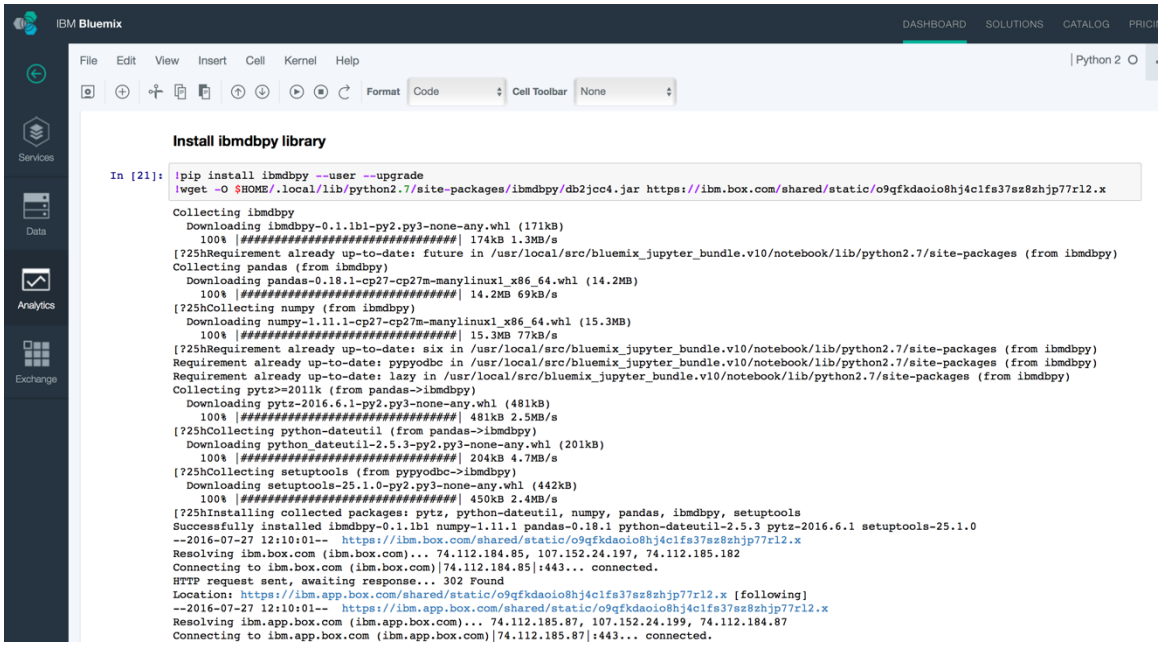


129. Next, we look at aggregations. First, the data is aggregated in Spark and visualized as a bar chart. When you run the code, it takes a while because Spark has to do the aggregation:

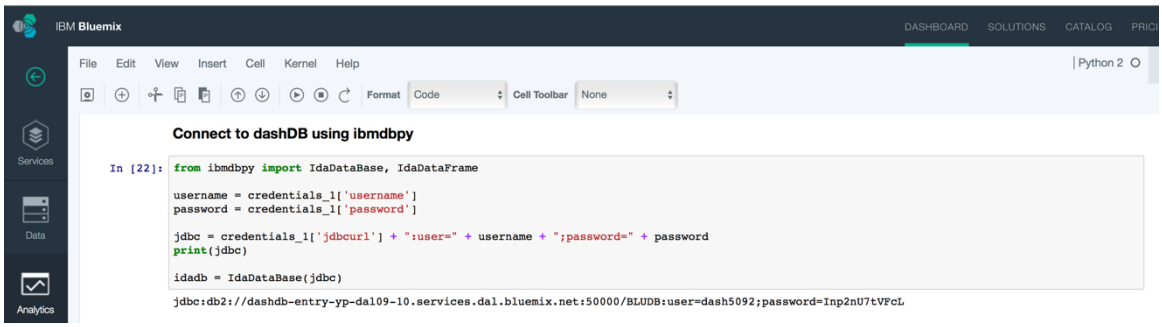


130. A different approach to query Db2 data is using the ibmdbpy library. This library offers a much simpler interface to query data.

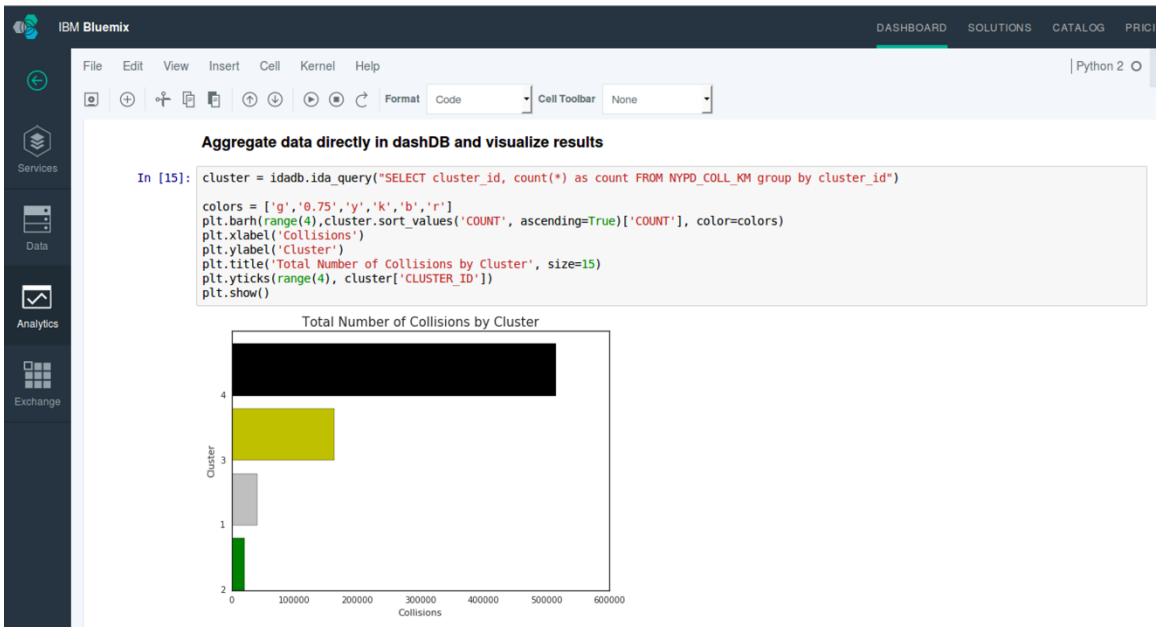
First you have to install the library and db2 driver:



After successful installation of ibmdbpy you can connect to your Db2 by running the cell code:



131. By executing the aggregation in Db2 the result is calculated much faster. The in-memory engine of Db2 is optimized to do this kind of operations and only the small result set is copied to the Spark environment. The bar chart, of course, is identical:

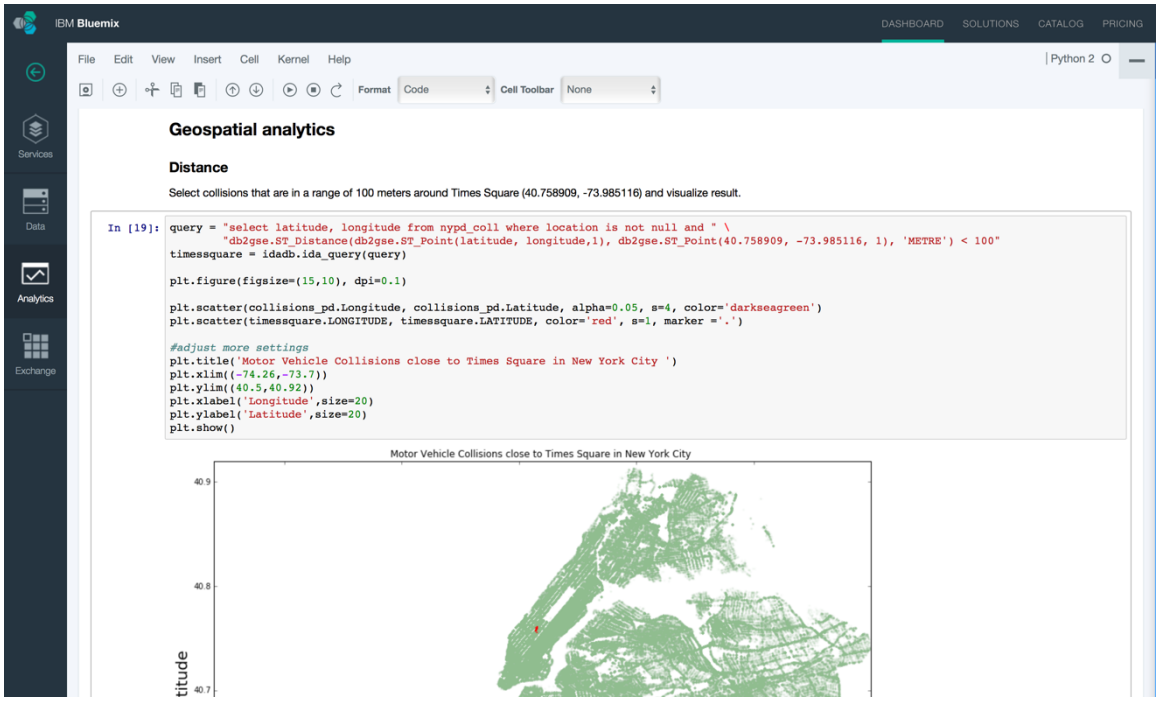


132. Db2 has advanced geospatial capabilties by leveraging the Spatial Extender (https://www.ibm.com/support/knowledgecenter/SS6NHC/com.ibm.db2.luw.spatial.topics.doc/doc/csbp1001.html). To give you an idea about dealing with geospatial data, two simple scenarios are shown here.

    First, we look at collisions that happened in a radius of 100 meters from Times Square. Times Square has the coordinates (40.758909, -73.985116). The following Db2 query select these collisions by calculating the distance of all collisions from that point:

```
select latitude, longitude from nypd_coll where location is not null and
  db2gse.ST_Distance(db2gse.ST_Point(latitude, longitude, 1),
              db2gse.ST_Point(40.758909, -73.985116, 1), 'METRE') < 100
```

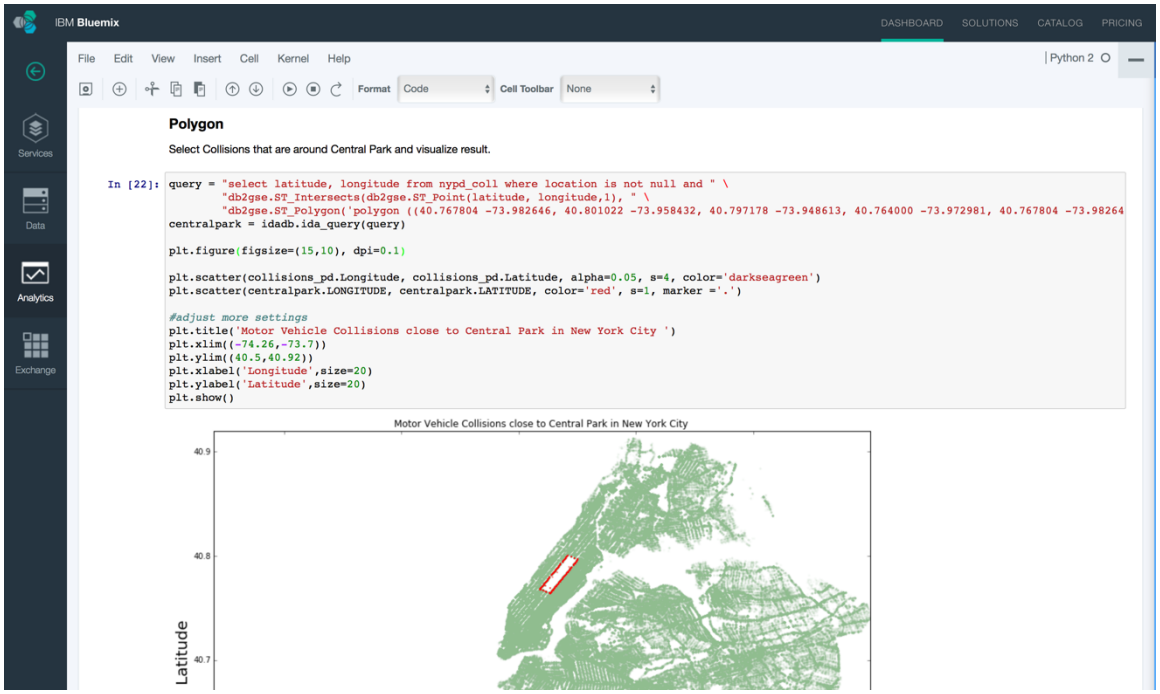
    By using the function ida_query(), the notebook executes this query against Db2 and retrieves the result as a data frame. The graph visualizes the complete dataset in green and the collisions close to Times Square in red.



133. Second, we look at collisions around Central Park. To do this, a polygon is created that includes Central Park and the streets around it. The following Db2 query select the collisions that are in this polygon by using the intersects function:

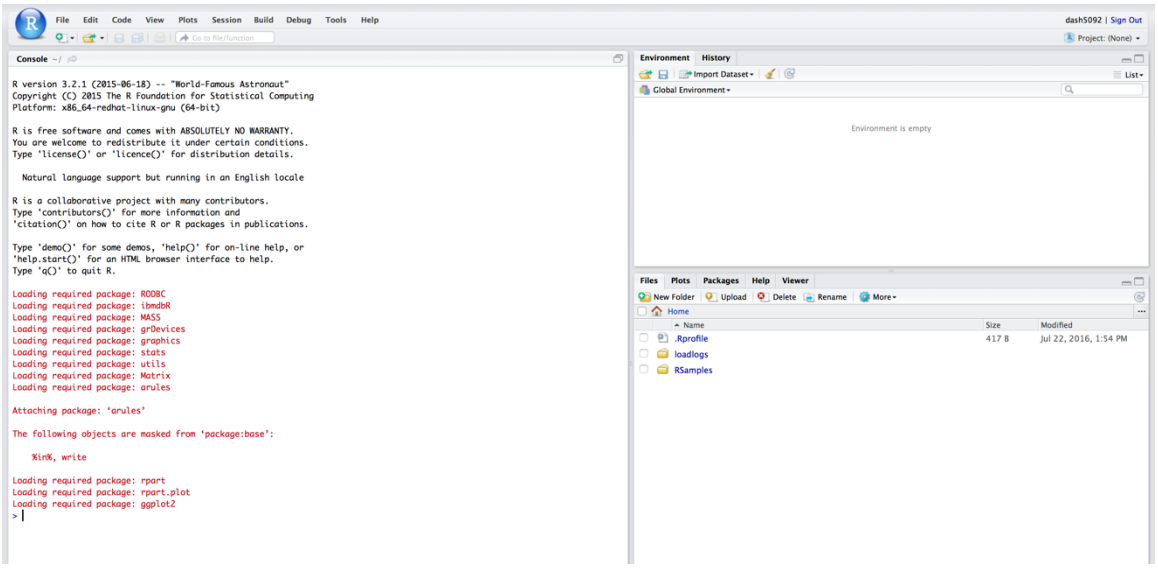
```
select latitude, longitude from nypd_coll where location is not null and
    db2gse.ST_Intersects(db2gse.ST_Point(latitude, longitude,1),
                         db2gse.ST_Polygon('polygon ((40.767804 -73.982646,
                                                      40.801022 -73.958432,
                                                      40.797178 -73.948613,
                                                      40.764000 -73.972981,
                                                      40.767804 -73.982646))',1))=1
```

By using the function ida_query(), the notebook executes this query against Db2 and retrieves the result as a data frame. The graph visualizes the complete dataset in green and the collisions around Central Park in red.


IBM Bluemix
DASHBOARD
SOLUTIONS
CATALOG
PRICING
File
Edit
View
Insert
Cell
Kernel
Help
Python 2
Format
Code
Cell Toolbar
None
Services
Data
Analytics
Exchange
Polygon
Select Collisions that are around Central Park and visualize result.
In [22]:
query = "select latitude, longitude from nypd_coll where location is not null and " \
"db2gse.ST_Intersects(db2gse.ST_Point(latitude, longitude,1), " \
"db2gse.ST_Polygon('polygon ((40.767804 -73.982646, 40.801022 -73.958432, 40.797178 -73.948613, 40.764000 -73.972981, 40.767804 -73.98264
centralpark = idadb.ida_query(query)
plt.figure(figsize=(15,10), dpi=0.1)
plt.scatter(collisions_pd.Longitude, collisions_pd.Latitude, alpha=0.05, s=4, color='darkseagreen')
plt.scatter(centralpark.LONGITUDE, centralpark.LATITUDE, color='red', s=1, marker ='.')
#adjust more settings
plt.title('Motor Vehicle Collisions close to Central Park in New York City ')
plt.xlim((-74.26,-73.7))
plt.ylim((40.5,40.92))
plt.xlabel('Longitude',size=20)
plt.ylabel('Latitude',size=20)
plt.show()
Motor Vehicle Collisions close to Central Park in New York City
40.9
40.8
40.7
Latitude

Lab 13: Analyzing data with R in RStudio

134. The Db2 service on Bluemix has an integrated RStudio environment and this is used in this lab to do analytics with R. First we have to start the RStudio environment.
When you start RStudio for the first time it will ask for credentials. For authentication, use the User ID and password that you find in the service credentials of the Db2 service in bluemix.
After successful authentication the RStudio web interface launches:

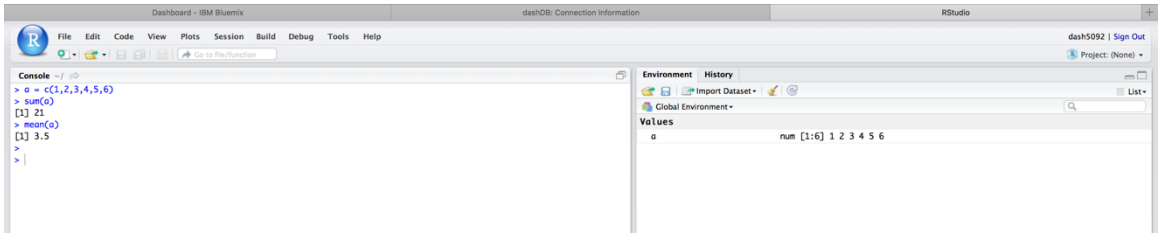


135. The RStudio interface starts with 3 areas. In the Console area you can interactively run R code. The Environment area in the top right shows the current variables and is empty at the beginning. The area in the lower right is showing the File tab. It shows the local storage that the Db2 services offers to store R scripts and data.

The RStudio environment is prepared with some R libraries that are necessary for integration with Db2. To warm-up with RStudio please press CTRL-L to clear the console area.

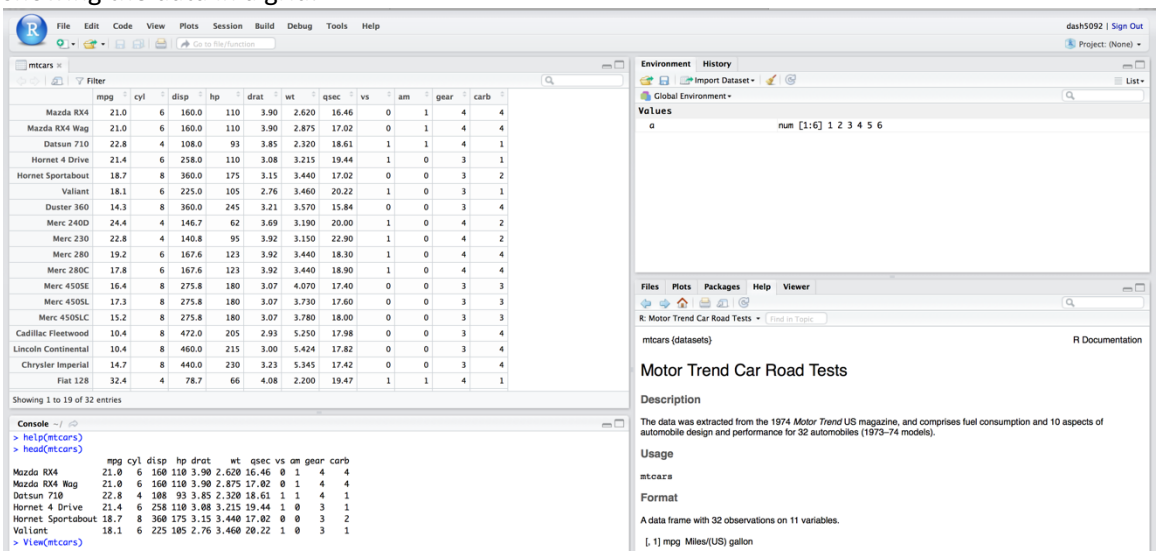
Create a variable with a list of values and do simple operations on the variable:

```
> a = c(1,2,3,4,5,6)
> sum(a)
> mean(a)
```



You can see the output of each statement right after the command. In the top right box the Variable with its contents also shows up.

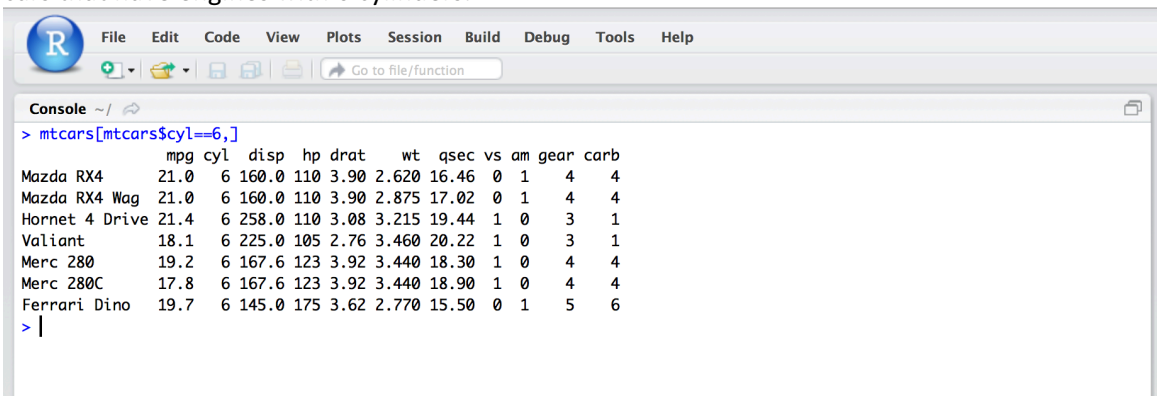
136. A convenient way to represent a table in R is a data frame. The sample data frame mtcars could be used to see how a data frame is handled in R. Type help(mtcars) to get an explaination for the data stored in mtcars. The command head(mtcars) gives you the first lines of the data frame. The View function opens a box showing the data in a grid.



There are many ways to manipulate data frames. A simple way to filter data is using the following syntax:

```
> mtcars[mtcars$cyl==6,]
```

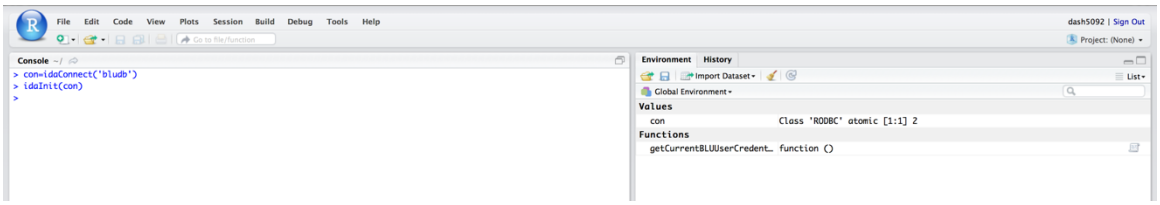
This filter selects the rows from mtcars where the value of column cyl is equal to 6, meaning selecting the cars that have engines with 6 cylinders.



137. To access data in Db2 we use the special library ibmdbR. This library is preinstalled in the Db2 RStudio environment. It is an official CRAN project and you can find documentation of ibmdbR here: https://cran.r-project.org/web/packages/ibmdbR/ibmdbR.pdf

    You initialize the package and connect to Db2 using these 2 statements:

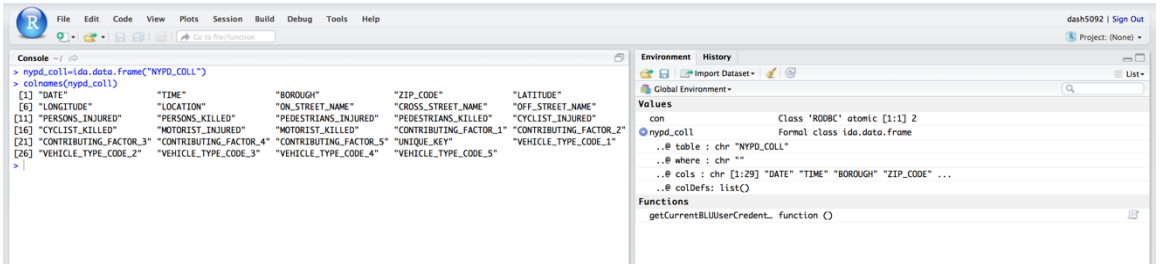
    ```
    > con=idaConnect('bludb')
    > idaInit(con)
    ```

    You can always use the autocompletion function of RStudio to get help with commands. The functions in ibmdbR start mostly with the prefix ida. Type idaS and select idaShowTables() to see all the tables that are available in your Db2 database.

138. The ida.data.frame function creates a special data frame inside the R environment that represents the given table. This function will not copy the data over to R but is just a pointer to the data in Db2:

    ```
    > nypd_coll=ida.data.frame("NYPD_COLL")
    > colnames(nypd_coll)
    ```

139. Although it is not a real data frame, many functions on data frames are still working on it. The operations on this data frame are pushed down to Db2 automatically and executed there. E.g you can calculate the number of rows in table:

    ```
    > nrow(nypd_coll)
    ```

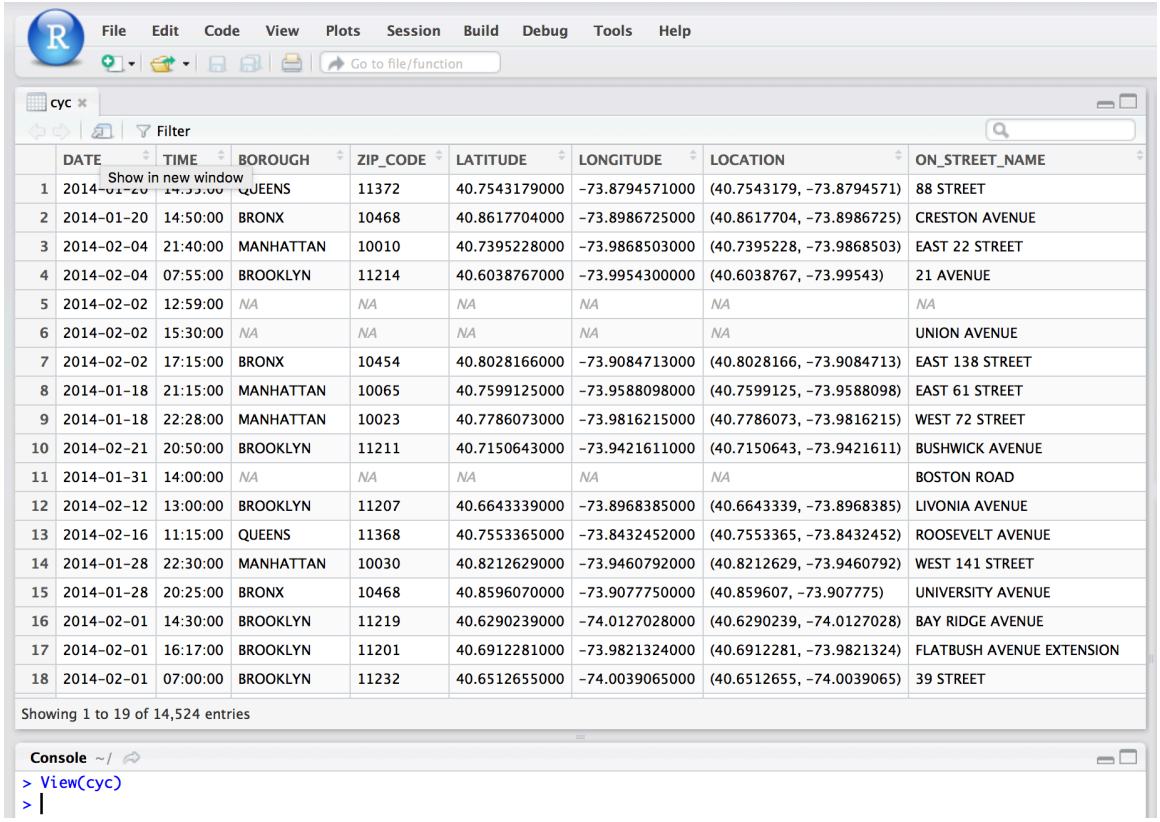
140. You can also define queries to Db2 in R syntax. If you run the following command, it will return the generated SQL query:

    ```
    > nypd_coll[nypd_coll$CYCLIST_INJURED == 1,]
    ```

141. If you store it into a variable you can also calculate the number of rows in the result set of that query.

```
> cyc = nypd_coll[nypd_coll$CYCLIST_INJURED == 1,]
> nrow(cyc)
```

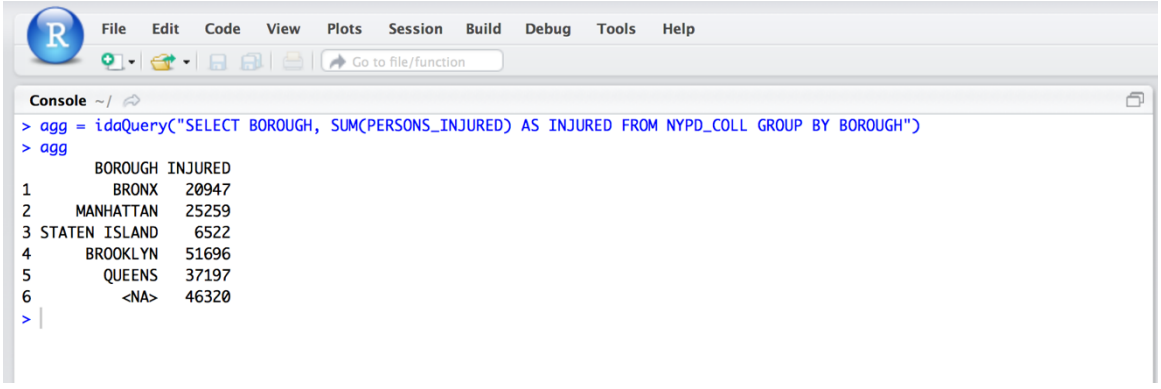
142. To look at the data in RStudio, the View function could be used on any ida data frame. It retrieves a sample of the rows fromt the Db2 table (or query) and shows it in a grid:



	DATE	TIME	BOROUGH	ZIP_CODE	LATITUDE	LONGITUDE	LOCATION	ON_STREET_NAME
1	2014-01-20	14:55:00	QUEENS	11372	40.7543179000	-73.8794571000	(40.7543179, -73.8794571)	88 STREET
2	2014-01-20	14:50:00	BRONX	10468	40.8617704000	-73.8986725000	(40.8617704, -73.8986725)	CRESTON AVENUE
3	2014-02-04	21:40:00	MANHATTAN	10010	40.7395228000	-73.9868503000	(40.7395228, -73.9868503)	EAST 22 STREET
4	2014-02-04	07:55:00	BROOKLYN	11214	40.6038767000	-73.9954300000	(40.6038767, -73.99543)	21 AVENUE
5	2014-02-02	12:59:00	NA	NA	NA	NA	NA	NA
6	2014-02-02	15:30:00	NA	NA	NA	NA	NA	UNION AVENUE
7	2014-02-02	17:15:00	BRONX	10454	40.8028166000	-73.9084713000	(40.8028166, -73.9084713)	EAST 138 STREET
8	2014-01-18	21:15:00	MANHATTAN	10065	40.7599125000	-73.9588098000	(40.7599125, -73.9588098)	EAST 61 STREET
9	2014-01-18	22:28:00	MANHATTAN	10023	40.7786073000	-73.9816215000	(40.7786073, -73.9816215)	WEST 72 STREET
10	2014-02-21	20:50:00	BROOKLYN	11211	40.7150643000	-73.9421611000	(40.7150643, -73.9421611)	BUSHWICK AVENUE
11	2014-01-31	14:00:00	NA	NA	NA	NA	NA	BOSTON ROAD
12	2014-02-12	13:00:00	BROOKLYN	11207	40.6643339000	-73.8968385000	(40.6643339, -73.8968385)	LIVONIA AVENUE
13	2014-02-16	11:15:00	QUEENS	11368	40.7553365000	-73.8432452000	(40.7553365, -73.8432452)	ROOSEVELT AVENUE
14	2014-01-28	22:30:00	MANHATTAN	10030	40.8212629000	-73.9460792000	(40.8212629, -73.9460792)	WEST 141 STREET
15	2014-01-28	20:25:00	BRONX	10468	40.8596070000	-73.9077750000	(40.859607, -73.907775)	UNIVERSITY AVENUE
16	2014-02-01	14:30:00	BROOKLYN	11219	40.6290239000	-74.0127028000	(40.6290239, -74.0127028)	BAY RIDGE AVENUE
17	2014-02-01	16:17:00	BROOKLYN	11201	40.6912281000	-73.9821324000	(40.6912281, -73.9821324)	FLATBUSH AVENUE EXTENSION
18	2014-02-01	07:00:00	BROOKLYN	11232	40.6512655000	-74.0039065000	(40.6512655, -74.0039065)	39 STREET

143. You can also run queries on Db2 and store the result in a local data frame. The idaQuery function does exacly that. Run it to calculate the number of persons injured by borough:

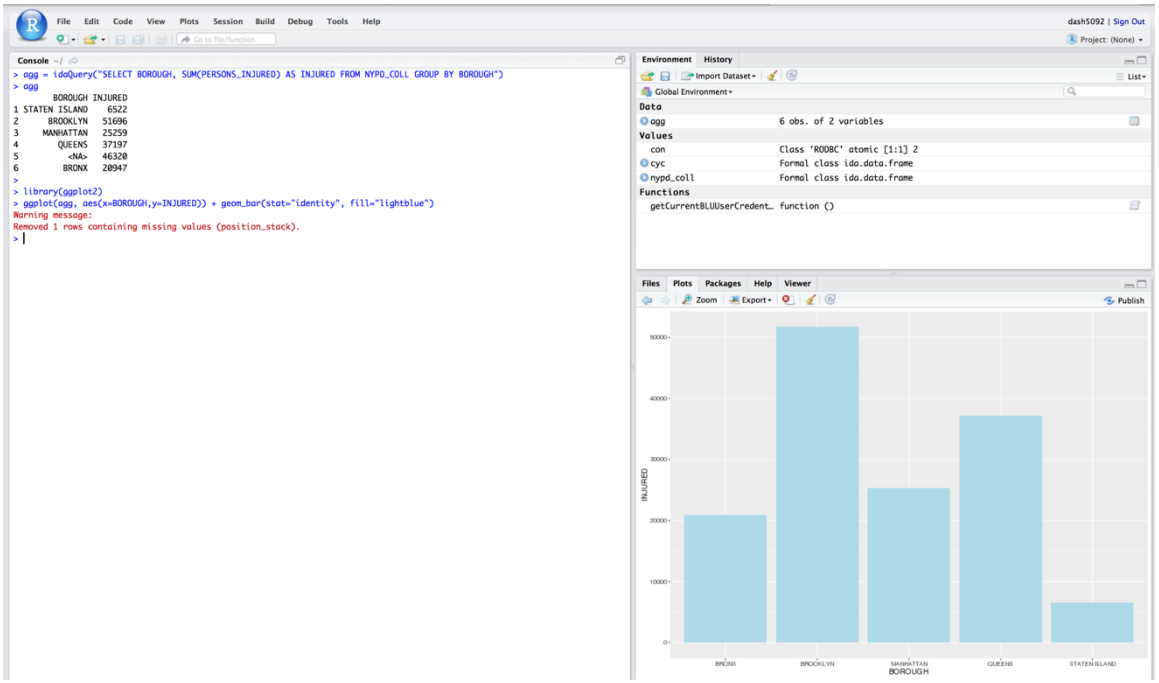
```
> agg = idaQuery("SELECT BOROUGH, SUM(PERSONS_INJURED) AS INJURED ",
                 "FROM NYPD_COLL GROUP BY BOROUGH")
> agg
```



It is advisable to use idaQuery only for queries that return not too many rows, because the complete result set is transferred to the R environment.

144. To visualize the data a simple way is to use the ggplot2 library. The following code loads the ggplot2 library and creates a bar plot with the data from the aggregation above:

```
> library(ggplot2)
> ggplot(agg, aes(x=BOROUGH,y=INJURED)) + geom_bar(stat="identity", fill="lightblue")
```



145. You can see, that one value is missing from the bar chart. It is the row with no information for borough. Identify the row number that has the missing content and to access it use the following code:

```
> agg[<row-num>,]
```

You can not only read the data but also set the values. Use this code to set a value for missing borough:

```
> agg[<row-num>,]$BOROUGH="none"
```

Rerun the ggplot statement to see the complete data in the graph.

146. The ibmdbR library supports the Db2 machine learning functionality directly in R. The k-means clustering showed in the “Maching Learning with k-means in Db2” lab could be executed in RStudio directly.

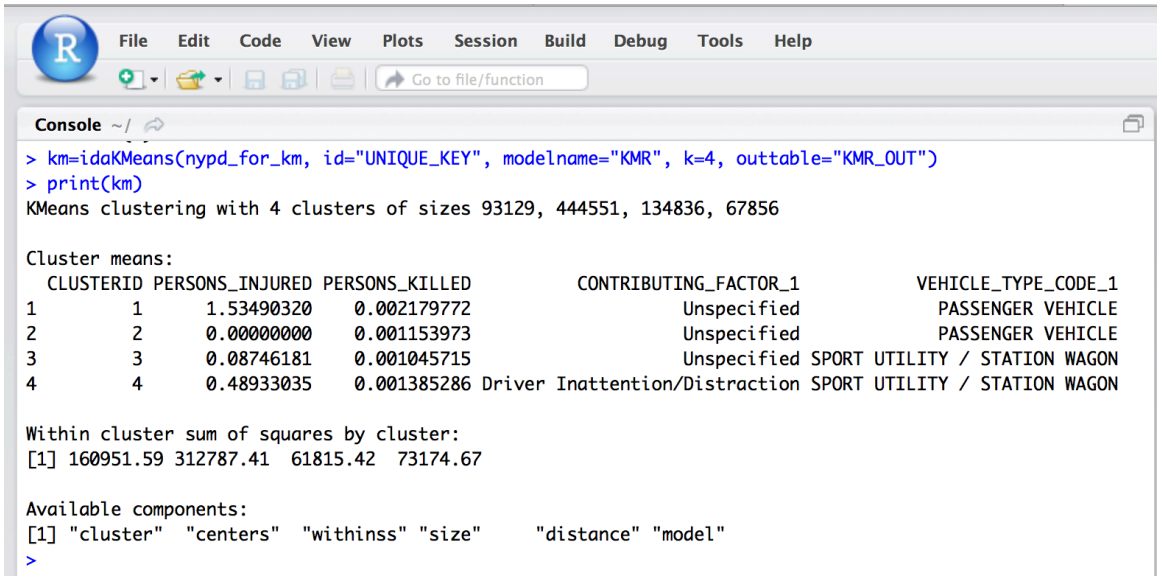
The clustering should be executed on only a subset of the columns. Therefore a new data frame is created that select just the columns needed:

```
> nypd_for_km=nypd_coll[,c('UNIQUE_KEY', 'PERSONS_INJURED', 'PERSONS_KILLED',
                           'CONTRIBUTING_FACTOR_1','VEHICLE_TYPE_CODE_1')]
```

147. The data frame nypd_for_km is not materialized. It is just a projection of the original table nypd_coll. The following call to the k-means function creates 4 clusters:

```
> km=idaKMeans(nypd_for_km, id="UNIQUE_KEY", modelname="KMR", k=4, outtable="KMR_OUT")
```

148. After creating the clustering model, it can be printed to the console. The printout shows the cluster sizes, cluster centers and the sum of the distances to the centers for all values.



149. The output table with the mapping of each row to its cluster is created in the Db2 database. It can be accessed and visualized by:

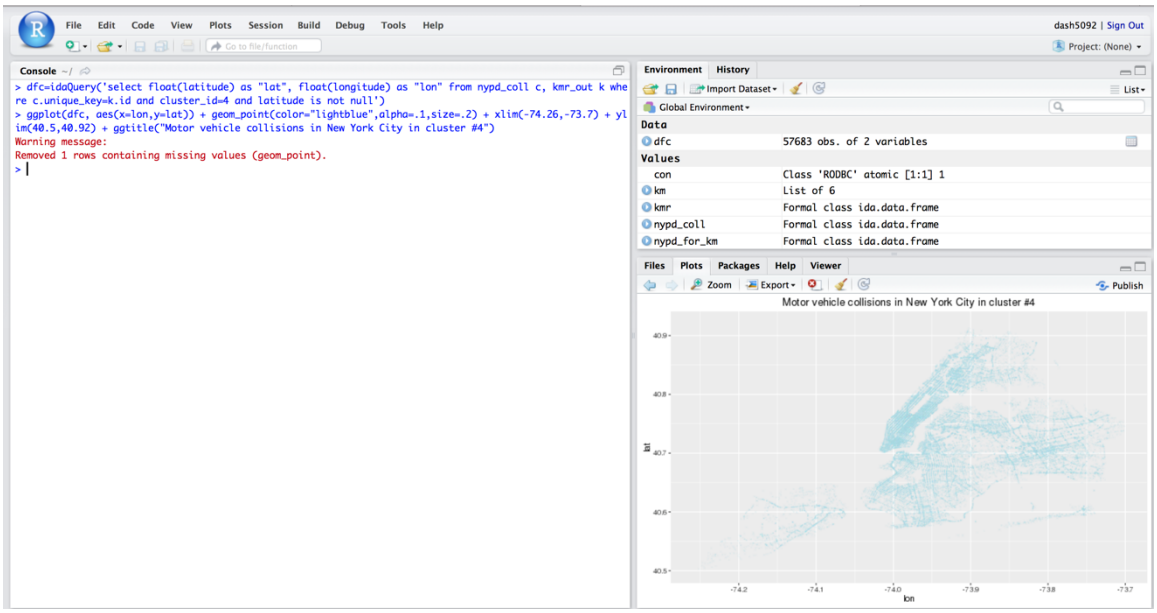
```
> kmr=ida.data.frame("KMR_OUT")
> View(kmr)
```

150. In order to visualize a cluster (e.g. cluster #4), the KMR_OUT table is joined with the NYPD_COLL table.

```
> dfc=idaQuery('select float(latitude) as "lat", float(longitude) as "lon"
                from nypd_coll c, kmr_out k
                where c.unique_key=k.id and cluster_id=4 and latitude is not null')
```

151. The idaQuery function joins the tables and selects the geo coordinates as floats. The result of the idaQuery function is a regular data frame in R that can now be visualized with ggplot2:

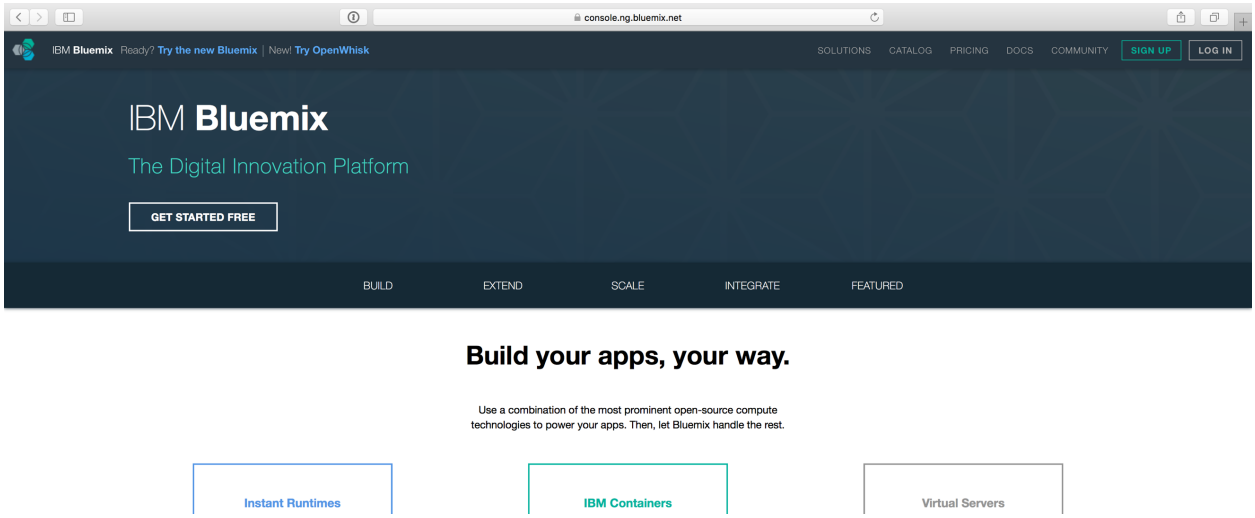
```
> ggplot(dfc, aes(x=lon,y=lat)) + geom_point(color="lightblue",alpha=.1,size=.2)
                + xlim(-74.26,-73.7) + ylim(40.5,40.92)
                + ggtitle("Motor vehicle collisions in New York City in cluster #4")
```



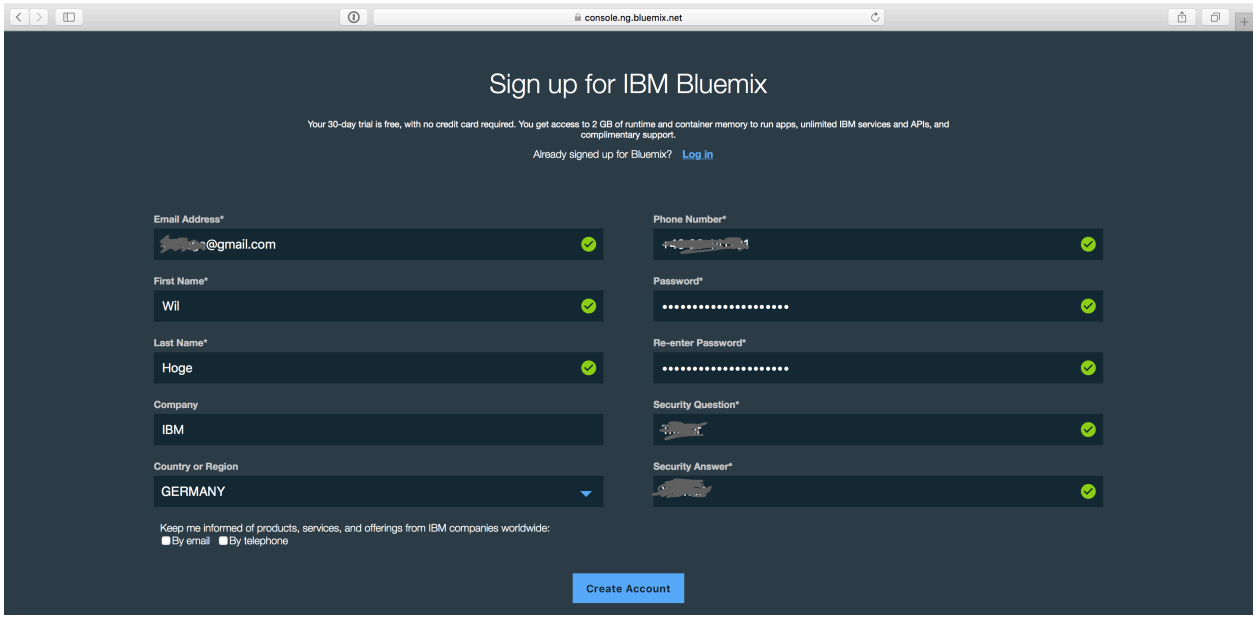
If you want to view the graph at a larger scale you can press the Zoom button upon the graphics. This starts a new browser window with a larger graphics.

Appendix I: Create Bluemix and DSX account

1. Go to the Bluemix website http://bluemix.net and click on “GET STARTED FREE” oder “SIGN UP”:

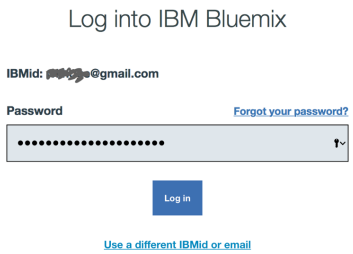


2. Enter your data and choose an email address where you have inbox access via browser or smartphone.

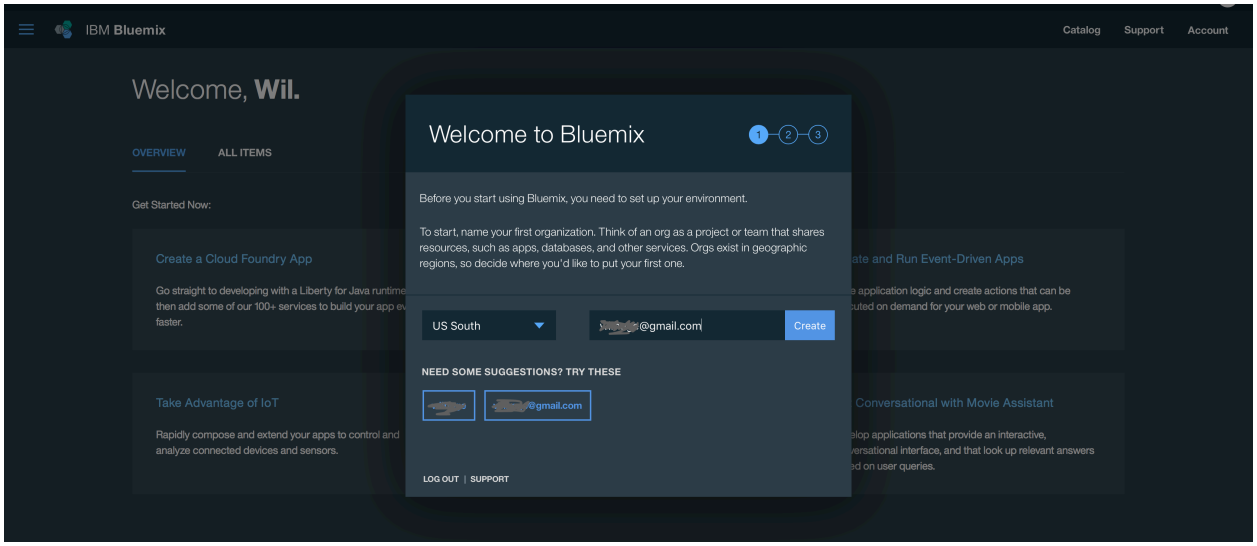


3. After pressing “Create Account” you will receive a registration email that includes the link “Confirm Account” you have to press. The registration email should be sent within minutes. If you are not receiving it, please check the spam folder also.

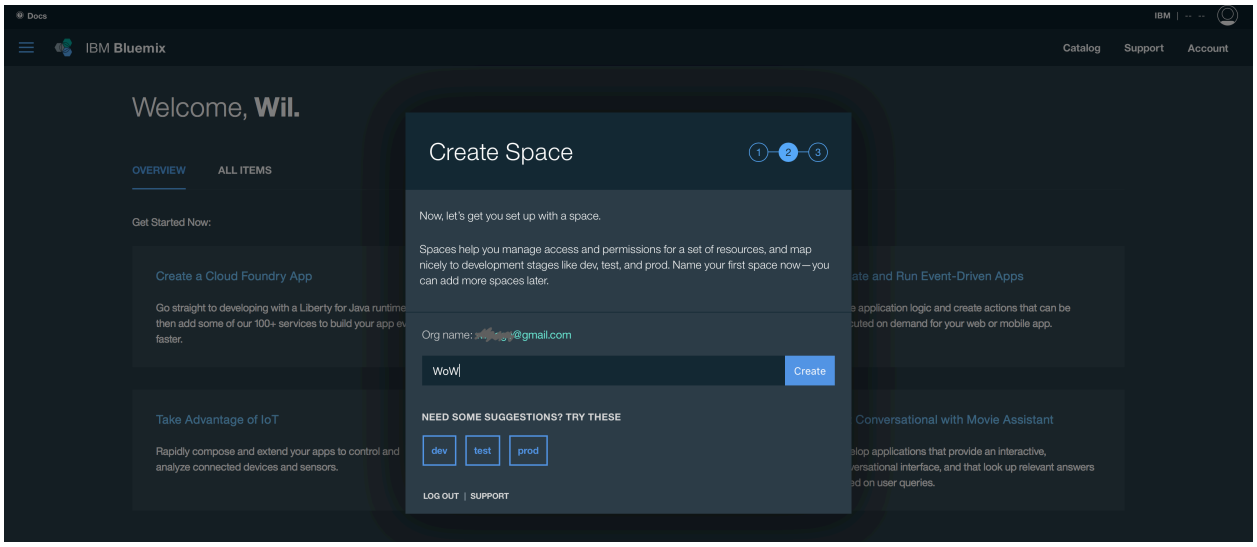
4. Log into Bluemix by clicking the “LOG IN” button on the bluemix website http://bluemix.net. Enter the email address you used above. This email address is now your IBMid.



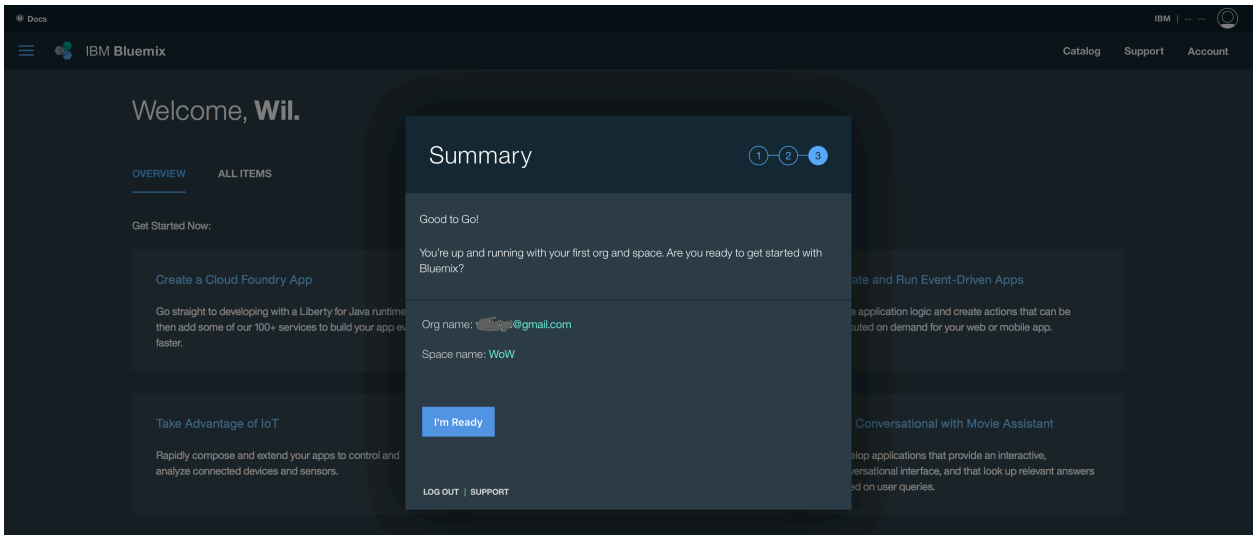
5. You have to setup your environment first. Choose “US South” for the region and use your email address as the organization name:



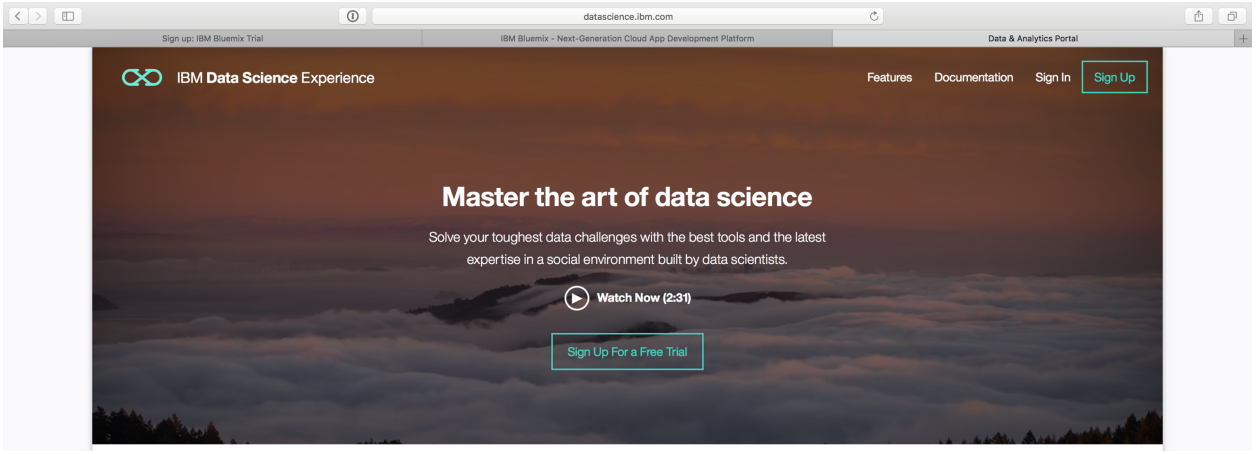
6. Next, you have to create a space for your services. You can choose any name, but “WoW” would be a good choice:



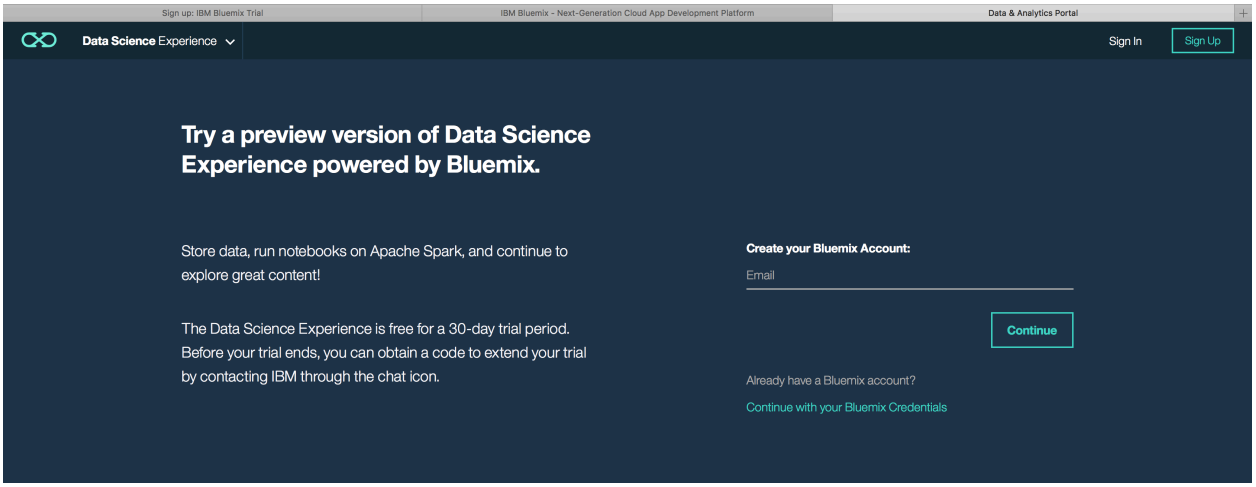
7. Now, having created a Bluemix account, setup region, organization, and space you are ready to do the lab. Press “I’m Ready” to move on.



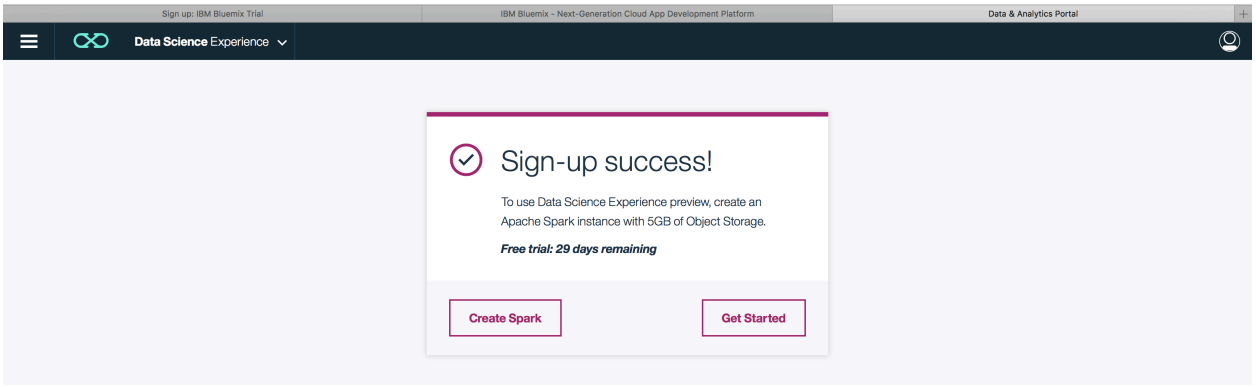
8. If you want to use Data Science Experience, the new UX interface for data scientists you can also register for it now. Go to http://datascience.ibm.com and press “Sign Up” or “Sign Up For a Free Trial”:



9. Choose “Continue with your Bluemix Credentials” to use your Bluemix account for Data Science Experience:



10. You are immediately registered for Data Science Experience. Pressing “Get Started” will bring you to the user interface. Be aware that we are not using Data Science Experience in this lab.



Appendix II: Start vmware image and docker container

1. Start the image ***Db2-Local*** in vmware workstation
   The user you are working with is
   name: `user`
   password: `user`

2. Open a terminal window to start the docker container for Db2 and RStudio. The commands are:
   ```
   .do docker start Db2
   .do docker start RStudio
   ```

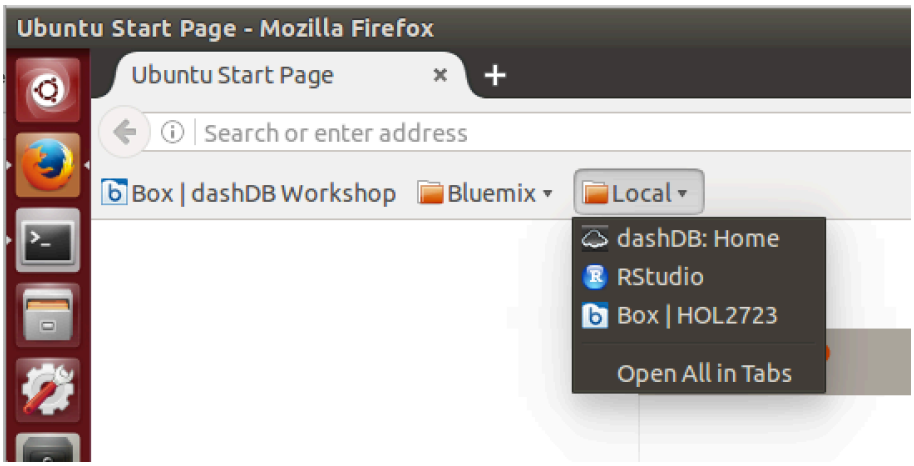
   Double check if the container are running with
   ```
   .do docker ps –a
   ```

   The output should look like this:
   ```
   user@ubuntu:~$ .do docker ps -a
   CONTAINER ID        IMAGE                              COMMAND                  CREATED
   STATUS              PORTS                    NAMES

   2b7c8c113d66        rstudio:latest                     "/bin/bash /tmp/start"   19 minutes
   ago      Up About a minute   0.0.0.0:8787->8787/tcp   RStudio

   9eb8f5c372b1        ibmdashdb/local:latest-linux       "/usr/sbin/init"         20 minutes
   ago      Up About a minute                               Db2
   ```

3. In a browser open in separate tabs the Db2 Warehouse console and RStudio
   Db2 Warehouse: https://127.0.1.1:8443
   RStudio: http://127.0.1.1:8787



Acknowledgements and Disclaimers

Availability. References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS-IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.