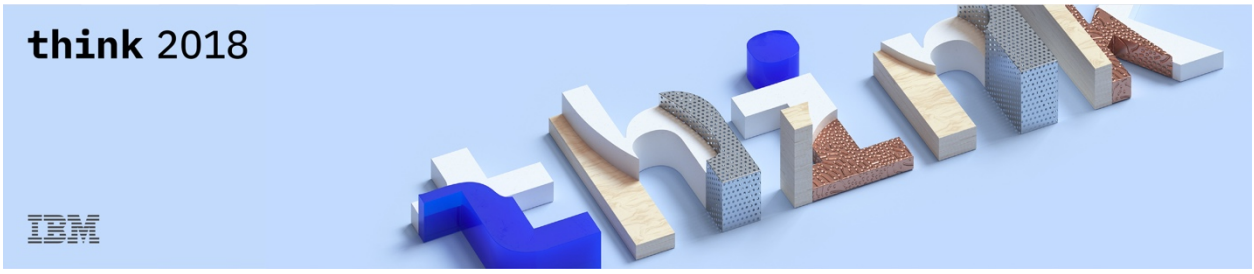



# Lab Center – Hands-on Lab

## Session 4822

## Advanced Analytics in Db2 Warehouse

Andreas Weininger, IBM, andreas.weininger@de.ibm.com

Stefan Hummel, IBM, stefan.hummel@de.ibm.com

# Table of Contents

## Disclaimer

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results like those stated here.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts.
In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

**Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.**

Performance data contained herein was generally obtained in controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may

need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.



**U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.**

## Objectives of this lab

The goal of this lab is to show how developers can perform advanced analytics with data stored in Db2 warehouse, 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?

In this workshop we are using a VMware image with all needed software components. All of them are available as well in the IBM Cloud, the Platform as a Service (PaaS) offering by IBM.

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/ibmuser/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 types 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

The VMware image contains this software products:

- Portainer
  A simple to use management user interface for your Docker environments.
  http://localhost:9000

- Db2 warehouse
  Installed in a docker container.
  https://localhost:8443

- RStudio
  Installed in a docker container.
  http://localhost:8787

- Jupyter Notebooks

In the Firefox web browser are all the bookmarks to the web frontends you need for the labs.

The user account are:

| System | User | Password |
|---|---|---|
| Linux | ibmuser | passw0rd<br>(zero instead of o) |
| Portainer | admin | passw0rd |
| Db2 | bluadmin | bluadmin |
| RStudio | Admin<br><br>rstudio | passw0rd<br><br>rstudio |
| Jupyter notebooks | n/a | passw0rd |

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, RStudio and Jupyter.

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

**Get the latest scripts and data:**

To update the scripts, data and documentation you can download them from github:

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

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

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

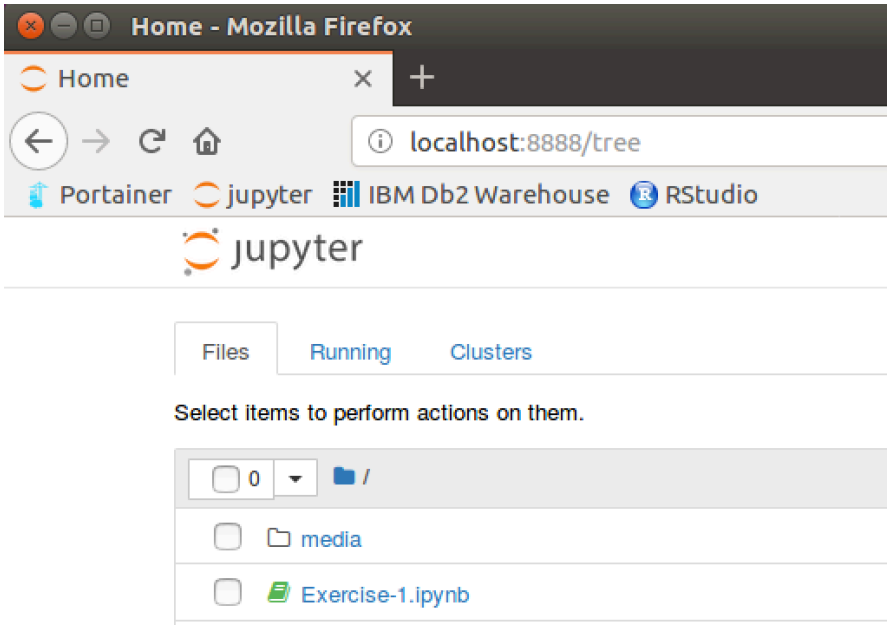
   ```
   drwxrwxr-x 11 ibmuser ibmuser 4096 Feb  8 18:39 ./
   drwxr-xr-x 23 ibmuser ibmuser 4096 Feb  8 10:57 ../
   drwxrwxr-x  8 ibmuser ibmuser 4096 Sep  6 11:26 .git/
   drwxrwxr-x  2 ibmuser ibmuser 4096 Feb  6 07:36 .ipynb_checkpoints/
   drwxrwxr-x  2 ibmuser ibmuser 4096 Sep  6 11:26 Documents/
   drwxrwxr-x  3 ibmuser ibmuser 4096 Feb  6 21:15 NYPD/
   -rw-rw-r--  1 ibmuser ibmuser  216 Aug 29 14:53 README.md
   drwxrwxr-x  5 ibmuser ibmuser 4096 Aug 29 14:53 SSB/
   drwxrwxr-x  4 ibmuser ibmuser 4096 Sep 11 16:12 linear-regression/
   drwxrwxr-x  4 ibmuser ibmuser 4096 Feb  8 18:03 notebooks/
   drwxrwxr-x  4 ibmuser ibmuser 4096 Aug 29 14:53 spark/
   drwxrwxr-x  2 ibmuser ibmuser 4096 Aug 29 14:53 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: Creating Tables

Since in the Db2 Warehouse docker container the instance and database is already deployed we can start immediately with creating tables and load data. The name of the database is *BLUDB*.

In this exercise, you will learn several methods how to create tables. To continue with the documentation select *Exercise1.ipynb* in Jupyter Notebook.




Come back here when you finished the exercises in *Exercise1.ipynb.*

## Lab 2: Loading Data

In this exercise, we will load data into the tables which we created in the previous lab. There are several methods for loading data:

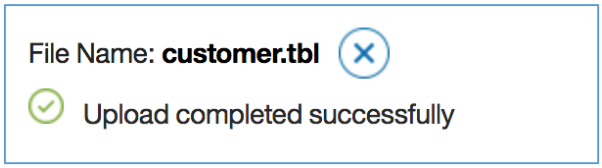
- load from file
- load from S3 and Swift
- load geospatial data
- load open data (publicly available data sets)

If you have data already stored in the cloud, you can also load these data directly which is very fast. In this workshop we load from our local virtual machine.

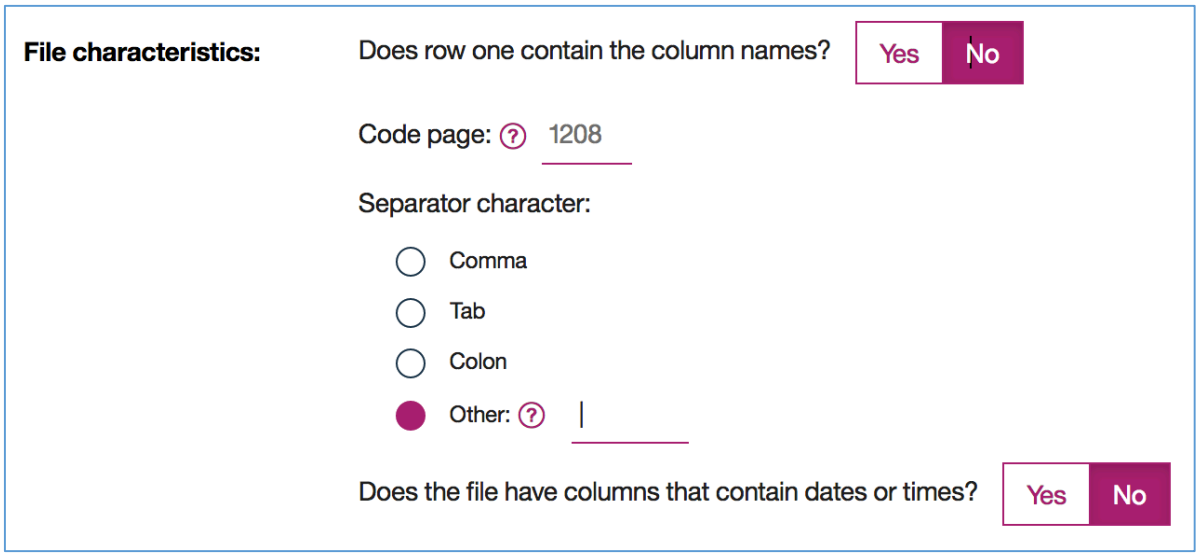
There are two data sets prepared in our virtual machine, one with 1GB of data, the other with 100MB. Because we probably have a slow internet connection we will use the smaller data set.

1. The first step for loading is selecting “Load” from the menu, and then selecting “Load from File”. Select load from client and “Choose a file”

2. In the file browser select the file “customer.tbl” in the directory
   `/home/ibmuser/Advanced-Analytics/SSB/loadfiles`
   Immediately after selecting the file it will be uploaded to the database server. In this environment it will take just a few seconds.




3. 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 “Finish” in the last step.

4. 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):

**Quick Stats:**

Number of rows committed = 3000
Number of rows deleted = 0
Number of rows loaded = 3000
Number of rows read = 3000
Number of rows rejected = 0
Number of rows skipped = 0

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

5. 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 chapter.

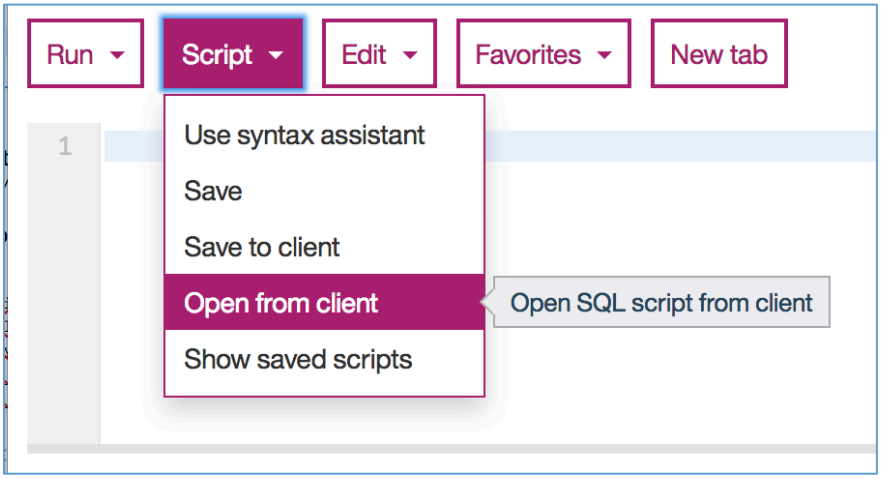
## Lab 3: Executing Queries

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” in the menu .

1. 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.

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

   Open the script with the function “Open from client”:

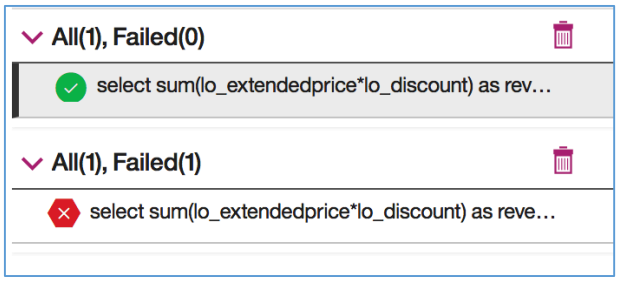



   Let's look at query q1_1a.sql:

```
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;
```

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

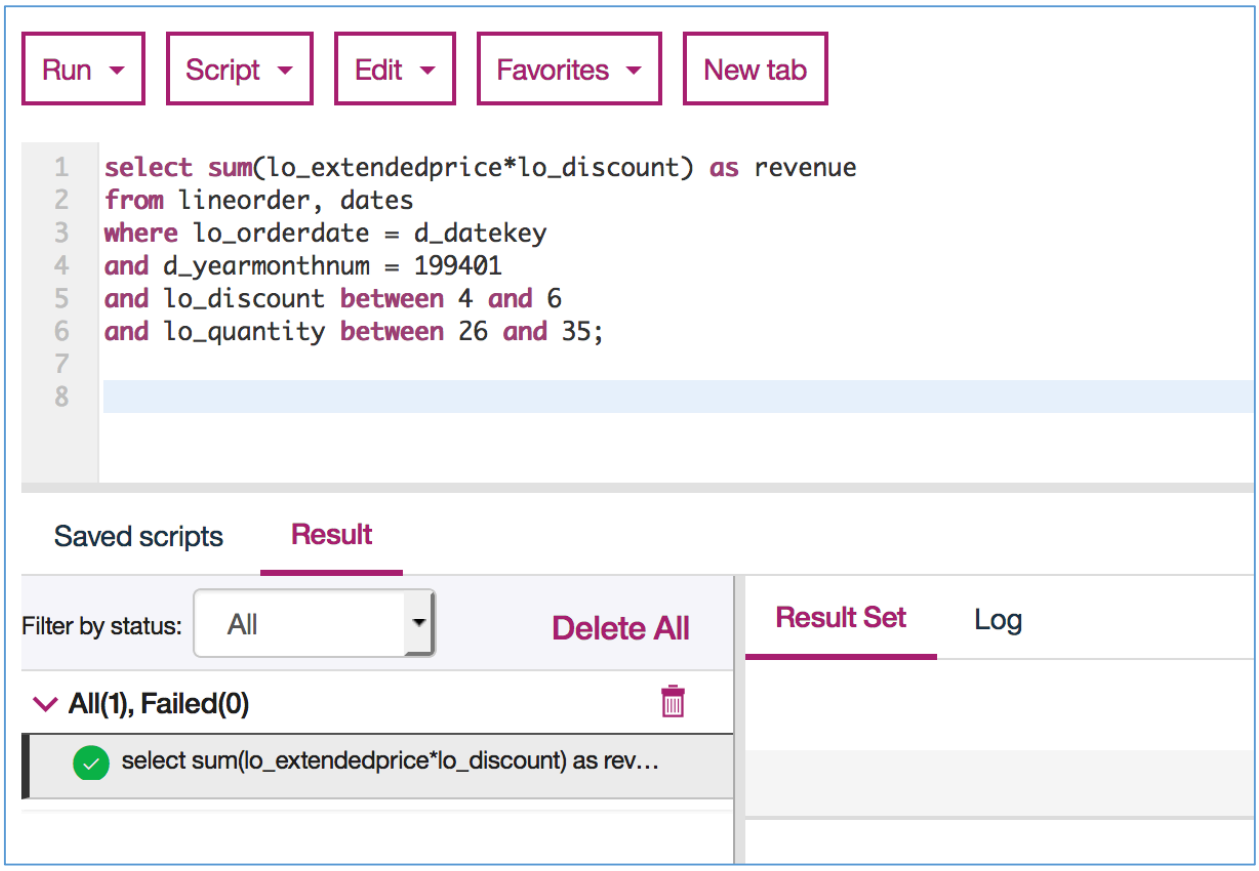
4. When you scroll down, you will also see the result (please note that the value corresponds to a 1GB data set. Therefore, your value may vary):




5. Now you can save the modified query by clicking the icon “Script” and “Save to client” and clicking “OK”. Please note that your workplace is in the cloud (or docker container) 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.

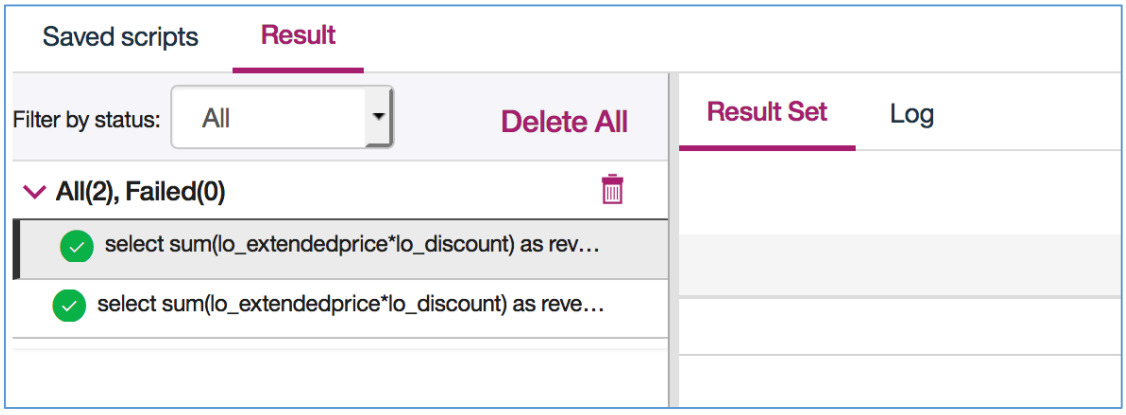
6. Let's run the next SQL statement by loading it from the local file system on our VMware. Let's use query q1_2.sql for this. Select “Script” and “Open from client”. Open a script from your local machine means to upload the file to the cloud.

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



8. Now run all the remaining queries in `/home/ibmuser/Advanced-Analytics/SSB/queries` with the methods which you learned.

9. 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 4: Linear Regression in SQL

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/ibmuser/Advanced-Analytics/linear-regression`

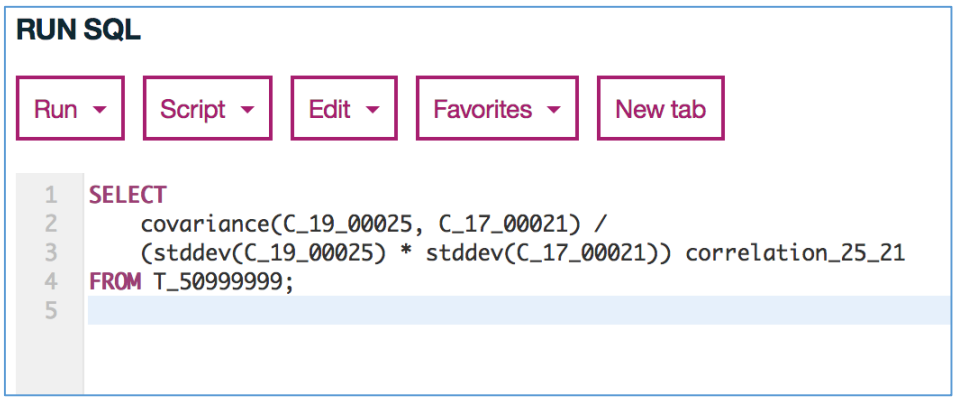
10. First the data have to be loaded in a table. For this, open the Db2 console. Next create a table T_50999999 like in “create-table-50999999.sql.

11. Load data from the file “lf_50999999_2018.unl” into this table. Select “Load” and “My computer” for this.
    Select the load file (located in `/home/ibmuser/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:SS” as format for columns which contain both dates and times. Finally click on “Preview”:

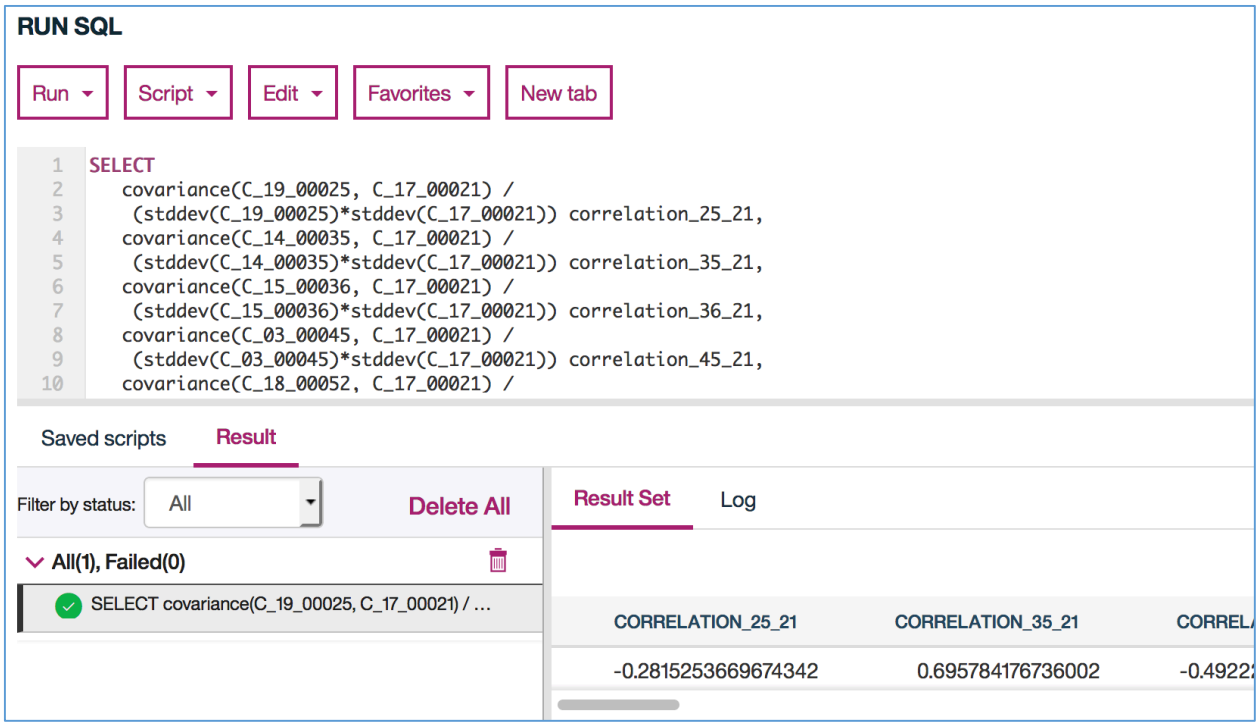
12. At next we want to find out whether any columns in our data set are correlated. We will try to do this with standard SQL. We will use the Pearson's correlation coefficient (https://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient) for determining whether two columns are correlated.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

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

13. 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.



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




15. Which columns are correlated most?

_____

16. 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/ibmuser/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.

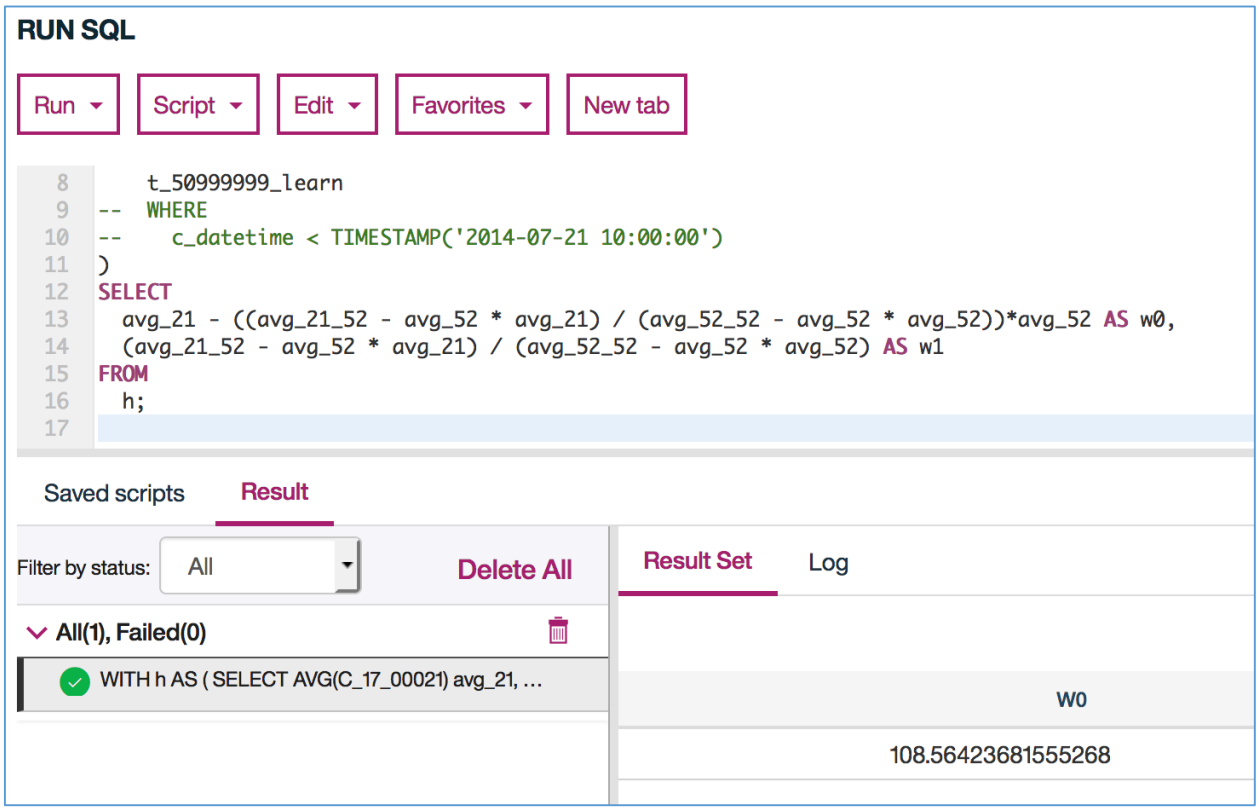
We will be doing this partitioning by defining two views *t_50999999_learn* and *t_50999999_test*. In the directory: `/home/ibmuser/Advanced-Analytics/linear-regression` there is already a script ***create-views.sql*** which creates these two views. Deploy both views.

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:

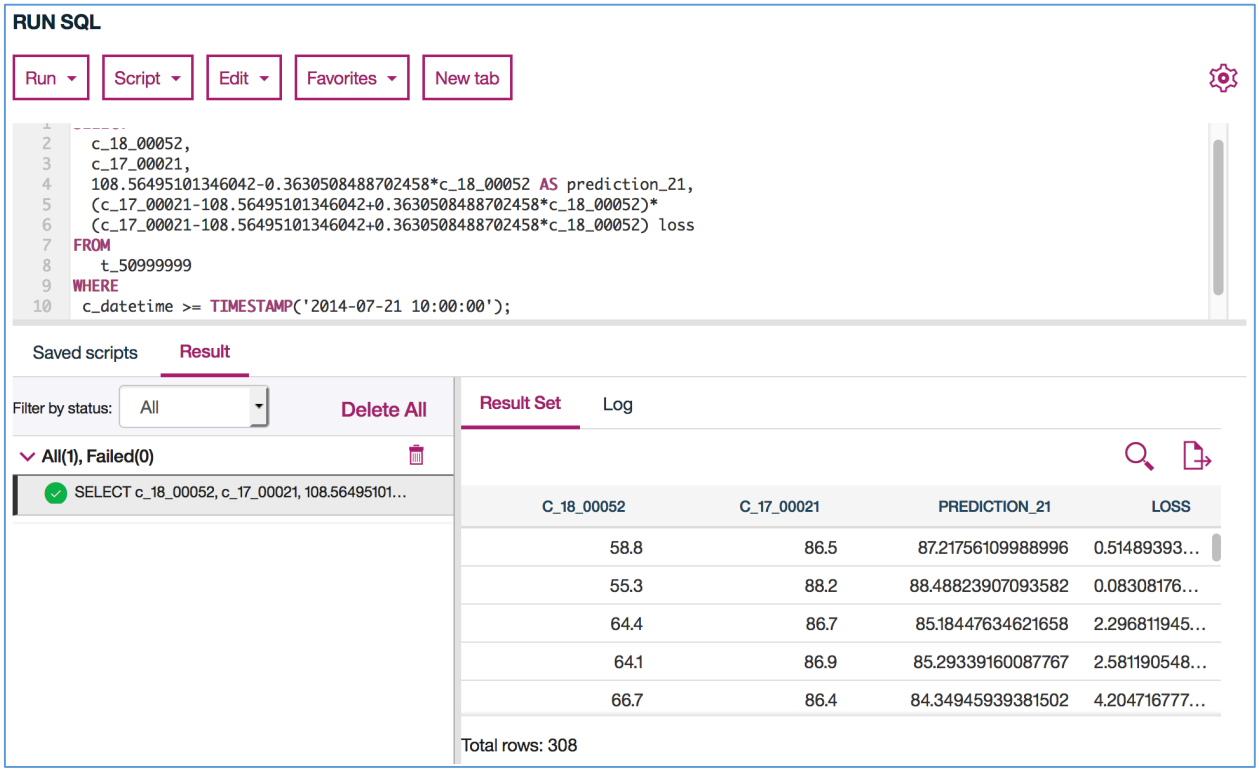


17. Now we can use these parameters to predict the air temperature for the second part of the data set. Write an SQL statement for this and execute it.

    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;
```

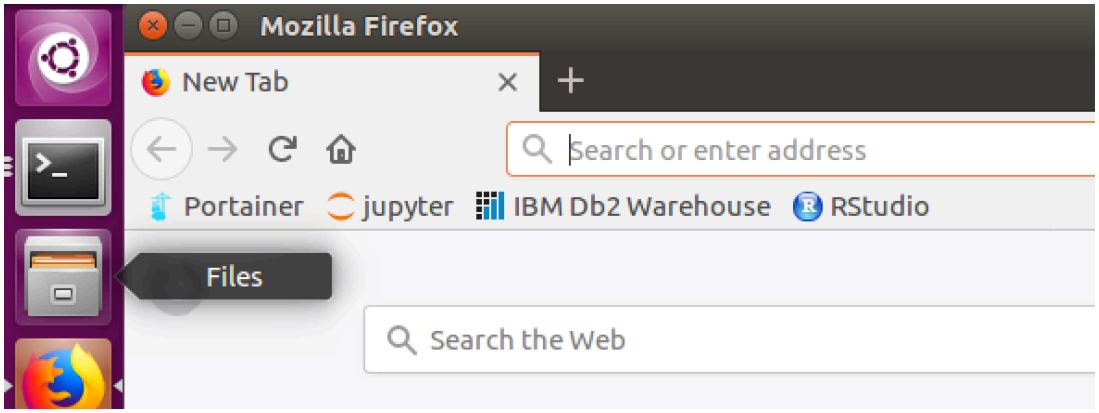
18. The file predict.sql contains this script. Run the script. The output should look like this:



## Lab 5: Linear Regression with R

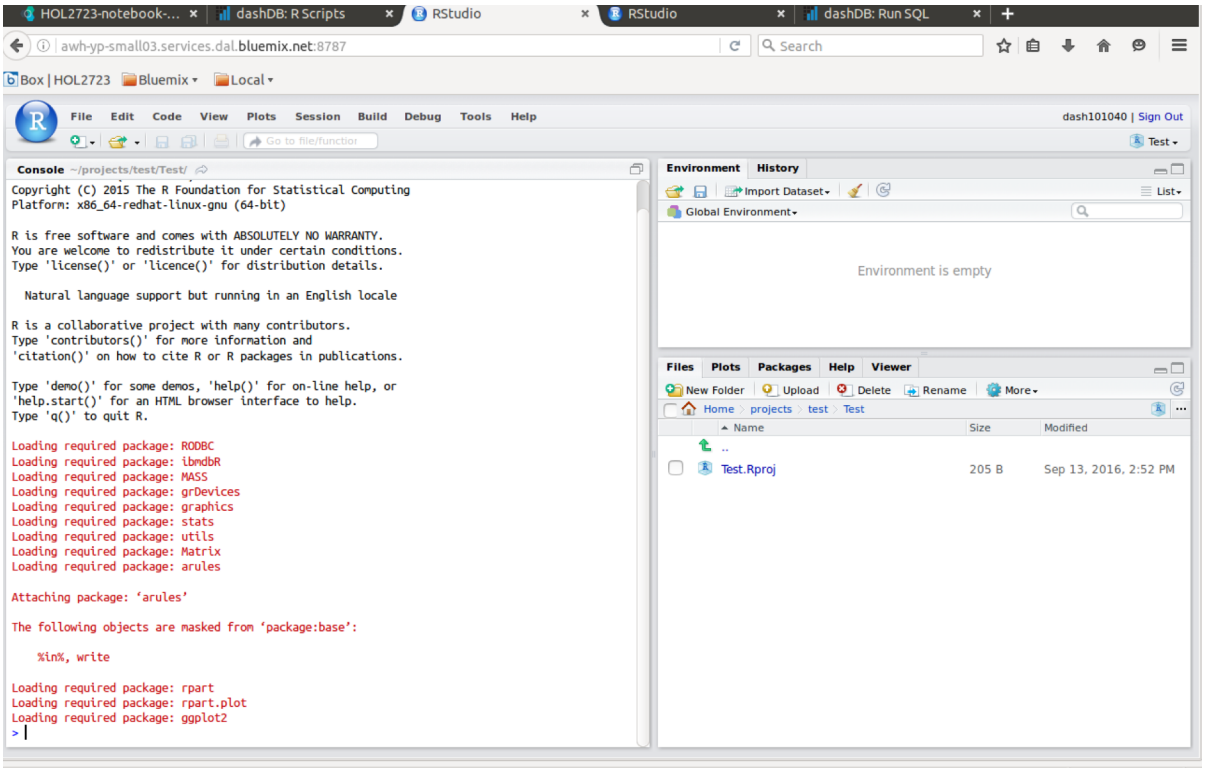
In this lab we want to use R and in-database analytics for linear regression. We will use RStudio for submitting the R code.

Start RStudio in your browser. There is already a bookmark in the “Bookmarks Toolbar”




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

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




We will use the console window for typing commands.

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

```
library(ibmdbR)
host.name <- "172.17.0.1"
user.name <-"bluadmin"
pwd <- "bluadmin"

con <- idaConnect(paste( "DASHDB",
                         ";Database=BLUDB;
                         Hostname=",host.name,";
                         Port=50000;
                         PROTOCOL=TCPIP;
                         UID=", user.name,";
                         PWD=",pwd,sep=""),"","" )

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](https://cran.r-project.org/web/packages/ibmdbR/ibmdbR.pdf)

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

```
idaInit(con)
```

5. 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()
```

6. 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
```

7. 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")
```

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

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

9. 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.

10. 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)
    ```

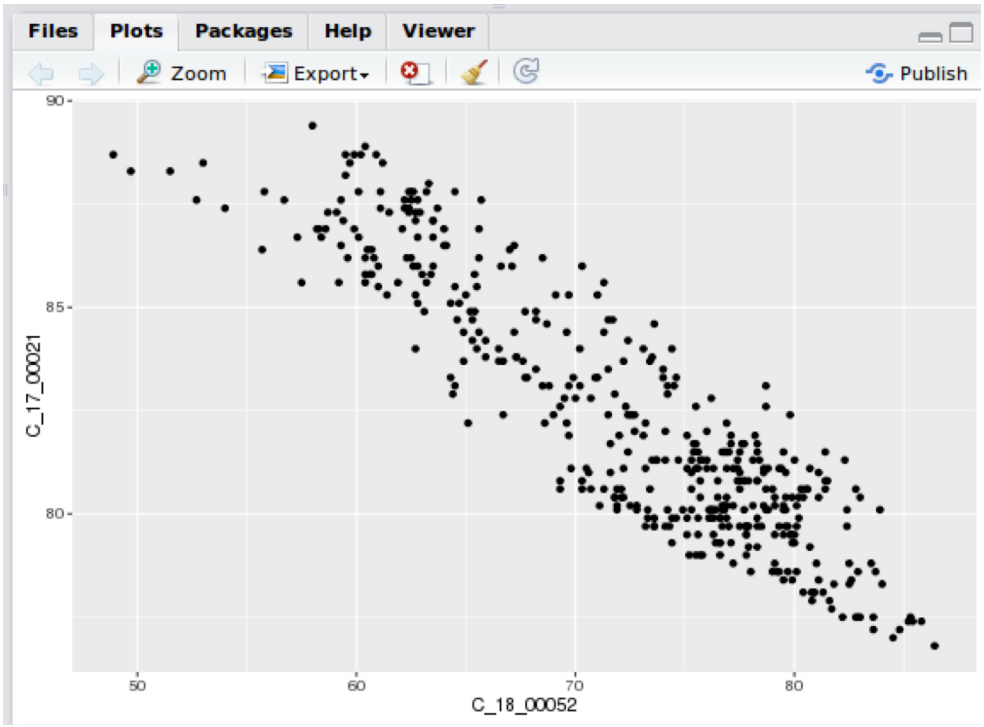
11. We have to convert the ida data frame to a regular R data frame first, since the plotting functions don't work on ida data frames directly. We do this with the following command and assign the data frame to a variable l:

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

12. The plotting is performed in this way:

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

13. We get the following plot in RStudio:




14. 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)
    ```

15. 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
```

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 6: Linear Regression with Python

In this chapter, 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.

1. When logged in as user *ibmuser*, open a terminal window. In the terminal window change the directory to `/home/ibmuser/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.
>>>
```

2. 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
```

3. 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 0.

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

4. Now we connect to the Db2 database:

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

5. 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
```

6. 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 <schema>.T_50999999_LEARN")
```

   Please note that the correct scheme is specified.

7. 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())
```

8. 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
```

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

```
>>> df.values.shape

(424, 2)
```

10. 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
```

...

11. 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)
```

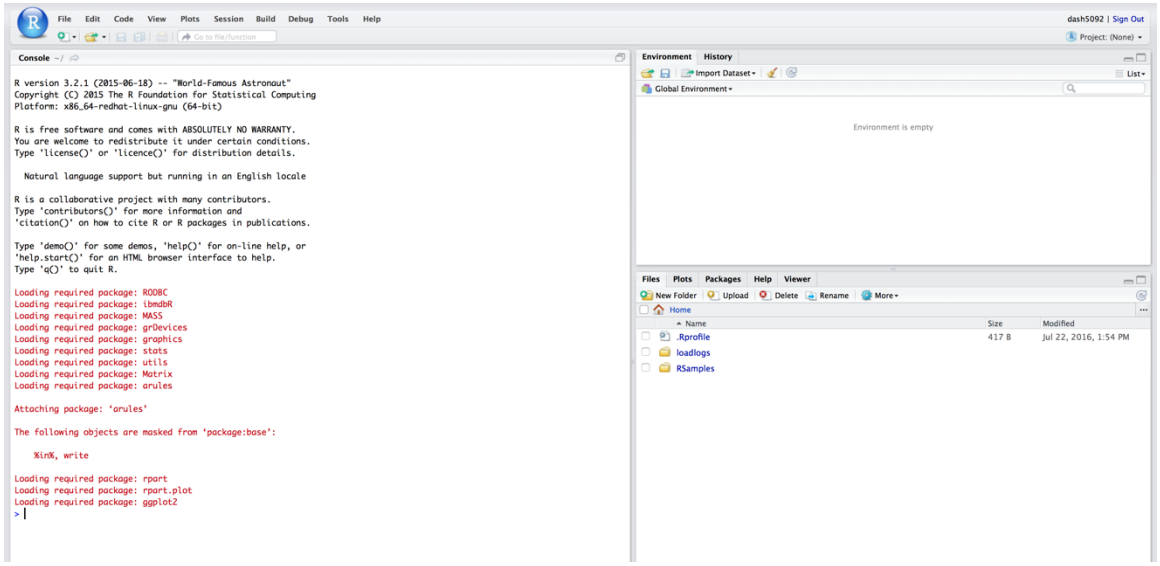
12. We can print the parameter of our model with intercept_ and coef_. Obviously, we are getting the same values as in the two previous labs:

```
>>> print(linReg.intercept_)
108.564951013
>>> print(linReg.coef_)
```

[-0.36305085]

# Lab 7: Analyzing data with R in RStudio

1. 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:

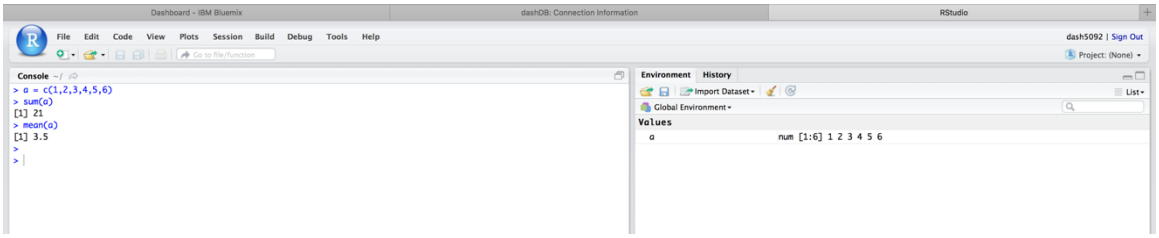



2. The RStudio interface starts with 3 areas. In the Console area you can interactively run R code. The Environment area in the top right shows the current variables and is empty at the beginning. The area in the lower right is showing the File tab. It shows the local storage that the Db2 services offers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

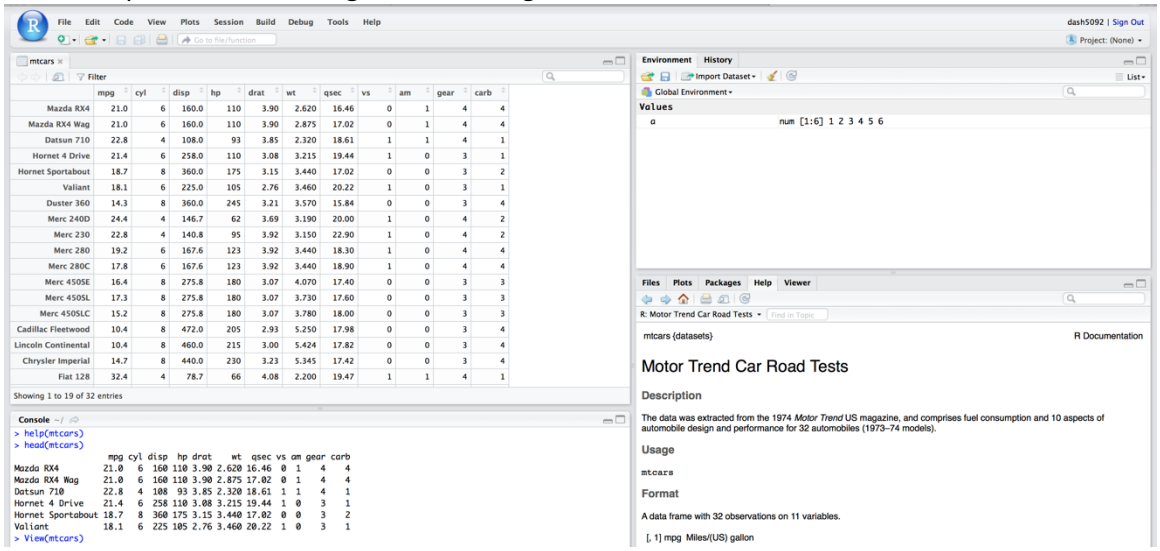
   Create a variable with a list of values and do simple operations on the variable:

   ```
   > 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.

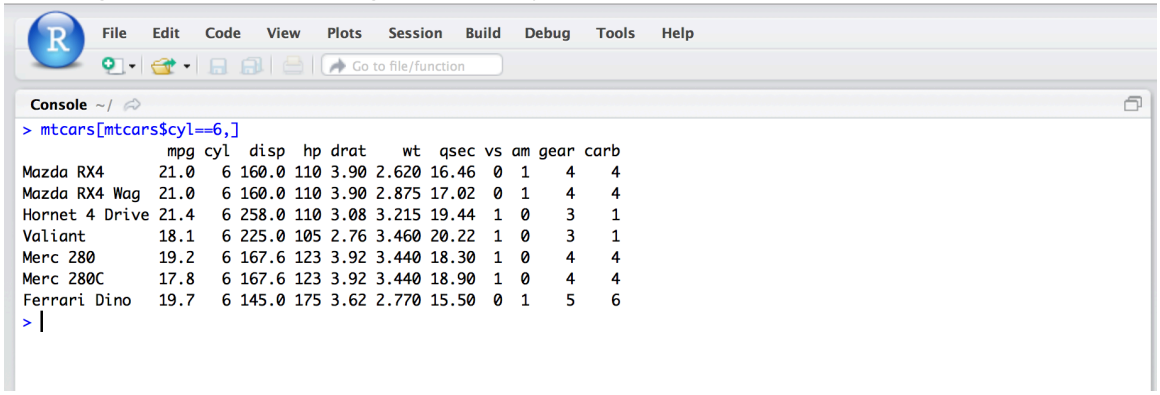
3. A convenient way to represent a table in R is a data frame. The sample data frame mtcars could be used to see how a data frame is handled in R. Type help(mtcars) to get an explaination for the data stored in mtcars. The command head(mtcars) gives you the first lines of the data frame. The View function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

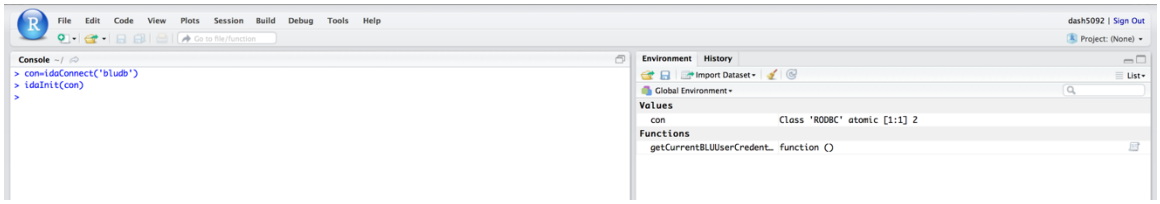
This filter selects the rows from mtcars where the value of column cyl is equal to 6, meaning selecting the cars that have engines with 6 cylinders.



4. 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:

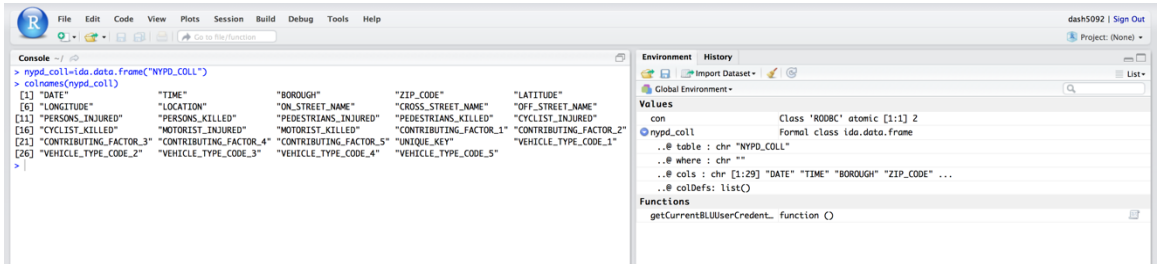
   ```
   > con=idaConnect('bludb')
   > idaInit(con)
   ```

   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.

5. 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)
   ```

6. 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)
   ```

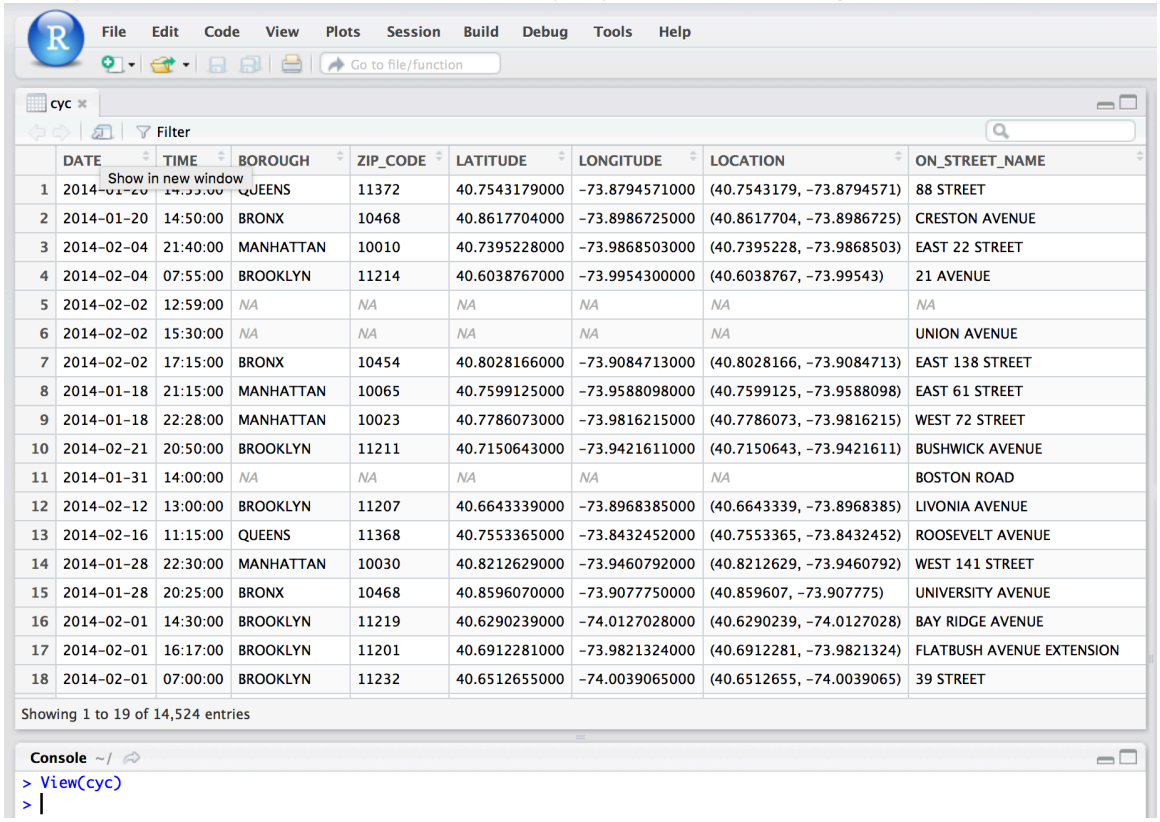
7. You can also define queries to Db2 in R syntax. If you run the following command, it will return the generated SQL query:

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

8. 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)
```

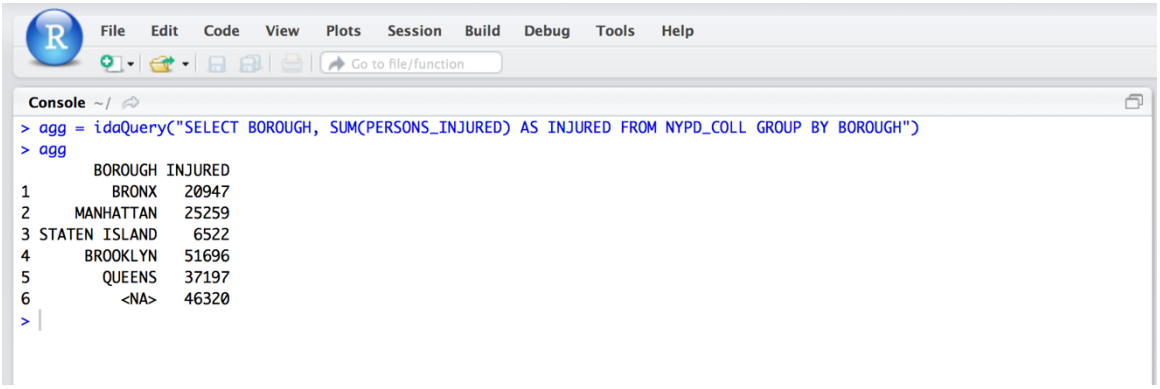
9. To look at the data in RStudio, the View function could be used on any ida data frame. It retrieves a sample of the rows fromt the Db2 table (or query) and shows it in a grid:




| | 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 |

10. 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
```

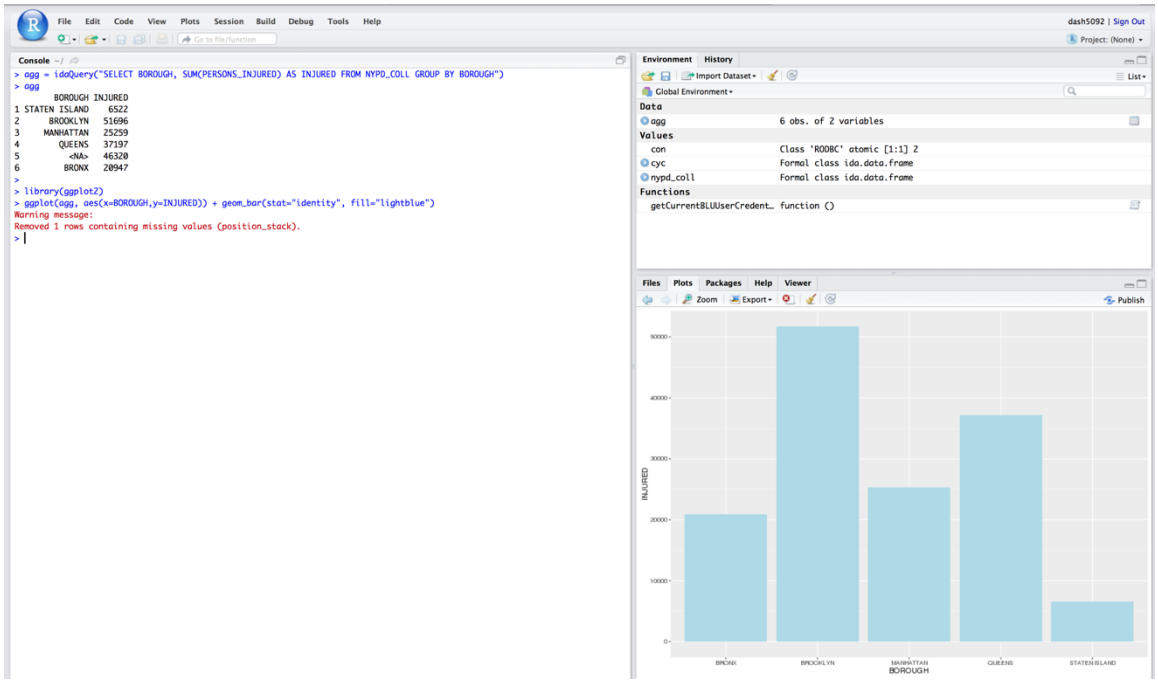



It is advisable to use idaQuery only for queries that return not too many rows, because the complete result set is transferred to the R environment.

11. 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")
```



12. 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.

13. 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.

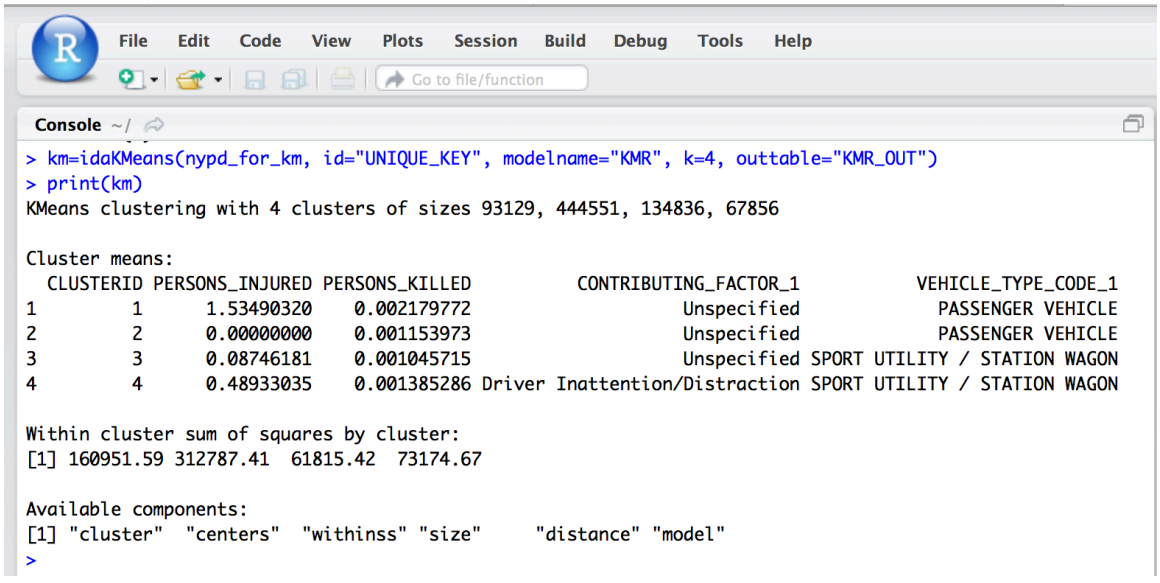
The clustering should be executed on only a subset of the columns. Therefore a new data frame is created that select just the columns needed:

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

14. 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")
```

15. 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.



16. 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:

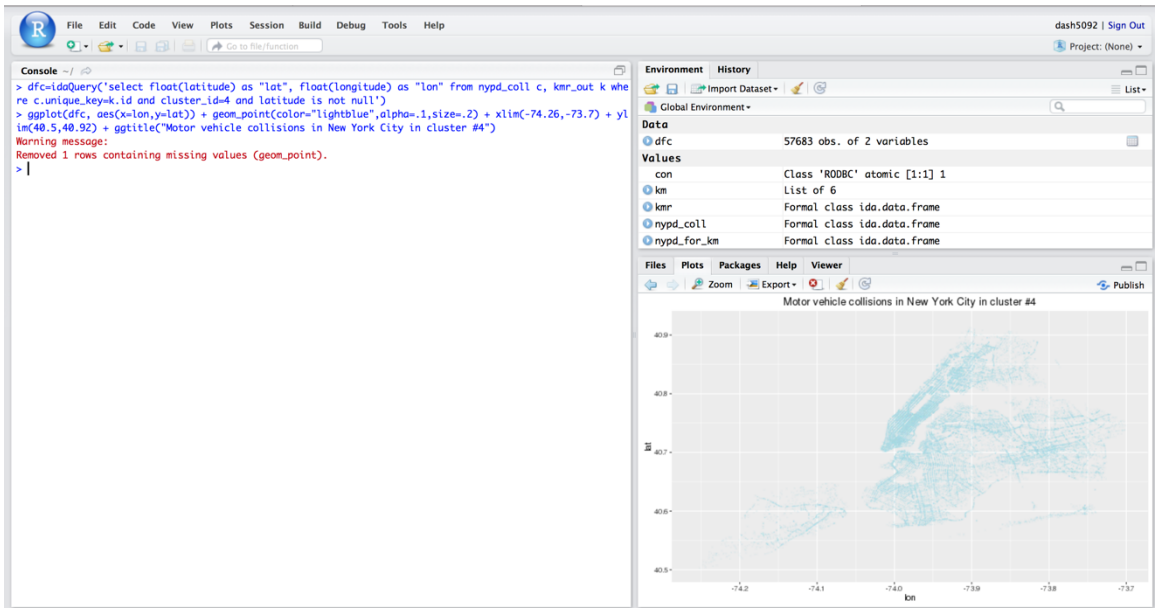
```
> kmr=ida.data.frame("KMR_OUT")
> View(kmr)
```

17. 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')
```

18. 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.

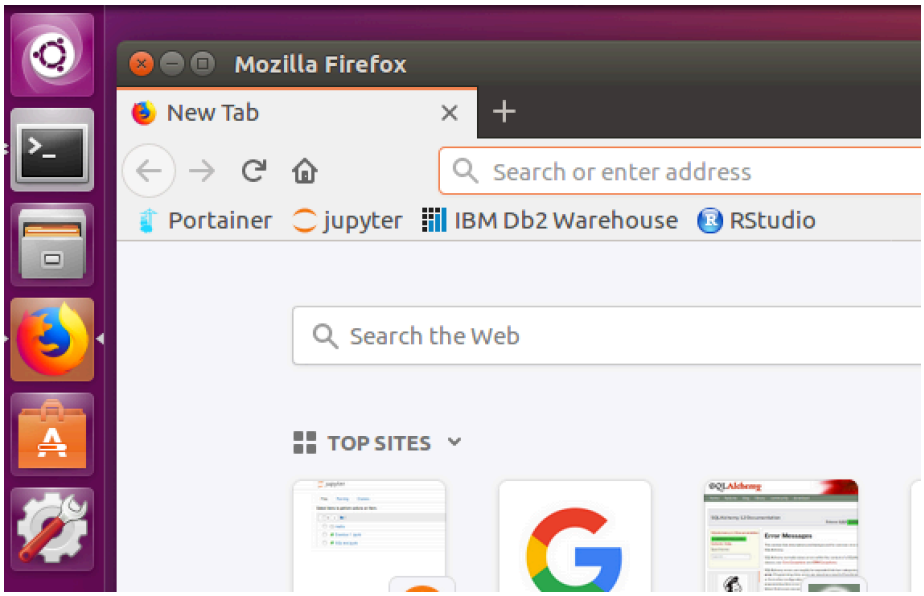
## Appendix I: Start vmware image and docker container

1. Start the image ***Db2-Local*** in vmware workstation
   The user you are working with is
   name: `ibmuser`
   password: `passw0rd`

2. Open a terminal window to start the docker container for *Db2*, *RStudio* and *Portainer*. The commands are:
   ```
   docker start Db2wh
   docker start RStudio
   docker start PT
   ```
   Start Db2 takes a few minutes. Take a short break and get a cup of coffee.
   Double check if all container are running with
   ```
   docker ps –a
   ```
   The output should look like this:
   ```
   ibmuser@linux:~/Advanced-Analytics$ docker ps -a
   CONTAINER ID        IMAGE                                      COMMAND             CREATED             STATUS              P
                       NAMES
   622a46248b04        rocker/rstudio                             "/init"             2 days ago          Up 10 hours         0
   .0:8787->8787/tcp   RStudio
   3b4c05be6558        portainer/portainer                        "/portainer"        2 days ago          Up 10 hours         0
   .0:9000->9000/tcp   PT
   9b21ca6569e3        store/ibmcorp/db2wh_ee:v2.4.0-db2wh-linux  "/usr/sbin/init"    2 days ago          Up 10 hours
                       Db2wh
   ibmuser@linux:~/Advanced-Analytics$
   ```

3. Start the Jupyter process. The commands is:
   ```
   nohup jupyter notebook &
   ```

4. In a browser open in separate tabs the Jupyter Notebook, Db2 Warehouse console and RStudio
   Jupyter Notebook: http://localhost:8888
   Db2 Warehouse: https://localhost:8443
   RStudio: http://localhost:8787



## We Value Your Feedback!

- Don’t forget to submit your Think 2018 session and speaker feedback! Your feedback is very important to us – we use it to continually improve the conference.
- Access the Think 2018 agenda tool to quickly submit your surveys from your smartphone, laptop or conference kiosk.