Lab-3

May 25, 2021

Hands-on Lab: Creating and Querying Database Objects from R

0.0.1 Welcome!

In this hands-on lab, we will create and query database objects from an R notebook in Jupyter, and use ggplot2 to plot the data using R libraries.

Tasks

Pre-requisites & Dataset

Load RODBC

Create a database connection

Create a connection string and connect to the database

View database and driver information

Create the tables

Load data into the database

Fetch data from the database

Plot the data (using ggplot2)

Dis-connect

Estimated Time Needed: 30 min

0.0.2 a. Pre-requisites & Dataset

Pre-requisite: In this lab we will use Jupyter Notebooks within SN Labs to access data in a Db2 on Cloud database using RODBC. Information about Jupyter notebooks, SN Labs, and Db2 services is provided in the previous labs.

Dataset used in this lab: For this lab we will utilize the Ontario public schools enrollment dataset. This data set is available under the Open Government License – Ontario and sourced from: https://www.ontario.ca/data/ontario-public-schools-enrolment

For simplicity we have already split it into two separate files: board.csv and school.csv.

Prior to starting the lab, ensure the data set files are present in the "/resources/data/samples/osb/" folder under My Data.

0.0.3 b. Load RODBC

The RODBC package and the ODBC driver for Db2 are pre-installed on your workbench. Let's load the RODBC package by clicking on the following cell and executing it (Shift+Enter):

[1]:
```
library(RODBC);
```

0.0.4 c. Create a database connection

[2]:
```
dsn_driver <- "{IBM DB2 ODBC Driver}"
dsn_database <- "BLUDB"
dsn_hostname <- "dashdb-txn-sbox-yp-dal09-08.services.dal.bluemix.net"
dsn_port <- "50000"
dsn_protocol <-"TCPIP"
dsn_uid <- "vgh86276"
dsn_pwd <- "5ft1hr51@rfbgt26"
```

IBM DB2 ODBC Driver

Click here to view/hide hint

```
# Fill in the <...>
dsn_driver <- "{...}"
dsn_database <- "..."
dsn_hostname <- "<Enter Hostname>"
dsn_port <- "..."
dsn_protocol <- "..."
dsn_uid <- "<Enter UserID>"
dsn_pwd <- "<Enter Password>"
```

Click here to view/hide solution

```
#Enter the values for you database connection
dsn_driver <- "{IBM DB2 ODBC Driver}"
dsn_database <- "BLUDB"            # e.g. "BLUDB"
dsn_hostname <- "<Enter Hostname>" # e.g.: "dashdb-txn-sbox-yp-dal09-11.services.dal.bluemix.ne
dsn_port <- "50000"                # e.g. "50000"
dsn_protocol <- "TCPIP"            # i.e. "TCPIP"
dsn_uid <- "<Enter UserID>"        # e.g. "zjh17769"
dsn_pwd <- "<Enter Password>"      # e.g. "zcwd4+8gbq9bm5k4"
```

0.0.5 d. Create a connection string and connect to the database

[3]:
```
conn_path <- paste("DRIVER=",dsn_driver,
                  ";DATABASE=",dsn_database,
                  ";HOSTNAME=",dsn_hostname,
                  ";PORT=",dsn_port,
                  ";PROTOCOL=",dsn_protocol,
                  ";UID=",dsn_uid,
                  ";PWD=",dsn_pwd,sep="")
```

```
conn <- odbcDriverConnect(conn_path)
conn
```

```
RODBC Connection 1
Details:
  case=nochange
  DRIVER={IBM DB2 ODBC DRIVER}
  UID=vgh86276
  PWD=******
  DATABASE=BLUDB
  HOSTNAME=dashdb-txn-sbox-yp-dal09-08.services.dal.bluemix.net
  PORT=50000
  PROTOCOL=TCPIP
```

Click here to view/hide hint

```
# Fill in the ...
conn_path <- paste("DRIVER=",...,
                  ";DATABASE=",...,
                  ";HOSTNAME=",...,
                  ";PORT=",...,
                  ";PROTOCOL=",...,
                  ";UID=",...,
                  ";PWD=",dsn_...,...="")
conn <- odbcDriverConnect(conn_path)
conn
```

Click here to view/hide solution

```
conn_path <- paste("DRIVER=",dsn_driver,
                  ";DATABASE=",dsn_database,
                  ";HOSTNAME=",dsn_hostname,
                  ";PORT=",dsn_port,
                  ";PROTOCOL=",dsn_protocol,
                  ";UID=",dsn_uid,
                  ";PWD=",dsn_pwd,sep="")
conn <- odbcDriverConnect(conn_path)
conn
```

0.0.6 e. View database and driver information

[4]:
```
sql.info <- sqlTypeInfo(conn)
conn.info <- odbcGetInfo(conn)
conn.info["DBMS_Name"]
conn.info["DBMS_Ver"]
conn.info["Driver_ODBC_Ver"]
```

DBMS\_Name: 'DB2/LINUXX8664'

DBMS_Ver: '11.01.0004'

Driver_ODBC_Ver: '03.51'

Click here to view/hide hint

```
# Fill in the ...
sql.... <- sql...(conn)
conn.... <- odbc...(conn)
conn....["..._Name"]
conn....["..._Ver"]
conn....["Driver_..._Ver"]
conn
```

Click here to view/hide solution

```
#View database and driver information
sql.info <- sqlTypeInfo(conn)
conn.info <- odbcGetInfo(conn)
conn.info["DBMS_Name"]
conn.info["DBMS_Ver"]
conn.info["Driver_ODBC_Ver"]
```

0.0.7 f. Create the tables

You will need to *remove* the BOARD and SCHOOL tables in case they already exist. **Note: Your Db2 non-system Schema name is your userID/username in uppercase used in creating database connection.**

[5]:
```
myschema <- "ZJH17769" # e.g. "ZJH17769"
tables <- c("BOARD", "SCHOOL")

    for (table in tables){
      # Drop School table if it already exists
      out <- sqlTables(conn, tableType = "TABLE", schema = myschema, tableName␣
 ↪=table)
      if (nrow(out)>0) {
        err <- sqlDrop (conn, paste(myschema,".",table,sep=""), errors=FALSE)
        if (err==-1){
          cat("An error has occurred.\n")
          err.msg <- odbcGetErrMsg(conn)
          for (error in err.msg) {
            cat(error,"\n")
          }
        } else {
          cat ("Table: ",  myschema,".",table," was dropped\n")
        }
      } else {
          cat ("Table: ",  myschema,".",table," does not exist\n")
      }
```

```
  }
```

```
Table:  ZJH17769 . BOARD  does not exist
Table:  ZJH17769 . SCHOOL  does not exist
```

Click here to view/hide hint

```
myschema <- "..."
tables <- c("...", "...L")

    for (table in ...){
      # Drop ... table if it already exists
      out <- sql...(conn, table... = "...", schema = my..., ...Name =table)
      if (nrow(...)>0) {
        err <- sql... (conn, paste(my...,".",...,...=""), errors=...)
        if (err==-1){
          ...("An error has occurred.\n")
          err.... <- odbc...Msg(conn)
          for (error in ....msg) {
            cat(...,"\n")
          }
        } else {
          cat ("...: ",  my...,".",table," was ...\n")
        }
      } else {
          cat ("...: ",  my...,".",table," does not ...\n")
      }
    }
```

Click here to view/hide solution

```
myschema <- "<Enter Schema>" # e.g. "ZJH17769"
tables <- c("BOARD", "SCHOOL")

    for (table in tables){
      # Drop School table if it already exists
      out <- sqlTables(conn, tableType = "TABLE", schema = myschema, tableName =table)
      if (nrow(out)>0) {
        err <- sqlDrop (conn, paste(myschema,".",table,sep=""), errors=FALSE)
        if (err==-1){
          cat("An error has occurred.\n")
          err.msg <- odbcGetErrMsg(conn)
          for (error in err.msg) {
            cat(error,"\n")
          }
        } else {
          cat ("Table: ",  myschema,".",table," was dropped\n")
        }
      } else {
        cat ("Table: ",  myschema,".",table," does not exist\n")
      }
    }
```

Let's create the BOARD table in the database.

```
[6]: df1 <- sqlQuery(conn, "CREATE TABLE BOARD (
                             B_ID CHAR(6) NOT NULL,
                             B_NAME VARCHAR(75) NOT NULL,
                             TYPE VARCHAR(50) NOT NULL,
                             LANGUAGE VARCHAR(50),
                             PRIMARY KEY (B_ID))",
                     errors=FALSE)
```

Click here to view/hide hint

```
# Fill in the ...
df1 <- sql...(conn, "CREATE ... BOARD (
                             B_ID ...(6) NOT ...,
                             B_NAME ...(75) NOT ...,
                             TYPE ...(50) NOT ...,
                             LANGUAGE ...(50),
                             ... KEY (B_ID))",
                ...=FALSE)
```

Click here to view/hide solution

```
df1 <- sqlQuery(conn, "CREATE TABLE BOARD (
                             B_ID CHAR(6) NOT NULL,
                             B_NAME VARCHAR(75) NOT NULL,
                             TYPE VARCHAR(50) NOT NULL,
                             LANGUAGE VARCHAR(50),
                             PRIMARY KEY (B_ID))",
                errors=FALSE)
```

Check if successful

```
[8]: if (df1 == -1){
       cat ("An error has occurred.\n")
       msg <- odbcGetErrMsg(conn)
       print (msg)
     } else {
       cat ("Table was created successfully.\n")
     }
```

```
Table was created successfully.
```

Click here to view/hide hint

```
# Fill in the ...
if (... == -1){
  cat ("An ... has occurred.\n")
  msg <- odbc...Msg(conn)
  print (...)
} else {
  cat ("Table was ... ....\n")
}
```

Click here to view/hide solution

```
if (df1 == -1){
  cat ("An error has occurred.\n")
  msg <- odbcGetErrMsg(conn)
  print (msg)
} else {
  cat ("Table was created successfully.\n")
}
```

Now let's create the SCHOOL table.

[9]:
```
df2 <- sqlQuery(conn, "CREATE TABLE SCHOOL (
                  B_ID CHAR(6) NOT NULL,
                  S_ID CHAR(6) NOT NULL,
                  S_NAME VARCHAR(100),
                  LEVEL VARCHAR(70),
                  ENROLLMENT INTEGER WITH DEFAULT 10,
                  PRIMARY KEY (B_ID, S_ID))", errors=FALSE)
```

Click here to view/hide hint

```
# Fill in the ...
df2 <- sql...(conn, "CREATE ... SCHOOL (
                  B_ID ...(6) NOT NULL,
                  S_ID ...(6) NOT NULL,
                  S_NAME ...(100),
                  LEVEL ...(70),
                  ENROLLMENT ... WITH ... 10,
                  PRIMARY ... (B_ID, S_ID))", ...=FALSE)
```

Click here to view/hide solution

```
df2 <- sqlQuery(conn, "CREATE TABLE SCHOOL (
                  B_ID CHAR(6) NOT NULL,
                  S_ID CHAR(6) NOT NULL,
                  S_NAME VARCHAR(100),
                  LEVEL VARCHAR(70),
                  ENROLLMENT INTEGER WITH DEFAULT 10,
                  PRIMARY KEY (B_ID, S_ID))", errors=FALSE)
```

Check if successful.

[10]:
```
if (df2 == -1){
  cat ("An error has occurred.\n")
  msg <- odbcGetErrMsg(conn)
  print (msg)
} else {
  cat ("Table was created successfully.\n")
}
```

```
Table was created successfully.
```

Click here to view/hide hint

```
# Fill in the ...
if (... == -1){
  cat ("An ... has occurred.\n")
  msg <- odbc...Msg(conn)
  print (...)
} else {
  cat ("Table was ... ....\n")
}
```

Click here to view/hide solution

```
if (df2 == -1){
  cat ("An error has occurred.\n")
  msg <- odbcGetErrMsg(conn)
  print (msg)
} else {
  cat ("Table was created successfully.\n")
}
```

0.0.8 g. Load the data into the database

Fetch the tables present in the current database schema.

[11]:
```
tab.frame <- sqlTables(conn, schema=myschema)
nrow(tab.frame)
tab.frame$TABLE_NAME
```

0

Click here to view/hide hint

```
# Fill in the ...
tab.... <- sql...(conn, ...=myschema)
n...(tab....)
tab....$..._NAME
```

Click here to view/hide solution

```
tab.frame <- sqlTables(conn, schema=myschema)
nrow(tab.frame)
tab.frame$TABLE_NAME
```

Print column 4, 6, 7, 18 details for the tables BOARD and SCHOOL

[12]:
```
for (table in tables){
        cat ("\nColumn info for table", table, ":\n")
        col.detail <- sqlColumns(conn, table)
        print(col.detail[c(4,6,7,18)], row.names=FALSE)
}
```

```
Column info for table BOARD :
 COLUMN_NAME TYPE_NAME COLUMN_SIZE IS_NULLABLE
        B_ID      CHAR           6          NO
      B_NAME   VARCHAR          75          NO
        TYPE   VARCHAR          50          NO
    LANGUAGE   VARCHAR          50         YES

Column info for table SCHOOL :
 COLUMN_NAME TYPE_NAME COLUMN_SIZE IS_NULLABLE
        B_ID      CHAR           6          NO
        S_ID      CHAR           6          NO
      S_NAME   VARCHAR         100         YES
       LEVEL   VARCHAR          70         YES
  ENROLLMENT   INTEGER          10         YES
```

Click here to view/hide hint

```
# Fill in the ...
for (table in ...){
        cat ("\n... info for table", ..., ":\n")
        col.detail <- sqlColumns(conn, ...)
        print(col....[c(4,6,7,18)], row....=FALSE)
}
```

Click here to view/hide solution

```
for (table in tables){
        cat ("\nColumn info for table", table, ":\n")
        col.detail <- sqlColumns(conn, table)
        print(col.detail[c(4,6,7,18)], row.names=FALSE)
}
```

Load the data from the board.csv into the BOARD dataframe.

[15]:
```
boarddf <- read.csv("board.csv", header = FALSE)
```

Click here to view/hide hint

```
# Fill in the ...
board... <- ....csv("/resources/.../samples/.../board.csv", header = ...)
```

Click here to view/hide solution

```
boarddf <- read.csv("/resources/data/samples/osb/board.csv", header = FALSE)
```

Display initial data from the BOARD dataframe.

```
[16]: head(boarddf)
```

A data.frame: 6 × 4

	V1	V2	V3	V4
	<fct>	<fct>	<fct>	<fct>
1	B28010	Algoma DSB	Public	English
2	B67202	Algonquin and Lakeshore CDSB	Roman Catholic	English
3	B66010	Avon Maitland DSB	Public	English
4	B66001	Bluewater DSB	Public	English
5	B67164	Brant Haldimand Norfolk CDSB	Roman Catholic	English
6	B67008	Bruce-Grey CDSB	Roman Catholic	English

Click here to view/hide hint

```
# Fill in the ...
...(board...)
```

Click here to view/hide solution

```
head(boarddf)
```

Save the dataframe to the database table BOARD.

```
[17]: sqlSave(conn, boarddf, "BOARD", append=TRUE, fast=FALSE, rownames=FALSE,␣
      ↪colnames=FALSE, verbose=FALSE)
```

Click here to view/hide hint

```
# Fill in the ...
sql...(conn, ...df, "BOARD", ...=TRUE, ...=FALSE, row...=FALSE, col...=FALSE)
```

Click here to view/hide solution

```
sqlSave(conn, boarddf, "BOARD", append=TRUE, fast=FALSE, rownames=FALSE, colnames=FALSE, verbos
```

Load the data from the school.csv into the SCHOOL dataframe

```
[18]: schooldf <- read.csv("school.csv", header = FALSE)
```

Click here to view/hide hint

```
# Fill in the ...
school... <- ....csv("/resources/.../samples/osb/....csv", ... = FALSE)
```

Click here to view/hide solution

```
schooldf <- read.csv("/resources/data/samples/osb/school.csv", header = FALSE)
```

Display some records from the beginning of the SCHOOL dataframe.

```
[19]: head(schooldf)
```

A data.frame: 6 × 5

	V1	V2	V3	V4	V5
	<fct>	<int>	<fct>	<fct>	<int>
1	B28010	891240	Alexander Henry HS	Secondary	145
2	B28010	902344	Algoma Education Connection SS	Secondary	385
3	B28010	19186	Anna McCrea PS	Elementary	177
4	B28010	67679	Arthur Henderson PS	Elementary	104
5	B28010	28932	Aweres PS	Elementary	95
6	B28010	43362	Ben R McMullin PS	Elementary	241

Click here to view/hide hint

```
# Fill in the ...
...(school...)
```

Click here to view/hide solution

```
head(schooldf)
```

Change the encoding of the 3rd column character vector from latin1 to ASCII//TRANSLIT

```
[20]: schooldf$V3 <- iconv(schooldf$V3, "latin1", "ASCII//TRANSLIT")
```

Click here to view/hide hint

```
# Fill in the ...
...df$V3 <- i...(...df$V3, "latin1", "ASCII//TRANSLIT")
```

Click here to view/hide solution

```
schooldf$V3 <- iconv(schooldf$V3, "latin1", "ASCII//TRANSLIT")
```

Save the dataframe to the database table SCHOOL.

```
[21]: sqlSave(conn, schooldf, "SCHOOL", append=TRUE, fast=FALSE, rownames=FALSE,␣
       ↪colnames=FALSE, verbose=FALSE)
```

Click here to view/hide hint

```
# Fill in the ...
sql...(conn, school..., "SCHOOL", append=..., fast=..., row...=FALSE, col...=FALSE)
```

Click here to view/hide solution

```
#NOTE: This may take a long time because of the large size of the database. When there is lot

sqlSave(conn, schooldf, "SCHOOL", append=TRUE, fast=FALSE, rownames=FALSE, colnames=FALSE, ver
```

0.0.9 h. Fetch data from the database

Fetch the data from the database table BOARD and display some rows from the end of the data.

```
[25]: boarddb <- sqlFetch(conn, "BOARD")
      tail(boarddb)
```

A data.frame: 6 × 4

	B_ID	B_NAME	TYPE	LANGUAGE
	<fct>	<fct>	<fct>	<fct>
71	B67148	Waterloo CDSB	Roman Catholic	English
72	B66176	Waterloo Region DSB	Public	English
73	B67130	Wellington CDSB	Roman Catholic	English
74	B67024	Windsor-Essex CDSB	Roman Catholic	English
75	B67075	York CDSB	Roman Catholic	English
76	B66095	York Region DSB	Public	English

Click here to view/hide hint

```
# Fill in the ...
...db <- sql...(conn, "...")
...(board...)
```

Click here to view/hide solution

```
boarddb <- sqlFetch(conn, "BOARD")
tail(boarddb)
```

Fetch the data from the database table SCHOOL and and display some rows from the end of the data.

[26]:
```
schooldb <- sqlFetch(conn, "SCHOOL")
tail(schooldb)
```

A data.frame: 6 × 5

	B_ID	S_ID	S_NAME	LEVEL	ENROLLMENT
	<fct>	<int>	<fct>	<fct>	<int>
4894	B66095	634565	Windham Ridge PS	Elementary	846
4895	B66095	549380	Wismer PS	Elementary	511
4896	B66095	954292	Woodbridge College	Secondary	985
4897	B66095	617318	Woodbridge PS	Elementary	504
4898	B66095	618896	Woodland PS	Elementary	442
4899	B66095	624101	Yorkhill E S	Elementary	293

Click here to view/hide hint

```
# Fill in the ...
...db <- sql...(conn, "...")
tail(...db)
```

Click here to view/hide solution

```
schooldb <- sqlFetch(conn, "SCHOOL")
tail(schooldb)
```

0.0.10 i. Plot the data (using ggplot2)

[27]:
```
library(ggplot2);
```

Click here to view/hide hint

```
# Fill in the ...
```

```
...(gg...2);
```

Click here to view/hide solution

```
library(ggplot2);
```

Get the elementary school data from the database from both tables in descending sequence.

```
[28]: elequery <- query <- paste("select s.enrollment as ENROLLMENT
      from school s, board b
      where b.b_name = 'Toronto DSB' and b.b_id=s.b_id
      and s.level = 'Elementary'
      order by enrollment desc")
```

Click here to view/hide hint

```
# Fill in the ...
...query <- ... <- paste("... s.enrollment as ...
from ... s, ... b
where b.b_... = 'Toronto DSB' and b.b_id=s....
and ....level = 'Elementary'
order by ... desc")
```

Click here to view/hide solution

```
elequery <- query <- paste("select s.enrollment as ENROLLMENT
from school s, board b
where b.b_name = 'Toronto DSB' and b.b_id=s.b_id
and s.level = 'Elementary'
order by enrollment desc")
```

create the elementary school dataframe.

```
[29]: eledf <- sqlQuery(conn, elequery)
      dim(eledf)
```

1. 476 2. 1

Click here to view/hide hint

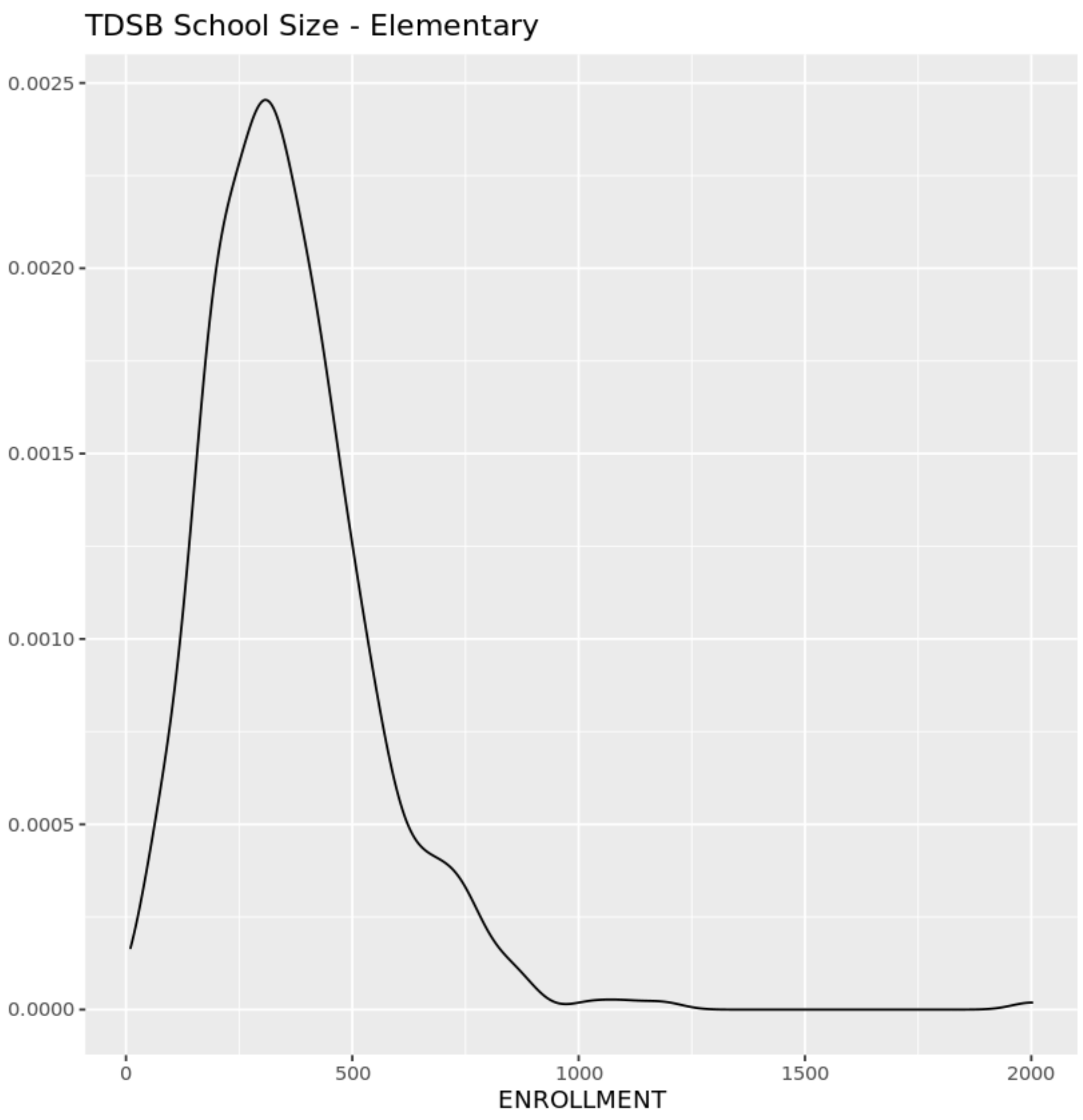
```
# Fill in the ...
ele... <- sql...(conn, ...query)
dim(...df)
```

Click here to view/hide solution

```
eledf <- sqlQuery(conn, elequery)
dim(eledf)
```

Create a density plot of elementary school enrollments.

```
[30]: qplot(ENROLLMENT, data=eledf, geom="density",  main="TDSB School Size -␣
       ↪Elementary")
```



Click here to view/hide hint

```
# Fill in the ...
q...(ENROLLMENT, ...=eledf, ...="density",  ...="TDSB School Size - ...")
```

Click here to view/hide solution

```
qplot(ENROLLMENT, data=eledf, geom="density",  main="TDSB School Size - Elementary")
```

Create the secondary school enrollments query in descending sequence.

[31]:
```
secquery <- paste("select s.enrollment as ENROLLMENT
from school s, board b
where b.b_name = 'Toronto DSB' and b.b_id=s.b_id
and s.level = 'Secondary'
order by enrollment desc")
```

Click here to view/hide hint

```
# Fill in the ...
sec... <- paste("... s.enrollment as ...
from ... s, board b
where b.b_... = 'Toronto ...' and b.b_id=s....
and s.... = 'Secondary'
order by ... desc")
```

Click here to view/hide solution

```
secquery <- paste("select s.enrollment as ENROLLMENT
from school s, board b
where b.b_name = 'Toronto DSB' and b.b_id=s.b_id
and s.level = 'Secondary'
order by enrollment desc")
```

Create the dataframe using the data in the database.

```
[32]: secdf <- sqlQuery(conn, secquery)
```

Click here to view/hide hint

```
# Fill in the ...
secdf <- sql...(conn, sec...)
```

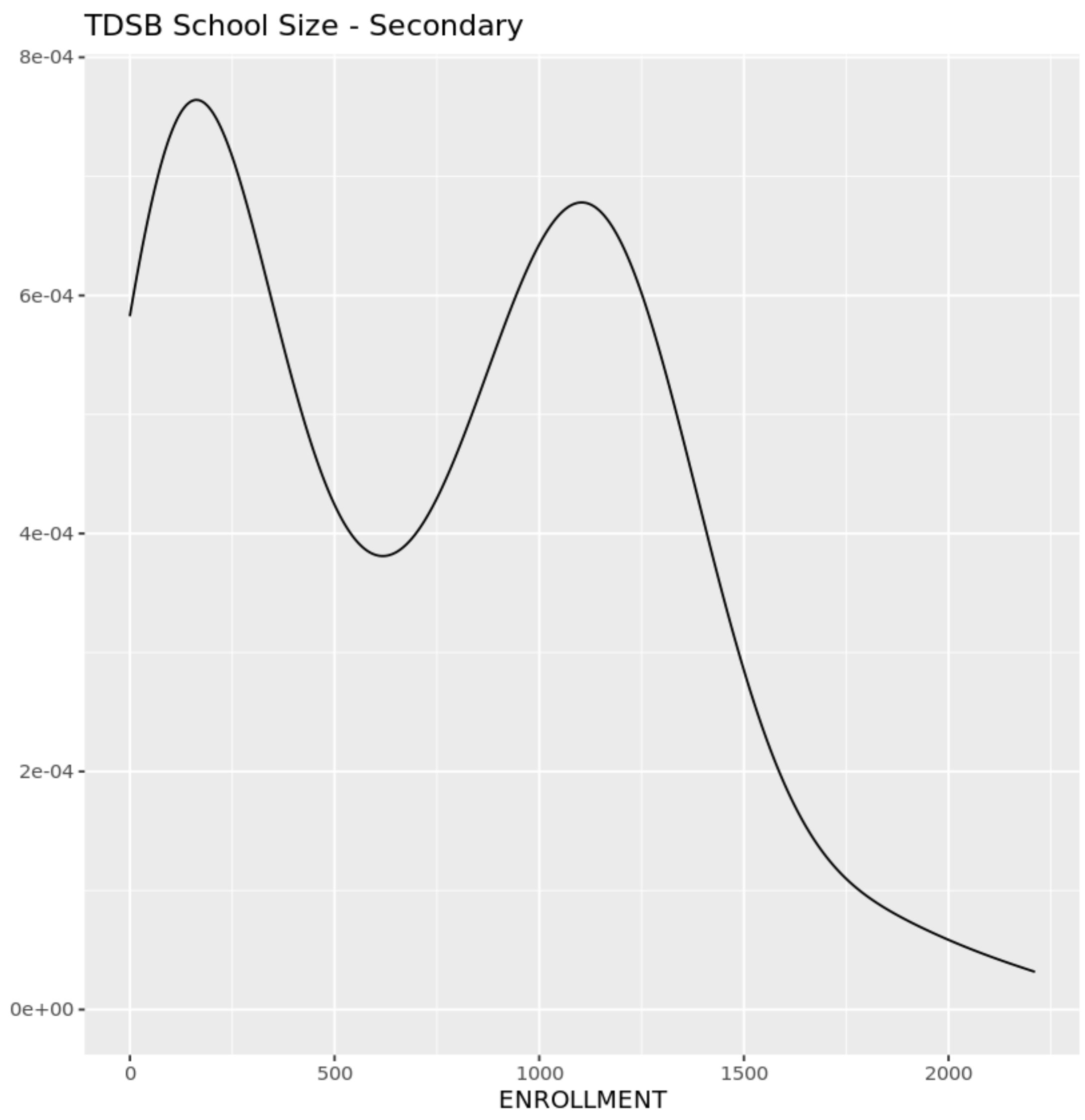
Click here to view/hide solution

```
secdf <- sqlQuery(conn, secquery)
```

Create a density plot of secondary school enrollments.

```
[33]: qplot(ENROLLMENT, data=secdf, geom="density", main="TDSB School Size -␣
      ↪Secondary")
```

```
Warning message:
"Removed 2 rows containing non-finite values (stat_density)."
```



Click here to view/hide hint

```
# Fill in the ...
q...(ENROLLMENT, ...=secdf, ...="density", ...="TDSB School Size - ...")
```

Click here to view/hide solution

```
qplot(ENROLLMENT, data=secdf, geom="density", main="TDSB School Size - Secondary")
```

Query the BOARD database for enrollments.

[34]:
```
denquery <- paste("select b.b_name, s.s_name, level as LEVEL, enrollment
 from board b, school s where b.b_id = s.b_id and b.b_name = 'Toronto DSB'")
```

Click here to view/hide hint

```
# Fill in the ...
den... <- paste("select b.b_..., s.s_..., level as ..., ...
 from board b, ... s where b.... = s.b_id and b.b_... = 'Toronto DSB'")
```

Click here to view/hide solution

```
denquery <- paste("select b.b_name, s.s_name, level as LEVEL, enrollment
 from board b, school s where b.b_id = s.b_id and b.b_name = 'Toronto DSB'")
```

Query the database.

[36]:
```
dendf <- sqlQuery(conn, denquery)
```

Click here to view/hide hint

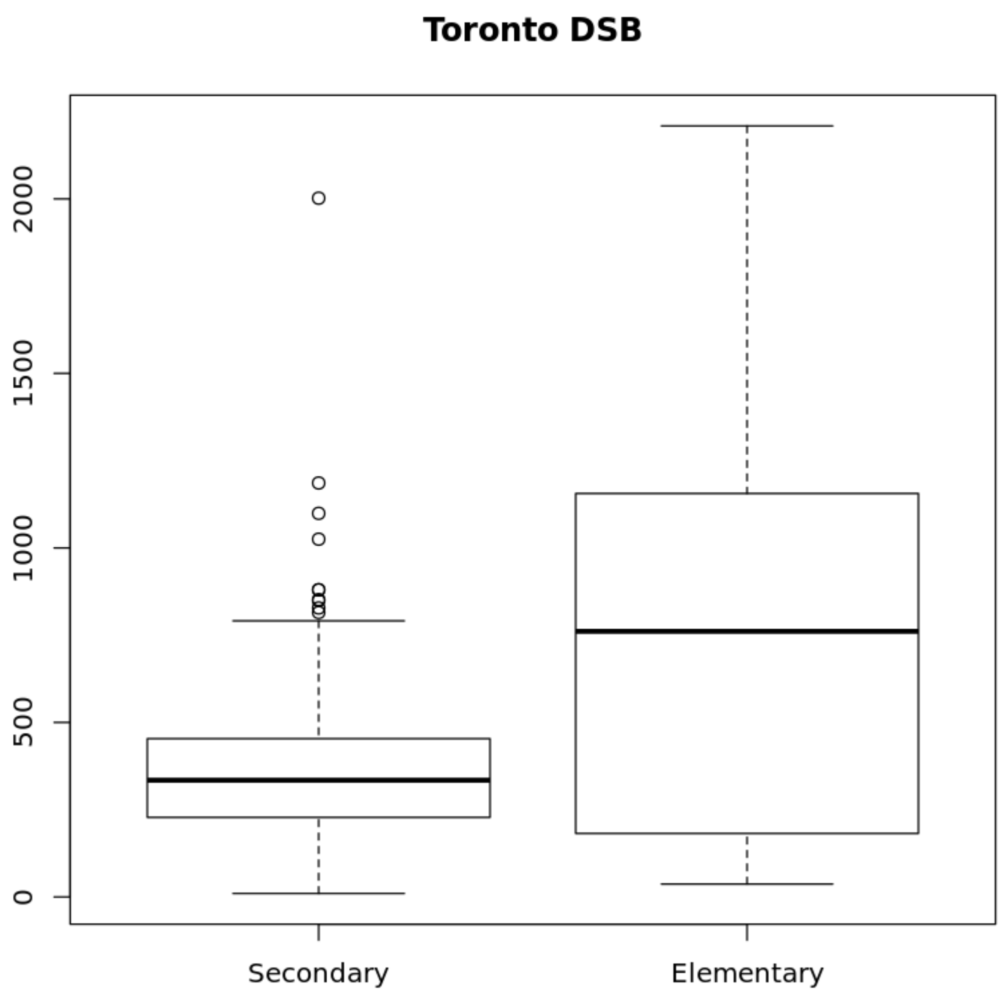
```
# Fill in the ...
d...f <- sql...(conn, den...)
```

Click here to view/hide solution

```
dendf <- sqlQuery(conn, denquery)
```

Create a box plot of enrollements in elementary and secondary schools in Toronto.

[37]:
```
dendf$LEVEL <- as.factor(dendf$LEVEL)
boxplot(ENROLLMENT ~ LEVEL, dendf, names =c("Secondary","Elementary"),␣
 ↪main="Toronto DSB")
```



Click here to view/hide hint

```
# Fill in the ...
d...f$LEVEL <- as....(d...f$LEVEL)
box...(ENROLLMENT ~ ..., d...f, names =c("...","..."), ...="Toronto DSB")
```

Click here to view/hide solution

```
dendf$LEVEL <- as.factor(dendf$LEVEL)
boxplot(ENROLLMENT ~ LEVEL, dendf, names =c("Secondary","Elementary"), main="Toronto DSB")
```

0.0.11 j. Dis-connect

Finally, as a best practice we should close the database connection once we're done with it.

[38]:
```
close(conn)
```

Click here to view/hide hint

```
# Fill in the ...
...(conn)
```

Click here to view/hide solution

```
close(conn)
```

0.0.12 Summary

In this lab you created and queried database objects from an R notebook in Jupyter, and you used ggplot2 to plot the data using R libraries.

Thank you for completing this module on creating and querying database objects from R.

0.1 Authors

- Rav Ahuja
- Agatha Colangelo
- Sandip Saha Joy

0.2 Changelog

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-01-22	2.0	Sandip Saha Joy	Created revised version of the lab
2017	1.0	Rav Ahuja & Agatha Colangelo	Created initial version of the lab