
# Practice Lab: Data type considerations for real world datasets

Estimated time needed: 30 minutes

## Objectives

Using this R notebook you will:

1. Understand two real world datasets
2. Create tables in in a Db2 database while paying particular attention to data types
3. Load the datasets into two separate tables in a Db2 database
4. Practice your SQL skills by using the RODBC package in R to solve some problems

# Understand the datasets

To complete the assignment problems in this notebook you will be using subsetted snapshots of one dataset from Statistics Canada, and one from the Bank of Canada. The links to the prepared datasets are provided in the next section; the interested student can explore the landing pages for the source datasets as follows:

1. Canadian Principal Crops (Data & Metadata)
2. Bank of Canada daily average exchange rates

### 1. Canadian Principal Crops Data

This dataset contains agricultural production measures for the principle crops grown in Canada, including a breakdown by province and teritory, for each year from 1908 to 2020.

For this assignment you will use a preprocessed snapshot of this dataset (see next section).

A detailed description of this dataset can be obtained from the StatsCan Data Portal at: https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=3210035901
Detailed information is included in the metadata file and as header text in the data file, which can be downloaded - look for the 'download options' link.

### 2. Bank of Canada daily average exchange rates

This dataset contains the daily average exchange rates for multiple foreign currencies. Exchange rates are expressed as 1 unit of the foreign currency converted into Canadian dollars. It includes only the latest four years of data, and the rates are published once each business day by 16:30 ET.

For this assignment you will use a snapshot of this dataset with only the USD-CAD exchange rates included (see next section).

A brief description of this dataset and the original dataset can be obtained from the Bank of Canada Data Portal at: https://www.bankofcanada.ca/rates/exchange/daily-exchange-rates/

### Dataset URLs

1. Annual Crop Data: https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Practice%20Assignment/Annual_Crop_Data.csv
2. Daily FX Data: https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Practice%20Assignment/Daily_FX.csv

**IMPORTANT:** You will be loading these datasets directly into R data frames from the URLs instead of from the StatsCan and Bank of Canada portals. The versions provided at these URLs are simplified and subsetted versions of the original datasets.

### Load the csv files into dataframes and inspect them

Check the datatypes and whether the data make sense given the column names. In particular, ensure the date columns are typed as <chr>.

In R and Python, for example, date-like strings may be interpreted by I/O functions as dates by default. Normally this is fine but keep in mind it may also have unintended consequences, such as when moving data from one environment to another.

[ ]:
```
crop_df <- read.csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Annual_Crop_Data.csv', colClasses=c(YEAR="character"
    fx_df <- read.csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Daily_FX.csv', colClasses=c(date="character"))

    head(crop_df)
    head(fx_df)
```

▼ Click here for a hint
You can override the default date interpretation by specifying 'colClasses=c(date="character")' when reading the csv files.

▼ Click here for the solution
```
    crop_df <- read.csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Annual_Crop_Data.csv',
colClasses=c(YEAR="character"))
    fx_df <- read.csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Daily_FX.csv',
colClasses=c(date="character"))

    head(crop_df)
    head(fx_df)
```

### Connect to the database

Load the RODBC database interface package and establish a connection with the Db2 DBMS.
Display the connection information.

[ ]:

▼ Click here for the solution
```
    library(RODBC);
```

```
dsn_driver <- "{IBM DB2 ODBC Driver}"
dsn_database <- "bludb"           # e.g. "bludb"
dsn_hostname <- "<Enter Hostname>" # e.g "54a2f15b-5c0f-46df-8954-.databases.appdomain.cloud"
dsn_port <- "<Enter port number"   # e.g. "32733"
dsn_protocol <- "TCPIP"           # i.e. "TCPIP"
dsn_uid <- "<Enter UserID>"       # e.g. "zjh17769"
dsn_pwd <- "<Enter Password>"     # e.g. "zcwd4+8gbq9bm5k4"
dsn_security <- "ssl"

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

# Dump connection info
############################################################
sql.info <- sqlTypeInfo(conn)
conn.info <- odbcGetInfo(conn)
conn.info["DBMS_Name"]
conn.info["DBMS_Ver"]
conn.info["Driver_ODBC_Ver"]
```

## Table creation steps:

Use the following names for the tables:

1. **CROP_DATA**
2. **DAILY_FX**

### Check whether these tables already exist, and drop them if so.

[ ]:

▼ Click here for the solution

```
tables <- c("CROP_DATA", "DAILY_FX")

for (table in tables) {
    # Drop tables if they already exist
    out <- sqlTables(conn, tableType = "TABLE",
                     tableName = table)
    if (nrow(out)>0) {
        err <- sqlDrop(conn, table,
                       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: ",table," was dropped\n")
        }
    }
    else {
         cat ("Table: ", table," does not exist\n")
    }
}
```

### Create "CROP_DATA" table in Db2.

Also, check whether the table was successfully created.

[ ]:

▼ Click here for the solution

```
df1 <- sqlQuery(conn,
                "CREATE TABLE CROP_DATA (
                                    CD_ID INTEGER NOT NULL,
                                    YEAR DATE NOT NULL,
                                    CROP_TYPE VARCHAR(20) NOT NULL,
                                    GEO VARCHAR(20) NOT NULL,
                                    SEEDED_AREA INTEGER NOT NULL,
                                    HARVESTED_AREA INTEGER NOT NULL,
                                    PRODUCTION INTEGER NOT NULL,
                                    AVG_YIELD INTEGER NOT NULL,
                                    PRIMARY KEY (CD_ID)
                                    )",
                errors=FALSE
                )

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

### Create "DAILY_FX" table in Db2.

[ ]:

▼ Click here for the solution

```
df3 <- sqlQuery(conn, "CREATE TABLE DAILY_FX (
                            DFX_ID INTEGER NOT NULL,
                            DATE DATE NOT NULL,
                            FXUSDCAD FLOAT(6),
                            PRIMARY KEY (DFX_ID)
```

```
                        )",
                errors=FALSE
                )

    if (df3 == -1){
        cat ("An error has occurred.\n")
        msg <- odbcGetErrMsg(conn)
        print (msg)
    } else {
        cat ("Table was created successfully.\n")
    }
```

### Load the dataframes into the Db2 tables you created.

[ ]:

▼ Click here for the solution

```
sqlSave(conn, crop_df, "CROP_DATA", append=TRUE, fast=FALSE, rownames=FALSE, colnames=FALSE, verbose=FALSE)
sqlSave(conn, fx_df, "DAILY_FX", append=TRUE, fast=FALSE, rownames=FALSE, colnames=FALSE, verbose=FALSE)
```

[ ]:

## Now let's solve some practice problems using SQL commands:

### Find the number of rows in each table.

[ ]:

▼ Click here for the solution

```
query = "SELECT COUNT(CD_ID) FROM CROP_DATA"
sqlQuery(conn,query)

query = "SELECT COUNT(DFX_ID) FROM DAILY_FX"
sqlQuery(conn,query)
```

### Query and display the first 6 rows of the crop data.

[ ]:

▼ Click here for the solution

```
query <- "SELECT * FROM CROP_DATA LIMIT 6;"
view <- sqlQuery(conn,query)
view
```

```

# Notice that we did not just query the entire table and then display first 6 rows of the dataframe.
# For larger datasets, using a LIMIT statement this way can save a lot of transit time in moving the data.
```

### List the types of crops in the crop dataset.

[ ]:

▼ Click here for the solution

```
query <- "SELECT DISTINCT(CROP_TYPE) FROM CROP_DATA;"
view <- sqlQuery(conn,query)
view
```

### Query and display the first 6 rows of the crop data for Rye.

**Did you know?** IBM Watson Studio lets you build and deploy an AI solution, using the best of open source and IBM software and giving your team a single environment to work in. Learn more here.

[ ]:

▼ Click here for the solution

```
query <- "SELECT * FROM CROP_DATA WHERE CROP_TYPE='Rye' LIMIT 6;"
view <- sqlQuery(conn,query)
view
```

### Which crops have had an average yield greater than or equal to 3000 KG per Hectare?

[ ]:

▼ Click here for the solution

```
query <-
"SELECT DISTINCT(CROP_TYPE)
FROM CROP_DATA
WHERE AVG_YIELD > 3000;"
view <- sqlQuery(conn,query)
view
```

### Find the first and last dates of each table.

[ ]:

▼ Click here for the solution

```
query <-
"SELECT min(DATE) FIRST_DATE, max(DATE) LAST_DATE
FROM DAILY_FX;
"
view <- sqlQuery(conn,query)
view

query <-
"SELECT min(YEAR) FIRST_DATE, max(YEAR) LAST_DATE
FROM CROP_DATA;
"
view <- sqlQuery(conn,query)
view
```

### List the top 10 years of Wheat production in Saskatchewan in terms of harvested area.

[ ]:

▼ Click here for the solution

```
query <-
"SELECT YEAR(YEAR) AS TOP_10_YRS, GEO, HARVESTED_AREA
FROM CROP_DATA
WHERE CROP_TYPE='Wheat' AND
      GEO='Saskatchewan'
ORDER BY HARVESTED_AREA DESC
LIMIT 10;"
view <- sqlQuery(conn,query)
view
```

### How many years did Barley yield at least 2000 KG per Hectare in Canada?

[ ]:

▼ Click here for the solution

```
query <-
"SELECT COUNT(DISTINCT(YEAR)) AS BLY_YRS_ABOVE_2MTPH
FROM CROP_DATA
WHERE AVG_YIELD > 2000 AND
      CROP_TYPE='Barley' AND
      GEO='Canada';"
view <- sqlQuery(conn,query)
view
```

### How much farm land was seeeded with Barley in Alberta but not harvested each year since the year 2000?

Express your answer as a percentage of seeded area. What assumption might be implied in reporting this calculation?

▼ Click here for a hint

`Create a new 'AS' column called something like 'PCT_UNHARVESTED_AREA'.`

▼ Click here for another hint

`SELECT ....., 100*(? - ?)/SEEDED_AREA AS`

[ ]:

▼ Click here for the solution

```
# Assumption: Barley crops are harvested in the same year they were seeded. Is that true?

query <-
"SELECT YEAR(YEAR) AS YEAR, GEO, CROP_TYPE,
        SEEDED_AREA, HARVESTED_AREA,
        100*(SEEDED_AREA-HARVESTED_AREA)/SEEDED_AREA AS PCT_UNHARVESTED_AREA
FROM CROP_DATA
WHERE YEAR(YEAR) >= 2000 AND
      GEO = 'Alberta' AND
      CROP_TYPE = 'Barley';"

view <- sqlQuery(conn,query)
view
```

### Over the last 3 calendar years of data, what was the average value of the Canadian dollar relative to the USD?

[ ]:

▼ Click here for the solution

```
query <-
"SELECT MIN(DATE) AS AS_OF_DATE,
        AVG(FXUSDCAD) AS FX_DAILY_AVG_CAD
FROM  DAILY_FX
WHERE DATE >= (SELECT MAX(DATE) - 3 YEARS FROM DAILY_FX);
"
view <- sqlQuery(conn,query)
view
```

[ ]:

### Use an implicit inner join to create a view of the crop data with an FX column included.

For simplicity, just use the FX values from December for each year.

[ ]:

▼ Click here for a hint

`Use the year and month parts of the date columns to align the tables on December of each year.`

▼ Click here for the solution

```
query <- "SELECT CD_ID, YEAR, CROP_TYPE, GEO, SEEDED_AREA, HARVESTED_AREA, PRODUCTION, AVG_YIELD, FXUSDCAD
FROM CROP_DATA, MONTHLY_FX
WHERE YEAR(CROP_DATA.YEAR)=YEAR(MONTHLY_FX.DATE) AND MONTH(CROP_DATA.YEAR)=MONTH(MONTHLY_FX.DATE)
LIMIT 5;"
view <- sqlQuery(conn,query)
view
```

### That's it!

Don't forget to close the connection when you are done.

[25]:
```
close(conn)
```

## Author(s)

**Jeff Grossman**

## Contributor(s)

Rav Ahuja

## Change log

| Date | Version | Changed by | Change Description |
|---|---|---|---|
| 2021-07-14 | 2.1 | Lakshmi Holla | Added ssl changes |
| 2021-03-12 | 0.3 | Jeff Grossman | Cleaned up content for production |
| 2021-03-10 | 0.2 | Jeff Grossman | Added introductory and intermediate level problems and removed some advanced problems |
| 2021-03-04 | 0.1 | Jeff Grossman | Started content creation |

© IBM Corporation 2021. All rights reserved.