Assignment: Notebook for Peer Assignment

Estimated time needed: 45 minutes

Assignment Scenario

Congratulations! You have just been hired by a US Venture Capital firm as a data analyst.

The company is considering foreign grain markets to help meet its supply chain requirements for its recent investments in the microbrewery and microdistillery industry, which is involved with the production and distribution of craft beers and spirits.

Your first task is to provide a high level analysis of crop production in Canada. Your stakeholders want to understand the current and historical performance of certain crop types in terms of supply and price volatility. For now they are mainly interested in a macro-view of Canada's crop farming industry, and how it relates to the relative value of the Canadian and US dollars.

Introduction

Using this R notebook you will:

1. Understand four datasets
2. Load the datasets into four separate tables in a Db2 database
3. Execute SQL queries unsing the RODBC R package to answer assignment questions

You have already encountered two of these datasets in the previous practice lab. You will be able to reuse much of the work you did there to prepare your database tables for executing SQL queries.

Understand the datasets

Understand the datasets

To complete the assignment problems in this notebook you will be using subsetted snapshots of two datasets 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) (https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMRP0203ENSkillsNetwork890-2022-01-01&pid=3210035901)
2. Farm product prices (Data & Metadata) (https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMRP0203ENSkillsNetwork890-2022-01-01&pid=3210007701)
3. Bank of Canada daily average exchange rates (https://www.bankofcanada.ca/rates/exchange/daily-exchange-rates?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMRP0203ENSkillsNetwork890-2022-01-01)

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 below).

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 (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. Farm product prices

This dataset contains monthly average farm product prices for Canadian crops and livestock by province and teritory, from 1980 to 2020 (or 'last year', whichever is greatest).

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

A description of this dataset can be obtained from the StatsCan Data Portal at: https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=3210007701 (https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=3210007701) The information is included in the metadata file, which can be downloaded - look for the 'download options' link.

3. 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). We have also prepared a monthly averaged version which you will be using below.

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/ (https://www.bankofcanada.ca/rates/exchange/daily-exchange-rates/)

(* these datasets are the same as the ones you used in the practice lab)

Dataset URLs

1. Annual Crop Data: https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Annual_Crop_Data.csv (https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Annual_Crop_Data.csv)
2. Farm product prices: https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Monthly_Farm_Prices.csv (https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Monthly_Farm_Prices.csv)
3. Daily FX Data: https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Daily_FX.csv (https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Daily_FX.csv)
4. Monthly FX Data: https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Monthly_FX.csv (https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Monthly_FX.csv)

IMPORTANT: You will be loading these datasets directly into R data frames from these 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.

Now let's load these datasets into four separate Db2 tables.

Let's first load the RODBC package:

In [ ]:

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

Problem 1

Create tables

Establish a connection to the Db2 database, and create the following four tables using the RODBC package in R. Use the separate cells provided below to create each of your tables.

1. **CROP_DATA**
2. **FARM_PRICES**
3. **DAILY_FX**
4. **MONTHLY_FX**

The previous practice lab will help you accomplish this.

Solution 1

In [2]:
```
# Establish database connection
dsn_driver <- "{IBM DB2 ODBC Driver}"
dsn_database <- "bludb"            # e.g. "bludb"
dsn_hostname <- "fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud" # e.g "
dsn_port <- "32731"   # e.g. "32733"
dsn_protocol <- "TCPIP"            # i.e. "TCPIP"
dsn_uid <- "rhf80803"        # e.g. "zjh17769"
dsn_pwd <- "7mXFONnLhUmwMhjm"      # 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)
conn
```

```
RODBC Connection 1
Details:
  case=nochange
  DRIVER={IBM DB2 ODBC DRIVER}
  UID=rhf80803
  PWD=******
  DATABASE=bludb
  HOSTNAME=fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud
  PORT=32731
  PROTOCOL=TCPIP
  SECURITY=SSL
```

In [3]:
```
#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"]
```

DBMS_Name: 'DB2/LINUXX8664'

DBMS_Ver: '11.05.0800'

Driver_ODBC_Ver: '03.51'

In [4]:
```
# CROP_DATA:
myschema <- "RHF80803" # e.g. "ZJH17769"
tables <- c("CROP_DATA")

    for (table in tables){
      # Drop CROP_DATA 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:  RHF80803 . CROP_DATA  was dropped
```

In [ ]:

In [5]:
```
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)
```

In [6]:
```
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.
```

In [7]:

```
# FARM_PRICES:

myschema <- "RHF80803" # e.g. "ZJH17769"
tables <- c("FARM_PRICES")

    for (table in tables){
      # Drop FARM_PRICES 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")
      }
    }

##############################
df1 <- sqlQuery(conn, "CREATE TABLE FARM_PRICES (
                            CD_ID INTEGER NOT NULL,
                                DATE DATE NOT NULL,
                                CROP_TYPE VARCHAR(20) NOT NULL,
                                GEO VARCHAR(20) NOT NULL,
                                PRICE_PRERMT FLOAT(6),
                                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")
```

```
}
```

```
Table:  RHF80803 . FARM_PRICES  was dropped
Table was created successfully.
```

In [20]:

```
# DAILY_FX:

myschema <- "RHF80803" # e.g. "ZJH17769"
tables <- c("DAILY_FX")

    for (table in tables){
      # Drop DAILY_FX 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")
      }
    }

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

```
}
```

```
Table:  RHF80803 . DAILY_FX  was dropped
Table was created successfully.
```

In [23]:

```
# MONTHLY_FX:

myschema <- "RHF80803" # e.g. "ZJH17769"
tables <- c("MONTHLY_FX")

    for (table in tables){
      # Drop MONTHLY_FX 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")
      }
    }

################################
df1 <- sqlQuery(conn, "CREATE TABLE MONTHLY_FX (
                            DFX_ID INTEGER NOT NULL,
                                DATE DATE NOT NULl,
                                FXUSDCAD FLOAT(6),
                                PRIMARY KEY (DFX_ID)
                            )",
                errors=FALSE)

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

```
}
```

```
Table:  RHF80803 . MONTHLY_FX  was dropped
Table was created successfully.
```

Problem 2

Read Datasets and Load Tables

Read the datasets into R dataframes using the urls provided above. Then load your tables.

Solution 2

In [ ]:

In [10]:
```
tables <- c("CROP_DATA","DAILY_FX","FARM_PRICES","MONTHLY_FX")
```

In [11]:
```
data_urls <- list(CROP_DATA = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203E
FARM_PRICES = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/
DAILY_FX = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/lab
MONTHLY_FX = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/l
```

In [12]:
```
data_urls
```

$CROP_DATA
'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Annual_Crop_Data.csv'
$FARM_PRICES
'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Monthly_Farm_Prices.csv'
$DAILY_FX
'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Daily_FX.csv'
$MONTHLY_FX
'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Monthly_FX.csv'

In [13]:
```
tab.frame <- sqlTables(conn, schema=myschema)
nrow(tab.frame)
tab.frame$TABLE_NAME

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)
}
```

14

'BOARD' 'CROP_DATA' 'DAILY_FX' 'DEPARTMENTS' 'EMPLOYEES' 'FARM_PRICES' 'JOBS' 'JOB_HISTORY' 'LOCATIONS' 'MONTHLY_FX' 'PETRESCUE' 'PETSALE' 'SCHOOL' 'SPACEX'

```
Column info for table CROP_DATA :
    COLUMN_NAME TYPE_NAME COLUMN_SIZE IS_NULLABLE
          CD_ID   INTEGER          10          NO
           YEAR      DATE          10          NO
      CROP_TYPE   VARCHAR          20          NO
            GEO   VARCHAR          20          NO
    SEEDED_AREA   INTEGER          10          NO
 HARVESTED_AREA   INTEGER          10          NO
     PRODUCTION   INTEGER          10          NO
      AVG_YIELD   INTEGER          10          NO

Column info for table DAILY_FX :
 COLUMN_NAME TYPE_NAME COLUMN_SIZE IS_NULLABLE
      DFX_ID   INTEGER          10          NO
        DATE      DATE          10          NO
    FXUSDCAD      REAL          24         YES

Column info for table FARM_PRICES :
  COLUMN_NAME TYPE_NAME COLUMN_SIZE IS_NULLABLE
        CD_ID   INTEGER          10          NO
         DATE      DATE          10          NO
    CROP_TYPE   VARCHAR          20          NO
          GEO   VARCHAR          20          NO
 PRICE_PRERMT      REAL          24         YES

Column info for table MONTHLY_FX :
 COLUMN_NAME TYPE_NAME COLUMN_SIZE IS_NULLABLE
      DFX_ID   INTEGER          10          NO
        DATE      DATE          10          NO
    FXUSDCAD      REAL          24         YES
```

In [14]:
```
class(data_urls[['CROP_DATA']])
```

'character'

In [15]:
```
# CROP_DATA
cropdatadf <- read.csv(data_urls[['CROP_DATA']], header = TRUE)
head(cropdatadf)
sqlSave(conn, cropdatadf, "CROP_DATA", append=TRUE, fast=FALSE, rownames=FALSE, colnames=FALSE, verbose=FALSE

testdf <- sqlFetch(conn, "CROP_DATA")
head(testdf)
```

A data.frame: 6 × 8

	CD_ID	YEAR	CROP_TYPE	GEO	SEEDED_AREA	HARVESTED_AREA	PRODUCTION	AVG_YIELD
	<int>	<fct>	<fct>	<fct>	<int>	<int>	<int>	<int>
1	0	1965-12-31	Barley	Alberta	1372000	1372000	2504000	1825
2	1	1965-12-31	Barley	Canada	2476800	2476800	4752900	1920
3	2	1965-12-31	Barley	Saskatchewan	708000	708000	1415000	2000
4	3	1965-12-31	Canola	Alberta	297400	297400	215500	725
5	4	1965-12-31	Canola	Canada	580700	580700	512600	885
6	5	1965-12-31	Canola	Saskatchewan	224600	224600	242700	1080

A data.frame: 6 × 8

	CD_ID	YEAR	CROP_TYPE	GEO	SEEDED_AREA	HARVESTED_AREA	PRODUCTION	AVG_YIELD
	<int>	<date>	<fct>	<fct>	<int>	<int>	<int>	<int>
1	0	1965-12-31	Barley	Alberta	1372000	1372000	2504000	1825
2	1	1965-12-31	Barley	Canada	2476800	2476800	4752900	1920
3	2	1965-12-31	Barley	Saskatchewan	708000	708000	1415000	2000
4	3	1965-12-31	Canola	Alberta	297400	297400	215500	725
5	4	1965-12-31	Canola	Canada	580700	580700	512600	885
6	5	1965-12-31	Canola	Saskatchewan	224600	224600	242700	1080

In [16]:
```
data_urls
```

$CROP_DATA
'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Annual_Crop_Data.csv'
$FARM_PRICES
'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Monthly_Farm_Prices.csv'
$DAILY_FX
'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Daily_FX.csv'
$MONTHLY_FX
'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Monthly_FX.csv'

In [17]:
```
# FARM_PRICES
FARM_PRICESdf <- read.csv(data_urls[['FARM_PRICES']], header = TRUE)
head(FARM_PRICESdf)
sqlSave(conn, FARM_PRICESdf, "FARM_PRICES", append=TRUE, fast=FALSE, rownames=FALSE, colnames=FALSE, verbose=

testdf <- sqlFetch(conn, "FARM_PRICES")
head(testdf)
```

A data.frame: 6 × 5

	CD_ID	DATE	CROP_TYPE	GEO	PRICE_PRERMT
	<int>	<fct>	<fct>	<fct>	<dbl>
1	0	1985-01-01	Barley	Alberta	127.39
2	1	1985-01-01	Barley	Saskatchewan	121.38
3	2	1985-01-01	Canola	Alberta	342.00
4	3	1985-01-01	Canola	Saskatchewan	339.82
5	4	1985-01-01	Rye	Alberta	100.77
6	5	1985-01-01	Rye	Saskatchewan	109.75

A data.frame: 6 × 5

	CD_ID	DATE	CROP_TYPE	GEO	PRICE_PRERMT
	<int>	<date>	<fct>	<fct>	<dbl>
1	0	1985-01-01	Barley	Alberta	127.39
2	1	1985-01-01	Barley	Saskatchewan	121.38
3	2	1985-01-01	Canola	Alberta	342.00
4	3	1985-01-01	Canola	Saskatchewan	339.82
5	4	1985-01-01	Rye	Alberta	100.77
6	5	1985-01-01	Rye	Saskatchewan	109.75

In [21]:
```
# DAILY_FX
DAILY_FXdf <- read.csv(data_urls[['DAILY_FX']], header = TRUE)
head(DAILY_FXdf)
sqlSave(conn, DAILY_FXdf, "DAILY_FX", append=TRUE, fast=FALSE, rownames=FALSE, colnames=FALSE, verbose=FALSE)

print('test')
testdf <- sqlFetch(conn, "DAILY_FX")
head(testdf)
```

A data.frame: 6 × 3

	DFX_ID	DATE	FXUSDCAD
	<int>	<fct>	<dbl>
1	0	2017-01-03	1.3435
2	1	2017-01-04	1.3315
3	2	2017-01-05	1.3244
4	3	2017-01-06	1.3214
5	4	2017-01-09	1.3240
6	5	2017-01-10	1.3213

```
[1] "test"
```

A data.frame: 6 × 3

	DFX_ID	DATE	FXUSDCAD
	<int>	<date>	<dbl>
1	0	2017-01-03	1.3435
2	1	2017-01-04	1.3315
3	2	2017-01-05	1.3244
4	3	2017-01-06	1.3214
5	4	2017-01-09	1.3240
6	5	2017-01-10	1.3213

In [24]:
```
# MONTHLY_FX
MONTHLY_FXdf <- read.csv(data_urls[['MONTHLY_FX']], header = TRUE)
head(MONTHLY_FXdf)
sqlSave(conn, MONTHLY_FXdf, "MONTHLY_FX", append=TRUE, fast=FALSE, rownames=FALSE, colnames=FALSE, verbose=FA

testdf <- sqlFetch(conn, "MONTHLY_FX")
head(testdf)
```

A data.frame: 6 × 3

	DFX_ID	DATE	FXUSDCAD
	<int>	<fct>	<dbl>
1	0	2017-01-01	1.319276
2	1	2017-02-01	1.310726
3	2	2017-03-01	1.338643
4	3	2017-04-01	1.344021
5	4	2017-05-01	1.360705
6	5	2017-06-01	1.329805

A data.frame: 6 × 3

	DFX_ID	DATE	FXUSDCAD
	<int>	<date>	<dbl>
1	0	2017-01-01	1.319276
2	1	2017-02-01	1.310726
3	2	2017-03-01	1.338643
4	3	2017-04-01	1.344021
5	4	2017-05-01	1.360705
6	5	2017-06-01	1.329805

Now execute SQL queries using the RODBC R package to solve the

assignment problems.

Problem 3

Solution 3

In [25]:
```
query = "SELECT COUNT(*) FROM FARM_PRICES"
sqlQuery(conn,query)
```

A
data.frame:
1 × 1

	1
	<int>
1	2678

Problem 4

Which geographies are included in the farm prices dataset?

Solution 4

In [26]:
```
query <- "SELECT DISTINCT GEO FROM FARM_PRICES"
sqlQuery(conn,query)
```

A data.frame: 2 × 1

	GEO
	<fct>
1	Alberta
2	Saskatchewan

Problem 5

How many hectares of Rye were harvested in Canada in 1968?

Solution 5

In [27]:
```
query <- "SELECT * FROM CROP_DATA LIMIT 5"
sqlQuery(conn,query)
```

A data.frame: 5 × 8

	CD_ID	YEAR	CROP_TYPE	GEO	SEEDED_AREA	HARVESTED_AREA	PRODUCTION	AVG_YIELD
	<int>	<date>	<fct>	<fct>	<int>	<int>	<int>	<int>
1	0	1965-12-31	Barley	Alberta	1372000	1372000	2504000	1825
2	1	1965-12-31	Barley	Canada	2476800	2476800	4752900	1920
3	2	1965-12-31	Barley	Saskatchewan	708000	708000	1415000	2000
4	3	1965-12-31	Canola	Alberta	297400	297400	215500	725
5	4	1965-12-31	Canola	Canada	580700	580700	512600	885

In [35]:
```
query <- "SELECT CROP_TYPE, YEAR, GEO, HARVESTED_AREA FROM CROP_DATA WHERE CROP_TYPE = 'Rye' AND
GEO = 'Canada' AND YEAR(YEAR) = '1968'"
sqlQuery(conn,query)
```

A data.frame: 1 × 4

	CROP_TYPE	YEAR	GEO	HARVESTED_AREA
	<fct>	<date>	<fct>	<int>
1	Rye	1968-12-31	Canada	274100

In [ ]:
```
```

Problem 6

Query and display the first 6 rows of the farm prices table for Rye.

Solution 6

In [37]:
```
query <- "SELECT * FROM FARM_PRICES WHERE CROP_TYPE = 'Rye' LIMIT 6"
sqlQuery(conn,query)
```

A data.frame: 6 × 5

	CD_ID	DATE	CROP_TYPE	GEO	PRICE_PRERMT
	<int>	<date>	<fct>	<fct>	<dbl>
1	4	1985-01-01	Rye	Alberta	100.77
2	5	1985-01-01	Rye	Saskatchewan	109.75
3	10	1985-02-01	Rye	Alberta	95.05
4	11	1985-02-01	Rye	Saskatchewan	103.46
5	16	1985-03-01	Rye	Alberta	96.77
6	17	1985-03-01	Rye	Saskatchewan	106.38

Problem 7

Which provinces grew Barley?

Solution 7

In [41]:
```
query <- "SELECT DISTINCT GEO FROM CROP_DATA WHERE CROP_TYPE = 'Barley'"
sqlQuery(conn,query)
```

A data.frame: 3 × 1

	GEO
	<fct>
1	Alberta
2	Canada
3	Saskatchewan

Problem 8

Find the first and last dates for the farm prices data.

Solution 8

In [44]:
```
query <- "SELECT MIN(DATE) AS FIRST_DATE, MAX(DATE) AS LAST_DATE FROM FARM_PRICES"
sqlQuery(conn,query)
```

A data.frame: 1 × 2

	FIRST_DATE	LAST_DATE
	<date>	<date>
1	1985-01-01	2020-12-01

Problem 9

Which crops have ever reached a farm price greater than or equal to $350 per metric tonne?

Solution 9

In [49]:
```
sqlQuery(conn,"SELECT DISTINCT CROP_TYPE FROM FARM_PRICES WHERE PRICE_PRERMT >= 350")
```

A data.frame: 1 × 1

	CROP_TYPE
	<fct>
1	Canola

Problem 10

Rank the crop types harvested in Saskatchewan in the year 2000 by their average yield. Which crop performed best?

Solution 10

In [59]:
```
sqlQuery(conn,"SELECT CROP_TYPE, AVG_YIELD FROM CROP_DATA
WHERE GEO = 'Saskatchewan' AND YEAR(YEAR) = '2000'
ORDER BY AVG_YIELD DESC")
```

A data.frame: 4 × 2

	CROP_TYPE	AVG_YIELD
	<fct>	<int>
1	Barley	2800
2	Wheat	2200
3	Rye	2100
4	Canola	1400

In [60]:
```
sqlQuery(conn,"SELECT CROP_TYPE, AVG_YIELD FROM CROP_DATA
WHERE GEO = 'Saskatchewan' AND YEAR(YEAR) = '2000'
ORDER BY AVG_YIELD DESC
LIMIT 1")
```

A data.frame: 1 × 2

	CROP_TYPE	AVG_YIELD
	<fct>	<int>
1	Barley	2800

Problem 11

Rank the crops and geographies by their average yield (KG per hectare) since the year 2000. Which crop and province had the highest average yield since the year 2000?

Solution 11

In [66]:
```
sqlQuery(conn,"SELECT CROP_TYPE, GEO, AVG(AVG_YIELD) AS AVG_OF_AVG_YIELD
FROM CROP_DATA
WHERE YEAR(YEAR) >= '2000'
GROUP BY CROP_TYPE, GEO
ORDER BY AVG_OF_AVG_YIELD DESC
")
```

A data.frame: 12 × 3

	CROP_TYPE	GEO	AVG_OF_AVG_YIELD
	<fct>	<fct>	<int>
1	Barley	Alberta	3450
2	Barley	Canada	3253
3	Wheat	Alberta	3100
4	Barley	Saskatchewan	2971
5	Wheat	Canada	2845
6	Rye	Alberta	2683
7	Rye	Canada	2543
8	Wheat	Saskatchewan	2429
9	Rye	Saskatchewan	2226
10	Canola	Alberta	1999
11	Canola	Canada	1873
12	Canola	Saskatchewan	1754

In [69]:
```
query <- "SELECT CROP_TYPE, GEO, YEAR, AVG_YIELD FROM CROP_DATA
        WHERE (YEAR(YEAR)>=2000)
        ORDER BY AVG_YIELD DESC
        LIMIT 6"
sqlQuery(conn, query)
```

A data.frame: 6 × 4

	CROP_TYPE	GEO	YEAR	AVG_YIELD
	<fct>	<fct>	<date>	<int>
1	Barley	Alberta	2013-12-31	4100
2	Barley	Alberta	2016-12-31	4100
3	Barley	Alberta	2020-12-31	3980
4	Wheat	Alberta	2013-12-31	3900
5	Wheat	Alberta	2016-12-31	3900
6	Barley	Canada	2016-12-31	3900

Problem 12

Use a subquery to determine how much wheat was harvested in Canada in the most recent year of the data.

Solution 12

In [81]:
```
sqlQuery(conn,"SELECT *
FROM CROP_DATA
WHERE CROP_TYPE = 'Wheat' AND GEO = 'Canada' AND YEAR(YEAR) = (SELECT MAX(YEAR(YEAR)) FROM CROP_DATA)
")
```

A data.frame: 1 × 8

	CD_ID	YEAR	CROP_TYPE	GEO	SEEDED_AREA	HARVESTED_AREA	PRODUCTION	AVG_YIELD
	<int>	<date>	<fct>	<fct>	<int>	<int>	<int>	<int>
1	670	2020-12-31	Wheat	Canada	10193600	10017800	35183000	3512

Problem 13

Use an implicit inner join to calculate the monthly price per metric tonne of Canola grown in Saskatchewan in both Canadian and US dollars. Display the most recent 6 months of the data.

Solution 13

In [ ]:
```

```

In [112]:
```
sqlQuery(conn,"SELECT P.DATE, F.DATE, CROP_TYPE, GEO, PRICE_PRERMT AS USD_PRICE,FXUSDCAD, PRICE_PRERMT*FXUSDC
FROM FARM_PRICES P, MONTHLY_FX F
WHERE P.DATE = F.DATE AND
CROP_TYPE = 'Canola' AND GEO = 'Saskatchewan'
ORDER BY P.DATE DESC
LIMIT 6
")
```

A data.frame: 6 × 7

	DATE	DATE.1	CROP_TYPE	GEO	USD_PRICE	FXUSDCAD	CAD_PRICE
	<date>	<date>	<fct>	<fct>	<dbl>	<dbl>	<dbl>
1	2020-12-01	2020-12-01	Canola	Saskatchewan	507.33	1.280771	649.7735
2	2020-11-01	2020-11-01	Canola	Saskatchewan	495.64	1.306820	647.7123
3	2020-10-01	2020-10-01	Canola	Saskatchewan	474.80	1.321471	627.4344
4	2020-09-01	2020-09-01	Canola	Saskatchewan	463.52	1.322810	613.1489
5	2020-08-01	2020-08-01	Canola	Saskatchewan	464.60	1.322205	614.2964
6	2020-07-01	2020-07-01	Canola	Saskatchewan	462.88	1.349850	624.8186

In [ ]:
```

```

Author(s)

Jeff Grossman

Contributor(s)

Rav Ahuja

Change log

Date	Version	Changed by	Change Description
2021-04-01	0.7	Jeff Grossman	Split Problem 1 solution cell into multiple cells, fixed minor bugs
2021-03-12	0.6	Jeff Grossman	Cleaned up content for production
2021-03-11	0.5	Jeff Grossman	Moved more advanced problems to optional honours module
2021-03-10	0.4	Jeff Grossman	Added introductory and intermediate level problems and removed some advanced problems
2021-03-04	0.3	Jeff Grossman	Moved some problems to a new practice lab as prep for this assignment
2021-03-04	0.2	Jeff Grossman	Sorted problems roughly by level of difficulty and relegated more advanced ones to ungraded bonus problems
2021-02-20	0.1	Jeff Grossman	Started content creation

In [ ]: