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

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=SkillsNetwork-Channel-SkillsNetworkCoursesIBMRP0203ENSkillsNetwork23863830-2021-01-01&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=SkillsNetwork-Channel-SkillsNetworkCoursesIBMRP0203ENSkillsNetwork23863830-2021-01-01&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=SkillsNetwork-Channel-SkillsNetworkCoursesIBMRP0203ENSkillsNetwork23863830-2021-01-01)

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?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMRP0203ENSkillsNetwork23863830-2021-01-01&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?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMRP0203ENSkillsNetwork23863830-2021-01-01&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/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMRP0203ENSkillsNetwork23863830-2021-01-01)

(* 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 [28]:

```
install.packages("RODBC")
library(RODBC)
```

```
Updating HTML index of packages in '.Library'
Making 'packages.html' ... done
```

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 [29]:

```
dsn_driver <- "{IBM DB2 ODBC Driver}"
dsn_database <- "bludb" # e.g. "bludb"
dsn_hostname <- "ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdo
main.cloud" # e.g,"54a2f15b-5c0f-46df-8954-.databases.appdomain.cloud"
dsn_port <- "31321" # e.g. "32733"
dsn_protocol <- "TCPIP" # i.e. "TCPIP"
dsn_uid <- "zdk88861" # e.g. "zjh17769"
dsn_pwd <- "Wsjzwwvp0HfhnK9u" # e.g. "zcwd4+8gbq9bm5k4"
dsn_security <- "ssl"


# Create a connection string and connect to the database
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, believeNRows=FALSE)
conn
```

```
RODBC Connection 2
Details:
  case=nochange
  DRIVER={IBM DB2 ODBC DRIVER}
  UID=zdk88861
  PWD=******
  DATABASE=bludb
  HOSTNAME=ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.database
s.appdomain.cloud
  PORT=31321
  PROTOCOL=TCPIP
  SECURITY=SSL
```

In [30]:

```
# CROP_DATA:

dfa <- sqlQuery(conn, "DROP TABLE CROP_DATA")


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") }
```

```
Table was created successfully.
```

In [31]:

```
dfa <- sqlQuery(conn, "DROP TABLE FARM_PRICES")

# FARM_PRICES:
df2 <- sqlQuery(conn, "CREATE TABLE FARM_PRICES ( DFX_ID INTEGER NOT NULL,
DATE DATE NOT NULL,
CROPTYPE VARCHAR(20),
GEO VARCHAR(20),
PRICEPERMT FLOAT(6),
PRIMARY KEY (DFX_ID) )",
                errors=FALSE )
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.
```

In [32]:

```
dfa <- sqlQuery(conn, "DROP TABLE DAILY_FX")

# DAILY_FX:
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") }
```

```
Table was created successfully.
```

In [33]:

```
dfa <- sqlQuery(conn, "DROP TABLE MONTHLY_FX")

# MONTHLY_FX:
df4 <- 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 (df4 == -1){ cat ("An error has occurred.\n")
msg <- odbcGetErrMsg(conn)
print (msg) } else {
cat ("Table was created successfully.\n") }
```

```
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 [34]:

```
CROP_DATA <- read.csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Annual_Crop_Data.csv")
head(CROP_DATA)
```

A data.frame: 6 × 8

	index	YEAR	cropType	GEO	seededArea	harvestedArea	production	avgYield
	<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

In [35]:

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

```
Error in dimnames(x) <- dn: length of 'dimnames' [2] not equal to array exten
t
Traceback:

1. sqlSave(conn, CROP_DATA, "CROP_DATA", append = TRUE, fast = FALSE,
 .     colnames = FALSE, rownames = FALSE, verbose = FALSE)
2. sqlwrite(channel, tablename, dat, verbose = verbose, fast = fast,
 .     test = test, nastring = nastring)
3. `colnames<-`(`*tmp*`, value = colnames)
```

In [36]:

```
#Farm product prices:
FARM_PRICES <-''
FARM_PRICES <- read.csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.clou
d/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Monthly_Farm_Prices.csv")
head(FARM_PRICES)
```

A data.frame: 6 × 5

	index	date	cropType	GEO	pricePerMT
	<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

In [37]:

```
sqlSave(conn, FARM_PRICES, "FARM_PRICES", append=TRUE, fast=FALSE, colnames=FALSE, rowname
s=FALSE, verbose=FALSE)
```

```
Error in dimnames(x) <- dn: length of 'dimnames' [2] not equal to array exten
t
Traceback:

1. sqlSave(conn, FARM_PRICES, "FARM_PRICES", append = TRUE, fast = FALSE,
 .     colnames = FALSE, rownames = FALSE, verbose = FALSE)
2. sqlwrite(channel, tablename, dat, verbose = verbose, fast = fast,
 .     test = test, nastring = nastring)
3. `colnames<-`(`*tmp*`, value = colnames)
```

In [38]:

```
DAILY_FX<- read.csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Daily_FX.csv")
head(DAILY_FX)
```

A data.frame: 6 × 3

	index	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

In [39]:

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

```
Error in dimnames(x) <- dn: length of 'dimnames' [2] not equal to array exten
t
Traceback:

1. sqlSave(conn, DAILY_FX, "DAILY_FX", append = TRUE, fast = FALSE,
 .     rownames = FALSE, colnames = FALSE, verbose = FALSE)
2. sqlwrite(channel, tablename, dat, verbose = verbose, fast = fast,
 .     test = test, nastring = nastring)
3. `colnames<-`(`*tmp*`, value = colnames)
```

In [40]:

```
MONTHLY_FX <- read.csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.clou
d/IBM-RP0203EN-SkillsNetwork/labs/Final%20Project/Monthly_FX.csv")
head(MONTHLY_FX)
```

A data.frame: 6 × 3

	index	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

In [41]:

```
sqlSave(conn, MONTHLY_FX, "MONTHLY_FX", append=TRUE, fast=FALSE, rownames=FALSE,colnames=F
ALSE, verbose=FALSE)
```

```
Error in dimnames(x) <- dn: length of 'dimnames' [2] not equal to array exten
t
Traceback:

1. sqlSave(conn, MONTHLY_FX, "MONTHLY_FX", append = TRUE, fast = FALSE,
 .     rownames = FALSE, colnames = FALSE, verbose = FALSE)
2. sqlwrite(channel, tablename, dat, verbose = verbose, fast = fast,
 .     test = test, nastring = nastring)
3. `colnames<-`(`*tmp*`, value = colnames)
```

In [42]:

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

A data.frame: 0 × 8

CD_ID	YEAR	CROP_TYPE	GEO	SEEDED_AREA	HARVESTED_AREA	PRODUCTION	AVG_YIE
<int>	<chr>	<chr>	<chr>	<int>	<int>	<int>	<i

In [43]:

```
query <- "SELECT * FROM FARM_PRICES;"
view <- sqlQuery(conn,query)
head(view)
```

A data.frame: 0 × 5

DFX_ID	DATE	CROPTYPE	GEO	PRICEPERMT
<int>	<chr>	<chr>	<chr>	<dbl>

In [44]:

```
query <- "SELECT * FROM DAILY_FX;"
view <- sqlQuery(conn,query)
head(view)
```

A data.frame: 0 × 3

DFX_ID	DATE	FXUSDCAD
<int>	<chr>	<dbl>

In [45]:

```
query <- "SELECT * FROM MONTHLY_FX;"
view <- sqlQuery(conn,query)
head(view)
```

A data.frame: 0 × 3

DFX_ID	DATE	FXUSDCAD
<int>	<chr>	<dbl>

Now execute SQL queries using the RODBC R package to solve the assignment problems.

Problem 3

How many records are in the farm prices dataset?

Solution 3

In [46]:

```
record_count_query <- query <- paste("select count(*) AS count from FARM_PRICES;")
record_count_df<- sqlQuery(conn, record_count_query)
record_count_df


# Non-SQL solution: data.frame(Farm_Product_Prices_df)
#2678 records are in the farm prices dataset
```

A data.frame:

1 × 1

	COUNT
	<int>
1	0

Problem 4

Which geographies are included in the farm prices dataset?

Solution 4

In [47]:

```
unique_farm_prices_geographies_query <- query <- paste("select DISTINCT (GEO) FROM FARM_PR
ICES;")
unique_farm_prices_geographies_df <-sqlQuery(conn,unique_farm_prices_geographies_query)
unique_farm_prices_geographies_df
#The geographies included in the farm prices dataset are Alberta and Saskatchewan.
```

A

data.frame:

0 × 1

GEO
<chr>

Problem 5

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

Solution 5

In [48]:

```
d <-"SELECT sum(harvestedArea) FROM CROP_DATA WHERE YEAR LIKE '1968%';"
v<-sqlQuery(conn,d)
head(v)
```

'42S22 -206 [IBM][CLI Driver][DB2/LINUXX8664] SQL0206N "HARVESTEDAREA" is not valid in the context where it is used. SQLSTATE=42703\n' ·
'[RODBC] ERROR: Could not SQLExecDirect \'SELECT sum(harvestedArea) FROM CROP_DATA WHERE YEAR LIKE \'1968%\';\'

Problem 6

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

Solution 6

In [49]:

```
farm_prices_df<-''
farm_prices_query <- query <- "SELECT * FROM FARM_PRICES WHERE CROP_TYPE = 'Rye';"
farm_prices_df<-sqlQuery(conn,farm_prices_query)
head(farm_prices_df)
```

'42S22 -206 [IBM][CLI Driver][DB2/LINUXX8664] SQL0206N "CROP_TYPE" is not valid in the context where it is used. SQLSTATE=42703\n' ·
'[RODBC] ERROR: Could not SQLExecDirect \'SELECT * FROM FARM_PRICES WHERE CROP_TYPE = \'Rye\';\'

Problem 7

Which provinces grew Barley?

Solution 7

In [50]:

```
barley_query<-"SELECT DISTINCT(GEO) FROM CROP_DATA WHERE croptype = 'BARLEY';"
barley_df<-sqlQuery(conn,barley_query)
barley_df

#Alberta, Saskatchewan
```

'42S22 -206 [IBM][CLI Driver][DB2/LINUXX8664] SQL0206N "CROPTYPE" is not valid in the context where it is used. SQLSTATE=42703\n' ·
'[RODBC] ERROR: Could not SQLExecDirect \'SELECT DISTINCT(GEO) FROM CROP_DATA WHERE croptype = \'BARLEY\';\'

Problem 8

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

Solution 8

In [51]:

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


#1985-01-01 / 2020-12-01
```

```
Error in charToDate(x): character string is not in a standard unambiguous for
mat
Traceback:

1. sqlQuery(conn, query)
2. sqlGetResults(channel, errors = errors, ...)
3. as.Date(data[[i]])
4. as.Date.character(data[[i]])
5. charToDate(x)
6. stop("character string is not in a standard unambiguous format")
```

Problem 9

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

Solution 9

In [52]:

```
high_price_query<-"SELECT * FROM FARM_PRICES WHERE PricePerMT >350;"
high_price_df<-sqlQuery(conn,high_price_query)
high_price_df

#Canola
```

A data.frame: 0 × 5

DFX_ID	DATE	CROPTYPE	GEO	PRICEPERMT
<int>	<chr>	<chr>	<chr>	<dbl>

Problem 10

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

Solution 10

In [53]:

```
avg_yield_query<-"SELECT *FROM CROP_DATA WHERE GEO = 'Saskatchewan' AND YEAR LIKE '200
0%';"
avg_yield_df<-sqlQuery(conn,avg_yield_query)
avg_yield_df

#Barley
```

A data.frame: 0 × 8

CD_ID	YEAR	CROP_TYPE	GEO	SEEDED_AREA	HARVESTED_AREA	PRODUCTION	AVG_YIE
<int>	<chr>	<chr>	<chr>	<int>	<int>	<int>	<i

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 [54]:

```
highest_yield_query<-"SELECT croptype, GEO FROM CROP_DATA WHERE YEAR >= '2000-1-1';"
highest_yield_df<-sqlQuery(conn,highest_yield_query)
highest_yield_df

#Barley, Alberta, 2020-12-31, 3980
```

'42S22 -206 [IBM][CLI Driver][DB2/LINUXX8664] SQL0206N "CROPTYPE" is not valid in the context where it is used. SQLSTATE=42703\n' ·
'[RODBC] ERROR: Could not SQLExecDirect \'SELECT croptype, GEO FROM CROP_DATA WHERE YEAR >= \'2000-1-1\';\'

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 [55]:

```
harvestquery <- "SELECT * FROM CROP_DATA WHERE AND CROP_TYPE = 'WHEAT' AND
YEAR in (SELECT * FROM CROP_DATA WHERE YEAR LIKE '2000%');"
view<-sqlQuery(conn, harvestquery)
view
#35183000
```

'42601 -104 [IBM][CLI Driver][DB2/LINUXX8664] SQL0104N An unexpected token "CROP_TYPE" was found following "CROP_DATA WHERE AND". Expected tokens may include: "<space>". SQLSTATE=42601\n' ·
'[RODBC] ERROR: Could not SQLExecDirect \'SELECT * FROM CROP_DATA WHERE AND CROP_TYPE = \'WHEAT\' AND \nYEAR in (SELECT * FROM CROP_DATA WHERE YEAR LIKE \'2000%\');\'

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 [56]:

```
query<-"SELECT * FROM MONTHLY_FX, FARM_PRICES;"
view<-sqlQuery(conn,query)
head(view)
```

A data.frame: 0 × 8

DFX_ID	DATE	FXUSDCAD	DFX_ID.1	DATE.1	CROPTYPE	GEO	PRICEPERMT
<int>	<chr>	<dbl>	<int>	<chr>	<chr>	<chr>	<dbl>

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 [ ]:

```

```