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 database
3. Execute SQL queries 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)
2. Farm product prices (Data & Metadata)
3. 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 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\ 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 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/

(* 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
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
3. Daily FX Data: 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

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 tables.

Let's first load the RSQLite package:

In [1]:
```
install.packages("RSQLite")
install.packages("odbc")
install.packages("RODBC")
```

```
Updating HTML index of packages in '.Library'
Making 'packages.html' ... done
Updating HTML index of packages in '.Library'
Making 'packages.html' ... done
Warning message:
“package ‘RODBC’ is not available (for R version 3.5.1)”
```

In [2]:
```
library(RSQLite)
library(RODBC)
library(odbc)
```

Problem 1

Create tables

Establish a connection **conn** to the RSQLite database **FinalDB.sqlite**, and create the following four tables.

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

The previous practice lab will help you accomplish this.

In [14]:
```
#  database connection
dsn_driver <- "{IBM DB2 ODBC Driver}"
dsn_database <- "bludb"
dsn_hostname <- "8c6db8df-c8ce-4041-90ca-346ba4dc9f6d.bs2ipr8s07b7bgm1ta80.databases.app
dsn_port <- "32400"
dsn_protocol <- "TCPIP"
dsn_uid <- "0149f402"
dsn_pwd <- "fW9o92RHDF042mmo"
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 2
```

```
Details:
  case=nochange
  DRIVER={IBM DB2 ODBC DRIVER}
  UID=0149f402
  PWD=******
  DATABASE=bludb
  HOSTNAME=8c6db8df-c8ce-4041-90ca-346ba4dc9f6d.bs2ipr8s07b7bgm1ta80.databases.appdomai
n.cloud
  PORT=32400
  PROTOCOL=TCPIP
  SECURITY=SSL
```

In [15]:
```
#check list of tables in the present db.
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 [7]:
```
# Write your query here
# 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")
    }
```

```
An error has occurred.
[1] "42S01 -601 [IBM][CLI Driver][DB2/LINUXX8664] SQL0601N  The name of the object to be
created is identical to the existing name \"0149F402.CROP_DATA\" of type \"TABLE\".  SQL
STATE=42710\n"
```

```
[2] "[RODBC] ERROR: Could not SQLExecDirect 'CREATE TABLE CROP_DATA (\n
                    CD_ID INTEGER NOT NULL,\n                                    YEAR
DATE NOT NULL,\n                                    CROP_TYPE VARCHAR(20) NOT NULL,\n
                                GEO VARCHAR(20) NOT NULL, \n
               SEEDED_AREA INTEGER NOT NULL,\n                                    HARV
ESTED_AREA INTEGER NOT NULL,\n                                    PRODUCTION INTEGER N
OT NULL,\n                                    AVG_YIELD INTEGER NOT NULL,\n
```

```
                                PRIMARY KEY (CD_ID)\n
)'"
```

In [26]:
```
# Define the SQL query for creating the table
query <- "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 DECIMAL NOT NULL,
                PRIMARY KEY (CD_ID)
            )"

# Execute the SQL query to create the table
result <- sqlQuery(conn, query, errors = FALSE)

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

```
Table was created successfully.
```

In [28]:
```
df3 <- sqlQuery(conn,
    "CREATE TABLE DAILY_FX (
        DFX_ID INTEGER NOT NULL,
        DATE DATE NOT NULL,
        FXUSDCAD FLOAT(6) NOT NULL,
        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 [29]:
```
# MONTHLY_FX:
df4 <- sqlQuery(conn,
                    "CREATE TABLE MONTHLY_FX (
                                DFX_ID INTEGER NOT NULL,
                                DATE DATE NOT NULL,
                                FXUSDCAD FLOAT(6) NOT NULL,
                                PRIMARY KEY (DFX_ID)
                                )",
                    errors=FALSE
                    )

    if (df4 == -1){
        shcat ("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 your tables in database

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

In [16]:
```
crop_df <- read.csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
farm_df <- read.csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
monthly_df <- read.csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.clo
daily_df <- read.csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud

head(crop_df)
head(farm_df)
head(monthly_df)
head(daily_df)
```

A data.frame: 6 × 8

	CD_ID	YEAR	CROP_TYPE	GEO	SEEDED_AREA	HARVESTED_AREA	PRODUCTION	AVG_YIELD
	<int>	<chr>	<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 × 5

	CD_ID	DATE	CROP_TYPE	GEO	PRICE_PRERMT
	<int>	<chr>	<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 × 3

	DFX_ID	DATE	FXUSDCAD
	<int>	<chr>	<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>	<chr>	<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 [17]:
```
sqlSave(conn, crop_df, "CROP_DATA", append=TRUE, fast=FALSE, rownames=FALSE, colnames=FA
sqlSave(conn, farm_df, "FARM_PRICES", append=TRUE, fast=FALSE, rownames=FALSE, colnames=
sqlSave(conn, monthly_df, "MONTHLY_FX", append=TRUE, fast=FALSE, rownames=FALSE, colname
sqlSave(conn, daily_df, "DAILY_FX", append=TRUE, fast=FALSE, rownames=FALSE, colnames=FA
```

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

Problem 3

How many records are in the farm prices dataset?

In [19]:
```
# Write your query here
query = "SELECT COUNT(CD_ID) AS RECORDS_IN_DATASET FROM FARM_PRICES"
sqlQuery(conn,query)
```

A data.frame: 1 × 1

	RECORDS_IN_DATASET
	<int>
1	2678

Problem 4

Which geographies are included in the farm prices dataset?

In [20]:
```
# Write your query here
query = "SELECT GEO AS GEOGRAPHIES_IN_DATASET FROM FARM_PRICES
GROUP BY GEO"
sqlQuery(conn,query)
```

A data.frame: 2 × 1

	GEOGRAPHIES_IN_DATASET
	<fct>
1	Alberta
2	Saskatchewan

Problem 5

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

In [21]:
```
# Write your query here
query = "SELECT SUM(HARVESTED_AREA) AS CANADA_HARVESTED_RYE_1968 FROM CROP_DATA
WHERE GEO = 'Canada' AND CROP_TYPE = 'Rye' AND YEAR(YEAR) = 1968"
sqlQuery(conn,query)
```

A data.frame: 1 × 1

	CANADA_HARVESTED_RYE_1968
	<int>
1	274100

Problem 6

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

In [22]:
```
# Wriet your query here
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>	<int>
1	4	1985-01-01	Rye	Alberta	100
2	5	1985-01-01	Rye	Saskatchewan	109
3	10	1985-02-01	Rye	Alberta	95
4	11	1985-02-01	Rye	Saskatchewan	103
5	16	1985-03-01	Rye	Alberta	96
6	17	1985-03-01	Rye	Saskatchewan	106

Problem 7

Which provinces grew Barley?

In [23]:
```
# Write your query here
query = "SELECT GEO AS BARLEY_GROWING_PROVINCES FROM CROP_DATA
WHERE CROP_TYPE='Barley' AND NOT GEO = 'Canada' GROUP BY GEO"
sqlQuery(conn,query)
```

A data.frame: 2 × 1

	BARLEY_GROWING_PROVINCES
	<fct>
1	Alberta
2	Saskatchewan

Problem 8

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

In [24]:
```
# Write your query here
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?

In [25]:
```
# Write your query here
query = "SELECT DISTINCT (CROP_TYPE),
PRICE_PRERMT, YEAR(DATE) AS YEAR FROM FARM_PRICES
WHERE PRICE_PRERMT >= 350 ORDER BY PRICE_PRERMT LIMIT 10"
sqlQuery(conn,query)
```

A data.frame: 10 × 3

	CROP_TYPE	PRICE_PRERMT	YEAR
	<fct>	<int>	<int>
1	Canola	350	1985
2	Canola	350	1995
3	Canola	350	2003
4	Canola	350	2007
5	Canola	351	1999
6	Canola	351	2007
7	Canola	352	1997
8	Canola	352	1998
9	Canola	352	2003
10	Canola	352	2007

Problem 10

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

In [26]:
```
# Write your query here
query = "SELECT CROP_TYPE AS CROP_RANK, MAX(AVG_YIELD)AS AVG_YIELD_YEAR_2000
FROM CROP_DATA
WHERE YEAR(YEAR)=2000 AND GEO='Saskatchewan'
GROUP BY CROP_TYPE
ORDER BY AVG_YIELD_YEAR_2000 desc"
sqlQuery(conn,query)
```

A data.frame: 4 × 2

	CROP_RANK	AVG_YIELD_YEAR_2000
	<fct>	<int>
1	Barley	2800
2	Wheat	2200
3	Rye	2100
4	Canola	1400

In [27]:
```
query = "SELECT CROP_TYPE AS BEST_PERFORMING_CROP, AVG_YIELD AS YIELD
FROM CROP_DATA
WHERE YEAR(YEAR)=2000 AND GEO='Saskatchewan' LIMIT 1"
sqlQuery(conn,query)
```

A data.frame: 1 × 2

	BEST_PERFORMING_CROP	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?

In [28]:
```
# Write your query here
query = "SELECT CROP_TYPE, GEO, AVG_YIELD
FROM CROP_DATA
WHERE YEAR(YEAR) >=2000
ORDER BY AVG_YIELD DESC LIMIT 10"
sqlQuery(conn,query)
```

A data.frame: 10 × 3

	CROP_TYPE	GEO	AVG_YIELD
	<fct>	<fct>	<int>
1	Barley	Alberta	4100
2	Barley	Alberta	4100
3	Barley	Alberta	3980
4	Wheat	Alberta	3900
5	Barley	Alberta	3900
6	Wheat	Alberta	3900
7	Barley	Canada	3900
8	Barley	Alberta	3890
9	Barley	Canada	3820
10	Barley	Canada	3810

In [29]:
```
query = "SELECT CROP_TYPE, GEO, AVG_YIELD AS HIGHEST_AVG_YIELD
FROM CROP_DATA
WHERE YEAR(YEAR) >=2000
ORDER BY AVG_YIELD DESC LIMIT 1"
sqlQuery(conn,query)
```

A data.frame: 1 × 3

	CROP_TYPE	GEO	HIGHEST_AVG_YIELD
	<fct>	<fct>	<int>
1	Barley	Alberta	4100

Problem 12

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

In [30]:
```
# Write your query here
query = "SELECT MAX(YEAR) AS MOST_RECENT_YEAR,
SUM(HARVESTED_AREA) AS TOT_HARVESTED_WHEAT_CANADA
FROM CROP_DATA
WHERE YEAR(YEAR)=(SELECT MAX(YEAR(YEAR))FROM CROP_DATA
WHERE GEO = 'Canada' AND CROP_TYPE  = 'Wheat')"
sqlQuery(conn,query)
```

A data.frame: 1 × 2

	MOST_RECENT_YEAR	TOT_HARVESTED_WHEAT_CANADA
	<date>	<int>
1	2020-12-31	38897100

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.

In [31]:
```
# Write your query here
query = "SELECT A.DATE,A.CROP_TYPE,A.GEO,A.PRICE_PRERMT AS CANADIAN_DOLLARS,
(A.PRICE_PRERMT / B.FXUSDCAD) AS US_DOLLARS, B.FXUSDCAD
FROM FARM_PRICES A , MONTHLY_FX B
WHERE YEAR(A.DATE) = YEAR(B.DATE)
AND MONTH(A.DATE) = MONTH(B.DATE)
AND CROP_TYPE ='Canola'
AND GEO = 'Saskatchewan'
ORDER BY DATE DESC LIMIT 6"
sqlQuery(conn,query)
```

A data.frame: 6 × 6

	DATE	CROP_TYPE	GEO	CANADIAN_DOLLARS	US_DOLLARS	FXUSDCAD
	<date>	<fct>	<fct>	<int>	<dbl>	<dbl>
1	2020-12-01	Canola	Saskatchewan	507	395.8553	1.280771
2	2020-11-01	Canola	Saskatchewan	495	378.7821	1.306820
3	2020-10-01	Canola	Saskatchewan	474	358.6912	1.321471
4	2020-09-01	Canola	Saskatchewan	463	350.0125	1.322810
5	2020-08-01	Canola	Saskatchewan	464	350.9290	1.322205
6	2020-07-01	Canola	Saskatchewan	462	342.2602	1.349850

In [32]:
```
close(conn)
```

Author(s)

Jeff Grossman

D.M. Naidu

Contributor(s)

Rav Ahuja

Lakshmi Holla

Change log

Date	Version	Changed by	Change Description
2022-03-03	1.0	D.M. Naidu	Converted intial version to RSQLite