Week 04: Data Manipulation and SQL with Python

Author: Chris Kennedy Course: MAC 854 Data Analytics for Accountants

In [1]:
```
import pandas as pd
import sqlite3
import re
```

Function for connecting to database. Try/Catch functions are useful for databases to understand if the process executed successfully. This is important as some of the activities are external to Python and aren't guaranteed to be captured back through this console.

In [2]:
```
def create_connection(path):
    connection = None
    try:
        connection = sqlite3.connect(path)
        print("Connection to SQLite DB successful")
    except sqlite3.Error as e:
        print(f"The error '{e}' occured")

    return connection
```

Helper function to interpret accounting numbers (which could be read in as strings) into numerical data. If a money-type class is available, use that over floating point numbers. Not used in this demo but shown for an example function to illustrate regular expressions and functions that can be used with .apply(f(x))

In [3]:
```
def clean_currency(x):
    """ Cleans strings of $ and () to numerical numbers that
    can be converted to a float
    """
    if isinstance(x, str):
        return (re.sub('[(]', '-', re.sub( '[\$,)]', '', x)))
    return (x)
```

Setup a database locally

In [4]:
```
connection = create_connection("localapp.sqlite")
```

```
Connection to SQLite DB successful
```

Read in dataframes for analysis

In [5]:
```
dfBase = pd.read_excel(r'W4 - Wholesale Customer Data.xlsx')
dfTax = pd.read_excel(r'W4 - Regional Tax Rates Data.xlsx')
```

In [6]:
```
dfBase.head(5)
```

Out[6]:

	Channel	Region	Fresh	Dairy	Grocery	Frozen	Cleaning	Deli
0	Restaurant	Other	12669	9656	7561	214	2674	1338
1	Restaurant	Other	7057	9810	9568	1762	3293	1776
2	Restaurant	Other	6353	8808	7684	2405	3516	7844
3	Retail	Other	13265	1196	4221	6404	507	1788
4	Restaurant	Other	22615	5410	7198	3915	1777	5185

In [7]:
```
dfTax.head(5)
```

Out[7]:

	Region	TaxRate
0	Oporto	0.08
1	Lisbon	0.14
2	Other	0.06

Add a primary key for the main table

In [8]:
```
dfBase['Row'] = dfBase.index + 1
```

Load tables into database

In [9]:
```
dfBase.to_sql('WholesaleData', connection, if_exists='replace', index=False)
dfTax.to_sql('TaxRates', connection, if_exists='replace', index=False)
```

Questions

1. In which regions are total expenditures by restaurants on frozen items greater than those for deli items?
2. Provide a rank-ordered list of the top 20 Retail customers based on total sales.
3. Create a query that shows total sales to each customer both before and after tax.

For each of these questions I'll provide an answer using SQL and with Python data manipulation using Pandas dataframes.

Q1: SQL

In which regions are total expenditures by restaurants on frozen items greater than those for deli items?

In [10]:
```
# Question 1: SQL
query = """
SELECT Region, SumFrozen, SumDeli
FROM
 (SELECT Region, sum(Frozen) as SumFrozen, sum(Deli) as SumDeli
  from WholesaleData
  where Channel = 'Restaurant'
  GROUP BY Region)
WHERE SumFrozen > SumDeli;
"""
result_q1_sql = pd.read_sql_query(query, connection)
print(result_q1_sql)
```

```
   Region  SumFrozen  SumDeli
0  Lisbon      46514    33695
1  Oporto      29271    23541
```

Q1: Python

In which regions are total expenditures by restaurants on frozen items greater than those for deli items?

I break this down into a few steps to make the execution easier to understand:

1. Filter on channel = Restaurant
2. Aggregate by Region
3. Filter on results where Frozen > Deli

In [11]:
```
# Question 1: Python
print("Intermediate table:")
filter = dfBase['Channel'] == 'Restaurant'
group_df = dfBase[filter].groupby(['Region']).sum()
print(group_df[['Frozen','Deli']])

print("\nFinal Table:")
filter = group_df['Frozen'] > group_df['Deli']
result_q1_py = group_df[filter]
print(result_q1_py[['Frozen','Deli']])
```

```
Intermediate table:
        Frozen    Deli
Region
Lisbon   46514   33695
Oporto   29271   23541
Other   158886  191752

Final Table:
        Frozen   Deli
Region
Lisbon   46514  33695
Oporto   29271  23541
```

Q2: SQL

Provide a rank-ordered list of the top 20 Retail customers based on total sales.

In [12]:
```
# Q2: SQL
query = """
SELECT Channel, (Fresh + Dairy + Grocery + Frozen + Cleaning + Deli) as TotalSales
FROM WholesaleData
WHERE Channel = 'Retail'
ORDER BY TotalSales Desc
LIMIT 20;
"""
result_q2_sql = pd.read_sql_query(query, connection)
print(result_q2_sql)
```

```
   Channel  TotalSales
0   Retail      190169
1   Retail      185683
2   Retail      130877
3   Retail      105046
4   Retail       97820
5   Retail       90498
6   Retail       78649
7   Retail       73302
8   Retail       73243
9   Retail       70746
10  Retail       70297
11  Retail       69812
12  Retail       68264
13  Retail       65695
14  Retail       65080
15  Retail       64617
16  Retail       62163
17  Retail       58383
18  Retail       57756
19  Retail       57502
```

Q2: Python

Provide a rank-ordered list of the top 20 Retail customers based on total sales.

Steps:

1. Create the calculated column for Total Sales
2. Filter on "Retail"
3. Sort by Total Sales, descending
4. Select first (top) 20 records

In [13]:
```
dfBase['Total Sales'] = dfBase['Fresh'] + dfBase['Dairy'] + dfBase['Grocery'] + dfB
ase['Frozen'] + \
                        dfBase['Cleaning'] + dfBase['Deli']

filter = dfBase['Channel'] == 'Retail'
result_q2_py = dfBase[filter].sort_values(by=['Total Sales'], ascending=False)

print(result_q2_py[['Channel','Region','Total Sales','Row']].head(20))
```

```
    Channel  Region  Total Sales  Row
181  Retail   Other       190169  182
183  Retail   Other       185683  184
325  Retail  Oporto       130877  326
125  Retail   Other       105046  126
284  Retail   Other        97820  285
103  Retail   Other        90498  104
87   Retail   Other        78649   88
435  Retail   Other        73302  436
258  Retail  Lisbon        73243  259
39   Retail   Other        70746   40
259  Retail  Lisbon        70297  260
427  Retail   Other        69812  428
176  Retail   Other        68264  177
282  Retail   Other        65695  283
196  Retail  Lisbon        65080  197
71   Retail   Other        64617   72
239  Retail  Lisbon        62163  240
202  Retail  Lisbon        58383  203
265  Retail  Lisbon        57756  266
382  Retail   Other        57502  383
```

Q3: SQL

Create a query that shows total sales to each customer both before and after tax.

In [14]:
```
query = """
SELECT A.Channel, A.Region,
      (A.Fresh + A.Dairy + A.Grocery + A.Frozen + A.Cleaning + A.Deli) as TotalPreT
axSales,
      B.TaxRate,
      ((A.Fresh + A.Dairy + A.Grocery + A.Frozen + A.Cleaning + A.Deli) * (1 - B.Ta
xRate)) as TotalPostTaxSales
FROM WholesaleData as A, TaxRates as B
WHERE A.Region = B.Region
ORDER BY TotalPostTaxSales DESC;
"""
result_q3_sql = pd.read_sql_query(query, connection)
print(result_q3_sql[['Region','Channel','TotalPreTaxSales','TotalPostTaxSales','Tax
Rate']])
```

```
    Region     Channel  TotalPreTaxSales  TotalPostTaxSales  TaxRate
0    Other  Restaurant            199891          187897.54     0.06
1    Other  Restaurant            192714          181151.16     0.06
2    Other      Retail            190169          178758.86     0.06
3    Other      Retail            185683          174542.02     0.06
4    Other  Restaurant            165881          155928.14     0.06
..     ...         ...               ...                ...      ...
435  Other      Retail              3730            3506.20     0.06
436  Other      Retail              3485            3275.90     0.06
437  Other      Retail              2476            2327.44     0.06
438  Other      Retail              2158            2028.52     0.06
439  Other      Retail               904             849.76     0.06

[440 rows x 5 columns]
```

Q3: Python

Create a query that shows total sales to each customer both before and after tax.

Steps:

1. First merge the tax data into the base dataframe
2. Create the calculated column for after tax sales
3. Order by Total After Tax Sales

In [15]:
```
dfFull = dfBase.merge(dfTax, on='Region')
dfFull['Total After Tax Sales'] = dfFull['Total Sales'] * (1 - dfFull['TaxRate'])
result_q3_py = dfFull.sort_values(by=['Total After Tax Sales'], ascending=False)
print(result_q3_py[['Region','Channel','Total Sales','Total After Tax Sales','TaxRa
te']])
```

```
    Region     Channel  Total Sales  Total After Tax Sales  TaxRate
85   Other  Restaurant       199891              187897.54     0.06
47   Other  Restaurant       192714              181151.16     0.06
181  Other      Retail       190169              178758.86     0.06
183  Other      Retail       185683              174542.02     0.06
61   Other  Restaurant       165881              155928.14     0.06
..     ...         ...          ...                    ...      ...
131  Other      Retail         3730                3506.20     0.06
231  Other      Retail         3485                3275.90     0.06
98   Other      Retail         2476                2327.44     0.06
97   Other      Retail         2158                2028.52     0.06
154  Other      Retail          904                 849.76     0.06

[440 rows x 5 columns]
```

End of Notebook!