Assignment4-3

November 23, 2021

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

[3]:
```
conn = sqlite3.connect('Store.db')
curs = conn.cursor()
curs.execute("PRAGMA foreign_keys=ON;")

# Get some customer data we can use for testing
data = pd.read_csv('./data/Sales_201901.csv', dtype={'zip':str})
```

[1]:
```
data1 = pd.read_csv('./data/Sales_201901.csv', dtype={'zip':str})
data2 = pd.read_csv('./data/Sales_201902.csv', dtype={'zip':str})
data3 = pd.read_csv('./data/Sales_201903.csv', dtype={'zip':str})
data4 = pd.read_csv('./data/Sales_201904.csv', dtype={'zip':str})
data5 = pd.read_csv('./data/Sales_201905.csv', dtype={'zip':str})
data6 = pd.read_csv('./data/Sales_201906.csv', dtype={'zip':str})
data7 = pd.read_csv('./data/Sales_201907.csv', dtype={'zip':str})
data8 = pd.read_csv('./data/Sales_201908.csv', dtype={'zip':str})
data9 = pd.read_csv('./data/Sales_201909.csv', dtype={'zip':str})
data10 = pd.read_csv('./data/Sales_201910.csv', dtype={'zip':str})
data11 = pd.read_csv('./data/Sales_201911.csv', dtype={'zip':str})
data12 = pd.read_csv('./data/Sales_201912.csv', dtype={'zip':str})
data13 = pd.read_csv('./data/Sales_202001.csv', dtype={'zip':str})
data14 = pd.read_csv('./data/Sales_202002.csv', dtype={'zip':str})
data15 = pd.read_csv('./data/Sales_202003.csv', dtype={'zip':str})
data16 = pd.read_csv('./data/Sales_202004.csv', dtype={'zip':str})
data17 = pd.read_csv('./data/Sales_202005.csv', dtype={'zip':str})
data18 = pd.read_csv('./data/Sales_202006.csv', dtype={'zip':str})
data19 = pd.read_csv('./data/Sales_202007.csv', dtype={'zip':str})
data20 = pd.read_csv('./data/Sales_202008.csv', dtype={'zip':str})
data21 = pd.read_csv('./data/Sales_202009.csv', dtype={'zip':str})
data_list = [data1, data2, data3, data4, data5, data6, data7, data8, data9,␣
 ↪data10, data11, data12, data13, data14, data15, data16, data17, data18,␣
 ↪data19, data20, data21]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
```

```
<ipython-input-1-b691aa6b49cb> in <module>
----> 1 data8 = pd.read_csv('./data/Sales_201901.csv', dtype={'zip':str})
      2 data1=pd.read_excel('./AllDivisions.xlsx')
      3 data2=pd.read_excel('./AllRegions.xlsx')
      4 data3=pd.read_excel('./Assignments_ByCust.xlsx')
      5 data4=pd.read_excel('./Assignments_ByDivision.xlsx')

NameError: name 'pd' is not defined
```

[5]:
```
import Store
```

```
data1
data2
data3
data4
data5
data6
data7
data8
data9
data10
data11
data12
data13
data14
data15
data16
data17
data18
data19
data20
data21
```

[6]:
```
Store.Rebuild()
```

[6]:
```
1
```

[8]:
```
#getcustomer id  from test loading sales data
def GetCustomerID(first_name,last_name,address,zip_code):
    '''Function will check if a record for customer exists.
       If so, return the customer id
       If multiple records are found, print a warning and return None
       If no record exists, create one and return the customer id.'''

    sql = """SELECT cust_id
                FROM tCust
                WHERE first_name = ?
                AND last_name = ?
                AND address = ?
                AND zip = ?;"""
    # Make sure to convert zip to string
    cust = pd.read_sql(sql, conn,␣
↪params=(first_name,last_name,address,str(zip_code)))

    # There should only be at most, one result
    if len(cust) > 1:
        print('Found multiple customers: ' + str(len(cust)))
        return None

    # If the customer did not exist, then create it
    if len(cust) == 0:
        sql_insert = """INSERT INTO tCust (first_name,last_name,address,zip)␣
↪VALUES (?,?,?,?);"""
        curs.execute(sql_insert, (first_name,last_name,address,str(zip_code)))
        cust = pd.read_sql(sql, conn,␣
↪params=(first_name,last_name,address,str(zip_code)))

    return cust['cust_id'][0]
```

[9]:
```
##get order id from test loading sales data
def GetOrderID(cust_id, day, month, year):
    # Check to see if an order already exists for this customer/day
    sql_check_order = """SELECT order_id
                            FROM tOrder
                            WHERE cust_id = ?
                            AND day = ?
                            AND month = ?
                            AND year = ?;"""
    order_id = pd.read_sql(sql_check_order, conn,
                          params=(cust_id, day, month, year))

    if len(order_id) == 0:
        # Enter the order
        sql_enter_order = """INSERT INTO tOrder (cust_id, day, month, year)
                            VALUES (?,?,?,?);"""
        curs.execute(sql_enter_order, (cust_id, day, month, year))
        order_id = pd.read_sql(sql_check_order, conn,
                              params =(cust_id, day, month, year))
    elif len(order_id)>1:
        # You might want to make this message a bit more informative
        print('WARNING! Multiple orders found...')
        return None
    else:
        print('Order information for customer ' + str(cust_id) +
                ' on ' + str(day) + '/' + str(month) + '/' + str(year)
                + ' already exists')

    return order_id['order_id'][0]
```

```
[10]: for data in data_list:
          # Append customer ids to the DataFrame
          cust = data[['first','last','addr','city','state','zip']].drop_duplicates()
          cust.head(3)

          cust_id = []
          for row in cust.values:
              cust_id.append(GetCustomerID(row[0], row[1], row[2], row[5]))
          cust['cust_id'] = cust_id
          data_with_cust = data.merge(cust, on=['first','last','addr','zip'])
          data_with_cust
          # Get all the customer id / dates
          orders  = data_with_cust[['cust_id', 'date']].drop_duplicates()
          orders[['year','month','day']] = orders['date'].str.split('-',expand=True)
          order_id = []
          for row in orders.values:
              order_id.append(GetOrderID(row[0], row[4], row[3], row[2]))
          orders['order_id'] = order_id
          data_with_cust_order = data_with_cust.merge(orders, on=['cust_id','date'])
          data_with_cust_order
          # Fill in tOrderDetail
          COL_ORDER_ID = 17
          COL_PROD_ID = 7
          COL_QTY = 10

          sql = "INSERT INTO tOrderDetail VALUES(?,?,?)"
          for row in data_with_cust_order.values:
              curs.execute(sql, (row[COL_ORDER_ID], row[COL_PROD_ID], row[COL_QTY]))
          pd.read_sql("SELECT * FROM tOrderDetail;", conn)
```

import sqlite3 import pandas as pd## DATA 311 - Fall 2020 ### Assignment #4 - Due Friday, October 30 by midnight — Load all of the sales data from the sales_data.zip file provided into our Store database.

- Make sure to start with a fresh, empty copy of the database.
- Destory the sales file we were using for testing in class - only use the new data provided
- Make sure to load the data in chronological order, so that we will all end up with the same values for order_id and cust_id
- The data provided is for all of 2019, and the first 9 months of 2020 (21 files total).
- The data was generated in such a way that our total sales every month are usually, but not always, increasing. You can use this fact as a sanity check to make sure the data was loaded correctly.
- I will be providing new sales data eventually, so make sure the loading process is seamless

and easy, and make sure to thoroughly test it.
- When loading a file, you might want to have your code move that file into a different directory once it is succesfully loaded, so that you don't accidentally try to load it again later. Let me know if you need help with that!

After doing so, answer the following questions:

---

1) Generate a summary, by month and year of how our store is performing.

Have your query return the following: - year - month - Sales: total sales for the month - NumOrders: number of orders placed for the month - NumCust: number of distinct customers who made a purchase (i.e. only count the customer at most once per month) - OrdersPerCust: average number of orders per customer (i.e. NumOrders/NumCust) - SalesPerCust: average sales per customer (i.e. Sales/NumCust) - SalesPerOrder: average sales per order (i.e. Sales/NumOrders)

The results should be grouped and sorted by year and month, in ascending order.

Keep in mind that you have data for all 12 months of 2019, and the first 9 months of 2020, so there should be 21 rows in your results. Also, watch out for integer division!

[225]:
```
pd.read_sql("""SELECT year, month, SUM(qty*unit_price) AS Sales, count(DISTINCT␣
↪order_id) AS NumOrders, count(DISTINCT cust_id) AS NumCust,
              (count(DISTINCT order_id)/count(DISTINCT cust_id)) AS␣
↪OrdersPerCust, (SUM(qty*unit_price)/count(DISTINCT cust_id)) AS␣
↪SalesPerCust,
              (SUM(qty*unit_price)/count(DISTINCT order_id)) AS SalesPerOrder
              FROM tOrder
              JOIN tOrderDetail USING(order_id)
              JOIN tProd USING(prod_id)
              GROUP BY  year, month
              ORDER BY year ASC;""",conn)
```

[225]:
```
    year  month      Sales  NumOrders  NumCust  OrdersPerCust  SalesPerCust  \
0   2019      1   68464.61         91       85              1    805.466000
1   2019      2   55560.32         80       73              1    761.100274
2   2019      3  100491.07        104       85              1   1182.247882
3   2019      4  110661.05        116       95              1   1164.853158
4   2019      5  125623.57        118       97              1   1295.088351
5   2019      6  137173.59        123      109              1   1258.473303
6   2019      7  158080.03        129      103              1   1534.757573
7   2019      8  228577.44        157      126              1   1814.106667
8   2019      9  229094.40        162      126              1   1818.209524
9   2019     10  354471.44        208      151              1   2347.492980
10  2019     11  313319.82        172      135              1   2320.887556
11  2019     12  584023.71        244      170              1   3435.433588
12  2020      1  480742.54        241      166              1   2896.039398
13  2020      2  562773.44        236      172              1   3271.938605
14  2020      3  908514.52        327      209              1   4346.959426
```

```
15  2020      4   743491.76        254       177             1   4200.518418
16  2020      5  1066398.25        314       197             1   5413.189086
17  2020      6  1403832.02        373       228             1   6157.157982
18  2020      7  1542354.28        428       243             1   6347.136955
19  2020      8  2197253.30        459       244             1   9005.136475
20  2020      9   106506.53        107        88             1   1210.301477

    SalesPerOrder
0      752.358352
1      694.504000
2      966.260288
3      953.974569
4     1064.606525
5     1115.232439
6     1225.426589
7     1455.907261
8     1414.162963
9     1704.189615
10    1821.626860
11    2393.539795
12    1994.782324
13    2384.633220
14    2778.331865
15    2927.132913
16    3396.172771
17    3763.624718
18    3603.631495
19    4787.044227
20     995.388131
```

---

2) Get our total sales for all states (50 + DC and PR, so 52 records total) for **January 2019 only**.

Have your query return: - st: The state abbreviation - state: The name of the state - Sales: The total sales in that state

Order the results by the state abbreviation, in ascending order.

Make sure that all states are returned even if they had no sales. In that case, have the query return 0 instead of NaN or Null.

[226]:
```
#Remeber to put an IFNULL STATMENT IN HERE BITch BOI

pd.read_sql("""SELECT st, state, IFNULL(Sales, 0) as Sales
                FROM tState
                LEFT JOIN (SELECT *, SUM(qty*unit_price) as Sales FROM
                    tZip
```

```
                                JOIN tCust USING(zip)
                                JOIN tOrder USING (cust_id)
                                JOIN tOrderDetail USING (order_id)
                                JOIN tProd USING (prod_id)
                                WHERE month==1 AND year==2019
                                GROUP BY st)
                                USING(st)
                        GROUP BY st
                        ORDER BY st;""",conn)
```

[226]:

```
    st                 state    Sales
0   AK                Alaska     0.00
1   AL               Alabama  2476.61
2   AR              Arkansas   597.43
3   AZ               Arizona  1959.23
4   CA            California     0.00
5   CO              Colorado   198.86
6   CT           Connecticut   223.68
7   DC  District of Columbia  1328.35
8   DE              Delaware  1650.19
9   FL               Florida   564.38
10  GA               Georgia     0.00
11  HI                Hawaii  3490.25
12  IA                  Iowa   517.85
13  ID                 Idaho  1683.08
14  IL              Illinois     0.00
15  IN               Indiana   856.41
16  KS                Kansas  6005.12
17  KY              Kentucky     0.00
18  LA             Louisiana  2389.68
19  MA         Massachusetts   902.67
20  MD              Maryland   439.72
21  ME                 Maine   523.48
22  MI              Michigan   191.83
23  MN             Minnesota   117.95
24  MO              Missouri   485.23
25  MS           Mississippi  4093.82
26  MT               Montana   275.90
27  NC        North Carolina   742.37
28  ND          North Dakota  1867.18
29  NE              Nebraska  4120.92
30  NH         New Hampshire    10.99
31  NJ            New Jersey   493.85
32  NM            New Mexico  4517.45
33  NV                Nevada     0.00
34  NY              New York   883.46
35  OH                  Ohio  3009.38
```

```
36  OK          Oklahoma   196.77
37  OR            Oregon     0.00
38  PA      Pennsylvania  4171.87
39  PR       Puerto Rico  4088.15
40  RI      Rhode Island   652.00
41  SC    South Carolina  2372.45
42  SD      South Dakota   298.47
43  TN         Tennessee     0.00
44  TX             Texas  1524.81
45  UT              Utah  3174.09
46  VA          Virginia    71.91
47  VT           Vermont   450.45
48  WA        Washington     0.00
49  WI         Wisconsin  2810.57
50  WV     West Virginia  1284.08
51  WY           Wyoming   751.67
```

---

3) Going back to question 1, you may have noticed that our sales were not very good last month!

Generate a list of all customers who did not place an order last month (September, 2020)

Have your query return:

- cust_id
- NumOrder: a count of the number of orders they placed last month (which should all be zero).

[227]:
```
curs.execute("""DROP VIEW IF EXISTS vCustSept2020;""")
curs.execute("""CREATE VIEW vCustSept2020 AS
                SELECT cust_id FROM tProd
                JOIN tOrderDetail USING (prod_id)
                JOIN tOrder USING (order_id)
                JOIN tCust USING(cust_id)
                WHERE month like '9' and year like '2020'
                GROUP BY cust_id;""")
```

[227]:
```
<sqlite3.Cursor at 0x7fde20851b90>
```

[228]:
```
pd.read_sql("""SELECT cust_id, 0 as NumOrder from tCust
                WHERE cust_id NOT IN vCustSept2020
                GROUP BY cust_id;""", conn)
```

[228]:
```
   cust_id  NumOrder
0        1         0
1        2         0
2        3         0
3        4         0
```

```
4          5          0
..         …          …
217       302         0
218       306         0
219       307         0
220       308         0
221       310         0

[222 rows x 2 columns]
```

---

4) Using the list of customers from the last question, add two new columns to the result containing 1) each customer's average sales for months 1 through 8 of 2020, and 2) their sales for September of 2019. Maybe we'll give that info to our sales team and see if we can do some marketing to those customers.

[15]:
```
curs.execute("""DROP VIEW IF EXISTS vCustSept2020;""")
curs.execute("""CREATE VIEW vCustSept2020 AS
                SELECT cust_id, NumOrders
                FROM tCust
                LEFT JOIN (SELECT *, count(DISTINCT order_id) AS NumOrders FROM␣
 ↪tOrder
                JOIN  tOrderDetail USING (order_id)
                JOIN tProd USING(prod_id)
                WHERE month=9 and year = 2020
                GROUP BY cust_id)
                USING (cust_id)
                WHERE order_id IS NULL
                GROUP BY cust_id;""")
curs.execute("""DROP VIEW IF EXISTS vAveSales2020;""")
curs.execute("""CREATE VIEW vAveSales2020 AS
                SELECT *, Sales2020/8 as AvgSales2020
                FROM tCust
                JOIN (SELECT cust_id, sum(qty*unit_price) as Sales2020 FROM␣
 ↪tOrder
                JOIN tOrderDetail USING (order_id)
                JOIN tProd USING (prod_id)
                WHERE cust_id IN (SELECT cust_id FROM vCustSept2020) AND year =␣
 ↪2020 and month IS NOT 9
                GROUP BY cust_id) USING (cust_id)
                GROUP BY cust_id;""")
curs.execute("""DROP VIEW IF EXISTS vSalesSept2019;""")
curs.execute("""CREATE VIEW vSalesSept2019 as
                SELECT *, SalesSept2019
                FROM tCust
                JOIN (SELECT cust_id, sum(qty*unit_price) as SalesSept2019 FROM␣
 ↪tOrder
```

```
                JOIN tOrderDetail USING (order_id)
                JOIN tProd USING (prod_id)
                WHERE cust_id IN (SELECT cust_id FROM vCustSept2020) AND year =␣
 ↪2019 and month = 9
                GROUP BY cust_id) USING (cust_id)
                GROUP BY cust_id;""")

pd.read_sql("""SELECT cust_id, IFNULL(NumOrders, 0) as NumOrders, AvgSales2020,␣
 ↪IFNULL(SalesSept2019, 0) AS SalesSept2019
                FROM vCustSept2020
                JOIN vAveSales2020 USING (cust_id)
                LEFT JOIN vSalesSept2019 USING (cust_id)
                GROUP BY cust_id;""", conn)
```

[15]:

```
     cust_id  NumOrders  AvgSales2020  SalesSept2019
0          1          0    1224.31000         208.42
1          2          0    4026.06875         358.50
2          3          0    3430.65250         825.95
3          4          0    7942.77375           0.00
4          5          0    6680.17500           0.00
..       …          …           …             …
217      302          0    3404.07750           0.00
218      306          0    6023.86125           0.00
219      307          0    5830.98000           0.00
220      308          0    4375.47000           0.00
221      310          0    2514.14500           0.00

[222 rows x 4 columns]
```

---

5) What is our top selling product (in terms of dollars) so far?

Have your query return:

- prod_id
- prod_name
- total quantity sold, based on all the data we have in the database
- total sales, based on all the current data in the database

[14]:

```
pd.read_sql("""SELECT prod_id, prod_name, sum(qty) AS TotalQty, qty*unit_price␣
 ↪as Sales
                    FROM tProd
                    JOIN tOrderDetail USING (prod_id)
                    GROUP BY prod_id
                    ORDER BY Sales DESC
                    LIMIT 1;""", conn)

#tProd has prod_id and unit_price
```

```
#tOrderDetail has prod_id and qty
```

```
[14]:    prod_id prod_name  TotalQty    Sales
      0      329  Chainsaw      8951  3999.92
```

```
[ ]: # Don't forget to close your connection when done!
     conn.close()
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```