ETL and OLAP with Sales Analysis using PostgreSQL

Author: Yang Liu

In this project, I expressed “complex” OLAP queries in SQL. The key point of the project is to observe a large gap between the complexity of expressing the type of such queries and that of evaluating them (such as writing Java programs to produce the same results). My mission (in addition to writing the SQL queries) is to consider the reasons for the gap (between the expression and evaluation of such queries) and how to narrow it.

Import the library

In [1]:
```python
import psycopg2
import pandas as pd
import re
import warnings
```

In [2]:
```python
pd.set_option('display.width', 1000)
warnings.filterwarnings("ignore")
```

Create a connection to the database

In [3]:
```python
try:
    conn = psycopg2.connect("host=127.0.0.1 dbname=CS-561 user=postgres
 password=xxxxxx")
except psycopg2.Error as e:
    print("Error: Could not make connection to the Postgres database")
    print(e)
```

Use the connection to get a cursor that can be used to execute queries.

In [4]:
```
try:
    cur = conn.cursor()
except psycopg2.Error as e:
    print("Error: Could not get curser to the Database")
    print(e)
```

In [5]:
```
# Set automatic commit to be true so that each action is committed
#without having to call conn.commit() after each command.
conn.set_session(autocommit=True)
```

Create a database to do the work in.

In [6]:
```
try:
    cur.execute("CREATE DATABASE testing")
except psycopg2.Error as e:
    print(e)
```

```
database "testing" already exists
```

In [7]:
```
# close our connection to the testing database
try:
    conn.close()
except psycopg2.Error as e:
    print(e)

# reconnect to the CS-561 database, and get a new cursor.
try:
    conn = psycopg2.connect("host=127.0.0.1 dbname=CS-561 user=postgres
 password=xxxxxx")
except psycopg2.Error as e:
    print("Error: Could not make connection to the CS-561 database")
    print(e)

try:
    cur = conn.cursor()
except psycopg2.Error as e:
    print("Error: Could not get cursor to the Database")
    print(e)

conn.set_session(autocommit=True)
```

Create a Sales Table that contains a combinenation of customer, product, month, transaction day, transaction month, transaction year, transaction state, and transaction quantity.

In [8]:
```
#Clean up the database
try:
    cur.execute("""DROP VIEW IF EXISTS avg_q, month_prod,
                current_month,this_month, comb, combination, v2, v3,bas
e;""")
except psycopg2.Error as e:
    print("Error: Issue dropping views")
    print (e)
```

In [9]:
```
# TO-DO: Finish writing the CREATE TABLE statement with the correct argu
ments
try:
    cur.execute("DROP TABLE IF EXISTS sales;")
except psycopg2.Error as e:
    print("Error: Issue creating table")
    print(e)

try:
    cur.execute("""CREATE TABLE IF NOT EXISTS sales (cust varchar(20),pr
od varchar(20),
                   day int,month int,year int,state char(2),quant intege
r);""")
except psycopg2.Error as e:
    print("Error: Issue creating table")
    print(e)
```

Insert the sales data into the table

In [10]:
```
#Read data from external resource
raw_data = pd.read_table('sales.txt',names = ['query'])
print(raw_data.shape)
raw_data.head()
```

```
(500, 1)
```

Out[10]:

	query
0	insert into sales values ('Bloom', 'Pepsi', 2,...
1	insert into sales values ('Knuth', 'Bread', 23...
2	insert into sales values ('Emily', 'Pepsi', 22...
3	insert into sales values ('Emily', 'Fruits', 1...
4	insert into sales values ('Helen', 'Milk', 7, ...

In [11]:
```
#Parse the data to insert
def extract(text):
    return re.findall(r"\w+\s\w+\s\w+\s\w+\s\W+(\w+)\W+(\w+)\W+(\w+)\W+
(\w+)\W+(\w+)\W+(\w+)\W+(\w+)",text)[0]

clean_data = raw_data['query'].map(lambda x: extract(x)).tolist()
records_to_insert = [(x[0], x[1], int(x[2]), int(x[3]), \
                      int(x[4]), x[5], int(x[6])) for x in clean_data]
```

In [12]:
```
print("The number of instances to insert: ",len(records_to_insert))
print("\nThe data after processed: ")
records_to_insert
```

```
The number of instances to insert:  500

The data after processed:
```

```
Out[12]: [('Bloom', 'Pepsi', 2, 12, 2001, 'NY', 4232),
          ('Knuth', 'Bread', 23, 5, 2005, 'PA', 4167),
          ('Emily', 'Pepsi', 22, 1, 2006, 'CT', 4404),
          ('Emily', 'Fruits', 11, 1, 2000, 'NJ', 4369),
          ('Helen', 'Milk', 7, 11, 2006, 'CT', 210),
          ('Emily', 'Soap', 2, 4, 2002, 'CT', 2549),
          ('Bloom', 'Eggs', 30, 11, 2000, 'NJ', 559),
          ('Bloom', 'Yogurt', 25, 7, 2004, 'PA', 17),
          ('Helen', 'Pepsi', 14, 3, 2002, 'NJ', 3891),
          ('Emily', 'Bread', 28, 9, 2005, 'PA', 42),
          ('Sam', 'Cookies', 20, 11, 2004, 'NY', 3376),
          ('Knuth', 'Milk', 5, 2, 2007, 'PA', 126),
          ('Helen', 'Coke', 11, 4, 2001, 'NY', 668),
          ('Emily', 'Butter', 5, 7, 2005, 'NJ', 3840),
          ('Emily', 'Yogurt', 7, 10, 2005, 'NY', 730),
          ('Sam', 'Soap', 12, 2, 2001, 'NJ', 165),
          ('Knuth', 'Coke', 6, 1, 2003, 'CT', 1557),
          ('Sam', 'Milk', 9, 8, 2001, 'NY', 1132),
          ('Helen', 'Yogurt', 6, 2, 2000, 'PA', 4001),
          ('Sam', 'Milk', 6, 1, 2003, 'PA', 2298),
          ('Knuth', 'Pepsi', 21, 12, 2008, 'CT', 653),
          ('Knuth', 'Eggs', 19, 12, 2006, 'NJ', 1339),
          ('Helen', 'Coke', 19, 10, 2002, 'NJ', 2662),
          ('Knuth', 'Milk', 25, 3, 2001, 'CT', 58),
          ('Bloom', 'Yogurt', 3, 4, 2001, 'NJ', 1203),
          ('Helen', 'Milk', 6, 11, 2001, 'NY', 2422),
          ('Knuth', 'Yogurt', 22, 4, 2005, 'CT', 301),
          ('Helen', 'Pepsi', 24, 9, 2004, 'CT', 2768),
          ('Helen', 'Fruits', 28, 6, 2005, 'PA', 2811),
          ('Knuth', 'Yogurt', 18, 9, 2001, 'NY', 3705),
          ('Helen', 'Pepsi', 1, 10, 2005, 'PA', 1407),
          ('Helen', 'Cookies', 28, 8, 2002, 'NY', 3494),
          ('Helen', 'Cookies', 23, 12, 2004, 'CT', 1194),
          ('Emily', 'Butter', 25, 7, 2000, 'PA', 4008),
          ('Knuth', 'Fruits', 4, 2, 2005, 'PA', 2430),
          ('Emily', 'Cookies', 3, 11, 2004, 'NJ', 2497),
          ('Bloom', 'Coke', 7, 2, 2000, 'NY', 1229),
          ('Emily', 'Bread', 4, 6, 2001, 'CT', 1626),
          ('Helen', 'Soap', 21, 7, 2004, 'PA', 4107),
          ('Helen', 'Coke', 10, 2, 2002, 'NJ', 4237),
          ('Emily', 'Soap', 4, 9, 2003, 'CT', 70),
          ('Knuth', 'Soap', 29, 5, 2001, 'PA', 2961),
          ('Knuth', 'Cookies', 12, 6, 2007, 'NJ', 2615),
          ('Knuth', 'Coke', 9, 5, 2005, 'NY', 1946),
          ('Helen', 'Milk', 10, 5, 2001, 'PA', 1457),
          ('Emily', 'Milk', 3, 9, 2008, 'NJ', 3839),
          ('Helen', 'Yogurt', 24, 2, 2004, 'NY', 900),
          ('Bloom', 'Coke', 6, 10, 2008, 'PA', 2867),
          ('Bloom', 'Soap', 6, 1, 2005, 'CT', 4623),
          ('Helen', 'Fruits', 25, 12, 2006, 'CT', 4015),
          ('Knuth', 'Fruits', 19, 4, 2007, 'NY', 2740),
          ('Emily', 'Eggs', 3, 11, 2006, 'PA', 1060),
          ('Bloom', 'Milk', 8, 9, 2000, 'NJ', 1106),
          ('Bloom', 'Milk', 19, 4, 2003, 'NY', 3516),
          ('Emily', 'Bread', 23, 12, 2005, 'NY', 2071),
          ('Sam', 'Yogurt', 28, 1, 2005, 'NJ', 2293),
          ('Sam', 'Soap', 23, 11, 2005, 'PA', 399),
          ('Emily', 'Fruits', 31, 8, 2005, 'PA', 3913),
          ('Emily', 'Yogurt', 28, 3, 2005, 'NY', 3822),
          ('Sam', 'Fruits', 15, 9, 2006, 'NJ', 4793),
          ('Bloom', 'Soap', 7, 6, 2005, 'PA', 3404),
          ('Emily', 'Fruits', 14, 12, 2003, 'NY', 3864),
          ('Helen', 'Pepsi', 14, 12, 2005, 'NY', 3002),
          ('Sam', 'Yogurt', 23, 6, 2006, 'CT', 363),
          ('Bloom', 'Milk', 15, 10, 2008, 'CT', 4870),
          ('Sam', 'Soap', 20, 11, 2004, 'NJ', 989),
          ('Sam', 'Butter', 20, 10, 2006, 'NY', 1966),
          ('Sam', 'Fruits', 15, 10, 2005, 'CT', 2802),
          ('Bloom', 'Fruits', 26, 11, 2006, 'NJ', 1832),
          ('Emily', 'Cookies', 30, 7, 2005, 'NY', 1844),
          ('Bloom', 'Fruits', 29, 7, 2005, 'NY', 3653),
          ('Helen', 'Milk', 6, 1, 2003, 'CT', 4429),
          ('Knuth', 'Fruits', 16, 9, 2003, 'PA', 1313),
          ('Sam', 'Milk', 7, 4, 2001, 'CT', 811),
          ('Helen', 'Pepsi', 23, 4, 2007, 'CT', 3609),
          ('Emily', 'Cookies', 9, 6, 2002, 'CT', 4925),
          ('Knuth', 'Soap', 16, 12, 2001, 'NY', 4307),
          ('Bloom', 'Fruits', 9, 4, 2000, 'PA', 570),
          ('Emily', 'Bread', 17, 11, 2001, 'PA', 2565),
          ('Helen', 'Eggs', 17, 4, 2006, 'CT', 1807),
          ('Emily', 'Milk', 12, 11, 2002, 'NY', 4986),
          ('Knuth', 'Pepsi', 6, 9, 2006, 'CT', 3760),
          ('Knuth', 'Cookies', 17, 7, 2003, 'CT', 4994),
          ('Sam', 'Pepsi', 13, 9, 2002, 'NJ', 3877),
          ('Helen', 'Eggs', 7, 2, 2003, 'PA', 813),
          ('Helen', 'Butter', 21, 7, 2001, 'NJ', 2086),
          ('Helen', 'Bread', 21, 5, 2000, 'NY', 389),
          ('Knuth', 'Soap', 13, 4, 2008, 'CT', 1160),
          ('Bloom', 'Butter', 16, 8, 2003, 'NY', 1054),
          ('Sam', 'Bread', 13, 11, 2008, 'PA', 93),
          ('Emily', 'Soap', 10, 10, 2000, 'NJ', 4081),
          ('Sam', 'Eggs', 20, 9, 2008, 'PA', 964),
          ('Helen', 'Fruits', 27, 2, 2007, 'CT', 442),
          ('Helen', 'Yogurt', 19, 1, 2007, 'NJ', 2096),
          ('Sam', 'Eggs', 14, 3, 2001, 'NY', 1744),
          ('Emily', 'Cookies', 27, 11, 2006, 'NJ', 3719),
          ('Emily', 'Fruits', 3, 11, 2000, 'NY', 859),
          ('Helen', 'Fruits', 3, 5, 2006, 'NY', 2774),
          ('Emily', 'Fruits', 6, 10, 2000, 'NY', 3027),
          ('Emily', 'Milk', 5, 4, 2005, 'PA', 3211),
          ('Knuth', 'Butter', 1, 4, 2007, 'NY', 839),
          ('Helen', 'Butter', 30, 3, 2008, 'PA', 1786),
          ('Bloom', 'Yogurt', 6, 10, 2000, 'NJ', 2788),
          ('Helen', 'Bread', 11, 10, 2007, 'NJ', 2513),
          ('Emily', 'Cookies', 28, 10, 2005, 'NY', 1656),
          ('Emily', 'Bread', 4, 9, 2000, 'PA', 1572),
          ('Sam', 'Bread', 18, 1, 2002, 'NJ', 1378),
          ('Emily', 'Pepsi', 14, 1, 2004, 'NJ', 2591),
          ('Knuth', 'Fruits', 6, 7, 2001, 'CT', 3756),
          ('Bloom', 'Soap', 3, 11, 2002, 'NJ', 4364),
          ('Sam', 'Milk', 7, 11, 2000, 'CT', 1607),
          ('Emily', 'Yogurt', 9, 6, 2006, 'NY', 3174),
          ('Emily', 'Butter', 23, 2, 2003, 'CT', 212),
          ('Helen', 'Butter', 1, 1, 2006, 'NY', 4281),
          ('Helen', 'Bread', 26, 5, 2006, 'CT', 1406),
          ('Helen', 'Bread', 15, 6, 2003, 'NJ', 1687),
          ('Bloom', 'Pepsi', 26, 5, 2001, 'NJ', 2367),
          ('Helen', 'Pepsi', 23, 6, 2000, 'PA', 1456),
          ('Emily', 'Eggs', 3, 9, 2003, 'NJ', 3126),
          ('Helen', 'Butter', 26, 11, 2006, 'NY', 1685),
          ('Bloom', 'Milk', 31, 10, 2003, 'PA', 553),
          ('Helen', 'Pepsi', 20, 10, 2002, 'NJ', 3817),
          ('Sam', 'Bread', 7, 7, 2007, 'PA', 872),
          ('Sam', 'Pepsi', 16, 3, 2008, 'NJ', 3014),
          ('Bloom', 'Fruits', 30, 1, 2003, 'PA', 3623),
          ('Sam', 'Pepsi', 16, 11, 2003, 'NJ', 3694),
          ('Emily', 'Yogurt', 9, 3, 2003, 'NJ', 2336),
          ('Knuth', 'Coke', 23, 2, 2003, 'PA', 4757),
          ('Bloom', 'Yogurt', 11, 9, 2001, 'NJ', 3172),
          ('Bloom', 'Yogurt', 17, 9, 2003, 'CT', 4004),
          ('Knuth', 'Bread', 16, 2, 2007, 'CT', 4127),
          ('Knuth', 'Cookies', 23, 12, 2008, 'NY', 1485),
          ('Bloom', 'Fruits', 31, 8, 2003, 'NJ', 4094),
          ('Helen', 'Soap', 22, 3, 2008, 'CT', 2203),
          ('Bloom', 'Milk', 20, 1, 2008, 'PA', 2220),
          ('Helen', 'Yogurt', 8, 8, 2007, 'NY', 4909),
          ('Emily', 'Cookies', 22, 7, 2007, 'NJ', 1827),
          ('Emily', 'Eggs', 8, 1, 2006, 'NJ', 4535),
          ('Emily', 'Bread', 21, 3, 2001, 'NY', 492),
          ('Helen', 'Yogurt', 20, 6, 2004, 'CT', 3832),
          ('Emily', 'Eggs', 19, 4, 2001, 'PA', 4658),
          ('Emily', 'Cookies', 2, 1, 2002, 'PA', 3238),
          ('Knuth', 'Eggs', 12, 2, 2007, 'NY', 4313),
          ('Helen', 'Fruits', 5, 6, 2003, 'NJ', 415),
          ('Emily', 'Milk', 9, 6, 2003, 'PA', 1274),
          ('Helen', 'Butter', 1, 11, 2006, 'PA', 278),
          ('Bloom', 'Cookies', 8, 6, 2006, 'NJ', 3887),
          ('Emily', 'Butter', 22, 8, 2004, 'NJ', 1312),
          ('Emily', 'Butter', 20, 2, 2007, 'CT', 1432),
          ('Helen', 'Coke', 15, 7, 2008, 'CT', 3198),
          ('Helen', 'Butter', 11, 4, 2000, 'NJ', 1712),
          ('Knuth', 'Yogurt', 16, 5, 2000, 'PA', 3396),
          ('Helen', 'Butter', 2, 1, 2000, 'NJ', 728),
          ('Helen', 'Fruits', 30, 3, 2007, 'CT', 2525),
          ('Bloom', 'Pepsi', 7, 8, 2006, 'NY', 254),
          ('Helen', 'Pepsi', 13, 2, 2001, 'NJ', 1983),
          ('Helen', 'Soap', 25, 10, 2006, 'NJ', 712),
          ('Knuth', 'Pepsi', 3, 2, 2005, 'NY', 3879),
          ('Emily', 'Milk', 5, 7, 2000, 'PA', 2125),
          ('Bloom', 'Coke', 13, 7, 2001, 'PA', 3654),
          ('Emily', 'Eggs', 8, 7, 2007, 'NJ', 3030),
          ('Bloom', 'Cookies', 20, 6, 2005, 'CT', 1676),
          ('Helen', 'Butter', 17, 2, 2002, 'PA', 566),
          ('Emily', 'Soap', 26, 10, 2000, 'PA', 742),
          ('Bloom', 'Fruits', 14, 8, 2000, 'NY', 3798),
          ('Knuth', 'Bread', 2, 11, 2002, 'NJ', 1013),
          ('Knuth', 'Soap', 12, 11, 2007, 'NY', 3063),
          ('Helen', 'Milk', 28, 2, 2006, 'NY', 4249),
          ('Helen', 'Eggs', 13, 5, 2000, 'CT', 1483),
          ('Emily', 'Eggs', 23, 7, 2007, 'NY', 2012),
          ('Bloom', 'Fruits', 15, 11, 2005, 'CT', 4020),
          ('Emily', 'Eggs', 26, 11, 2002, 'PA', 128),
          ('Bloom', 'Fruits', 13, 4, 2004, 'PA', 2692),
          ('Bloom', 'Milk', 15, 12, 2006, 'NJ', 4616),
          ('Sam', 'Eggs', 31, 1, 2005, 'PA', 4610),
          ('Sam', 'Coke', 23, 1, 2006, 'CT', 2672),
          ('Sam', 'Fruits', 16, 12, 2005, 'PA', 4644),
          ('Sam', 'Butter', 22, 12, 2000, 'NJ', 1715),
          ('Emily', 'Coke', 22, 11, 2003, 'CT', 81),
          ('Knuth', 'Yogurt', 9, 6, 2001, 'NJ', 2167),
          ('Bloom', 'Fruits', 25, 3, 2004, 'PA', 699),
          ('Emily', 'Fruits', 5, 4, 2000, 'PA', 1573),
          ('Sam', 'Eggs', 11, 7, 2004, 'NY', 1417),
          ('Helen', 'Coke', 25, 5, 2008, 'CT', 2554),
          ('Bloom', 'Milk', 27, 1, 2007, 'NY', 4583),
          ('Sam', 'Milk', 28, 12, 2007, 'NY', 1164),
          ('Bloom', 'Cookies', 7, 7, 2003, 'CT', 4546),
          ('Helen', 'Milk', 28, 2, 2006, 'PA', 4794),
          ('Knuth', 'Pepsi', 2, 3, 2002, 'NJ', 2548),
          ('Bloom', 'Pepsi', 30, 11, 2008, 'NJ', 4515),
          ('Helen', 'Butter', 7, 7, 2007, 'PA', 3859),
          ('Sam', 'Soap', 19, 6, 2007, 'NY', 1798),
          ('Helen', 'Coke', 4, 5, 2005, 'CT', 3345),
          ('Bloom', 'Milk', 16, 10, 2001, 'NJ', 2004),
          ('Emily', 'Butter', 16, 11, 2004, 'CT', 2862),
          ('Bloom', 'Bread', 4, 9, 2008, 'NJ', 1698),
          ('Sam', 'Soap', 6, 1, 2003, 'CT', 2461),
          ('Emily', 'Pepsi', 15, 6, 2008, 'NY', 538),
          ('Helen', 'Yogurt', 22, 8, 2006, 'NY', 787),
          ('Sam', 'Yogurt', 22, 1, 2008, 'PA', 2142),
          ('Sam', 'Butter', 14, 12, 2007, 'NJ', 3019),
          ('Sam', 'Pepsi', 19, 12, 2002, 'CT', 4865),
          ('Bloom', 'Fruits', 11, 1, 2004, 'PA', 3006),
          ('Emily', 'Yogurt', 16, 8, 2000, 'CT', 557),
          ('Sam', 'Eggs', 12, 2, 2006, 'NJ', 1091),
          ('Knuth', 'Eggs', 8, 4, 2006, 'PA', 326),
          ('Bloom', 'Yogurt', 7, 12, 2006, 'PA', 783),
          ('Emily', 'Soap', 15, 11, 2003, 'CT', 4106),
          ('Sam', 'Eggs', 7, 11, 2007, 'PA', 4674),
          ('Sam', 'Milk', 7, 4, 2004, 'CT', 385),
          ('Bloom', 'Yogurt', 28, 3, 2008, 'NJ', 1660),
          ('Knuth', 'Coke', 9, 4, 2002, 'CT', 4027),
          ('Emily', 'Cookies', 9, 2, 2007, 'PA', 2500),
          ('Sam', 'Yogurt', 22, 6, 2001, 'CT', 4728),
          ('Knuth', 'Soap', 23, 11, 2002, 'CT', 2915),
          ('Bloom', 'Milk', 4, 12, 2004, 'CT', 1639),
          ('Bloom', 'Pepsi', 10, 7, 2000, 'CT', 94),
          ('Sam', 'Milk', 4, 1, 2008, 'NY', 987),
          ('Sam', 'Yogurt', 21, 5, 2008, 'PA', 4564),
          ('Helen', 'Eggs', 21, 6, 2004, 'NY', 3200),
          ('Knuth', 'Pepsi', 29, 8, 2001, 'NJ', 277),
          ('Knuth', 'Butter', 2, 10, 2007, 'NY', 156),
          ('Sam', 'Soap', 11, 6, 2005, 'NJ', 1883),
          ('Sam', 'Pepsi', 26, 3, 2005, 'PA', 2841),
          ('Knuth', 'Pepsi', 29, 4, 2002, 'CT', 1481),
          ('Sam', 'Yogurt', 2, 9, 2007, 'NY', 1317),
          ('Helen', 'Milk', 19, 9, 2002, 'PA', 3314),
          ('Knuth', 'Coke', 23, 6, 2007, 'CT', 2660),
          ('Sam', 'Coke', 21, 6, 2003, 'NY', 142),
          ('Helen', 'Fruits', 17, 12, 2004, 'CT', 268),
          ('Bloom', 'Fruits', 24, 3, 2004, 'NY', 1268),
          ('Bloom', 'Eggs', 17, 5, 2005, 'CT', 4177),
          ('Helen', 'Pepsi', 18, 1, 2006, 'NJ', 2232),
          ('Helen', 'Butter', 21, 4, 2004, 'NY', 1978),
          ('Helen', 'Butter', 11, 9, 2000, 'NJ', 803),
          ('Helen', 'Milk', 9, 4, 2000, 'NY', 77),
          ('Emily', 'Pepsi', 5, 5, 2001, 'PA', 1682),
          ('Emily', 'Yogurt', 29, 5, 2002, 'CT', 2126),
          ('Emily', 'Yogurt', 4, 8, 2004, 'CT', 3765),
          ('Bloom', 'Cookies', 8, 10, 2002, 'NY', 4570),
          ('Bloom', 'Soap', 10, 4, 2007, 'PA', 1216),
          ('Knuth', 'Soap', 18, 11, 2004, 'PA', 586),
          ('Bloom', 'Bread', 13, 3, 2006, 'NY', 3792),
          ('Emily', 'Coke', 29, 2, 2001, 'NY', 4258),
          ('Bloom', 'Eggs', 24, 1, 2000, 'NY', 666),
          ('Knuth', 'Fruits', 5, 7, 2002, 'NJ', 733),
          ('Sam', 'Bread', 11, 4, 2006, 'NY', 2282),
          ('Knuth', 'Coke', 8, 5, 2004, 'NJ', 1583),
          ('Helen', 'Pepsi', 13, 4, 2006, 'NY', 1931),
          ('Knuth', 'Bread', 9, 10, 2002, 'CT', 1796),
          ('Knuth', 'Pepsi', 17, 6, 2008, 'PA', 2390),
          ('Knuth', 'Eggs', 2, 11, 2004, 'CT', 4317),
          ('Sam', 'Milk', 6, 7, 2005, 'NY', 3120),
          ('Sam', 'Cookies', 8, 2, 2008, 'CT', 2523),
          ('Sam', 'Pepsi', 27, 7, 2008, 'NY', 3189),
          ('Sam', 'Eggs', 11, 5, 2007, 'NY', 3363),
          ('Bloom', 'Bread', 4, 7, 2006, 'CT', 1494),
          ('Helen', 'Butter', 18, 8, 2001, 'NY', 163),
          ('Emily', 'Eggs', 10, 11, 2004, 'CT', 2722),
          ('Sam', 'Coke', 20, 5, 2007, 'CT', 2819),
          ('Emily', 'Milk', 2, 4, 2008, 'CT', 3257),
          ('Bloom', 'Cookies', 9, 9, 2005, 'PA', 1858),
          ('Sam', 'Fruits', 2, 11, 2004, 'NY', 4738),
          ('Knuth', 'Soap', 17, 4, 2008, 'CT', 436),
          ('Bloom', 'Milk', 19, 10, 2008, 'PA', 4377),
          ('Bloom', 'Cookies', 8, 10, 2000, 'CT', 3545),
          ('Emily', 'Fruits', 5, 11, 2001, 'NJ', 3743),
          ('Sam', 'Soap', 28, 8, 2000, 'PA', 2956),
          ('Sam', 'Bread', 29, 8, 2000, 'CT', 1599),
          ('Knuth', 'Fruits', 23, 12, 2008, 'NJ', 3939),
          ('Knuth', 'Yogurt', 14, 3, 2007, 'PA', 4361),
          ('Knuth', 'Soap', 9, 2, 2005, 'NJ', 657),
          ('Sam', 'Coke', 24, 8, 2004, 'PA', 3053),
          ('Sam', 'Pepsi', 16, 4, 2003, 'CT', 160),
          ('Emily', 'Coke', 13, 2, 2000, 'CT', 2909),
          ('Emily', 'Butter', 13, 2, 2001, 'NY', 2826),
          ('Bloom', 'Soap', 5, 7, 2002, 'CT', 387),
          ('Emily', 'Milk', 14, 3, 2005, 'CT', 2589),
          ('Sam', 'Soap', 6, 4, 2003, 'PA', 4954),
          ('Helen', 'Eggs', 13, 6, 2000, 'PA', 2817),
          ('Bloom', 'Coke', 14, 1, 2008, 'CT', 928),
          ('Emily', 'Cookies', 27, 1, 2001, 'CT', 3654),
          ('Knuth', 'Cookies', 7, 10, 2001, 'PA', 1367),
          ('Bloom', 'Fruits', 9, 7, 2002, 'NY', 612),
          ('Helen', 'Yogurt', 2, 12, 2005, 'NY', 1128),
          ('Bloom', 'Bread', 22, 1, 2003, 'CT', 2220),
          ('Knuth', 'Coke', 13, 12, 2007, 'PA', 2928),
          ('Bloom', 'Fruits', 5, 5, 2002, 'PA', 4444),
          ('Knuth', 'Fruits', 12, 1, 2002, 'NJ', 1368),
          ('Knuth', 'Cookies', 6, 6, 2003, 'NJ', 3683),
          ('Bloom', 'Eggs', 22, 9, 2006, 'NY', 3150),
          ('Sam', 'Butter', 27, 2, 2006, 'PA', 673),
          ('Helen', 'Bread', 24, 1, 2002, 'CT', 3530),
          ('Bloom', 'Pepsi', 1, 6, 2003, 'NJ', 1368),
          ('Sam', 'Soap', 4, 1, 2003, 'NJ', 2356),
          ('Knuth', 'Pepsi', 2, 6, 2002, 'NY', 3936),
          ('Emily', 'Coke', 6, 11, 2008, 'NY', 1917),
          ('Knuth', 'Fruits', 26, 8, 2001, 'NY', 24),
          ('Sam', 'Cookies', 22, 12, 2008, 'PA', 1657),
          ('Emily', 'Fruits', 28, 4, 2005, 'NY', 2102),
          ('Helen', 'Butter', 10, 2, 2005, 'NJ', 3288),
          ('Emily', 'Coke', 15, 2, 2007, 'NY', 1330),
          ('Knuth', 'Eggs', 6, 2, 2006, 'NY', 3788),
          ('Emily', 'Cookies', 3, 11, 2008, 'NJ', 173),
          ('Sam', 'Cookies', 9, 10, 2007, 'CT', 132),
          ('Bloom', 'Bread', 14, 9, 2007, 'CT', 3261),
          ('Emily', 'Bread', 2, 4, 2005, 'NJ', 4467),
          ('Knuth', 'Coke', 17, 8, 2007, 'PA', 2812),
          ('Emily', 'Cookies', 19, 5, 2001, 'NY', 2685),
          ('Knuth', 'Pepsi', 11, 9, 2004, 'NY', 4942),
          ('Knuth', 'Pepsi', 12, 8, 2004, 'CT', 1244),
          ('Bloom', 'Soap', 17, 9, 2002, 'PA', 270),
          ('Emily', 'Cookies', 1, 10, 2000, 'NJ', 658),
          ('Helen', 'Soap', 3, 9, 2007, 'NJ', 2934),
          ('Knuth', 'Soap', 16, 1, 2001, 'CT', 1098),
          ('Knuth', 'Eggs', 9, 4, 2003, 'CT', 2151),
          ('Knuth', 'Milk', 13, 4, 2006, 'NY', 2424),
          ('Knuth', 'Pepsi', 13, 12, 2007, 'PA', 2538),
          ('Knuth', 'Butter', 8, 10, 2008, 'PA', 2852),
          ('Emily', 'Soap', 4, 8, 2001, 'NJ', 3901),
          ('Emily', 'Coke', 20, 2, 2004, 'NJ', 1301),
          ('Bloom', 'Bread', 5, 3, 2007, 'NJ', 2277),
          ('Knuth', 'Fruits', 18, 9, 2003, 'CT', 159),
          ('Helen', 'Yogurt', 1, 10, 2007, 'CT', 3366),
          ('Emily', 'Bread', 13, 5, 2001, 'NJ', 232),
          ('Helen', 'Milk', 24, 12, 2006, 'NY', 311),
          ('Helen', 'Butter', 12, 11, 2001, 'NY', 1357),
          ('Bloom', 'Coke', 30, 9, 2000, 'NJ', 2714),
          ('Helen', 'Milk', 6, 3, 2003, 'CT', 1249),
          ('Helen', 'Butter', 15, 7, 2005, 'NJ', 942),
          ('Sam', 'Fruits', 21, 11, 2006, 'CT', 3399),
          ('Knuth', 'Yogurt', 2, 10, 2006, 'PA', 2497),
          ('Emily', 'Milk', 2, 7, 2008, 'CT', 26),
          ('Sam', 'Pepsi', 28, 9, 2008, 'NY', 4797),
          ('Helen', 'Bread', 3, 6, 2008, 'NY', 4855),
          ('Sam', 'Bread', 13, 6, 2007, 'NJ', 3187),
          ('Emily', 'Bread', 21, 6, 2000, 'NJ', 141),
          ('Knuth', 'Bread', 5, 8, 2000, 'CT', 2063),
          ('Emily', 'Pepsi', 4, 10, 2003, 'CT', 4135),
          ('Knuth', 'Bread', 27, 3, 2008, 'CT', 4372),
          ('Sam', 'Pepsi', 19, 6, 2001, 'NJ', 3368),
          ('Helen', 'Fruits', 21, 1, 2005, 'CT', 2360),
          ('Helen', 'Bread', 12, 12, 2004, 'CT', 4114),
          ('Bloom', 'Yogurt', 28, 4, 2005, 'NY', 4043),
          ('Bloom', 'Bread', 25, 8, 2004, 'PA', 3100),
          ('Bloom', 'Milk', 3, 5, 2001, 'NJ', 1794),
          ('Knuth', 'Pepsi', 27, 7, 2008, 'NJ', 4689),
          ('Emily', 'Eggs', 1, 12, 2003, 'NY', 1595),
          ('Bloom', 'Coke', 5, 4, 2003, 'PA', 2681),
          ('Helen', 'Butter', 5, 7, 2004, 'NJ', 1174),
          ('Knuth', 'Eggs', 18, 10, 2001, 'NJ', 2353),
          ('Knuth', 'Coke', 9, 10, 2002, 'NJ', 766),
          ('Bloom', 'Bread', 22, 6, 2008, 'NJ', 944),
          ('Emily', 'Bread', 31, 4, 2008, 'NJ', 2135),
          ('Helen', 'Coke', 23, 5, 2006, 'CT', 1099),
          ('Emily', 'Fruits', 7, 7, 2007, 'PA', 111),
          ('Sam', 'Butter', 20, 12, 2002, 'NY', 1234),
          ('Bloom', 'Cookies', 29, 11, 2006, 'CT', 796),
          ('Emily', 'Pepsi', 11, 5, 2005, 'NY', 4901),
          ('Sam', 'Soap', 29, 9, 2007, 'NY', 3260),
          ('Emily', 'Soap', 27, 1, 2006, 'NY', 3218),
          ('Knuth', 'Milk', 6, 10, 2003, 'PA', 1088),
          ('Bloom', 'Fruits', 31, 1, 2003, 'CT', 4085),
          ('Emily', 'Fruits', 22, 7, 2004, 'PA', 2243),
          ('Knuth', 'Coke', 30, 6, 2008, 'PA', 3166),
          ('Knuth', 'Cookies', 20, 2, 2000, 'NJ', 4237),
          ('Bloom', 'Fruits', 12, 2, 2001, 'NJ', 776),
          ('Sam', 'Cookies', 9, 5, 2004, 'PA', 454),
          ('Knuth', 'Bread', 19, 6, 2005, 'PA', 1932),
          ('Emily', 'Pepsi', 22, 3, 2004, 'CT', 4531),
          ('Sam', 'Butter', 30, 9, 2006, 'NY', 4893),
          ('Helen', 'Soap', 26, 8, 2005, 'CT', 1301),
          ('Helen', 'Milk', 21, 1, 2006, 'NJ', 1839),
          ('Knuth', 'Fruits', 24, 7, 2008, 'NY', 1062),
          ('Bloom', 'Coke', 22, 7, 2008, 'CT', 3640),
          ('Bloom', 'Bread', 25, 2, 2000, 'NJ', 4778),
          ('Sam', 'Bread', 22, 7, 2006, 'CT', 207),
          ('Bloom', 'Bread', 20, 4, 2006, 'CT', 417),
          ('Sam', 'Fruits', 23, 5, 2000, 'NJ', 3743),
          ('Helen', 'Butter', 28, 2, 2002, 'NJ', 2483),
          ('Helen', 'Cookies', 16, 11, 2006, 'CT', 1164),
          ('Sam', 'Soap', 9, 3, 2002, 'PA', 2994),
          ('Sam', 'Pepsi', 28, 5, 2002, 'NJ', 3468),
          ('Knuth', 'Bread', 13, 1, 2002, 'PA', 3049),
          ('Helen', 'Bread', 15, 11, 2008, 'PA', 1637),
          ('Helen', 'Coke', 4, 2, 2007, 'NJ', 755),
          ('Emily', 'Eggs', 20, 4, 2006, 'PA', 771),
          ('Knuth', 'Soap', 22, 8, 2005, 'NY', 1002),
          ('Knuth', 'Eggs', 12, 11, 2004, 'CT', 1950),
          ('Emily', 'Cookies', 30, 4, 2005, 'NY', 3669),
          ('Bloom', 'Eggs', 30, 11, 2004, 'CT', 1028),
          ('Helen', 'Coke', 31, 12, 2007, 'NJ', 3804),
          ('Sam', 'Cookies', 23, 6, 2003, 'PA', 362),
          ('Sam', 'Fruits', 21, 9, 2005, 'PA', 3800),
          ('Helen', 'Coke', 14, 4, 2001, 'CT', 2331),
          ('Bloom', 'Pepsi', 20, 9, 2000, 'PA', 2034),
          ('Knuth', 'Coke', 29, 1, 2005, 'PA', 4364),
          ('Sam', 'Fruits', 27, 7, 2007, 'PA', 2240),
          ('Sam', 'Fruits', 12, 10, 2005, 'NY', 2409),
          ('Knuth', 'Fruits', 1, 2, 2003, 'PA', 4475),
          ('Sam', 'Bread', 26, 9, 2001, 'NJ', 3809),
          ('Helen', 'Coke', 4, 12, 2003, 'CT', 2172),
          ('Emily', 'Cookies', 18, 12, 2007, 'PA', 4935),
          ('Knuth', 'Butter', 4, 3, 2008, 'CT', 2232),
          ('Bloom', 'Fruits', 14, 7, 2005, 'CT', 2813),
          ('Emily', 'Coke', 17, 1, 2006, 'NJ', 2192),
          ('Helen', 'Coke', 9, 10, 2008, 'NJ', 3065),
          ('Bloom', 'Pepsi', 20, 1, 2000, 'NJ', 526),
          ('Emily', 'Soap', 31, 7, 2007, 'NJ', 623),
          ('Bloom', 'Coke', 26, 7, 2003, 'NJ', 3058),
          ('Sam', 'Milk', 4, 2, 2002, 'PA', 1126),
          ('Sam', 'Coke', 14, 1, 2003, 'NY', 2668),
          ('Emily', 'Eggs', 20, 10, 2004, 'NJ', 4689),
          ('Bloom', 'Fruits', 29, 6, 2001, 'PA', 26),
          ('Helen', 'Pepsi', 25, 5, 2003, 'NY', 2590),
          ('Bloom', 'Eggs', 3, 1, 2004, 'NJ', 4972),
          ('Helen', 'Milk', 13, 4, 2000, 'NY', 888),
          ('Sam', 'Butter', 9, 6, 2008, 'NJ', 3659),
          ('Knuth', 'Yogurt', 3, 6, 2004, 'PA', 1285),
          ('Sam', 'Yogurt', 2, 7, 2003, 'PA', 2683),
          ('Knuth', 'Bread', 3, 2, 2004, 'CT', 1914),
          ('Helen', 'Coke', 9, 2, 2008, 'PA', 3132),
          ('Emily', 'Coke', 25, 8, 2002, 'CT', 4379),
          ('Helen', 'Butter', 7, 2, 2000, 'PA', 1025),
          ('Knuth', 'Coke', 3, 10, 2000, 'NJ', 4672),
          ('Sam', 'Cookies', 17, 11, 2002, 'PA', 3725),
          ('Helen', 'Yogurt', 12, 9, 2003, 'CT', 3015),
          ('Helen', 'Coke', 10, 7, 2000, 'NY', 2401),
          ('Sam', 'Eggs', 10, 5, 2008, 'NJ', 1114),
          ('Bloom', 'Coke', 2, 8, 2003, 'NJ', 279),
          ('Helen', 'Butter', 4, 6, 2001, 'NY', 4969),
          ('Knuth', 'Fruits', 30, 11, 2002, 'PA', 1718),
          ('Knuth', 'Eggs', 22, 9, 2003, 'PA', 3532),
          ('Sam', 'Milk', 19, 12, 2006, 'NJ', 4522),
          ('Bloom', 'Cookies', 20, 9, 2005, 'NJ', 4240),
          ('Bloom', 'Eggs', 2, 4, 2000, 'CT', 2865),
          ('Sam', 'Fruits', 12, 12, 2006, 'NJ', 1311),
          ('Bloom', 'Milk', 11, 5, 2007, 'NY', 1820),
          ('Knuth', 'Soap', 27, 10, 2006, 'CT', 2231),
          ('Sam', 'Milk', 6, 8, 2005, 'PA', 2124),
          ('Helen', 'Eggs', 21, 10, 2002, 'CT', 4162),
          ('Sam', 'Pepsi', 15, 3, 2008, 'CT', 1470),
          ('Emily', 'Butter', 7, 2, 2004, 'PA', 1491),
          ('Sam', 'Milk', 29, 1, 2005, 'CT', 879),
          ('Knuth', 'Bread', 7, 5, 2003, 'PA', 3664),
          ('Emily', 'Fruits', 24, 3, 2008, 'PA', 1469),
          ('Bloom', 'Butter', 9, 10, 2000, 'CT', 4279),
          ('Bloom', 'Bread', 13, 6, 2000, 'CT', 256),
          ('Knuth', 'Fruits', 25, 10, 2006, 'NY', 1181),
          ('Knuth', 'Cookies', 25, 10, 2004, 'NJ', 2154),
          ('Knuth', 'Cookies', 7, 3, 2002, 'PA', 488),
          ('Bloom', 'Eggs', 22, 4, 2007, 'PA', 3723),
          ('Helen', 'Butter', 1, 2, 2005, 'NJ', 2160),
          ('Helen', 'Cookies', 19, 7, 2004, 'NY', 1080),
          ('Knuth', 'Butter', 28, 3, 2004, 'PA', 516),
          ('Bloom', 'Bread', 3, 12, 2008, 'NJ', 1106),
          ('Knuth', 'Pepsi', 15, 6, 2001, 'NJ', 2064),
          ('Sam', 'Pepsi', 30, 10, 2001, 'CT', 3008),
          ('Helen', 'Eggs', 18, 10, 2004, 'CT', 1754),
          ('Helen', 'Yogurt', 18, 4, 2005, 'NY', 2516),
          ('Bloom', 'Eggs', 5, 7, 2000, 'NY', 1219),
          ('Emily', 'Eggs', 18, 8, 2007, 'NY', 4754),
          ('Knuth', 'Eggs', 2, 12, 2008, 'CT', 1312),
          ('Knuth', 'Cookies', 29, 6, 2006, 'CT', 2635),
          ('Emily', 'Cookies', 24, 12, 2006, 'CT', 3308),
          ('Knuth', 'Cookies', 3, 7, 2003, 'CT', 2423),
          ('Sam', 'Yogurt', 1, 5, 2005, 'NJ', 1286),
          ('Helen', 'Butter', 25, 11, 2000, 'CT', 306),
          ('Helen', 'Fruits', 19, 2, 2003, 'CT', 951),
          ('Helen', 'Bread', 30, 5, 2008, 'NY', 4901),
          ('Helen', 'Soap', 19, 8, 2005, 'NJ', 3618),
          ('Sam', 'Pepsi', 8, 11, 2008, 'NJ', 1097),
          ('Knuth', 'Bread', 14, 4, 2002, 'NY', 3370),
          ('Emily', 'Cookies', 17, 7, 2004, 'PA', 1477),
          ('Sam', 'Yogurt', 2, 3, 2001, 'NJ', 1943),
          ('Knuth', 'Yogurt', 29, 2, 2000, 'NY', 3445),
          ('Helen', 'Eggs', 5, 10, 2003, 'PA', 1045),
          ('Helen', 'Soap', 27, 11, 2001, 'PA', 925),
          ('Knuth', 'Coke', 8, 7, 2001, 'CT', 211),
          ('Helen', 'Coke', 15, 10, 2005, 'CT', 3840),
          ('Sam', 'Butter', 18, 4, 2004, 'NJ', 4453),
          ('Knuth', 'Bread', 16, 8, 2004, 'CT', 1095),
          ('Emily', 'Fruits', 29, 8, 2002, 'PA', 4731),
          ('Bloom', 'Butter', 17, 9, 2008, 'NJ', 3718),
          ('Bloom', 'Milk', 30, 3, 2001, 'CT', 2843),
          ('Bloom', 'Eggs', 24, 11, 2004, 'CT', 1573),
          ('Helen', 'Butter', 2, 7, 2003, 'NY', 4364),
          ('Helen', 'Cookies', 15, 7, 2005, 'NJ', 1965),
          ('Sam', 'Milk', 4, 2, 2008, 'CT', 3434),
          ('Emily', 'Butter', 13, 6, 2008, 'CT', 2869),
          ('Knuth', 'Butter', 9, 6, 2004, 'NY', 950),
          ('Bloom', 'Eggs', 13, 4, 2003, 'NY', 3621),
          ('Emily', 'Coke', 3, 6, 2004, 'NJ', 3314),
          ('Helen', 'Soap', 18, 11, 2007, 'NJ', 1991),
          ('Helen', 'Soap', 10, 1, 2003, 'NJ', 4702),
          ('Helen', 'Pepsi', 31, 1, 2008, 'NJ', 4412),
          ('Helen', 'Pepsi', 12, 12, 2002, 'CT', 1515),
          ('Bloom', 'Cookies', 19, 9, 2002, 'PA', 2504),
          ('Emily', 'Fruits', 6, 4, 2008, 'CT', 3055),
          ('Knuth', 'Fruits', 8, 4, 2005, 'CT', 973)]
```

In [13]:
```
#Insert data to the sales table
def bulkInsert(records):
    try:
        sql_insert_query = """ INSERT INTO sales (cust, prod, day, month
,year, state, quant)
                           VALUES (%s,%s,%s,%s,%s,%s,%s) """
        # executemany() to insert multiple rows rows
        result = cur.executemany(sql_insert_query, records)
        print(cur.rowcount, "Record inserted successfully into mobile ta
ble")
    except (Exception, psycopg2.Error) as error:
        print("Failed inserting record into mobile table {}".format(erro
r))

bulkInsert(records_to_insert)
```

```
500 Record inserted successfully into mobile table
```

In [14]:
```
# Validate the data was inserted into the table.
try:
    cur.execute("""SELECT COUNT(*) FROM sales;""")
except psycopg2.Error as e:
    print("Error: select *")
    print (e)

row = cur.fetchone()
while row:
   print(row)
   row = cur.fetchone()
```

```
(500,)
```

Take a look at the sales table

```
In [15]: try:
             cur.execute("""SELECT * FROM sales;""")
         except psycopg2.Error as e:
             print("Error: select *")
             print(e)

         row = cur.fetchone()
         sales = []
         while row:
             sales.append(row)
             row = cur.fetchone()

         sales_table = pd.DataFrame(sales, columns=['Customer', 'Product', 'Dat
         e,', 'Month', 'Year',
                                                     'State', 'Qutant'])
         print(sales_table.shape)
         print(sales_table)
```

```
(500, 7)
    Customer  Product  Date,  Month  Year State  Qutant
0      Bloom    Pepsi      2     12  2001    NY    4232
1      Knuth    Bread     23      5  2005    PA    4167
2      Emily    Pepsi     22      1  2006    CT    4404
3      Emily   Fruits     11      1  2000    NJ    4369
4      Helen     Milk      7     11  2006    CT     210
5      Emily     Soap      2      4  2002    CT    2549
6      Bloom     Eggs     30     11  2000    NJ     559
7      Bloom   Yogurt     25      7  2004    PA      17
8      Helen    Pepsi     14      3  2002    NJ    3891
9      Emily    Bread     28      9  2005    PA      42
10       Sam  Cookies     20     11  2004    NY    3376
11     Knuth     Milk      5      2  2007    PA     126
12     Helen     Coke     11      4  2001    NY     668
13     Emily   Butter      5      7  2005    NJ    3840
14     Emily   Yogurt      7     10  2005    NY     730
15       Sam     Soap     12      2  2001    NJ     165
16     Knuth     Coke      6      1  2003    CT    1557
17       Sam     Milk      9      8  2001    NY    1132
18     Helen   Yogurt      6      2  2000    PA    4001
19       Sam     Milk      6      1  2003    PA    2298
20     Knuth    Pepsi     21     12  2008    CT     653
21     Knuth     Eggs     19     12  2006    NJ    1339
22     Helen     Coke     19     10  2002    NJ    2662
23     Knuth     Milk     25      3  2001    CT      58
24     Bloom   Yogurt      3      4  2001    NJ    1203
25     Helen     Milk      6     11  2001    NY    2422
26     Knuth   Yogurt     22      4  2005    CT     301
27     Helen    Pepsi     24      9  2004    CT    2768
28     Helen   Fruits     28      6  2005    PA    2811
29     Knuth   Yogurt     18      9  2001    NY    3705
..       ...      ...    ...    ...   ...   ...     ...
470    Helen     Soap     19      8  2005    NJ    3618
471      Sam    Pepsi      8     11  2008    NJ    1097
472    Knuth    Bread     14      4  2002    NY    3370
473    Emily  Cookies     17      7  2004    PA    1477
474      Sam   Yogurt      2      3  2001    NJ    1943
475    Knuth   Yogurt     29      2  2000    NY    3445
476    Helen     Eggs      5     10  2003    PA    1045
477    Helen     Soap     27     11  2001    PA     925
478    Knuth     Coke      8      7  2001    CT     211
479    Helen     Coke     15     10  2005    CT    3840
480      Sam   Butter     18      4  2004    NJ    4453
481    Knuth    Bread     16      8  2004    CT    1095
482    Emily   Fruits     29      8  2002    PA    4731
483    Bloom   Butter     17      9  2008    NJ    3718
484    Bloom     Milk     30      3  2001    CT    2843
485    Bloom     Eggs     24     11  2004    CT    1573
486    Helen   Butter      2      7  2003    NY    4364
487    Helen  Cookies     15      7  2005    NJ    1965
488      Sam     Milk      4      2  2008    CT    3434
489    Emily   Butter     13      6  2008    CT    2869
490    Knuth   Butter      9      6  2004    NY     950
491    Bloom     Eggs     13      4  2003    NY    3621
492    Emily     Coke      3      6  2004    NJ    3314
493    Helen     Soap     18     11  2007    NJ    1991
```

```
494    Helen     Soap     10      1  2003    NJ    4702
495    Helen    Pepsi     31      1  2008    NJ    4412
496    Helen    Pepsi     12     12  2002    CT    1515
497    Bloom  Cookies     19      9  2002    PA    2504
498    Emily   Fruits      6      4  2008    CT    3055
499    Knuth   Fruits      8      4  2005    CT     973

[500 rows x 7 columns]
```

For each customer, compute the minimum and maximum sales quantities along with the corresponding products (purchased), dates (i.e., dates of those minimum and maximum sales quantities) and the states in which the sale transactions took place. If there are >1 occurrences of the min or max, display all. For the same customer, compute the average sales quantity.

In [16]:
```
try:
    cur.execute(
        """
                WITH base AS
                    (
                        SELECT cust, min(quant) AS min_q, max(quant) AS
 max_q,
                               avg(quant) AS avg_q FROM sales GROUP BY c
ust
                    )
                        SELECT minq.cust AS customer, minq.min_q, minq.p
rod AS min_prod,
                               minq.month||'/'||minq.day||'/'||minq.year
AS min_date,
                               minq.state AS st, maxq.max_q, maxq.prod A
S max_prod,
                               maxq.month||'/'||maxq.day||'/'||maxq.year
AS max_date,
                               maxq.state AS st, round(minq.avg_q,0) AS
 avg_q from
                        (
                            SELECT b.cust, s.prod, s.month, s.day, s.yea
r, s.state, b.min_q, b.avg_q
                            FROM base b, sales s
                            WHERE b.cust = s.cust AND b.min_q = s.quant
                        ) AS minq
                        INNER JOIN
                        (
                            SELECT b.cust, s.prod, s.month, s.day, s.yea
r, s.state, b.max_q
                            FROM base b, sales s
                            WHERE b.cust = s.cust AND b.max_q = s.quant
                        ) AS maxq
                        ON minq.cust = maxq.cust;"""
    )

except psycopg2.Error as e:
    print("Error: select")
    print(e)

row = cur.fetchone()
list_1 = []
while row:
    list_1.append(row)
    row = cur.fetchone()

df_1 = pd.DataFrame(list_1, columns=['Cust', 'Min_q', 'Min_prod,', 'Min_
date', 'State',
                                     'Max_q', 'Max_prod', 'Max_date', 'S
tate', 'Avg_q'])
print(df_1.shape)
print(df_1)
```

```
(5, 10)
    Cust  Min_q Min_prod,    Min_date State  Max_q Max_prod    Max_date
State Avg_q
0  Knuth     24    Fruits   8/26/2001    NY   4994  Cookies   7/17/2003
CT  2303
1  Helen     77      Milk    4/9/2000    NY   4969   Butter    6/4/2001
NY  2301
2    Sam     93     Bread  11/13/2008    PA   4954     Soap    4/6/2003
PA  2385
3  Bloom     17    Yogurt   7/25/2004    PA   4972     Eggs    1/3/2004
NJ  2457
4  Emily     26      Milk    7/2/2008    CT   4986     Milk  11/12/2002
NY  2512
```

For each of the 12 months (regardless of the year), find the most “popular” and least “popular” products (those products with most and least total sales quantities) and the corresponding total sales quantities (i.e., SUMs).

In [17]:
```
try:
    cur.execute(
        """
                WITH base AS
                    (SELECT prod, month, SUM(quant) AS sum_q  FROM sales
GROUP BY prod, month)
                    SELECT agg.month, base_most.prod AS most_popular_pro
d,
                    base_most.sum_q AS most_pop_total_q,
                    base_least.prod AS least_popular_prod,
                    base_least.sum_q AS least_pop_total_q
                    FROM
                    (
                        (SELECT b1.month, MAX(b1.sum_q) AS max_q, MIN(b
1.sum_q) AS min_q
                        FROM base AS b1
                        INNER JOIN base AS b2
                        ON b1.month = b2.month AND b1.sum_q >= b2.sum_q
                        GROUP BY b1.month) AS agg
                        LEFT JOIN base AS base_most
                        ON agg.max_q = base_most.sum_q AND agg.month = b
ase_most.month
                        LEFT JOIN base AS base_least
                        ON agg.min_q = base_least.sum_q AND agg.month =
 base_least.month
                    )
                    ORDER BY agg.month;"""
    )

except psycopg2.Error as e:
    print("Error: select")
    print(e)

row = cur.fetchone()
list_2 = []
while row:
    list_2.append(row)
    row = cur.fetchone()

df_2 = pd.DataFrame(list_2, columns=['Month', 'Most_popular_prod', 'Most
_pop_total_q,',
                                     'Least_popular_prod', 'Least_pop_to
tal_q'])
print(df_2.shape)
print(df_2)
```

```
(12, 5)
    Month Most_popular_prod  Most_pop_total_q, Least_popular_prod  Leas
t_pop_total_q
0       1            Fruits              18811             Butter
5009
1       2              Coke              23908               Soap
822
2       3             Pepsi              18295            Cookies
488
3       4              Eggs              19922            Cookies
3669
4       5             Pepsi              15008               Soap
2961
5       6           Cookies              19783               Milk
1274
6       7            Butter              20273              Bread
2573
7       8            Fruits              16560              Pepsi
1775
8       9             Pepsi              22178               Coke
2714
9      10              Coke              17872              Bread
4309
10     11            Fruits              20309               Coke
1998
11     12            Fruits              18041             Yogurt
1911
```

For each product, find the “most favorable” month (when most amount of the product was sold) and the “least favorable” month (when the least amount of the product was sold).

```
In [18]: try:
             cur.execute(
                 """
                         WITH base AS
                             (SELECT prod, month, SUM(quant) AS sum_q FROM sales
          GROUP BY prod, month)
                             SELECT max_month.product, max_month.month AS most_fa
         v_mo,
                             min_month.month AS least_fav_mo
                             FROM
                             (
                                 (
                                     (SELECT b1.prod AS product, MAX(b1.sum_q) AS
         max_q  FROM base AS b1
                                     INNER JOIN base AS b2
                                     ON b1.prod = b2.prod AND b1.sum_q >= b2.sum_
         q
                                     GROUP BY b1.prod) AS agg_max
                                     INNER JOIN base
                                     ON agg_max.product = base.prod AND agg_max.m
         ax_q = base.sum_q
                                 ) AS max_month
                                 INNER JOIN
                                 (
                                     (SELECT b3.prod AS product, MIN(b3.sum_q) AS
         min_q  FROM base AS b3
                                     INNER JOIN base AS b4
                                     ON b3.prod = b4.prod AND b3.sum_q <= b4.sum_
         q
                                     GROUP BY b3.prod) AS agg_min
                                     INNER JOIN base
                                     ON agg_min.product = base.prod AND agg_min.m
         in_q = base.sum_q
                                 ) AS min_month
                                 ON max_month.product = min_month.product
                             )
                             ORDER BY max_month.product;"""
             )

         except psycopg2.Error as e:
             print("Error: select")
             print(e)

         row = cur.fetchone()
         list_3 = []
         while row:
             list_3.append(row)
             row = cur.fetchone()

         df_3 = pd.DataFrame(
             list_3, columns=['Product', 'Most_favorable_month', 'Least_favorable
         _month'])
         print(df_3.shape)
         print(df_3)
```

```
(10, 3)
   Product  Most_favorable_month  Least_favorable_month
0    Bread                     5                      7
1   Butter                     7                      8
2     Coke                     2                     11
3  Cookies                     7                      3
4     Eggs                     4                      3
5   Fruits                    11                      6
6     Milk                     1                      6
7    Pepsi                     9                      8
8     Soap                    11                      2
9   Yogurt                     6                     12
```

Show for each customer and product combination, the average sales quantities for 4 quarters, Q1, Q2, Q3 and Q4 (in four separate columns) – Q1 being the first 3 months of the year (Jan, Feb & Mar), Q2 the next 3 months (Apr, May & Jun), and so on – ignore the YEAR component of the dates (i.e., 3/11/2001 is considered the same date as 3/11/2002, etc.). Also compute the average for the “whole” year (again ignoring the YEAR component, meaning simply compute AVG) along with the total quantities (SUM) and the counts (COUNT).

In [19]:
```
try:
    cur.execute(
        """
                SELECT avg_s.cust,avg_s.prod, q1, q2, q3, q4, average, t
otal, count FROM
                    (SELECT cust, prod, ROUND(AVG(quant)) AS q1 FROM sal
es
                    WHERE sales.month IN (1, 2, 3) GROUP BY cust, prod)
 AS sq_1
                    RIGHT JOIN
                    (SELECT cust, prod, ROUND(AVG(quant)) AS q2 FROM sal
es
                    WHERE sales.month IN (4, 5, 6) GROUP BY cust, prod)
 AS sq_2
                    ON sq_1.cust = sq_2.cust AND sq_1.prod = sq_2.prod
                    RIGHT JOIN
                    (SELECT cust, prod, ROUND(AVG(quant)) AS q3 FROM sal
es
                    WHERE sales.month IN (7, 8, 9) GROUP BY cust, prod)
 AS sq_3
                    ON sq_2.cust = sq_3.cust AND sq_2.prod = sq_3.prod
                    RIGHT JOIN
                    (SELECT cust, prod, ROUND(AVG(quant)) AS q4 FROM sal
es
                    WHERE sales.month IN (10, 11, 12) GROUP BY cust, pro
d) AS sq_4
                    ON sq_3.cust = sq_4.cust AND sq_3.prod = sq_4.prod
                    RIGHT JOIN
                    (SELECT cust, prod, ROUND(AVG(quant)) AS average,
                    SUM(quant) AS total, COUNT(*) AS count
                    FROM sales GROUP BY cust, prod) AS avg_s
                    ON sq_4.cust = avg_s.cust AND sq_4.prod = avg_s.prod
                    ORDER BY cust;"""
    )

except psycopg2.Error as e:
    print("Error: select")
    print(e)

row = cur.fetchone()
list_4 = []
while row:
    list_4.append(row)
    row = cur.fetchone()

df_4 = pd.DataFrame(list_4, columns=['Customer', 'Product', 'Q1_avg,',
'Q2_avg',
                                     'Q3_avg', 'Q4_avg', 'Average', 'Tot
al', 'Count'])
print(df_4.shape)
print(df_4)
```

```
(50, 9)
   Customer  Product Q1_avg, Q2_avg Q3_avg Q4_avg Average  Total  Count
0     Bloom    Bread    3267    539   2388   1106    2112  25343     12
1     Bloom   Butter    None   None   2386   4279    3017   9051      3
2     Bloom     Coke    1079   2681   2669   2867    2339  21050      9
3     Bloom  Cookies    None   2782   3287   2970    3069  27622      9
4     Bloom     Eggs    2819   3597   2185   1053    2505  27553     11
5     Bloom   Fruits    2243   1933   2994   2926    2471  42011     17
6     Bloom     Milk    3215   2377   1106   3010    2765  35941     13
7     Bloom    Pepsi     526   1868    794   4374    1924  15390      8
8     Bloom     Soap    4623   2310    329   4364    2377  14264      6
9     Bloom   Yogurt    1660   2623   2398   1786    2209  17670      8
10    Emily    Bread     492   1720    807   2318    1534  15343     10
11    Emily   Butter    1490   2869   3053   2862    2317  20852      9
12    Emily     Coke    2398   3314   4379    999    2409  21681      9
13    Emily  Cookies    3131   3760   1716   2421    2673  42765     16
14    Emily     Eggs    4535   2715   3231   2039    2757  33080     12
15    Emily   Fruits    2919   2243   2750   2873    2697  35059     13
16    Emily     Milk    2589   2581   1997   4986    2663  21307      8
17    Emily    Pepsi    None   None   None   4135    3255  22782      7
18    Emily     Soap    3218   2549   1531   2976    2411  19290      8
19    Emily   Yogurt    3079   2650   2161    730    2359  16510      7
20    Helen    Bread    None   None   None   2755    2781  25032      9
21    Helen   Butter    2040   2886   1913    907    1909  41993     22
22    Helen     Coke    2708   1999   2800   3109    2618  39263     15
23    Helen  Cookies    None   None   2180   1179    1779   8897      5
24    Helen     Eggs    None   None   None   2320    2135  17081      8
25    Helen   Fruits    None   None   None   2142    1840  16561      9
26    Helen     Milk    3312    807   3314    981    2103  25239     12
27    Helen    Pepsi    3130   2397   2768   2435    2663  34613     13
28    Helen     Soap    None   None   2990   1209    2499  22493      9
29    Helen   Yogurt    2332   3174   2904   2247    2655  26550     10
30    Knuth    Bread    3366   3283   1579   1405    2714  32562     12
31    Knuth   Butter    None   None   None   1504    1258   7545      6
32    Knuth     Coke    3559   2676   1512   2789    2727  35449     13
33    Knuth  Cookies    2363   2978   3709   1669    2608  26081     10
34    Knuth     Eggs    4051   1239   3532   2254    2538  25381     10
35    Knuth   Fruits    2758   1857   1175   2279    1848  25871     14
36    Knuth     Milk    None   None   None   1088     924   3696      4
37    Knuth    Pepsi    3214   2468   2982   1596    2646  34401     13
38    Knuth     Soap     878   1519   1002   2620    1856  20416     11
39    Knuth   Yogurt    3903   1787   3705   2497    2645  21157      8
40      Sam    Bread    1378   2735   1622     93    1678  13427      8
41      Sam   Butter     673   4056   4893   1984    2702  21612      8
42      Sam     Coke    None   None   None   None    2271  11354      5
43      Sam  Cookies    None   None   None   2223    1747  12229      7
44      Sam     Eggs    2482   2239   1191   4674    2372  18977      8
45      Sam   Fruits    None   3743   3611   3217    3388  33879     10
46      Sam     Milk    1745    598   2125   2431    1815  23589     13
47      Sam    Pepsi    2442   2332   3954   3166    2988  38848     13
48      Sam     Soap    1994   2878   3108    694    2201  24215     11
49      Sam   Yogurt    None   None   None   None    2369  21319      9
```

For each customer, product and state combination, compute (1) the product’s average sale of this customer for the state (i.e., the simple AVG for the group-by attributes. This is the easiest part), (2) the average sale of the product and the state but for all of the other customers and (3) the customer’s average sale for the given state, but for all of the other products.

In [20]:
```
try:
    cur.execute(
        """
                SELECT cps.cust, cps.prod, cps.state,
                   ROUND(AVG(avg_q)) AS prod_avg,
                   ROUND(AVG(other_cust.quant)) AS other_cust_q,
                   ROUND(AVG(other_prod.quant)) AS other_prod_avg FROM
                   (
                      (SELECT cust, prod, state, AVG(quant) AS avg_q
                       FROM sales GROUP BY cust, prod, state) AS cps
                       RIGHT JOIN
                       (SELECT cust AS customer, prod, state, quant FROM
sales) AS other_cust
                       ON cps.prod = other_cust.prod AND cps.state = oth
er_cust.state
                       AND cps.cust != other_cust.customer
                       RIGHT JOIN
                       (SELECT cust AS customer, prod, state, quant FROM
sales) AS other_prod
                       ON cps.cust = other_prod.customer and cps.state =
other_prod.state
                       AND cps.prod != other_prod.prod
                    )
                  GROUP BY cps.cust, cps.prod, cps.state
                  ORDER BY cps.cust, cps.prod, cps.state;"""
    )

except psycopg2.Error as e:
    print("Error: select")
    print(e)

row = cur.fetchone()
list_5 = []
while row:
    list_5.append(row)
    row = cur.fetchone()

df_5 = pd.DataFrame(list_5, columns=['customer', 'product', 'state,',
                                     'avg_product', 'other_cust_avg', 'o
ther_prod_avg'])
print(df_5.shape)
print(df_5)
```

```
(187, 6)
    customer  product state, avg_product other_cust_avg other_prod_avg
0      Bloom    Bread    CT         1530           2321           2782
1      Bloom    Bread    NJ         2161           2056           2567
2      Bloom    Bread    NY         3792           2623           2546
3      Bloom    Bread    PA         3100           1959           2146
4      Bloom   Butter    CT         4279           1652           2472
5      Bloom   Butter    NJ         3718           2225           2453
6      Bloom   Butter    NY         1054           2262           2707
7      Bloom     Coke    CT         2284           2491           2563
8      Bloom     Coke    NJ         2017           2577           2552
9      Bloom     Coke    NY         1229           1916           2697
10     Bloom     Coke    PA         3067           3459           2056
11     Bloom  Cookies    CT         2641           2695           2523
12     Bloom  Cookies    NJ         4064           2353           2381
13     Bloom  Cookies    NY         4570           2411           2501
14     Bloom  Cookies    PA         2181           2020           2189
15     Bloom     Eggs    CT         2411           2406           2565
16     Bloom     Eggs    NJ         2766           2660           2477
17     Bloom     Eggs    NY         2164           2910           2745
18     Bloom     Eggs    PA         3723           2117           2118
19     Bloom   Fruits    CT         3639           2059           2398
20     Bloom   Fruits    NJ         2234           2713           2527
21     Bloom   Fruits    NY         2333           2253           2696
22     Bloom   Fruits    PA         2151           2677           2204
23     Bloom     Milk    CT         3117           1578           2466
24     Bloom     Milk    NJ         2380           3400           2516
25     Bloom     Milk    NY         3306           1978           2477
26     Bloom     Milk    PA         2383           2085           2159
27     Bloom    Pepsi    CT           94           2686           2639
28     Bloom    Pepsi    NJ         2194           2939           2546
29     Bloom    Pepsi    NY         2243           3371           2662
..       ...      ...   ...          ...            ...            ...
157      Sam   Butter    PA          673           1820           2510
158      Sam     Coke    CT         2746           2433           1935
159      Sam     Coke    NY         1405           1964           2536
160      Sam     Coke    PA         3053           3373           2410
161      Sam  Cookies    CT         1328           2905           2093
162      Sam  Cookies    NY         3376           2560           2381
163      Sam  Cookies    PA         1550           2296           2605
164      Sam     Eggs    NJ         1103           3075           2709
165      Sam     Eggs    NY         2175           2832           2471
166      Sam     Eggs    PA         3416           1887           2302
167      Sam   Fruits    CT         3101           2263           1896
168      Sam   Fruits    NJ         3282           2363           2495
169      Sam   Fruits    NY         3574           2074           2308
170      Sam   Fruits    PA         3561           2325           2283
171      Sam     Milk    CT         1423           2117           2214
172      Sam     Milk    NJ         4522           2533           2508
173      Sam     Milk    NY         1601           2528           2623
174      Sam     Milk    PA         1849           2231           2516
175      Sam    Pepsi    CT         2376           2563           1926
176      Sam    Pepsi    NJ         3086           2663           2435
177      Sam    Pepsi    NY         3993           3021           2264
178      Sam    Pepsi    PA         2841           1918           2419
179      Sam     Soap    CT         2461           1923           1993
180      Sam     Soap    NJ         1348           2758           2811
```

```
181      Sam     Soap     NY        2529          2898          2418
182      Sam     Soap     PA        2826          1776          2362
183      Sam   Yogurt     CT        2546          2621          1957
184      Sam   Yogurt     NJ        1841          2203          2683
185      Sam   Yogurt     NY        1317          2651          2484
186      Sam   Yogurt     PA        3130          2334          2342

[187 rows x 6 columns]
```

For customer and product, show the average sales before and after each month (e.g., for February, show average sales of January and March. For “before” January and “after” December, display . The “YEAR” attribute is not considered for this query – for example, both January of 2007 and January of 2008 are considered January regardless of the year.

In [21]:
```
try:
    cur.execute("""
                SELECT * FROM
                (
                    SELECT base.cust, base.prod, base.month,
                    ROUND(AVG(before_month.quant)) AS before_avg,
                    ROUND(AVG(after_month.quant)) AS after_avg
                    FROM
                    (
                        (SELECT cust, prod, month FROM sales GROUP BY cu
st, prod, month) AS base
                        LEFT JOIN sales as before_month
                        ON base.cust = before_month.cust AND base.prod =
before_month.prod
                        AND base.month = before_month.month+1
                        LEFT JOIN sales as after_month
                        ON base.cust = after_month.cust AND base.prod =
 after_month.prod
                        AND base.month = after_month.month-1
                    )
                    GROUP BY base.cust, base.prod, base.month
                    UNION
                    SELECT base_1.cust, base_1.prod, base_1.before_mo,
                    ROUND(AVG(before_month.quant)) AS before_avg,
                    ROUND(AVG(after_month.quant)) AS after_avg FROM
                    (
                        (SELECT cust, prod, month-1 AS before_mo FROM sa
les
                        GROUP BY cust, prod, month) AS base_1
                        LEFT JOIN sales as before_month
                        ON base_1.cust = before_month.cust AND base_1.pr
od = before_month.prod
                        AND base_1.before_mo = before_month.month+1
                        LEFT JOIN sales as after_month
                        ON base_1.cust = after_month.cust AND base_1.pro
d = after_month.prod
                        AND base_1.before_mo = after_month.month-1
                    )
                    WHERE base_1.before_mo != 0
                    GROUP BY base_1.cust, base_1.prod, base_1.before_mo
                    UNION
                    SELECT base_2.cust, base_2.prod, base_2.after_mo,
                    ROUND(AVG(before_month.quant)) AS before_avg,
                    ROUND(AVG(after_month.quant)) AS after_avg FROM
                    (
                        (SELECT cust, prod, month+1 AS after_mo FROM sal
es
                        GROUP BY cust, prod, month ) AS base_2
                        LEFT JOIN sales as before_month
                        ON base_2.cust = before_month.cust AND base_2.pr
od = before_month.prod
                        AND base_2.after_mo = before_month.month+1
                        LEFT JOIN sales as after_month
                        ON base_2.cust = after_month.cust AND base_2.pro
d = after_month.prod
                        AND base_2.after_mo = after_month.month-1
                    )
                    WHERE base_2.after_mo != 13
                    GROUP BY base_2.cust, base_2.prod, base_2.after_mo
                ) AS final
                WHERE final.before_avg IS NOT NULL OR final.after_avg IS
NOT NUll
                ORDER BY final.cust, final.prod, final.month;"""
                )

except psycopg2.Error as e:
    print("Error: select")
    print(e)

row = cur.fetchone()
list_6 = []

while row:
    list_6.append(row)
    row = cur.fetchone()

df_6 = pd.DataFrame(
    list_6, columns=['Customer', 'Product', 'Month', 'Before_avg', 'Afte
r_avg'])
print(df_6.shape)
print(df_6)
```

```
(456, 5)
    Customer  Product  Month Before_avg After_avg
0      Bloom    Bread      1       None      4778
1      Bloom    Bread      2       2220      3035
2      Bloom    Bread      3       4778       417
3      Bloom    Bread      4       3035      None
4      Bloom    Bread      5        417       600
5      Bloom    Bread      6       None      1494
6      Bloom    Bread      7        600      3100
7      Bloom    Bread      8       1494      2480
8      Bloom    Bread      9       3100      None
9      Bloom    Bread     10       2480      None
10     Bloom    Bread     11       None      1106
11     Bloom   Butter      7       None      1054
12     Bloom   Butter      8       None      3718
13     Bloom   Butter      9       1054      4279
14     Bloom   Butter     10       3718      None
15     Bloom   Butter     11       4279      None
16     Bloom     Coke      1       None      1229
17     Bloom     Coke      2        928      None
18     Bloom     Coke      3       1229      2681
19     Bloom     Coke      5       2681      None
20     Bloom     Coke      6       None      3451
21     Bloom     Coke      7       None       279
22     Bloom     Coke      8       3451      2714
23     Bloom     Coke      9        279      2867
24     Bloom     Coke     10       2714      None
25     Bloom     Coke     11       2867      None
26     Bloom  Cookies      5       None      2782
27     Bloom  Cookies      6       None      4546
28     Bloom  Cookies      7       2782      None
29     Bloom  Cookies      8       4546      2867
..       ...      ...    ...        ...       ...
426      Sam     Milk     11       None      2843
427      Sam     Milk     12       1607      None
428      Sam    Pepsi      2       None      2442
429      Sam    Pepsi      3       None       160
430      Sam    Pepsi      4       2442      3468
431      Sam    Pepsi      5        160      3368
432      Sam    Pepsi      6       3468      3189
433      Sam    Pepsi      7       3368      None
434      Sam    Pepsi      8       3189      4337
435      Sam    Pepsi      9       None      3008
436      Sam    Pepsi     10       4337      2396
437      Sam    Pepsi     11       3008      4865
438      Sam    Pepsi     12       2396      None
439      Sam     Soap      1       None       165
440      Sam     Soap      2       2409      2994
441      Sam     Soap      3        165      4954
442      Sam     Soap      4       2994      None
443      Sam     Soap      5       4954      1841
444      Sam     Soap      7       1841      2956
445      Sam     Soap      8       None      3260
446      Sam     Soap      9       2956      None
447      Sam     Soap     10       3260       694
448      Sam     Soap     12        694      None
449      Sam   Yogurt      2       2218      1943
```

```
450      Sam   Yogurt      4        1943      2925
451      Sam   Yogurt      5        None      2546
452      Sam   Yogurt      6        2925      2683
453      Sam   Yogurt      7        2546      None
454      Sam   Yogurt      8        2683      1317
455      Sam   Yogurt     10        1317      None

[456 rows x 5 columns]
```

For customer and product, find the month by which time, a half of the sales quantities have been purchased. Again for this query, the “YEAR” attribute is not considered. Another way to view this problem (as in problem #2 above) is to pretend all 500 rows of sales data are from the same year.

In [22]:
```
try:
    cur.execute("""
                SELECT cust, prod, MIN(month) as half_purchased_month FR
OM
                (
                    SELECT * FROM
                    (
                        SELECT base.cust, base.prod, base.month, SUM(pr
e.pre_q) as pre_total_q,
                        AVG(total.total_q) as totalq FROM
                        (
                            (SELECT cust, prod, month, SUM(quant) as mon
th_q
                            FROM sales GROUP BY cust, prod, month) AS ba
se
                            LEFT JOIN
                            (SELECT cust, prod, month, SUM(quant) as pre
_q
                            FROM sales GROUP BY cust, prod, month) AS pr
e
                            ON base.cust = pre.cust AND base.prod = pre.
prod
                            AND base.month >= pre.month
                            LEFT JOIN
                            (SELECT cust, prod, SUM(quant) as total_q
                            FROM sales GROUP BY cust, prod) AS total
                            on base.cust = total.cust AND base.prod = to
tal.prod
                        )
                        GROUP BY base.cust, base.prod, base.month
                    ) AS comb
                    WHERE comb.pre_total_q >= (comb.totalq/2)
                ) AS comparison
                GROUP BY comparison.cust, comparison.prod
                ORDER BY comparison.cust, comparison.prod;"""
                )

except psycopg2.Error as e:
    print("Error: select")
    print(e)

row = cur.fetchone()
list_7 = []

while row:
    list_7.append(row)
    row = cur.fetchone()

df_7 = pd.DataFrame(list_7, columns=['customer', 'product', 'month'])
print(df_7.shape)
print(df_7)
```

```
(50, 3)
   customer  product  month
0     Bloom    Bread      3
1     Bloom   Butter      9
2     Bloom     Coke      7
3     Bloom  Cookies      9
4     Bloom     Eggs      4
5     Bloom   Fruits      5
6     Bloom     Milk     10
7     Bloom    Pepsi     11
8     Bloom     Soap      6
9     Bloom   Yogurt      9
10    Emily    Bread      6
11    Emily   Butter      7
12    Emily     Coke      2
13    Emily  Cookies      7
14    Emily     Eggs      8
15    Emily   Fruits      8
16    Emily     Milk      7
17    Emily    Pepsi      3
18    Emily     Soap      8
19    Emily   Yogurt      5
20    Helen    Bread      6
21    Helen   Butter      6
22    Helen     Coke      7
23    Helen  Cookies      8
24    Helen     Eggs      6
25    Helen   Fruits      5
26    Helen     Milk      2
27    Helen    Pepsi      4
28    Helen     Soap      8
29    Helen   Yogurt      6
30    Knuth    Bread      4
31    Knuth   Butter      6
32    Knuth     Coke      5
33    Knuth  Cookies      6
34    Knuth     Eggs      9
35    Knuth   Fruits      7
36    Knuth     Milk      4
37    Knuth    Pepsi      7
38    Knuth     Soap     11
39    Knuth   Yogurt      5
40      Sam    Bread      6
41      Sam   Butter      9
42      Sam     Coke      5
43      Sam  Cookies     11
44      Sam     Eggs      5
45      Sam   Fruits     10
46      Sam     Milk      7
47      Sam    Pepsi      9
48      Sam     Soap      4
49      Sam   Yogurt      5
```

And finally close your cursor and connection.

In [23]:
```
cur.close()
conn.close()
```