DATA 311 - Final Project

Due midnight, Dec 21

Make sure to submit as a PDF file with all code legible

As a reminder, the sales rep data is based on geographical rules, with smaller size areas overriding rules for larger areas. In order of highest priority to lowest priority, these areas are:

- Zip
- State
- Division
- Region

For example, if an assignment exists for an account's zip code, that takes precedence over the rest of the rules.

Every region is assigned to a sales rep, so if no other more specific rules exist, the sales rep would be determined by the region the account is located in.

Hint: Chances are, you'll be making a lot of use of IFNULL statements!

In [1]:

```
import sqlite3, pandas as pd
conn = sqlite3.connect('sales.db')
curs = conn.cursor()
```

In [ ]:

```
# New tCustRep table (not in ERD)

def GetRepID(cust_id):
    '''Check to see what sales rep this customer has, add to tCust table'''
    cust_id = str(cust_id)
    sql = """SELECT cust_id, zip, st, div, reg FROM tCust LEFT JOIN tZip USING(zip)
        LEFT JOIN tStatetoDiv USING(st) LEFT JOIN tDivision USING(div) WHERE cust_id = " + cust_id + ";"""
    cust_loc = pd.read_sql(sql, conn)
    rep_zips = pd.read_sql("SELECT * FROM tZipRep;", conn)
    rep_st = pd.read_sql("SELECT * FROM tStateRep;", conn)
    rep_div = pd.read_sql("SELECT * FROM tDivisionRep;", conn)
    rep_reg = pd.read_sql("SELECT * FROM tRegion;", conn)
    idx = 0
    cust_loc['rep_id'] = 0
    for row in cust_loc.values:
        cust_id = str(row[0])
        cust_zip = str(row[1])
        cust_st = str(row[2])
        cust_div = str(row[3])
        cust_reg = str(row[4])
        if cust_zip in list(rep_zips['zip'].unique()):
            cust_rep = rep_zips[rep_zips['zip'] == cust_zip].values[0][1]
        elif cust_st in list(rep_st['st'].unique()):
            cust_rep = rep_st[rep_st['st'] == cust_st].values[0][1]
        elif cust_div in list(rep_div['div'].unique()):
            cust_rep = rep_div[rep_div['div'] == cust_div].values[0][1]
        else:
            cust_rep = rep_reg[rep_reg['reg'] == cust_reg].values[0][1]
    return cust_rep


curs.execute("DROP TABLE IF EXISTS tCustRep;")
curs.execute("""CREATE TABLE tCustRep(
                cust_id INTEGER NOT NULL REFERENCES tCust(cust_id),
                rep_id INTEGER NOT NULL REFERENCES tRep(rep_id),
                primary key(cust_id, rep_id));""")

tCustRep = pd.read_sql("SELECT cust_id FROM tCust;", conn)
tCustRep['rep_id'] = '0'
idx = 0
for row in tCustRep.values:
    cust_id = str(row[0])
    rep_id = GetRepID(cust_id)
    tCustRep.at[idx, 'rep_id'] = rep_id
    idx+=1
FillTable('tCustRep', tCustRep, curs)
```

1) What are our total sales for all data in the database, grouped by sales rep?

Return two columns:

- Sales rep name
- Total sales

In [13]:

```
pd.read_sql("""SELECT rep_name, TotalSales
                FROM
                (SELECT rep_id, SUM(Sales) AS TotalSales
                FROM vSalesOrderDetail
                JOIN tCustRep USING(cust_id)
                GROUP BY rep_id)
                JOIN tRep USING(rep_id);""", conn)
```

Out[13]:

	rep_name	TotalSales
0	Andy	306958.08
1	Beth	84558.14
2	Chen	1287986.98
3	Diana	605854.03
4	Edgar	29146.97
5	Frank	209197.32

---

2) How many customers are assigned to each sales rep?

Return two columns:

- Sales rep name
- Number of customers assigned

In [6]:

```
pd.read_sql("""SELECT rep_name, NumCust
                FROM
                (SELECT rep_id, count(cust_id) AS NumCust
                FROM vCustomerAddress
                JOIN tCustRep using(cust_id)
                GROUP BY rep_id)
                JOIN tRep USING(rep_id);""", conn)
```

Out[6]:

	rep_name	NumCust
0	Andy	41

	rep_name	NumCust
1	Beth	13
2	Chen	149
3	Diana	73
4	Edgar	5
5	Frank	29

---

3) What are our total sales for all data in the database, grouped by region?

Return two columns:

- Region
- Total sales

In [14]:

```
pd.read_sql("""SELECT reg, SUM(Sales) as TotalSales
                FROM vSalesOrderDetail
                JOIN tCust USING(cust_id)
                JOIN tZip USING(zip)
                JOIN tStateToDiv USING(st)
                JOIN tDivision USING(div)
                JOIN tRegion USING(reg)
                GROUP BY reg;""", conn)
```

Out[14]:

	reg	TotalSales
0	Midwest	516155.40
1	Northeast	438160.59
2	PR	29146.97
3	South	934384.53
4	West	605854.03

---

4) What are our total sales for all data in the database, grouped by division?

Return two columns:

- Division
- Total sales

In [3]:

```
pd.read_sql("""SELECT div, IFNULL(sum(TotalSales),0) as TotalSales
                FROM vCustomerAddress
                LEFT JOIN vTotalSalesByCust USING(cust_id)
                LEFT JOIN tState USING(st)
                LEFT JOIN tStateToDiv USING(st)
                LEFT JOIN tDivision USING(div)
                LEFT JOIN tRegion USING(reg)
                LEFT JOIN tRep USING(rep_id)
                GROUP BY div;""", conn)
```

Out[3]:

	div	TotalSales
0	East North Central	209197.32
1	East South Central	256887.34
2	Middle Atlantic	155926.72
3	Mountain	400839.98
4	New England	282233.87
5	PR	29146.97
6	Pacific	205014.05
7	South Atlantic	453990.54
8	West North Central	306958.08
9	West South Central	223506.65

1. Compare total sales (i.e. sum(qty*unit_price)) for months 1 through 9 in 2020 vs. 2021, for the Northeast region.

Group the sales by sales rep (there should be more than one!)

Return four columns:

- Sales rep name
- Total sales for months 1-9 of 2020
- Total sales for months 1-9 of 2021
- The third column minus the second column (i.e., how much sales have gone up or down relative to last year).

In [100…

```
curs.execute("DROP VIEW IF EXISTS v20;")
curs.execute("""CREATE VIEW v20 AS
                SELECT rep_name, SUM(qty*unit_price) AS TotalSales20
                FROM tDivision
                JOIN tStateToDiv USING(div)
                JOIN vCustomerAddress USING(st)
                JOIN tOrder USING(cust_id)
                JOIN tCustRep USING(cust_id)
                JOIN tRep USING(rep_id)
                JOIN tOrderDetail USING(order_id)
                JOIN tProd USING(prod_id)
                WHERE reg = 'Northeast' AND month BETWEEN '1' AND '9' AND year = '2020'
                GROUP BY rep_name;""")
```

Out[100…

```
<sqlite3.Cursor at 0x7fb9c741be30>
```

In [102…

```
curs.execute("DROP VIEW IF EXISTS v21;")
curs.execute("""CREATE VIEW v21 AS
                SELECT rep_name, SUM(qty*unit_price) AS TotalSales21
                FROM tDivision
                JOIN tStateToDiv USING(div)
                JOIN vCustomerAddress USING(st)
                JOIN tOrder USING(cust_id)
                JOIN tCustRep USING(cust_id)
                JOIN tRep USING(rep_id)
                JOIN tOrderDetail USING(order_id)
                JOIN tProd USING(prod_id)
                WHERE reg = 'Northeast' AND month BETWEEN '1' AND '9' AND year = '2021'
                GROUP BY rep_name;""")
```

Out[102…

```
<sqlite3.Cursor at 0x7fb9c741be30>
```

In [104…

```
pd.read_sql("""SELECT rep_name, IFNULL(TotalSales20,0) AS TotalSales20, IFNULL(TotalSales21,0) AS TotalSales21,
                        (TotalSales21 - TotalSales20) AS SalesDifference
                FROM tRep
```

```
                 JOIN tRegion USING(rep_id)
                 JOIN tCustRep USING(rep_id)
                 JOIN v20 USING(rep_name)
                 JOIN v21 USING(rep_name)
                 GROUP BY rep_name;""", conn)
```

Out[104…

	rep_name	TotalSales20	TotalSales21	SalesDifference
0	Beth	13497.55	45335.95	31838.40
1	Chen	73778.36	191485.83	117707.47