In [585]:
```
import pandas as pd
import sqlite3
```

In [586]:
```
conn = sqlite3.connect('./data/RepStore.db')
curs = conn.cursor()
curs.execute("PRAGMA foreign_keys=ON;")
```

Out[586]:
```
<sqlite3.Cursor at 0x7ff68f1ce880>
```

In [587]:
```
import glob

files = glob.glob('./data/RepAssignments_work/*')
files.sort()
for f in files:
    print(f)
```

```
./data/RepAssignments_work/AllDivisions.xlsx
./data/RepAssignments_work/AllRegions.xlsx
./data/RepAssignments_work/Assignments_ByCust.xlsx
./data/RepAssignments_work/Assignments_ByDivision.xlsx
./data/RepAssignments_work/Assignments_ByRegion.xlsx
./data/RepAssignments_work/Assignments_ByState.xlsx
./data/RepAssignments_work/Assignments_ByZip.xlsx
./data/RepAssignments_work/DivisionToRegion.xlsx
./data/RepAssignments_work/SalesRepList.xlsx
./data/RepAssignments_work/StateToDivision.xlsx
```

In [588]:
```
# Drop all tables if they exist
curs.execute("DROP TABLE IF EXISTS tOrderDetail;")
curs.execute("DROP TABLE IF EXISTS tProd;")
curs.execute("DROP TABLE IF EXISTS tOrder;")
curs.execute("DROP TABLE IF EXISTS tCust;")
curs.execute("DROP TABLE IF EXISTS tZip;")
curs.execute("DROP TABLE IF EXISTS tState;")
```

Out[588]:
```
<sqlite3.Cursor at 0x7ff68f1ce880>
```

In [589]:
```
# Create the product table
sql = """CREATE TABLE tProd (
            prod_id INTEGER PRIMARY KEY,
            prod_name TEXT NOT NULL,
            unit_price NUMERIC NOT NULL);"""
curs.execute(sql)
```

Out[589]:
```
<sqlite3.Cursor at 0x7ff68f1ce880>
```

In [590]:
```
#rep table
curs.execute("DROP TABLE IF EXISTS tRep;")

sql = """CREATE TABLE tRep(
            rep_id INTEGER PRIMARY KEY,
            rep_name TEXT NOT NULL);"""
curs.execute(sql)
```

Out[590]:
```
<sqlite3.Cursor at 0x7ff68f1ce880>
```

In [591]:
```
#representative by region
curs.execute("DROP TABLE IF EXISTS tRepByReg;")

sql = """CREATE TABLE tRepByReg(
            reg TEXT PRIMARY KEY,
            rep_id INTEGER NOT NULL REFERENCES tRep(rep_id));"""
curs.execute(sql)
```

Out[591]:
```
<sqlite3.Cursor at 0x7ff68f1ce880>
```

In [592]:
```
#division to region
curs.execute("DROP TABLE IF EXISTS tDivToReg;")

sql = """CREATE TABLE tDivToReg(
            div TEXT PRIMARY KEY,
            reg TEXT NOT NULL REFERENCES tRepByReg(reg));"""
curs.execute(sql)
```

Out[592]:
```
<sqlite3.Cursor at 0x7ff68f1ce880>
```

In [593]:
```
#rep by division
curs.execute("DROP TABLE IF EXISTS tRepByDiv;")

sql = """CREATE TABLE tRepByDiv(
            div TEXT REFERENCES tDivToReg(div),
            rep_id INTEGER NOT NULL REFERENCES tRep(rep_id),
            PRIMARY KEY (div));"""
curs.execute(sql)
```

Out[593]:
```
<sqlite3.Cursor at 0x7ff68f1ce880>
```

In [594]:
```
# Create the state table
sql = """CREATE TABLE tState (
            st TEXT PRIMARY KEY CHECK(length(st)==2),
            state TEXT NOT NULL,
            div TEXT NOT NULL REFERENCES tDivToReg(div));"""
curs.execute(sql)
```

Out[594]:
```
<sqlite3.Cursor at 0x7ff68f1ce880>
```

In [595]:
```
# Zip code table
sql = """CREATE TABLE tZip (
            zip TEXT PRIMARY KEY CHECK(length(zip)==5),
            city TEXT NOT NULL,
            st TEXT NOT NULL REFERENCES tState(st));"""
curs.execute(sql)
```

Out[595]:
```
<sqlite3.Cursor at 0x7ff68f1ce880>
```

In [596]:
```
#representative by state
curs.execute("DROP TABLE IF EXISTS tRepByState;")

sql = """CREATE TABLE tRepByState(
            st TEXT REFERENCES tState(st),
            rep_id INTEGER NOT NULL REFERENCES tRep(rep_id),
            PRIMARY KEY(st));"""
curs.execute(sql)
```

Out[596]:
```
<sqlite3.Cursor at 0x7ff68f1ce880>
```

In [597]:
```
#representative by zip
curs.execute("DROP TABLE IF EXISTS tRepByZip;")

sql = """CREATE TABLE tRepByZip(
            zip TEXT REFERENCES tZip(zip),
            rep_id INTEGER NOT NULL REFERENCES tRep(rep_id),
            PRIMARY KEY (zip));"""
curs.execute(sql)
```

Out[597]:
```
<sqlite3.Cursor at 0x7ff68f1ce880>
```

In [598]:
```
# Customer table
sql = """CREATE TABLE tCust (
            cust_id INTEGER PRIMARY KEY AUTOINCREMENT,
            first_name TEXT NOT NULL,
            last_name TEXT NOT NULL,
            address TEXT NOT NULL,
            zip TEXT REFERENCES tZip(zip));"""
curs.execute(sql)
```

Out[598]:
```
<sqlite3.Cursor at 0x7ff68f1ce880>
```

In [599]:
```
#representative by customer
curs.execute("DROP TABLE IF EXISTS tRepByCust;")

sql = """CREATE TABLE tRepByCust(
            cust_id INTEGER REFERENCES tCust(cust_id),
            rep_id INTEGER NOT NULL REFERENCES tRep(rep_id),
            PRIMARY KEY (cust_id));"""
curs.execute(sql)
```

Out[599]:
```
<sqlite3.Cursor at 0x7ff68f1ce880>
```

In [600]:
```
# Order table
sql = """CREATE TABLE tOrder (
            order_id INTEGER PRIMARY KEY AUTOINCREMENT,
            cust_id INTEGER REFERENCES tCust(cust_id),
            day INTEGER NOT NULL,
            month INTEGER NOT NULL,
            year INTEGER NOT NULL CHECK(length(year)==4));"""
curs.execute(sql)
```

Out[600]:
```
<sqlite3.Cursor at 0x7ff68f1ce880>
```

In [601]:
```
# Create tOrderDetail
sql = """CREATE TABLE tOrderDetail (
               order_id INTEGER REFERENCES tOrder(order_id),
               prod_id INTEGER REFERENCES tProd(prod_id),
               qty INTEGER NOT NULL,
               PRIMARY KEY (order_id, prod_id));"""
curs.execute(sql)
```

Out[601]:
```
<sqlite3.Cursor at 0x7ff68f1ce880>
```

Load Data

In [602]:
```
# Product data
tProd = pd.read_csv('./data/prods.csv')
sql = "INSERT INTO tProd VALUES(?,?,?);"
for row in tProd.values:
    curs.execute(sql, tuple(row))
```

In [603]:
```
#load representative data
data_rep = pd.read_excel("./data/RepAssignments_work/SalesRepList.xlsx")
sql = "INSERT INTO tRep(rep_name) VALUES (?);"
for row in data_rep.values:
    curs.execute(sql, tuple(row))
```

In [604]:
```
rep = pd.read_sql("SELECT * FROM tRep;", conn)
rep
```

Out[604]:

	rep_id	rep_name
0	1	Frank
1	2	Edgar
2	3	Bob
3	4	Cathy
4	5	Alice
5	6	Diane

In [605]:
```
#load rep_id by region
data_reg = pd.read_excel("./data/RepAssignments_work/Assignments_ByRegion.x
data = pd.merge(data_reg, rep, left_on = 'rep', right_on = 'rep_name')
data = data.drop(['rep_name', 'rep'], axis = 1)
sql = "INSERT INTO tRepByReg VALUES (?,?);"
for row in data.values:
    curs.execute(sql, (row[0], row[1]))
```

In [606]:
```
reg = pd.read_sql("SELECT * FROM tRepByReg;", conn)
reg
```

Out[606]:

	reg	rep_id
0	Midwest	5
1	Northeast	3
2	South	4
3	West	6
4	PR	2

In [607]:
```
#div to reg
div_reg = pd.read_excel('./data/RepAssignments/DivisionToRegion.xlsx')
sql = "INSERT INTO tDivToReg VALUES (?,?);"
for row in div_reg.values:
    curs.execute(sql, (row[0], row[1]))
```

In [608]:
```
div = pd.read_sql("SELECT * FROM tDivToReg;", conn)
div
```

Out[608]:

	div	reg
0	Pacific	West
1	East South Central	South
2	West South Central	South
3	Mountain	West
4	New England	Northeast
5	South Atlantic	South
6	West North Central	Midwest
7	East North Central	Midwest
8	Middle Atlantic	Northeast
9	PR	PR

In [609]:
```
#RepByDiv
data = pd.merge(div, reg, left_on = 'reg', right_on = 'reg')
data_reg = data.drop(['reg'], axis = 1)
rep_div = pd.read_excel('./data/RepAssignments/Assignments_ByDivision.xlsx'
rep_div = rep_div.merge(rep, left_on = 'rep', right_on = 'rep_name')
rep_div = rep_div[['div', 'rep_id']]
combined = data_reg.append(rep_div).drop_duplicates(['div'],keep='last').sc
sql = "INSERT INTO tRepByDiv VALUES (?,?);"
for row in combined.values:
    curs.execute(sql, (row[0], row[1]))
```

In [610]:
```
Repdiv = pd.read_sql("SELECT * FROM tRepByDiv;", conn)
Repdiv
```

Out[610]:

	div	rep_id
0	PR	2
1	Middle Atlantic	3
2	East North Central	3
3	East South Central	4
4	West South Central	4
5	South Atlantic	4
6	New England	4
7	West North Central	5
8	Pacific	6
9	Mountain	6

In [611]:
```
#tState
tState = pd.read_csv('./data/states.csv')
state_div = pd.read_excel('./data/RepAssignments/StateToDivision.xlsx')
data = pd.merge(tState, state_div, left_on='st', right_on = 'state')
data_state = data.drop(['state_y'], axis = 1)
sql = "INSERT INTO tState VALUES (?,?,?);"
for row in data_state.values:
    curs.execute(sql, (row[1], row[0], row[2]))
```

In [612]:
```
states = pd.read_sql("SELECT * FROM tState;", conn)
```

In [613]:
```
#tZip
tZip = pd.read_csv('./data/zips.csv', dtype = {'zip':str})

sql = "INSERT INTO tZip VALUES (?,?,?);"
for row in tZip.values:
    curs.execute(sql, tuple(row))
```

In [614]:
```
zips = pd.read_sql("SELECT * FROM tZip;", conn)
zips
```

Out[614]:

	zip	city	st
0	00601	Adjuntas	PR
1	00602	Aguada	PR
2	00603	Aguadilla	PR
3	00606	Maricao	PR
4	00610	Anasco	PR
...	...	...	...
33094	99923	Hyder	AK
33095	99925	Klawock	AK
33096	99926	Metlakatla	AK
33097	99927	Point Baker	AK
33098	99929	Wrangell	AK

33099 rows × 3 columns

In [615]:
```
#tRepByState
data_by_state = pd.merge(states, Repdiv, left_on = 'div', right_on = 'div')
#print(data_by_state)
data_by_state = data_by_state.drop(['state','div'], axis = 1)
reg_state = pd.read_excel('./data/RepAssignments/Assignments_ByState.xlsx')
reg_state = reg_state.merge(rep, left_on = 'rep', right_on = 'rep_name')
reg_state = reg_state[['st', 'rep_id']]
combined = data_by_state.append(reg_state).drop_duplicates(['st'],keep='las
sql = "INSERT INTO tRepByState VALUES (?,?);"
for row in combined.values:
    curs.execute(sql, (row[0], row[1]))
```

In [616]:
```
by_state = pd.read_sql("SELECT * FROM tRepByState ORDER BY st;", conn)
by_state
```

Out[616]:

	st	rep_id
0	AK	6
1	AL	4
2	AR	4
3	AZ	6
4	CA	6
5	CO	6
6	CT	4
7	DC	4
8	DE	4
9	FL	4
10	GA	4
11	HI	6
12	IA	5
13	ID	6
14	IL	3
15	IN	3
16	KS	5
17	KY	4
18	LA	4
19	MA	3
20	MD	4
21	ME	4
22	MI	3
23	MN	5
24	MO	5
25	MS	4
26	MT	6
27	NC	4
28	ND	5
29	NE	5
30	NH	4
31	NJ	3
32	NM	6

	st	rep_id
33	NV	6
34	NY	3
35	OH	3
36	OK	4
37	OR	6
38	PA	3
39	PR	2
40	RI	4
41	SC	4
42	SD	5
43	TN	4
44	TX	4
45	UT	6
46	VA	4
47	VT	4
48	WA	6
49	WI	3
50	WV	4
51	WY	6

In [617]:
```
#tRepByZip
states_zips = pd.merge(by_state, zips, left_on = 'st', right_on= 'st')
states_zips = states_zips[['zip','rep_id']]
reg_zip_override = pd.read_excel('./data/RepAssignments/Assignments_ByZip.x
reg_zip_override = reg_zip_override.astype(str)
reg_zip_override = reg_zip_override.merge(rep, left_on = 'rep', right_on =
reg_zip_override = reg_zip_override[['zip', 'rep_id']]
combined = states_zips.append(reg_zip_override).drop_duplicates(['zip'],kee
#reg_zip_override = reg_zip_override.drop(['rep', 'rep_name'], axis = 1)
#states_zips = pd.merge(states_zips, reg_zip_override, left_on = 'zip', rig
#combined.loc[combined['zip']=='92155']
#combined
sql = "INSERT INTO tRepByZip VALUES (?,?);"
for row in combined.values:
    try:
        curs.execute(sql, (row[0], row[1]))
    except:
        print(row)
```

In [618]:
```
by_zip = pd.read_sql("SELECT * FROM tRepByZip;", conn)
by_zip
```

Out[618]:

	zip	rep_id
0	00601	2
1	00602	2
2	00603	2
3	00606	2
4	00610	2
...	...	...
33094	99923	6
33095	99925	6
33096	99926	6
33097	99927	6
33098	99929	6

33099 rows × 2 columns

In [619]:
```
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,

        # 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
            curs.execute(sql_insert, (first_name,last_name,address,str(zip_
            cust = pd.read_sql(sql, conn, params=(first_name,last_name,addr

        return cust['cust_id'][0]

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, y
                                   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]
```

In [620]:
```
def LoadData(file):
    data = pd.read_csv(str(file), dtype={'zip':str})

    cust = data[['first','last','addr','city','state','zip']].drop_duplicat

    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'])
    to_drop = [x for x in data_with_cust if x.endswith('_y')]
    data_with_cust.drop(to_drop, axis = 1, inplace = True)
    # Get all the customer id / dates
    orders  = data_with_cust[['cust_id', 'date']].drop_duplicates()
    orders[['year','month','day']] = orders['date'].str.split('-',expand=Tr
    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

    # Fill in tOrderDetail
    COL_ORDER_ID = 15
    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
```

In [621]:
```
files_sales = glob.glob('./data/sales_data/*S*')
files_sales.sort()
for f in files_sales:
    print(f)
```

```
./data/sales_data/Sales_201901.csv
./data/sales_data/Sales_201902.csv
./data/sales_data/Sales_201903.csv
./data/sales_data/Sales_201904.csv
./data/sales_data/Sales_201905.csv
./data/sales_data/Sales_201906.csv
./data/sales_data/Sales_201907.csv
./data/sales_data/Sales_201908.csv
./data/sales_data/Sales_201909.csv
./data/sales_data/Sales_201910.csv
./data/sales_data/Sales_201911.csv
./data/sales_data/Sales_201912.csv
./data/sales_data/Sales_202001.csv
./data/sales_data/Sales_202002.csv
./data/sales_data/Sales_202003.csv
./data/sales_data/Sales_202004.csv
./data/sales_data/Sales_202005.csv
./data/sales_data/Sales_202006.csv
./data/sales_data/Sales_202007.csv
./data/sales_data/Sales_202008.csv
./data/sales_data/Sales_202009.csv
./data/sales_data/Sales_202010.csv
```

In [622]:
```
for f in files_sales:
    LoadData(f)
```

In [623]:
```
pd.read_sql("SELECT * FROM sqlite_master;", conn)
```

Out[623]:

	type	name	tbl_name	rootpage	sql
0	table	tProd	tProd	2	CREATE TABLE tProd (\n prod_id INTE...
1	table	tRep	tRep	3	CREATE TABLE tRep(\n rep_id INTEGER...
2	table	tRepByReg	tRepByReg	4	CREATE TABLE tRepByReg(\n reg TEXT ...
3	index	sqlite_autoindex_tRepByReg_1	tRepByReg	5	None
4	table	tDivToReg	tDivToReg	6	CREATE TABLE tDivToReg(\n div TEXT ...
5	index	sqlite_autoindex_tDivToReg_1	tDivToReg	7	None
6	table	tRepByDiv	tRepByDiv	8	CREATE TABLE tRepByDiv(\n div TEXT ...
7	index	sqlite_autoindex_tRepByDiv_1	tRepByDiv	9	None
8	table	tState	tState	10	CREATE TABLE tState (\n st TEXT PRI...
9	index	sqlite_autoindex_tState_1	tState	11	None
10	table	tZip	tZip	12	CREATE TABLE tZip (\n zip TEXT PRIM...
11	index	sqlite_autoindex_tZip_1	tZip	13	None
12	table	tRepByState	tRepByState	14	CREATE TABLE tRepByState(\n st TEXT...
13	index	sqlite_autoindex_tRepByState_1	tRepByState	15	None
14	table	tRepByZip	tRepByZip	16	CREATE TABLE tRepByZip(\n zip TEXT ...
15	index	sqlite_autoindex_tRepByZip_1	tRepByZip	17	None
16	table	tCust	tCust	18	CREATE TABLE tCust (\n cust_id INTE...
17	table	sqlite_sequence	sqlite_sequence	19	CREATE TABLE sqlite_sequence(name,seq)
18	table	tRepByCust	tRepByCust	20	CREATE TABLE tRepByCust(\n cust_id ...
19	table	tOrder	tOrder	21	CREATE TABLE tOrder (\n order_id IN...
20	table	tOrderDetail	tOrderDetail	22	CREATE TABLE tOrderDetail (\n o...
21	index	sqlite_autoindex_tOrderDetail_1	tOrderDetail	23	None

In [624]:
```
cust = pd.read_sql("SELECT * FROM tCust;", conn)
cust
```

Out[624]:

	cust_id	first_name	last_name	address	zip
0	1	Bib Fortuna	Walker	6829 2nd Street	10177
1	2	Unkar Plutt	Jennings	5295 4th Street South	35130
2	3	Dodonna	Garza	3639 Briarwood Court	79783
3	4	Rabe	Woodward	2517 Lake Avenue	18505
4	5	Plo Koon	Ferguson	3332 Prospect Street	14433
...	...	...	...	...	...
305	306	C-3Po	Alvarez	3798 Park Avenue	59079
306	307	Lama Su	Vincent	2114 4th Street	23866
307	308	Lama Su	Schmidt	1947 College Street	06441
308	309	Rabe	Greene	8560 Pheasant Run	42746
309	310	Tc-14	Rodriguez	3572 Franklin Court	16323

310 rows × 5 columns

In [625]:
```
#load tRepByCust
cust = cust[['cust_id', 'zip']]
cust_reps = pd.merge(cust, by_zip, left_on = 'zip', right_on = 'zip')
cust_reps = cust_reps.drop('zip', axis = 1)
reg_clust = pd.read_excel('./data/RepAssignments/Assignments_ByCust.xlsx')
reg_clust = pd.merge(reg_clust, rep, left_on = 'rep', right_on = 'rep_name'
reg_clust = reg_clust[['cust_id', 'rep_id']]
combined = cust_reps.append(reg_clust).drop_duplicates(['cust_id'],keep='la
sql = "INSERT INTO tRepByCust VALUES (?,?);"
for row in combined.values:
    curs.execute(sql, (int(row[0]), int(row[1])))
```

In [626]:
```
rep_cust = pd.read_sql("SELECT * FROM tRepByCust;", conn)
rep_cust
```

Out[626]:

	cust_id	rep_id
0	1	3
1	2	4
2	3	4
3	4	3
4	5	3
...	...	...
305	306	6
306	307	4
307	308	4
308	309	4
309	310	3

310 rows × 2 columns

In [627]:
```
x = pd.read_sql("""SELECT name
                FROM sqlite_master
                WHERE type = 'table'
                AND name LIKE 't%';""",conn)
#x
for table in x.values:
    sql = "PRAGMA table_info(" + table[0] + ");"
    print(table)
    print(pd.read_sql(sql,conn))
    print('\n')
```

```
['tProd']
   cid        name     type  notnull dflt_value  pk
0    0     prod_id  INTEGER        0       None   1
1    1   prod_name     TEXT        1       None   0
2    2  unit_price  NUMERIC        1       None   0


['tRep']
   cid      name     type  notnull dflt_value  pk
0    0    rep_id  INTEGER        0       None   1
1    1  rep_name     TEXT        1       None   0


['tRepByReg']
   cid    name     type  notnull dflt_value  pk
0    0     reg     TEXT        0       None   1
1    1  rep_id  INTEGER        1       None   0


['tDivToReg']
   cid name  type  notnull dflt_value  pk
0    0  div  TEXT        0       None   1
1    1  reg  TEXT        1       None   0


['tRepByDiv']
   cid    name     type  notnull dflt_value  pk
0    0     div     TEXT        0       None   1
1    1  rep_id  INTEGER        1       None   0


['tState']
   cid   name  type  notnull dflt_value  pk
0    0     st  TEXT        0       None   1
1    1  state  TEXT        1       None   0
2    2    div  TEXT        1       None   0


['tZip']
   cid  name  type  notnull dflt_value  pk
0    0   zip  TEXT        0       None   1
1    1  city  TEXT        1       None   0
2    2    st  TEXT        1       None   0


['tRepByState']
```

```
   cid    name     type  notnull dflt_value  pk
0    0      st     TEXT        0       None   1
1    1  rep_id  INTEGER        1       None   0


['tRepByZip']
   cid    name     type  notnull dflt_value  pk
0    0     zip     TEXT        0       None   1
1    1  rep_id  INTEGER        1       None   0


['tCust']
   cid        name     type  notnull dflt_value  pk
0    0     cust_id  INTEGER        0       None   1
1    1  first_name     TEXT        1       None   0
2    2   last_name     TEXT        1       None   0
3    3     address     TEXT        1       None   0
4    4         zip     TEXT        0       None   0


['tRepByCust']
   cid     name     type  notnull dflt_value  pk
0    0  cust_id  INTEGER        0       None   1
1    1   rep_id  INTEGER        1       None   0


['tOrder']
   cid      name     type  notnull dflt_value  pk
0    0  order_id  INTEGER        0       None   1
1    1   cust_id  INTEGER        0       None   0
2    2       day  INTEGER        1       None   0
3    3     month  INTEGER        1       None   0
4    4      year  INTEGER        1       None   0


['tOrderDetail']
   cid      name     type  notnull dflt_value  pk
0    0  order_id  INTEGER        0       None   1
1    1   prod_id  INTEGER        0       None   2
2    2       qty  INTEGER        1       None   0
```

In [628]:
```
pd.read_sql("SELECT * FROM tOrderDetail;", conn)
```

Out[628]:

	order_id	prod_id	qty
0	1	309	6
1	2	307	7
2	2	311	8
3	2	327	5
4	2	306	5
...	...	...	...
53424	4627	323	8
53425	4627	306	4
53426	4627	313	8
53427	4627	327	4
53428	4627	326	9

53429 rows × 3 columns

In [629]:
```
conn.commit()
```

In [630]:
```
conn.close()
```