SQLProject

November 4, 2024

1 Data 1050 SQL Project

1.1 Create Database

[1]:
```
pip install mysql-connector-python # type: ignore
```

```
Requirement already satisfied: mysql-connector-python in
/opt/anaconda3/lib/python3.12/site-packages (9.0.0)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
import mysql.connector # type: ignore

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password= "7@u*sqJX"    #REPLACE THIS WITH THE PASSWORD YOU SET
)

print(mydb)

if mydb.is_connected():
    print("CONNECTION SUCCESSFUL")
```

```
<mysql.connector.connection_cext.CMySQLConnection object at 0x1082fef30>
CONNECTION SUCCESSFUL
```

[3]:
```
#create a database
mycursor = mydb.cursor()
mycursor.execute("DROP DATABASE IF EXISTS data1050SQLProject")
mycursor.execute("CREATE DATABASE data1050SQLProject")
```

[4]:
```
mycursor = mydb.cursor()
mycursor.execute("SHOW DATABASES")

for x in mycursor:
  print(x)
```

```
('data1050f24',)
('data1050SQLProject',)
('information_schema',)
```

```
('mysql',)
('performance_schema',)
('sys',)
```

1.1.1 Connecting to a database and showing tables

[5]:
```
#connect to a specific database
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password= "7@u*sqJX", #REPLACE THIS WITH YOUR PASSWORD
  database = "data1050SQLProject" #connecting to testDatabase
)
```

[6]:
```
mycursor = mydb.cursor()

mycursor.execute("SHOW TABLES")

for x in mycursor:
  print(x)
```

[7]:
```
import pandas as pd # type: ignore
```

[8]:
```
pip install fsspec # type: ignore
```

```
Requirement already satisfied: fsspec in /opt/anaconda3/lib/python3.12/site-
packages (2024.6.1)
Note: you may need to restart the kernel to use updated packages.
```

1.1.2 Add Tables

[9]:
```
#create a table
mycursor = mydb.cursor()
mycursor.execute("DROP TABLE IF EXISTS data1050SQLProject.physicians")
mycursor.execute('''CREATE TABLE data1050SQLProject.physicians (
                    ssn VARCHAR(128) PRIMARY KEY,
                    name VARCHAR(128),
                    primary_specialty VARCHAR(128),
                    experience_years INT);''')
```

[10]:
```
#create a table
mycursor = mydb.cursor()
mycursor.execute("DROP TABLE IF EXISTS data1050SQLProject.patients")
mycursor.execute('''CREATE TABLE data1050SQLProject.patients (
                    ssn VARCHAR(128) PRIMARY KEY,
                    name VARCHAR(128),
                    address VARCHAR(128),
                    birth_date VARCHAR(128),
```

```
                    physician_id VARCHAR(128),
                    foreign key(physician_id) references data1050SQLProject.
 ↪physicians(ssn));''')
```

[11]:
```
#create a table
mycursor = mydb.cursor()
mycursor.execute("DROP TABLE IF EXISTS data1050SQLProject.pharmacies")
mycursor.execute('''CREATE TABLE data1050SQLProject.pharmacies (
                    id INT PRIMARY KEY,
                    name VARCHAR(128),
                    address VARCHAR(128),
                    phone VARCHAR(128));''')
```

[12]:
```
#create a table
mycursor = mydb.cursor()
mycursor.execute("DROP TABLE IF EXISTS data1050SQLProject.drugs")
mycursor.execute('''CREATE TABLE data1050SQLProject.drugs (
                    id VARCHAR(128),
                    name VARCHAR(128) PRIMARY KEY);''')
```

[13]:
```
#create a table
mycursor = mydb.cursor()
mycursor.execute("DROP TABLE IF EXISTS data1050SQLProject.prescriptions")
mycursor.execute('''CREATE TABLE data1050SQLProject.prescriptions (
                    id INT PRIMARY KEY,
                    patient_id VARCHAR(128),
                    physician_id VARCHAR(128),
                    drug_name VARCHAR(128),
                    date VARCHAR(128),
                    quantity INT,
                    INDEX(patient_id,drug_name),
                    foreign key(patient_id) references data1050SQLProject.
 ↪patients(ssn),
                    foreign key(physician_id) references data1050SQLProject.
 ↪physicians(ssn),
                    foreign key(drug_name) references data1050SQLProject.
 ↪drugs(name));''')
```

[14]:
```
#create a table
mycursor = mydb.cursor()
mycursor.execute("DROP TABLE IF EXISTS data1050SQLProject.adverse_interactions")
mycursor.execute('''CREATE TABLE data1050SQLProject.adverse_interactions (
                    drug_name VARCHAR(128),
                    drug_name_2 VARCHAR(128),
                    PRIMARY KEY (drug_name,drug_name_2),
                    foreign key(drug_name) references data1050SQLProject.
 ↪drugs(name));''')
```

[15]:
```
#create a table
mycursor = mydb.cursor()
mycursor.execute("DROP TABLE IF EXISTS data1050SQLProject.alerts")
mycursor.execute('''CREATE TABLE data1050SQLProject.alerts (
                    patient_id VARCHAR(128),
                    physician_id VARCHAR(128),
                    alert_date VARCHAR(128),
                    drug1 VARCHAR(128),
                    drug2 VARCHAR(128),
                    PRIMARY KEY␣
 ↪(patient_id,physician_id,alert_date,drug1,drug2),
                    foreign key(patient_id) references data1050SQLProject.
 ↪patients(ssn),
                    foreign key(physician_id) references data1050SQLProject.
 ↪physicians(ssn),
                    foreign key(patient_id,drug1) references data1050SQLProject.
 ↪prescriptions(patient_id,drug_name),
                    foreign key(patient_id,drug2) references data1050SQLProject.
 ↪prescriptions(patient_id,drug_name)
                );''')
```

[16]:
```
mycursor = mydb.cursor()
mycursor.execute("DROP TABLE IF EXISTS data1050SQLProject.pharmacy_fills")
mycursor.execute('''CREATE TABLE data1050SQLProject.pharmacy_fills (
                    pharmacy_id INT,
                    prescription_id INT,
                    date VARCHAR(128),
                    cost DECIMAL(5,2),
                    PRIMARY KEY (prescription_id,pharmacy_id),
                    foreign key(prescription_id) references data1050SQLProject.
 ↪prescriptions(id),
                    foreign key(pharmacy_id) references data1050SQLProject.
 ↪pharmacies(id));''')
```

[17]:
```
mycursor = mydb.cursor()
mycursor.execute("DROP TABLE IF EXISTS data1050SQLProject.companies")
mycursor.execute('''CREATE TABLE data1050SQLProject.companies (
                    id INT PRIMARY KEY,
                    name VARCHAR(128),
                    address VARCHAR(128),
                    contact_phone VARCHAR(128),
                    contact_name VARCHAR(128));''')
```

[18]:
```
mycursor = mydb.cursor()
mycursor.execute("DROP TABLE IF EXISTS data1050SQLProject.contracts")
mycursor.execute('''CREATE TABLE data1050SQLProject.contracts (
                    id INT PRIMARY KEY,
```

```
                    drug_name VARCHAR(128),
                    dosage INT,
                    pharmacy_id INT,
                    company_id INT,
                    quantity INT,
                    date VARCHAR(128),
                    price INT,
                    foreign key(company_id) references data1050SQLProject.
 ↪companies(id),
                    foreign key(pharmacy_id) references data1050SQLProject.
 ↪pharmacies(id),
                    foreign key(drug_name) references data1050SQLProject.
 ↪drugs(name));''')
```

[19]:
```
mycursor = mydb.cursor()

mycursor.execute("SHOW TABLES")

for x in mycursor:
  print(x)
```

```
('adverse_interactions',)
('alerts',)
('companies',)
('contracts',)
('drugs',)
('patients',)
('pharmacies',)
('pharmacy_fills',)
('physicians',)
('prescriptions',)
```

1.1.3 Add Data to Tables

[20]:
```
#point the path to where in your hard drive you have stored the file physicians.
 ↪csv
physicians_df = pd.read_csv("physicians.csv")
physicians_df.head()
```

[20]:

	SSN	name	primary_specialty	experience_years
0	614-57-6885	Srinivasan	Cardiology	4
1	702-16-8749	Wu	Dermatology	10
2	571-13-9020	Mozart	Cardiology	0
3	718-27-0905	Einstein	Psychiatry	29
4	230-12-3219	El Said	Psychiatry	12

[21]:
```
physicians_df.dtypes
```

[21]:
```
SSN                 object
name                object
primary_specialty   object
experience_years     int64
dtype: object
```

[22]:
```
for i,row in physicians_df.iterrows():
            sql = "INSERT INTO physicians VALUES (%s,%s,%s,%s)"
            mycursor.execute(sql, tuple(row))
            print("Record inserted")
            # the connection is not autocommitted by default, so we
            # must commit to save our changes
            mydb.commit()
```

```
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
```

[23]:
```
query = ''' SELECT * from physicians'''
mycursor.execute(query)
```

[24]:
```
for x in mycursor:
    print(x)
```

```
('118-66-5958', 'Katz', 'Orthopedics', 3)
('156-28-1945', 'Singh', 'Orthopedics', 25)
('163-50-5535', 'Gold', 'Neurology', 8)
('230-12-3219', 'El Said', 'Psychiatry', 12)
('357-93-5814', 'Califieri', 'Cardiology', 21)
('460-35-6754', 'Kim', 'Orthopedics', 2)
('510-55-9776', 'Brandt', 'Psychiatry', 25)
('522-86-5827', 'Crick', 'Neurology', 0)
('571-13-9020', 'Mozart', 'Cardiology', 0)
('614-57-6885', 'Srinivasan', 'Cardiology', 4)
('702-16-8749', 'Wu', 'Dermatology', 10)
('718-27-0905', 'Einstein', 'Psychiatry', 29)
```

[25]:
```
#point the path to where in your hard drive you have stored the file patients.
 ↪csv
```

```
data = pd.read_csv("patients.csv")
data.head()
```

[25]:

	ssn	name	address	birthdate	physician_id
0	478-34-0781	Florance Saiz	7 Fair Oaks Place	1988-11-03T23:25:38Z	614-57-6885
1	885-94-4721	Merry Di Pietro	1 Old Shore Court	1991-02-07T22:00:41Z	702-16-8749
2	777-39-3296	Myron Cottem	75875 Fulton Crossing	1986-02-20T04:43:29Z	718-27-0905
3	227-08-7452	Bearnard Remer	18669 Heffernan Point	2008-01-09T05:34:30Z	230-12-3219
4	805-15-2755	Roxana Worster	54 Hudson Junction	1982-11-12T18:11:55Z	163-50-5535

[26]:
```
data.dtypes
```

[26]:
```
ssn             object
name            object
address         object
birthdate       object
physician_id    object
dtype: object
```

[27]:
```
query = "SELECT * FROM physicians WHERE ssn = '614-57-6885';"
mycursor.execute(query)
```

[28]:
```
for x in mycursor:
    print(x)
```

```
('614-57-6885', 'Srinivasan', 'Cardiology', 4)
```

[29]:
```
for i,row in data.iterrows():
            sql = "INSERT INTO patients VALUES (%s,%s,%s,%s,%s)"
            mycursor.execute(sql, tuple(row))
            print("Record inserted")
            # the connection is not autocommitted by default, so we
            # must commit to save our changes
            mydb.commit()
```

```
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
```

```
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
```

[30]:
```
query = "SELECT * from patients"
mycursor.execute(query)
```

[31]:
```
for x in mycursor:
    print(x)
```

```
('192-33-2887', 'Jacinda Stowe', '8 Colorado Alley', '1970-04-15T00:24:26Z',
'357-93-5814')
('227-08-7452', 'Bearnard Remer', '18669 Heffernan Point',
'2008-01-09T05:34:30Z', '230-12-3219')
('303-13-5928', 'Krystyna Luckie', '54106 Barnett Plaza',
'1950-02-11T12:20:13Z', '571-13-9020')
('360-47-2098', 'Peter Lukasen', '552 Ryan Court', '1969-01-10T19:33:03Z',
'522-86-5827')
('478-34-0781', 'Florance Saiz', '7 Fair Oaks Place', '1988-11-03T23:25:38Z',
'614-57-6885')
('501-47-2038', 'Elvyn Rudinger', '48 Bowman Parkway', '2006-02-28T16:26:43Z',
'156-28-1945')
('631-75-6048', 'Avrom Messer', '5030 Garrison Center', '1929-02-04T06:34:10Z',
'510-55-9776')
('691-21-7304', 'Myrlene Yegoshin', '2 Sunnyside Court', '2001-06-03T23:02:52Z',
'460-35-6754')
('758-08-7274', 'Susanetta Petruska', '16276 Sutteridge Avenue',
'1922-08-05T18:36:12Z', '118-66-5958')
('777-39-3296', 'Myron Cottem', '75875 Fulton Crossing', '1986-02-20T04:43:29Z',
'718-27-0905')
('805-15-2755', 'Roxana Worster', '54 Hudson Junction', '1982-11-12T18:11:55Z',
'163-50-5535')
('885-94-4721', 'Merry Di Pietro', '1 Old Shore Court', '1991-02-07T22:00:41Z',
'702-16-8749')
```

[32]:
```
#point the path to where in your hard drive you have stored the file pharmacies.
↪csv
data = pd.read_csv("pharmacies.csv")
data.head()
```

[32]:

	id	name	address
0	1	Springfield Pharmacy	123 Main St, Springfield, IL 62701
1	2	Peachtree Meds	456 Elm St, Atlanta, GA 30303
2	3	Lone Star Drugs	789 Oak St, Dallas, TX 75201
3	4	Mile High Meds	101 Pine St, Denver, CO80202
4	5	Emerald City Pharmacy	121 Spruce St, Seattle, WA 98101

```
              phone
0   (217) 555-1234
1   (404) 555-5678
2   (214) 555-9101
3   (303) 555-1121
4   (206) 555-1314
```

[33]:
```
data.dtypes
```

[33]:
```
id           int64
 name       object
 address    object
phone       object
dtype: object
```

[34]:
```
for i,row in data.iterrows():
            sql = "INSERT INTO pharmacies VALUES (%s,%s,%s,%s)"
            mycursor.execute(sql, tuple(row))
            print("Record inserted")
            # the connection is not autocommitted by default, so we
            # must commit to save our changes
            mydb.commit()
```

```
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
```

[35]:
```
query = " SELECT * FROM pharmacies"
mycursor.execute(query)
```

[36]:
```
for x in mycursor:
    print(x)
```

```
(1, ' Springfield Pharmacy', ' 123 Main St,  Springfield,  IL 62701', ' (217)
555-1234')
```

```
(2, ' Peachtree Meds', ' 456 Elm St,  Atlanta, GA 30303', ' (404) 555-5678')
(3, ' Lone Star Drugs', ' 789 Oak St,  Dallas, TX 75201', ' (214) 555-9101')
(4, ' Mile High Meds', ' 101 Pine St, Denver, CO80202', ' (303) 555-1121')
(5, ' Emerald City Pharmacy', ' 121 Spruce St,  Seattle,  WA 98101', ' (206)
555-1314')
(6, ' Golden Gate Drugs', ' 234 Market St, San Francisco, CA 94105', ' (415)
555-1515')
(7, ' Sunshine Pharmacy', ' 345 Palm Ave, Miami, FL 33101', ' (305) 555-1616')
(8, ' Liberty Meds', ' 567 Broadway St, New York, NY 10001', ' (212) 555-1717')
(9, ' Lakeside Drugs', ' 678 Lake Rd,  Minneapolis, MN 55401', ' (612)
555-1818')
(10, ' Desert Bloom Pharmacy', ' 890 Desert Blvd, Phoenix, AZ 85001', ' (602)
555-1919')
(11, ' Bayside Pharmacy', ' 112 Harbor Dr, San Diego, CA 92101', ' (619)
555-2020')
(12, ' Capital Meds', ' 345 Capitol St, Washington, DC 20001', ' (202)
555-2121')
(13, ' Windy City Pharmacy', ' 567 Windy Ave, Chicago, IL 60601', ' (312)
555-2222')
(14, ' Beantown Drugs', ' 890 Beacon St, Boston,  MA 02101', ' (617) 555-2323')
(15, ' Gateway Meds', ' 123 Arch St, St. Louis, MO 63101', ' (314) 555-2424')
```

[37]:
```
#point the path to where in your hard drive you have stored the file drugs.csv
data = pd.read_csv("drugs.csv")
data.head()
```

[37]:

	drug_id	drug_name
0	1	Primalovir
1	2	Olanzanafine
2	3	Avafoxin
3	4	Quixiposide
4	5	Cleotrana

[38]:
```
data.dtypes
```

[38]:
```
drug_id       int64
drug_name    object
dtype: object
```

[39]:
```
for i,row in data.iterrows():
            sql = "INSERT INTO drugs VALUES (%s,%s)"
            mycursor.execute(sql, tuple(row))
            print("Record inserted")
            # the connection is not autocommitted by default, so we
            # must commit to save our changes
            mydb.commit()
```

```
Record inserted
```

```
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
```

[40]:
```
query = "SELECT * FROM drugs"
mycursor.execute(query)
```

[41]:
```
for x in mycursor:
    print(x)
```

```
('9', 'Abnazole Toleluble')
('3', 'Avafoxin')
('5', 'Cleotrana')
('10', 'Dantopex Quixilinum')
('8', 'Divisporine Acetaclotide')
('7', 'Glucozepam Amcipentin')
('6', 'Kanulin')
('2', 'Olanzanafine')
('1', 'Primalovir')
('4', 'Quixiposide')
```

[42]:
```
#point the path to where in your hard drive you have stored the file␣
 ↪prescriptions.csv
data = pd.read_csv("prescriptions.csv")
data.head()
```

[42]:

	id	patient_id	physician_id	drug_name	date	quantity
0	1	478-34-0781	614-57-6885	Avafoxin	3/11/2023	90
1	2	758-08-7274	118-66-5958	Cleotrana	3/12/2023	10
2	3	758-08-7274	118-66-5958	Primalovir	4/11/2023	20
3	4	758-08-7274	118-66-5958	Glucozepam Amcipentin	5/13/2023	12
4	5	303-13-5928	571-13-9020	Olanzanafine	5/24/2023	25

[43]:
```
data.dtypes
```

[43]:
```
id              int64
patient_id     object
physician_id   object
drug_name      object
date           object
quantity        int64
dtype: object
```

[44]:
```
for i,row in data.iterrows():
            sql = "INSERT INTO prescriptions VALUES (%s,%s,%s,%s,%s,%s)"
            mycursor.execute(sql, tuple(row))
            print("Record inserted")
            # the connection is not autocommitted by default, so we
            # must commit to save our changes
            mydb.commit()
```

```
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
```

[45]:
```
query = "SELECT * FROM prescriptions"
mycursor.execute(query)
```

[46]:
```
for x in mycursor:
    print(x)
```

```
(1, '478-34-0781', '614-57-6885', 'Avafoxin', '3/11/2023', 90)
(2, '758-08-7274', '118-66-5958', 'Cleotrana', '3/12/2023', 10)
(3, '758-08-7274', '118-66-5958', 'Primalovir', '4/11/2023', 20)
(4, '758-08-7274', '118-66-5958', 'Glucozepam Amcipentin', '5/13/2023', 12)
(5, '303-13-5928', '571-13-9020', 'Olanzanafine', '5/24/2023', 25)
(6, '303-13-5928', '571-13-9020', 'Primalovir', '5/24/2023', 16)
(7, '303-13-5928', '571-13-9020', 'Abnazole Toleluble', '5/24/2023', 5)
(8, '478-34-0781', '614-57-6885', 'Avafoxin', '6/14/2023', 60)
(9, '303-13-5928', '571-13-9020', 'Glucozepam Amcipentin', '6/22/2023', 3)
(10, '501-47-2038', '156-28-1945', 'Cleotrana', '7/18/2023', 20)
(11, '777-39-3296', '718-27-0905', 'Dantopex Quixilinum', '8/2/2023', 1)
(12, '501-47-2038', '156-28-1945', 'Cleotrana', '8/21/2023', 10)
(13, '478-34-0781', '614-57-6885', 'Avafoxin', '9/17/2023', 30)
(14, '478-34-0781', '614-57-6885', 'Quixiposide', '9/17/2023', 2)
(15, '501-47-2038', '156-28-1945', 'Avafoxin', '9/22/2023', 15)
(16, '501-47-2038', '156-28-1945', 'Kanulin', '9/22/2023', 8)
```

[47]:
```
#point the path to where in your hard drive you have stored the file␣
 ↪prescriptions.csv
data = pd.read_csv("prescriptions.csv")
data.head()
```

[47]:

	id	patient_id	physician_id	drug_name	date	quantity
0	1	478-34-0781	614-57-6885	Avafoxin	3/11/2023	90
1	2	758-08-7274	118-66-5958	Cleotrana	3/12/2023	10
2	3	758-08-7274	118-66-5958	Primalovir	4/11/2023	20
3	4	758-08-7274	118-66-5958	Glucozepam Amcipentin	5/13/2023	12
4	5	303-13-5928	571-13-9020	Olanzanafine	5/24/2023	25

[48]:
```
#point the path to where in your hard drive you have stored the file␣
 ↪adverse_interactions.csv
data = pd.read_csv("adverse_reactions.csv")
data.head()
```

[48]:

	drug_name_1	drug_name_2
0	Cleotrana	Kanulin
1	Primalovir	Abnazole Toleluble
2	Primalovir	Olanzanafine
3	Olanzanafine	Glucozepam Amcipentin
4	Avafoxin	Kanulin

[49]:
```
data.dtypes
```

[49]:
```
drug_name_1    object
drug_name_2    object
dtype: object
```

[50]:
```
for i,row in data.iterrows():
            sql = "INSERT INTO adverse_interactions VALUES (%s,%s)"
            mycursor.execute(sql, tuple(row))
            print("Record inserted")
            # the connection is not autocommitted by default, so we
            # must commit to save our changes
            mydb.commit()
```

```
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
```

[51]:
```
query = "SELECT * FROM adverse_interactions"
mycursor.execute(query)
```

[52]:
```
for x in mycursor:
    print(x)
```

```
('Avafoxin', 'Kanulin')
('Cleotrana', 'Avafoxin')
('Cleotrana', 'Kanulin')
('Cleotrana', 'Quixiposide')
('Olanzanafine', 'Glucozepam Amcipentin')
('Primalovir', 'Abnazole Toleluble')
('Primalovir', 'Olanzanafine')
('Quixiposide', 'Avafoxin')
('Quixiposide', 'Dantopex Quixilinum')
```

[53]:
```
#point the path to where in your hard drive you have stored the file␣
 ↪pharmacy_fills.csv
data = pd.read_csv("pharmacy_fills.csv")
data.head()
```

[53]:

	pharmacy_id	prescription_id	date	cost
0	1	3	3/15/2023	60.53
1	3	4	5/16/2023	41.50
2	1	2	3/12/2023	18.00
3	8	1	3/12/2023	46.53
4	10	5	5/26/2023	47.50

[54]:
```
data.dtypes
```

[54]:
```
pharmacy_id          int64
prescription_id      int64
date                object
cost               float64
dtype: object
```

[55]:
```
for i,row in data.iterrows():
            sql = "INSERT INTO pharmacy_fills VALUES (%s,%s,%s,%s)"
            mycursor.execute(sql, tuple(row))
            print("Record inserted")
            # the connection is not autocommitted by default, so we
            # must commit to save our changes
            mydb.commit()
```

```
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
```

```
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
```

[56]:
```
query = "SELECT * FROM pharmacy_fills"
mycursor.execute(query)
```

[57]:
```
for x in mycursor:
    print(x)
```

```
(8, 1, '3/12/2023', Decimal('46.53'))
(1, 2, '3/12/2023', Decimal('18.00'))
(1, 3, '3/15/2023', Decimal('60.53'))
(3, 4, '5/16/2023', Decimal('41.50'))
(10, 5, '5/26/2023', Decimal('47.50'))
(7, 6, '5/24/2023', Decimal('92.10'))
(5, 7, '5/28/2023', Decimal('41.65'))
(5, 8, '6/15/2023', Decimal('94.60'))
(11, 9, '6/22/2023', Decimal('31.00'))
(2, 10, '7/22/2023', Decimal('14.55'))
(12, 11, '8/4/2023', Decimal('92.00'))
(14, 12, '8/21/2023', Decimal('42.85'))
(6, 13, '9/19/2023', Decimal('31.65'))
(7, 14, '9/19/2023', Decimal('11.00'))
(9, 15, '9/25/2023', Decimal('46.80'))
(4, 16, '9/22/2023', Decimal('42.75'))
```

[58]:
```
#point the path to where in your hard drive you have stored the file companies.
↪csv
data = pd.read_csv("companies.csv")
data.head()
```

[58]:

	id	name	address	contact_phone	\
0	1	Goodrx	123 Main St, San Francisco, CA	123-456-7890	
1	2	PHARMASEE	456 Elm St, New York, NY	234-567-8901	
2	3	DRUGXO	789 Maple St, Los Angeles, CA	345-678-9012	
3	4	Pharmachoice	101 Pine St, Chicago, IL	456-789-0123	
4	5	Castox	234 Oak St, Houston, TX	567-890-1234	

	contact_name

```
0       Holly Jolly
1       Faker Maker
2       Silly Putty
3      Connie Honey
4  Laxmi Kant Sheth
```

[59]:
```
data.dtypes
```

[59]:
```
id                int64
name             object
address          object
contact_phone    object
contact_name     object
dtype: object
```

[60]:
```
for i,row in data.iterrows():
            sql = "INSERT INTO companies VALUES (%s,%s,%s,%s,%s)"
            mycursor.execute(sql, tuple(row))
            print("Record inserted")
            # the connection is not autocommitted by default, so we
            # must commit to save our changes
            mydb.commit()
```

```
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
```

[61]:
```
query = "SELECT * FROM companies"
mycursor.execute(query)
```

[62]:
```
for x in mycursor:
    print(x)
```

```
(1, 'Goodrx', '123 Main St, San Francisco, CA', '123-456-7890', 'Holly Jolly')
(2, 'PHARMASEE', '456 Elm St, New York, NY', '234-567-8901', 'Faker Maker')
(3, 'DRUGXO', '789 Maple St, Los Angeles, CA', '345-678-9012', 'Silly Putty')
(4, 'Pharmachoice', '101 Pine St, Chicago, IL', '456-789-0123', 'Connie Honey')
(5, 'Castox', '234 Oak St, Houston, TX', '567-890-1234', 'Laxmi Kant Sheth')
(6, 'Doktera', '567 Cedar St, Philadelphia, PA', '678-901-2345', 'I.P. Green')
(7, 'Lipdrugz', '890 Birch St, Phoenix, AZ', '789-012-3456', 'Boris Kotchakoff')
(8, 'Nurfarma', '123 Fir St, San Antonio, TX', '890-123-4567', 'Wu Liu')
```

```
(9, 'Munimed', '456 Redwood St, San Diego, CA', '901-234-5678', 'Kim Park')
(10, 'Arkmed', '789 Sequoia St, Dallas, TX', '012-345-6789', 'James Bond')
```

[63]:
```
#point the path to where in your hard drive you have stored the file contracts.csv
data = pd.read_csv("contracts.csv")
data.head()
```

[63]:

	Contract_Id	drug	dosage	pharmacy_id	pharm_company_id	quantity	\
0	1	Cleotrana	50	5	10	40	
1	2	Primalovir	500	5	10	20	
2	3	Kanulin	1000	4	9	20	
3	4	Olanzanafine	50	6	3	80	
4	5	Avafoxin	5	15	5	30	

	date	price
0	10/3/2023	100.0
1	9/26/2023	40.5
2	9/20/2023	10.5
3	9/24/2023	150.0
4	9/29/2023	18.0

[64]:
```
data.dtypes
```

[64]:
```
Contract_Id           int64
drug                 object
dosage                int64
pharmacy_id           int64
pharm_company_id      int64
quantity              int64
date                 object
price               float64
dtype: object
```

[65]:
```
for i,row in data.iterrows():
            sql = "INSERT INTO contracts VALUES (%s,%s,%s,%s,%s,%s,%s,%s)"
            mycursor.execute(sql, tuple(row))
            print("Record inserted")
            # the connection is not autocommitted by default, so we
            # must commit to save our changes
            mydb.commit()
```

```
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
```

```
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
```

[66]:
```
query = "SELECT * FROM contracts"
mycursor.execute(query)
```

[67]:
```
for x in mycursor:
    print(x)
```

```
(1, 'Cleotrana', 50, 5, 10, 40, '10/3/2023', 100)
(2, 'Primalovir', 500, 5, 10, 20, '9/26/2023', 41)
(3, 'Kanulin', 1000, 4, 9, 20, '9/20/2023', 11)
(4, 'Olanzanafine', 50, 6, 3, 80, '9/24/2023', 150)
(5, 'Avafoxin', 5, 15, 5, 30, '9/29/2023', 18)
(6, 'Quixiposide', 25, 5, 1, 30, '10/2/2023', 12)
(7, 'Glucozepam Amcipentin', 20, 10, 2, 80, '9/27/2023', 145)
(8, 'Divisporine Acetaclotide', 15, 12, 9, 30, '10/2/2023', 42)
(9, 'Abnazole Toleluble', 30, 14, 6, 40, '9/21/2023', 45)
(10, 'Dantopex Quixilinum', 100, 14, 8, 20, '9/26/2023', 20)
(11, 'Olanzanafine', 50, 1, 3, 90, '9/30/2023', 160)
(12, 'Olanzanafine', 75, 2, 3, 40, '9/23/2023', 60)
(13, 'Olanzanafine', 75, 3, 3, 20, '9/23/2023', 35)
(14, 'Glucozepam Amcipentin', 40, 8, 2, 20, '10/2/2023', 14)
(15, 'Divisporine Acetaclotide', 10, 9, 9, 40, '9/28/2023', 68)
(16, 'Dantopex Quixilinum', 100, 13, 8, 50, '9/30/2023', 50)
(16, 'Dantopex Quixilinum', 100, 13, 8, 50, '9/30/2023', 50)
```

1.2 Stored Procedure and Trigger

[68]:
```
mycursor.execute("DROP PROCEDURE IF EXISTS physicianinfo")

query_procedure = '''
CREATE PROCEDURE physicianinfo (IN id VARCHAR(128))
BEGIN
        SELECT primary_specialty,experience_years
    FROM physicians
    WHERE physicians.ssn = id;
END
```

```
'''
mycursor.execute(query_procedure)

# Test the procedure
mycursor.callproc("physicianinfo",('614-57-6885',))

for result in mycursor.stored_results():
    for row in result.fetchall():
        print(row)
```

```
('Cardiology', 4)
```

Trigger

[69]:
```
mycursor.execute("DROP TRIGGER IF EXISTS alert_addition")

query_trigger = '''

CREATE TRIGGER alert_addition
AFTER INSERT ON prescriptions
FOR EACH ROW
BEGIN
    DECLARE earlier_drug VARCHAR(128);
    DECLARE interaction_exists INT;

    -- Find an earlier prescribed drug that interacts with the new drug
    SELECT p.drug_name INTO earlier_drug
    FROM prescriptions p
    JOIN adverse_interactions ai ON (p.drug_name = ai.drug_name AND NEW.
 ↪drug_name = ai.drug_name_2)
                                 OR (p.drug_name = ai.drug_name_2 AND NEW.
 ↪drug_name = ai.drug_name)
    WHERE p.patient_id = NEW.patient_id
      AND p.date <= NEW.date
    LIMIT 1;

    -- Check if an interaction exists
    SET interaction_exists = (earlier_drug IS NOT NULL);

    -- If an interaction exists, insert an alert
    IF interaction_exists THEN
        INSERT INTO alerts (patient_id, physician_id, alert_date, drug1, drug2)
        VALUES (
            NEW.patient_id,
            NEW.physician_id,
            NEW.date,
            earlier_drug,
            NEW.drug_name
```

```
        );
    END IF;
END
;
'''
mycursor.execute(query_trigger)

# # Test the trigger
# for x in mycursor:
#     print(x)
# results = mycursor.fetchall()
```

[70]:
```
# Test the trigger
mycursor.execute("DELETE FROM alerts")
mycursor.execute("DELETE FROM pharmacy_fills")
mycursor.execute("DELETE FROM prescriptions")

# Repopulate the tables
#point the path to where in your hard drive you have stored the file␣
 ↪prescriptions.csv
data = pd.read_csv("prescriptions.csv")
data.head()

for i,row in data.iterrows():
            sql = "INSERT INTO prescriptions VALUES (%s,%s,%s,%s,%s,%s)"
            mycursor.execute(sql, tuple(row))
            # the connection is not autocommitted by default, so we
            # must commit to save our changes
            mydb.commit()

#point the path to where in your hard drive you have stored the file␣
 ↪pharmacy_fills.csv
data_pf = pd.read_csv("pharmacy_fills.csv")
data_pf.head()

for i,row in data_pf.iterrows():
            sql = "INSERT INTO pharmacy_fills VALUES (%s,%s,%s,%s)"
            mycursor.execute(sql, tuple(row))
            # print("Record inserted")
            # the connection is not autocommitted by default, so we
            # must commit to save our changes
            mydb.commit()

# Trigger Execution

query = "SELECT * FROM alerts"
mycursor.execute(query)
```

```
for x in mycursor:
        print(x)
```

```
('501-47-2038', '156-28-1945', '9/22/2023', 'Avafoxin', 'Kanulin')
('501-47-2038', '156-28-1945', '9/22/2023', 'Cleotrana', 'Avafoxin')
('303-13-5928', '571-13-9020', '5/24/2023', 'Olanzanafine', 'Primalovir')
('303-13-5928', '571-13-9020', '5/24/2023', 'Primalovir', 'Abnazole Toleluble')
('303-13-5928', '571-13-9020', '6/22/2023', 'Olanzanafine', 'Glucozepam
Amcipentin')
('478-34-0781', '614-57-6885', '9/17/2023', 'Avafoxin', 'Quixiposide')
```

1.3 Queries

Question 1: Find the physicians (ssn) who have most prescribed drugs which caused alerts (due to possible adverse interaction with a previously prescribed drug, not necessarily by the same physician).

[71]:
```
query = '''
WITH alert_causing_prescriptions AS (
    SELECT DISTINCT
        p2.physician_id,
        p2.id AS prescription_id
    FROM
        prescriptions p1
    JOIN
        prescriptions p2 ON p1.patient_id = p2.patient_id
    JOIN
        adverse_interactions ai
        ON (ai.drug_name = p1.drug_name AND ai.drug_name_2 = p2.drug_name)
        OR (ai.drug_name_2 = p1.drug_name AND ai.drug_name = p2.drug_name)
    WHERE
        p1.date < p2.date
        AND p1.id <> p2.id
),
physician_alert_counts AS (
    SELECT
        physician_id,
        COUNT(DISTINCT prescription_id) AS alert_count
    FROM
        alert_causing_prescriptions
    GROUP BY
        physician_id
)
SELECT
    p.ssn,
    pac.alert_count
FROM
    physicians p
```

```
JOIN
    physician_alert_counts pac ON p.ssn = pac.physician_id
WHERE
    pac.alert_count = (
        SELECT MAX(alert_count)
        FROM physician_alert_counts
    )
ORDER BY
    p.ssn;
        '''
mycursor.execute(query)
```

[72]:
```
for x in mycursor:
    print(x)
```

```
('156-28-1945', 2)
```

Question 2: Find the physicians (ssn) who have prescribed two drugs to the same patient which have adverse interactions.

[73]:
```
query = '''
SELECT DISTINCT
   (physician_id)
FROM
   alerts;
        '''
mycursor.execute(query)
```

[74]:
```
for x in mycursor:
    print(x)
```

```
('156-28-1945',)
('571-13-9020',)
('614-57-6885',)
```

Question 3: Find the physicians who have prescribed most drugs supplied by company DRUGXO.

[75]:
```
query = '''
SELECT
    physicians.ssn,
    COUNT(DISTINCT prescriptions.drug_name) AS prescribed_drugs_count
FROM
    physicians
        INNER JOIN
    prescriptions ON prescriptions.physician_id = physicians.ssn
        INNER JOIN
    drugs ON drugs.name = prescriptions.drug_name
        INNER JOIN
    contracts ON contracts.drug_name = drugs.name
```

```
        INNER JOIN
    companies ON companies.id = contracts.company_id
WHERE
    companies.name = 'DRUGXO'
GROUP BY physicians.ssn
ORDER BY prescribed_drugs_count DESC
LIMIT 1;
         '''
mycursor.execute(query)
```

[76]:
```
for x in mycursor:
    print(x)
```

```
('571-13-9020', 1)
```

Question 4: For each drug supplied by company PHARMASEE display the price (per unit of quantity) charged by that company for that drug along with the average price charged for that drug (by companies, not pharmacies). Note: As it happens in the data we supplied each drug is supplied by only one company, but your query should not be based on that.

[77]:
```
query = '''
WITH avg_prices AS (
    SELECT
        drug_name,
        AVG(price) AS avg_price
    FROM
        contracts
    GROUP BY
        drug_name
)
SELECT
    c.drug_name,
    c.price AS pharmasee_price,
    ap.avg_price
FROM
    contracts c
JOIN
    companies co ON c.company_id = co.id
JOIN
    avg_prices ap ON c.drug_name = ap.drug_name
WHERE
    co.name = 'PHARMASEE';
         '''
mycursor.execute(query)
```

[78]:
```
for x in mycursor:
    print(x)
```

```
('Glucozepam Amcipentin', 145, Decimal('79.5000'))
```

```
('Glucozepam Amcipentin', 14, Decimal('79.5000'))

('Glucozepam Amcipentin', 14, Decimal('79.5000'))
```

Question 5: For each drug and for each pharmacy, find the percentage of the markup (per unit of quantity) for that drug by that pharmacy.

[79]:
```
query = '''
WITH drugpurchaseprice AS (
    SELECT d.name AS drug_name, p.name AS pharmacy, c.price AS drug_price, c.
 ↪quantity AS contract_quantity
    FROM drugs d
    INNER JOIN contracts c ON c.drug_name = d.name
    INNER JOIN pharmacies p ON c.pharmacy_id = p.id
),
pharmacyfillcost AS (
        SELECT d.name AS drug_name, p.name AS pharmacy, pf.cost AS drug_cost,pr.
 ↪quantity AS fill_quantity
    FROM drugs d
        INNER JOIN contracts c
        ON c.drug_name = d.name
    INNER JOIN pharmacies p
        ON c.pharmacy_id = p.id
        INNER JOIN pharmacy_fills pf
        ON pf.pharmacy_id = p.id
    INNER JOIN prescriptions pr
    ON pr.id = pf.prescription_id
)
SELECT
    dpp.drug_name,
    dpp.pharmacy,
    ((pfc.drug_cost / pfc.fill_quantity) - (dpp.drug_price / dpp.
 ↪contract_quantity)) / (dpp.drug_price / dpp.contract_quantity) * 100 AS␣
 ↪percentage_markup
FROM drugpurchaseprice dpp
INNER JOIN pharmacyfillcost pfc
    ON dpp.drug_name = pfc.drug_name AND dpp.pharmacy = pfc.pharmacy;
    '''
mycursor.execute(query)
```

[80]:
```
for x in mycursor:
    print(x)
```

```
('Cleotrana', ' Emerald City Pharmacy', Decimal('-36.9333333600'))
('Cleotrana', ' Emerald City Pharmacy', Decimal('233.2000000000'))
('Primalovir', ' Emerald City Pharmacy', Decimal('-23.0894309268'))
('Primalovir', ' Emerald City Pharmacy', Decimal('306.3414634146'))
('Kanulin', ' Mile High Meds', Decimal('871.5909090909'))
```

```
('Olanzanafine', ' Golden Gate Drugs', Decimal('-43.7333333333'))
('Quixiposide', ' Emerald City Pharmacy', Decimal('294.1666665000'))
('Quixiposide', ' Emerald City Pharmacy', Decimal('1982.5000000000'))
('Glucozepam Amcipentin', ' Desert Bloom Pharmacy', Decimal('4.8275862069'))
('Divisporine Acetaclotide', ' Capital Meds', Decimal('6471.4285714286'))
('Abnazole Toleluble', ' Beantown Drugs', Decimal('280.8888888889'))
('Dantopex Quixilinum', ' Beantown Drugs', Decimal('328.5000000000'))
('Olanzanafine', ' Springfield Pharmacy', Decimal('70.2406250745'))
('Olanzanafine', ' Springfield Pharmacy', Decimal('1.2500000443'))
('Olanzanafine', ' Peachtree Meds', Decimal('-51.5000000000'))
('Olanzanafine', ' Lone Star Drugs', Decimal('97.6190476000'))
('Glucozepam Amcipentin', ' Liberty Meds', Decimal('-26.1428571429'))
('Divisporine Acetaclotide', ' Lakeside Drugs', Decimal('83.5294117647'))
```

Question 6: For each drug, find the average time between when a patient was prescribed a drug and when the prescription was filled at a pharmacy. (You will need to extract the components of a date to compute this. mySQL provides functions for doing this and the official documentation here can help provide the appropriate functions: https://dev.mysql.com/doc/refman/8.4/en/date-and-time-functions.html Links to an external site.).

[81]:
```
query = '''
SELECT
    p.drug_name,
    AVG(TIMESTAMPDIFF(HOUR,
        STR_TO_DATE(p.date, '%m/%d/%Y'),
        STR_TO_DATE(pf.date, '%m/%d/%Y'))) AS avg_hours_to_fill
FROM
    prescriptions p
        JOIN
    pharmacy_fills pf ON p.id = pf.prescription_id
GROUP BY p.drug_name;
    '''
mycursor.execute(query)
```

[82]:
```
for x in mycursor:
    print(x)
```

```
('Avafoxin', Decimal('42.0000'))
('Cleotrana', Decimal('32.0000'))
('Primalovir', Decimal('-324.0000'))
('Glucozepam Amcipentin', Decimal('36.0000'))
('Olanzanafine', Decimal('48.0000'))
('Abnazole Toleluble', Decimal('96.0000'))
('Dantopex Quixilinum', Decimal('48.0000'))
('Quixiposide', Decimal('48.0000'))
('Kanulin', Decimal('0.0000'))
```

Question 7: For each pharmacy, find all the drugs that were prescribed to a patient and never filled at that pharmacy.

[83]:
```
query = '''
(SELECT p.id AS pharmacy_id, p.name AS pharmacy_name, pr.drug_name
 FROM pharmacies p
 CROSS JOIN prescriptions pr)
EXCEPT
(SELECT pf.pharmacy_id, p.name AS pharmacy_name, pr.drug_name
 FROM pharmacy_fills pf
 JOIN pharmacies p ON pf.pharmacy_id = p.id
 JOIN prescriptions pr ON pf.prescription_id = pr.id)
ORDER BY pharmacy_id, drug_name;
    '''
mycursor.execute(query)
```

[84]:
```
for x in mycursor:
    print(x)
```

```
(1, ' Springfield Pharmacy', 'Abnazole Toleluble')
(1, ' Springfield Pharmacy', 'Avafoxin')
(1, ' Springfield Pharmacy', 'Dantopex Quixilinum')
(1, ' Springfield Pharmacy', 'Glucozepam Amcipentin')
(1, ' Springfield Pharmacy', 'Kanulin')
(1, ' Springfield Pharmacy', 'Olanzanafine')
(1, ' Springfield Pharmacy', 'Quixiposide')
(2, ' Peachtree Meds', 'Abnazole Toleluble')
(2, ' Peachtree Meds', 'Avafoxin')
(2, ' Peachtree Meds', 'Dantopex Quixilinum')
(2, ' Peachtree Meds', 'Glucozepam Amcipentin')
(2, ' Peachtree Meds', 'Kanulin')
(2, ' Peachtree Meds', 'Olanzanafine')
(2, ' Peachtree Meds', 'Primalovir')
(2, ' Peachtree Meds', 'Quixiposide')
(3, ' Lone Star Drugs', 'Abnazole Toleluble')
(3, ' Lone Star Drugs', 'Avafoxin')
(3, ' Lone Star Drugs', 'Cleotrana')
(3, ' Lone Star Drugs', 'Dantopex Quixilinum')
(3, ' Lone Star Drugs', 'Kanulin')
(3, ' Lone Star Drugs', 'Olanzanafine')
(3, ' Lone Star Drugs', 'Primalovir')
(3, ' Lone Star Drugs', 'Quixiposide')
(4, ' Mile High Meds', 'Abnazole Toleluble')
(4, ' Mile High Meds', 'Avafoxin')
(4, ' Mile High Meds', 'Cleotrana')
(4, ' Mile High Meds', 'Dantopex Quixilinum')
(4, ' Mile High Meds', 'Glucozepam Amcipentin')
(4, ' Mile High Meds', 'Olanzanafine')
(4, ' Mile High Meds', 'Primalovir')
(4, ' Mile High Meds', 'Quixiposide')
(5, ' Emerald City Pharmacy', 'Cleotrana')
```

```
(5, ' Emerald City Pharmacy', 'Dantopex Quixilinum')
(5, ' Emerald City Pharmacy', 'Glucozepam Amcipentin')
(5, ' Emerald City Pharmacy', 'Kanulin')
(5, ' Emerald City Pharmacy', 'Olanzanafine')
(5, ' Emerald City Pharmacy', 'Primalovir')
(5, ' Emerald City Pharmacy', 'Quixiposide')
(6, ' Golden Gate Drugs', 'Abnazole Toleluble')
(6, ' Golden Gate Drugs', 'Cleotrana')
(6, ' Golden Gate Drugs', 'Dantopex Quixilinum')
(6, ' Golden Gate Drugs', 'Glucozepam Amcipentin')
(6, ' Golden Gate Drugs', 'Kanulin')
(6, ' Golden Gate Drugs', 'Olanzanafine')
(6, ' Golden Gate Drugs', 'Primalovir')
(6, ' Golden Gate Drugs', 'Quixiposide')
(7, ' Sunshine Pharmacy', 'Abnazole Toleluble')
(7, ' Sunshine Pharmacy', 'Avafoxin')
(7, ' Sunshine Pharmacy', 'Cleotrana')
(7, ' Sunshine Pharmacy', 'Dantopex Quixilinum')
(7, ' Sunshine Pharmacy', 'Glucozepam Amcipentin')
(7, ' Sunshine Pharmacy', 'Kanulin')
(7, ' Sunshine Pharmacy', 'Olanzanafine')
(8, ' Liberty Meds', 'Abnazole Toleluble')
(8, ' Liberty Meds', 'Cleotrana')
(8, ' Liberty Meds', 'Dantopex Quixilinum')
(8, ' Liberty Meds', 'Glucozepam Amcipentin')
(8, ' Liberty Meds', 'Kanulin')
(8, ' Liberty Meds', 'Olanzanafine')
(8, ' Liberty Meds', 'Primalovir')
(8, ' Liberty Meds', 'Quixiposide')
(9, ' Lakeside Drugs', 'Abnazole Toleluble')
(9, ' Lakeside Drugs', 'Cleotrana')
(9, ' Lakeside Drugs', 'Dantopex Quixilinum')
(9, ' Lakeside Drugs', 'Glucozepam Amcipentin')
(9, ' Lakeside Drugs', 'Kanulin')
(9, ' Lakeside Drugs', 'Olanzanafine')
(9, ' Lakeside Drugs', 'Primalovir')
(9, ' Lakeside Drugs', 'Quixiposide')
(10, ' Desert Bloom Pharmacy', 'Abnazole Toleluble')
(10, ' Desert Bloom Pharmacy', 'Avafoxin')
(10, ' Desert Bloom Pharmacy', 'Cleotrana')
(10, ' Desert Bloom Pharmacy', 'Dantopex Quixilinum')
(10, ' Desert Bloom Pharmacy', 'Glucozepam Amcipentin')
(10, ' Desert Bloom Pharmacy', 'Kanulin')
(10, ' Desert Bloom Pharmacy', 'Primalovir')
(10, ' Desert Bloom Pharmacy', 'Quixiposide')
(11, ' Bayside Pharmacy', 'Abnazole Toleluble')
(11, ' Bayside Pharmacy', 'Avafoxin')
(11, ' Bayside Pharmacy', 'Cleotrana')
```

```
(11, ' Bayside Pharmacy', 'Dantopex Quixilinum')
(11, ' Bayside Pharmacy', 'Kanulin')
(11, ' Bayside Pharmacy', 'Olanzanafine')
(11, ' Bayside Pharmacy', 'Primalovir')
(11, ' Bayside Pharmacy', 'Quixiposide')
(12, ' Capital Meds', 'Abnazole Toleluble')
(12, ' Capital Meds', 'Avafoxin')
(12, ' Capital Meds', 'Cleotrana')
(12, ' Capital Meds', 'Glucozepam Amcipentin')
(12, ' Capital Meds', 'Kanulin')
(12, ' Capital Meds', 'Olanzanafine')
(12, ' Capital Meds', 'Primalovir')
(12, ' Capital Meds', 'Quixiposide')
(13, ' Windy City Pharmacy', 'Abnazole Toleluble')
(13, ' Windy City Pharmacy', 'Avafoxin')
(13, ' Windy City Pharmacy', 'Cleotrana')
(13, ' Windy City Pharmacy', 'Dantopex Quixilinum')
(13, ' Windy City Pharmacy', 'Glucozepam Amcipentin')
(13, ' Windy City Pharmacy', 'Kanulin')
(13, ' Windy City Pharmacy', 'Olanzanafine')
(13, ' Windy City Pharmacy', 'Primalovir')
(13, ' Windy City Pharmacy', 'Quixiposide')
(14, ' Beantown Drugs', 'Abnazole Toleluble')
(14, ' Beantown Drugs', 'Avafoxin')
(14, ' Beantown Drugs', 'Dantopex Quixilinum')
(14, ' Beantown Drugs', 'Glucozepam Amcipentin')
(14, ' Beantown Drugs', 'Kanulin')
(14, ' Beantown Drugs', 'Olanzanafine')
(14, ' Beantown Drugs', 'Primalovir')
(14, ' Beantown Drugs', 'Quixiposide')
(15, ' Gateway Meds', 'Abnazole Toleluble')
(15, ' Gateway Meds', 'Avafoxin')
(15, ' Gateway Meds', 'Cleotrana')
(15, ' Gateway Meds', 'Dantopex Quixilinum')
(15, ' Gateway Meds', 'Glucozepam Amcipentin')
(15, ' Gateway Meds', 'Kanulin')
(15, ' Gateway Meds', 'Olanzanafine')
(15, ' Gateway Meds', 'Primalovir')
(15, ' Gateway Meds', 'Quixiposide')
```