In [11]:
```
import pandas as pd
import sqlite3

con = sqlite3.connect("transactions.sqlite")
cur = con.cursor()
```

In [12]:
```
df = pd.read_excel("transaction_data.xlsx")
```

In [13]:
```
# - ALL THE TABLES IN THE DATABASE

cur.executescript('''
DROP TABLE IF EXISTS Customer;
DROP TABLE IF EXISTS Income;
DROP TABLE IF EXISTS ProductFamily;
DROP TABLE IF EXISTS ProductDepartment;
DROP TABLE IF EXISTS ProductCategory;
DROP TABLE IF EXISTS City;
DROP TABLE IF EXISTS State;
DROP TABLE IF EXISTS Country;
DROP TABLE IF EXISTS Transactions;



CREATE TABLE Customer(
ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
CustomerID INTEGER UNIQUE,
Gender CHAR(1),
MaritalStatus CHAR (1),
HomeOwner CHAR (1),
Children INTEGER,
Income INTEGER,
FOREIGN KEY (Income) REFERENCES Income(ID)
);

CREATE TABLE Income(
ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
Income TEXT NOT NULL UNIQUE
);

CREATE TABLE ProductDepartment(
ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
ProductDepartment CHAR,
Income INTEGER,
FOREIGN KEY (Income) REFERENCES Income(ID)
);

CREATE TABLE ProductCategory(
ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
ProductCategory CHAR,
Income INTEGER,
FOREIGN KEY (Income) REFERENCES Income(ID)
);

CREATE TABLE ProductFamily(
ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
ProductFamily CHAR,
Income INTEGER,
FOREIGN KEY (Income) REFERENCES Income(ID)
);

CREATE TABLE City(
ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
City TEXT UNIQUE,
Income INTEGER,
FOREIGN KEY (Income) REFERENCES Income(ID)
);

CREATE TABLE State(
ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
State TEXT UNIQUE,
Income INTEGER,
FOREIGN KEY (Income) REFERENCES Income(ID)
);

CREATE TABLE Country(
ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
Country TEXT UNIQUE,
Income INTEGER,
FOREIGN KEY (Income) REFERENCES Income(ID)
);

CREATE TABLE Transactions(
ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
TransactionID INTEGER,
Date TEXT,
ItemsSold INTEGER,
Amount FLOAT
);

''')
con.commit()
```

In [14]:

```
# - THE LOCATION TABLES (UNIQUE VALUES ONLY) - city, state, country

city = df["City"]
city = city.unique()
for i in city:
    cur.execute('''INSERT OR IGNORE INTO City (City) VALUES(?)''',(i,))
state = df["State or Province"]
state = state.unique()
for i in state:
    cur.execute('''INSERT OR IGNORE INTO State (State) VALUES(?)''',(i,))
country = df["Country"]
country = country.unique()
for i in country:
    cur.execute('''INSERT OR IGNORE INTO Country (Country) VALUES(?)''',(i,))

con.commit()
```

In [15]:

```
# - THE PRODUCT TABLES (UNIQUE VALUES ONLY) - department, family, category

pdept = df["Product Department"]
pdept = pdept.unique()
for i in pdept:
    cur.execute('''INSERT OR REPLACE INTO ProductDepartment (ProductDepartment) VALUES(?)''',(i,))
pfam = df["Product Family"]
pfam = pfam.unique()
for i in pfam:
    cur.execute('''INSERT OR REPLACE INTO ProductFamily (ProductFamily) VALUES(?)''',(i,))
pcat = df["Product Category"]
pcat = pcat.unique()
for i in pcat:
    cur.execute('''INSERT OR REPLACE INTO ProductCategory (ProductCategory) VALUES(?)''',(i,))

con.commit()
```

In [16]:

```
# - THE INCOME TABLE (UNIQUE VALUES ONLY) - the main foreign key
```

```
Income = df["Annual Income"]
Income = Income.unique()
for i in Income:
    cur.execute('''INSERT OR IGNORE INTO Income (Income) VALUES(?)''',(i,)) #When a table does not have foreign
con.commit()
```

In [17]:

```
# - THE CUSTOMER TABLE (ALL VALUES) with the foreign key

customer = df[['Customer ID','Gender','Marital Status','Homeowner', 'Children','Annual Income']]
i = 0
while i < len(customer):
    cid = customer.iloc[i].values[:1] #THIS IS A DATA FRAME
    cid = cid[0].tolist()             #This is an array that has to be changed from individual value
    g = customer.iloc[i].values[1:2]
    g = g[0]
    ms = customer.iloc[i].values[2:3]
    ms = ms[0]
    ho = customer.iloc[i].values[3:4]
    ho = ho[0]
    c = customer.iloc[i].values[4:5]
    c = int(c[0])                   #The children data is a string in the dataset so you need to change it to a
    ai = customer.iloc[i].values[5:6]
    ai = ai[0]
    cur.execute('SELECT ID FROM Income WHERE Income = ?', (ai, )) #This is different because the Income data is
    #Inc = cur.fetchone()[0] #This is here to avoid continuous/repeated input of income data
    cur.execute('''INSERT OR REPLACE INTO Customer(CustomerID, Gender, MaritalStatus, Homeowner, Children, Inco
                VALUES (?,?,?,?,?,?)''', (cid,g,ms,ho,c,ai, )) #When a table has foreign keys, use REPLACE
    i+=1
con.commit()
```

In [18]:

```
# - THE TRANSACTION TABLE (ALL VALUES)

transaction = df[['Transaction','Purchase Date','Units Sold','Revenue']]
i = 0
while i < len(customer):
    tid = transaction.iloc[i].values[:1]
    tid = int(tid[0])
    d = transaction.iloc[i].values[1:2]
    d = d[0]
    u = transaction.iloc[i].values[2:3]
    u = int(u[0])
    r = transaction.iloc[i].values[3:4]
    r = r[0]
    cur.execute('''INSERT OR REPLACE INTO Transactions(TransactionID, Date, ItemsSold, Amount)VALUES (?,?,?,?)'
    i+=1
con.commit()
```

In [10]:

```
con.close()

#                                              - THE END -
```

In [ ]:

```
# - OTHER QUESTIONS/EXCERCISES
```

In [19]:

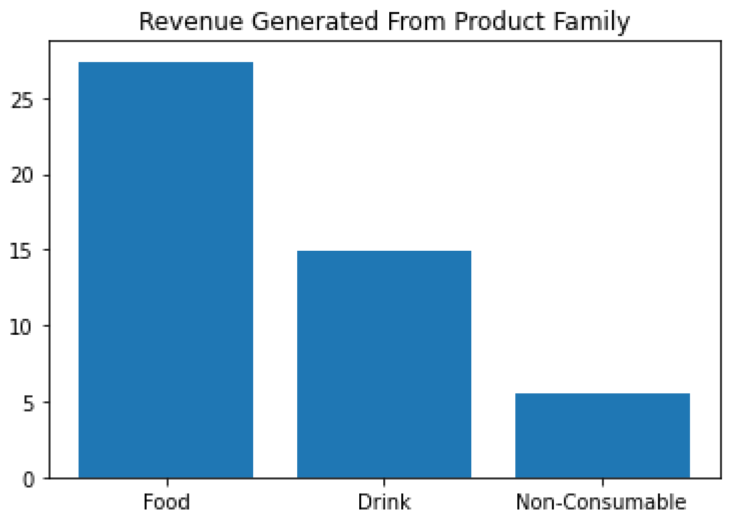
```
import pandas as pd
import sqlite3
con = sqlite3.connect("transactions.sqlite")
def x(q):
    return pd.read_sql_query(q, con)
x('SELECT * FROM Customer limit 3')
```

Out[19]:

	ID	CustomerID	Gender	MaritalStatus	HomeOwner	Children	Income
0	6	6696	F	M	Y	3	$10K-30K
1	9	1293	M	M	Y	3	$10K-30K
2	13	2741	M	S	N	3	$70K-90K

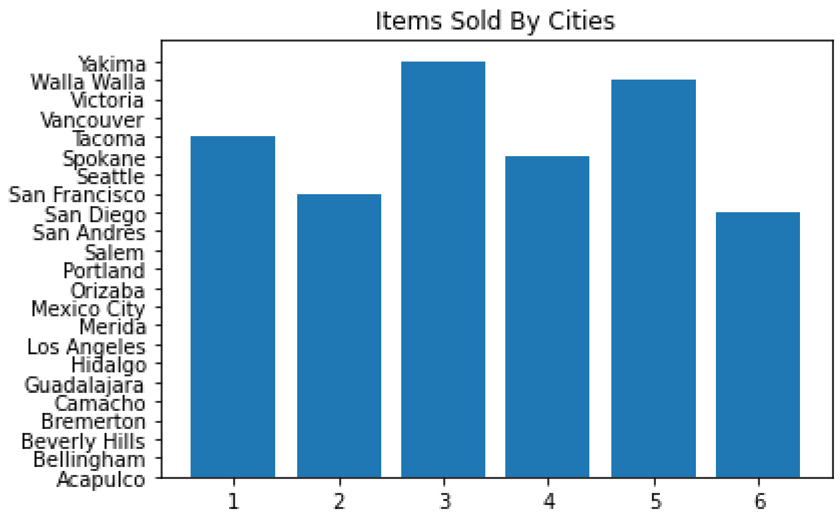
In [20]:

```
import seaborn as sns
import matplotlib.pyplot as plt

x = """SELECT ProductFamily, Amount FROM Transactions natural join ProductFamily"""

data = pd.read_sql(x, con)
plt.bar(data.ProductFamily, data.Amount)
plt.title("Revenue Generated From Product Family")
plt.show()
```



In [21]:

```
x = """SELECT City,ItemsSold FROM Transactions natural join City"""

data = pd.read_sql(x, con)
plt.bar(data.ItemsSold, data.City)
plt.title("Items Sold By Cities")
plt.show()
```


Items Sold By Cities
Yakima
Walla Walla
Victoria
Vancouver
Tacoma
Spokane
Seattle
San Francisco
San Diego
San Andres
Salem
Portland
Orizaba
Mexico City
Merida
Los Angeles
Hidalgo
Guadalajara
Camacho
Bremerton
Beverly Hills
Bellingham
Acapulco