Databases

1.

a.

```
import sqlite3 as dbapi

con = dbapi.connect('census.db')
```

b.

```
cur = con.cursor()
cur.execute('''CREATE TABLE Density(Province TEXT,
                                    Population INTEGER, Area REAL)''')
con.commit()
```

c.

```
table = [
    ('Newfoundland and Labrador', 512930, 370501.69),
    ('Prince Edward Island', 135294, 5684.39),
    ('Nova Scotia', 908007, 52917.43),
    ('New Brunswick', 729498, 71355.67),
    ('Quebec', 7237479, 1357743.08),
    ('Ontario', 11410046, 907655.59),
    ('Manitoba', 1119583, 551937.87),
    ('Saskatchewan', 978933, 586561.35),
    ('Alberta', 2974807, 639987.12),
    ('British Columbia', 3907738, 926492.48),
    ('Yukon Territory', 28674, 474706.97),
    ('Northwest Territories', 37360, 1141108.37),
    ('Nunavut', 26745, 1925460.18),
]

for row in table:
    cur.execute('INSERT INTO Density VALUES (?, ?, ?)', row)
con.commit()
```

d.

```
cur.execute('SELECT * FROM Density')
for row in cur.fetchall():
    print(row)
```

e.

```
cur.execute('SELECT Population FROM Density')
for row in cur.fetchall():
    print(row)
```

f.

```
cur.execute('''SELECT Province FROM Density
               WHERE Population < 1000000''')
for row in cur.fetchall():
    print(row)
```

g.

```
cur.execute('''SELECT Province FROM Density
               WHERE Population < 1000000
               OR Population > 5000000''')
for row in cur.fetchall():
    print(row)
```

h.

```
cur.execute('''SELECT Province FROM Density
               WHERE NOT(Population < 1000000
               OR Population > 5000000)''')
for row in cur.fetchall():
    print(row)
```

i.

```
cur.execute('''SELECT Population FROM Density
               WHERE Area > 200000''')
for row in cur.fetchall():
    print(row)
```

j.

```
cur.execute('SELECT Province, Population / Area FROM Density')
for row in cur.fetchall():
    print(row)
```

2.

```
import sqlite3 as dbapi

con = dbapi.connect('census.db')
cur = con.cursor()

cur.execute('''CREATE TABLE Capitals(Province TEXT,
                                     Capital TEXT, Population INTEGER)''')
con.commit()

table = [
    ('Newfoundland and Labrador', "St. John's", 172918),
    ('Prince Edward Island', 'Charlottetown', 58358),
    ('Nova Scotia', 'Halifax', 359183),
    ('New Brunswick', 'Fredericton', 81346),
```

```
    ('Quebec', 'Qeubec City', 682757),
    ('Ontario', 'Toronto', 4682897),
    ('Manitoba', 'Winnipeg', 671274),
    ('Saskatchewan', 'Regina', 192800),
    ('Alberta', 'Edmonton', 937845),
    ('British Columbia', 'Victoria', 311902),
    ('Yukon Territory', 'Whitehorse', 21405),
    ('Northwest Territories', 'Yellowknife', 16541),
    ('Nunavut', 'Iqaluit', 5236),
]

for row in table:
    cur.execute('INSERT INTO Capitals VALUES (?, ?, ?)', row)
con.commit()
```

a.

```
cur.execute('SELECT * FROM Capitals')
for row in cur.fetchall():
    print(row)
```

b.

```
cur.execute('''SELECT Density.Population, Capitals.Population
               FROM Capitals INNER JOIN Density
               WHERE Capitals.Province = Density.Province''')
for row in cur.fetchall():
    print(row)
```

c.

```
cur.execute('''SELECT Density.Area
               FROM Capitals INNER JOIN Density
               WHERE Capitals.Province = Density.Province
               AND Capitals.Population > 100000''')
for row in cur.fetchall():
    print(row)
```

d.

Note: This query doesn't return any results.

```
cur.execute('''SELECT Density.Province
               FROM Capitals INNER JOIN Density
               WHERE Capitals.Province = Density.Province
               AND Density.Population / Density.Area < 2
               AND Capitals.Population > 500000''')
for row in cur.fetchall():
    print(row)
```

e.

```
cur.execute('SELECT SUM(Area) FROM Density')
print(cur.fetchone())
```

f.

```
cur.execute('SELECT AVG(Population) FROM Capitals')
print(cur.fetchone())
```

g.

```
cur.execute('SELECT MIN(Population) FROM Capitals')
print(cur.fetchone())
```

h.

```
cur.execute('SELECT MAX(Population) FROM Density')
print(cur.fetchone())
```

i.

```
cur.execute('''SELECT A.Province, B.Province
               FROM Density A INNER JOIN Density B
               WHERE A.Province < B.Province
               AND ABS(A.Population / A.Area - B.Population / B.Area) <
0.5''')
for row in cur.fetchall():
    print(row)
```

3. In Python, division by zero results in an error being thrown:

```
>>> value = 1
>>> 1/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero

>>> 1/0 and value > 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero

>>> value > 0 and 1/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
```

In SQL, division by zero results in a special value that represents infinity (if the numerator is not zero) or “not a number” when the numerator is zero. However, even

though this special value is non-zero it does not evaluate to `True`. So in each select statement, no rows are returned.