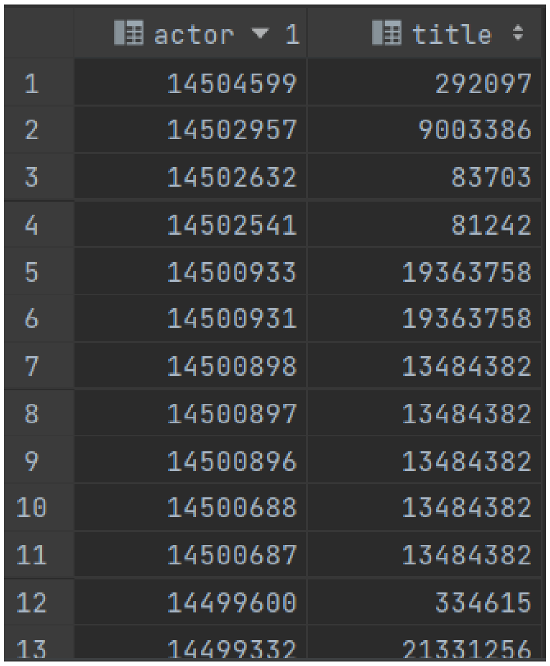
**CSCI.620.01/03** **Homework7** **Suraj Sureshkumar**

## Task 1:

Creating the table popular_movie_actors

```
CREATE TABLE Popular_Movie_Actors AS
SELECT ta.*
FROM hw_schema."Title_Actor" ta
     JOIN (SELECT id, title, runtime
          FROM hw_schema."Title" t
          WHERE type = 'movie'
            AND "avgRating" > 5) m ON ta.title = m.id;
```

**Output:**



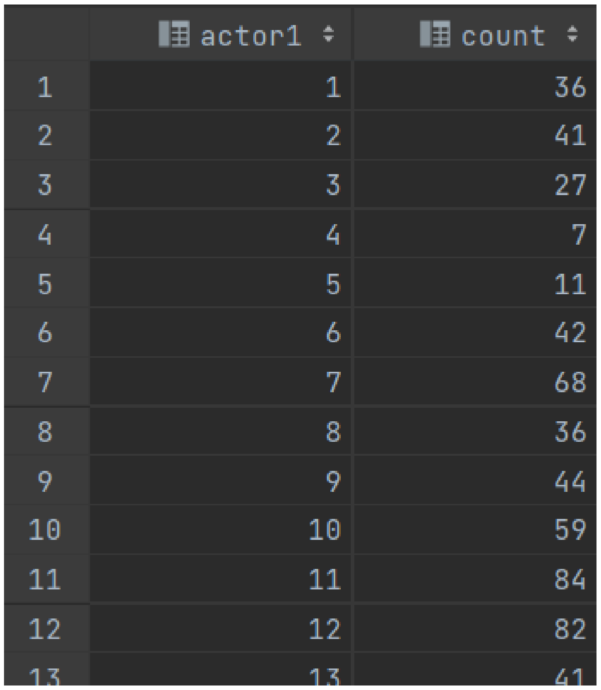
| | actor | title |
|---|---|---|
| 1 | 14504599 | 292097 |
| 2 | 14502957 | 9003386 |
| 3 | 14502632 | 83703 |
| 4 | 14502541 | 81242 |
| 5 | 14500933 | 19363758 |
| 6 | 14500931 | 19363758 |
| 7 | 14500898 | 13484382 |
| 8 | 14500897 | 13484382 |
| 9 | 14500896 | 13484382 |
| 10 | 14500688 | 13484382 |
| 11 | 14500687 | 13484382 |
| 12 | 14499600 | 334615 |
| 13 | 14499332 | 21331256 |

## Task 2:

Creating a table L1 using the popular_movie_actors and making sure it is equal to and above the minimum support.

```
CREATE TABLE L1 AS
SELECT pma.actor AS actor1, COUNT(*) AS count
FROM public.popular_movie_actors pma
GROUP BY pma.actor
HAVING COUNT(*) >= 5;
```

**Output**



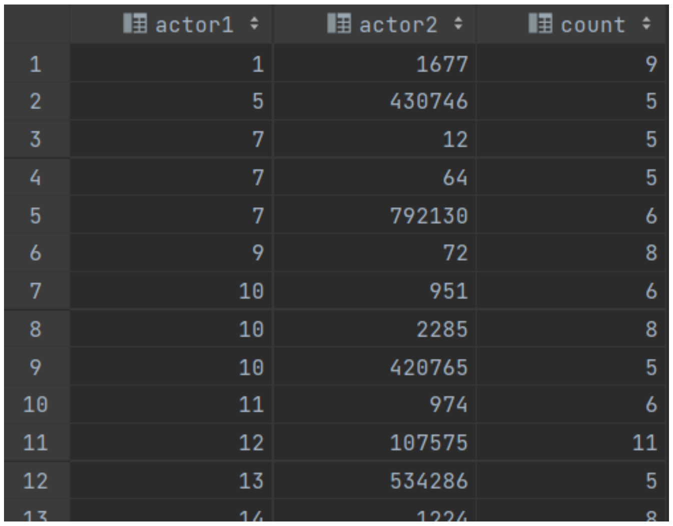
| | actor1 | count |
|---|---|---|
| 1 | 1 | 36 |
| 2 | 2 | 41 |
| 3 | 3 | 27 |
| 4 | 4 | 7 |
| 5 | 5 | 11 |
| 6 | 6 | 42 |
| 7 | 7 | 68 |
| 8 | 8 | 36 |
| 9 | 9 | 44 |
| 10 | 10 | 59 |
| 11 | 11 | 84 |
| 12 | 12 | 82 |
| 13 | 13 | 41 |

## Task 3:

Creating a table L2 by using L1 and joining L1 on itself. And popular_movie_actors on itself. Necessary joins are performed in place and making sure it is equal to and above the minimum support. The less than is added so that actor1 is paired up with the other actors which are actor2 and actor3 when these two are there.

```
CREATE TABLE L2 AS
SELECT l1.actor1, l2.actor1 as actor2, COUNT(*) AS count
FROM public.l1 l1,
     public.l1 l2,
     public.popular_movie_actors pma,
     public.popular_movie_actors pma1
WHERE l1.actor1 = pma.actor
  and l2.actor1 = pma1.actor
  and l1.actor1 < l2.actor1
  and pma.title = pma1.title
GROUP BY l1.actor1, l2.actor1
HAVING COUNT(*) >= 5;
```

**Output**



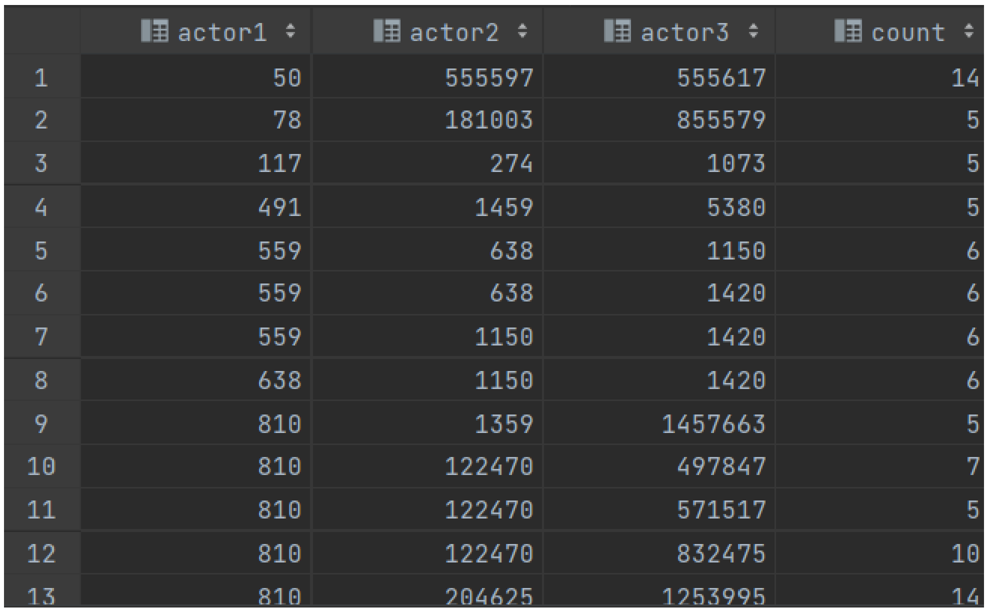
| | actor1 | actor2 | count |
|---|---|---|---|
| 1 | 1 | 1677 | 9 |
| 2 | 5 | 430746 | 5 |
| 3 | 7 | 12 | 5 |
| 4 | 7 | 64 | 5 |
| 5 | 7 | 792130 | 6 |
| 6 | 9 | 72 | 8 |
| 7 | 10 | 951 | 6 |
| 8 | 10 | 2285 | 8 |
| 9 | 10 | 420765 | 5 |
| 10 | 11 | 974 | 6 |
| 11 | 12 | 107575 | 11 |
| 12 | 13 | 534286 | 5 |
| 13 | 14 | 1224 | 8 |

## Task 4:

Creating a table for L3 by joining the popular_movie_actors on itself for three times and the l2 on itself. The less than is added so that actor2 is paired up with the other actors which is actor3 and actor3 cannot pair up with anything else as there is nothing after that.

```
CREATE TABLE L3 AS
SELECT a.actor1, a.actor2, b.actor2 as actor3, COUNT(*) AS count
FROM public.l2 a,
     public.l2 b,
     public.popular_movie_actors pma1,
     public.popular_movie_actors pma2,
     public.popular_movie_actors pma3
WHERE a.actor1 = pma1.actor
  and a.actor2 = pma2.actor
  and b.actor2 = pma3.actor
  and a.actor1 = b.actor1
  and a.actor2 < b.actor2
  and pma1.title = pma2.title
  and pma2.title = pma3.title
GROUP BY a.actor1, a.actor2, b.actor2
HAVING COUNT(*) >= 5;
```

### Output



| | actor1 | actor2 | actor3 | count |
|---|---|---|---|---|
| 1 | 50 | 555597 | 555617 | 14 |
| 2 | 78 | 181003 | 855579 | 5 |
| 3 | 117 | 274 | 1073 | 5 |
| 4 | 491 | 1459 | 5380 | 5 |
| 5 | 559 | 638 | 1150 | 6 |
| 6 | 559 | 638 | 1420 | 6 |
| 7 | 559 | 1150 | 1420 | 6 |
| 8 | 638 | 1150 | 1420 | 6 |
| 9 | 810 | 1359 | 1457663 | 5 |
| 10 | 810 | 122470 | 497847 | 7 |
| 11 | 810 | 122470 | 571517 | 5 |
| 12 | 810 | 122470 | 832475 | 10 |
| 13 | 810 | 204625 | 1253995 | 14 |

## Task 5:

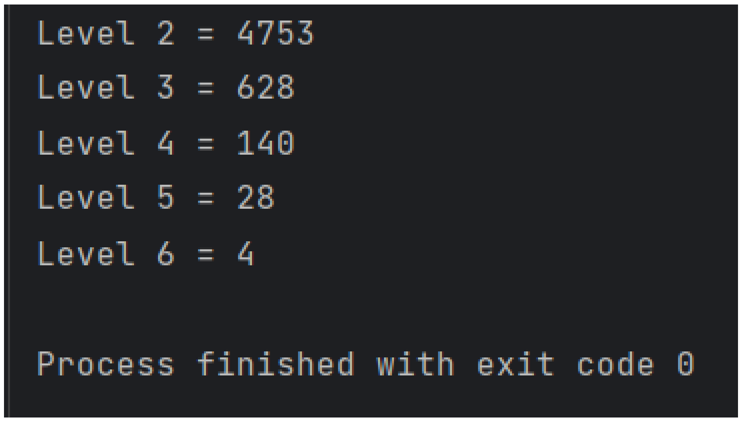
Program which makes a connection to sql database. Min_support and current level is declared.

```
import psycopg2

conn = psycopg2.connect(
   host="localhost",
   port=5432,
   dbname="hw2",
   user="postgres",
   password="surajsuri1456@#$"
)
cur = conn.cursor()

min_support = 5

current_level = 2

while True:
   query1 = 'CREATE TABLE L{} AS SELECT '.format(current_level)

   for level in range(1, current_level):
      query1 += 'a.actor{},'.format(level)

   query1 += 'b.actor{} as actor{}, COUNT(*) AS count FROM '.format(current_level - 1,
current_level)
   for alias in ['a', 'b']:
      query1 += 'public.l{} {},'.format(current_level - 1, alias)

   for alias in range(current_level):
      query1 += 'public.popular_movie_actors pma{}'.format(alias + 1)
      if alias != current_level - 1:
         query1 += ','
   query1 += ' WHERE '

   for alias in range(1, current_level):
      query1 += 'a.actor{} = pma{}.actor and '.format(alias, alias)

   query1 += 'b.actor{} = pma{}.actor and '.format(current_level - 1, current_level)

   for alias in range(1, current_level - 1):
      query1 += 'a.actor{} = b.actor{} and '.format(alias, alias)
```

```
    query1 += 'a.actor{} < b.actor{} and '.format(current_level - 1, current_level - 1)

    for alias in range(1, current_level):
        query1 += 'pma{}.title = pma{}.title'.format(alias, alias + 1)
        if alias != current_level - 1:
            query1 += ' and '

    query1 += ' GROUP BY '

    for alias in range(1, current_level):
        query1 += 'a.actor{}, '.format(alias)
    query1 += 'b.actor{}'.format(current_level - 1)
    query1 += ' HAVING COUNT(*) >= {};'.format(min_support)
    # print(query1)
    cur.execute(query1)
    conn.commit()
    query2 = 'SELECT * from l{}'.format(current_level)
    cur.execute(query2)

    number_of_records = len(cur.fetchall())

    if number_of_records == 0:
        break
    else:
        print('Level {} = {}'.format(current_level, number_of_records))
        current_level += 1

cur.close()
conn.close()
```

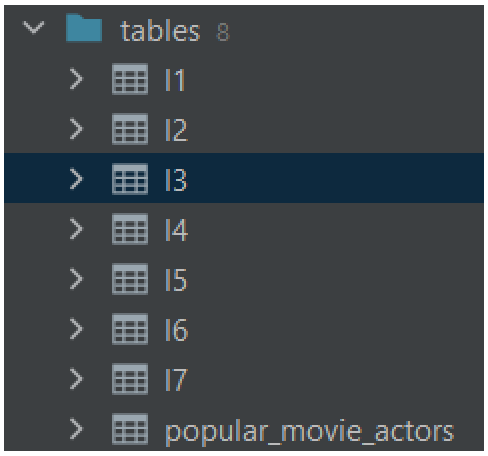
**Output**:

The output when the program is run



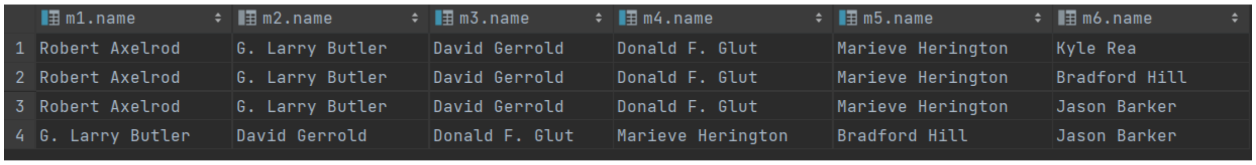
```
Level 2 = 4753
Level 3 = 628
Level 4 = 140
Level 5 = 28
Level 6 = 4


Process finished with exit code 0
```

As you can see the lattices are generated up to 7 for the popular_movie_actor file.


```
tables 8
  l1
  l2
  l3
  l4
  l5
  l6
  l7
  popular_movie_actors
```

Including the names of the actors in each frequent itemset from the last level of the lattice by joining the member and l6.

```
SELECT m1.name,m2.name,m3.name,m4.name,m5.name,m6.name FROM
hw_schema."Member" m1,
          hw_schema."Member" m2,
          hw_schema."Member" m3,
          hw_schema."Member" m4,
          hw_schema."Member" m5,
          hw_schema."Member" m6,
          public.l6 l6
WHERE m1.id = l6.actor1
and m2.id = l6.actor2
and m3.id = l6.actor3
and m4.id = l6.actor4
and m5.id = l6.actor5
and m6.id = l6.actor6;
```



| | m1.name | m2.name | m3.name | m4.name | m5.name | m6.name |
|---|---|---|---|---|---|---|
| 1 | Robert Axelrod | G. Larry Butler | David Gerrold | Donald F. Glut | Marieve Herington | Kyle Rea |
| 2 | Robert Axelrod | G. Larry Butler | David Gerrold | Donald F. Glut | Marieve Herington | Bradford Hill |
| 3 | Robert Axelrod | G. Larry Butler | David Gerrold | Donald F. Glut | Marieve Herington | Jason Barker |
| 4 | G. Larry Butler | David Gerrold | Donald F. Glut | Marieve Herington | Bradford Hill | Jason Barker |