Results from parts B1 and B2

```
/*

* These queries initialize the database and load it with data from
the parsed CSV files that are

* located in the root folder of this project under the 'db_files'
directory. This .sql script is intended

* to be called directly from the command line using the Postgres CLI.
If you run this script, be sure to change

* the paths to the CSV files below, as they are hard coded.

*/

/*** INITIALIZE TABLES ***/

DROP TABLE IF EXISTS Movies;

DROP TABLE
DROP TABLE IF EXISTS Genres;

DROP TABLE
DROP TABLE IF EXISTS Users;

DROP TABLE
DROP TABLE IF EXISTS Ratings;

DROP TABLE
DROP TABLE IF EXISTS Tags;

DROP TABLE
DROP TABLE IF EXISTS Has_genre;

DROP TABLE

/* entity sets */

CREATE TABLE Movies(
   id Integer PRIMARY KEY,

   title varchar,

   year Integer

);

CREATE TABLE


CREATE TABLE Genres(

   title varchar PRIMARY KEY

);

CREATE TABLE
CREATE TABLE Users(

   id Integer PRIMARY KEY

);

CREATE TABLE


/*relationship sets */

CREATE TABLE Ratings(

   user_id Integer,

   movie_id Integer,

   rating decimal,

   time_stamp bigint,

   PRIMARY KEY (user_id, movie_id)

);
```

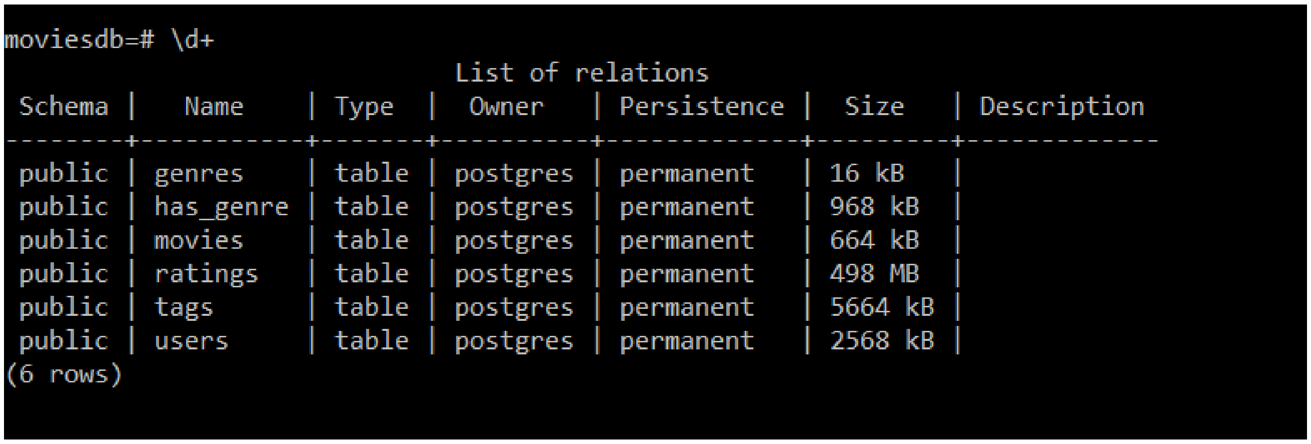
```
CREATE TABLE
CREATE TABLE Tags(

   user_id Integer,

   movie_id Integer,

   tag varchar,

   time_stamp bigint

);

CREATE TABLE
CREATE TABLE Has_genre(

   movie_id Integer,

   title varchar,

   PRIMARY KEY (movie_id, title)

);

CREATE TABLE

/*** LOAD DATA ***/

COPY Movies(id, title, year)

FROM
'C:\\Users\\Dominic\\DB-Project\\MoviesDB\\db_files\\movies_parsed.cs
v'

DELIMITER ',';

COPY 10681


COPY Genres(title)
```

```
FROM
'C:\\Users\\Dominic\\DB-Project\\MoviesDB\\db_files\\genres_parsed.cs
v'

DELIMITER ',';

COPY 19


COPY Ratings(user_id, movie_id, rating, time_stamp)

FROM
'C:\\Users\\Dominic\\DB-Project\\MoviesDB\\db_files\\ratings_parsed.c
sv'

DELIMITER ',';

COPY 10000054


COPY Tags(user_id, movie_id, tag, time_stamp)

FROM
'C:\\Users\\Dominic\\DB-Project\\MoviesDB\\db_files\\tags_parsed.csv'

DELIMITER ',';

COPY 95580


COPY Has_genre(movie_id, title)

FROM
'C:\\Users\\Dominic\\DB-Project\\MoviesDB\\db_files\\has_genre_parsed
.csv'

DELIMITER ',';

COPY 21564


INSERT INTO Users
```

```
SELECT R.user_id

FROM Ratings R

UNION

SELECT T.user_id

FROM Tags T


INSERT 0 71567
```

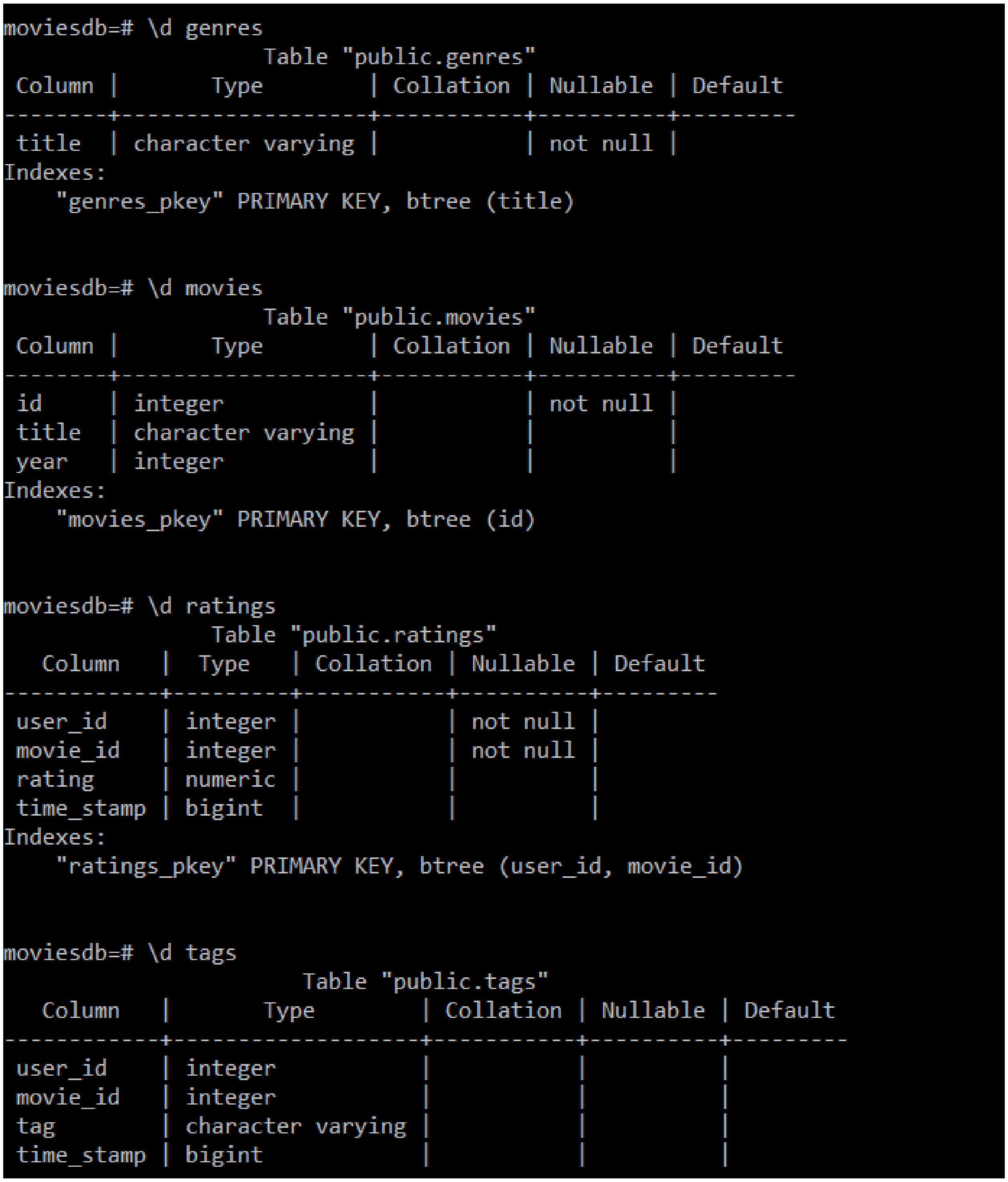
Results from part B4

A) Listing tables:

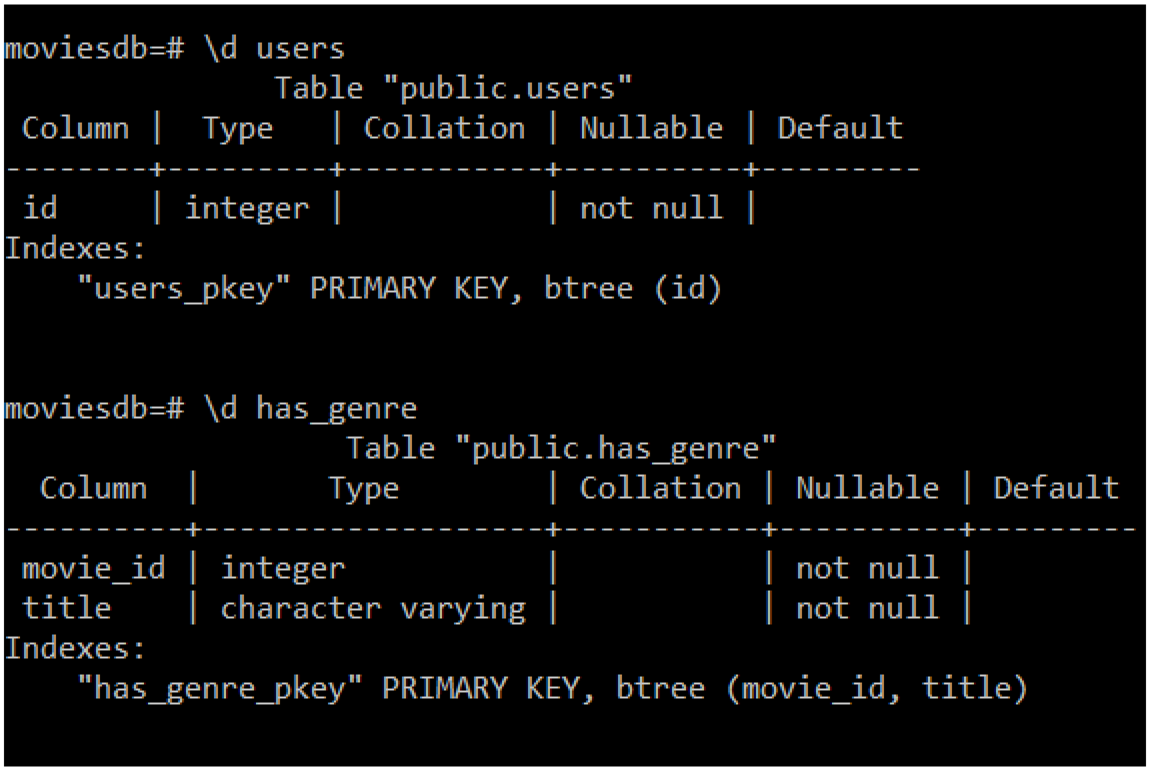


```
moviesdb=# \d+
                              List of relations
 Schema |   Name    | Type  |  Owner   | Persistence |  Size   | Description
--------+-----------+-------+----------+-------------+---------+-------------
 public | genres    | table | postgres | permanent   | 16 kB   |
 public | has_genre | table | postgres | permanent   | 968 kB  |
 public | movies    | table | postgres | permanent   | 664 kB  |
 public | ratings   | table | postgres | permanent   | 498 MB  |
 public | tags      | table | postgres | permanent   | 5664 kB |
 public | users     | table | postgres | permanent   | 2568 kB |
(6 rows)
```

B) Listing data types of tables:

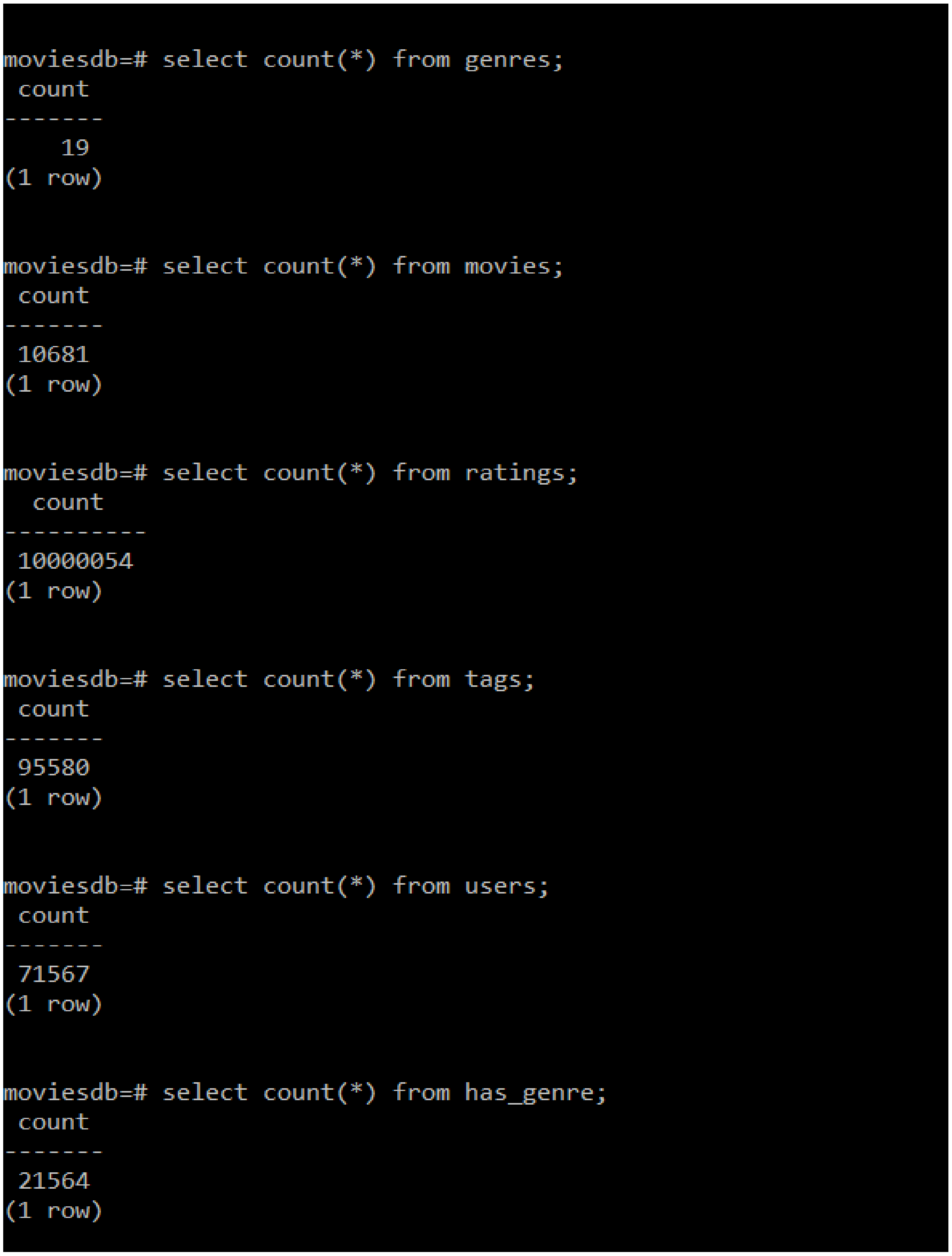


```
moviesdb=# \d genres
                    Table "public.genres"
 Column |       Type        | Collation | Nullable | Default
--------+-------------------+-----------+----------+---------
 title  | character varying |           | not null |
Indexes:
    "genres_pkey" PRIMARY KEY, btree (title)


moviesdb=# \d movies
                    Table "public.movies"
 Column |       Type        | Collation | Nullable | Default
--------+-------------------+-----------+----------+---------
 id     | integer           |           | not null |
 title  | character varying |           |          |
 year   | integer           |           |          |
Indexes:
    "movies_pkey" PRIMARY KEY, btree (id)


moviesdb=# \d ratings
                Table "public.ratings"
   Column   |  Type   | Collation | Nullable | Default
------------+---------+-----------+----------+---------
 user_id    | integer |           | not null |
 movie_id   | integer |           | not null |
 rating     | numeric |           |          |
 time_stamp | bigint  |           |          |
Indexes:
    "ratings_pkey" PRIMARY KEY, btree (user_id, movie_id)


moviesdb=# \d tags
                       Table "public.tags"
   Column   |       Type        | Collation | Nullable | Default
------------+-------------------+-----------+----------+---------
 user_id    | integer           |           |          |
 movie_id   | integer           |           |          |
 tag        | character varying |           |          |
 time_stamp | bigint            |           |          |
```



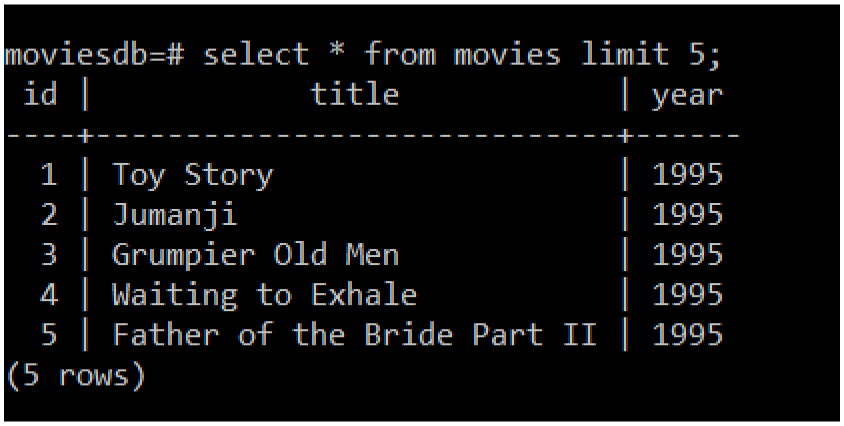
```
moviesdb=# \d users
               Table "public.users"
 Column |  Type   | Collation | Nullable | Default
--------+---------+-----------+----------+---------
 id     | integer |           | not null |
Indexes:
    "users_pkey" PRIMARY KEY, btree (id)


moviesdb=# \d has_genre
                   Table "public.has_genre"
  Column  |       Type        | Collation | Nullable | Default
----------+-------------------+-----------+----------+---------
 movie_id | integer           |           | not null |
 title    | character varying |           | not null |
Indexes:
    "has_genre_pkey" PRIMARY KEY, btree (movie_id, title)
```

C) Counting rows in tables:



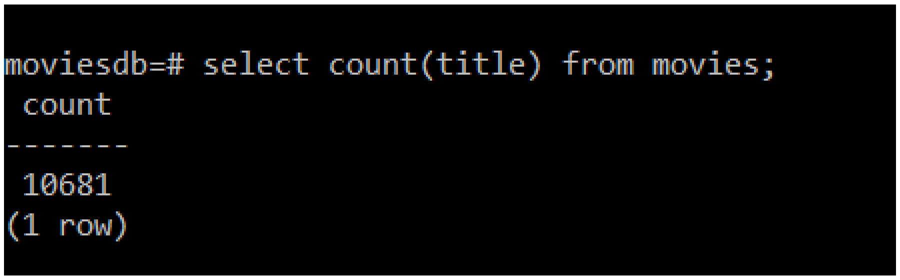
```
moviesdb=# select count(*) from genres;
 count
-------
    19
(1 row)


moviesdb=# select count(*) from movies;
 count
-------
 10681
(1 row)


moviesdb=# select count(*) from ratings;
  count
----------
 10000054
(1 row)


moviesdb=# select count(*) from tags;
 count
-------
 95580
(1 row)


moviesdb=# select count(*) from users;
 count
-------
 71567
(1 row)


moviesdb=# select count(*) from has_genre;
 count
-------
 21564
(1 row)
```

D)
First 5 lines of movies table:



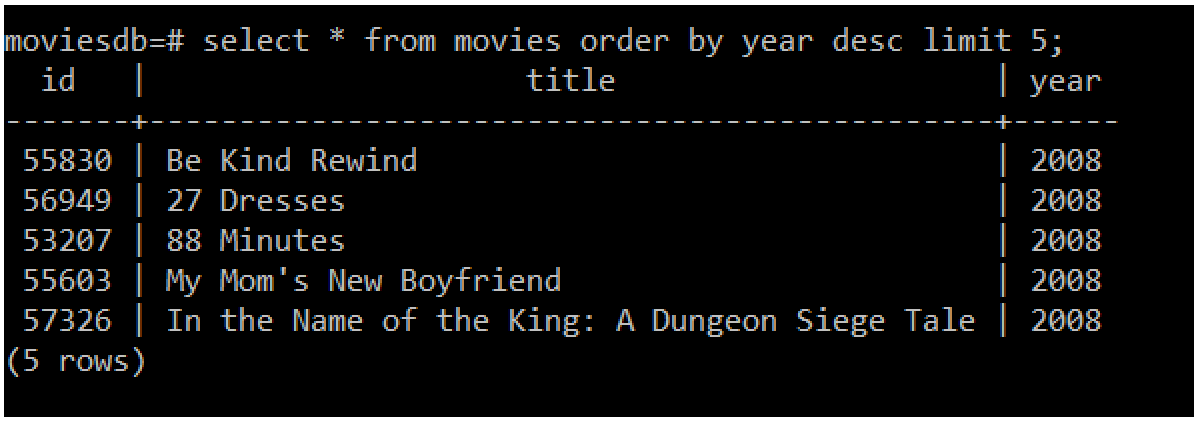
```
moviesdb=# select * from movies limit 5;
 id |             title             | year
----+-------------------------------+------
  1 | Toy Story                     | 1995
  2 | Jumanji                       | 1995
  3 | Grumpier Old Men              | 1995
  4 | Waiting to Exhale             | 1995
  5 | Father of the Bride Part II   | 1995
(5 rows)
```

Number of non NULL titles:



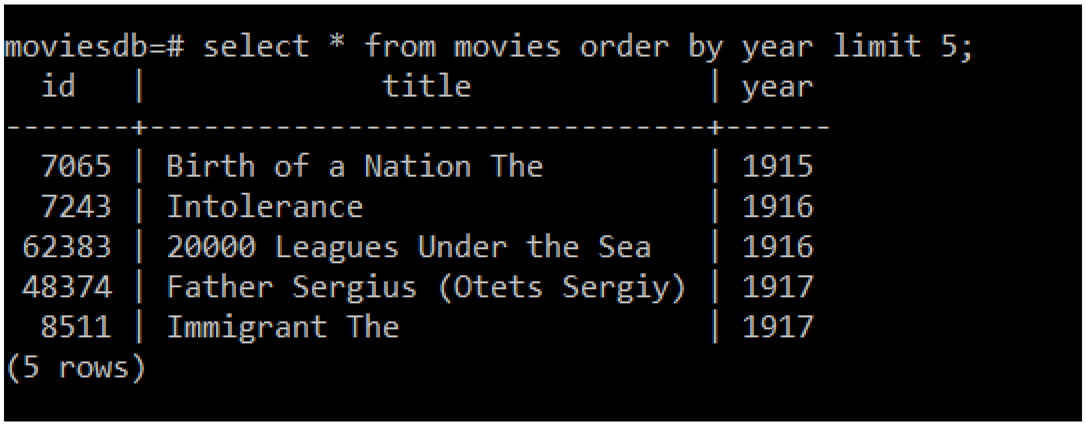
```
moviesdb=# select count(title) from movies;
 count
-------
 10681
(1 row)
```

Last 5 lines of movies table:



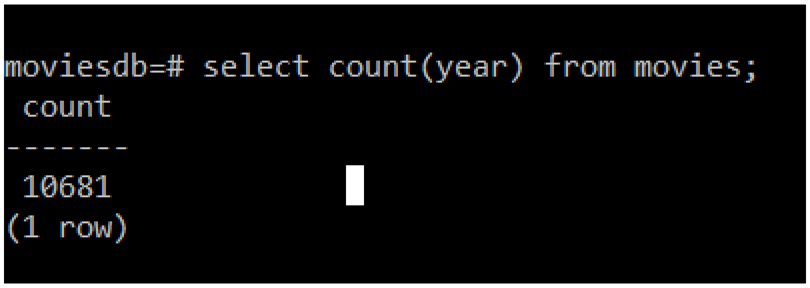
```
moviesdb=# select * from movies order by year desc limit 5;
  id   |                      title                       | year
-------+--------------------------------------------------+------
 55830 | Be Kind Rewind                                   | 2008
 56949 | 27 Dresses                                       | 2008
 53207 | 88 Minutes                                       | 2008
 55603 | My Mom's New Boyfriend                           | 2008
 57326 | In the Name of the King: A Dungeon Siege Tale    | 2008
(5 rows)
```

Sorting by year, limiting to 5 rows:



```
moviesdb=# select * from movies order by year limit 5;
  id   |             title              | year
-------+--------------------------------+------
  7065 | Birth of a Nation The          | 1915
  7243 | Intolerance                    | 1916
 62383 | 20000 Leagues Under the Sea    | 1916
 48374 | Father Sergius (Otets Sergiy) | 1917
  8511 | Immigrant The                  | 1917
(5 rows)
```

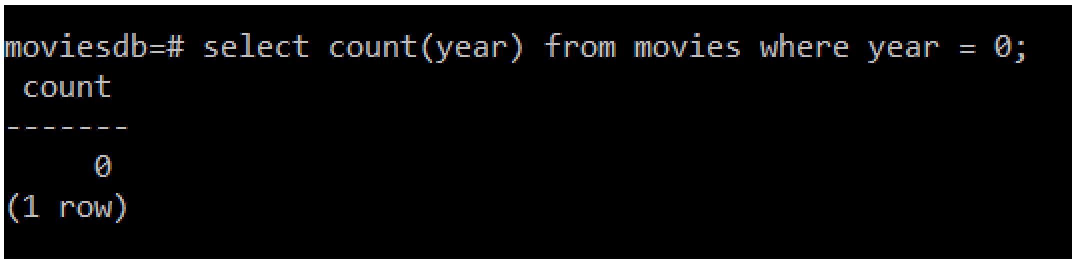
Checking NULL values in year column:



```
moviesdb=# select count(year) from movies;
 count
-------
 10681
(1 row)
```

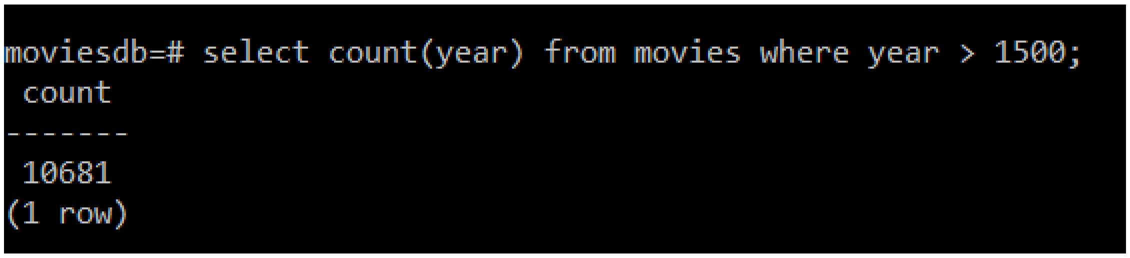
There are no NULL values in the year column

Checking rows where year = 0:



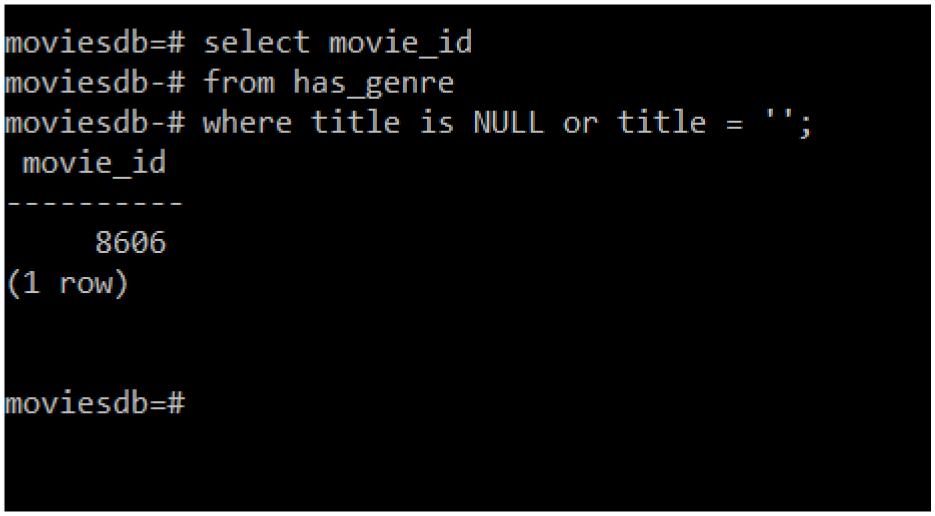
```
moviesdb=# select count(year) from movies where year = 0;
 count
-------
     0
(1 row)
```

Checking rows where year >1500 or non-zero:



```
moviesdb=# select count(year) from movies where year > 1500;
 count
-------
 10681
(1 row)
```

Checking movies that have no genres associated:

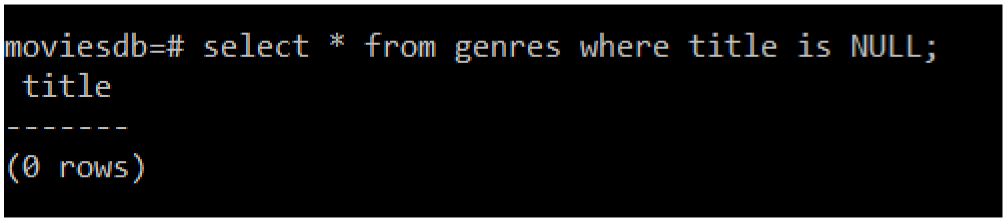


```
moviesdb=# select movie_id
moviesdb-# from has_genre
moviesdb-# where title is NULL or title = '';
 movie_id
----------
     8606
(1 row)


moviesdb=#
```

Movie with id 8606 has no genre

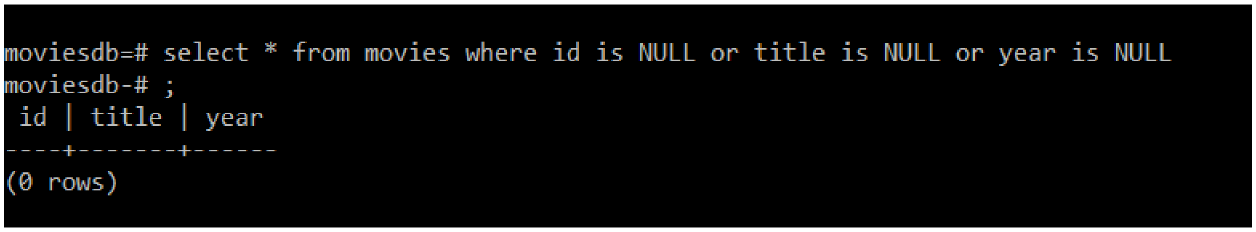
1) Find unknown or invalid data in any of the attributes for all tables

Check genres:



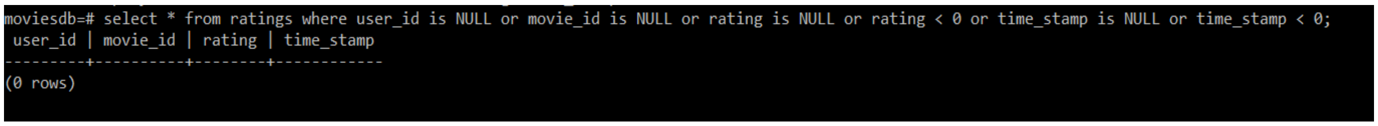
```
moviesdb=# select * from genres where title is NULL;
 title
-------
(0 rows)
```

Check movies:



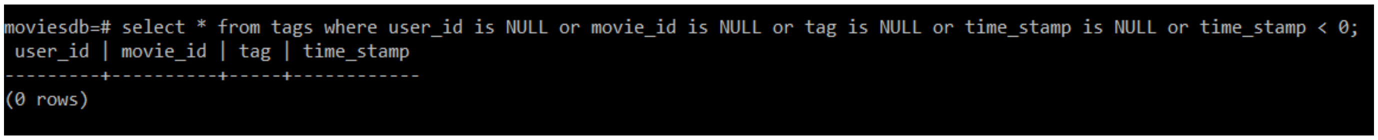
```
moviesdb=# select * from movies where id is NULL or title is NULL or year is NULL
moviesdb-# ;
 id | title | year
----+-------+------
(0 rows)
```

Check ratings:



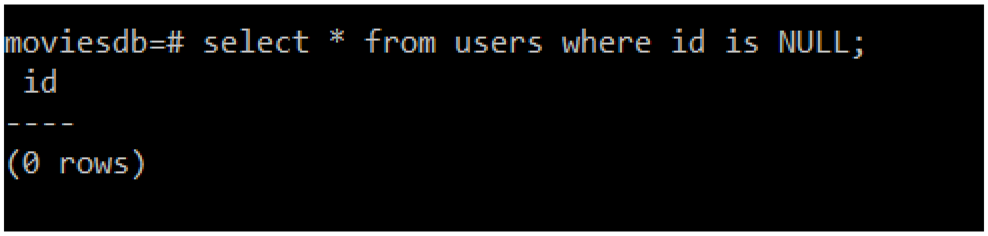
```
moviesdb=# select * from ratings where user_id is NULL or movie_id is NULL or rating is NULL or rating < 0 or time_stamp is NULL or time_stamp < 0;
 user_id | movie_id | rating | time_stamp
---------+----------+--------+------------
(0 rows)
```

Check tags:



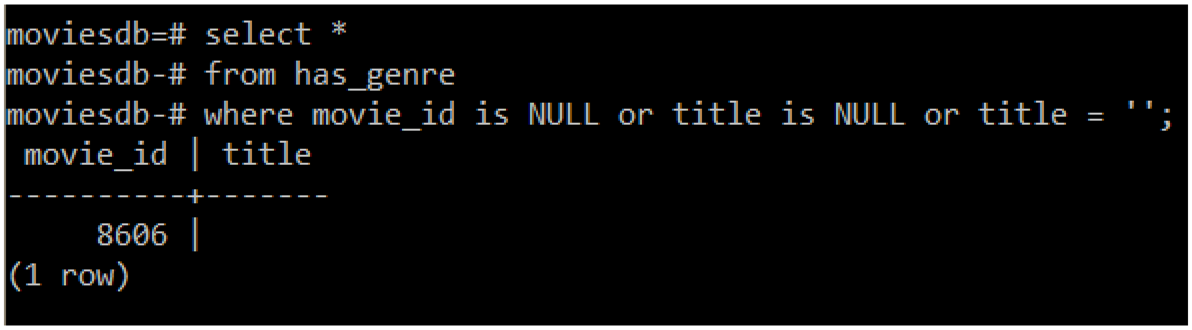
```
moviesdb=# select * from tags where user_id is NULL or movie_id is NULL or tag is NULL or time_stamp is NULL or time_stamp < 0;
 user_id | movie_id | tag | time_stamp
---------+----------+-----+------------
(0 rows)
```

Check users:



```
moviesdb=# select * from users where id is NULL;
 id
----
(0 rows)
```

Check has_genre:



```
moviesdb=# select *
moviesdb-# from has_genre
moviesdb-# where movie_id is NULL or title is NULL or title = '';
 movie_id | title
----------+-------
     8606 |
(1 row)
```

2) Find the distribution of the values for attribute year in movies:

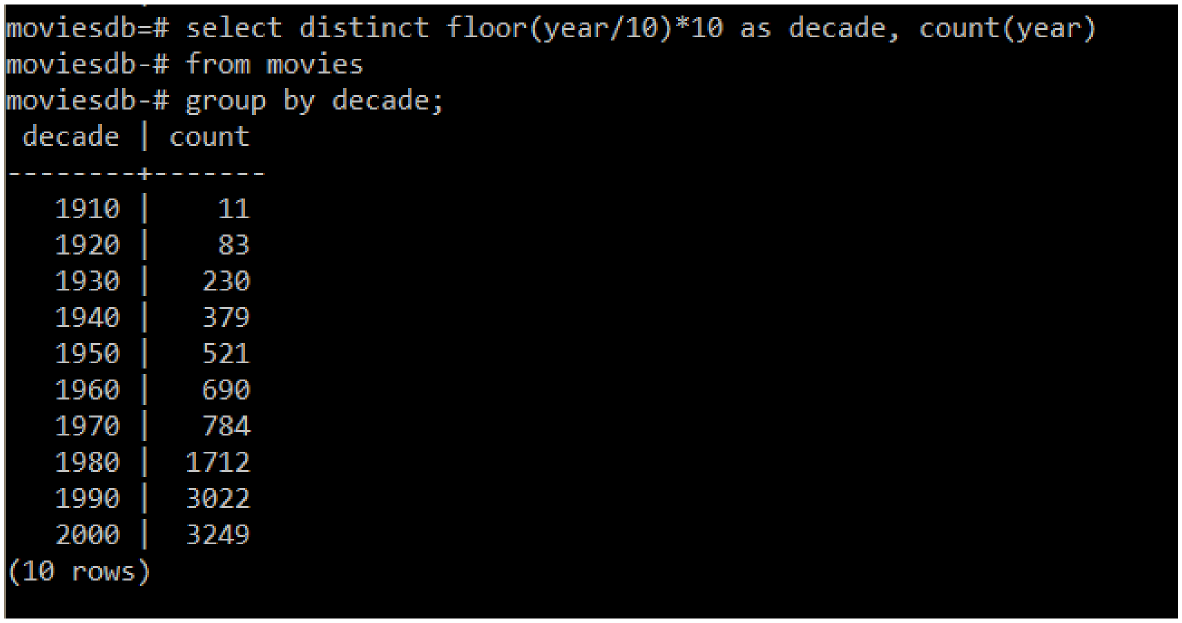
moviesdb=# select distinct year, count(year)
moviesdb-# from movies
moviesdb-# group by year
moviesdb-# order by year asc;

```
 year | count
------+-------
 1915 |     1
 1916 |     2
 1917 |     2
 1918 |     2
 1919 |     4
 1920 |     5
 1921 |     3
 1922 |     7
 1923 |     6
 1924 |     6
 1925 |    10
 1926 |    10
 1927 |    19
 1928 |    10
 1929 |     7
```

1930	15
1931	16
1932	22
1933	23
1934	18
1935	18
1936	32
1937	30
1938	19
1939	37
1940	40
1941	28
1942	38
1943	40
1944	37
1945	36
1946	38
1947	39
1948	46
1949	37
1950	44
1951	44
1952	40
1953	55
1954	43
1955	57
1956	53
1957	62
1958	62
1959	61
1960	66
1961	57
1962	69
1963	63
1964	72
1965	72
1966	87
1967	68
1968	72
1969	64
1970	71
1971	73
1972	83
1973	81

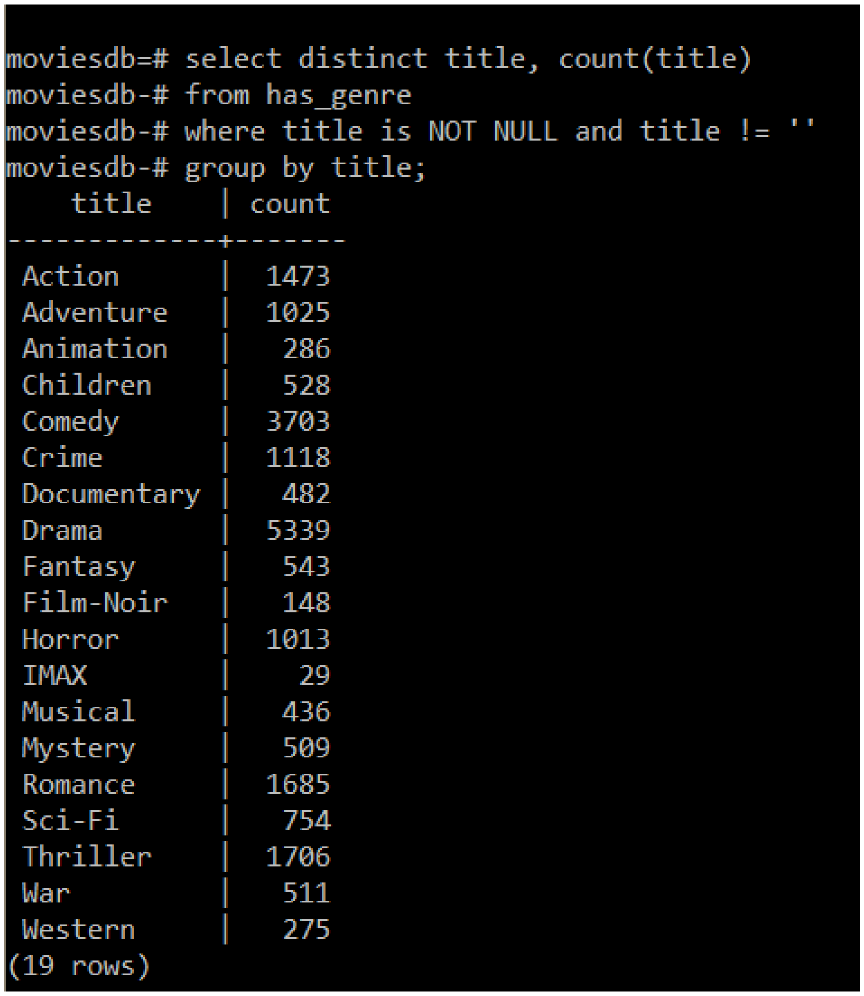
```
 1974 |    75
 1975 |    74
 1976 |    75
 1977 |    83
 1978 |    82
 1979 |    87
 1980 |   161
 1981 |   178
 1982 |   170
 1983 |   111
 1984 |   137
 1985 |   158
 1986 |   166
 1987 |   205
 1988 |   214
 1989 |   212
 1990 |   200
 1991 |   188
 1992 |   212
 1993 |   258
 1994 |   307
 1995 |   362
 1996 |   384
 1997 |   370
 1998 |   384
 1999 |   357
 2000 |   405
 2001 |   403
 2002 |   441
 2003 |   366
 2004 |   342
 2005 |   332
 2006 |   345
 2007 |   364
 2008 |   251
(94 rows)
```

3) Find the distribution of movies across different decades



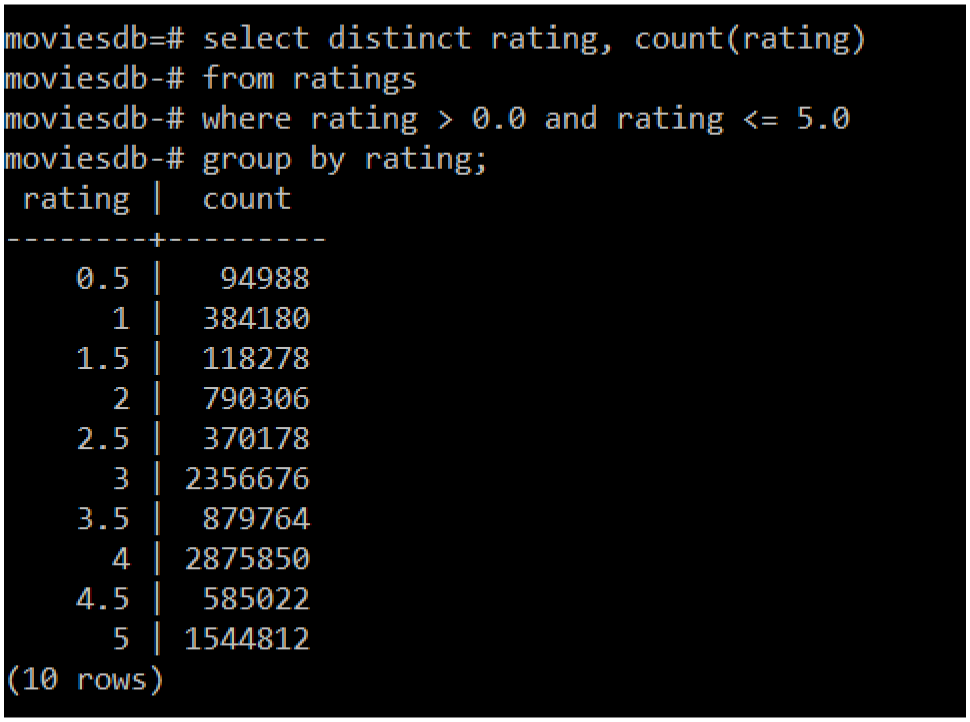
```
moviesdb=# select distinct floor(year/10)*10 as decade, count(year)
moviesdb-# from movies
moviesdb-# group by decade;
 decade | count
--------+-------
   1910 |    11
   1920 |    83
   1930 |   230
   1940 |   379
   1950 |   521
   1960 |   690
   1970 |   784
   1980 |  1712
   1990 |  3022
   2000 |  3249
(10 rows)
```

4) Find the distribution of genres across movies



```
moviesdb=# select distinct title, count(title)
moviesdb-# from has_genre
moviesdb-# where title is NOT NULL and title != ''
moviesdb-# group by title;
    title     | count
-------------+-------
 Action      |  1473
 Adventure   |  1025
 Animation   |   286
 Children    |   528
 Comedy      |  3703
 Crime       |  1118
 Documentary |   482
 Drama       |  5339
 Fantasy     |   543
 Film-Noir   |   148
 Horror      |  1013
 IMAX        |    29
 Musical     |   436
 Mystery     |   509
 Romance     |  1685
 Sci-Fi      |   754
 Thriller    |  1706
 War         |   511
 Western     |   275
(19 rows)
```

5) Find the distribution of ratings

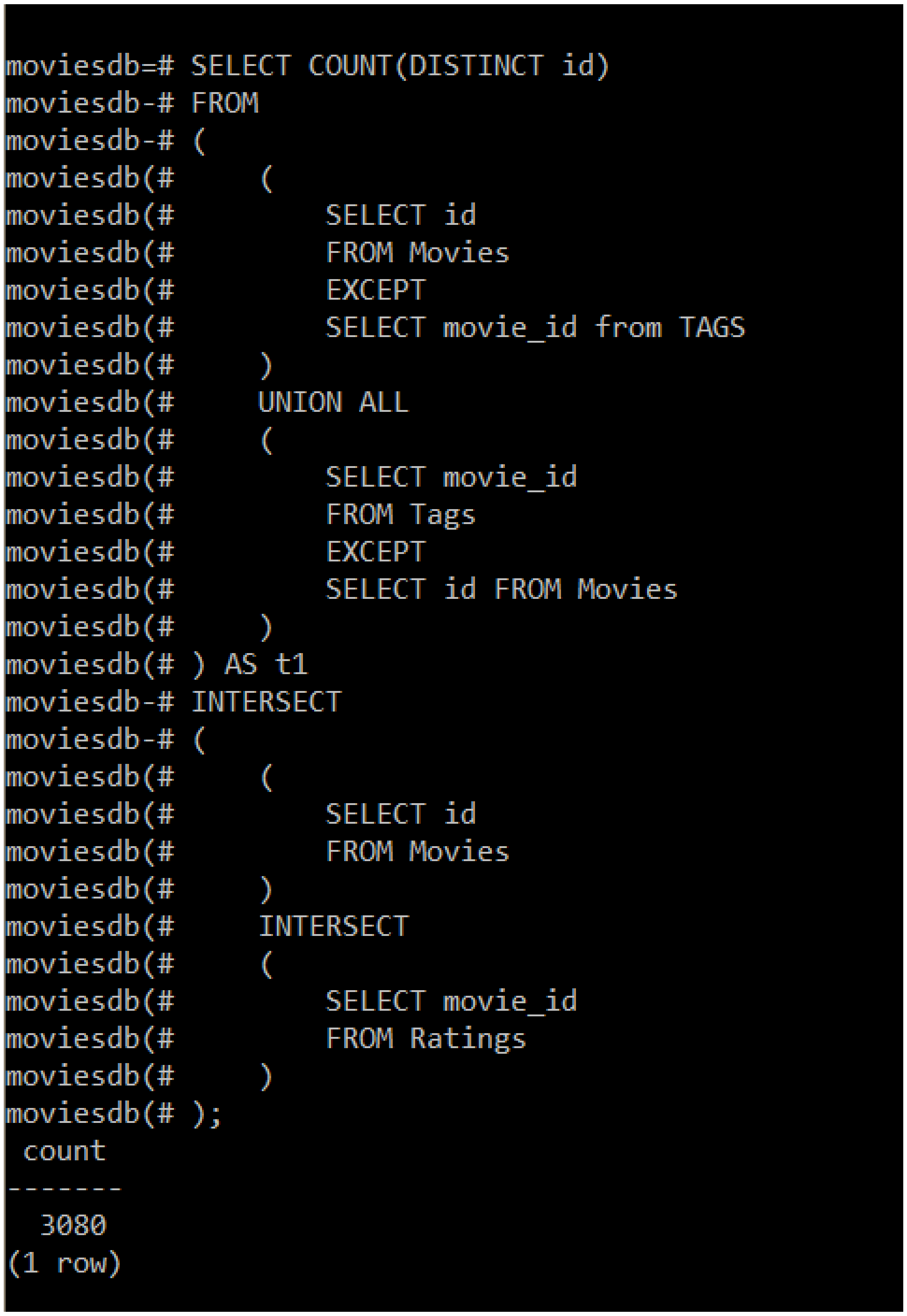


```
moviesdb=# select distinct rating, count(rating)
moviesdb-# from ratings
moviesdb-# where rating > 0.0 and rating <= 5.0
moviesdb-# group by rating;
 rating |  count
--------+---------
    0.5 |   94988
      1 |  384180
    1.5 |  118278
      2 |  790306
    2.5 |  370178
      3 | 2356676
    3.5 |  879764
      4 | 2875850
    4.5 |  585022
      5 | 1544812
(10 rows)
```

6) Find how many movies have:

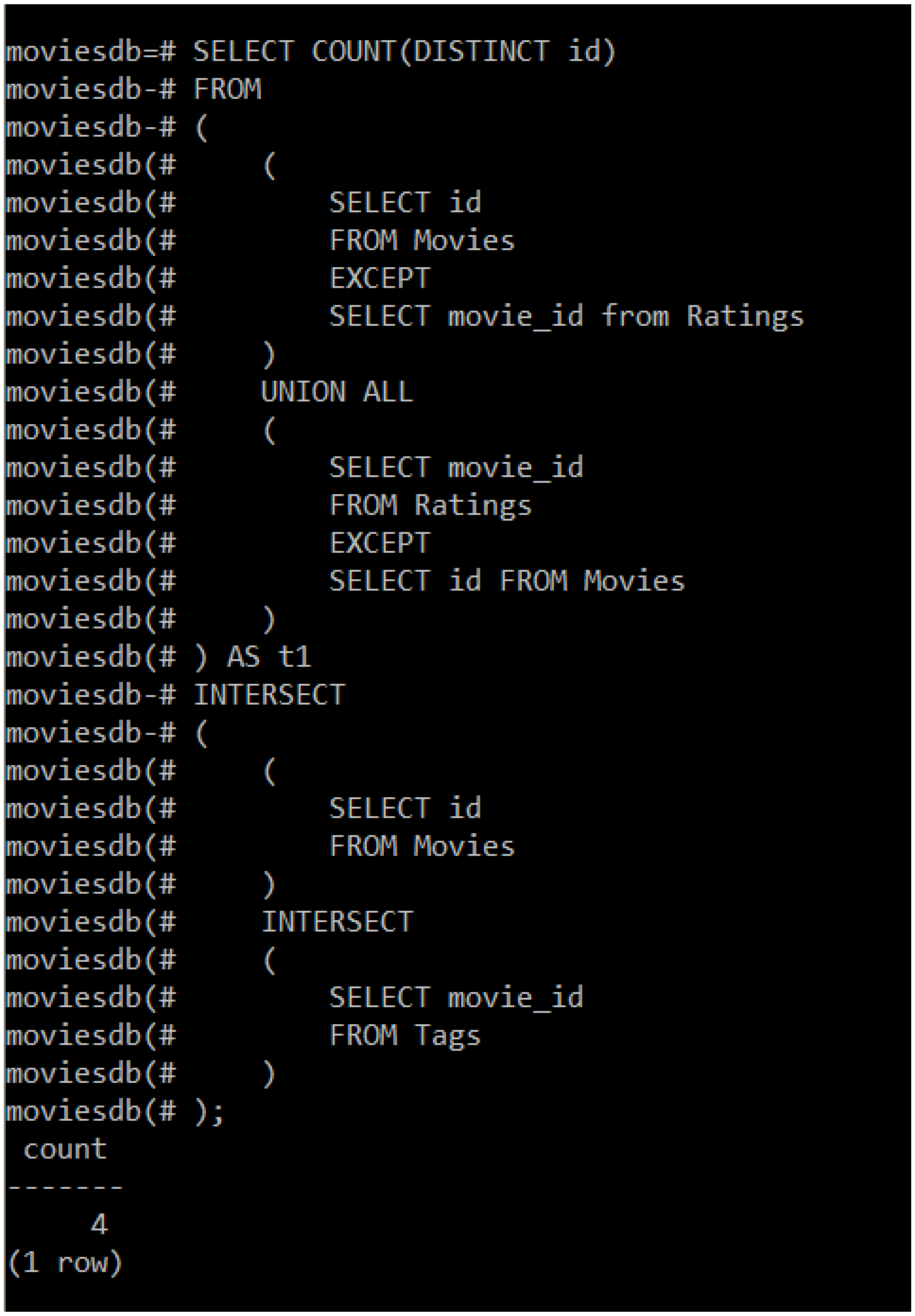
i) No tags, but has ratings

- Movies with no tags = difference of (Movies - Tags) on movie_id
- Movies with ratings = intersection of Movies and Ratings on movie_id
- Movies with no tags, but with ratings = intersection of the results above ^^



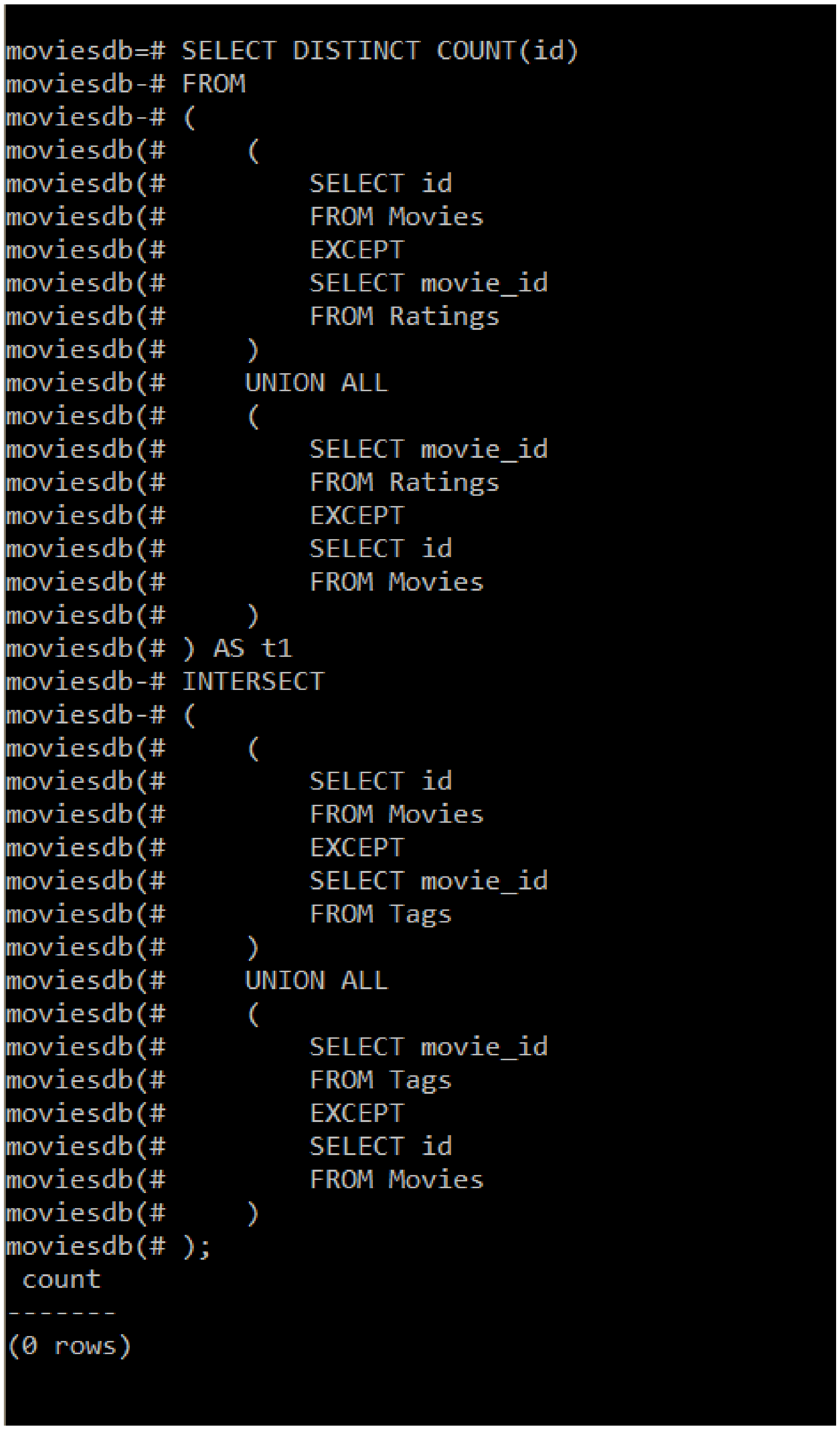
```
moviesdb=# SELECT COUNT(DISTINCT id)
moviesdb-# FROM
moviesdb-# (
moviesdb(#     (
moviesdb(#         SELECT id
moviesdb(#         FROM Movies
moviesdb(#         EXCEPT
moviesdb(#         SELECT movie_id from TAGS
moviesdb(#     )
moviesdb(#     UNION ALL
moviesdb(#     (
moviesdb(#         SELECT movie_id
moviesdb(#         FROM Tags
moviesdb(#         EXCEPT
moviesdb(#         SELECT id FROM Movies
moviesdb(#     )
moviesdb(# ) AS t1
moviesdb-# INTERSECT
moviesdb-# (
moviesdb(#     (
moviesdb(#         SELECT id
moviesdb(#         FROM Movies
moviesdb(#     )
moviesdb(#     INTERSECT
moviesdb(#     (
moviesdb(#         SELECT movie_id
moviesdb(#         FROM Ratings
moviesdb(#     )
moviesdb(# );
 count
-------
  3080
(1 row)
```

ii) No ratings, but has tags



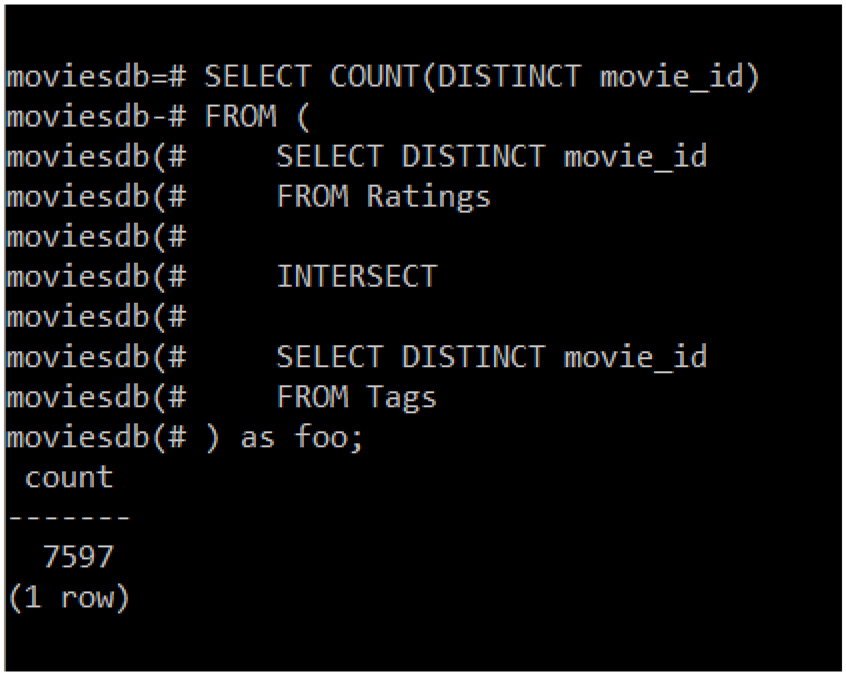
```
moviesdb=# SELECT COUNT(DISTINCT id)
moviesdb-# FROM
moviesdb-# (
moviesdb(#     (
moviesdb(#         SELECT id
moviesdb(#         FROM Movies
moviesdb(#         EXCEPT
moviesdb(#         SELECT movie_id from Ratings
moviesdb(#     )
moviesdb(#     UNION ALL
moviesdb(#     (
moviesdb(#         SELECT movie_id
moviesdb(#         FROM Ratings
moviesdb(#         EXCEPT
moviesdb(#         SELECT id FROM Movies
moviesdb(#     )
moviesdb(# ) AS t1
moviesdb-# INTERSECT
moviesdb-# (
moviesdb(#     (
moviesdb(#         SELECT id
moviesdb(#         FROM Movies
moviesdb(#     )
moviesdb(#     INTERSECT
moviesdb(#     (
moviesdb(#         SELECT movie_id
moviesdb(#         FROM Tags
moviesdb(#     )
moviesdb(# );
 count
-------
     4
(1 row)
```

iii) No tags and no ratings
Movies with no tags = difference of Movies and Tags
Movies with no ratings = difference of Movies and Ratings
Movies with no tags and ratings = intersection of two tables above ^^



```
moviesdb=# SELECT DISTINCT COUNT(id)
moviesdb-# FROM
moviesdb-# (
moviesdb(#     (
moviesdb(#         SELECT id
moviesdb(#         FROM Movies
moviesdb(#         EXCEPT
moviesdb(#         SELECT movie_id
moviesdb(#         FROM Ratings
moviesdb(#     )
moviesdb(#     UNION ALL
moviesdb(#     (
moviesdb(#         SELECT movie_id
moviesdb(#         FROM Ratings
moviesdb(#         EXCEPT
moviesdb(#         SELECT id
moviesdb(#         FROM Movies
moviesdb(#     )
moviesdb(# ) AS t1
moviesdb-# INTERSECT
moviesdb-# (
moviesdb(#     (
moviesdb(#         SELECT id
moviesdb(#         FROM Movies
moviesdb(#         EXCEPT
moviesdb(#         SELECT movie_id
moviesdb(#         FROM Tags
moviesdb(#     )
moviesdb(#     UNION ALL
moviesdb(#     (
moviesdb(#         SELECT movie_id
moviesdb(#         FROM Tags
moviesdb(#         EXCEPT
moviesdb(#         SELECT id
moviesdb(#         FROM Movies
moviesdb(#     )
moviesdb(# );
 count
-------
(0 rows)
```

iv) Both tags and ratings



```
moviesdb=# SELECT COUNT(DISTINCT movie_id)
moviesdb-# FROM (
moviesdb(#     SELECT DISTINCT movie_id
moviesdb(#     FROM Ratings
moviesdb(#
moviesdb(#     INTERSECT
moviesdb(#
moviesdb(#     SELECT DISTINCT movie_id
moviesdb(#     FROM Tags
moviesdb(# ) as foo;
 count
-------
  7597
(1 row)
```