JS_ST_Assignment05v3

October 29, 2015

1 MovieLens Rating Data

Data was retrieved from www.BigML.com. The name of the dataset is MovieLens-1M Ratings. To download the dataset:

Go to www.BigML.com
Create a username (free) and change your account to "production" mode.
Click on the dropdown menu for Gallery, then select Public.
Switch from searching for Models to searching for Datasets.
Under "All Categories" filter for the "Consumer & Retail" datasets.
Find the dataset called "MovieLens - 1M Ratings"
Clone the dataset to your own account.
Request an Export of the data, then download the csv file that is created.
Change the name of the data to MovieLens.csv.
Update the permissions of the csv so that all user groups have read, write, and execute permissions.

About the data:

MovieLens is a movie rating system similar to NetFlix. Users provide basic personal information, and then rate movies. MovieLens then recommends movies that it thinks they will enjoy.
The dataset includes 1 million such ratings.

```
In [26]: %%bash
         cat MovieLens.csv | head -500 | csvstat
```

```
1. User age range
        <type 'unicode'>
        Nulls: False
        Values: Under 18
  2. User Gender
        <type 'unicode'>
        Nulls: False
        Values: MALE, FEMALE
  3. User Occupation
        <type 'unicode'>
        Nulls: False
        Values: other or not specified
  4. User ZIP
        <type 'unicode'>
        Nulls: False
        Unique values: 7
        5 most frequent values:
                11803:        180
                08055:        97
                48133:        76
                75244:        48
```

```
              97222:        39
      Max length: 5
 5. Genre Action
      <type 'bool'>
      Nulls: False
      Unique values: 2
      5 most frequent values:
              False:        390
              True:        109
 6. Genre Adventure
      <type 'bool'>
      Nulls: False
      Unique values: 2
      5 most frequent values:
              False:        447
              True:        52
 7. Genre Animation
      <type 'bool'>
      Nulls: False
      Unique values: 2
      5 most frequent values:
              False:        455
              True:        44
 8. Genre Children
      <type 'bool'>
      Nulls: False
      Unique values: 2
      5 most frequent values:
              False:        457
              True:        42
 9. Genre Comedy
      <type 'bool'>
      Nulls: False
      Unique values: 2
      5 most frequent values:
              False:        319
              True:        180
10. Genre Crime
      <type 'bool'>
      Nulls: False
      Unique values: 2
      5 most frequent values:
              False:        467
              True:        32
11. Genre Documentary
      <type 'bool'>
      Nulls: False
      Unique values: 1
      5 most frequent values:
              False:        499
12. Genre Drama
      <type 'bool'>
      Nulls: False
      Unique values: 2
```

```
       5 most frequent values:
              False:        378
              True:         121
13. Genre Fantasy
       <type 'bool'>
       Nulls: False
       Unique values: 2
       5 most frequent values:
              False:        490
              True:         9
14. Genre Film-Noir
       <type 'bool'>
       Nulls: False
       Unique values: 2
       5 most frequent values:
              False:        493
              True:         6
15. Genre Horror
       <type 'bool'>
       Nulls: False
       Unique values: 2
       5 most frequent values:
              False:        412
              True:         87
16. Genre Musical
       <type 'bool'>
       Nulls: False
       Unique values: 2
       5 most frequent values:
              False:        478
              True:         21
17. Genre Mystery
       <type 'bool'>
       Nulls: False
       Unique values: 2
       5 most frequent values:
              False:        478
              True:         21
18. Genre Romance
       <type 'bool'>
       Nulls: False
       Unique values: 2
       5 most frequent values:
              False:        455
              True:         44
19. Genre Sci-Fi
       <type 'bool'>
       Nulls: False
       Unique values: 2
       5 most frequent values:
              False:        404
              True:         95
20. Genre Thriller
       <type 'bool'>
```

```
        Nulls: False
        Unique values: 2
        5 most frequent values:
                False:        358
                True:         141
 21. Genre War
        <type 'bool'>
        Nulls: False
        Unique values: 2
        5 most frequent values:
                False:        474
                True:         25
 22. Genre Western
        <type 'bool'>
        Nulls: False
        Unique values: 2
        5 most frequent values:
                False:        495
                True:         4
 23. Rating
        <type 'int'>
        Nulls: False
        Values: 1, 2, 3, 4, 5

Row count: 499

cat: write error: Broken pipe
```

The file includes four columns of user data (age range, gender, occupation, and zip code). Then, there are 18 columns to determine the genre of the movie being rated. Finally, there is a rating column where movies receive a score between 1 and 5.

MovieLens can use this dataset to build recommendation algorithms for users.

```
In [27]: %load_ext sql

The sql extension is already loaded. To reload it, use:
  %reload_ext sql

In [28]: !echo "DROP DATABASE movielensdb" | mysql --user=mysqluser --password=mysqlpass

In [29]: !echo "CREATE DATABASE movielensdb" | mysql --user=mysqluser --password=mysqlpass

In [30]: %sql mysql://mysqluser:mysqlpass@localhost/movielensdb

Out[30]: u'Connected: mysqluser@movielensdb'
```

First, create a temporary table (ratings_temp) to house the raw data:

```
In [31]: %%sql
         CREATE TABLE ratings_temp (
           user_age_range VARCHAR(10),
           user_gender VARCHAR(6),
           user_occupation VARCHAR(30),
           user_ZIP VARCHAR(20),
           genre_action VARCHAR(10),
           genre_adventure VARCHAR(10),
```

```
            genre_animation VARCHAR(10),
            genre_children VARCHAR(10),
            genre_comedy VARCHAR(10),
            genre_crime VARCHAR(10),
            genre_documentary VARCHAR(10),
            genre_drama VARCHAR(10),
            genre_fantasy VARCHAR(10),
            genre_filmnoir VARCHAR(10),
            genre_horror VARCHAR(10),
            genre_musical VARCHAR(10),
            genre_mystery VARCHAR(10),
            genre_romance VARCHAR(10),
            genre_scifi VARCHAR(10),
            genre_thriller VARCHAR(10),
            genre_war VARCHAR(10),
            genre_western VARCHAR(10),
            rating int
            )
```

```
0 rows affected.
```

```
Out[31]: []
```

```
In [32]: %%sql
         LOAD DATA INFILE '/home/vagrant/DWH Notebooks/MovieLens.csv'
         REPLACE
         INTO TABLE ratings_temp
         FIELDS TERMINATED BY ','
                OPTIONALLY ENCLOSED BY '"'
         IGNORE 1 LINES
```

```
1000209 rows affected.
```

```
Out[32]: []
```

```
In [33]: %%sql
         SELECT * FROM ratings_temp
         LIMIT 10
```

```
10 rows affected.
```

```
Out[33]: [('Under 18', 'MALE', 'other or not specified', '75070', 'TRUE', 'FALSE', 'FALSE', 'FALSE', 'FA
          ('Under 18', 'MALE', 'other or not specified', '75070', 'FALSE', 'FALSE', 'TRUE', 'TRUE', 'TRU
          ('Under 18', 'MALE', 'other or not specified', '75070', 'FALSE', 'FALSE', 'FALSE', 'FALSE', '
          ('Under 18', 'MALE', 'other or not specified', '75070', 'FALSE', 'FALSE', 'FALSE', 'FALSE', '
          ('Under 18', 'MALE', 'other or not specified', '75070', 'FALSE', 'FALSE', 'FALSE', 'FALSE', '
          ('Under 18', 'MALE', 'other or not specified', '75070', 'FALSE', 'FALSE', 'FALSE', 'FALSE', '
          ('Under 18', 'MALE', 'other or not specified', '75070', 'FALSE', 'FALSE', 'TRUE', 'FALSE', 'TR
          ('Under 18', 'MALE', 'other or not specified', '75070', 'FALSE', 'FALSE', 'FALSE', 'FALSE', '
          ('Under 18', 'MALE', 'other or not specified', '75070', 'FALSE', 'FALSE', 'FALSE', 'FALSE', '
          ('Under 18', 'MALE', 'other or not specified', '75070', 'FALSE', 'FALSE', 'TRUE', 'FALSE', 'TR
```

Next, to normalize the data we will place the users' personal information into a separate table called user_dimension. Each set of unique attributes will be assigned a unique id called user_dim_id. This will allow us to use one ID column rather than the 4 user dimension columns in the rating table.

The user_dim_id is a primary key which auto-increments each time a record is added.

```
In [34]: %%sql
         SELECT DISTINCT user_age_range, user_gender, user_occupation, user_ZIP
         FROM ratings_temp
         LIMIT 10
```

```
10 rows affected.
```

```
Out[34]: [('Under 18', 'MALE', 'other or not specified', '75070'),
          ('Under 18', 'MALE', 'other or not specified', '48133'),
          ('Under 18', 'FEMALE', 'other or not specified', '90210'),
          ('Under 18', 'MALE', 'other or not specified', '97222'),
          ('Under 18', 'MALE', 'other or not specified', '75244'),
          ('Under 18', 'MALE', 'other or not specified', '08055'),
          ('Under 18', 'MALE', 'other or not specified', '11803'),
          ('Under 18', 'MALE', 'other or not specified', '77479'),
          ('Under 18', 'MALE', 'other or not specified', '94525'),
          ('Under 18', 'FEMALE', 'other or not specified', '95404')]
```

```
In [35]: %%sql
         CREATE TABLE user_dimension (
           user_dim_id int NOT NULL AUTO_INCREMENT,
           user_age_range VARCHAR(10),
           user_gender VARCHAR(6),
           user_occupation VARCHAR(30),
           user_ZIP VARCHAR(20),
           PRIMARY KEY(user_dim_id)
           )
```

```
0 rows affected.
```

```
Out[35]: []
```

```
In [36]: %%sql
         INSERT INTO user_dimension (user_age_range, user_gender, user_occupation, user_ZIP)
         SELECT DISTINCT user_age_range, user_gender, user_occupation, user_ZIP
         FROM ratings_temp
```

```
5796 rows affected.
```

```
Out[36]: []
```

```
In [37]: %%sql
         SELECT * FROM user_dimension
         LIMIT 10
```

```
10 rows affected.
```

```
Out[37]: [(1L, 'Under 18', 'MALE', 'other or not specified', '75070'),
          (2L, 'Under 18', 'MALE', 'other or not specified', '48133'),
          (3L, 'Under 18', 'FEMALE', 'other or not specified', '90210'),
          (4L, 'Under 18', 'MALE', 'other or not specified', '97222'),
          (5L, 'Under 18', 'MALE', 'other or not specified', '75244'),
          (6L, 'Under 18', 'MALE', 'other or not specified', '08055'),
          (7L, 'Under 18', 'MALE', 'other or not specified', '11803'),
          (8L, 'Under 18', 'MALE', 'other or not specified', '77479'),
          (9L, 'Under 18', 'MALE', 'other or not specified', '94525'),
          (10L, 'Under 18', 'FEMALE', 'other or not specified', '95404')]
```

Now we will create a table (ratings) to permanently store the ratings.
The first column will be a unique id (rating_id).
The next column will be the user_dim_id from the user_dimension table.
The remaining columns relating to genre and the rating score itself will
come from the ratings_temp table.

The rating_id will be a primary key that auto-increments as records are
inserted. User_dim_id is a foreign key that references the user_dim_id
in the user_dimension table.

```
In [38]: %%sql
         CREATE TABLE ratings (
           rating_id int NOT NULL AUTO_INCREMENT,
           user_dim_id int,
           genre_action VARCHAR(10),
           genre_adventure VARCHAR(10),
           genre_animation VARCHAR(10),
           genre_children VARCHAR(10),
           genre_comedy VARCHAR(10),
           genre_crime VARCHAR(10),
           genre_documentary VARCHAR(10),
           genre_drama VARCHAR(10),
           genre_fantasy VARCHAR(10),
           genre_filmnoir VARCHAR(10),
           genre_horror VARCHAR(10),
           genre_musical VARCHAR(10),
           genre_mystery VARCHAR(10),
           genre_romance VARCHAR(10),
           genre_scifi VARCHAR(10),
           genre_thriller VARCHAR(10),
           genre_war VARCHAR(10),
           genre_western VARCHAR(10),
           rating int,
           PRIMARY KEY (rating_id),
           FOREIGN KEY (user_dim_id) REFERENCES user_dimension(user_dim_id)
           )

0 rows affected.

Out[38]: []
```

We will create an index on the four user dimension columns because we
will need to join the ratings_temp table to the user_dimension table
on these four attributes in order to perform the insert into the ratings table.
This will speed up the insert process.

```
In [39]: %%sql
         CREATE INDEX idx_user_dim ON user_dimension (user_age_range, user_gender, user_occupation, use

0 rows affected.

Out[39]: []

In [40]: %%sql
         EXPLAIN SELECT * FROM user_dimension

1 rows affected.
```

```
Out[40]: [(1L, 'SIMPLE', 'user_dimension', 'index', None, 'idx_user_dim', '78', None, 5973L, 'Using inde
```

```
In [41]: %%sql
         INSERT INTO ratings (user_dim_id,genre_action,genre_adventure,genre_animation,
                              genre_children,genre_comedy,genre_crime,genre_documentary,
                              genre_drama,genre_fantasy,genre_filmnoir,genre_horror,
                              genre_musical,genre_mystery,genre_romance,genre_scifi,
                              genre_thriller,genre_war,genre_western,rating)
         SELECT u.user_dim_id,
             r.genre_action,
             r.genre_adventure,
             r.genre_animation,
             r.genre_children,
             r.genre_comedy,
             r.genre_crime,
             r.genre_documentary,
             r.genre_drama,
             r.genre_fantasy,
             r.genre_filmnoir,
             r.genre_horror,
             r.genre_musical,
             r.genre_mystery,
             r.genre_romance,
             r.genre_scifi,
             r.genre_thriller,
             r.genre_war,
             r.genre_western,
             r.rating
         FROM user_dimension u
         INNER JOIN ratings_temp r on u.user_age_range=r.user_age_range
             AND u.user_gender=r.user_gender
             AND u.user_occupation=r.user_occupation
             AND u.user_ZIP=r.user_ZIP

1000209 rows affected.
```

```
Out[41]: []
```

```
In [42]: %%sql
         SELECT * FROM ratings
         LIMIT 10

10 rows affected.
```

```
Out[42]: [(1L, 1L, 'TRUE', 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'FALS
          (2L, 1L, 'FALSE', 'FALSE', 'TRUE', 'TRUE', 'TRUE', 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'FALSE
          (3L, 1L, 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'TRUE', 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'FALS
          (4L, 1L, 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'TRUE', 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'FALS
          (5L, 1L, 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'TRUE', 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'FALS
          (6L, 1L, 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'TRUE', 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'FALS
          (7L, 1L, 'FALSE', 'FALSE', 'TRUE', 'FALSE', 'TRUE', 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'FALSE
          (8L, 1L, 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'TRUE', 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'FALS
          (9L, 1L, 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'TRUE', 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'FALS
          (10L, 1L, 'FALSE', 'FALSE', 'TRUE', 'FALSE', 'TRUE', 'FALSE', 'FALSE', 'FALSE', 'FALSE', 'FALS
```

We no longer need the ratings_temp table, so it can be dropped.

```
In [43]: %%sql
         DROP TABLE ratings_temp;
```

```
0 rows affected.
```

```
Out[43]: []
```

There is redundancy in the genre classifications of the ratings table. We can normalize this by creating a new table to store the genre values. There will be a single row for each unique genre associated with each rating_id. For example, if rating_id 1 is both Action and Sci-fi, the genre table will contain two rows: (1, Action) and (1, Sci-fi). We will use a loop to iterate over each rating_id, using conditional statements to insert as many rows in the genre table as needed.

The primary key will be a composite key of rating_id and genre. This will ensure that there are no duplicate entries in the table. The rating_id is a foreign key which references the rating_id in the ratings table.

```
In [44]: %%sql
         CREATE TABLE genre (
           rating_id int,
           genre VARCHAR(20),
           CONSTRAINT genrekey PRIMARY KEY (rating_id,genre),
           FOREIGN KEY (rating_id) REFERENCES ratings (rating_id)
           )
```

```
0 rows affected.
```

```
Out[44]: []
```

```
In [45]: import MySQLdb as myDB
         import numpy as np
         conn = myDB.connect('localhost', 'mysqluser', 'mysqlpass')
         cursor = conn.cursor()
         qry = 'USE movielensdb;'
         cursor.execute(qry)
         qry = 'SET autocommit=1;'
         cursor.execute(qry)
         qry = 'SELECT rating_id, genre_action,genre_adventure,genre_animation,genre_children,genre_com
             genre_documentary,genre_drama,genre_fantasy,genre_filmnoir,genre_horror,genre_musical,genr
             genre_romance,genre_scifi,genre_thriller,genre_war,genre_western FROM ratings'
         cursor.execute(qry)

         for r in cursor:
             if r[1] == "TRUE":
                 qry = "Insert into genre values ("+str(r[0])+",'Action')"
                 cursor.execute(qry)
             if r[2] == "TRUE":
                 qry = "Insert into genre values ("+str(r[0])+",'Adventure')"
                 cursor.execute(qry)
             if r[3] == "TRUE":
                 qry = "Insert into genre values ("+str(r[0])+",'Animation')"
                 cursor.execute(qry)
             if r[4] == "TRUE":
                 qry = "Insert into genre values ("+str(r[0])+",'Children')"
                 cursor.execute(qry)
```

```
            if r[5] == "TRUE":
                qry = "Insert into genre values ("+str(r[0])+",'Comedy')"
                cursor.execute(qry)
            if r[6] == "TRUE":
                qry = "Insert into genre values ("+str(r[0])+",'Crime')"
                cursor.execute(qry)
            if r[7] == "TRUE":
                qry = "Insert into genre values ("+str(r[0])+",'Documentary')"
                cursor.execute(qry)
            if r[8] == "TRUE":
                qry = "Insert into genre values ("+str(r[0])+",'Drama')"
                cursor.execute(qry)
            if r[9] == "TRUE":
                qry = "Insert into genre values ("+str(r[0])+",'Fantasy')"
                cursor.execute(qry)
            if r[10] == "TRUE":
                qry = "Insert into genre values ("+str(r[0])+",'Film-Noir')"
                cursor.execute(qry)
            if r[11] == "TRUE":
                qry = "Insert into genre values ("+str(r[0])+",'Horror')"
                cursor.execute(qry)
            if r[12] == "TRUE":
                qry = "Insert into genre values ("+str(r[0])+",'Musical')"
                cursor.execute(qry)
            if r[13] == "TRUE":
                qry = "Insert into genre values ("+str(r[0])+",'Mystery')"
                cursor.execute(qry)
            if r[14] == "TRUE":
                qry = "Insert into genre values ("+str(r[0])+",'Romance')"
                cursor.execute(qry)
            if r[15] == "TRUE":
                qry = "Insert into genre values ("+str(r[0])+",'Sci-Fi')"
                cursor.execute(qry)
            if r[16] == "TRUE":
                qry = "Insert into genre values ("+str(r[0])+",'Thriller')"
                cursor.execute(qry)
            if r[17] == "TRUE":
                qry = "Insert into genre values ("+str(r[0])+",'War')"
                cursor.execute(qry)
            if r[18] == "TRUE":
                qry = "Insert into genre values ("+str(r[0])+",'Western')"
                cursor.execute(qry)
        cursor.close()
```

```
In [46]: %%sql
         SELECT * FROM genre
         LIMIT 10

10 rows affected.

Out[46]: [(1L, 'Action'),
          (1L, 'Sci-Fi'),
          (2L, 'Animation'),
          (2L, 'Children'),
          (2L, 'Comedy'),
```

```
 (3L, 'Comedy'),
 (4L, 'Comedy'),
 (5L, 'Comedy'),
 (5L, 'Romance'),
 (6L, 'Comedy')]
```

We no longer need the genre classifications to be stored in the ratings table. Therefore, we drop these columns.

```
In [47]: %%sql
         ALTER TABLE ratings
         DROP genre_action, DROP genre_adventure, DROP genre_animation, DROP genre_children, DROP genre
         DROP genre_crime, DROP genre_documentary, DROP genre_drama, DROP genre_fantasy, DROP genre_fil
         DROP genre_horror, DROP genre_musical, DROP genre_mystery, DROP genre_romance, DROP genre_scif
         DROP genre_thriller, DROP genre_war, DROP genre_western

1000209 rows affected.

Out[47]: []
```

The final normalized ratings table contains only 3 columns: rating_id, user_dim_id, and rating.

```
In [48]: %%sql
         SELECT * from ratings
         LIMIT 10

10 rows affected.

Out[48]: [(1L, 1L, 3L),
          (2L, 1L, 4L),
          (3L, 1L, 1L),
          (4L, 1L, 3L),
          (5L, 1L, 3L),
          (6L, 1L, 4L),
          (7L, 1L, 4L),
          (8L, 1L, 5L),
          (9L, 1L, 5L),
          (10L, 1L, 4L)]
```

Since we will need to join the ratings table with the genre table in future queries, we create an index on the rating_id field.

```
In [49]: %%sql
         CREATE INDEX idx_rating_id ON genre (rating_id)

0 rows affected.

Out[49]: []
```

Since we will need to join the ratings table with the user_dimension table in future queries, we create an index on the user_dim_id field.

```
In [50]: %%sql
         CREATE INDEX idx_user_dim_id ON user_dimension (user_dim_id)

0 rows affected.
```

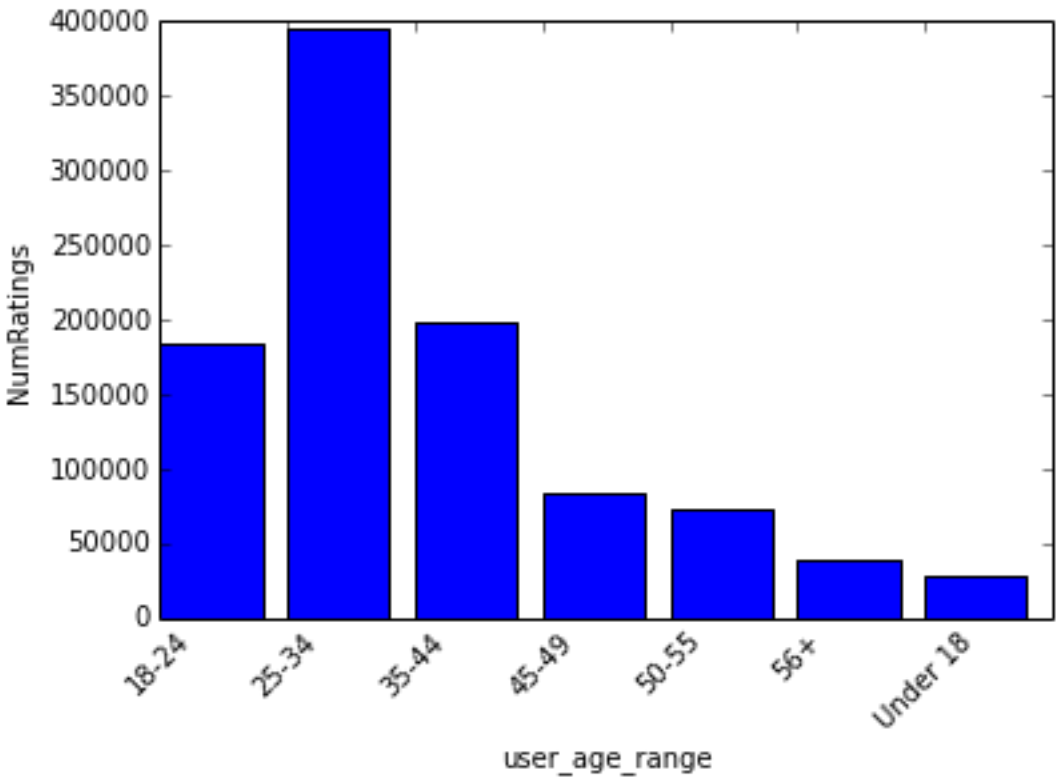
```
Out[50]: []
```

First we will check to make sure that all the rows have been properly imported from the file. Note that the count of records from the csv file will be 1 more than the records in the ratings able because of the header.

```
In [51]: %%sql
         SELECT count(*)
         FROM ratings

1 rows affected.

Out[51]: [(1000209L,)]

In [52]: %%bash
         wc -l MovieLens.csv

1000210 MovieLens.csv
```

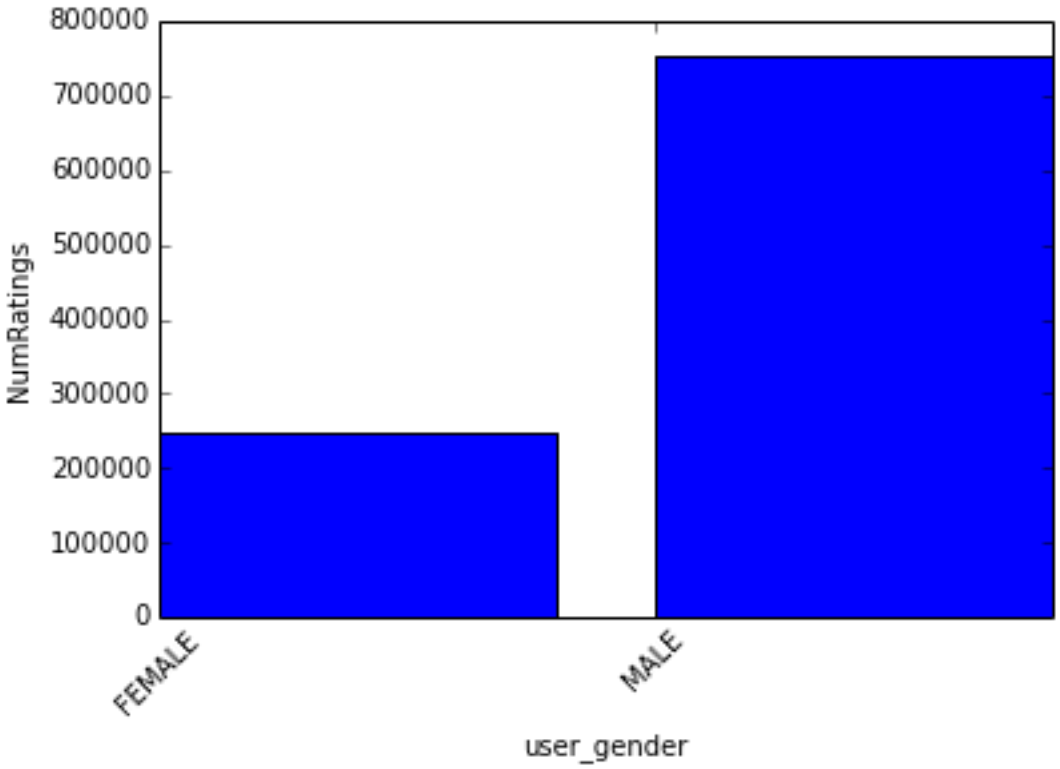
Now we want to run some queries to pull descriptive statistics about the users who provided the ratings. The first query is to count the number of ratings provided by users in each age range. Note that the dataset did not include a unique user identifier. One user may have provided more than one rating. Therefore, we do not know how many distinct users fall into each category.

```
In [53]: %matplotlib inline

In [54]: %%sql
         SELECT user_age_range, count(*) as NumRatings
         FROM ratings r join user_dimension u
         on r.user_dim_id = u.user_dim_id
         group by user_age_range

7 rows affected.

Out[54]: [('18-24', 183536L),
          ('25-34', 395556L),
          ('35-44', 199003L),
          ('45-49', 83633L),
          ('50-55', 72490L),
          ('56+', 38780L),
          ('Under 18', 27211L)]

In [55]: result = _
         result.bar()

Out[55]: <Container object of 7 artists>
```



The following query determines the number of ratings for each gender.

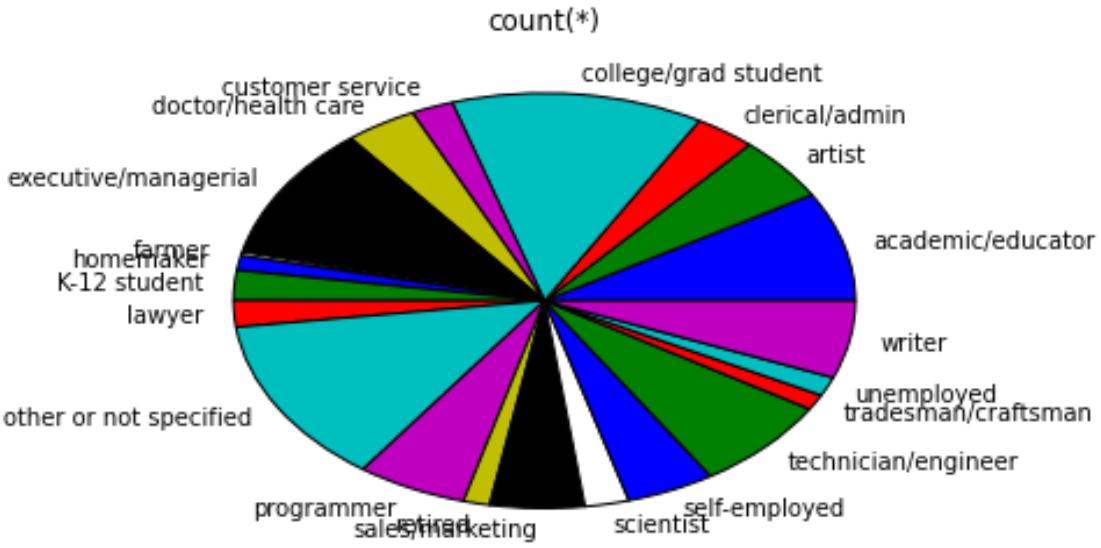
```
In [56]: %%sql
         SELECT user_gender, count(*) as NumRatings
         FROM ratings r join user_dimension u
         on r.user_dim_id = u.user_dim_id
         group by user_gender

2 rows affected.

Out[56]: [('FEMALE', 246440L), ('MALE', 753769L)]

In [57]: result = _
         result.bar()

Out[57]: <Container object of 2 artists>
```



The followng query gives the number of ratings associated with each user occupation.

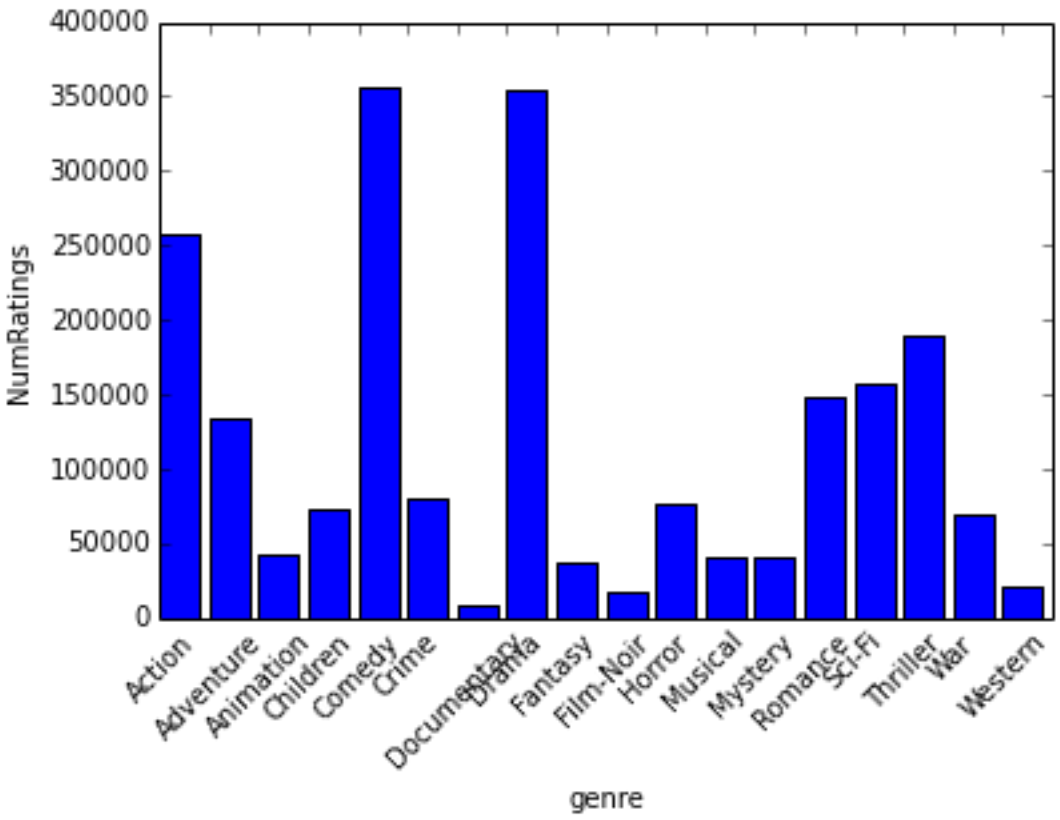
```
In [58]: %%sql
         SELECT user_occupation, count(*)
         FROM ratings r join user_dimension u
         on r.user_dim_id = u.user_dim_id
         group by user_occupation

21 rows affected.

Out[58]: [('academic/educator', 85351L),
          ('artist', 50068L),
          ('clerical/admin', 31623L),
          ('college/grad student', 131032L),
          ('customer service', 21850L),
          ('doctor/health care', 37205L),
          ('executive/managerial', 105425L),
          ('farmer', 2706L),
          ('homemaker', 11345L),
          ('K-12 student', 23290L),
          ('lawyer', 20563L),
          ('other or not specified', 130499L),
          ('programmer', 57214L),
          ('retired', 13754L),
          ('sales/marketing', 49109L),
          ('scientist', 22951L),
          ('self-employed', 46021L),
```

```
          ('technician/engineer', 72816L),
          ('tradesman/craftsman', 12086L),
          ('unemployed', 14904L),
          ('writer', 60397L)]

In [59]: result = _
         result.pie()

Out[59]: ([<matplotlib.patches.Wedge at 0x7ff920283b50>,
           <matplotlib.patches.Wedge at 0x7ff920291590>,
           <matplotlib.patches.Wedge at 0x7ff920291f10>,
           <matplotlib.patches.Wedge at 0x7ff92029c8d0>,
           <matplotlib.patches.Wedge at 0x7ff9202a9310>,
           <matplotlib.patches.Wedge at 0x7ff9202a9c90>,
           <matplotlib.patches.Wedge at 0x7ff920236650>,
           <matplotlib.patches.Wedge at 0x7ff920243090>,
           <matplotlib.patches.Wedge at 0x7ff920243a10>,
           <matplotlib.patches.Wedge at 0x7ff9202503d0>,
           <matplotlib.patches.Wedge at 0x7ff920250d50>,
           <matplotlib.patches.Wedge at 0x7ff92025d710>,
           <matplotlib.patches.Wedge at 0x7ff92026a150>,
           <matplotlib.patches.Wedge at 0x7ff92026aad0>,
           <matplotlib.patches.Wedge at 0x7ff9201f6490>,
           <matplotlib.patches.Wedge at 0x7ff9201f6e10>,
           <matplotlib.patches.Wedge at 0x7ff9202047d0>,
           <matplotlib.patches.Wedge at 0x7ff920211190>,
           <matplotlib.patches.Wedge at 0x7ff920211b10>,
           <matplotlib.patches.Wedge at 0x7ff92021d4d0>,
           <matplotlib.patches.Wedge at 0x7ff92021de50>],
          [<matplotlib.text.Text at 0x7ff920291150>,
           <matplotlib.text.Text at 0x7ff920291b50>,
           <matplotlib.text.Text at 0x7ff92029c510>,
           <matplotlib.text.Text at 0x7ff92029cdd0>,
           <matplotlib.text.Text at 0x7ff9202a98d0>,
           <matplotlib.text.Text at 0x7ff920236290>,
           <matplotlib.text.Text at 0x7ff920236b50>,
           <matplotlib.text.Text at 0x7ff920243650>,
           <matplotlib.text.Text at 0x7ff920243fd0>,
           <matplotlib.text.Text at 0x7ff920250990>,
           <matplotlib.text.Text at 0x7ff92025d350>,
           <matplotlib.text.Text at 0x7ff92025dc10>,
           <matplotlib.text.Text at 0x7ff92026a710>,
           <matplotlib.text.Text at 0x7ff92026afd0>,
           <matplotlib.text.Text at 0x7ff9201f6a50>,
           <matplotlib.text.Text at 0x7ff920204410>,
           <matplotlib.text.Text at 0x7ff920204d90>,
           <matplotlib.text.Text at 0x7ff920211750>,
           <matplotlib.text.Text at 0x7ff92021d110>,
           <matplotlib.text.Text at 0x7ff92021da90>,
           <matplotlib.text.Text at 0x7ff920229450>])
```



The next query gives the number of ratings for each genre. Note that the total of this will be greater than the total number of ratings because one movie may fall into multiple genres.

```
In [60]: %%sql
         SELECT genre, count(distinct rating_id) as NumRatings
         FROM genre
         group by genre

18 rows affected.

Out[60]: [('Action', 257457L),
          ('Adventure', 133953L),
          ('Animation', 43293L),
          ('Children', 72186L),
          ('Comedy', 356580L),
          ('Crime', 79541L),
          ('Documentary', 7910L),
          ('Drama', 354529L),
          ('Fantasy', 36301L),
          ('Film-Noir', 18261L),
          ('Horror', 76386L),
          ('Musical', 41533L),
          ('Mystery', 40178L),
          ('Romance', 147523L),
          ('Sci-Fi', 157294L),
          ('Thriller', 189680L),
          ('War', 68527L),
          ('Western', 20683L)]

In [61]: result = _
         result.bar()

Out[61]: <Container object of 18 artists>
```



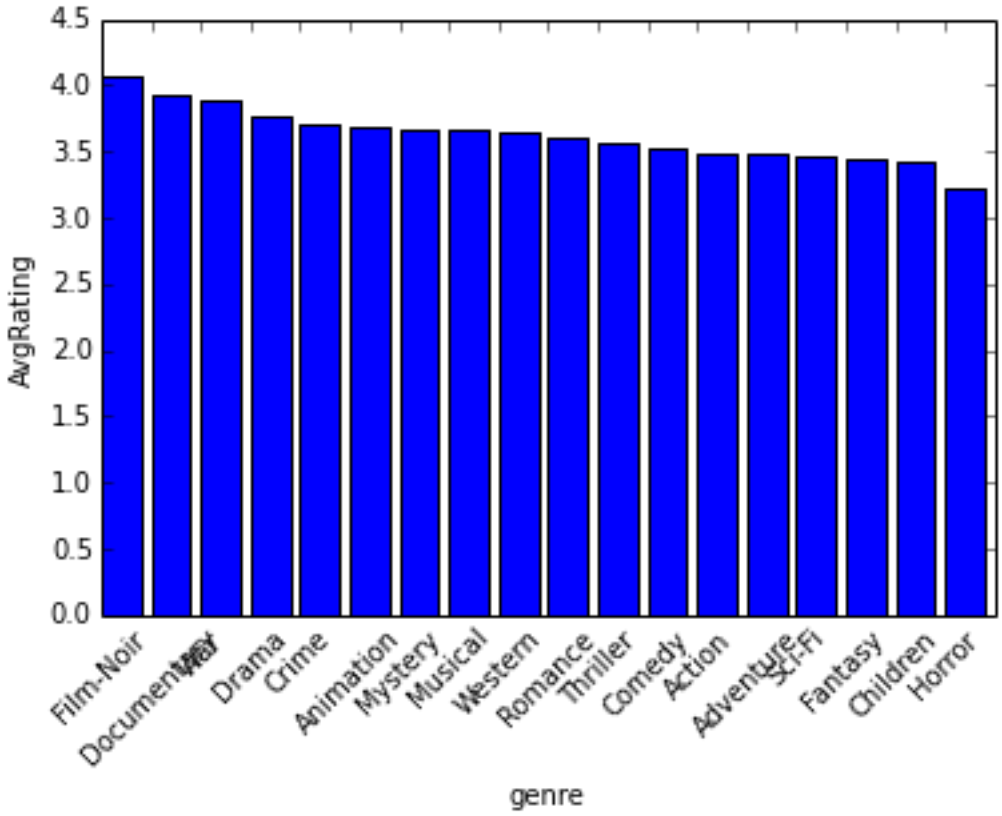
The following query gives the average ratings for each genre.

```
In [62]: %%sql
         SELECT genre, avg(rating) as AvgRating
         FROM genre g join ratings r
         on g.rating_id = r.rating_id
         group by genre
         order by AvgRating desc

18 rows affected.

Out[62]: [('Film-Noir', Decimal('4.0752')),
          ('Documentary', Decimal('3.9331')),
          ('War', Decimal('3.8933')),
          ('Drama', Decimal('3.7663')),
          ('Crime', Decimal('3.7087')),
          ('Animation', Decimal('3.6849')),
          ('Mystery', Decimal('3.6681')),
          ('Musical', Decimal('3.6655')),
          ('Western', Decimal('3.6378')),
          ('Romance', Decimal('3.6075')),
          ('Thriller', Decimal('3.5705')),
          ('Comedy', Decimal('3.5221')),
          ('Action', Decimal('3.4912')),
          ('Adventure', Decimal('3.4773')),
          ('Sci-Fi', Decimal('3.4665')),
```

```
          ('Fantasy', Decimal('3.4474')),
          ('Children', Decimal('3.4220')),
          ('Horror', Decimal('3.2150'))]
```

```
In [63]: result = _
         result.bar()
```

```
Out[63]: <Container object of 18 artists>
```



The following query gives the average rating by genre and user_age_range. We can use this to see which genres are preferred by each age group.

```
In [64]: %%sql
         SELECT user_age_range, genre, avg(rating) as AvgRating
         FROM genre g join ratings r
         on g.rating_id = r.rating_id
         join user_dimension u
         on u.user_dim_id = r.user_dim_id
         group by genre, u.user_age_range
         order by user_age_range, AvgRating desc
```

```
126 rows affected.
```

```
Out[64]: [('18-24', 'Film-Noir', Decimal('3.9974')),
          ('18-24', 'Documentary', Decimal('3.8659')),
          ('18-24', 'War', Decimal('3.8531')),
```

```
('18-24', 'Drama', Decimal('3.7219')),
('18-24', 'Crime', Decimal('3.6681')),
('18-24', 'Animation', Decimal('3.6240')),
('18-24', 'Musical', Decimal('3.5603')),
('18-24', 'Romance', Decimal('3.5343')),
('18-24', 'Mystery', Decimal('3.5254')),
('18-24', 'Thriller', Decimal('3.4946')),
('18-24', 'Western', Decimal('3.4715')),
('18-24', 'Comedy', Decimal('3.4604')),
('18-24', 'Action', Decimal('3.4471')),
('18-24', 'Sci-Fi', Decimal('3.4261')),
('18-24', 'Adventure', Decimal('3.4085')),
('18-24', 'Fantasy', Decimal('3.3538')),
('18-24', 'Children', Decimal('3.2943')),
('18-24', 'Horror', Decimal('3.1727')),
('25-34', 'Film-Noir', Decimal('4.0587')),
('25-34', 'Documentary', Decimal('3.9467')),
('25-34', 'War', Decimal('3.8412')),
('25-34', 'Drama', Decimal('3.7264')),
('25-34', 'Animation', Decimal('3.7012')),
('25-34', 'Crime', Decimal('3.6803')),
('25-34', 'Musical', Decimal('3.6199')),
('25-34', 'Mystery', Decimal('3.6108')),
('25-34', 'Western', Decimal('3.6078')),
('25-34', 'Romance', Decimal('3.5467')),
('25-34', 'Thriller', Decimal('3.5355')),
('25-34', 'Comedy', Decimal('3.4904')),
('25-34', 'Action', Decimal('3.4534')),
('25-34', 'Fantasy', Decimal('3.4525')),
('25-34', 'Sci-Fi', Decimal('3.4438')),
('25-34', 'Adventure', Decimal('3.4432')),
('25-34', 'Children', Decimal('3.4269')),
('25-34', 'Horror', Decimal('3.2001')),
('35-44', 'Film-Noir', Decimal('4.0649')),
('35-44', 'Documentary', Decimal('3.9537')),
('35-44', 'War', Decimal('3.9011')),
('35-44', 'Drama', Decimal('3.7825')),
('35-44', 'Animation', Decimal('3.7405')),
('35-44', 'Crime', Decimal('3.7337')),
('35-44', 'Musical', Decimal('3.7216')),
('35-44', 'Mystery', Decimal('3.6974')),
('35-44', 'Western', Decimal('3.6793')),
('35-44', 'Romance', Decimal('3.6511')),
('35-44', 'Thriller', Decimal('3.6159')),
('35-44', 'Comedy', Decimal('3.5620')),
('35-44', 'Action', Decimal('3.5381')),
('35-44', 'Children', Decimal('3.5184')),
('35-44', 'Adventure', Decimal('3.5153')),
('35-44', 'Sci-Fi', Decimal('3.5021')),
('35-44', 'Fantasy', Decimal('3.4823')),
('35-44', 'Horror', Decimal('3.2760')),
('45-49', 'Film-Noir', Decimal('4.1054')),
('45-49', 'Documentary', Decimal('3.9665')),
('45-49', 'War', Decimal('3.9606')),
```

```
('45-49', 'Drama', Decimal('3.7844')),
('45-49', 'Mystery', Decimal('3.7543')),
('45-49', 'Crime', Decimal('3.7507')),
('45-49', 'Musical', Decimal('3.7445')),
('45-49', 'Animation', Decimal('3.7349')),
('45-49', 'Romance', Decimal('3.6860')),
('45-49', 'Western', Decimal('3.6671')),
('45-49', 'Thriller', Decimal('3.6397')),
('45-49', 'Comedy', Decimal('3.5918')),
('45-49', 'Fantasy', Decimal('3.5325')),
('45-49', 'Adventure', Decimal('3.5290')),
('45-49', 'Action', Decimal('3.5285')),
('45-49', 'Children', Decimal('3.5276')),
('45-49', 'Sci-Fi', Decimal('3.4825')),
('45-49', 'Horror', Decimal('3.2623')),
('50-55', 'Film-Noir', Decimal('4.1754')),
('50-55', 'War', Decimal('3.9742')),
('50-55', 'Documentary', Decimal('3.9081')),
('50-55', 'Mystery', Decimal('3.8858')),
('50-55', 'Drama', Decimal('3.8784')),
('50-55', 'Crime', Decimal('3.8107')),
('50-55', 'Musical', Decimal('3.7983')),
('50-55', 'Animation', Decimal('3.7800')),
('50-55', 'Romance', Decimal('3.7581')),
('50-55', 'Western', Decimal('3.7413')),
('50-55', 'Thriller', Decimal('3.7097')),
('50-55', 'Comedy', Decimal('3.6469')),
('50-55', 'Adventure', Decimal('3.6282')),
('50-55', 'Action', Decimal('3.6113')),
('50-55', 'Fantasy', Decimal('3.5816')),
('50-55', 'Sci-Fi', Decimal('3.5645')),
('50-55', 'Children', Decimal('3.5566')),
('50-55', 'Horror', Decimal('3.1589')),
('56+', 'Film-Noir', Decimal('4.1259')),
('56+', 'War', Decimal('4.0673')),
('56+', 'Documentary', Decimal('3.9615')),
('56+', 'Drama', Decimal('3.9335')),
('56+', 'Mystery', Decimal('3.8905')),
('56+', 'Musical', Decimal('3.8867')),
('56+', 'Crime', Decimal('3.8325')),
('56+', 'Romance', Decimal('3.8165')),
('56+', 'Western', Decimal('3.7922')),
('56+', 'Animation', Decimal('3.7562')),
('56+', 'Thriller', Decimal('3.7197')),
('56+', 'Comedy', Decimal('3.6509')),
('56+', 'Adventure', Decimal('3.6491')),
('56+', 'Children', Decimal('3.6218')),
('56+', 'Action', Decimal('3.6107')),
('56+', 'Fantasy', Decimal('3.5327')),
('56+', 'Sci-Fi', Decimal('3.4977')),
('56+', 'Horror', Decimal('3.2544')),
('Under 18', 'Film-Noir', Decimal('4.1455')),
('Under 18', 'War', Decimal('3.8954')),
('Under 18', 'Drama', Decimal('3.7947')),
```

```
('Under 18', 'Documentary', Decimal('3.7308')),
('Under 18', 'Crime', Decimal('3.7102')),
('Under 18', 'Mystery', Decimal('3.6315')),
('Under 18', 'Romance', Decimal('3.6213')),
('Under 18', 'Western', Decimal('3.5761')),
('Under 18', 'Musical', Decimal('3.5683')),
('Under 18', 'Thriller', Decimal('3.5504')),
('Under 18', 'Action', Decimal('3.5064')),
('Under 18', 'Comedy', Decimal('3.4975')),
('Under 18', 'Sci-Fi', Decimal('3.4787')),
('Under 18', 'Animation', Decimal('3.4761')),
('Under 18', 'Adventure', Decimal('3.4500')),
('Under 18', 'Fantasy', Decimal('3.3176')),
('Under 18', 'Horror', Decimal('3.2542')),
('Under 18', 'Children', Decimal('3.2416'))]
```

The next query describes the average ratings for each genre by gender.

```
In [65]: %%sql
         SELECT user_gender, genre, avg(rating) as AvgRating
         FROM genre g join ratings r
         on g.rating_id = r.rating_id
         join user_dimension u
         on u.user_dim_id = r.user_dim_id
         group by genre, u.user_gender
         order by user_gender, AvgRating desc

36 rows affected.

Out[65]: [('FEMALE', 'Film-Noir', Decimal('4.0181')),
          ('FEMALE', 'Documentary', Decimal('3.9464')),
          ('FEMALE', 'War', Decimal('3.8931')),
          ('FEMALE', 'Musical', Decimal('3.8091')),
          ('FEMALE', 'Drama', Decimal('3.7657')),
          ('FEMALE', 'Animation', Decimal('3.7447')),
          ('FEMALE', 'Crime', Decimal('3.6893')),
          ('FEMALE', 'Mystery', Decimal('3.6865')),
          ('FEMALE', 'Romance', Decimal('3.6736')),
          ('FEMALE', 'Thriller', Decimal('3.5734')),
          ('FEMALE', 'Children', Decimal('3.5725')),
          ('FEMALE', 'Comedy', Decimal('3.5719')),
          ('FEMALE', 'Western', Decimal('3.5519')),
          ('FEMALE', 'Fantasy', Decimal('3.5131')),
          ('FEMALE', 'Adventure', Decimal('3.5129')),
          ('FEMALE', 'Action', Decimal('3.4903')),
          ('FEMALE', 'Sci-Fi', Decimal('3.4503')),
          ('FEMALE', 'Horror', Decimal('3.2029')),
          ('MALE', 'Film-Noir', Decimal('4.0923')),
          ('MALE', 'Documentary', Decimal('3.9288')),
          ('MALE', 'War', Decimal('3.8934')),
          ('MALE', 'Drama', Decimal('3.7666')),
          ('MALE', 'Crime', Decimal('3.7137')),
          ('MALE', 'Mystery', Decimal('3.6620')),
          ('MALE', 'Animation', Decimal('3.6613')),
          ('MALE', 'Western', Decimal('3.6551')),
```

```
('MALE', 'Musical', Decimal('3.5963')),
('MALE', 'Romance', Decimal('3.5733')),
('MALE', 'Thriller', Decimal('3.5697')),
('MALE', 'Comedy', Decimal('3.5037')),
('MALE', 'Action', Decimal('3.4914')),
('MALE', 'Sci-Fi', Decimal('3.4700')),
('MALE', 'Adventure', Decimal('3.4681')),
('MALE', 'Fantasy', Decimal('3.4266')),
('MALE', 'Children', Decimal('3.3590')),
('MALE', 'Horror', Decimal('3.2179'))]
```

In the future, it would be interesting to look more deeply at how movie preferences differ across various user attributes. This could help us determine overlap between certain preferences, i.e. do users who enjoy one type of movie typically enjoy another type as well.