CSCI620 - Introduction to Big Data

Project Phase III

Group 1

Topic: MyAnimeList Data Analysis

Sources:
- Kaggle: https://www.kaggle.com/datasets/azathoth42/myanimelist
- MyAnimeList: https://myanimelist.net/

Authors – Athina Stewart (as1986)

Archit Joshi (aj6082)

Chengzi Cao (cc3773)

Parijat Kawale (pk7145)

1. Data Cleaning and Data Integration

Note –

1. Script cleanData.sql followed by createTables.sql can be reviewed for code related to this section for relational database (PostgreSQL).

2. Script mongodb_datainsert.py can be reviewed for code related to this section for document database model (MongoDB).

Being an anime and manga portal with a global scope, more often than not users will enter falsified information to protect their identity or bypass age restricted content.

The scripts need to be run in order on an empty schema so that data is cleaned while its being loaded the first time.

Noticeable Issues in the Data –

- Completeness -> Some address, gender, location data for users is incomplete
- Uniformity -> Runtime for episodes is in hours as well as minutes.
- Accuracy -> Fake birthdates for user profiles to bypass age restricted content.
- Validity -> Invalid values for usernames and addresses for users.

Examples –

gender	user_location	birth	access_rank
Male	╭∩╮(●'.'●)╭∩╮	<null>	<null>
Male	私立あべにゅう学園	<null>	<null>
Male	532	2012-01-01	<null>
Male	1871	1998-12-01	<null>
Male	034	2000-05-01	<null>
Male	ΛΑΡΙΣΑ	1993-11-03	<null>
<null>	021	<null>	<null>
<null>	139	1991-04-26	<null>
<null>	188	<null>	<null>
Male	Академия Сити, Япония	1996-06-28	<null>
Male	287	2000-08-15	<null>

A brief review of steps for cleaning and integrating the data –

1. TRANSFORMATIONS TO DATA
    - Based on our observations (which led to the relational diagram in phase 1), there were many-to-many relationships among producer-anime, licensor-anime, studio-anime, and genre-anime.
    - As such, these were split into their own tables and multiple column foreign keys were used to relate the licensors, producers, studios, and genres to the anime.

2. FILTERING
    - Removed tuple with null entities in the gender, birth date and location columns

3. CLEANING
    - Truncated all users who have watched more episodes of an anime then actually exists for that anime (Example of incorrect data).
    - Truncated users that were too young or too old. The user birthdates were casted as date type first (Example of invalid data).
    - Users can input their location leading to some inaccurate descriptions or gibberish. One way to clean was to constrain entries to be letters only (Example of invalid data).
    - Truncated users who reportedly spent more days watching anime than possible for the average human lifespan the limit has been set to 50 years (Example of incorrect data).
    - Added primary keys to the user and anime table to prevent having empty rows.
    - Added functional dependencies in all relation tables to prevent having duplicate rows.
    - Empty columns: access rank and related as (rows deleted).
    - Truncated anime that had 0 episodes (invalid data).
    - Truncated users who watched less than 1 episode (invalid data).

2. Statistics from the data

Note – Script stats.py can be used to review the code related to this section. We use matplotlib to plot the data retrieved from postgres database.

1. Average rating across all anime produced by top 20 studios – Bar Graph

Query –

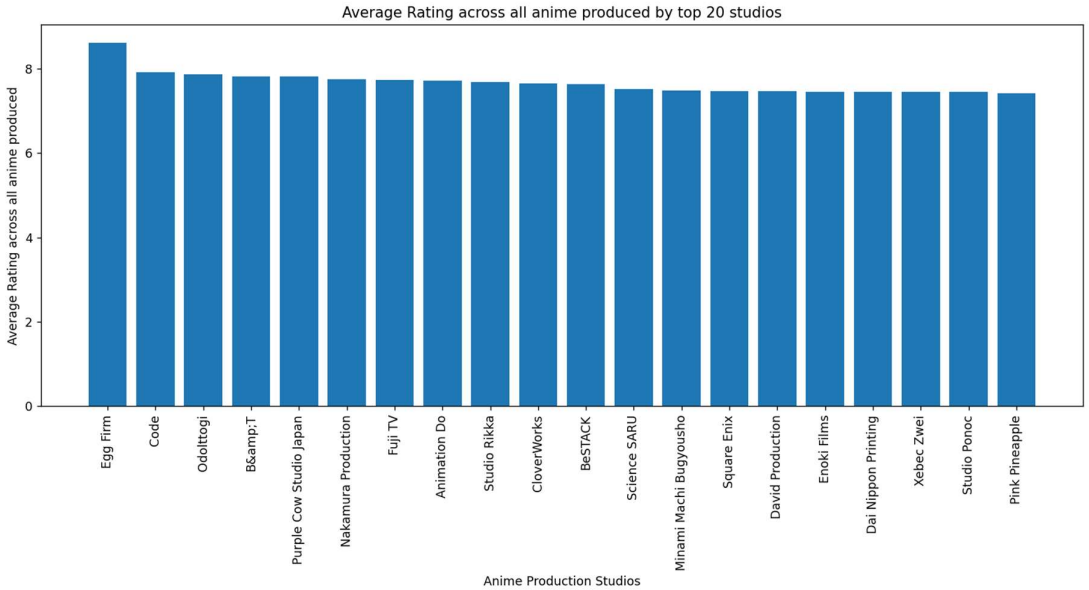
```
select studio.studio , avg(a.score) as average_rating
from studio
   inner join created_by cb on studio.studio_id = cb.studio_id
   inner join anime a on cb.anime_id = a.anime_id
group by studio.studio order by average_rating DESC limit 20

[2023-04-21 11:21:55] 20 rows retrieved starting from 1 in 75 ms (execution: 46 ms, fetching: 29 ms)
```

Sample output data –

	studio	average_rating
1	Egg Firm	8.615
2	Code	7.91
3	Odolttogi	7.87
4	B&T	7.82
5	Purple Cow Studio Japan	7.82
6	Nakamura Production	7.75
7	Fuji TV	7.73
8	Animation Do	7.720000000000001
9	Studio Rikka	7.688571428571428
10	CloverWorks	7.65

Matplotlib Bar graph –



2. Number of anime premiering over the years – Time Series Plot

Query –

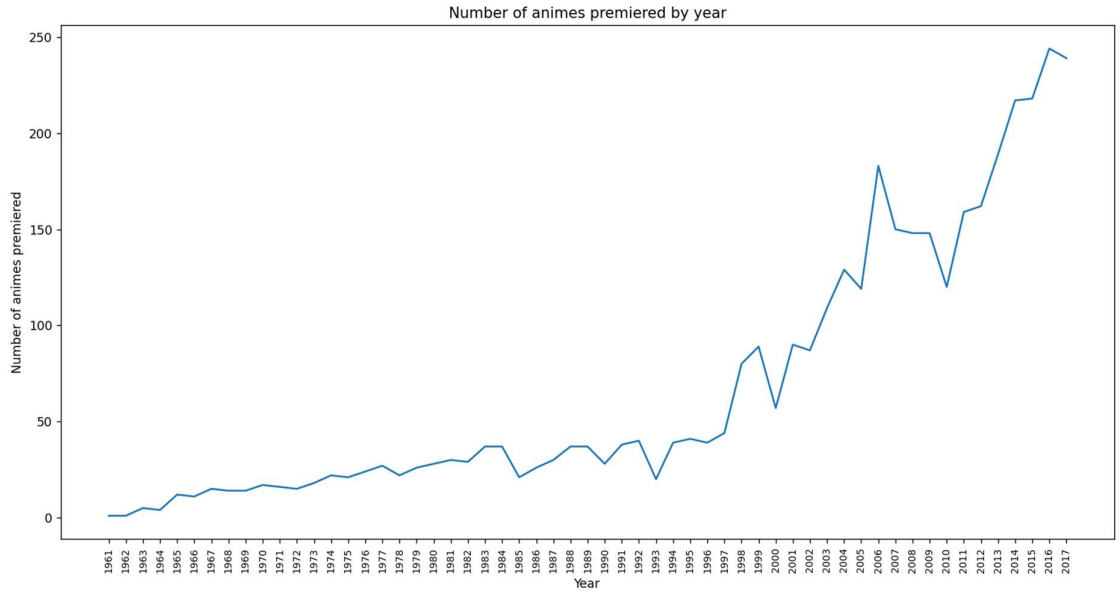
```
select (string_to_array(premiered, ' '))[2] as year,
   count((string_to_array(premiered, ' '))[2]) as count
from anime
where (string_to_array(premiered, ' '))[2] is not null
group by (string_to_array(premiered, ' '))[2] order by (string_to_array(premiered, ' '))[2]

[2023-04-21 11:20:30] 58 rows retrieved starting from 1 in 133 ms (execution: 99 ms, fetching: 34 ms)
```

Sample output data –

	year	count
1	1961	1
2	1962	1
3	1963	5
4	1964	4
5	1965	12
6	1966	11
7	1967	15
8	1968	14
9	1969	14
10	1970	17
11	1971	16
12	1972	15
13	1973	18
14	1974	22
15	1975	21
16	1976	24
17	1977	27
18	1978	22
19	1979	26

Matplotlib Time Series Graph –



3. Distribution of genre over all anime produced – Pie Chart

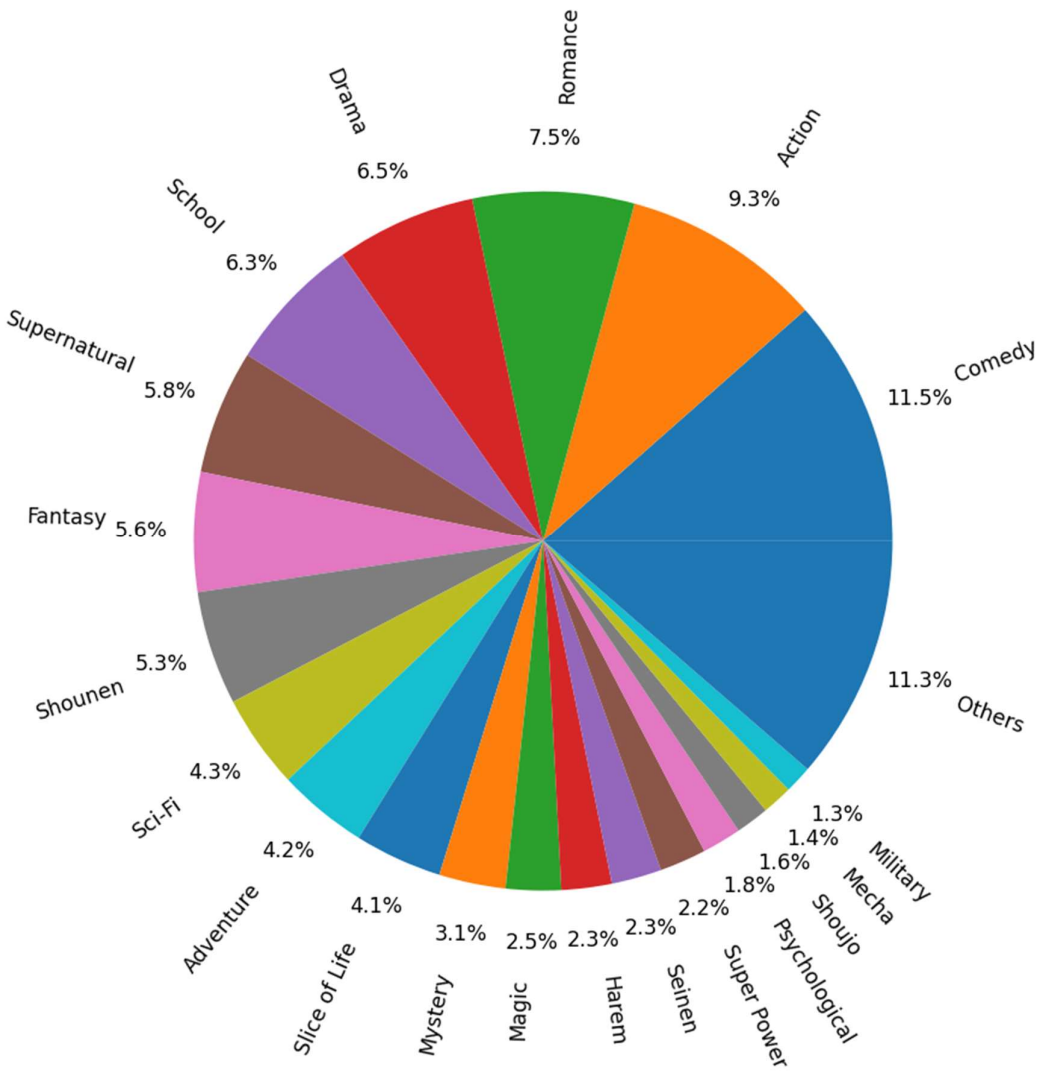
Query –

```
select distinct count(anime.anime_id), genre
from anime
   inner join has_genre hg on anime.anime_id = hg.anime_id
   inner join genre g on hg.genre_id = g.genre_id
   inner join watches w on anime.anime_id = w.anime_id
group by genre
order by count(anime.anime_id) DESC
limit 20;
[2023-04-21 11:18:55] 20 rows retrieved starting from 1 in 1 m 11 s 530 ms (execution: 1 m 11 s 386 ms,
fetching: 144 ms)
```

Sample output data –

	count	genre
1	17181713	Comedy
2	14066985	Action
3	10519072	Romance
4	9854302	Drama
5	8765799	Fantasy
6	8432972	School
7	8356628	Supernatural
8	7973355	Shounen
9	7163702	Sci-Fi
10	7157839	Adventure
11	5503692	Slice of Life
12	4240864	Mystery
13	3637143	Magic
14	3176326	Seinen
15	3174562	Super Power
16	2931420	Harem
17	2558932	Psychological
18	2458911	Mecha
19	2127385	Military
20	2106467	Shoujo

Matplotlib Pie Chart –



3. Item Set Mining

Note –

1. Script monogodb_itemsetmining.py can be used to review code for this section for Document database model (MongoDB).

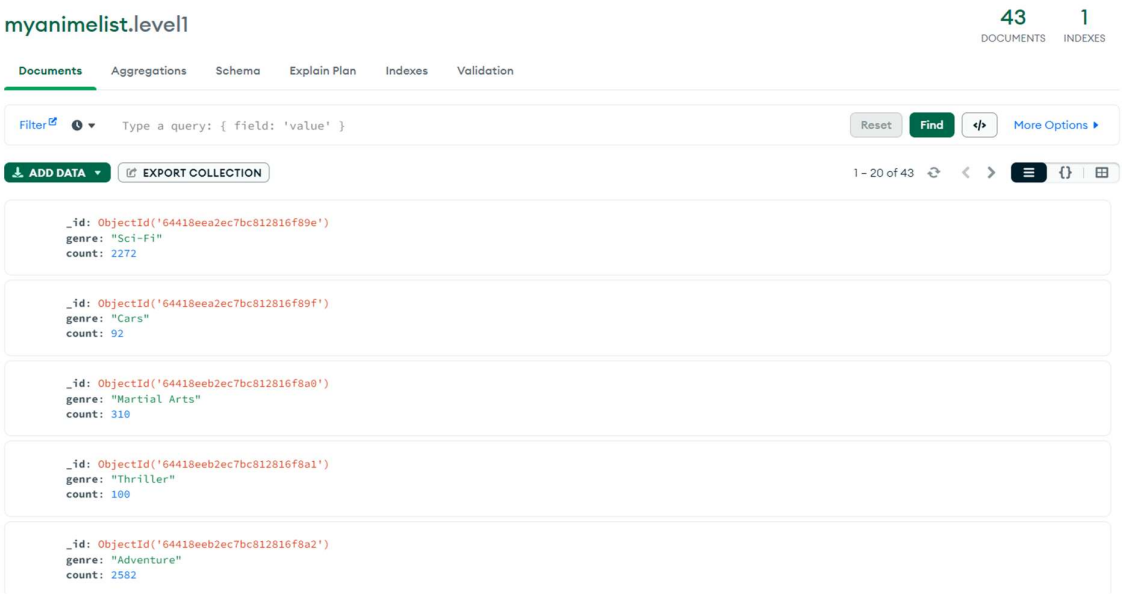
2. Script postgres_itemsetmining.py can be used to review code for this section for Relational database model (PostgreSQL).

For the dataset chosen item mining using apriori algorithm can help us determine which genres can be paired together more often. This understanding can eventually help us predict user viewing patterns and suggesting genres depending on what has already been consumed before based on watch history.

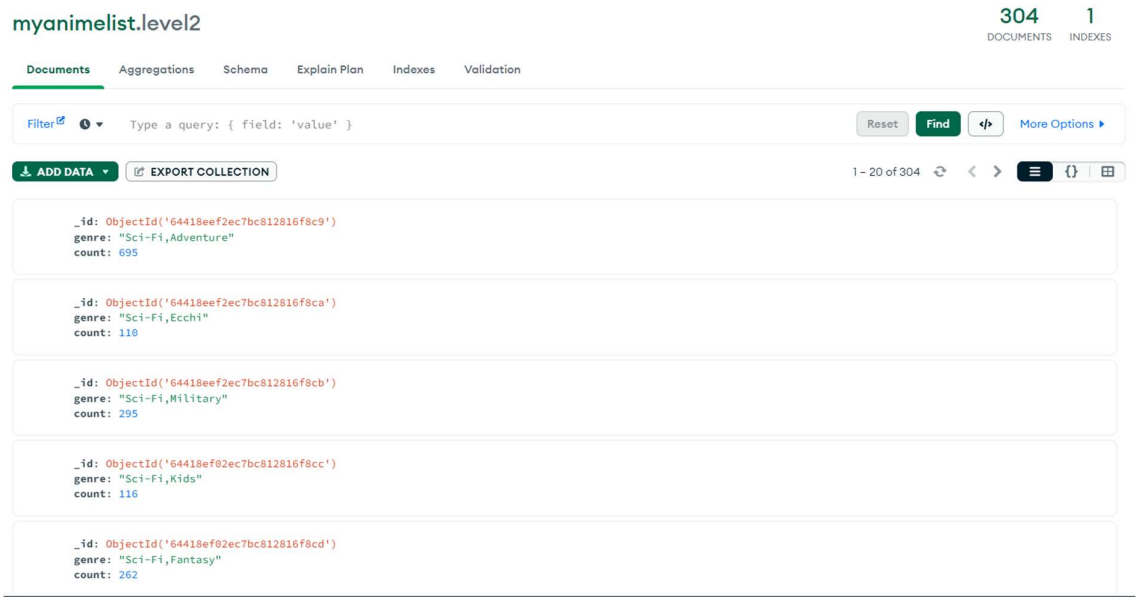
Problem Statement: You are an anime creator and want to know which are most popular genres of animes that go together.

- This issue can be solved using apriori algorithm and discover association rules between the genres.
- In the program, we start with level one of association rule discovery, where we get the individual genre count for all animes across the document database model.
- The minimum support count set to 100 in order filter those combinations that have less chance of occurrence together.

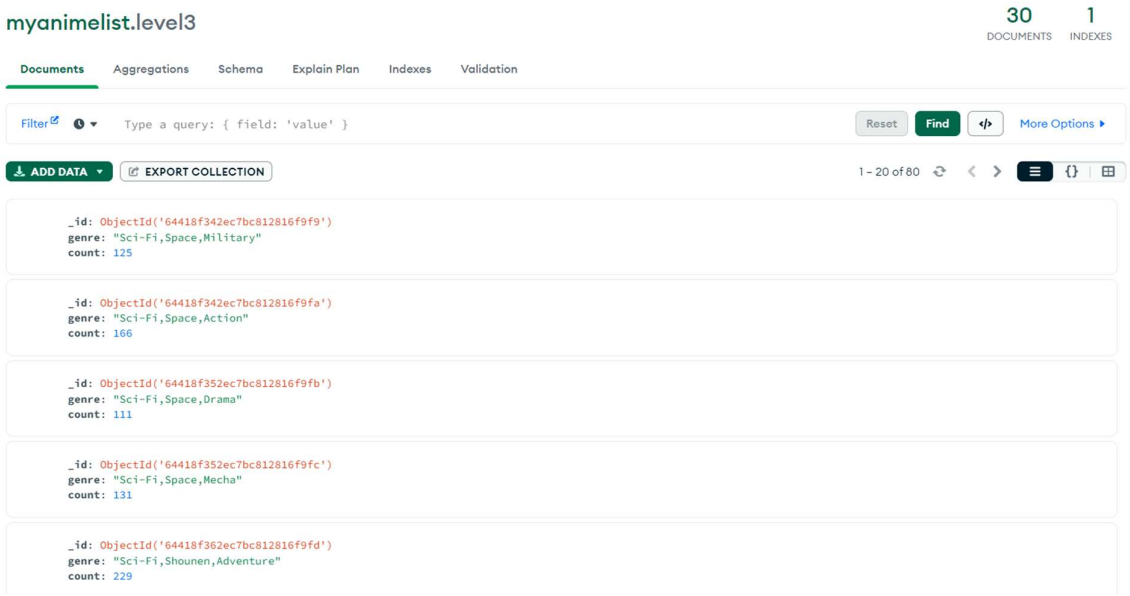
- **Level 1:**



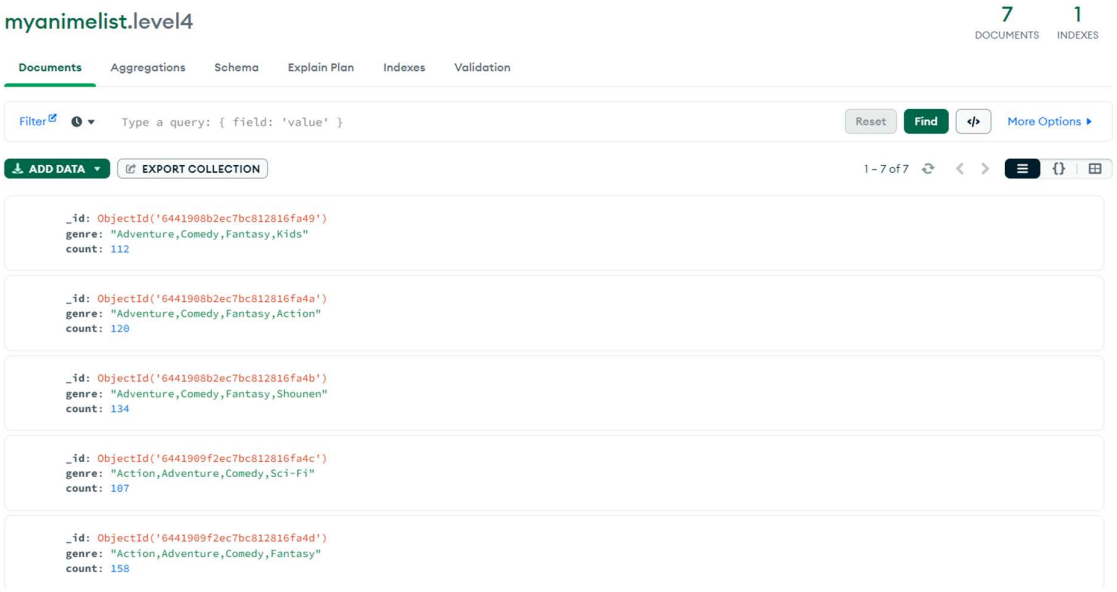
- **Level 2:**



- **Level 3**



- **Level 4**



Tabulated Output –

Genre 1	Genre 2	Genre 3	Genre 4
Adventure	Comedy	Fantasy	Kids
Adventure	Comedy	Fantasy	Action
Adventure	Comedy	Fantasy	Shounen
Action	Adventure	Comedy	Sci-Fi
Action	Adventure	Comedy	Fantasy
Action	Adventure	Comedy	Shounen
Comedy	Fantasy	Kids	Adventure