CSCI620 - Introduction to Big Data

Project Phase I

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

PARIJAT KAWALE pk7145

CHENGZI CAO cc3773

Table of Contents

1. Data Description

MyAnimeList, often abbreviated as **MAL**, is an anime and manga social networking and social cataloging application website run by volunteers. Our data set contains a list of users and the anime list which has been cataloged by such volunteers. We also have a mapping of each user and the anime which he/she has watched with relevant statistics for the same.

The raw data set contains the following files:

1. AnimeList.csv

This file contains a list of animes along with relevant attributes such as their titles (in English and Japanese), opening/closing theme songs, type (tv series, movie, music, etc), rating (PG, MA, etc), the source from which it has been adapted from (video games, manga, etc) and its genre. There is also other relevant broadcasting information for each episode such as episode count, duration, airing date and current airing status. Also included is statistical information for each title such as its average score on a scale of 10, number of votes, popularity and a follow count (members). We also have its production information such as producer, licensor, studio, etc. Each anime has a unique ID which will act as our primary key.

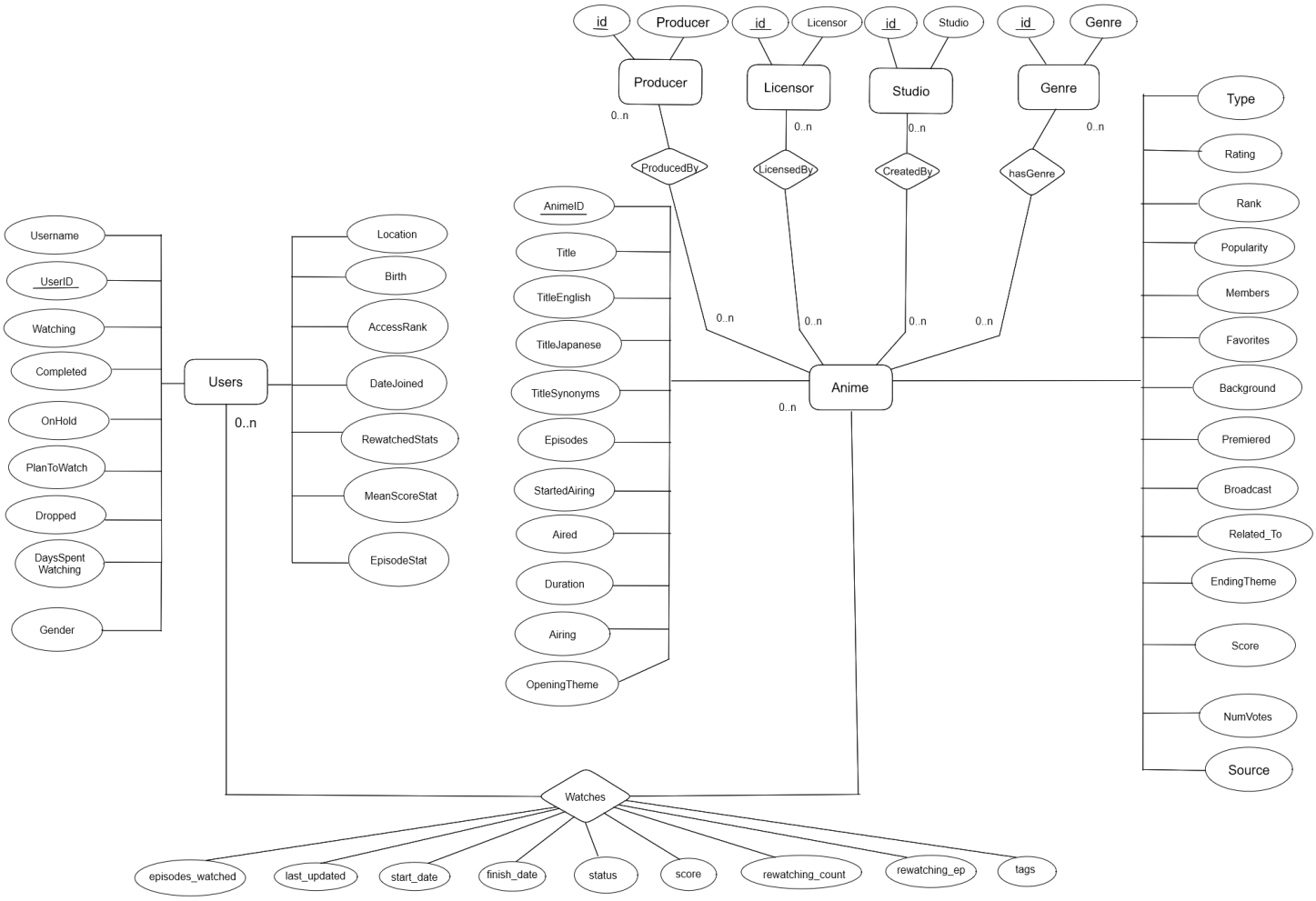
2. UserList.csv

This file contains a user's biographical data such as his username, gender, birth date, location, and the date he joined the platform. We also have statistical data for each user such as anime count for what they are watching, what they have completed, what they left midway (user_dropped), what they have currently paused (user_hold) and what they plan to watch (user_plantowatch). There is also the mean score (average rating given for all the anime they have watched), their user rank (an arbitrary metric to rank the access privileges of a user by other users), how many times they have watched an anime and how many episodes they have watched.

3. UserAnimeList.csv

This data acts as a junction between the anime list and the users. It contains a n: n mapping of the users with the animes that they have watched. We also have statistical data for each user start date, finish date, score, what episode he/she is re-watching currently, how many times he/she has rewatched the episode for each mapping.

2. Entity Relationship Diagram



3. Relational Model

A. Entities

1. Anime = (anime_id, title, title_english, title_japanese, title_synonyms, aired, aired_from_to, duration, isAiring, rank, popularity, members, favorites, background, premiered, broadcast, related_to, opening_theme, ending_theme, score, num_votes)

Anime entity table represents unique anime in our data with it being populated from Anime_List.csv which has been described in Section 1.1. Attribute anime_id acts as the primary key identifying each unique tuple in the table.

2. Users = (user_id, username, gender, birth, user_watching, user_completed, user_onhold, user_dropped, user_plantowatch, user_days_spent_watching, user_location, access_rank, join_date, stats_mean_score, stats_rewatched, stats_episode)

User entity table represent unique users in our data with the table being populated with data from User_List.csv which has been described in Section 1.2. Attribute user_id acts as the primary key identifying each unique tuple in the table.

3. Genre = (genre_id, genre)

Genre table represents a list of all the distinct genres present for all the animes in our dataset. genre_id acts as the primary key for the table and genre data is populated from Anime_list which has been described in Section 1.1. We use unnest() to split the list of all genres for each anime so we can filter out individual genre values and store them for a relation/junction table later on.

4. Producer = (producer_id, producer)

Producer table represents a list of all distinct producers present for all the animes in our dataset. producer_id acts as the primary key for the table and the data is populated from Anime_list which has been described in Section 1.1.

5. Licensor = (licensor_id, licensor)

Licensor table represents a list of all distinct licensors present for all the animes in our dataset. licensor_id acts as the primary key for the table and the data is populated from Anime_list which has been described in Section 1.1.

6. Studio = (studio_id, studio)

Studio table represents a list of all distinct producers that produce the anime in our dataset. Producer_id acts as the primary key for the table and the data is populated from Anime_list which has been described in Section 1.1.

B. Relations

1. Watches = (user_id, anime_id, my_watched_episodes, my_start_date, my_finish_date, my_score, my_status, my_rewatching, my_rewatching_ep, my_last_updated, my_tags)

Watches relation table is responsible for creating the unique mappings for each user with each anime he has watched/is watching. user_id and anime_id act as the primary key. user_id is a foreign key referencing the Users table from section A whereas anime_id is a foreign key referencing the Animes table from section A.
The relation also has its own set of attributes which represents statistical data which has been populated from UserAnimeList.csv which has been described in Section 1.3.
This is a many : many relationship.

2. ProducedBy = (anime_id, producer_id)

ProducedBy relation contains a mapping of all the distinct anime and producer pairs. anime_id and producer_id act as primary keys. anime_id acts as the foreign key referencing Anime table and producer_id acts as the foreign key referencing Producer table. Data for ProducedBy has been populated with the help of both of these tables.
This is a many : many relationship.

3. LicensedBy = (anime_id, licensor_id)

LicensedBy relation contains a mapping of all the distinct anime and licensor pairs. anime_id and licensor_id act as primary keys. anime_id acts as the foreign key referencing Anime table and licensor_id acts as the foreign key referencing Licensor table. Data for LicensedBy has been populated with the help of both of these tables.
This is a many : many relationship.

4. CreatedBy = (anime_id, studio_id)

CreatedBy relation contains a mapping of all the distinct anime and studio pairs. anime_id and studio_id act as primary keys. anime_id acts as the foreign key referencing Anime table and studio_id acts as the foreign key referencing studio table. Data for LicensedBy has been populated with the help of both of these tables.
This is a many : many relationship.

5. hasGenre = (anime_id, genre_id)

hasGenre relation contains a mapping of all the distinct anime and their genre. anime_id and genre_id act as primary keys. anime_id acts as the foreign key referencing Anime table and genre_id acts as the foreign key referencing studio table. Data for LicensedBy has been populated with the help of both of these tables.
This is a many : many relationship.

4. Populating Data in the database

Data will be populated in the database by running the following SQL scripts in order.
The screenshots for the scripts can be found in the Appendix.

1. loadRawData.sql -
    a. Creates the temporary tables user, anime, user_anime and copies the data from the csv files which have been pulled from the dataset on kaggle.
    The absolute path for the csv file locations would need to be changed depending on where they are stored.
2. createTables.sql -
    b. Creates the entity and relation tables and populates them with data using the temporary data tables created above.

5. Proof of Data

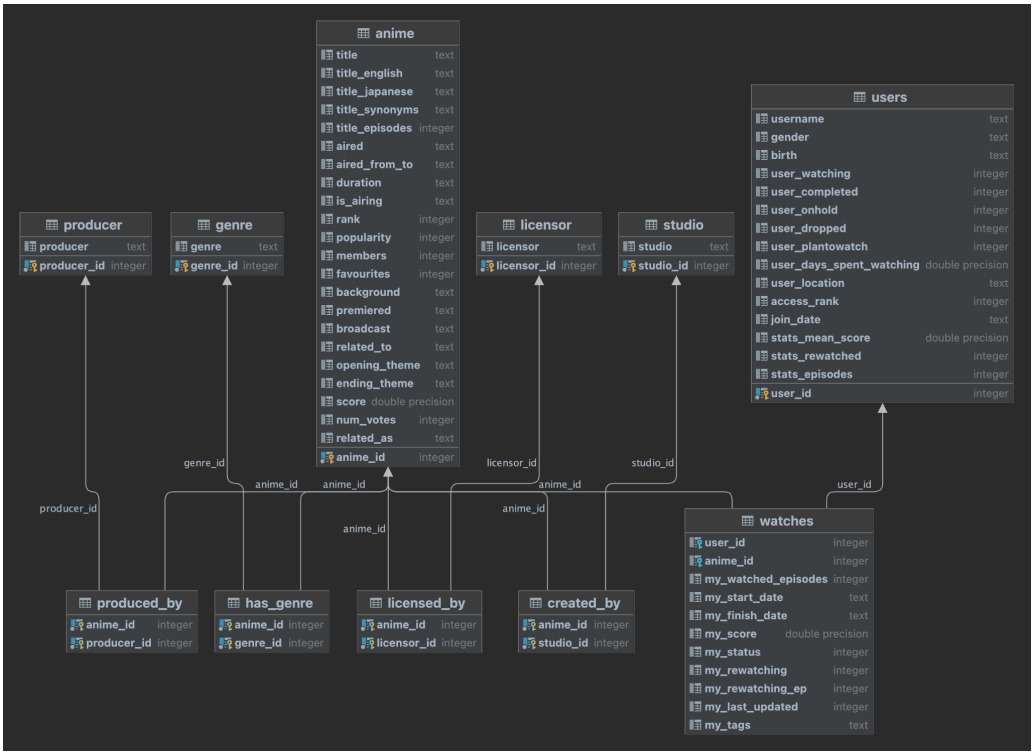


Figure 1: Visualization of Database Generated by DataGrip

	anim...	title	title_english	title_japanese	title_synonyms	title_episodes	aired	aired_from_to	duration	is_
1	1	Cowboy Bebop	Cowboy Bebop	カウボーイビバップ	<null>	26	Apr 3, 199…	{'from': '1998-04…	24 min. per ep.	False
2	5	Cowboy Bebo…	Cowboy Bebop: The…	カウボーイビバップ天国の	Cowboy Bebop: Knoc…	1	Sep 1, 2001	{'from': '2001-09…	1 hr. 54 min.	False
3	6	Trigun	Trigun	トライガン	<null>	26	Apr 1, 199…	{'from': '1998-04…	24 min. per ep.	False
4	7	Witch Hunte…	Witch Hunter Robin	Witch Hunter ROBIN	WHR	26	Jul 2, 200…	{'from': '2002-07…	25 min. per ep.	False
5	8	Beet the Va…	Beet the Vandel B…	冒険王ビィト	Adventure King Bee…	52	Sep 30, 20…	{'from': '2004-09…	23 min. per ep.	False
6	15	Eyeshield 21	<null>	アイシールド21	Eyeshield21	145	Apr 6, 200…	{'from': '2005-04…	23 min. per ep.	False
7	16	Hachimitsu …	Honey and Clover	ハチミツとクローバー	HachiKuro, Honey &…	24	Apr 15, 20…	{'from': '2005-04…	23 min. per ep.	False
8	17	Hungry Hear…	<null>	ハングリーハート Wil…	<null>	52	Sep 11, 20…	{'from': '2002-09…	23 min. per ep.	False
9	18	Initial D F…	<null>	頭文字〈イニシャル〉D…	Initial D 4th Stage	24	Apr 17, 20…	{'from': '2004-04…	27 min. per ep.	False
10	19	Monster	Monster	モンスター	<null>	74	Apr 7, 200…	{'from': '2004-04…	24 min. per ep.	False
11	20	Naruto	Naruto	ナルト	NARUTO	220	Oct 3, 200…	{'from': '2002-10…	23 min. per ep.	False
12	21	One Piece	One Piece	ONE PIECE	OP	0	Oct 20, 19…	{'from': '1999-10…	24 min.	True
13	22	Tennis no O…	The Prince of Ten…	テニスの王子様	<null>	178	Oct 10, 20…	{'from': '2001-10…	22 min. per ep.	False
14	23	Ring ni Kak…	<null>	リングにかけろ1	Put it all in the …	12	Oct 6, 200…	{'from': '2004-10…	25 min. per ep.	False
15	24	School Rumb…	School Rumble	スクールランブル	<null>	26	Oct 5, 200…	{'from': '2004-10…	23 min. per ep.	False
16	25	Sunabouzu	Desert Punk	砂ぼうず	Sunabozu	24	Oct 6, 200…	{'from': '2004-10…	24 min. per ep.	False
17	26	Texhnolyze	Texhnolyze	TEXHNOLYZE	Technolyze	22	Apr 17, 20…	{'from': '2003-04…	23 min. per ep.	False
18	27	Trinity Blo…	Trinity Blood	トリニティ・ブラッド	<null>	24	Apr 29, 20…	{'from': '2005-04…	24 min. per ep.	False
19	28	Yakitate!! …	Yakitate!! Japan	焼きたて!! ジャぱん	Freshly Baked!! Ja…	69	Oct 12, 20…	{'from': '2004-10…	24 min. per ep.	False
20	29	Zipang	<null>	ジパング	<null>	26	Oct 8, 200…	{'from': '2004-10…	24 min. per ep.	False
21	30	Neon Genesi…	Neon Genesis Evan…	新世紀エヴァンゲリオン	Shinseiki Evangeli…	26	Oct 4, 199…	{'from': '1995-10…	24 min. per ep.	False
22	31	Neon Genesi…	Neon Genesis Evan…	DCS新世紀エヴァンゲリオ	Shinseiki Evangeli…	1	Mar 15, 19…	{'from': '1997-03…	1 hr. 41 min.	False
23	32	Neon Genesi…	Neon Genesis Evan…	新世紀エヴァンゲリオン…	Shinseiki Evangeli…	1	Jul 19, 19…	{'from': '1997-07…	1 hr. 27 min.	False
24	33	Kenpuu Denk…	Berserk	剣風伝奇ベルセルク	Berserk: The Chron…	25	Oct 8, 199…	{'from': '1997-10…	25 min. per ep.	False

Figure 2: Screenshot of Anime Table

	use... 1	username	gender	birth	user_watching	user_completed	user_onhold	user_dropped	user_plantowatch	user_day
1	1	Xinil	Male	1985-03-04	4	230	8	89	60	
2	3	Aokaado	Male	1988-11-11	48	133	54	44	36	
3	4	Crystal	Female	1989-01-10	13	571	307	0	45	
4	20	vondur	Male	1988-01-25	10	88	11	2	19	
5	23	Amuro	<null>	1988-02-22	20	298	5	19	50	
6	36	Baman	Male	1988-08-04	34	722	11	43	322	
7	37	megan	Female	1987-06-18	8	71	9	6	14	
8	44	Koreth	Male	<null>	5	71	1	5	14	
9	47	kei-clone	Male	1988-01-01	15	104	22	3	19	
10	48	seif	Female	1987-12-31	16	488	35	18	2	
11	53	Ladholyman	<null>	<null>	0	47	0	0	0	
12	66	Hiromi	Male	1990-02-09	34	148	13	6	10	
13	70	Cruzle	Male	1983-11-09	30	135	9	8	22	
14	72	james2k	Male	<null>	4	50	11	11	0	
15	77	Emp	Female	<null>	13	297	0	0	2	
16	80	koalatees	<null>	1991-10-24	4	31	0	41	0	
17	81	Ramp	Male	1983-10-17	6	86	4	4	1	
18	82	Achtor	Male	1989-06-10	9	149	31	20	123	
19	83	jaames	Male	<null>	2	15	0	0	0	
20	91	marvin_9mar…	Female	<null>	4	57	4	31	29	
21	95	sorairo	Male	<null>	1	114	10	7	26	
22	108	Keitarou	Male	<null>	221	1496	0	0	83	
23	110	happykawaii…	Female	1989-08-03	3	80	0	19	0	
24	112	luffykun	Male	1983-06-09	1	26	0	0	0	

vices

Figure 3: Screenshot of Users Table

	genre_id	genre
1	1	Sports
2	2	Vampire
3	3	Kids
4	4	Police
5	5	Thriller
6	6	Romance
7	7	Adventure
8	8	Demons
9	9	Shounen
10	10	Mecha
11	11	Shounen Ai
12	12	Comedy
13	13	Samurai
14	14	Psychological
15	15	Hentai
16	16	Super Power
17	17	Harem
18	18	Historical
19	19	Ecchi
20	20	Magic
21	21	Music
22	22	Fantasy
23	23	Seinen
24	24	Horror
25	25	Drama

Figure 4: Screenshot of Genre Table

	producer_id	producer
1	1	Blue Eyes
2	2	Armor
3	3	Being
4	4	Sun TV
5	5	Myung Films
6	6	TOMY Company
7	7	Brave Hearts
8	8	Echoes
9	9	Kitty Media
10	10	Koei
11	11	Hoods Entertainment
12	12	Gold Bear
13	13	Amino
14	14	Toranoana
15	15	Ministry of the Navy
16	16	Shanghai Tiantan Culture & Media
17	17	Studio Unicorn
18	18	Kokusai Eigasha
19	19	Coastline Animation Studio
20	20	Dentsu
21	21	1st PLACE
22	22	Fujio Production
23	23	Queen Bee
24	24	Shingeki no Kyojin Team
25	25	Nippon Columbia

Figure 5: Screenshot of Producer Table

	licensor_id	licensor
1	1	Media Blasters
2	2	Crunchyroll
3	3	Miramax Films
4	4	Elevenarts
5	5	Kitty Media
6	6	Maiden Japan
7	7	Ketchup Entertainment
8	8	Konami
9	9	DreamWorks
10	10	Central Park Media
11	11	SoftCel Pictures
12	12	Aniplex of America
13	13	Marvel Entertainment
14	14	Bandai Visual USA
15	15	Enoki Films
16	16	ADV Films
17	17	Viz Media
18	18	Kadokawa Pictures USA
19	19	feel.
20	20	Nintendo of America
21	21	Geneon Entertainment USA
22	22	Haoliners Animation League
23	23	Tencent Animation
24	24	Geneon Universal Entertainment
25	25	bilibili

Figure 6: Screenshot of Licensor Table

	studio_id	studio
1	1	Tama Production
2	2	Zero-G
3	3	Rikuentai
4	4	Planet
5	5	Office DCI
6	6	Suzuki Mirano
7	7	Studio Meditation With a Pencil
8	8	Silver Link.
9	9	Echoes
10	10	Charaction
11	11	Pollyanna Graphics
12	12	Studio Z5
13	13	Barnum Studio
14	14	Telescreen BV
15	15	Mook Animation
16	16	Daewon Media
17	17	Hoods Entertainment
18	18	Image House
19	19	Studio! Cucuri
20	20	Studio 3Hz
21	21	LMD
22	22	Studio Unicorn
23	23	Kokusai Eigasha
24	24	Coastline Animation Studio
25	25	TriF Studio

Figure 7: Screenshot of Studio Table

	user_id ▲ 1	anime_id	my_watched_episodes	my_start_date	my_finish_date	my_score	my_status	my_rewatching	my_rewatching_e
1	1	202	26	2003-06-22	2004-03-01	8	2	0	
2	1	165	26	2003-04-15	2003-04-23	8	2	0	
3	1	2001	27	2007-04-12	2007-10-07	8	2	0	
4	1	203	5	2003-05-19	2003-05-22	7	2	0	
5	1	79	1	2007-03-20	0000-00-00	0	4	0	
6	1	160	13	2003-04-08	2003-04-15	9	2	0	
7	1	841	16	2006-04-04	2006-04-10	6	2	0	
8	1	1965	5	2007-04-15	0000-00-00	6	4	0	
9	1	11113	0	0000-00-00	0000-00-00	0	6	0	
10	1	208	3	2005-03-31	2005-03-31	8	2	0	
11	1	52	26	2002-02-04	2002-02-06	7	2	0	
12	1	72	12	2003-08-28	2003-11-20	10	2	0	
13	1	330	13	2005-11-19	2005-11-25	7	2	0	
14	1	159	5	2005-09-27	0000-00-00	0	4	0	
15	1	458	3	2003-10-13	2003-10-13	4	2	0	
16	1	834	0	0000-00-00	0000-00-00	0	6	0	
17	1	1453	35	2006-08-29	0000-00-00	6	4	0	
18	1	1943	1	2008-06-09	2008-06-09	7	2	<null>	
19	1	3001	11	2007-10-21	2008-01-17	8	2	0	
20	1	11061	148	2013-04-23	2014-09-23	9	2	<null>	
21	1	304	1	2003-01-24	2003-01-25	8	2	0	
22	1	210	161	2004-10-12	2004-11-09	7	2	0	
23	1	101	5	2005-01-13	0000-00-00	0	4	0	
24	1	45	95	2002-03-24	2002-04-02	8	2	0	
25	1	71	24	2003-01-24	2003-01-26	9	2	0	

Figure 8: Screenshot of Watches Table

	anime_id	producer_id
1	2268	912
2	33433	827
3	6064	110
4	883	720
5	9135	885
6	17919	1041
7	3025	269
8	22563	804
9	25429	865
10	198	676
11	8599	327
12	10079	529
13	33573	460
14	11013	505
15	11013	454
16	11013	95
17	11013	720
18	11013	735
19	2104	641
20	2104	327
21	2104	454
22	2104	890
23	5262	641
24	5262	890
25	721	941

Figure 9: Screenshot of Produced_By Table

	anime_id	producer_id ▲ 1
1	2946	1
2	4496	1
3	3303	1
4	3107	1
5	3911	1
6	4365	2
7	32606	3
8	35434	3
9	30740	3
10	34156	3
11	33419	4
12	31845	4
13	34403	4
14	32601	4
15	12917	5
16	6902	6
17	36524	7
18	17827	7
19	36515	7
20	32271	7
21	33534	8
22	33535	8
23	33537	8
24	31057	8
25	33536	8

Figure 10: Screenshot of Licensed_By Table

	anime_id	studio_id
1	11013	184
2	2104	282
3	5262	123
4	721	270
5	12365	153
6	6586	139
7	6586	335
8	178	418
9	2787	153
10	4477	153
11	853	48
12	4814	292
13	7054	153
14	1557	292
15	11123	292
16	14227	183
17	269	139
18	59	324
19	6045	122
20	1735	139
21	210	292
22	4224	153
23	10030	153
24	74	457
25	4722	270

Figure 11: Screenshot of Created By Tables

	anime_id	genre_id
1	21085	39
2	67	39
3	6919	39
4	9613	39
5	2332	39
6	961	39
7	712	39
8	1675	39
9	13041	39
10	36561	39
11	13117	39
12	6973	39
13	6682	39
14	36160	39
15	2884	39
16	34248	39
17	9202	39
18	20723	39
19	2144	39
20	30061	39
21	9731	39
22	18283	39
23	30965	39
24	9465	39
25	30915	39

Figure 12: Screenshot of Has_Genre Table

6. APPENDIX

1. loadRawData.sql

```
-------------------------------------------------------------
-- CSCI-620: INTRO TO BIG DATA
-- PROJECT PHASE 1
-- FILENAME: loadRawData.sql
-- AUTHORS: ATHINA STEWART  as1986
--          ARCHIT JOSHI    aj6082
--          PARIJAT KAWALE  pk7145
--          CHENGZI CAO     cc3773
-------------------------------------------------------------

-------------------------------------------------------------
-- THIS SCRIPT LOADS DATA FROM RAW DATASETS INTO THE DATABASE
-------------------------------------------------------------

-- create anime schema
create schema anime;

-- set search path to anime schema
set search_path to anime;

-- create raw user data
CREATE table USERS_Raw(
   username text,
   user_id int,
   user_watching int,
   user_completed int,
   user_onhold int,
   user_dropped int,
   user_plantowatch int,
   user_days_spent_watching float,
   gender text,
   user_location text,
   birth text,
   access_rank int,
   join_date text,
   last_online text,
   stats_mean_score float,
   stats_rewatched int,
   stats_episodes int
);

-- copy raw user data from the file into the table
```

```
COPY anime."users_raw" FROM '/Users/Athina/Public/UserList.csv' DELIMITER ','
CSV HEADER;

-------------------------------------------------------------

-- create raw anime data
CREATE table AnimeList_Raw(
   anime_id int,
   title text,
   title_english text,
   title_japanese text,
   title_synonyms text,
   image_url text,
   anime_type text,
   source text,
   episodes int,
   status text,
   airing text,
   aired_string text,
   aired text,
   duration text,
   rating text,
   score float,
   scored_by int,
   rank int,
   popularity int,
   members int,
   favourites int,
   background text,
   premiered text,
   broadcast text,
   related text,
   producer text,
   licensor text,
   studio text,
   genre text,
   opening_theme text,
   ending_theme text
);

-- copy raw anime data from the file into the table
COPY anime."animelist_raw" FROM '/Users/Athina/Public/AnimeList.csv'
DELIMITER ',' CSV HEADER;

-------------------------------------------------------------

-- create raw user-anime junction data
CREATE table User_Anime_Raw(
   username text,
```

```
   anime_id int,
   my_watched_episodes int,
   my_start_date text,
   my_finish_date text,
   my_score float,
   my_status int,
   my_rewatching int,
   my_rewatching_ep int,
   my_last_updated int,
   my_tags text
);

-- copy raw user_anime data from the file into the table
COPY anime."user_anime_raw" FROM '/Users/Athina/Public/UserAnimeList.csv'
DELIMITER ',' CSV HEADER;
```

2. createTables.sql

```
-------------------------------------------------------------
-- CSCI-620: INTRO TO BIG DATA
-- PROJECT PHASE 1
-- FILENAME: createTables.sql
-- AUTHORS: ATHINA STEWART  as1986
--          ARCHIT JOSHI    aj6082
--          PARIJAT KAWALE  pk7145
--          CHENGZI CAO     cc3773
-------------------------------------------------------------

-------------------------------------------------------------
-- THIS SCRIPT CREATES MAIN TABLES AND RELATION TABLES
-------------------------------------------------------------

-- set search path to the anime schema
SET search_path to anime;

-------------------------------------------------------------
-- CREATING MAIN TABLES
-------------------------------------------------------------

-- create anime table
CREATE TABLE Anime(
   anime_id int PRIMARY KEY,
   title text,
   title_english text,
   title_japanese text,
   title_synonyms text,
```

```
   title_episodes int,
   aired text,
   aired_from_to text,
   duration text,
   is_airing text,
   rank int,
   popularity int,
   members int,
   favourites int,
   background text,
   premiered text,
   broadcast text,
   related_to text,
   opening_theme text,
   ending_theme text,
   score float,
   num_votes int,
   related_as text
);

-- into data from raw dataset into main anime table
INSERT INTO anime (anime_id, title, title_english, title_japanese,
title_synonyms,
                                    title_episodes, aired, aired_from_to,
duration, is_airing,
                                    rank, popularity, members, favourites,
background, premiered,
                                    broadcast, related_to, opening_theme,
ending_theme, score, num_votes)
SELECT anime_id, title, title_english, title_japanese, title_synonyms,
episodes, aired_string, aired,
      duration, airing, rank, popularity, members, favourites, background,
premiered, broadcast, related,
      opening_theme, ending_theme, score, scored_by
FROM animelist_raw;
-------------------------------------------------------------

-- create users table
CREATE table Users(
   user_id int PRIMARY KEY,
   username text,
   gender text,
   birth text,
   user_watching int,
   user_completed int,
   user_onhold int,
   user_dropped int,
   user_plantowatch int,
   user_days_spent_watching float,
```

```
   user_location text,
   access_rank int,
   join_date text,
   stats_mean_score float,
   stats_rewatched int,
   stats_episodes int
);

-- into data from raw dataset into main users table
INSERT INTO Users (user_id, username, gender, birth, user_watching,
user_completed,
                                    user_onhold, user_dropped,
user_plantowatch,
                                    user_days_spent_watching, user_location,
                                    access_rank, join_date, stats_mean_score,
                                    stats_rewatched, stats_episodes)
SELECT user_id, username, gender, birth, user_watching, user_completed,
                                    user_onhold, user_dropped,
user_plantowatch,
                                    user_days_spent_watching, user_location,
                                    access_rank, join_date, stats_mean_score,
                                    stats_rewatched, stats_episodes
FROM USERS_RAW
ON CONFLICT DO NOTHING;
-------------------------------------------------------------

-- create genre table
CREATE table Genre (
   genre_id SERIAL PRIMARY KEY,
   genre text
);

-- insert values into genre table
INSERT INTO Genre(genre)
SELECT DISTINCT(genre) FROM
(SELECT unnest(string_to_array(genre, ', '))  as genre FROM animelist_raw) as
distinctgenre;
-------------------------------------------------------------

-- create producer table
CREATE table Producer (
   producer_id SERIAL PRIMARY KEY,
   producer text
);

-- insert values into producer table
INSERT INTO Producer(producer)
SELECT DISTINCT(producer) FROM
```

```
(SELECT unnest(string_to_array(producer, ', '))  as producer FROM
animelist_raw) as distinctproducer;
-------------------------------------------------------------

-- create licensor table
CREATE table Licensor (
   licensor_id SERIAL PRIMARY KEY,
   licensor text
);

-- insert values into licensor table
INSERT INTO Licensor(licensor)
SELECT DISTINCT(licensor) FROM
(SELECT unnest(string_to_array(licensor, ', '))  as licensor FROM
animelist_raw) as distinctlicensor;
-------------------------------------------------------------

-- create studio table
CREATE table Studio (
   studio_id SERIAL PRIMARY KEY,
   studio text
);

-- insert values into studio table
INSERT INTO Studio(studio)
SELECT DISTINCT(studio) FROM
(SELECT unnest(string_to_array(studio, ', '))  as studio FROM animelist_raw)
as distinctstudio;

-------------------------------------------------------------
-- CREATING RELATION TABLES
-------------------------------------------------------------

-- created watches table (many:many relation between user and anime)

CREATE table Watches(
   user_id int,
   FOREIGN KEY (user_id) REFERENCES users(user_id),
   anime_id int,
   FOREIGN KEY (anime_id) REFERENCES anime (anime_id),
   my_watched_episodes int,
   my_start_date text,
   my_finish_date text,
   my_score float,
   my_status int,
   my_rewatching int,
   my_rewatching_ep int,
   my_last_updated int,
   my_tags text
```

```
);

-- insert values into watches table
INSERT INTO Watches(user_id, anime_id, my_watched_episodes, my_start_date,
                    my_finish_date, my_score, my_status, my_rewatching,
                        my_rewatching_ep, my_last_updated, my_tags)
SELECT UAO.user_id,
   user_anime_raw.anime_id,
   my_watched_episodes int,
   my_start_date       text,
   my_finish_date      text,
   my_score            float,
   my_status           int,
   my_rewatching       int,
   my_rewatching_ep    int,
   my_last_updated     int,
   my_tags             text
   FROM user_anime_raw
INNER JOIN Users UAO on user_anime_raw.username = UAO.username;
-------------------------------------------------------------

-- create has_genre table (many:many relation between anime and genre)
CREATE table has_genre(
   anime_id int,
   FOREIGN KEY (anime_id) REFERENCES anime (anime_id),
   genre_id int,
   FOREIGN KEY (genre_id) REFERENCES genre(genre_id),
   PRIMARY KEY (anime_id, genre_id)
);

-- create temp table with genre_id, genre and anime_id values
CREATE table Anime_Genre_Temp (
   genre_id int,
   genre text,
   anime int
);

-- unnest the genres rows to obtain the individual genres
INSERT INTO Anime_Genre_Temp(genre, anime)
SELECT unnest(string_to_array(genre, ', ')) as "genre", anime_id FROM
animelist_raw;

-- update the temp table to include matching ids
UPDATE Anime_Genre_Temp
   SET genre_id = Genre.genre_id
   FROM genre
   WHERE genre.genre = Anime_Genre_Temp.genre;

-- insert genre_ids into main table
```

```sql
INSERT INTO has_genre(anime_id, genre_id)
SELECT anime, genre_id FROM anime_genre_temp;

-- drop temp table
drop table anime_genre_temp;
-------------------------------------------------------------

-- create produced_by table (many:many relation between anime and producer)
CREATE table Produced_By(
   anime_id int,
   FOREIGN KEY (anime_id) REFERENCES anime (anime_id),
   producer_id int,
   FOREIGN KEY (producer_id) REFERENCES producer(producer_id),
   PRIMARY KEY (anime_id, producer_id)
);

-- create temp table with producer_id, producer and anime_id values
CREATE table Anime_Producer_Temp (
   producer_id int,
   producer text,
   anime int
);

-- unnest the producers rows to obtain the individual producers
INSERT INTO Anime_Producer_Temp(producer, anime)
SELECT unnest(string_to_array(producer, ', ')) as "producer", anime_id FROM
animelist_raw;

-- update the temp table to include matching ids
UPDATE Anime_Producer_Temp
   SET producer_id = producer.producer_id
   FROM producer
   WHERE Producer.producer = anime_producer_temp.producer;

-- insert producer_ids into main table
INSERT INTO produced_by(anime_id, producer_id)
SELECT anime, producer_id FROM anime_producer_temp;

-- drop temp table
drop table anime_producer_temp;
-------------------------------------------------------------

-- create licensed_by table (many:many relation between anime and licensor)
CREATE table Licensed_By(
   anime_id int,
   FOREIGN KEY (anime_id) REFERENCES anime (anime_id),
   licensor_id int,
   FOREIGN KEY (licensor_id) REFERENCES licensor(licensor_id),
   PRIMARY KEY (anime_id, licensor_id)
```

```
);

-- create temp table with licensor_id, licensor and anime_id values
CREATE table Anime_Licensor_Temp (
   licensor_id int,
   licensor text,
   anime int
);

-- unnest the licensors rows to obtain the individual licensors
INSERT INTO Anime_Licensor_Temp(licensor, anime)
SELECT unnest(string_to_array(licensor, ', ')) as "licensor", anime_id FROM
animelist_raw;

-- update the temp table to include matching ids
UPDATE Anime_Licensor_Temp
   SET licensor_id = Licensor.licensor_id
   FROM licensor
   WHERE Licensor.licensor = anime_licensor_temp.licensor;

-- insert licensor_ids into main table
INSERT INTO licensed_by(anime_id, licensor_id)
SELECT anime, licensor_id FROM anime_licensor_temp;

-- drop temp table
drop table anime_licensor_temp;

------------------------------------------------------------

-- create has_genre table (many:many relation between anime and studio)
CREATE table created_by(
   anime_id int,
   FOREIGN KEY (anime_id) REFERENCES anime (anime_id),
   studio_id int,
   FOREIGN KEY (studio_id) REFERENCES studio(studio_id),
   PRIMARY KEY (anime_id, studio_id)
);

-- create temp table with studio_id, studio and anime_id values
CREATE table Anime_Studio_Temp (
   studio_id int,
   studio text,
   anime int
);

-- unnest the studio rows to obtain the individual studios
INSERT INTO Anime_Studio_Temp(studio, anime)
SELECT unnest(string_to_array(studio, ', ')) as "studio", anime_id FROM
animelist_raw;
```

```sql
-- update the temp table to include matching ids
UPDATE Anime_Studio_Temp
   SET studio_id = studio.studio_id
   FROM studio
   WHERE studio.studio = anime_studio_temp.studio;

-- insert studio_ids into main table
INSERT INTO created_by(anime_id, studio_id)
SELECT anime, studio_id FROM anime_studio_temp;

-- drop temp table
drop table anime_studio_temp;

-- drop raw tables
drop table animelist_raw;
drop table user_anime_raw;
drop table users_raw;
```