Convert SQL to NoSQL and Social Media

ABSTRACT

I am working on converting an SQL database to NoSQL without any loss of data. The dataset used is Anime and MongoDB will be used for NoSQL. Also, data related to our dataset will be retrieved from Twitter by interacting with Twitter API and interesting findings will be made by querying the database.

DATA has been taken from the following source:

- CSV files

DATA acquired from CSV files

In [53]:
```
# importing necessary Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import pymongo
import json
```

In [17]:
```
#Reading data from csv file which includes anime details
dataframe_entire = pd.read_csv("animelist.csv",encoding='latin1')
dataframe_entire
```

Out[17]:

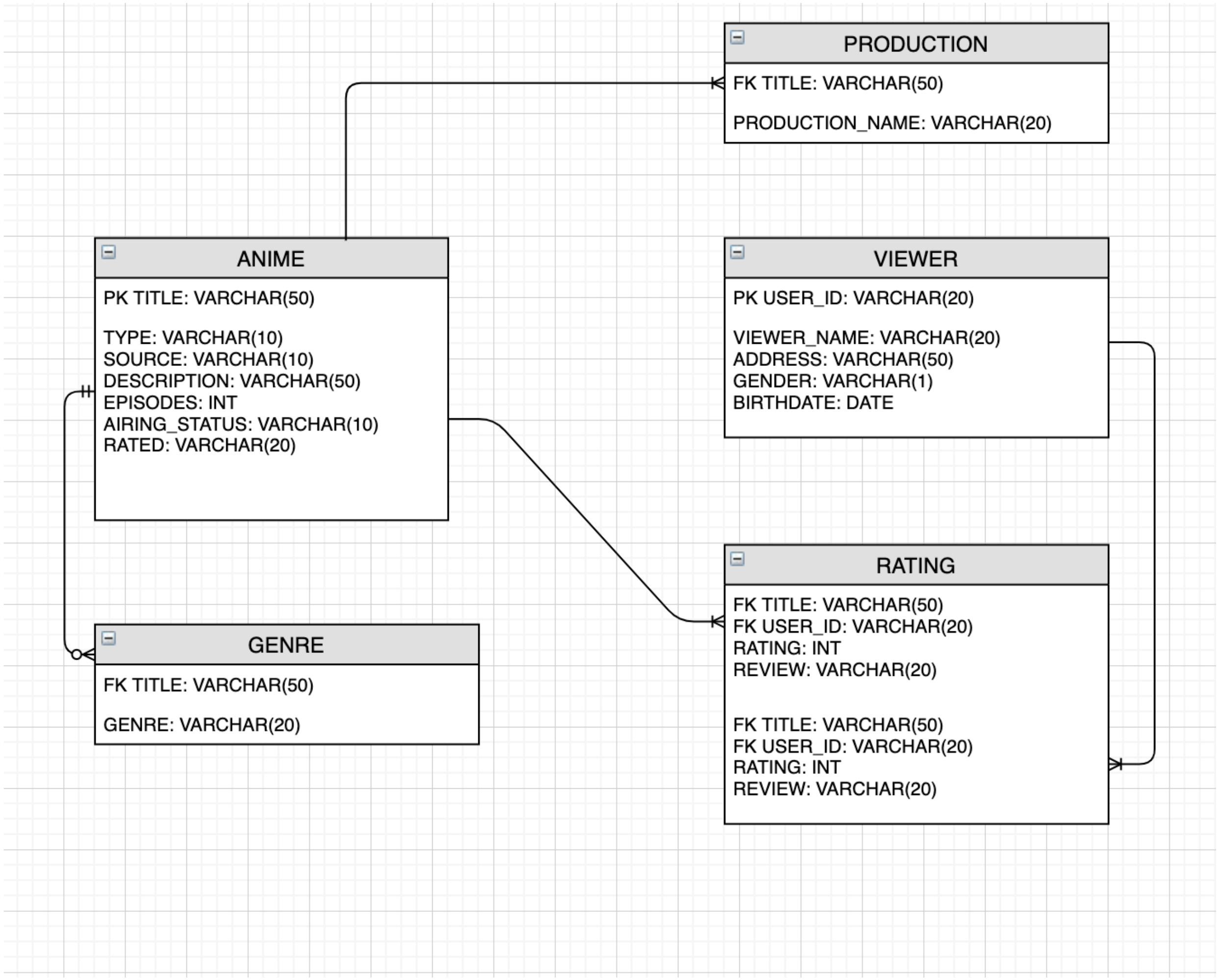
	title	type	source	episodes	status	rating	background	producer	studio
0	Inu x Boku SS	TV	Manga	12	Finished Airing	PG-13 - Teens 13 or older	Inu x Boku SS was licensed by Sentai Filmworks...	Aniplex	David Production
1	Seto no Hanayome	TV	Manga	26	Finished Airing	PG-13 - Teens 13 or older	NaN	TV Tokyo	Gonzo
2	Shugo Chara!! Doki	TV	Manga	51	Finished Airing	PG - Children	NaN	TV Tokyo	Satelight
3	Princess Tutu	TV	Original	38	Finished Airing	PG-13 - Teens 13 or older	Princess Tutu aired in two parts. The first pa...	Memory-Tech	Hal Film Maker
4	Bakuman. 3rd Season	TV	Manga	25	Finished Airing	PG-13 - Teens 13 or older	NaN	NHK	J.C.Staff
...	...	...	...	...	...	...	...	...	...
724	Busou Shinki Moon Angel	ONA	Other	10	Finished Airing	PG-13 - Teens 13 or older	NaN	NaN	TNK, Kinema Citrus
725	Nagi no Asukara	TV	Original	26	Finished Airing	PG-13 - Teens 13 or older	Episode 1 was previewed at a screening in Toky...	Geneon Universal Entertainment	P.A. Works
726	Tenjou Tenge: The Past Chapter	Movie	Manga	1	Finished Airing	R - 17+ (violence & profanity)	NaN	NaN	Madhouse
727	Shisha no Teikoku	Movie	Novel	1	Finished Airing	R - 17+ (violence & profanity)	Winner of the Platinum Grand Prize during the ...	NaN	Wit Studio
728	Final Fantasy VII: Advent Children	Movie	Game	1	Finished Airing	PG-13 - Teens 13 or older	The film received the Honorary Maria Award at ...	NaN	Square Enix

729 rows × 12 columns

Entity Relatonship Diagram

In [19]:
```
from IPython.display import Image
Image("/Users/shashank/Pers/NEU 2nd Sem/DMDD/assignment3/Images/ER.png")
```

Out[19]:



Entities being saved in MongoDB

Anime

In [20]:
```
# inserting data in MongoDB
from pymongo import MongoClient

try:
    conn = MongoClient('localhost', 27017)
    print("Connected successfully!!!")
except:
    print("Could not connect to MongoDB")

# database
db = conn.AnimeDatabase

# Created or Switched to collection
collection = db.animecollection

#Iterating over the complete anime details and generating Anime Table
df1 = dataframe_entire.iloc[:, [6,7,8,11]]
AnimeTable = dataframe_entire[[col for col in dataframe_entire.columns if c
AnimeTable

# Insert Data by converting to JSON
collection.insert_many(AnimeTable.to_dict("records"))

# Printing the data inserted
data_inserted = collection.find()
for record in data_inserted:
    print(record)
```

```
'type': 'TV', 'source': 'Manga', 'episodes': 12, 'status': 'Finished Airi
ng', 'rating': 'PG-13 - Teens 13 or older', 'duration_min': 24.0, 'aired_
from_year': 2012}
{'_id': ObjectId('5e8aca021dfb19c32ff3debf'), 'title': 'Seto no Hanayom
e', 'type': 'TV', 'source': 'Manga', 'episodes': 26, 'status': 'Finished
Airing', 'rating': 'PG-13 - Teens 13 or older', 'duration_min': 24.0, 'ai
red_from_year': 2007}
{'_id': ObjectId('5e8aca021dfb19c32ff3dec0'), 'title': 'Shugo Chara!! Dok
i', 'type': 'TV', 'source': 'Manga', 'episodes': 51, 'status': 'Finished
Airing', 'rating': 'PG - Children', 'duration_min': 24.0, 'aired_from_yea
r': 2008}
{'_id': ObjectId('5e8aca021dfb19c32ff3dec1'), 'title': 'Princess Tutu',
'type': 'TV', 'source': 'Original', 'episodes': 38, 'status': 'Finished A
iring', 'rating': 'PG-13 - Teens 13 or older', 'duration_min': 16.0, 'air
ed_from_year': 2002}
{'_id': ObjectId('5e8aca021dfb19c32ff3dec2'), 'title': 'Bakuman. 3rd Seas
on', 'type': 'TV', 'source': 'Manga', 'episodes': 25, 'status': 'Finished
Airing', 'rating': 'PG-13 - Teens 13 or older', 'duration_min': 24.0, 'ai
red_from_year': 2012}
{'_id': ObjectId('5e8aca021dfb19c32ff3dec3'), 'title': 'Yume-iro Pätissið
```

Production

In [21]:
```
# database
db = conn.AnimeDatabase

# Created or Switched to collection
collection = db.productioncollection

#Iterating over the complete anime details and generating Production Table
ProductionTable = dataframe_entire.iloc[:, [0,7,8]]
ProductionTable

# Insert Data by converting to JSON
collection.insert_many(ProductionTable.to_dict("records"))

# Printing the data inserted
data_inserted1 = collection.find()
for record in data_inserted1:
    print(record)
```

```
{'_id': ObjectId('5e8aca201dfb19c32ff3e1d5'), 'title': 'Oniku Daisuki! Ze
ushi-kun: Suteki na Hamburger', 'producer': nan, 'studio': 'DLE'}
{'_id': ObjectId('5e8aca201dfb19c32ff3e1d6'), 'title': 'Soul Buster', 'pr
oducer': 'MAGES.', 'studio': 'Studio Pierrot'}
{'_id': ObjectId('5e8aca201dfb19c32ff3e1d7'), 'title': 'Death Billiards',
'producer': nan, 'studio': 'Madhouse'}
{'_id': ObjectId('5e8aca201dfb19c32ff3e1d8'), 'title': 'Live On Cardliver
Kakeru', 'producer': 'Sotsu', 'studio': 'TMS Entertainment'}
{'_id': ObjectId('5e8aca201dfb19c32ff3e1d9'), 'title': 'Triage X', 'produ
cer': 'DAX Production', 'studio': 'Xebec'}
{'_id': ObjectId('5e8aca201dfb19c32ff3e1da'), 'title': 'Shinkai no Kanta
i: Submarine 707', 'producer': nan, 'studio': 'J.C.Staff, Toei Animatio
n'}
{'_id': ObjectId('5e8aca201dfb19c32ff3e1db'), 'title': 'Nanatsu no Bitok
u', 'producer': 'Hobby Japan', 'studio': 'Bridge'}
{'_id': ObjectId('5e8aca201dfb19c32ff3e1dc'), 'title': 'Hai to Gensou no
Grimgar Special', 'producer': nan, 'studio': 'A-1 Pictures'}
{'_id': ObjectId('5e8aca201dfb19c32ff3e1dd'), 'title': 'Medamayaki no Kim
i Itsu Tsubusu?', 'producer': 'NHK', 'studio': 'Fanworks'}
{'_id': ObjectId('5e8aca201dfb19c32ff3e1de'), 'title': 'Bounty Dog: Getsu
```

Genre

```
In [22]: # database
         db = conn.AnimeDatabase

         # Created or Switched to collection
         collection = db.genrecollection

         #Iterating over the complete anime details and generating Genre Table
         GenreTable = dataframe_entire.iloc[:, [0,11]]
         GenreTable

         # Insert Data by converting to JSON
         collection.insert_many(GenreTable.to_dict("records"))

         # Printing the data inserted
         data_inserted = collection.find()
         for record in data_inserted:
             print(record)
```

```
{'_id': ObjectId('5e8aca341dfb19c32ff3e470'), 'title': 'Inu x Boku SS',
'genre': 'Comedy'}
{'_id': ObjectId('5e8aca341dfb19c32ff3e471'), 'title': 'Seto no Hanayom
e', 'genre': 'Comedy'}
{'_id': ObjectId('5e8aca341dfb19c32ff3e472'), 'title': 'Shugo Chara!! Dok
i', 'genre': 'Comedy'}
{'_id': ObjectId('5e8aca341dfb19c32ff3e473'), 'title': 'Princess Tutu',
'genre': 'Comedy'}
{'_id': ObjectId('5e8aca341dfb19c32ff3e474'), 'title': 'Bakuman. 3rd Seas
on', 'genre': 'Comedy'}
{'_id': ObjectId('5e8aca341dfb19c32ff3e475'), 'title': 'Yume-iro Pätissið
re', 'genre': 'Kids'}
{'_id': ObjectId('5e8aca341dfb19c32ff3e476'), 'title': 'Ultra Maniac', 'g
enre': 'Magic'}
{'_id': ObjectId('5e8aca341dfb19c32ff3e477'), 'title': 'Shakugan no Shana
II (Second)', 'genre': 'Action'}
{'_id': ObjectId('5e8aca341dfb19c32ff3e478'), 'title': 'Nodame Cantabile:
Paris-hen', 'genre': 'Music'}
{'_id': ObjectId('5e8aca341dfb19c32ff3e479'), 'title': 'Ouran Koukou Host
```

Viewer

In [23]:
```
# database
db = conn.AnimeDatabase

# Created or Switched to collection
collection = db.viewercollection

#Reading data from csv file which includes viewers details
ViewerTable = pd.read_csv("viewerslist.csv",encoding='latin1')
ViewerTable

# Insert Data by converting to JSON
collection.insert_many(ViewerTable.to_dict("records"))

# Printing the data inserted
data_inserted = collection.find()
for record in data_inserted:
    print(record)
```

```
{'_id': ObjectId('5e8aca3e1dfb19c32ff3e749'), 'user_id': 'user1', 'userna
me': 'karthiga', 'gender': 'Female', 'birth_date': '4/29/90', 'location':
'Chennai'}
{'_id': ObjectId('5e8aca3e1dfb19c32ff3e74a'), 'user_id': 'user2', 'userna
me': 'RedvelvetDaisuki', 'gender': 'Female', 'birth_date': '1/1/95', 'loc
ation': 'Manila'}
{'_id': ObjectId('5e8aca3e1dfb19c32ff3e74b'), 'user_id': 'user3', 'userna
me': 'Damonashu', 'gender': 'Male', 'birth_date': '8/1/91', 'location':
'Detroit'}
{'_id': ObjectId('5e8aca3e1dfb19c32ff3e74c'), 'user_id': 'user4', 'userna
me': 'bskai', 'gender': 'Male', 'birth_date': '12/14/90', 'location': 'Na
yarit'}
{'_id': ObjectId('5e8aca3e1dfb19c32ff3e74d'), 'user_id': 'user5', 'userna
me': 'terune_uzumaki', 'gender': 'Female', 'birth_date': '8/24/98', 'loca
tion': 'Malaysia'}
{'_id': ObjectId('5e8aca3e1dfb19c32ff3e74e'), 'user_id': 'user6', 'userna
me': 'Bas_G', 'gender': 'Male', 'birth_date': '10/24/99', 'location': 'Ni
jmegen'}
{'_id': ObjectId('5e8aca3e1dfb19c32ff3e74f'), 'user_id': 'user7', 'userna
```

Rating

```
In [24]: # database
db = conn.AnimeDatabase

# Created or Switched to collection
collection = db.ratingcollection

#Reading data from csv file which includes Rating details
RatingTable = pd.read_csv("ratinglist.csv",encoding='latin1')
RatingTable

# Insert Data by converting to JSON
collection.insert_many(RatingTable.to_dict("records"))

# Printing the data inserted
data_inserted = collection.find()
for record in data_inserted:
    print(record)
```

```
{'_id': ObjectId('5e8aca461dfb19c32ff3eb93'), 'userid': 'user1', 'title':
'Zombie Clay Animation: Stuck!!', 'Rating': 2}
{'_id': ObjectId('5e8aca461dfb19c32ff3eb94'), 'userid': 'user1', 'title':
'Yami Shibai', 'Rating': 2}
{'_id': ObjectId('5e8aca461dfb19c32ff3eb95'), 'userid': 'user2', 'title':
'Yuusha ni Narenakatta Ore wa Shibushibu Shuushoku wo Ketsui Shimashit
a.', 'Rating': 5}
{'_id': ObjectId('5e8aca461dfb19c32ff3eb96'), 'userid': 'user3', 'title':
'Yuuki Yuuna wa Yuushabu Shozoku 3', 'Rating': 9}
{'_id': ObjectId('5e8aca461dfb19c32ff3eb97'), 'userid': 'user4', 'title':
'Yuu_Yuu_Hakusho: Eizou Hakusho', 'Rating': 6}
{'_id': ObjectId('5e8aca461dfb19c32ff3eb98'), 'userid': 'user4', 'title':
'Xia Lan', 'Rating': 3}
{'_id': ObjectId('5e8aca461dfb19c32ff3eb99'), 'userid': 'user4', 'title':
'Wind: A Breath of Heart (TV)', 'Rating': 4}
{'_id': ObjectId('5e8aca461dfb19c32ff3eb9a'), 'userid': 'user5', 'title':
'Yuru Camp_ Specials', 'Rating': 2}
{'_id': ObjectId('5e8aca461dfb19c32ff3eb9b'), 'userid': 'user6', 'title':
'Yume-iroæ', 'Rating': 6}
```

Retrieving Anime data from social media (Twitter)

In [31]:
```
# importing libraries required for downloading data
import tweepy
import twitter

# keys for accesing twitter api
consumerKey = 'lsDkpS786UbLVbxkYOONbeik5'
consumerSecret = 'BhSSMMpwmc6KtFPXWVbzVQezJ1osNthgQHaNDxgrg6TzQhSNUy'
ACCESS_TOKEN = '2483851159-GSH3yLT4Ilon3fD6lfpAYZPRZCaGjP30iAlOQS3'
ACCESS_SECRET = 'j6WQUKvxVSNkKsPMoKv9zrqDvuERqD0sVloCBS1gOT5Vn'

auth = tweepy.OAuthHandler(consumer_key=consumerKey, consumer_secret=consum

#Connect to the Twitter API using the authentication
api = tweepy.API(auth)
```

In [32]:
```
# Retrieving the tweets with anime hashtag
results = []
search_term = "%anime%"

#Collecting tweets
for tweet in tweepy.Cursor(api.search, q=search_term, since='2019-04-06',un
            results.append(tweet)

results[0]
```

Out[32]:
```
Status(_api=<tweepy.api.API object at 0x1a24de5550>, _json={'created_at':
'Sun Apr 05 23:59:58 +0000 2020', 'id': 1246950742050684929, 'id_str': '1
246950742050684929', 'text': 'RT @blackbuIls: Mfers be like “ my life an
anime “ yeah one piece of shit', 'truncated': False, 'entities': {'hashta
gs': [], 'symbols': [], 'user_mentions': [{'screen_name': 'blackbuIls',
'name': '\u200f', 'id': 1150470440881676289, 'id_str': '115047044088176 2
89', 'indices': [3, 14]}], 'urls': []}, 'metadata': {'iso_language_code':
'en', 'result_type': 'recent'}, 'source': '<a href="http://twitter.com/do
wnload/iphone" rel="nofollow">Twitter for iPhone</a>', 'in_reply_to_statu
s_id': None, 'in_reply_to_status_id_str': None, 'in_reply_to_user_id': No
ne, 'in_reply_to_user_id_str': None, 'in_reply_to_screen_name': None, 'us
er': {'id': 771404318524268545, 'id_str': '771404318524268545', 'name':
'A⚡ ', 'screen_name': 'anaissalma', 'location': '', 'description': '', 'u
rl': None, 'entities': {'description': {'urls': []}}, 'protected': False,
'followers_count': 27, 'friends_count': 130, 'listed_count': 0, 'created_
at': 'Thu Sep 01 17:48:16 +0000 2016', 'favourites_count': 9938, 'utc_off
set': None, 'time_zone': None, 'geo_enabled': False, 'verified': False,
'statuses_count': 618, 'lang': None, 'contributors_enabled': False, 'is_t
ranslator': False, 'is_translation_enabled': False, 'profile_background_c
olor': 'F5F8FA', 'profile_background_image_url': None, 'profile_backgroun
```

In [41]:
```
def createDataFrame(tweets):

    DF = pd.DataFrame()
    DF['tweetID'] = [tweet.id for tweet in tweets]
    DF['tweetText'] = [tweet.text.encode('utf-8') for tweet in tweets]
    DF['tweetUser'] = [tweet.user.screen_name for tweet in tweets]
    DF['tweetUserLocation'] = [tweet.user.location for tweet in tweets]
    DF['tweetRetweetCt'] = [tweet.retweet_count for tweet in tweets]
    DF['tweetCreated'] = [tweet.created_at for tweet in tweets]
    DF['hashTags'] = [tweet.entities.get('hashtags') for tweet in tweets]

    return DF

#Passing the tweets list to the above function to create a DataFrame
AnimeTweetData = createDataFrame(results)
```

In [42]:
```
#Verifying the tweets data with anime
AnimeTweetData.head()
```

Out[42]:

	tweetID	tweetText	tweetUser	tweetUserLocation	tweetRetweetCt	twee
0	1246950742050684929	b'RT @blackbulls: Mfers be like \xe2\x80\x9c m...	anaissalma		669	20
1	1246950742050643970	b'@sauerclout_ Girl idk who that anime person is'	campaaliyah98		0	20
2	1246950741144559616	b"RT @izukuuu_shonen: Anime who doesn't hesita...	MacLoushien	The Internet	5585	20
3	1246950740050030593	b'RT @OliverJia1014: Japan is a country where ...	Wet_Paper	',:)	4297	20
4	1246950738720325632	b'RT @The5thLeaf: Aht aht, we not tolerating a...	kingfadedz	The Spade Kingdom♠	5	20

In [43]:
```
def df_to_json(df):
    json_list = df.to_json(orient='records')
    json_list = json.loads(json_list)
    return json_list

AnimeTweetData = df_to_json(AnimeTweetData)
```

In [44]:
```
AnimeTweetData[0]
```

Out[44]:
```
{'tweetID': 1246950742050684929,
 'tweetText': 'RT @blackbuIls: Mfers be like “ my life an anime “ yeah on
e piece of shit',
 'tweetUser': 'anaissalma',
 'tweetUserLocation': '',
 'tweetRetweetCt': 669,
 'tweetCreated': 1586131198000,
 'hashTags': []}
```

Inserting Anime Tweets into MongoDB

In [48]:
```
# inserting data in MongoDB
from pymongo import MongoClient

try:
    conn = MongoClient('localhost', 27017)
    print("Connected successfully!!!")
except:
    print("Could not connect to MongoDB")

# database
db = conn.AnimeDatabase

# Created or Switched to collection
collection = db.animeTweetsCollection

# Insert Data
for data in AnimeTweetData:
    collection.insert_one(data)

# Printing the data inserted
data_inserted = collection.find()
for record in data_inserted:
    print(record)
```

```
Connected successfully!!!
{'_id': ObjectId('5e8e892a806e8fdd87abe5af'), 'tweetID': 1246950742050684
929, 'tweetText': 'RT @blackbuIls: Mfers be like “ my life an anime “ yea
h one piece of shit', 'tweetUser': 'anaissalma', 'tweetUserLocation': '',
'tweetRetweetCt': 669, 'tweetCreated': 1586131198000, 'hashTags': []}
{'_id': ObjectId('5e8e892a806e8fdd87abe5b0'), 'tweetID': 1246950742050643
970, 'tweetText': '@sauerclout_ Girl idk who that anime person is', 'twee
tUser': 'campaaliyah98', 'tweetUserLocation': '', 'tweetRetweetCt': 0, 't
weetCreated': 1586131198000, 'hashTags': []}
{'_id': ObjectId('5e8e892a806e8fdd87abe5b1'), 'tweetID': 1246950741144559
616, 'tweetText': "RT @izukuuu_shonen: Anime who doesn't hesitate to kill
a character https://t.co/xfiWBtwxSU", (https://t.co/xfiWBtwxSU",) 'tweetU
ser': 'MacLoushien', 'tweetUserLocation': 'The Internet', 'tweetRetweetC
t': 5585, 'tweetCreated': 1586131198000, 'hashTags': []}
{'_id': ObjectId('5e8e892a806e8fdd87abe5b2'), 'tweetID': 1246950740050030
593, 'tweetText': 'RT @OliverJia1014: Japan is a country where 98% of the
population is ethnically homogenous, yet the stories and characters shown
in anime h…', 'tweetUser': 'Wet_Paper', 'tweetUserLocation': "',:)", 'twe
etRetweetCt': 4297, 'tweetCreated': 1586131198000, 'hashTags': []}
```

Trending Topics using count of HashTags

In [13]:
```
# Create a dictionary
d = dict()

# Saving HashTag name and its count in all the tweets and saving as keys an
for tweet in range(0, len(results)):
    hashTag = results[tweet].entities.get('hashtags')
    for i in range(0, len(hashTag)):
        HashTag = hashTag[i]['text']
        if HashTag in d:
            d[HashTag] = d[HashTag] + 1
        else:
            d[HashTag] = 1

# Dictionary converted to a Dataframe
HashTag_DF = pd.DataFrame(list(d.items()),columns = ['HashTag','Count'])
HashTag_DF
```

Out[13]:

	HashTag	Count
0	ノイエ銀英伝	2
1	more	3
2	members	3
3	discord	3
4	Peaceful	3
...	...	...
485	NightcoreSongs	1
486	NightcoreMix	1
487	is_anime	1
488	forceofwill	1
489	forceofwilltcg	1

490 rows × 2 columns

In [14]:
```
#Sorting the dataframe as per the count
HashTag_DF = HashTag_DF.sort_values(by='Count', ascending=False)
HashTag_DF
```

Out[14]:

	HashTag	Count
8	anime	96
7	haikyuu	64
146	AniList	38
57	ギヴン	29
10	BLEACH	26
...	...	...
252	AnimeGifts	1
251	태용	1
250	TAEYONG	1
247	mewmew_new	1
489	forceofwilltcg	1

490 rows × 2 columns

MongoDB Query

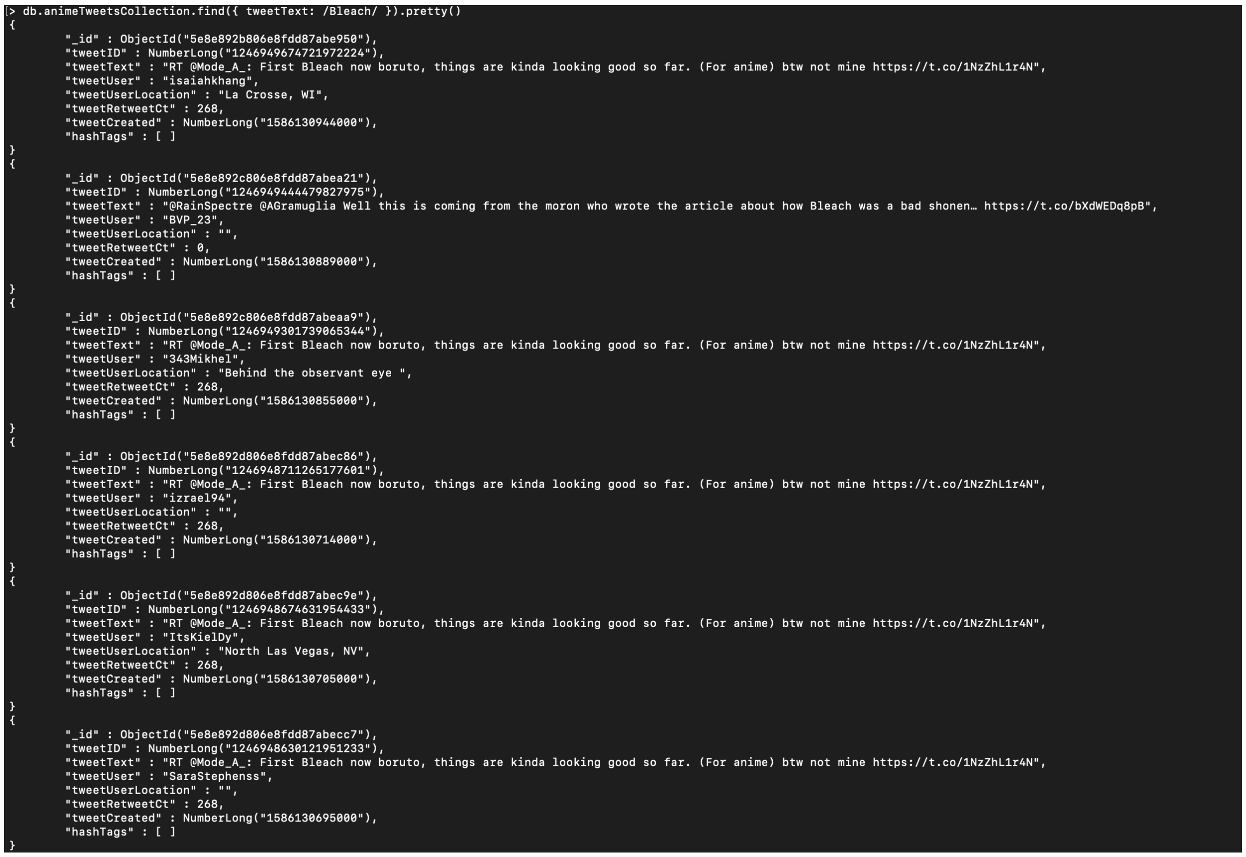
Find tweets with any anime name hashtag

QUERY USED:

- db.animeTweetsCollection.find({ tweetText: /Bleach/ }).pretty()

In [49]:
```python
from IPython.display import Image
Image("/Users/shashank/Pers/NEU 2nd Sem/DMDD/assignment3/Images/hashtag.png
```

Out[49]:



```
> db.animeTweetsCollection.find({ tweetText: /Bleach/ }).pretty()
{
        "_id" : ObjectId("5e8e892b806e8fdd87abe950"),
        "tweetID" : NumberLong("1246949674721972224"),
        "tweetText" : "RT @Mode_A_: First Bleach now boruto, things are kinda looking good so far. (For anime) btw not mine https://t.co/1NzZhL1r4N",
        "tweetUser" : "isaiahkhang",
        "tweetUserLocation" : "La Crosse, WI",
        "tweetRetweetCt" : 268,
        "tweetCreated" : NumberLong("1586130944000"),
        "hashTags" : [ ]
}
{
        "_id" : ObjectId("5e8e892c806e8fdd87abea21"),
        "tweetID" : NumberLong("1246949444479827975"),
        "tweetText" : "@RainSpectre @AGramuglia Well this is coming from the moron who wrote the article about how Bleach was a bad shonen… https://t.co/bXdWEDq8pB",
        "tweetUser" : "BVP_23",
        "tweetUserLocation" : "",
        "tweetRetweetCt" : 0,
        "tweetCreated" : NumberLong("1586130889000"),
        "hashTags" : [ ]
}
{
        "_id" : ObjectId("5e8e892c806e8fdd87abeaa9"),
        "tweetID" : NumberLong("1246949301739065344"),
        "tweetText" : "RT @Mode_A_: First Bleach now boruto, things are kinda looking good so far. (For anime) btw not mine https://t.co/1NzZhL1r4N",
        "tweetUser" : "343Mikhel",
        "tweetUserLocation" : "Behind the observant eye ",
        "tweetRetweetCt" : 268,
        "tweetCreated" : NumberLong("1586130855000"),
        "hashTags" : [ ]
}
{
        "_id" : ObjectId("5e8e892d806e8fdd87abec86"),
        "tweetID" : NumberLong("1246948711265177601"),
        "tweetText" : "RT @Mode_A_: First Bleach now boruto, things are kinda looking good so far. (For anime) btw not mine https://t.co/1NzZhL1r4N",
        "tweetUser" : "izrael94",
        "tweetUserLocation" : "",
        "tweetRetweetCt" : 268,
        "tweetCreated" : NumberLong("1586130714000"),
        "hashTags" : [ ]
}
{
        "_id" : ObjectId("5e8e892d806e8fdd87abec9e"),
        "tweetID" : NumberLong("1246948674631954433"),
        "tweetText" : "RT @Mode_A_: First Bleach now boruto, things are kinda looking good so far. (For anime) btw not mine https://t.co/1NzZhL1r4N",
        "tweetUser" : "ItsKielDy",
        "tweetUserLocation" : "North Las Vegas, NV",
        "tweetRetweetCt" : 268,
        "tweetCreated" : NumberLong("1586130705000"),
        "hashTags" : [ ]
}
{
        "_id" : ObjectId("5e8e892d806e8fdd87abecc7"),
        "tweetID" : NumberLong("1246948630121951233"),
        "tweetText" : "RT @Mode_A_: First Bleach now boruto, things are kinda looking good so far. (For anime) btw not mine https://t.co/1NzZhL1r4N",
        "tweetUser" : "SaraStephenss",
        "tweetUserLocation" : "",
        "tweetRetweetCt" : 268,
        "tweetCreated" : NumberLong("1586130695000"),
        "hashTags" : [ ]
}
```

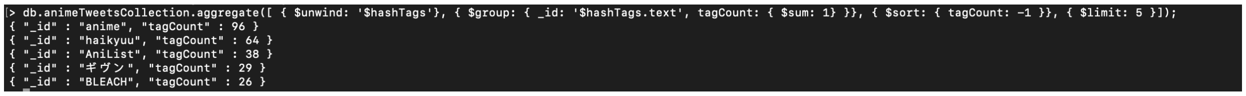
To retrieve top 5 trending topics in the database

QUERY USED:

- db.animeTweetsCollection.aggregate([{ *unwind* :'hashTags'}, {$group:{ _id:'$hashTags.text', tagCount: {$sum: 1}}}, { $sort: { tagCount: -1 }}, { $limit: 5 }]);

In [50]:
```
from IPython.display import Image
Image("/Users/shashank/Pers/NEU 2nd Sem/DMDD/assignment3/Images/Top5Trendin
```

Out[50]:



```
> db.animeTweetsCollection.aggregate([ { $unwind: '$hashTags'}, { $group: { _id: '$hashTags.text', tagCount: { $sum: 1} }}, { $sort: { tagCount: -1 }}, { $limit: 5 }]);
{ "_id" : "anime", "tagCount" : 96 }
{ "_id" : "haikyuu", "tagCount" : 64 }
{ "_id" : "AniList", "tagCount" : 38 }
{ "_id" : "ギヴン", "tagCount" : 29 }
{ "_id" : "BLEACH", "tagCount" : 26 }
```

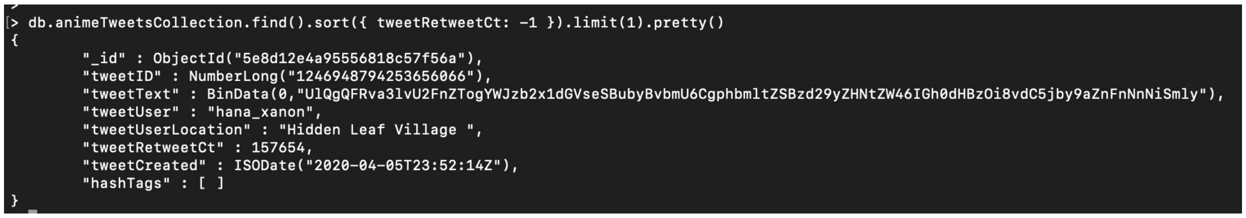
Popular Tweet

QUERY USED:

- db.animeTweetsCollection.find().sort({ tweetRetweetCt: -1 }).limit(1).pretty()

In [51]:
```
from IPython.display import Image
Image("/Users/shashank/Pers/NEU 2nd Sem/DMDD/assignment3/Images/PopularTwee
```

Out[51]:



```
> db.animeTweetsCollection.find().sort({ tweetRetweetCt: -1 }).limit(1).pretty()
{
        "_id" : ObjectId("5e8d12e4a95556818c57f56a"),
        "tweetID" : NumberLong("1246948794253656066"),
        "tweetText" : BinData(0,"UlQgQFRva3lvU2FnZTogYWJzb2x1dGVseSBubyBvbmU6CgphbmltZSBzd29yZHNtZW46IGh0dHBzOi8vdC5jby9aZnFnNnNiSmly"),
        "tweetUser" : "hana_xanon",
        "tweetUserLocation" : "Hidden Leaf Village ",
        "tweetRetweetCt" : 157654,
        "tweetCreated" : ISODate("2020-04-05T23:52:14Z"),
        "hashTags" : [ ]
}
```

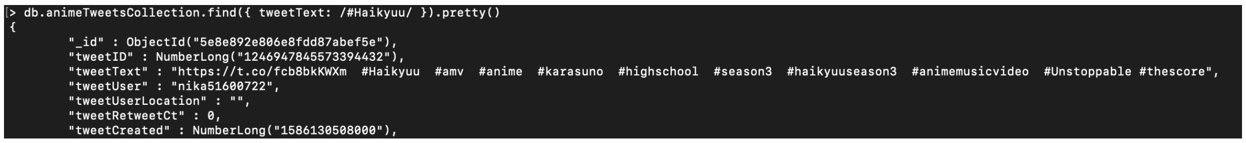
To find people having similar tweets

QUERY USED:

- db.animeTweetsCollection.find({ tweetText: /#Haikyuu/ }).pretty()

In [52]:
```
from IPython.display import Image
Image("/Users/shashank/Pers/NEU 2nd Sem/DMDD/assignment3/Images/similar.png
```

Out[52]:



```
> db.animeTweetsCollection.find({ tweetText: /#Haikyuu/ }).pretty()
{
        "_id" : ObjectId("5e8e892e806e8fdd87abef5e"),
        "tweetID" : NumberLong("1246947845573394432"),
        "tweetText" : "https://t.co/fcb8bkKWXm  #Haikyuu  #amv  #anime  #karasuno  #highschool  #season3  #haikyuuseason3  #animemusicvideo  #Unstoppable #thescore",
        "tweetUser" : "nika51600722",
        "tweetUserLocation" : "",
        "tweetRetweetCt" : 0,
        "tweetCreated" : NumberLong("1586130508000"),
```

AUDIT VALIDITY/ACCURACY

By using few commands, most of the unwanted null values were deleted from the above rows and columns which gives a report on valid and accuarate data.

AUDIT COMPLETNESS

In the real world, when a viewer requests for anime, a list of it will be displayed, similarly when we compare it with the above data, we get proper real-time data showing correct information for all the top-rated anime.

AUDIT CONSISTENCY/UNIFORMITY

The data which has been used in this assignment shows a uniform relationship since they are linked to each other by a common attribute.

REPORT

Source of data

Raw data on anime and users has been accessed from csv files

Entities being converted to Views

- Anime
- Viewers
- Production
- Genre
- Rating

Functions used

- createDataFrame This is used to retrieve data from tweets and convert it into a dataframe
- df_to_json This is used to convert the dataframe to json to insert into MongoDB

Code used to insert entity data into MongoDB

- from pymongo import MongoClient
- try:
- conn = MongoClient('localhost', 27017)
- print("Connected successfully!!!")
- except:
- print("Could not connect to MongoDB")
- db = conn.AnimeDatabase
- collection = db.animecollection

#Iterating over the complete anime details and generating Anime Table

- df1 = dataframe_entire.iloc[:, [6,7,8,11]]
- AnimeTable = dataframe_entire[[col for col in dataframe_entire.columns if col not in df1.columns]]
- collection.insert_many(AnimeTable.to_dict("records"))

CONCLUSION

Primary focus of this assignment is to learn how to convert an SQL database to NoSQL and to find interesting information from social media by interacting with its API.

CONTRIBUTION

Your contribution towards project. How much code did you write and how much you took from other site or some other source.

I contributed By Own: 40%
By External source: 60%

CITATIONS

Sources from where you have gained knowledge or used codes, data. It may include Web links, github links, code taken from somewhere etc.

- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html)
- https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json (https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json)
- https://beginanalyticsblog.wordpress.com/2018/02/07/twitter-data-analysis-using-python/ (https://beginanalyticsblog.wordpress.com/2018/02/07/twitter-data-analysis-using-python/)

LICENSE