CIS 9440:

Project - Phase 3

Author(s):

1. Chau Hoang
2. Tanay Mukherjee

In [41]:
```
# To install any package
# !pip install <package_name>
```

In [42]:
```
# All packages and libraries
import pandas as pd
import numpy as np
import json
from datetime import datetime
import glob, os
import pycountry

import warnings
warnings.filterwarnings("ignore")
```

Dataset 1: Netflix data

In [43]:
```
# Reading the netflix data
# URL: https://www.kaggle.com/shivamb/netflix-shows
nflx = pd.read_csv("C:\\Users\\its_t\\Documents\\CUNY\\Spring 2021\\CIS 9440 - Data Warehousing and Analytic
```

In [44]:
```
nflx.head()
```

Out[44]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	TV Show	3%	NaN	João Miguel, Bianca Comparato, Michel Gomes, R...	Brazil	August 14, 2020	2020	TV-MA	4 Seasons	International TV Shows, TV Dramas, TV Sci-Fi &...	In a future where the elite inhabit an island ...
1	s2	Movie	7:19	Jorge Michel Grau	Demián Bichir, Héctor Bonilla, Oscar Serrano, ...	Mexico	December 23, 2016	2016	TV-MA	93 min	Dramas, International Movies	After a devastating earthquake hits Mexico Cit...
2	s3	Movie	23:59	Gilbert Chan	Tedd Chan, Stella Chung, Henley Hii, Lawrence ...	Singapore	December 20, 2018	2011	R	78 min	Horror Movies, International Movies	When an army recruit is found dead, his fellow...
3	s4	Movie	9	Shane Acker	Elijah Wood, John C. Reilly, Jennifer Connelly...	United States	November 16, 2017	2009	PG-13	80 min	Action & Adventure, Independent Movies, Sci-Fi...	In a postapocalyptic world, rag-doll robots hi...
4	s5	Movie	21	Robert Luketic	Jim Sturgess, Kevin Spacey, Kate Bosworth, Aar...	United States	January 1, 2020	2008	PG-13	123 min	Dramas	A brilliant group of students become card-coun...

In [45]:
```
nflx.shape
```

Out[45]:
```
(7787, 12)
```

In [46]:
```
nflx.dtypes
```

Out[46]:
```
show_id         object
type            object
title           object
director        object
cast            object
country         object
date_added      object
release_year     int64
rating          object
duration        object
listed_in       object
description     object
dtype: object
```

In [47]:
```
nflx.isnull().sum()
```

Out[47]:
```
show_id            0
type               0
title              0
director        2389
cast             718
country          507
date_added        10
release_year       0
rating             7
duration           0
listed_in          0
description        0
dtype: int64
```

In [48]:
```
# Drop and fill missing values
nflx = nflx.drop(nflx[nflx['date_added'].isnull()].index,axis = 0)
nflx = nflx.fillna("Other")
```

In [49]:
```
nflx.isnull().sum().sum()
```

Out[49]:
```
0
```

Creating Dimensions (Netflix)

In [50]:
```
# 1. Create Content_nflx Dimension
content_nflx_dim = pd.DataFrame(nflx[['show_id','type','title','director','listed_in','description']])
content_nflx_dim.rename(columns={'listed_in': 'category'}, inplace=True)
content_nflx_dim.head()
```

Out[50]:

	show_id	type	title	director	category	description
0	s1	TV Show	3%	Other	International TV Shows, TV Dramas, TV Sci-Fi &...	In a future where the elite inhabit an island ...
1	s2	Movie	7:19	Jorge Michel Grau	Dramas, International Movies	After a devastating earthquake hits Mexico Cit...
2	s3	Movie	23:59	Gilbert Chan	Horror Movies, International Movies	When an army recruit is found dead, his fellow...
3	s4	Movie	9	Shane Acker	Action & Adventure, Independent Movies, Sci-Fi...	In a postapocalyptic world, rag-doll robots hi...
4	s5	Movie	21	Robert Luketic	Dramas	A brilliant group of students become card-coun...

In [51]:

```
# 2. Create Country Dimension
country_dim = nflx[['country']]
country_dim['country'] = country_dim['country'].str.split(", ")
country_dim = country_dim.explode('country').reset_index(drop=True)
country_dim = country_dim.drop_duplicates()
country_dim.insert(0, 'country_id', range(1000, 1000 + len(country_dim)))
country_dim['country']=country_dim['country'].str.replace(',','')
country_dim.head(125)
```

Out[51]:

	country_id	country
0	1000	Brazil
1	1001	Mexico
2	1002	Singapore
3	1003	United States
5	1004	Turkey
...	...	...
8322	1117	Panama
9111	1118	United Kingdom
9183	1119	Uganda
9367	1120	East Germany
9485	1121	Montenegro

122 rows × 2 columns

In [52]:

```
# 3. Date Dimension
date_rows = []
date_id = []
year = []
month = []
day = []
day_of_week = []
for date in nflx['date_added']:
    date_rows.append(datetime.strptime(date.replace(" ",""),"%B%d,%Y"))
for row in date_rows:
    date_id.append(row.strftime("%Y%m%d"))
    year.append(row.strftime("%Y"))
    month.append(row.strftime("%m"))
    day.append(row.strftime("%d"))
    day_of_week.append(row.weekday())
date_dim_nflx = pd.DataFrame({"date_id": date_id,
            "year": year,
              "month": month,
              "day": day,
                        "day_of_week": day_of_week})
date_dim_nflx.set_index("date_id")
date_dim_nflx['date_id'] = date_dim_nflx.date_id
date_dim_nflx
```

	date_id	year	month	day	day_of_week
0	20200814	2020	08	14	4
1	20161223	2016	12	23	4
2	20181220	2018	12	20	3
3	20171116	2017	11	16	3
4	20200101	2020	01	01	2
...	...	...	...	...	...
7772	20201019	2020	10	19	0
7773	20190302	2019	03	02	5
7774	20200925	2020	09	25	4
7775	20201031	2020	10	31	5

	date_id	year	month	day	day_of_week
7776	20200301	2020	03	01	6

In [53]:
```python
# 4. Create fact table 1: Content_Popularity_Netflix
nflx_fact = pd.DataFrame({'show_id': nflx['show_id'],
                          'date_id': date_id,
                          'country': nflx['country'],
                          'cast': nflx['cast'],
                          'rating': nflx['rating'],
                          'duration': nflx['duration']})
nflx_fact
```

Out[53]:

	show_id	date_id	country	cast	rating	duration
0	s1	20200814	Brazil	João Miguel, Bianca Comparato, Michel Gomes, R...	TV-MA	4 Seasons
1	s2	20161223	Mexico	Demián Bichir, Héctor Bonilla, Oscar Serrano, ...	TV-MA	93 min
2	s3	20181220	Singapore	Tedd Chan, Stella Chung, Henley Hii, Lawrence ...	R	78 min
3	s4	20171116	United States	Elijah Wood, John C. Reilly, Jennifer Connelly...	PG-13	80 min
4	s5	20200101	United States	Jim Sturgess, Kevin Spacey, Kate Bosworth, Aar...	PG-13	123 min
...	...	...	...	...	...	...
7782	s7783	20201019	Sweden, Czech Republic, United Kingdom, Denmar...	Imad Creidi, Antoinette Turk, Elias Gergi, Car...	TV-MA	99 min
7783	s7784	20190302	India	Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...	TV-14	111 min
7784	s7785	20200925	Other	Nasty C	TV-MA	44 min
7785	s7786	20201031	Australia	Adriano Zumbo, Rachel Khoo	TV-PG	1 Season
7786	s7787	20200301	United Kingdom, Canada, United States	Other	TV-MA	90 min

7777 rows × 6 columns

Dataset 2: Youtube data

In [54]:
```
# Reading the JSON file to get categories
# It is same for all countries to we can pick it for any one of them
# URL: https://www.kaggle.com/datasnaek/youtube-new?select=CA_category_id.json

category = open("C:\\Users\\its_t\\Documents\\CUNY\\Spring 2021\\CIS 9440 - Data Warehousing and Analytics\\
```

In [55]:
```
data = json.load(category)
```

In [56]:
```
data['items']
```

Out[56]:
```
[{'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/Xy1mB4_yLrHy_BmKmPBggty2mZQ"',
  'id': '1',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'Film & Animation',
   'assignable': True}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/UZ1oLIIz2dxIhO45ZTFR3a3NyTA"',
  'id': '2',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'Autos & Vehicles',
   'assignable': True}},
 {'kind': 'youtube#videoCategory',
  'etag': '"ld9biNPKjAjgjV7EZ4EKeEGrhao/nqRIq97-xe5XRZTxbknKFVe5Lmg"',
  'id': '10',
  'snippet': {'channelId': 'UCBR8-60-B28hp2BmDPdntcQ',
   'title': 'Music',
   'assignable': True}},
 {'kind': 'youtube#videoCategory',
```

In [57]:
```
ca_category = {}
for i in data['items']:
    ca_category[i['id']] = i['snippet']['title']
category_dim = pd.DataFrame(ca_category.items(), columns=['category_id', 'category'])
```

In [58]:
```
category_dim.head(10)
```

Out[58]:

	category_id	category
0	1	Film & Animation
1	2	Autos & Vehicles
2	10	Music
3	15	Pets & Animals
4	17	Sports
5	18	Short Movies
6	19	Travel & Events
7	20	Gaming
8	21	Videoblogging
9	22	People & Blogs

In [59]:

```
# There are multiple csv files. This code is to merge them all together
# URL: https://www.kaggle.com/datasnaek/youtube-new

file_list = []
os.chdir('C:\\Users\\its_t\\Documents\\CUNY\\Spring 2021\\CIS 9440 - Data Warehousing and Analytics\\Project

for file in os.listdir():
    if file.endswith('.csv'):
        df = pd.read_csv(file, sep=",", encoding='ISO-8859-1')
        # The new column below is basically the file name but gives away the country_code info in the name
        df['country_code'] = file
        file_list.append(df)

all_yt = pd.concat(file_list, ignore_index=True)

# Remove the extra details from the country_code column
all_yt['country_code']=all_yt['country_code'].str.replace('videos.csv','')
```

In [60]:
```
all_yt.head()
```

Out[60]:

	video_id	trending_date	title	channel_title	category_id	publish_time	tag
0	n1WpP7iowLc	17.14.11	Eminem - Walk On Water (Audio) ft. BeyoncÃ©	EminemVEVO	10	2017-11-10T17:00:03.000Z	Eminem|"Walk"|"On"|"Water"|"Aftermath/Shady/In
1	0dBIkQ4Mz1M	17.14.11	PLUSH - Bad Unboxing Fan Mail	iDubbbzTV	23	2017-11-13T17:00:00.000Z	plush|"bad unboxing"|"unboxing"|"fan mail"|"id
2	5qpjK5DgCt4	17.14.11	Racist Superman | Rudy Mancuso, King Bach & Le...	Rudy Mancuso	23	2017-11-12T19:05:24.000Z	racist superman|"rudy"|"mancuso"|"king"|"bach"
3	d380meD0W0M	17.14.11	I Dare You: GOING BALD!?	nigahiga	24	2017-11-12T18:01:41.000Z	ryan|"higa"|"higatv"|"nigahiga"|"i dare you"|"
4	2Vv-BfVoq4g	17.14.11	Ed Sheeran - Perfect (Official Music Video)	Ed Sheeran	10	2017-11-09T11:04:14.000Z	edsheeran|"ed sheeran"|"acoustic"|"live"|"cove

In [61]:
```
all_yt.isnull().sum()
```

Out[61]:
```
video_id                    0
trending_date               0
title                       0
channel_title               0
category_id                 0
publish_time                0
tags                        0
views                       0
likes                       0
dislikes                    0
comment_count               0
thumbnail_link              0
comments_disabled           0
ratings_disabled            0
video_error_or_removed      0
description             19478
country_code                0
dtype: int64
```

In [62]:
```
def clean(df):
    if df.isnull().sum().sum() > 0:
        df = df.fillna("Other")
    return df
```

In [63]:
```
youtube = clean(all_yt)
```

In [64]:
```
youtube.isnull().sum().sum()
```

Out[64]:
```
0
```

Creating Dimensions (YouTube)

In [65]:
```
# Country Dimension
# It is alredy created earlier for netflix and can be used again.
# We just need to add country name for each country_code for analysis later
youtube['country'] = youtube['country_code'].apply(lambda x: pycountry.countries.get(alpha_2=x).name)
```

In [66]:
```
youtube.head()
```

Out[66]:

	video_id	trending_date	title	channel_title	category_id	publish_time	tag
0	n1WpP7iowLc	17.14.11	Eminem - Walk On Water (Audio) ft. BeyoncÃ©	EminemVEVO	10	2017-11-10T17:00:03.000Z	Eminem|"Walk"|"On"|"Water"|"Aftermath/Shady/In.
1	0dBIkQ4Mz1M	17.14.11	PLUSH - Bad Unboxing Fan Mail	iDubbbzTV	23	2017-11-13T17:00:00.000Z	plush|"bad unboxing"|"unboxing"|"fan mail"|"id.
2	5qpjK5DgCt4	17.14.11	Racist Superman | Rudy Mancuso, King Bach & Le...	Rudy Mancuso	23	2017-11-12T19:05:24.000Z	racist superman|"rudy"|"mancuso"|"king"|"bach".
3	d380meD0W0M	17.14.11	I Dare You: GOING BALD!?	nigahiga	24	2017-11-12T18:01:41.000Z	ryan|"higa"|"higatv"|"nigahiga"|"i dare you"|".
4	2Vv-BfVoq4g	17.14.11	Ed Sheeran - Perfect (Official Music Video)	Ed Sheeran	10	2017-11-09T11:04:14.000Z	edsheeran|"ed sheeran"|"acoustic"|"live"|"cove.

In [67]:
```
yt_country = youtube['country'].unique()
yt_country
```

Out[67]:
```
array(['Canada', 'Germany', 'France', 'United Kingdom', 'India', 'Japan',
       'Korea, Republic of', 'Mexico', 'Russian Federation',
       'United States'], dtype=object)
```

In [68]:
```
selection_country = country_dim['country'].isin(yt_country)
check = country_dim[selection_country]
check
```

Out[68]:

	country_id	country
1	1001	Mexico
3	1003	United States
8	1006	India
22	1013	United Kingdom
27	1014	Japan
33	1017	Canada
53	1022	France
80	1026	Germany
1703	1075	United States
9111	1118	United Kingdom

In [69]:
```
missing_countries = ['Russia', 'South Korea']
selection_country = country_dim['country'].isin(missing_countries)
check = country_dim[selection_country]
check
```

Out[69]:

	country_id	country
30	1015	South Korea
406	1044	Russia

In [70]:
```
# So when we compare we realise that Russia and South Korea is there in the original country_dimension
# we created from the netflix data, but the name was Russia and not Russian federeation where as
# it was South Korea instead of Republic or Korea
```

In [71]:
```
# Replace Russian Federation in the Youtube data to Russia as it is in the country_dimension
# Replace Kore, Republic of in the Youtube data to South Korea as it is in the country_dimension
youtube['country'] = youtube['country'].str.replace("Russian Federation","Russia")
youtube['country'] = youtube['country'].str.replace("Korea, Republic of","South Korea")
```

In [72]:
```
# Let's verify now:
yt_country = youtube['country'].unique()
selection_country = country_dim['country'].isin(yt_country)
check = country_dim[selection_country]
check

# We can now see that Russia, South Korea  being picked by country_dimension which means our update above on
# dataframe worked perfectly
```

Out[72]:

	country_id	country
1	1001	Mexico
3	1003	United States
8	1006	India
22	1013	United Kingdom
27	1014	Japan
30	1015	South Korea
33	1017	Canada
53	1022	France
80	1026	Germany
406	1044	Russia
1703	1075	United States
9111	1118	United Kingdom

In [73]:
```
# Doing a exact match for each category id and category name as received from json file earlier
category_dim['category_id'] = pd.to_numeric(category_dim['category_id'])
youtube = youtube.merge(category_dim, on='category_id', how='left')
```

In [74]:
```
# 5. Create Content_YT Dimension
content_yt_dim = pd.DataFrame(youtube[['video_id','title','channel_title','thumbnail_link','tags', 'descript
                                       'category','comments_disabled', 'ratings_disabled', 'video_error_or_r
content_yt_dim.rename(columns={'listed_in': 'genre'}, inplace=True)
content_yt_dim.head()
```

thumbnail_link	tags	description	category	comments_disabled	ratings_disabled
ytimg.com/vi/n1WpP7iowLc/default.jpg	Eminem|"Walk"|"On"|"Water"|"Aftermath/Shady/In...	Eminem's new track Walk on Water ft. BeyoncÃ© ...	Music	False	False
ytimg.com/vi/0dBIkQ4Mz1M/default.jpg	plush|"bad unboxing"|"unboxing"|"fan mail"|"id...	STill got a lot of packages. Probably will las...	Comedy	False	False
i.ytimg.com/vi/5qpjK5DgCt4/default.jpg	racist superman|"rudy"|"mancuso"|"king"|"bach"...	WATCH MY PREVIOUS VIDEO â□¶ \n\nSUBSCRIBE â□º ...	Comedy	False	False

In [75]:
```
# Date Dimension for YouTube
date_rows = []
date_id_yt = []
year = []
month = []
day = []
day_of_week = []

for date in youtube['trending_date']:
    date_rows.append(datetime.strptime(date.replace(".",""),"%y%d%m"))
for row in date_rows:
    date_id_yt.append(row.strftime("%y%m%d"))
    year.append(row.strftime("%y"))
    month.append(row.strftime("%m"))
    day.append(row.strftime("%d"))
    day_of_week.append(row.weekday())
date_dim_yt = pd.DataFrame({"date_id": date_id_yt,
            "year": year,
              "month": month,
              "day": day,
                        "day_of_week": day_of_week})
date_dim_yt.set_index("date_id")
```

Out[75]:

	year	month	day	day_of_week
date_id				
171114	17	11	14	1
171114	17	11	14	1
171114	17	11	14	1
171114	17	11	14	1
171114	17	11	14	1
...	...	...	...	...
180614	18	06	14	3
180614	18	06	14	3
180614	18	06	14	3

	year	month	day	day_of_week
date_id				
180614	18	06	14	3
180614	18	06	14	3

375942 rows × 4 columns

In [76]:
```
date_dim_yt['date_id'] = '20' + date_dim_yt['date_id'].astype(str)
date_dim_yt['year'] = '20' + date_dim_yt['year'].astype(str)
date_dim_yt.head()
```

Out[76]:

	date_id	year	month	day	day_of_week
0	20171114	2017	11	14	1
1	20171114	2017	11	14	1
2	20171114	2017	11	14	1
3	20171114	2017	11	14	1
4	20171114	2017	11	14	1

In [77]:
```
# Merging dataframes for date dimension from netflix and youtube together to make it as one
date_dim = pd.concat([date_dim_nflx,date_dim_yt], ignore_index=True)
```

In [78]:
```
# Final date_dimension with duplciate records removed

# 6.
date_dim = pd.DataFrame.drop_duplicates(date_dim)
date_dim.head()
```

Out[78]:

	date_id	year	month	day	day_of_week
0	20200814	2020	08	14	4
1	20161223	2016	12	23	4
2	20181220	2018	12	20	3
3	20171116	2017	11	16	3
4	20200101	2020	01	01	2

In [79]:
```
date_dim.shape
```

Out[79]:
```
(1562, 5)
```

In [80]:
```
# 7. Create fact table 2: Content_Popularity_YouTube
yt_fact = pd.DataFrame({'video_id': youtube['video_id'],
                        'date_id': date_dim_yt['date_id'],
                        'country': youtube['country'],
                        'views': youtube['views'],
                        'likes': youtube['likes'],
                        'dislikes': youtube['dislikes'],
                        'comment_count': youtube['comment_count']
                       })
yt_fact
```

Out[80]:

	video_id	date_id	country	views	likes	dislikes	comment_count
0	n1WpP7iowLc	20171114	Canada	17158579	787425	43420	125882
1	0dBIkQ4Mz1M	20171114	Canada	1014651	127794	1688	13030
2	5qpjK5DgCt4	20171114	Canada	3191434	146035	5339	8181
3	d380meD0W0M	20171114	Canada	2095828	132239	1989	17518
4	2Vv-BfVoq4g	20171114	Canada	33523622	1634130	21082	85067
...	...	...	...	...	...	...	...
375937	BZt0qjTWNhw	20180614	United States	1685609	38160	1385	2657
375938	1h7KV2sjUWY	20180614	United States	1064798	60008	382	3936
375939	D6Oy4LfoqsU	20180614	United States	1066451	48068	1032	3992
375940	oV0zkMe1K8s	20180614	United States	5660813	192957	2846	13088
375941	ooyjaVdt-jA	20180614	United States	10306119	357079	212976	144795

375942 rows × 7 columns

Loading it to Google Big Query

In [81]:
```
# pip install google-cloud-bigquery
```

In [82]:
```
# pip show google-cloud-bigquery
```

In [83]:
```
# pip install pyarrow
```

In [84]:
```
# pip install google-cloud-bigquery-storage
```

In [85]:
```
# import libraries
import pandas as pd
from google.cloud import bigquery
from google.oauth2 import service_account
```

In [86]:
```
key_path = "C:/Users/its_t/Documents/CUNY/Spring 2021/CIS 9440 - Data Warehousing and Analytics/Project/Phas
```

In [87]:
```
credentials = service_account.Credentials.from_service_account_file(
    key_path, scopes=["https://www.googleapis.com/auth/cloud-platform"],
)
```

In [88]:
```
client = bigquery.Client(credentials = credentials, project = credentials.project_id)
```

In [89]:
```
client = bigquery.Client(credentials=credentials, project=credentials.project_id,)

dataset_id =  'cis9440-project-group12:phase_3'

dataset_ref = client.dataset(dataset_id)
job_config = bigquery.LoadJobConfig()
job_config.autodetect = True
job_config.write_disposition = "WRITE_TRUNCATE"

load_job = client.load_table_from_dataframe(content_nflx_dim, 'phase_3.content_nflx_dim',
                                            job_config=job_config)

print("Starting job {}".format(load_job))
print("Done!", 200)
```

```
Starting job <google.cloud.bigquery.job.load.LoadJob object at 0x0000027552AD3C18>
Done! 200
```

In [90]:
```
table_id = 'phase_3.content_nflx_dim'

table = client.get_table(table_id)

print(
    "Loaded {} rows and {} columns to {}".format(
        table.num_rows, len(table.schema), table_id
    )
)
```

```
Loaded 7777 rows and 6 columns to phase_3.content_nflx_dim
```

In [91]:
```
dataset_ref = client.dataset(dataset_id)
job_config = bigquery.LoadJobConfig()
job_config.autodetect = True
job_config.write_disposition = "WRITE_TRUNCATE"

load_job = client.load_table_from_dataframe(country_dim, 'phase_3.country_dim',
                                            job_config=job_config)

print("Starting job {}".format(load_job))
print("Done!", 200)
```

```
Starting job <google.cloud.bigquery.job.load.LoadJob object at 0x0000027552AC7F60>
Done! 200
```

In [92]:
```
table_id = 'phase_3.country_dim'

table = client.get_table(table_id)

print(
    "Loaded {} rows and {} columns to {}".format(
        table.num_rows, len(table.schema), table_id
    )
)
```

```
Loaded 122 rows and 2 columns to phase_3.country_dim
```

In [93]:
```
dataset_ref = client.dataset(dataset_id)
job_config = bigquery.LoadJobConfig()
job_config.autodetect = True
job_config.write_disposition = "WRITE_TRUNCATE"

load_job = client.load_table_from_dataframe(date_dim, 'phase_3.date_dim',
                                            job_config=job_config)

print("Starting job {}".format(load_job))
print("Done!", 200)
```

```
Starting job <google.cloud.bigquery.job.load.LoadJob object at 0x0000027552AC7B38>
Done! 200
```

In [94]:
```
table_id = 'phase_3.date_dim'

table = client.get_table(table_id)

print(
    "Loaded {} rows and {} columns to {}".format(
        table.num_rows, len(table.schema), table_id
    )
)
```

```
Loaded 1562 rows and 5 columns to phase_3.date_dim
```

In [95]:
```
dataset_ref = client.dataset(dataset_id)
job_config = bigquery.LoadJobConfig()
job_config.autodetect = True
job_config.write_disposition = "WRITE_TRUNCATE"

load_job = client.load_table_from_dataframe(nflx_fact, 'phase_3.nflx_fact',
                                            job_config=job_config)

print("Starting job {}".format(load_job))
print("Done!", 200)
```

```
Starting job <google.cloud.bigquery.job.load.LoadJob object at 0x0000027552ABF8D0>
Done! 200
```

In [96]:
```
table_id = 'phase_3.nflx_fact'

table = client.get_table(table_id)

print(
    "Loaded {} rows and {} columns to {}".format(
        table.num_rows, len(table.schema), table_id
    )
)
```

```
Loaded 7777 rows and 6 columns to phase_3.nflx_fact
```

In [97]:
```
dataset_ref = client.dataset(dataset_id)
job_config = bigquery.LoadJobConfig()
job_config.autodetect = True
job_config.write_disposition = "WRITE_TRUNCATE"

load_job = client.load_table_from_dataframe(content_yt_dim, 'phase_3.content_yt_dim',
                                            job_config=job_config)

print("Starting job {}".format(load_job))
print("Done!", 200)
```

```
Starting job <google.cloud.bigquery.job.load.LoadJob object at 0x0000027512714278>
Done! 200
```

In [98]:
```
table_id = 'phase_3.content_yt_dim'

table = client.get_table(table_id)

print(
    "Loaded {} rows and {} columns to {}".format(
        table.num_rows, len(table.schema), table_id
    )
)
```

```
Loaded 375942 rows and 10 columns to phase_3.content_yt_dim
```

In [99]:
```
dataset_ref = client.dataset(dataset_id)
job_config = bigquery.LoadJobConfig()
job_config.autodetect = True
job_config.write_disposition = "WRITE_TRUNCATE"

load_job = client.load_table_from_dataframe(yt_fact, 'phase_3.yt_fact',
                                            job_config=job_config)

print("Starting job {}".format(load_job))
print("Done!", 200)
```

```
Starting job <google.cloud.bigquery.job.load.LoadJob object at 0x0000027512714470>
Done! 200
```

In [100]:
```
table_id = 'phase_3.yt_fact'

table = client.get_table(table_id)

print(
    "Loaded {} rows and {} columns to {}".format(
        table.num_rows, len(table.schema), table_id
    )
)
```

```
Loaded 375942 rows and 7 columns to phase_3.yt_fact
```

Run Sample queries

In [101]:
```
sql = """SELECT * FROM `cis9440-project-group12.phase_3.content_nflx_dim` LIMIT 100;"""
```

In [106]:

```
df = client.query(sql).to_dataframe()
df.head()
```

Out[106]:

	show_id	video_id	country
0	s487	hWLjYJ4BzvI	United Kingdom
1	s487	bNcj9iR956M	United Kingdom
2	s487	-KeFvjm_hcA	United Kingdom
3	s487	tQR5G3kvfNQ	United Kingdom
4	s487	VaGcPRMY5UM	United Kingdom

In [103]:

```
sql = """SELECT n.show_id, y.video_id, y.country FROM `cis9440-project-group12.phase_3.nflx_fact` n
          JOIN `cis9440-project-group12.phase_3.yt_fact` y ON
          n.date_id = y.date_id
          WHERE y.country IN ('United Kingdom', 'United States') LIMIT 10;"""
```

In [105]:

```
df_1 = client.query(sql).to_dataframe()
df_1.head()
```

Out[105]:

	show_id	video_id	country
0	s487	hWLjYJ4BzvI	United Kingdom
1	s487	bNcj9iR956M	United Kingdom
2	s487	-KeFvjm_hcA	United Kingdom
3	s487	tQR5G3kvfNQ	United Kingdom
4	s487	VaGcPRMY5UM	United Kingdom