In [1]:
```python
"""
Created on Mon Aug  2 13:08:28 2021

@author: HK
"""
import requests
import json
import csv
import os
import mysql.connector
import flask
```

In [2]:
```python
def get_json_data(query):
    response =  requests.get(query)
    if (response.status_code == 200):
        array = response.json()
        text = json.dumps(array)
        return (text)
    else:
        return ("Error")
```

In [3]:
```python
def provider_checker(series_ID,API_key):
    query = 'https://api.themoviedb.org/3/tv/'
    query += str(series_ID) + '/watch/providers?'
    query += 'api_key=' + API_key
    #query += '&locale=TR'

    text = get_json_data(query)
    if text != 'Error':
        dataset = json.loads(text)
        try:
            provider = dataset['results']['TR']['flatrate'][0]['provider_name']
            if provider == 'Amazon Prime Video':
                return True
        except:
            return False

```

In [4]:
```python
#Making the database connection
def connect_to_db():
    mydb = mysql.connector.connect(
        host="localhost",
        user="root",
        password="123456",
        database="themoviedb"
    )
    return mydb
```

In [5]:
```python
def insert_series_to_db(series_id, name, overview, number_of_seasons, number_of_episodes,
                popularity, vote_average,vote_count, first_air_date, last_air_date, status):
    mycursor = mydb.cursor()

    sql = "INSERT IGNORE INTO TV_SERIES (ID,Name,Overview,Number_of_Seasons,Number_of_Episodes,
    val = (series_id, name, overview, number_of_seasons, number_of_episodes,
            popularity, vote_average,vote_count, first_air_date, last_air_date, status)
    mycursor.execute(sql, val)

    mydb.commit()
```

In [6]:
```python
def insert_person_to_db(cast_id, name, gender, known_for_department, popularity):
    mycursor = mydb.cursor()

    sql = "INSERT IGNORE INTO PERSON (ID,Name,Gender,Known_For_Department,Popularity) VALUES (%
    val = (cast_id, name, gender, known_for_department, popularity)
    mycursor.execute(sql, val)

    mydb.commit()
```

In [7]:
```
def insert_cast_to_db(series_id,person_id,character):
    mycursor = mydb.cursor()

    sql = "INSERT IGNORE INTO CAST (Series_ID,Person_ID,Type,Character_Name) VALUES (%s, %s, %s
    val = (series_id,person_id,'Cast',character)
    mycursor.execute(sql, val)

    mydb.commit()
```

In [8]:
```
def insert_crew_to_db(series_id,person_id):
    mycursor = mydb.cursor()

    sql = "INSERT IGNORE INTO CAST (Series_ID,Person_ID,Type) VALUES (%s, %s, %s)"
    val = (series_id,person_id,'Crew')
    mycursor.execute(sql, val)

    mydb.commit()
```

In [9]:
```
# Get every series from database
def get_every_series_from_db(mydb):
    mycursor = mydb.cursor()

    mycursor.execute("SELECT * FROM TV_SERIES")

    myresult = mycursor.fetchall()

    for x in myresult:
        print(x)
        print('')
```

In [10]:
```
# Get every person from database
def get_every_person_from_db(mydb):
    mycursor = mydb.cursor()

    mycursor.execute("SELECT * FROM PERSON")

    myresult = mycursor.fetchall()

    print("Number of results: ", len(myresult))

    for x in myresult:
        print(x)
```

In [11]:
```
# Get every cast from database
def get_every_cast_from_db(mydb):
    mycursor = mydb.cursor()

    mycursor.execute("SELECT * FROM CAST")

    myresult = mycursor.fetchall()

    print(len(myresult))

    for x in myresult:
        print(x)
```

In [12]:
```
# Execute any query
def query_executer(query):
    mycursor = mydb.cursor()

    mycursor.execute(query)

    myresult = mycursor.fetchall()

    print("Number of results: ", len(myresult))

    for x in myresult:
        print(x)
```

In [13]:
```python
mydb = connect_to_db()
```

In [14]:
```python
#Setting the filtering query

prefix = 'https://api.themoviedb.org/3/discover/tv?'
API_key = 'b8f5ea4374a732cdb90a9457176f7034'
minimum_vote_count = '750'
minimum_vote_average = '8.3'
provider_name = 'Amazon%20Prime%20Video'
watch_region = 'TR'
sort_by = 'first_air_date.asc'
page = 1

query = prefix
query += 'api_key=' + API_key
query += '&vote_count.gte=' + minimum_vote_count
query += '&vote_average.gte=' + minimum_vote_average
query += '&provider_name=' + provider_name
query += '&watch_region=' + watch_region
query += '&page=' + str(page)
query += '&sort_by=' + sort_by
```

In [15]:
```python
#Getting series ids according to filter
text = get_json_data(query)
list_series_ids = [] # list of the series ids
if text != 'Error':
    dataset = json.loads(text)
    count = 0
    i = 0;
    while(count < 3):
        if(provider_checker(dataset['results'][i]['id'],API_key)):
            list_series_ids.append(dataset['results'][i]['id'])
            count += 1
        i += 1
        if(i == 20):
            page +=1
            query = 'https://api.themoviedb.org/3/discover/tv?'
            query += 'api_key=' + API_key
            query += '&vote_count.gte=' + minimum_vote_count
            query += '&vote_average.gte=' + minimum_vote_average
            query += '&provider_name=' + provider_name
            query += '&watch_region=' + watch_region
            query += '&page=' + str(page)
            query += '&sort_by=' + sort_by
            text = get_json_data(query)
            dataset = json.loads(text)
            i=0
```

In [16]:
```python
#Getting the asked information of the first 3 series
list_series_details = []
print("Related Links:")
for i in range (3):
    query = 'https://api.themoviedb.org/3/tv/'
    query += str(list_series_ids[i])
    query += '?api_key=' + API_key
    print(query)
    text = get_json_data(query)
    if text != 'Error':
        dataset = json.loads(text)
        series_id = dataset['id']
        name = dataset['name']
        overview = dataset['overview']
        number_of_seasons = dataset['number_of_seasons']
        number_of_episodes = dataset['number_of_episodes']
        popularity = dataset['popularity']
        vote_average = dataset['vote_average']
        vote_count = dataset['vote_count']
        first_air_date = dataset['first_air_date']
        last_air_date = dataset['last_air_date']
        status = dataset['status']

        insert_series_to_db(series_id, name, overview, number_of_seasons, number_of_episodes,
                popularity, vote_average,vote_count, first_air_date, last_air_date, status)


        result = [series_id, name, overview, number_of_seasons, number_of_episodes,
                    popularity, vote_average,vote_count, first_air_date, last_air_date, status]

        list_series_details.append(result)
```

```
Related Links:
https://api.themoviedb.org/3/tv/4087?api_key=b8f5ea4374a732cdb90a9457176f7034 (https://api.themovi
edb.org/3/tv/4087?api_key=b8f5ea4374a732cdb90a9457176f7034)
https://api.themoviedb.org/3/tv/1408?api_key=b8f5ea4374a732cdb90a9457176f7034 (https://api.themovi
edb.org/3/tv/1408?api_key=b8f5ea4374a732cdb90a9457176f7034)
https://api.themoviedb.org/3/tv/2316?api_key=b8f5ea4374a732cdb90a9457176f7034 (https://api.themovi
edb.org/3/tv/2316?api_key=b8f5ea4374a732cdb90a9457176f7034)
```

In [17]:
```python
get_every_series_from_db(mydb)
```

```
(1408, 'House', 'Dr. Gregory House, a drug-addicted, unconventional, misanthropic medical genius,
leads a team of diagnosticians at the fictional Princeton–Plainsboro Teaching Hospital in New Jers
ey.', 8, 177, 320.944, 8.6, 3814, datetime.date(2004, 11, 16), datetime.date(2012, 5, 21), 'Ende
d')

(2316, 'The Office', 'The everyday lives of office employees in the Scranton, Pennsylvania branch
of the fictional Dunder Mifflin Paper Company.', 9, 195, 172.149, 8.5, 1795, datetime.date(2005,
3, 24), datetime.date(2013, 5, 16), 'Ended')

(4087, 'The X-Files', "The exploits of FBI Special Agents Fox Mulder and Dana Scully who investiga
te X-Files: marginalized, unsolved cases involving paranormal phenomena. Mulder believes in the ex
istence of aliens and the paranormal while Scully, a skeptic, is assigned to make scientific analy
ses of Mulder's discoveries that debunk Mulder's work and thus return him to mainstream cases.", 1
1, 218, 130.652, 8.4, 1923, datetime.date(1993, 9, 10), datetime.date(2018, 3, 21), 'Ended')
```

In [18]:
```python
#Getting the cast information of the first 3 series
list_casts = []
list_crews = []
print("Related Links:")
for j in range (3):
    query = 'https://api.themoviedb.org/3/tv/'
    query += str(list_series_ids[j]) + '/credits'
    query += '?api_key=' + API_key
    print(query)
    text = get_json_data(query)
    if text != 'Error':
        list_cast = []
        list_casts.append(list_cast)
        list_crew = []
        list_crews.append(list_crew)
        dataset = json.loads(text)
        i=0
        while(1):
            try:
                cast_id = dataset['cast'][i]['id']
                name = dataset['cast'][i]['name']
                gender = dataset['cast'][i]['gender']
                known_for_department = dataset['cast'][i]['known_for_department']
                popularity = dataset['cast'][i]['popularity']
                character = dataset['cast'][i]['character']

                insert_person_to_db(cast_id, name, gender, known_for_department, popularity)
                insert_cast_to_db(list_series_ids[j],cast_id,character)

                result = [cast_id, name, gender, known_for_department, popularity]
                list_casts[j].append(result)
                i+=1
            except:
                break
        i=0
        while(1):
            try:
                crew_id = dataset['crew'][i]['id']
                name = dataset['crew'][i]['name']
                gender = dataset['crew'][i]['gender']
                known_for_department = dataset['crew'][i]['known_for_department']
                popularity = dataset['crew'][i]['popularity']

                insert_person_to_db(crew_id, name, gender, known_for_department, popularity)
                insert_crew_to_db(list_series_ids[j],crew_id)

                result = [crew_id, name, gender, known_for_department, popularity]
                list_crews[j].append(result)
                i+=1
            except:
                break
```

```
Related Links:
https://api.themoviedb.org/3/tv/4087/credits?api_key=b8f5ea4374a732cdb90a9457176f7034 (https://ap
i.themoviedb.org/3/tv/4087/credits?api_key=b8f5ea4374a732cdb90a9457176f7034)
https://api.themoviedb.org/3/tv/1408/credits?api_key=b8f5ea4374a732cdb90a9457176f7034 (https://ap
i.themoviedb.org/3/tv/1408/credits?api_key=b8f5ea4374a732cdb90a9457176f7034)
https://api.themoviedb.org/3/tv/2316/credits?api_key=b8f5ea4374a732cdb90a9457176f7034 (https://ap
i.themoviedb.org/3/tv/2316/credits?api_key=b8f5ea4374a732cdb90a9457176f7034)
```

```
In [19]:  get_every_person_from_db(mydb)
```

```
Number of results:  61
(2151, 'Amy Lippens', 1, 'Production', 0.84)
(2692, 'Robert Sean Leonard', 2, 'Acting', 5.902)
(4495, 'Steve Carell', 2, 'Acting', 9.399)
(4987, 'Omar Epps', 2, 'Acting', 2.578)
(9032, 'Bryan Singer', 2, 'Directing', 4.262)
(11678, 'Rainn Wilson', 2, 'Acting', 4.987)
(12214, 'Gillian Anderson', 1, 'Acting', 6.851)
(12256, 'Gerrit van der Meer', 2, 'Production', 0.6)
(12640, 'David Duchovny', 2, 'Acting', 2.262)
(12644, 'Mitch Pileggi', 2, 'Acting', 4.108)
(17697, 'John Krasinski', 2, 'Acting', 9.157)
(17835, 'Ricky Gervais', 2, 'Acting', 3.548)
(27105, 'Ed Helms', 2, 'Acting', 6.241)
(29009, 'Ken Kwapis', 2, 'Directing', 0.84)
(31514, 'Peter Jacobson', 2, 'Acting', 3.79)
(39189, 'Stephen Merchant', 2, 'Acting', 2.228)
(41419, 'Hugh Laurie', 2, 'Acting', 4.203)
(41422, 'Jesse Spencer', 2, 'Acting', 3.469)
(41424, 'Katie Jacobs', 1, 'Production', 0.6)
(45543, 'Paul Attanasio', 2, 'Writing', 0.628)
(51856, 'Jenna Fischer', 1, 'Acting', 5.399)
(51992, 'Odette Annable', 1, 'Acting', 9.314)
(58274, 'Howard Klein', 0, 'Production', 0.608)
(58403, 'Glen Morgan', 2, 'Writing', 2.689)
(84416, 'Gene Stupnitsky', 2, 'Writing', 0.84)
(84417, 'Lee Eisenberg', 2, 'Writing', 0.6)
(93285, 'Charlyne Yi', 1, 'Acting', 7.074)
(107770, 'B.J. Novak', 2, 'Acting', 3.671)
(114407, 'Garrett Lerner', 2, 'Writing', 0.6)
(125167, 'Mindy Kaling', 1, 'Acting', 2.77)
(169061, 'David Shore', 2, 'Writing', 1.851)
(1035869, 'Ben Silverman', 2, 'Production', 1.528)
(1213125, 'Paul Lieberstein', 2, 'Writing', 3.022)
(1213567, 'Russel Friend', 2, 'Writing', 1.176)
(1216625, 'Daniel Chun', 2, 'Writing', 0.6)
(1216630, 'Greg Daniels', 2, 'Writing', 0.6)
(1217228, 'Steve Hely', 0, 'Writing', 0.6)
(1220649, 'Jennifer Celotta', 1, 'Writing', 0.6)
(1221087, 'Dan Sterling', 2, 'Writing', 0.6)
(1223964, 'Peter Blake', 2, 'Production', 1.22)
(1223966, 'Thomas L. Moran', 2, 'Writing', 1.214)
(1223981, 'Dustin Paddock', 0, 'Writing', 0.6)
(1224008, 'Marcy G. Kaplan', 1, 'Production', 1.094)
(1225604, 'Warren Lieberstein', 0, 'Writing', 1.128)
(1226276, 'Charlie Grandy', 2, 'Writing', 0.652)
(1226308, 'Michael Schur', 2, 'Creator', 1.22)
(1229493, 'Aaron Shure', 2, 'Writing', 0.6)
(1230847, 'Justin Spitzer', 2, 'Writing', 0.98)
(1230848, 'Halsted Sullivan', 2, 'Writing', 0.6)
(1230854, 'Peter Ocko', 2, 'Writing', 1.128)
(1230857, 'Randy Cordray', 2, 'Production', 0.6)
(1230858, 'Kent Zbornak', 0, 'Production', 0.6)
(1230859, 'Teri Weinberg', 0, 'Production', 0.694)
(1488571, 'Elan Soltes', 0, 'Visual Effects', 0.694)
(1493983, 'Jon Ehrlich', 0, 'Sound', 0.677)
(1533790, 'Ira Hurvitz', 0, 'Directing', 1.442)
(1577074, 'Michael Lyle', 0, 'Sound', 0.6)
(1604953, 'Jason Derlatka', 2, 'Sound', 0.6)
(1604955, 'Cathy Crandall', 0, 'Costume & Make-Up', 0.6)
(1622443, 'Michael Baber', 0, 'Sound', 0.828)
(1646234, 'Steve De Leon', 0, 'Crew', 0.6)
```

In [20]:
```
get_every_cast_from_db(mydb)
```

```
62
(1408, 2151, 'Crew', None)
(1408, 2692, 'Cast', 'James Wilson')
(1408, 4987, 'Cast', 'Eric Foreman')
(1408, 9032, 'Crew', None)
(1408, 12256, 'Crew', None)
(1408, 31514, 'Cast', 'Chris Taub')
(1408, 41419, 'Cast', 'Gregory House')
(1408, 41422, 'Cast', 'Robert Chase')
(1408, 41424, 'Crew', None)
(1408, 45543, 'Crew', None)
(1408, 51992, 'Cast', 'Jessica Adams')
(1408, 93285, 'Cast', 'Chi Park')
(1408, 114407, 'Crew', None)
(1408, 169061, 'Crew', None)
(1408, 1213567, 'Crew', None)
(1408, 1223964, 'Crew', None)
(1408, 1223966, 'Crew', None)
(1408, 1223981, 'Crew', None)
(1408, 1224008, 'Crew', None)
(1408, 1488571, 'Crew', None)
(1408, 1493983, 'Crew', None)
(1408, 1533790, 'Crew', None)
(1408, 1577074, 'Crew', None)
(1408, 1604953, 'Crew', None)
(1408, 1604955, 'Crew', None)
(1408, 1622443, 'Crew', None)
(1408, 1646234, 'Crew', None)
(2316, 4495, 'Crew', None)
(2316, 11678, 'Cast', 'Dwight Schrute')
(2316, 17697, 'Cast', 'Jim Halpert')
(2316, 17835, 'Crew', None)
(2316, 27105, 'Cast', 'Andy Bernard')
(2316, 29009, 'Crew', None)
(2316, 39189, 'Crew', None)
(2316, 51856, 'Cast', 'Pam Beesly')
(2316, 58274, 'Crew', None)
(2316, 84416, 'Crew', None)
(2316, 84417, 'Crew', None)
(2316, 107770, 'Crew', None)
(2316, 125167, 'Crew', None)
(2316, 1035869, 'Crew', None)
(2316, 1213125, 'Crew', None)
(2316, 1216625, 'Crew', None)
(2316, 1216630, 'Crew', None)
(2316, 1217228, 'Crew', None)
(2316, 1220649, 'Crew', None)
(2316, 1221087, 'Crew', None)
(2316, 1225604, 'Crew', None)
(2316, 1226276, 'Crew', None)
(2316, 1226308, 'Crew', None)
(2316, 1229493, 'Crew', None)
(2316, 1230847, 'Crew', None)
(2316, 1230848, 'Crew', None)
(2316, 1230854, 'Crew', None)
(2316, 1230857, 'Crew', None)
(2316, 1230858, 'Crew', None)
(2316, 1230859, 'Crew', None)
(4087, 2692, 'Cast', 'Huveyscan Kamar')
(4087, 12214, 'Cast', 'Dana Scully')
(4087, 12640, 'Cast', 'Fox Mulder')
(4087, 12644, 'Cast', 'Walter Skinner')
(4087, 58403, 'Crew', None)
```

SQL DESIGN

SQL Queries

1-

```
SELECT * FROM PERSON ORDER BY Popularity DESC  LIMIT 10;
```

Link: https://drive.google.com/file/d/1qkF-2F9JpBgNs-ZkjTC1-9CMGFzm_Aha/view?usp=sharing (https://drive.google.com/file/d/1qkF-2F9JpBgNs-ZkjTC1-9CMGFzm_Aha/view?usp=sharing)

2-

```
SET @row_number = 0;
SELECT
    (@row_number:=@row_number + 1) AS 'Popularity Order',
    Name AS 'Actress/Actor Name',
    CASE
        WHEN GENDER = 1 THEN "Female"
        ELSE "Male"
    END AS Gender
 FROM PERSON ORDER BY Popularity DESC  LIMIT 10;
```

Link: https://drive.google.com/file/d/1eD29M-BZKkCqX1XsTEe25HhXukP71wqZ/view?usp=sharing (https://drive.google.com/file/d/1eD29M-BZKkCqX1XsTEe25HhXukP71wqZ/view?usp=sharing)

3-

```
SELECT p.Name, s.Name, c.Character_Name FROM PERSON p
INNER JOIN CAST c ON c.Person_ID = p.ID
INNER JOIN TV_SERIES s ON s.ID = c.Series_ID
Where p.ID IN(
SELECT p.ID FROM PERSON p
INNER JOIN CAST c ON c.Person_ID = p.ID
INNER JOIN TV_SERIES s ON s.ID = c.Series_ID
GROUP BY p.Name
HAVING Count(p.ID)>1);

#INSERT INTO CAST (Series_ID,Person_ID,Type,Character_Name) VALUES (4087,2692,'Cast','Huve
yscan Kamar');
```

Link: https://drive.google.com/file/d/1Yb6rcyKxEwU3vhN9HjhyxnAfOJRc9wLN/view?usp=sharing (https://drive.google.com/file/d/1Yb6rcyKxEwU3vhN9HjhyxnAfOJRc9wLN/view?usp=sharing)

4-

```
SET @row_number = 0;
SELECT
(SELECT (@row_number:=@row_number + 1)) AS 'Popularity Order',
(SELECT Name FROM PERSON WHERE Known_For_Department = 'Acting' ORDER BY Popularity DESC LI
MIT 1) AS Acting,
(SELECT Name FROM PERSON WHERE Known_For_Department = 'Directing' ORDER BY Popularity DESC
LIMIT 1) AS Directing,
(SELECT Name FROM PERSON WHERE Known_For_Department = 'Writing' ORDER BY Popularity DESC L
IMIT 1) AS Writing
UNION
SELECT
(SELECT (@row_number:=@row_number + 1)) AS 'Popularity Order',
(SELECT Name FROM PERSON WHERE Known_For_Department = 'Acting' ORDER BY Popularity DESC LI
MIT 1,1) AS Acting,
(SELECT Name FROM PERSON WHERE Known_For_Department = 'Directing' ORDER BY Popularity DESC
LIMIT 1,1) AS Directing,
(SELECT Name FROM PERSON WHERE Known_For_Department = 'Writing' ORDER BY Popularity DESC
 LIMIT 1,1) AS Writing
UNION
SELECT
(SELECT (@row_number:=@row_number + 1)) AS 'Popularity Order',
(SELECT Name FROM PERSON WHERE Known_For_Department = 'Acting' ORDER BY Popularity DESC LI
MIT 2,1) AS Acting,
(SELECT Name FROM PERSON WHERE Known_For_Department = 'Directing' ORDER BY Popularity DESC
LIMIT 2,1) AS Directing,
(SELECT Name FROM PERSON WHERE Known_For_Department = 'Writing' ORDER BY Popularity DESC
 LIMIT 2,1) AS Writing;
```

Link: https://drive.google.com/file/d/1xqd-nyq5czYFVSKEJYMEvo3ctkJ-BjAV/view?usp=sharing (https://drive.google.com/file/d/1xqd-nyq5czYFVSKEJYMEvo3ctkJ-BjAV/view?usp=sharing)