TVDB – Television Database

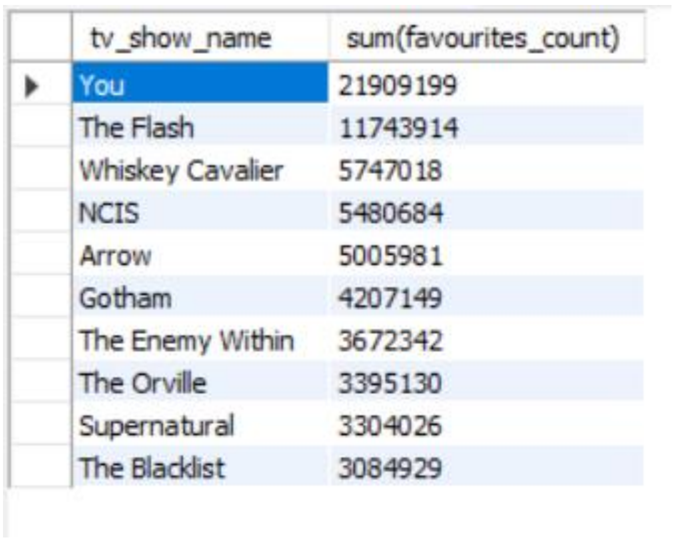
Use Cases

1. What are the TOP 10 most talked about TV shows on Twitter?

Query:

```
SELECT tv_show_name,sum(favourites_count)  FROM tweets group by tv_show_name

ORDER BY sum(favourites_count)  DESC LIMIT 10;
```

Result:



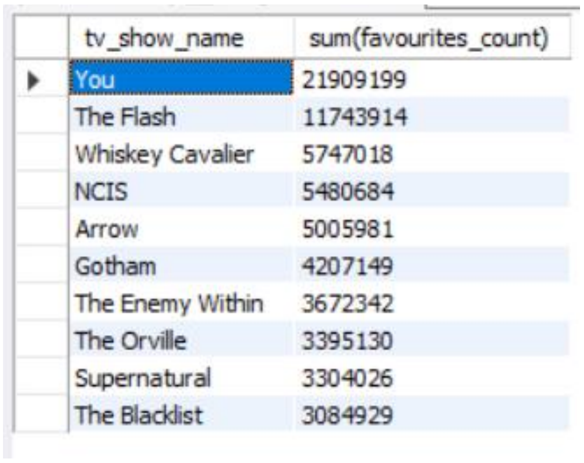
tv_show_name	sum(favourites_count)
You	21909199
The Flash	11743914
Whiskey Cavalier	5747018
NCIS	5480684
Arrow	5005981
Gotham	4207149
The Enemy Within	3672342
The Orville	3395130
Supernatural	3304026
The Blacklist	3084929

2. Which is the most anticipated TV show on Youtube?

Query:

```
select

        t.tv_show_name,views

        ,end_year

from youtube  y

join tv_show t

        on y.tv_show_name  = t.tv_show_name

order by

        views desc

limit 10;
```

Result:



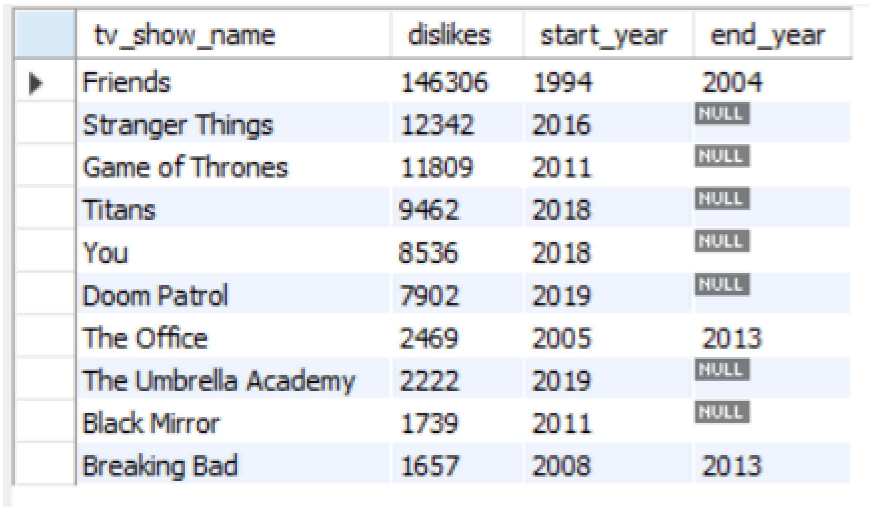
tv_show_name	sum(favourites_count)
You	21909199
The Flash	11743914
Whiskey Cavalier	5747018
NCIS	5480684
Arrow	5005981
Gotham	4207149
The Enemy Within	3672342
The Orville	3395130
Supernatural	3304026
The Blacklist	3084929

3. Which are the top 10 most hated (disliked) TV show on Youtube?

Query:

```
select t.tv_show_name

,dislikes

,start_year,end_year

from youtube  y

join tv_show t

        on y.tv_show_name  = t.tv_show_name

order by dislikes  desc limit  10;
```

Result:



tv_show_name	dislikes	start_year	end_year
Friends	146306	1994	2004
Stranger Things	12342	2016	NULL
Game of Thrones	11809	2011	NULL
Titans	9462	2018	NULL
You	8536	2018	NULL
Doom Patrol	7902	2019	NULL
The Office	2469	2005	2013
The Umbrella Academy	2222	2019	NULL
Black Mirror	1739	2011	NULL
Breaking Bad	1657	2008	2013

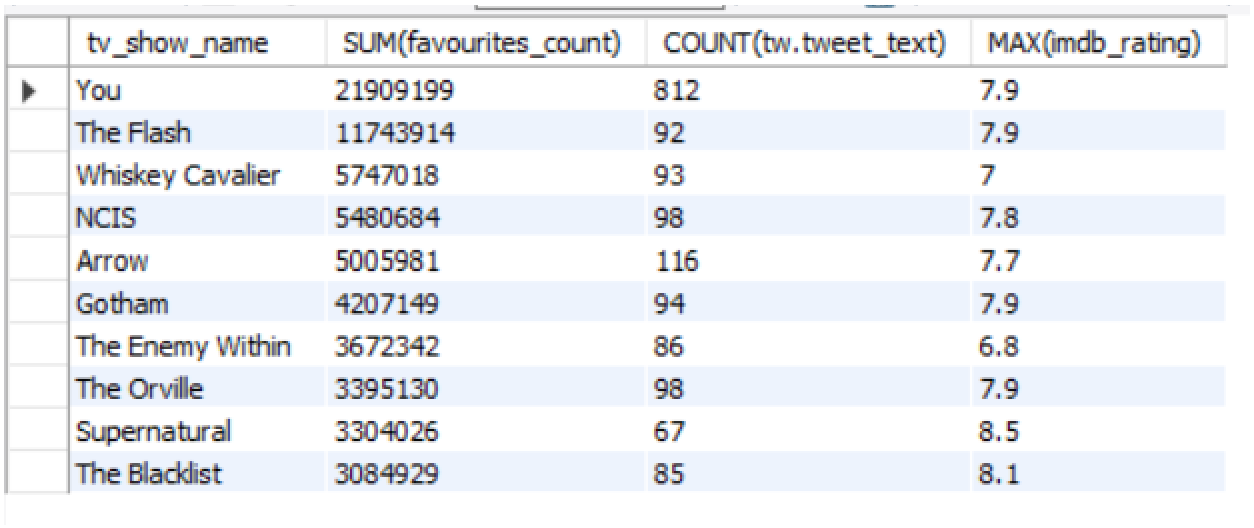
4. Are the highest rated movies on IMDB also the most talked about on Twitter?

Query:

```
select tv.tv_show_name,SUM(favourites_count),COUNT(tw.tweet_text),MAX(imdb_rating)

from tweets tw

join tv_show tv
```

```
on tv.tv_show_name=tw.tv_show_name

group by tv.tv_show_name

order by SUM(favourites_count)  desc

limit 10;
```

Result:



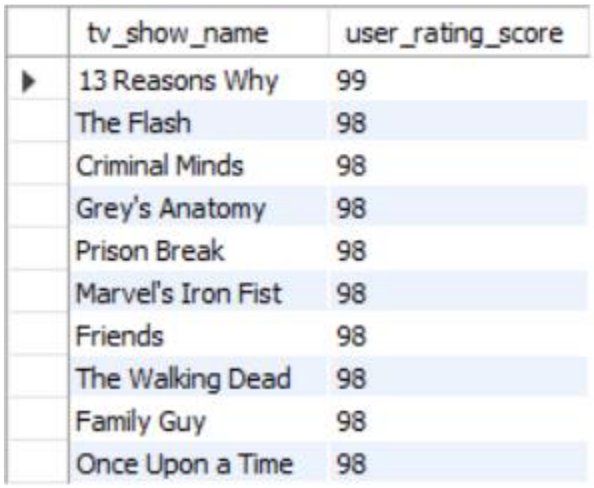
tv_show_name	SUM(favourites_count)	COUNT(tw.tweet_text)	MAX(imdb_rating)
You	21909199	812	7.9
The Flash	11743914	92	7.9
Whiskey Cavalier	5747018	93	7
NCIS	5480684	98	7.8
Arrow	5005981	116	7.7
Gotham	4207149	94	7.9
The Enemy Within	3672342	86	6.8
The Orville	3395130	98	7.9
Supernatural	3304026	67	8.5
The Blacklist	3084929	85	8.1

5. Top 10 TV shows by Netflix ratings

Query:

```
SELECT tv_show_name, user_rating_score

FROM netflix_ratings

ORDER BY user_rating_score  DESC LIMIT 10;
```

Result:



tv_show_name	user_rating_score
13 Reasons Why	99
The Flash	98
Criminal Minds	98
Grey's Anatomy	98
Prison Break	98
Marvel's Iron Fist	98
Friends	98
The Walking Dead	98
Family Guy	98
Once Upon a Time	98

6. Is there a correlation between number of seasons and popularity(favorite count)?

Query:

```
SELECT TV.TV_SHOW_NAME,COUNT(DISTINCT
TV.SEASON_NUM),SUM(TW.FAVOURITEs_COUNT),COUNT(TW.TWEET_TEXT)
```

```
FROM TWEETS TW JOIN EPISODES TV

ON TW.TV_SHOW_NAME=TV.TV_SHOW_NAME

GROUP BY TV.TV_SHOW_NAME;
```

Result:

TV_SHOW_NAME	COUNT(DISTINCT TV.SEASON_NUM)	SUM(TW.FAVOURITEs_COUNT)	COUNT(TW.TWEET_TEXT)
A Million Little Things	1	20231020	1003
American Gods	1	24491096	688
Arrow	7	760909112	17632
Black Mirror	4	9945322	1159
Breaking Bad	5	35195044	5766
Criminal Minds	10	829543587	20971
Dirty John	1	7861192	744
Doom Patrol	1	4812984	279
Friends	10	411496680	14868
Game of Thrones	7	171938415	8442
Gotham	5	403886304	9024
Homeland	7	110470752	6468
How to Get Away with Murder	5	70634325	6525
Lucifer	3	152779893	5187
NCIS	10	2022372396	36162
Peaky Blinders	4	35622480	2016
Riverdale	3	130871760	3888
Russian Doll	1	7916000	600
Sex Education	1	20888880	608
Shameless	9	108152198	6649
Stranger Things	2	11195044	1054
Suits	8	166603176	5828
Supernatural	10	991207800	20100
The Big Bang Theory	10	198462624	16864
The Blacklist	6	370191480	10200
The Crown	2	36097440	1420
The Enemy Within	1	7344684	172
The Expanse	3	43600032	2700
The Flash	5	1244854884	9752
The Good Doctor	2	77921725	3220
The OA	1	7586128	696
The Office	9	129989968	15416
The Orville	2	71297730	2058
The Umbrella Academy	1	18328820	960
The Walking Dead	9	356228650	11049
The Widow	1	6213280	544
This Is Us	3	138897115	4263
Titans	1	16987949	979
True Detective	3	23553816	1584

Vikings	5	88827357	5796
Whiskey Cavalier	1	5747018	93
You	1	219091990	8120

7. Are most talked about TV shows on Twitter also as popular (most views) on YouTube ?

Query:

SELECT T.TV_SHOW_NAME, SUM(VIEWS) AS 'VIEWS',COUNT(T.TWEET_TEXT),SUM(FAVOURITES_COUNT)

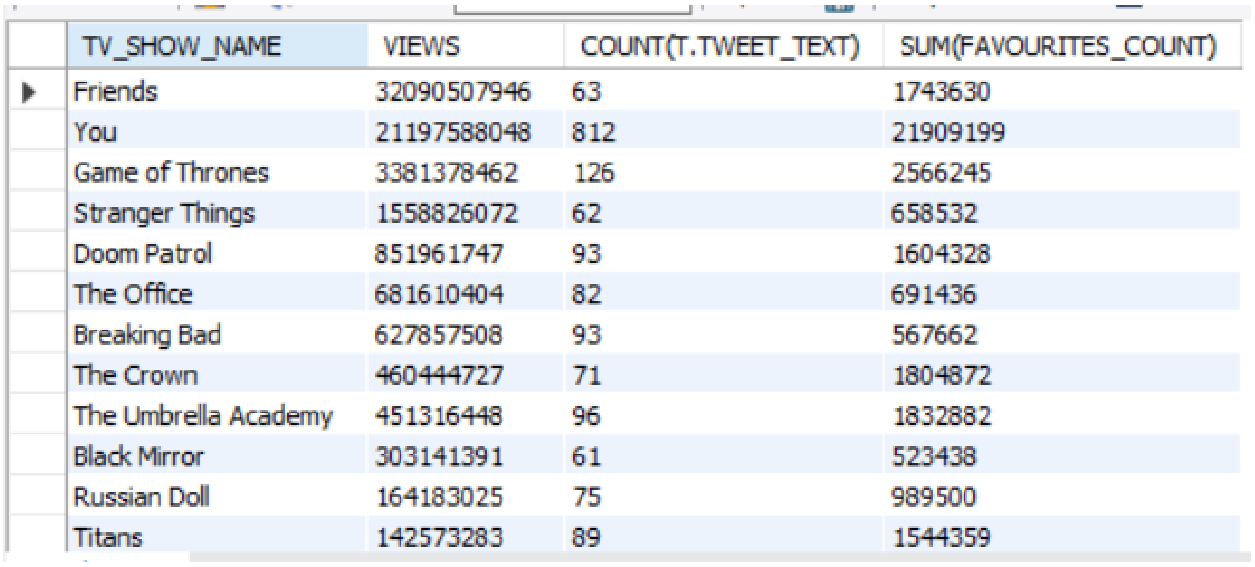
FROM TWEETS T

JOIN YOUTUBE Y ON T.TV_SHOW_NAME=Y.TV_SHOW_NAME

GROUP BY T.TV_SHOW_NAME

ORDER BY VIEWS DESC;

Result:



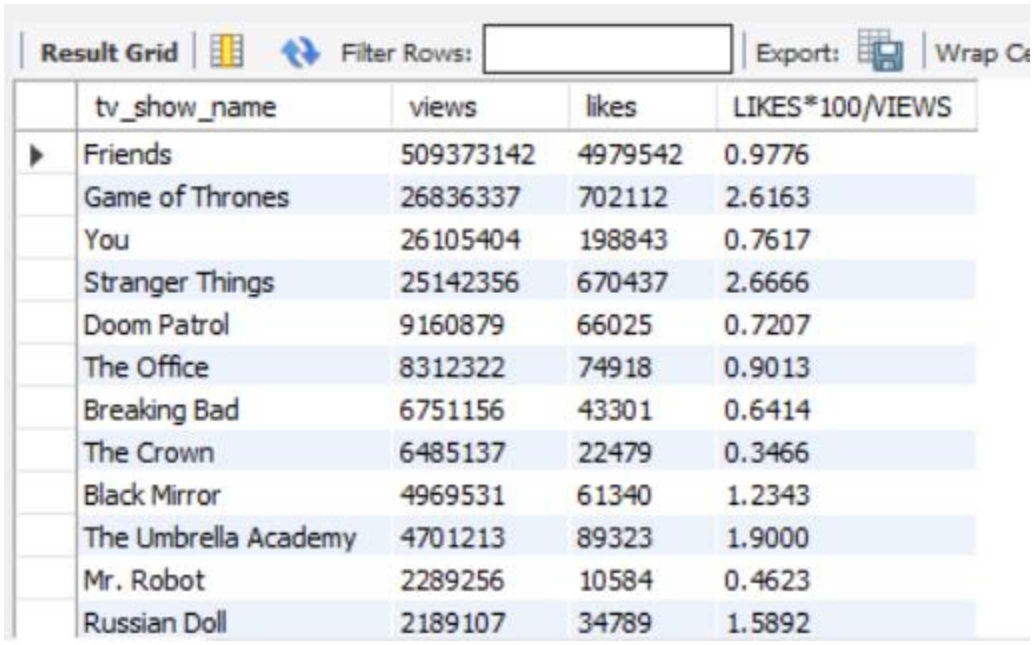
TV_SHOW_NAME	VIEWS	COUNT(T.TWEET_TEXT)	SUM(FAVOURITES_COUNT)
Friends	32090507946	63	1743630
You	21197588048	812	21909199
Game of Thrones	3381378462	126	2566245
Stranger Things	1558826072	62	658532
Doom Patrol	851961747	93	1604328
The Office	681610404	82	691436
Breaking Bad	627857508	93	567662
The Crown	460444727	71	1804872
The Umbrella Academy	451316448	96	1832882
Black Mirror	303141391	61	523438
Russian Doll	164183025	75	989500
Titans	142573283	89	1544359

8. Correlation between Views and Likes for Videos on YouTube?

Query:

SELECT tv_show_name, views, likes,LIKES*100/VIEWS FROM youtube ORDER BY Views DESC;

Result:



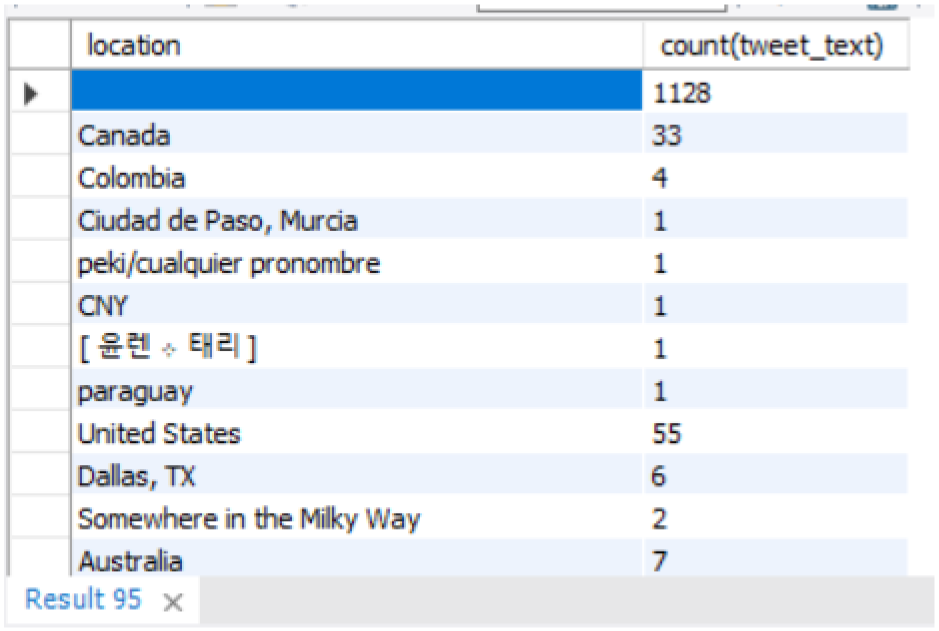
tv_show_name	views	likes	LIKES*100/VIEWS
Friends	509373142	4979542	0.9776
Game of Thrones	26836337	702112	2.6163
You	26105404	198843	0.7617
Stranger Things	25142356	670437	2.6666
Doom Patrol	9160879	66025	0.7207
The Office	8312322	74918	0.9013
Breaking Bad	6751156	43301	0.6414
The Crown	6485137	22479	0.3466
Black Mirror	4969531	61340	1.2343
The Umbrella Academy	4701213	89323	1.9000
Mr. Robot	2289256	10584	0.4623
Russian Doll	2189107	34789	1.5892

9. WHERE are the most tweets originating FROM (by TV shows)?

Query:

SELECT location, count(tweet_text) FROM tweets GROUP BY location;

Result:



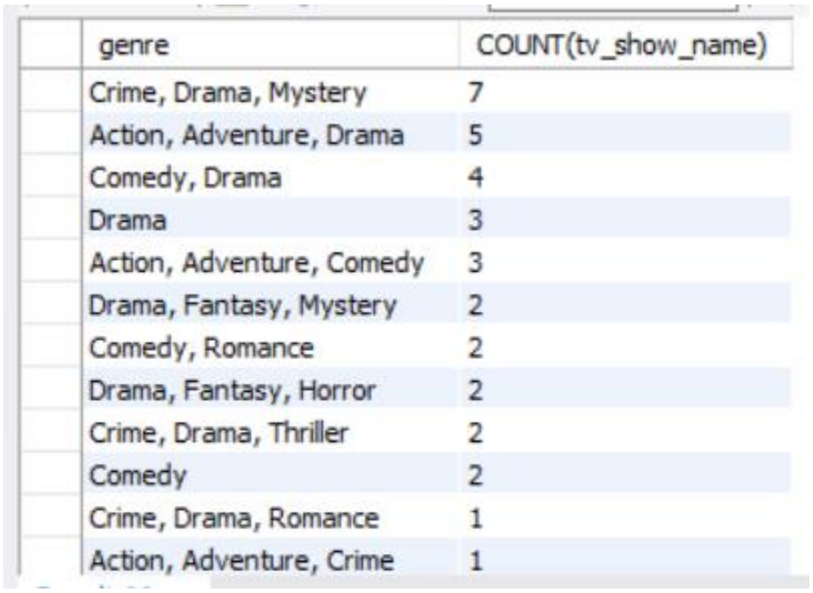
location	count(tweet_text)
	1128
Canada	33
Colombia	4
Ciudad de Paso, Murcia	1
peki/cualquier pronombre	1
CNY	1
[윤렌 ✧ 태리]	1
paraguay	1
United States	55
Dallas, TX	6
Somewhere in the Milky Way	2
Australia	7

10. What genres are the most popular?

Query:

SELECT genre, COUNT(tv_show_name) FROM tv_show GROUP BY genre ORDER BY COUNT(tv_show_name) DESC;

Result:



genre	COUNT(tv_show_name)
Crime, Drama, Mystery	7
Action, Adventure, Drama	5
Comedy, Drama	4
Drama	3
Action, Adventure, Comedy	3
Drama, Fantasy, Mystery	2
Comedy, Romance	2
Drama, Fantasy, Horror	2
Crime, Drama, Thriller	2
Comedy	2
Crime, Drama, Romance	1
Action, Adventure, Crime	1

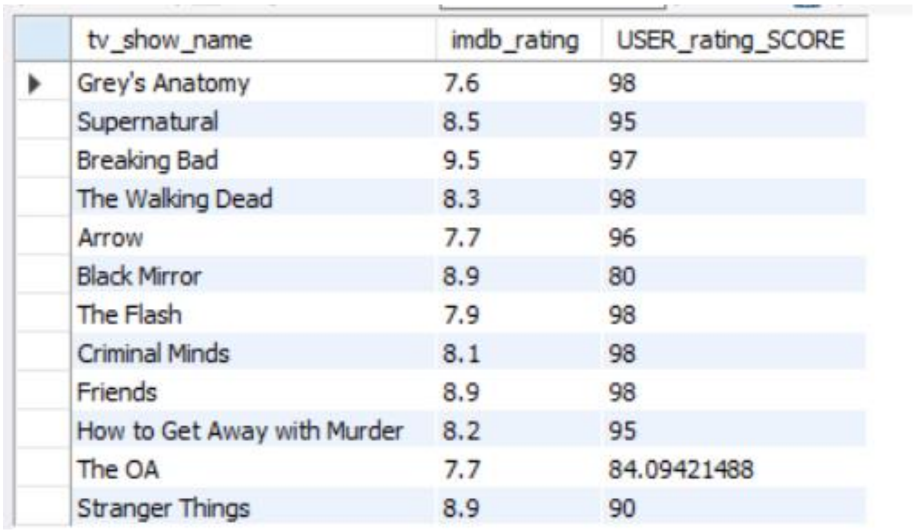
11. Compare IMDB ratings and Netflix Ratings

Query:

SELECT tv_show.tv_show_name, tv_show.imdb_rating, USER_rating_SCORE

FROM tv_show JOIN netflix_RATINGS ON tv_show.tv_show_name = netflix_RATINGS.tv_show_name;

Result:



tv_show_name	imdb_rating	USER_rating_SCORE
Grey's Anatomy	7.6	98
Supernatural	8.5	95
Breaking Bad	9.5	97
The Walking Dead	8.3	98
Arrow	7.7	96
Black Mirror	8.9	80
The Flash	7.9	98
Criminal Minds	8.1	98
Friends	8.9	98
How to Get Away with Murder	8.2	95
The OA	7.7	84.09421488
Stranger Things	8.9	90

Views for Use-cases

#Creation of Views

View 1

create view view_1 as

```
SELECT tv_show_name,sum(favourites_count)  FROM tweets group by tv_show_name

ORDER BY sum(favourites_count)  DESC LIMIT 10;
```

View 2

```
create view view_2 as

select

        t.tv_show_name,views

        ,end_year

from youtube  y

join tv_show t

        on y.tv_show_name  = t.tv_show_name

order by

        views desc

limit 10;
```

View 3

```
create view view_3 as

select t.tv_show_name

,dislikes

,start_year,end_year

from youtube  y

join tv_show t

        on y.tv_show_name  = t.tv_show_name

order by dislikes  desc limit 10;
```

View 4

```
create view view_4 as

select tv.tv_show_name,SUM(favourites_count),COUNT(tw.tweet_text),MAX(imdb_rating)

from tweets tw

join tv_show tv

on tv.tv_show_name=tw.tv_show_name

group by tv.tv_show_name

order by SUM(favourites_count)  desc

limit 10;
```

View 5

```
create view view_5 as

SELECT tv_show_name, user_rating_score

FROM netflix_ratings

ORDER BY user_rating_score DESC LIMIT 10;
```

View 6

```
create view view_6 as

SELECT TV.TV_SHOW_NAME,COUNT(DISTINCT
TV.SEASON_NUM),SUM(TW.FAVOURITEs_COUNT),COUNT(TW.TWEET_TEXT)

FROM TWEETS TW JOIN EPISODES TV

ON TW.TV_SHOW_NAME=TV.TV_SHOW_NAME

GROUP BY TV.TV_SHOW_NAME;
```

View 7

```
create view view_7 as

SELECT T.TV_SHOW_NAME, SUM(VIEWS) AS 'VIEWS',COUNT(T.TWEET_TEXT),SUM(FAVOURITES_COUNT)

FROM TWEETS T

JOIN YOUTUBE Y ON T.TV_SHOW_NAME=Y.TV_SHOW_NAME

GROUP BY T.TV_SHOW_NAME

ORDER BY VIEWS DESC;
```

View 8

```
create view view_8 as

SELECT tv_show_name, views, likes,LIKES*100/VIEWS FROM youtube ORDER BY Views DESC;
```

View 9

```
create view view_9 as

SELECT location, count(tweet_text) FROM tweets GROUP BY location;
```

View 10

```
create view view_10 as

SELECT genre, COUNT(tv_show_name) FROM tv_show GROUP BY genre ORDER BY COUNT(tv_show_name) DESC;
```

View 11

```
create view view_11 as

SELECT tv_show.tv_show_name, tv_show.imdb_rating, USER_rating_SCORE
```

```
FROM tv_show JOIN netflix_RATINGS ON tv_show.tv_show_name = netflix_RATINGS.tv_show_name;
```

View 12

```
#New Use Case

create view view_12 as

select tv_show_name,

case

        when isnull(end_year)  then YEAR(CURDATE())-start_year

        else end_year-start_year

end as total_duration_of_show_in_years

from tv_show;
```

Indexes

```
#index to search tweet_texts

create index tweet_text on tweets(tweet_text);


#Index to help with roll up and drill down on time data in the tweets table using the created_at column

create index tweet_created_at on tweets(created_at);


#creating an index on favorites_count in tweets table to improve view4 performance

create index favourites_count on tweets(favourites_count);


#index to help filter by imdb_rating

create index imdb_rating on tv_show(imdb_rating);


#creating an index on actors to easily retrieve the list of shows actors have starred in

create index actors on actors(actors);


#creating an index to improve view3 performance:

create index dislikes on youtube(dislikes);
```

#creating index on season number and episode number to improve the performance of the function

create index season_and_ep on episodes(season_num,episode_num);

#creating index to improve the performance of function 'descript'

create index descript on episodes(imdb_ratings,tv_show_name);

#creating an index on netflix_ratings for function that gets the netflix audience rating of shows

create index netflix_aud on netflix_ratings(tv_show_name,rating);

#creating an index on genre table to easily retrieve tv_shows when a user calls a procedure which selects tv_shows based on genre

create index genre_pref on genre(genre,tv_show_name)

Functions

Get the audience rating for a TV Show from the list of Netflix Shows

CREATE DEFINER=`root`@`localhost` FUNCTION `get_audience_rating_from_netflix`(show_name varchar(128)) RETURNS varchar(128) CHARSET utf8mb4

    DETERMINISTIC

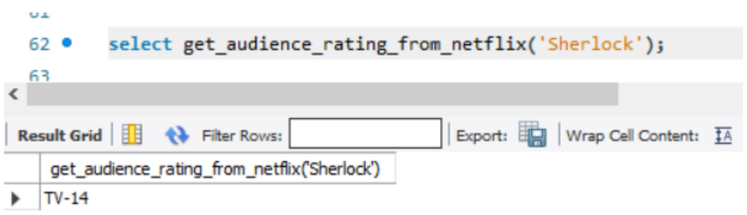
BEGIN

DECLARE rating_op varchar(32);

select rating into rating_op from netflix_ratings where tv_show_name=show_name;

RETURN rating_op;

END



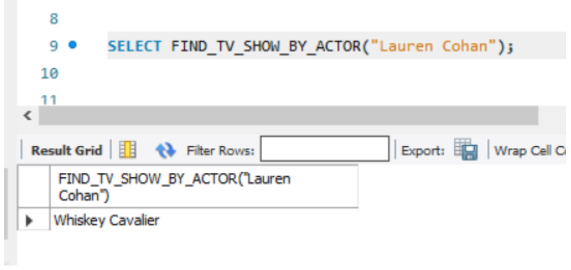
```
select get_audience_rating_from_netflix('Sherlock');
```

```
get_audience_rating_from_netflix('Sherlock')
TV-14
```

Find the latest TV Show an actor has starred in:

CREATE DEFINER=`root`@`localhost` FUNCTION `FIND_TV_SHOW_BY_ACTOR`(ACTOR_NAME VARCHAR(128)) RETURNS varchar(128) CHARSET utf8mb4
    DETERMINISTIC
BEGIN
DECLARE TV_SHOWS VARCHAR(128);

SELECT T.TV_SHOW_NAME INTO TV_SHOWS FROM ACTORS A JOIN TV_SHOW T ON A.TV_SHOW_NAME=T.TV_SHOW_NAME
WHERE ACTORS=ACTOR_NAME ORDER BY start_year desc limit 1;
RETURN TV_SHOWS;
END

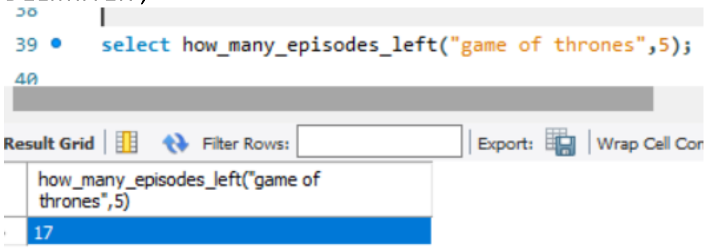


```
SELECT FIND_TV_SHOW_BY_ACTOR("Lauren Cohan");
```

```
FIND_TV_SHOW_BY_ACTOR("Lauren Cohan")
Whiskey Cavalier
```

Given a season number how many episodes do I have left to watch for a particular show?

CREATE DEFINER=`root`@`localhost` FUNCTION `how_many_episodes_left`(show_name varchar(128),seas_num int)
RETURNS int(11)
    DETERMINISTIC
BEGIN
DECLARE ep_count int;
select count(*) into ep_count
from episodes
where season_num>seas_num and tv_show_name=show_name;
RETURN ep_count;
END$$
DELIMITER ;



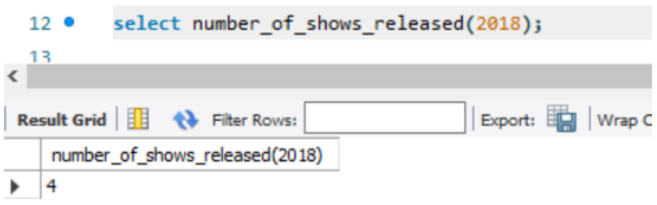
```
select how_many_episodes_left("game of thrones",5);
```



```
how_many_episodes_left("game of thrones",5)
17
```

How many TV Shows were released in a given year?

DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `number_of_shows_released`(yr int) RETURNS int(11)
    DETERMINISTIC
BEGIN
DECLARE no_of_shows int;
select count(*) into no_of_shows from tv_show where start_year=yr;
RETURN no_of_shows;
END$$
DELIMITER ;

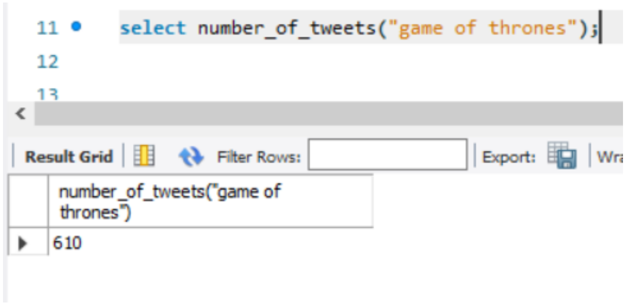


```
select number_of_shows_released(2018);
```

```
number_of_shows_released(2018)
4
```

How many tweets are out there about a given TV show?

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `number_of_tweets`(tv_show varchar(128)) RETURNS int(11)
    DETERMINISTIC
BEGIN
DECLARE tweet_count int;
select count(*) into tweet_count from tweets where tv_show_name = tv_show;
RETURN tweet_count;
END$$
DELIMITER ;
```

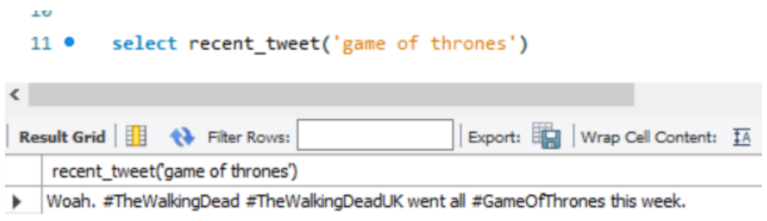


```
select number_of_tweets("game of thrones");
```

number_of_tweets("game of thrones")
610

What is the most recent tweet of a particular TV show?

```
CREATE DEFINER=`root`@`localhost` FUNCTION `recent_tweet`(show_name varchar(128)) RETURNS varchar(5000)
CHARSET utf8mb4
BEGIN
DECLARE t1 varchar(5000);
select tweet_text into t1 from (select lead(created_at) over (partition by tv_show_name order by created_at) as
recent_tweet, t.* from tweets t where tv_show_name = show_name) as recent_tv_show_tweet where recent_tweet is
null;
RETURN t1;
END$$
```

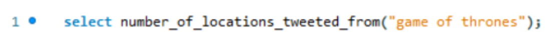


```
select recent_tweet('game of thrones')
```

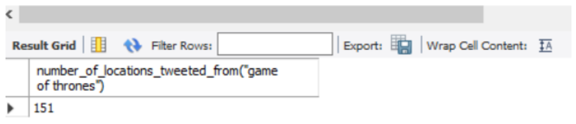
recent_tweet('game of thrones')
Woah. #TheWalkingDead #TheWalkingDeadUK went all #GameOfThrones this week.

How many distinct locations did a tweet originate from for a given TV Show?

```
CREATE DEFINER=`root`@`localhost` FUNCTION `number_of_locations_tweeted_from`(show_name varchar(128))
RETURNS int(11)
    DETERMINISTIC
BEGIN
DECLARE loc_count int;
select count(distinct location) into loc_count FROM tweets where tv_show_name=show_name;
RETURN loc_count;
END
```



```
select number_of_locations_tweeted_from("game of thrones");
```



```
number_of_locations_tweeted_from("game of thrones")
151
```

Give the Description of the TV Show

CREATE DEFINER=`root`@`localhost` FUNCTION `descript`(show_name varchar(128)) RETURNS varchar(5000) CHARSET utf8mb4

BEGIN

DECLARE a1 int;

DECLARE a2 int;

DECLARE t1 varchar(5000);

select avg(imdb_ratings) into a1 from episodes where tv_show_name = show_name group by tv_show_name; #episode avg for that tv_show

select avg(epi_avg_rating) from (select avg(imdb_ratings) as epi_avg_rating from episodes group by tv_show_name) as epi_avg into a2; #average of episode averages

select tv_show_description into t1 from tv_show where tv_show_name = show_name;

IF a1 > a2 THEN

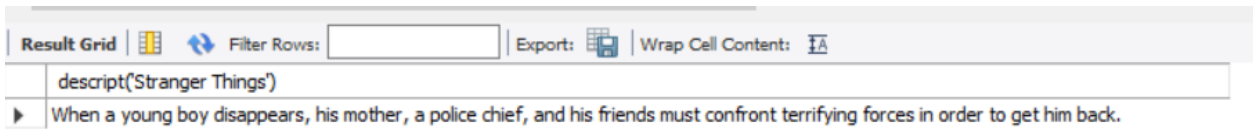
RETURN t1;

ELSE

RETURN 'Can wait for better Shows';

END IF;

END$$



```
descript('Stranger Things')
When a young boy disappears, his mother, a police chief, and his friends must confront terrifying forces in order to get him back.
```

Takes tv show name and judge it based on its ratings

CREATE DEFINER=`root`@`localhost` FUNCTION `rating`(show_name varchar(128)) RETURNS text CHARSET utf8mb4

BEGIN

DECLARE i int;

select imdb_rating into i from tv_show where tv_show_name = show_name;

IF i > 8.5 THEN

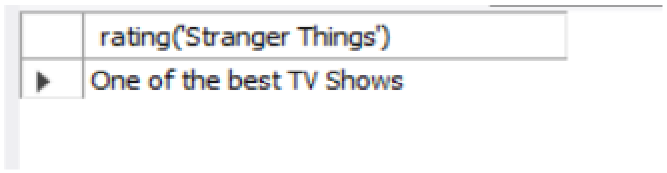
RETURN 'One of the best TV Shows';

ELSE

RETURN 'Sorry, TV Show is not cool enough';

END IF;

END$$



	rating('Stranger Things')
▶	One of the best TV Shows

Tells if it is a popular show based on the Youtube Views

CREATE DEFINER=`root`@`localhost` FUNCTION `popular`(show_name varchar(128)) RETURNS text CHARSET utf8mb4

BEGIN

DECLARE v1 int;

DECLARE v2 int;

SELECT Views INTO v1 FROM youtube where tv_show_name = show_name;

SELECT avg(Views) INTO v2 from youtube;

IF v1 > v2 THEN

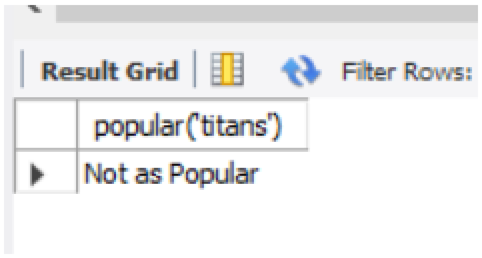
RETURN 'Popular Show';

ELSE

RETURN 'Not as Popular';

END IF;

END$$



	popular('titans')
▶	Not as Popular

PROCEDURES

#Procedure 1

#Creating TV show table
DELIMITER $$
USE `tvdb`$$
CREATE PROCEDURE tv_show_proc ()
BEGIN
insert into
	tv_show
select

```
		tv_show_name,
    tv_show_id,
    imdb_rating,
    votes,
    tv_show_description,
    runtime,
    genre,
    star_cast,
    CAST(start_year as UNSIGNED)
    ,CAST((case when ltrim(rtrim(end_year))=''  then null else end_year end) as UNSIGNED)
from stage_tv_show;
END$$
```

#Procedure 2

```
#Creating Episode Table
CREATE PROCEDURE `epi_proc` ()
BEGIN
insert into episodes
select
episode_name
,tv_show_name
,(case when imdb_ratings REGEXP '[0-9]' then imdb_ratings else NULL end) #using implicit conversion here from varchar to float
,runtime_in_mins
,director
,cast(ltrim(rtrim(substring(season,instr(season,'|')-2,2))) as unsigned) as 'season_num'
,cast(ltrim(rtrim(substring(season,LENGTH(season)-1,2))) as unsigned) as 'episode_num'
from stage_episode
where tv_show_name in (select tv_show_name from tv_show);
END$$
```

#Procedure 3

```
#Creating tweets table
CREATE PROCEDURE `tweets_proc` ()
BEGIN
insert into tweets
select
tv_show_name
,text
,cast(CONCAT(RIGHT(created_at,4),'-'
,case
		when substring(created_at,5,3) = 'Jan' then '01'
    when substring(created_at,5,3) = 'Feb' then '02'
    when substring(created_at,5,3) = 'Mar' then '03'
    when substring(created_at,5,3) = 'Apr' then '04'
    when substring(created_at,5,3) = 'May' then '05'
    when substring(created_at,5,3) = 'Jun' then '06'
    when substring(created_at,5,3) = 'Jul' then '07'
    when substring(created_at,5,3) = 'Aug' then '08'
    when substring(created_at,5,3) = 'Sep' then '09'
    when substring(created_at,5,3)  = 'Oct' then '10'
    when substring(created_at,5,3)  = 'Nov' then '11'
    when substring(created_at,5,3)  = 'Dec' then '12'
end #as 'month'
,'-',SUBSTRING(created_at,9,2), ' ', substring(created_at,12,8))  as datetime)
,favourites_count
,screen_name
,location
from stage_tweets st join tv_show ts on INSTR(st.text, replace(ts.tv_show_name,'  ',''));
END$$
```

#Procedure 4

```
#Creating youtube table
CREATE PROCEDURE `youtube_proc` ()
BEGIN
insert into youtube
 select
tv_show_name
,LTRIM(RTRIM(video_id))
,Video_title
,description
,views
,likes
,dislikes
 ,CASE
WHEN REGEXP_SUBSTR(left(video_duration,  INSTR(video_duration,'H')),  '[0-9]+') IS NOT NULL
THEN CAST(REGEXP_SUBSTR(left(video_duration,  INSTR(video_duration,'H')),  '[0-9]+') AS UNSIGNED)*3600
ELSE 0
END
+
CASE
                WHEN INSTR(video_duration,  'H') > 0
                        THEN CASE
                                        WHEN REGEXP_SUBSTR(SUBSTRING(video_duration,  INSTR(video_duration,  'H')
+ 1, INSTR(video_duration,  'M') - 1), '[0-9]+') IS NOT NULL
                                                THEN CAST(REGEXP_SUBSTR(SUBSTRING(video_duration,
INSTR(video_duration,  'H') + 1, INSTR(video_duration,  'M') - 1), '[0-9]+') AS UNSIGNED) * 60
                                        ELSE 0
                                        END
                ELSE CASE
                                WHEN REGEXP_SUBSTR(SUBSTRING(video_duration,  INSTR(video_duration,  'PT') + 1,
INSTR(video_duration,  'M') - 1), '[0-9]+') IS NOT NULL
                                        THEN CAST(REGEXP_SUBSTR(SUBSTRING(video_duration,
INSTR(video_duration,  'PT') + 1, INSTR(video_duration,  'M') - 1), '[0-9]+') AS UNSIGNED) * 60
                                ELSE 0
                                END
                END
+

CASE
                WHEN INSTR(video_duration,  'M') > 0
```

```
                              THEN CASE
                                              WHEN REGEXP_SUBSTR(SUBSTRING(video_duration,  INSTR(video_duration,  'M')
+ 1, INSTR(video_duration,  'S') - 1), '[0-9]+') IS NOT NULL
                                                      THEN CAST(REGEXP_SUBSTR(SUBSTRING(video_duration,
INSTR(video_duration,  'M') + 1, INSTR(video_duration,  'S') - 1), '[0-9]+') AS UNSIGNED)
                                              ELSE 0
                                              END
                      ELSE CASE
                                      WHEN REGEXP_SUBSTR(SUBSTRING(video_duration,  INSTR(video_duration,  'PT') + 1,
INSTR(video_duration,  'S') - 1), '[0-9]+') IS NOT NULL
                                              THEN CAST(REGEXP_SUBSTR(SUBSTRING(video_duration,
INSTR(video_duration,  'PT') + 1, INSTR(video_duration,  'S') - 1), '[0-9]+') AS UNSIGNED)
                                      ELSE 0
                                      END
END 'SECONDS'
 from stage_youtube
 where tv_show_name in (select tv_show_name from tv_show);
END$$
```

#Procedure 5

```
#Creating youtube comments table
CREATE PROCEDURE `youtube_comments_proc`  ()
BEGIN
insert into youtube_comments
select YC.VideoID,YC.COMMENTS from stage_youtube_comments YC
JOIN YOUTUBE Y ON ltrim(rtrim(Y.VIDEO_ID))=ltrim(rtrim(YC.VIDEOID));
END$$
```

#Procedure 6

```
#Separating Actor table
CREATE PROCEDURE `actors_proc` ()
BEGIN
insert into actors
WITH recursive tmp(tv_show_name,  actor, star_cast) AS
(
  SELECT
    tv_show_name,
    LEFT(star_cast, Locate(',', concat(star_cast ,',')) - 1),
    Insert(star_cast, 1, Locate(',', concat(star_cast ,',')), '')
  FROM tv_show
  UNION all

  SELECT
    tv_show_name,
    LEFT(star_cast, Locate(',', concat(star_cast ,',')) - 1),
    Insert(star_cast, 1, Locate(',', concat(star_cast ,',')), '')
  FROM tmp
  WHERE
    star_cast>''
)
```

```
SELECT
    tv_show_name,
    actor
FROM tmp
ORDER BY tv_show_name;

UPDATE actors
   SET actors = TRIM(BOTH '[' FROM actors);
UPDATE actors
   SET actors = TRIM(BOTH ']' FROM actors);
UPDATE actors
   SET actors = TRIM(BOTH '"' FROM actors);

#Deleting Star Cast from TV Show table
ALTER TABLE tv_show
DROP COLUMN star_cast;
END$$
```

#Procedure 7

```
#Separating Genre table

CREATE PROCEDURE `genre` ()
BEGIN
insert into genre
WITH recursive gen(tv_show_name, genres, genre) AS
(
    SELECT
        tv_show_name,
        LEFT(genre, Locate(',', concat(genre ,',')) - 1),
        Insert(genre, 1, Locate(',', concat(genre ,',')), '')
    FROM tv_show
    UNION all

    SELECT
        tv_show_name,
        LEFT(genre, Locate(',', concat(genre ,',')) - 1),
        Insert(genre, 1, Locate(',', concat(genre ,',')), '')
    FROM gen
    WHERE
        genre>''
)
SELECT
    tv_show_name,
    genres
FROM gen
ORDER BY tv_show_name;

#Cleaning out white spaces
UPDATE genre SET genres = REPLACE(genres, ' ', '');

#Deleting Genre from TV Show table
```

```
ALTER TABLE tv_show
DROP COLUMN genre;
END$$
```

#Procedure 8

```
#Suggests TV Shows based on Genres
CREATE DEFINER=`root`@`localhost` PROCEDURE `genre_of_tv_show`(gen varchar(32))
BEGIN
select tv_show_name from genre where genres = gen;
END$$
```

#Procedure 9

```
#Actors of a tv show
CREATE DEFINER=`root`@`localhost` PROCEDURE `star_cast`(show_name varchar(32))
BEGIN
select actors from actors where tv_show_name = show_name;
END$$
```

#Procedure 10

```
#User Names and Location who tweeted about a tv show
CREATE DEFINER=`root`@`localhost` PROCEDURE `twitter_users`(show_name  varchar(32))
BEGIN
select screen_name, location from tweets where tv_show_name = show_name;
END$$
```