



CIS 556 DATABASE SYSTEMS

PROJECT REPORT

ANALYSIS OF NETFLIX DATASET

Project Goals:

The main goal of this project is to analyze and extract interesting facts from the dataset containing the list of movies and TV shows in the popular streaming service 'Netflix'. The dataset contains the data of the movies and TV shows last updated till the year 2022. This dataset provides information on all movies and TV shows available on Netflix including their titles, release dates, runtimes, cast, crews, production, IMDB score, and more.

The following key operations are performed in this project:

- **Data Transformation:** Converting the data into a suitable format to import into the PostgreSQL platform.
- **Data Modelling:** Design the Relational Schema with tables and relationships, including constraints if required. Create the Entity-Relationship (ER) model related to the dataset, representing efficient data organization and structure.
- **Querying:** Design and execute several PostgreSQL queries to derive interesting facts from the dataset. These queries are also used to manipulate and analyze the data, for example, top-rated movies in a particular period, best movies, and TV shows in a popular genre, etc.
- **Indexing:** Create indexes to the columns of the dataset to improve query performance and efficiency of data access, to easily retrieve the data for the queries executed.

Chosen Domain/Dataset:

- **Dataset Source:** [Netflix Top Rated Movies and TV Shows \(2022 updated\)](#)
- **Data Content:** The dataset comprises six tables, each offering different perspectives and details about the Netflix shows and movies.
 - **raw_titles.csv:** It contains details of Netflix movies and TV shows, including title, type, release year, age certification, runtime, genres, production countries, seasons, IMDb score, and votes.
 - **raw_credits.csv:** This file contains information on the cast and crew of each movie or TV show in the dataset including name, character, and role.
 - **Best Shows Netflix.csv:** This file contains a list of the best TV shows on Netflix, as determined by their IMDb score and number of votes.
 - **Best Show by Year Netflix.csv:** This file contains the best TV show by year, as determined by IMDb score and number of votes.
 - **Best Movies Netflix.csv:** This file contains the best movies on Netflix according to the IMDb score and number of votes.
 - **Best Movies by Year Netflix.csv:** The file contains the best movie by year as determined by IMDb score and number of votes.

Decisions Made in the Design Stage:

- **Data Population:** Populated the database directly by utilizing the import feature available in PostgreSQL.
- **Key Constraints:** Applied Primary and Foreign Key constraints to enforce relationships and to ensure data accuracy.
- **Data Type Optimization:** Refined datatypes for appropriate columns, based on the data nature, for efficient storage and data retrieval.
- **Column Renaming:** Renamed few columns to minimize ambiguity and improve data retrieval efficiency.
- **Enforced Constraints:** Applied few constraints like UNIQUE, NOT NULL for appropriate columns to achieve consistent data.

- **Indexing:** Created index to a column to improve query performance.

Database Design:

- **Relational Schema:**

1. RAW_TITLES {ID, Index, Title, Type, Release_Year, Age_Certification, Runtime, Genres, Production_Countries, Seasons, Imdb_Id, Imdb_Score, Imdb_Votes}
 2. RAW_CREDITS {ID, Index, Person_Id, Name, Character, Actor}
 3. BESTMOVIESNETIFLIX {ID, Index, Title, Release_Year, Score, Number_Of_Votes, Duration, Main_Genre, Main_Production}
 4. BESTSHOWSNETFLIX {ID, Index, Title, Release_Year, Score, Number_Of_Votes, Duration, Number_Of_Seasons, Main_Genre, Main_Production}
 5. BESTMOVIESBYYEAR {ID, Index, Title, Release_Year, Score, Main_Genre, Main_Production}
 6. BESTSHOWSBYYEAR {Id, Index, Title, Release_Year, Score, Number_Of_Seasons, Main_Genre, Main_Production}
- Constraints:*
- raw_titles
 - ID is the primary key for the raw_titles table. It cannot be NULL.
 - raw_credits
 - ID is the foreign key of raw_credits derived from the relation raw_titles
 - $\delta(\pi_{ID}(\text{raw_credits})) \subseteq \pi_{ID}(\text{raw_titles})$
 - BestShowsByYear
 - ID is the foreign key of BestShowsByYear derived from the relation raw_titles
 - $\delta(\pi_{ID}(\text{BestShowsByYear})) \subseteq \pi_{ID}(\text{raw_titles})$
 - BestMoviesByYear
 - ID is the foreign key of BestMoviesByYear derived from the relation raw_titles
 - $\delta(\pi_{ID}(\text{BestMoviesByYear})) \subseteq \pi_{ID}(\text{raw_titles})$
 - BestShowsNetflix
 - ID is the foreign key of BestShowsNetflix derived from the relation raw_titles
 - $\delta(\pi_{ID}(\text{BestShowsNetflix})) \subseteq \pi_{ID}(\text{raw_titles})$
 - BestMoviesNetflix
 - ID is the foreign key of BestMoviesNetflix derived from the relation raw_titles
 - $\delta(\pi_{ID}(\text{BestMoviesNetflix})) \subseteq \pi_{ID}(\text{raw_titles})$
- Structure of each table in a tabular format:

Table: raw_titles

Column	Data Type	Constraints
ID	VARCHAR(255)	Primary Key (PK)
Index	VARCHAR(20)	
Title	VARCHAR(255)	
Type	VARCHAR(255)	
Release_year	INT	
Age_certification	VARCHAR(255)	
Runtime	INT	
Genres	VARCHAR(255)	
Production_countries	VARCHAR(255)	
Seasons	VARCHAR(255)	

Imdb_id	VARCHAR(255)	
Imdb_score	FLOAT	
Imdb_votes	VARCHAR(255)	

Table: raw_credits

Column	Data Type	Constraints
ID	VARCHAR(255)	Foreign Key(FK)
Index	VARCHAR(20)	
Person_id	VARCHAR(255)	
Name	VARCHAR(255)	
Character	VARCHAR(255)	
Actor	VARCHAR(255)	

Table: BestMoviesNetflix

Column	Data Type	Constraints
ID	VARCHAR(255)	Foreign Key (FK)
Index	VARCHAR(20)	
Title	VARCHAR(255)	
Release_year	INT	
Score	FLOAT	
Number_of_votes	INT	
Duration	INT	
Main_genre	VARCHAR(255)	
Main_production	VARCHAR(255)	

Table: BestMoviesByYear

Column	Data Type	Constraints
ID	VARCHAR(255)	Foreign Key (FK)
Index	VARCHAR(20)	
Title	VARCHAR(255)	
Release_year	INT	
Score	FLOAT	
Main_genre	VARCHAR(255)	
Main_production	VARCHAR(255)	

Table: BestShowsNetflix

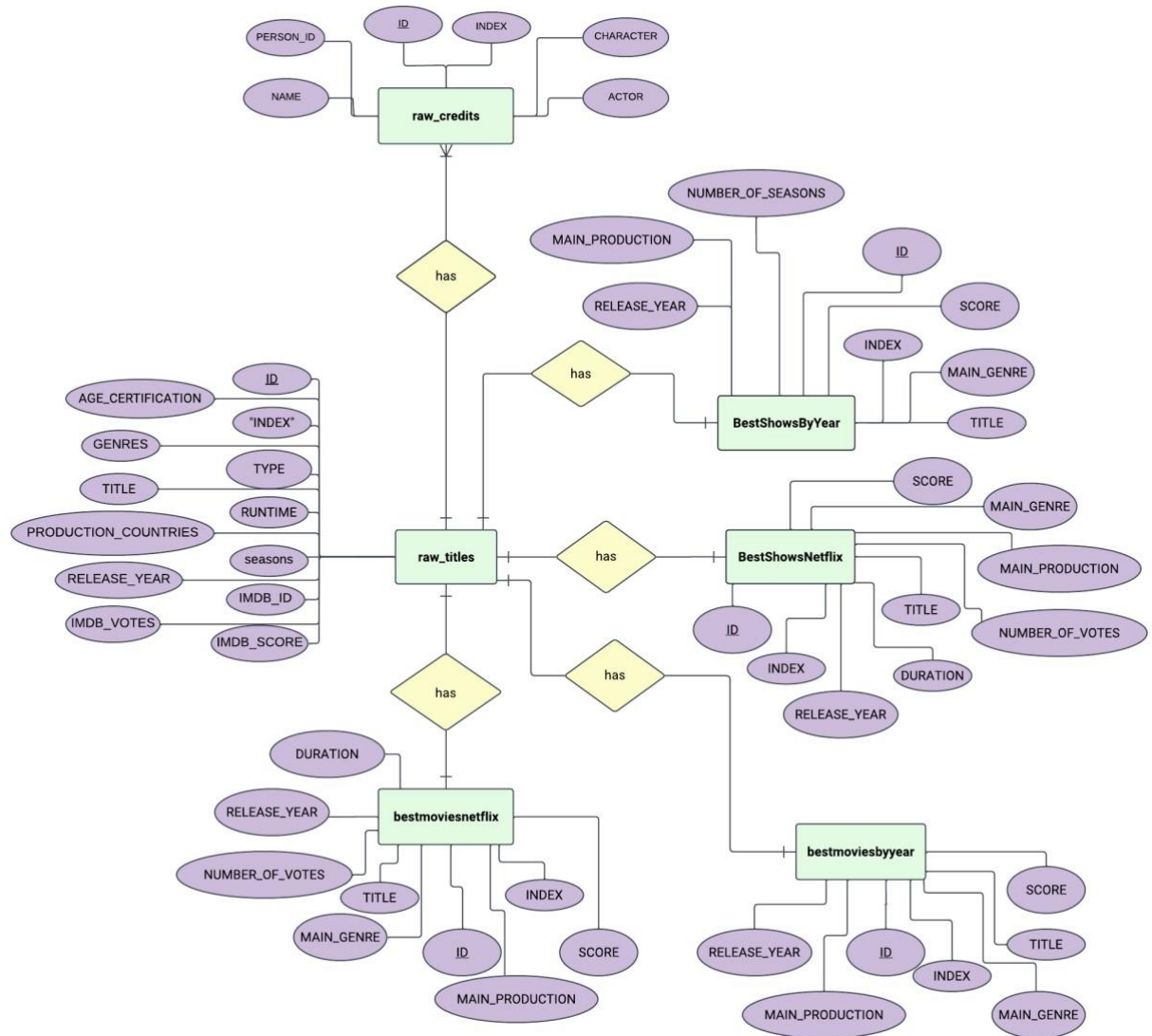
Column	Data Type	Constraints
ID	VARCHAR(255)	Foreign Key (FK)
Index	VARCHAR(20)	
Title	VARCHAR(255)	

Release_year	INT	
Score	FLOAT	
Number_of_votes	INT	
Duration	INT	
Number_of_seasons	INT	
Main_genre	VARCHAR(255)	
Main_production	VARCHAR(255)	

Table: BestShowsByYear

Column	Data Type	Constraints
ID	VARCHAR(255)	Foreign Key (FK)
Index	VARCHAR(20)	
Title	VARCHAR(255)	
Release_year	INT	
Score	FLOAT	
Number_of_seasons	INT	
Main_genre	VARCHAR(255)	
Main_production	VARCHAR(255)	

- **ER DIAGRAM:**



DDL Statements to Implement the Model in PostgreSQL:

```

DROP TABLE IF EXISTS raw_titles;
CREATE TABLE raw_titles (
    ID VARCHAR(100) PRIMARY KEY,
    "Index" VARCHAR(20),
    Title VARCHAR(255),
    Type VARCHAR(255),
    Release_year INT,
    Age_certification VARCHAR(255),
    Runtime INT,
    Genres VARCHAR(255),
    Production_countries VARCHAR(255),
    Seasons VARCHAR(255),
    Imdb_Id VARCHAR(255),
    Imdb_score FLOAT,

```

```
Imdb_votes VARCHAR(255));
```

```
DROP TABLE IF EXISTS raw_credits;  
CREATE TABLE raw_credits (  
    ID VARCHAR(100) NOT NULL,  
    "Index" VARCHAR(20) UNIQUE,  
    Person_Id VARCHAR(255),  
    Name VARCHAR(255) NOT NULL,  
    Character VARCHAR(255),  
    Actor VARCHAR(255) NOT NULL,  
    FOREIGN KEY (ID) REFERENCES raw_titles(ID)  
);
```

```
DROP TABLE IF EXISTS BestShowsByYear;  
CREATE TABLE BestShowsByYear (  
    ID VARCHAR(100),  
    "Index " VARCHAR(20),  
    Title VARCHAR(255) NOT NULL,  
    Release_year INT NOT NULL,  
    Score FLOAT,  
    Number_of_seasons INT,  
    Main_genre VARCHAR(255) NOT NULL,  
    Main_production VARCHAR(255) NOT NULL,  
    FOREIGN KEY (ID) REFERENCES raw_titles(ID),  
    UNIQUE ("Index")  
);
```

```
DROP TABLE IF EXISTS BestMoviesByYear;  
CREATE TABLE BestMoviesByYear (  
    ID VARCHAR(100),  
    "Index" VARCHAR(20) UNIQUE,  
    Title VARCHAR(255) NOT NULL,  
    Release_year INT NOT NULL,  
    Score FLOAT NOT NULL,  
    Main_genre VARCHAR(255) NOT NULL,  
    Main_production VARCHAR(255) NOT NULL,  
    FOREIGN KEY (ID) REFERENCES raw_titles(ID)  
    UNIQUE ("Index")  
);
```

```
DROP TABLE IF EXISTS BestShowsNetflix;  
CREATE TABLE BestShowsNetflix (  
    ID VARCHAR(100),  
    "Index" VARCHAR(20) UNIQUE,  
    Title VARCHAR(255) NOT NULL,
```

```

Release_year INT,
Score FLOAT,
NUMBER_OF_VOTES INT,
Duration INT,
Number_of_seasons INT,
Main_genre VARCHAR(255) NOT NULL,
Main_production VARCHAR(255) NOT NULL,
FOREIGN KEY (ID) REFERENCES raw_titles(ID)
UNIQUE ("Index")
);

```

```

DROP TABLE IF EXISTS BestMoviesNetflix;
CREATE TABLE BestMoviesNetflix (
    ID VARCHAR(100),
    "Index" VARCHAR(20) UNIQUE,
    Title VARCHAR(255),
    Release_year INT,
    Score FLOAT,
    NUMBER_OF_VOTES INT,
    Duration INT,
    Main_genre VARCHAR(255),
    Main_production VARCHAR(255),
    FOREIGN KEY (ID) REFERENCES raw_titles(ID)
    UNIQUE ("Index")
);

```

DML Queries to Insert the Data:

```

UPDATE BestShowsByYear
SET ID = raw_titles.ID
FROM raw_titles WHERE BestShowsByYear.Title =raw_titles.Title ;

```

```

UPDATE BestMoviesByYear
SET ID = raw_titles.ID
FROM raw_titles WHERE BestMoviesByYear.Title =raw_titles.title ; UPDATE BestShowsNetflix
SET ID = raw_titles.ID
FROM raw_titles WHERE BestShowsNetflix.Title =raw_titles.title ;

```

```

UPDATE BestMoviesNetflix
SET ID = raw_titles.ID
FROM raw_titles WHERE BestMoviesNetflix.Title =raw_titles.Title ;

```

SQL Queries to Extract Interesting Facts from the Dataset:

1. Top 10 years with the highest number of movies released.


```

WITH moviecount AS (
SELECT Release_year, COUNT(*) AS NumberofMovies
FROM raw_titles
WHERE type = 'MOVIE'
GROUP BY Release_year
)
SELECT Release_year, NumberofMovies
FROM moviecount
ORDER BY NumberofMovies DESC
LIMIT 10;

```

Result of the query showcasing: Top 10 years with the highest number of movies released.

	release_year integer	numberofmovies bigint
1	2019	540
2	2020	499
3	2018	473
4	2021	455
5	2017	397
6	2016	229
7	2015	143
8	2014	114
9	2022	108
10	2013	105

2. Average IMDB Score for Movies and Shows.

```

SELECT 'Movie' AS Type, AVG(Imdb_score) AS Averagescore
FROM raw_titles WHERE
Type = 'MOVIE'
UNION ALL
SELECT 'Show' AS Type, AVG(Imdb_score) AS Averagescore
FROM raw_titles
WHERE Type = 'SHOW';

```

Result of the query showcasing: Average IMDB Score for Movies and Shows

	type text	averagescore double precision
1	Movie	6.266979747578528
2	Show	7.0173773987206784

3. List of Best Movies by Year in Each Genre Category.

```
SELECT Main_genre, COUNT (*) AS numberofmovies
FROM BestMoviesByYear
GROUP BY Main_genre
ORDER BY numberofmovies DESC;
```



Result of the query showcasing: List of Best Movies by Year in Each Genre Category

	main_genre character varying (255)	numberofmovies bigint
1	drama	22
2	comedy	8
3	thriller	5
4	romance	3
5	scifi	2
6	horror	2
7	crime	2
8	war	1
9	documentary	1
10	western	1
11	fantasy	1
12	action	1

4. Top 10 movies with highest number of actors in it.

```
SELECT rt.Title AS movietitle, COUNT(DISTINCT rc.Name) AS numberofactors
FROM raw_titles rt
JOIN raw_credits rc ON rt.ID = rc.ID
where Type='MOVIE' AND Role='ACTOR'
GROUP BY rt.Title
ORDER BY numberofactors DESC
LIMIT 10;
```

Result of the query showcasing: Top 10 movies with highest number of actors in it.

	movietitle character varying (255) 	numberofactors bigint 
1	Les Misérables	207
2	The Irishman	173
3	Hairspray	149
4	Homecoming: A Film by Beyoncé	137
5	Contagion	136
6	Memoirs of a Geisha	131
7	tick, tick... BOOM!	126
8	Extraction	117
9	Argo	115
10	Django Unchained	112

5. Popular Genres from Netflix’s Best Movie Category by number of votes.

```
SELECT main_genre, SUM(number_of_votes) AS totalvotes
FROM BestMoviesNetflix
GROUP BY main_genre
ORDER BY totalvotes DESC;
```

Result of the query showcasing: Popular Genres from Netflix’s Best Movie Category by number of votes.

	main_genre character varying (255) 🔒	totalvotes bigint 🔒
1	drama	21804564
2	thriller	8398468
3	comedy	4703792
4	crime	3857710
5	scifi	3832671
6	western	2770109
7	fantasy	2639958
8	horror	1759154
9	romance	1428590
10	action	535189
11	documentary	504297
12	animation	341513
13	war	141682
14	musical	94236
15	sports	21558

6. Top 5 Actors with highest appearances in Movies and Shows.

```
(SELECT Name AS PersonName, COUNT(*) AS moviecount, 'Movie' AS Type
FROM raw_credits
WHERE Role = 'ACTOR' AND ID IN (SELECT ID FROM raw_titles WHERE Type =
'MOVIE')
GROUP BY Name
ORDER BY moviecount DESC
LIMIT 5)
UNION ALL
(SELECT Name AS PersonName, COUNT(*) AS showcount, 'Show' AS Type
FROM raw_credits
WHERE Role = 'ACTOR' AND ID IN (SELECT ID FROM raw_titles WHERE Type =
'SHOW')
GROUP BY Name
ORDER BY showcount DESC
LIMIT 5);
```

Result of this query showcasing: Top 5 Actors with highest appearances in Movies and Shows.

	personname character varying (100) 🔒	moviecount bigint 🔒	type text 🔒
1	Shah Rukh Khan	30	Movie
2	Kareena Kapoor Khan	25	Movie
3	Boman Irani	25	Movie
4	Anupam Kher	24	Movie
5	Paresh Rawal	22	Movie
6	Takahiro Sakurai	15	Show
7	Junichi Suwabe	15	Show
8	Ai Kayano	11	Show
9	Yoshimasa Hosoya	11	Show
10	Natsuki Hanae	10	Show

7. Top 10 directors whose movies made it to 'Best Movies Netflix' category:

```
SELECT COUNT(*) AS numberofmovies, rc.Name AS directorname
FROM raw_credits AS rc
JOIN BestMoviesNetflix bm ON rc.ID = bm.ID AND Role='DIRECTOR'
GROUP BY rc.Name
ORDER BY numberofmovies DESC
LIMIT 10;
```

Result of the query showcasing: Top 10 directors whose movies made it to 'Best Movies Netflix' category.

	numberofmovies bigint	directorname character varying (100)
1	4	Wilson Yip
2	4	Farhan Akhtar
3	3	Rajkumar Hirani
4	3	Bo Burnham
5	3	Mani Ratnam
6	3	Abhishek Kapoor
7	3	Anurag Kashyap
8	3	Guy Ritchie
9	3	Ashutosh Gowariker
10	3	Imtiaz Ali

8. Top 10 Directors who got highest IMDB Ratings overall for their Movies:

```

SELECT rc.Name AS directorname, MAX(rt.imdb_score) AS maximumscore
FROM raw_titles rt
JOIN raw_credits rc ON rc.ID = rt.ID
WHERE rc.Role = 'DIRECTOR' AND rt.Type = 'MOVIE' AND rt.Imdb_score IS NOT
NULL
GROUP BY rc.name
ORDER BY maximumscore DESC
LIMIT 10;

```

Result of the query showcasing: Top 10 Directors who got highest IMDB Scores overall for their Movies:

	directorname character varying (100) 🔒	maximumscore double precision 🔒
1	Alastair Fothergill	9
2	Jonathan Hughes	9
3	Samir Al Asfory	9
4	Venkatesh Maha	9
5	Keith Scholey	9
6	Robert Zemeckis	8.8
7	Sơn Tùng M-TP	8.8
8	Christopher Nolan	8.8
9	Bo Burnham	8.7
10	Julia Reichert	8.7

9. Top 10 shows with highest number of seasons

```
SELECT Title, Seasons
FROM raw_titles
WHERE Type = 'SHOW'
ORDER BY Seasons DESC
LIMIT 10;
```

Result of the query showcasing: Top 10 shows with highest number of seasons

	title character varying (255) 🔒	seasons double precision 🔒
1	Survivor	42
2	Wheel of Fortune	39
3	The Challenge	37
4	Power Rangers	29
5	Thomas & Friends	24
6	Pokémon	24
7	America's Next Top Model	24
8	One Piece	21
9	NCIS	19
10	Grey's Anatomy	18

10. Average Runtime for Movies and Shows:

```
(SELECT AVG(runtime) AS avgruntimeinmins, 'Movie' AS Type
FROM raw_titles
WHERE Type = 'MOVIE')
union all
(SELECT AVG(runtime) AS avgruntimeinmins, 'Show' AS Type
FROM raw_titles
```

WHERE Type = 'SHOW');

Result of the query showcasing: Average Runtime for Movies and Shows.

	avgruntimeinmins numeric	type text
1	98.7853152434158021	Movie
2	38.8212017586712262	Show

11. Top 10 Production Countries with highest count of Genres in their Best Movie category:

```
SELECT Main_production as ProductionCountry, COUNT(DISTINCT Main_genre) AS  
numofgenres  
FROM BestMoviesNetflix  
GROUP BY Main_production  
ORDER BY numofgenres DESC  
LIMIT 10;
```

Result of the query showcasing: Top 10 Production Countries with highest count of Genres in their Best Movie category

	productioncountry character varying (255)	numofgenres bigint
1	US	14
2	GB	8
3	IN	8
4	JP	7
5	ES	5
6	DE	5
7	FR	4
8	TR	3
9	CA	3
10	AU	3

12. Top 10 movies with highest Imdb scores which are under 60 minutes duration.

```
SELECT Title, Imdb_score  
FROM raw_titles  
WHERE Runtime<=60 AND Type='MOVIE' AND Imdb_score IS NOT NULL
```



```
ORDER BY Imdb_Score DESC
LIMIT 10;
```

Result of the query showcasing: Top 10 movies with highest Imdb scores which are under 60 minutes duration.

	title character varying (255)	imdb_score double precision
1	Best Wishes, Warmest Regards: A Schitt's Creek Farew...	8.6
2	Bo Burnham: Make Happy	8.4
3	Frontiers of Dreams and Fears	8.3
4	Bo Burnham: What.	8.3
5	John Mulaney: New in Town	8.2
6	Children of Shatila	8.2
7	The Trailer Park Boys Xmas Special	8
8	Sinbad: Afros and Bellbottoms	7.8
9	Monty Python: Live at Aspen	7.8
10	Trailer Park Boys: Say Goodnight to the Bad Guys	7.8

13. Top 10 release years with highest cumulative IMDB scores for movies.

```
SELECT Release_year,ROUND(SUM(Imdb_score)::numeric, 2) AS cumulativescore FROM
raw_titles
WHERE Type = 'MOVIE'
GROUP BY Release_year
ORDER BY cumulativescore DESC
LIMIT 10;
```

Result of the query showcasing: TOP 10 release years with highest cumulative IMDB scores for movies.

	release_year integer	cumulativescore numeric
1	2019	2925.40
2	2018	2777.20
3	2021	2479.00
4	2020	2467.40
5	2017	2357.30
6	2016	1328.60
7	2015	837.80
8	2014	711.30
9	2013	625.50
10	2022	537.20

14. Shows awarded as best in the documentary genre on Netflix but not awarded as the best show by year.

```
SELECT Title, Score FROM BestShowsNetflix WHERE Main_genre ='documentary'
EXCEPT
SELECT Title, Score FROM BestShowsByYear WHERE Main_genre = 'documentary';
```

Result of the query showcasing: Shows awarded as best in the documentary genre on Netflix but not awarded as the best show by year.

	title character varying (255)	score double precision
1	Our Planet	9.3
2	Formula 1: Drive to Survive	8.6
3	Manhunt	8.1
4	Rise of Empires: Ottoman	7.9
5	Inside Bill's Brain: Decoding Bill Gates	7.9
6	The Devil Next Door	7.6

15. Top-ranked movies from the best movies category in Netflix based on IMDB scores, within each genre.

```
SELECT Title, Main_genre, Score
FROM (
  SELECT Title, Main_genre, Score,
  ROW_NUMBER () OVER (PARTITION BY Main_genre ORDER BY Score DESC ) AS
Rank
  FROM BestMoviesNetflix
) AS rankedmovies
```

WHERE Rank=1;

Result of the query showcasing: Top-ranked movies from the best Movies in Netflix based on IMDb scores, within each genre.

	title character varying (255) 🔒	main_genre character varying (255) 🔒	score double precision 🔒
1	Dangal	action	8.4
2	The Mitchells vs. the Machines	animation	7.6
3	Bo Burnham: Inside	comedy	8.7
4	Black Friday	crime	8.4
5	David Attenborough: A Life on Our Planet	documentary	9
6	Forrest Gump	drama	8.8
7	Bāhubali 2: The Conclusion	fantasy	8.2
8	The Exorcist	horror	8.1
9	What Happened, Miss Simone?	musical	7.6
10	Bombay	romance	8.1
11	Inception	scifi	8.8
12	Schumacher	sports	7.4
13	Super Deluxe	thriller	8.4
14	Virunga	war	8.2
15	Django Unchained	western	8.4

16. Movies released between 2021 and 2022 along with the name of the director under Best Movie Category in Netflix.

```
SELECT bm.Title,rc.Name
FROM raw_credits AS rc
JOIN BestMoviesByYear AS bm ON rc.ID = bm.ID
WHERE Release_year <= 2022 AND Release_year >=2021 AND rc.Role = 'DIRECTOR';
```


Result of the query showcasing: Movies released between 2021 and 2022 along with the name of the director under Best Movie Category in Netflix.

	title character varying (255) 🔒	name character varying (100) 🔒
1	The Tinder Swindler	Felicity Morris
2	Bo Burnham: Inside	Bo Burnham

17. Best Comedy Shows with an episode runtime of 25 minutes or less on Netflix.

```
SELECT Title
FROM BestShowsByYear
WHERE Main_genre = 'comedy'
INTERSECT
SELECT Title
FROM raw_titles
WHERE Runtime <= 25 AND Type = 'SHOW';
```

Result of the query showcasing: Best Comedy Shows with an episode runtime of 25 minutes or less on Netflix.

	title character varying (255) 
1	Trailer Park Boys
2	Chappelle's Show
3	Seinfeld
4	Community

18. Top 10 directors whose movies has the longest runtime in minutes.

```
SELECT rc.Name , rt.Runtime
FROM raw_credits AS rc
JOIN raw_titles AS rt ON rc.ID = rt.ID
WHERE rt.Type = 'MOVIE' AND rc.Role='DIRECTOR'
ORDER BY rt.Runtime DESC
LIMIT 10;
```

Result of the query showcasing: Top 10 directors whose movies has the longest runtime in minutes.

	name character varying (100) 🔒	runtime integer 🔒
1	جلال الشرقاوي	251
2	عبدالله الشيخ	251
3	Hossam El Din Mostafa	251
4	Bruce Beresford	240
5	Samir Al Asfory	235
6	Fouad El-Mohandes	230
7	Sergio Leone	229
8	Julia Reichert	225
9	Steven Bognar	225
10	Ashutosh Gowariker	224

19. List of actors whose movies were released before 1980.

```
SELECT DISTINCT rc.Name, rt.Release_year
FROM raw_credits AS rc
JOIN raw_titles AS rt ON rc.ID = rt.ID
WHERE rt.type = 'SHOW'
AND Release_year < 1980
AND rc.Role = 'ACTOR'
ORDER BY Release_year;
```

Result of the query showcasing: List of actors whose movies were released before 1980.

	name character varying (100) 🔒	release_year integer 🔒
1	Eric Idle	1969
2	Graham Chapman	1969
3	Michael Palin	1969
4	Terry Gilliam	1969
5	Terry Jones	1969
6	Eric Idle	1972
7	Graham Chapman	1972
8	John Cleese	1972
9	Michael Palin	1972
10	Terry Jones	1972

20. Production Countries with Highest number of movies and shows produced.

```
(SELECT Production_Countries, COUNT(*) AS NumofMovies, 'MOVIE' AS Type
FROM raw_titles
WHERE Type = 'MOVIE' AND Production_Countries IS NOT NULL
GROUP BY Production_Countries
ORDER BY NumofMovies DESC
LIMIT 2)
UNION ALL
(SELECT Production_Countries, COUNT(*) AS NumofMovies, 'SHOW' AS Type
FROM raw_titles
WHERE Type = 'SHOW' AND Production_Countries IS NOT NULL
GROUP BY Production_Countries
ORDER BY NumofMovies DESC
LIMIT 2);
```

Result of the query showcasing: Production countries with Highest number of movies and shows produced.

	production_countries character varying (50) 🔒	numofmovies bigint 🔒	type text 🔒
1	['US']	1206	MOVIE
2	['IN']	562	MOVIE
3	['US']	744	SHOW
4	['JP']	166	SHOW

Indexing:

Created an Index to a column of a table with highest number of rows, to observe an improvement in performance, by using 'EXPLAIN ANALYSE' command.

Query:

```
EXPLAIN ANALYSE
SELECT Name, Character
FROM raw_credits
WHERE name = 'shahrukhkhan';
```

Execution plan and actual run-time statistics for above SQL statement:

```
"Seq Scan on raw_credits (cost=0.00..1821.16 rows=2 width=27) (actual time=70.904..70.905
rows=0 loops=1)"
"Filter: ((name)::text = 'shahrukhkhan')::text)"
"Rows Removed by Filter: 77213"
```

"Planning Time: 0.123 ms"

"Execution Time: 70.939 ms"

Observations: PostgreSQL has performed a sequential scan of all rows on 'raw_credits' table, it also estimated the cost of executing the query, provided the actual time taken to execute this query(in milliseconds), applied the condition to the query and returned the result.

Time taken to generate query plan is 0.123 ms.

Time taken to execute the query is 70.939 ms.

DDL Statement to create an Index:

```
CREATE INDEX indxonname ON raw_credits(Name);
```

Execution plan and actual run-time statistics after creating Index:

```
"Bitmap Heap Scan on raw_credits (cost=4.43..12.17 rows=2 width=27) (actual  
time=0.022..0.022 rows=0 loops=1)"
```

```
" Recheck Cond: ((name)::text = 'shahrukhkhan'::text)"
```

```
" -> Bitmap Index Scan on indxonname (cost=0.00..4.43 rows=2 width=0) (actual  
time=0.020..0.020 rows=0 loops=1)"
```

```
" Index Cond: ((name)::text = 'shahrukhkhan'::text)"
```

"Planning Time: 0.134 ms"

"Execution Time: 0.050 ms"

Observations:

PostgreSQL has performed a bitmap heap scan on 'raw_credits' table, to access the data, it estimated the cost of executing the query, provided the actual time taken to execute this query(in milliseconds), applied the condition to the query using bitmap index scan and returned the result.

Time taken to generate query plan is 0.134 ms.

Time taken to execute the query is 0.050 ms.

We can observe that the execution time has been reduced significantly, from 70.939 ms to 0.050 ms. Though our data does not contain large number of rows, explaining why the execution time is in milliseconds, and the difference after using indexing is not that evident. We can conclude that indexing can result in faster execution times for large amount of data, when frequent data extraction is required.