Michael Kupferstein (Koop)

IMDB Relational

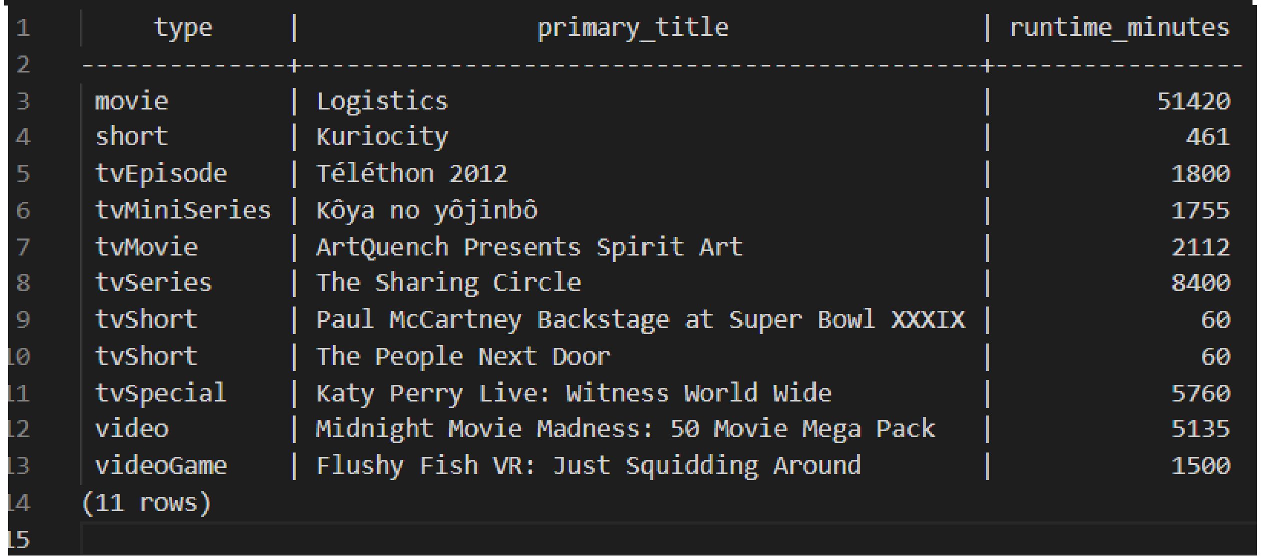
Table of Contents

(With quick links)

Query 1 - Longest duration for each movie type

Query: In one query, for each movie type, determine the "longest duration" over all instances of that type. "Duration" is determined by"runtime minutes".The result set should contain these attributes: type, primary_title, runtime_minutes.Presentation: the result set should be ordered by ascending "type". Within a type, if multiple titles qualify as "long duration", include all of them in the result set and order them by ascending primary titles.

```sql
WITH MaxRuntimes AS (
    SELECT type, MAX(runtime_minutes) AS max_runtime
    FROM titles
    WHERE runtime_minutes IS NOT NULL
    GROUP BY type
)
SELECT t.type, t.primary_title, t.runtime_minutes
FROM titles t
JOIN MaxRuntimes m ON t.type = m.type AND t.runtime_minutes = m.max_runtime
ORDER BY t.type ASC, t.primary_title ASC;
```

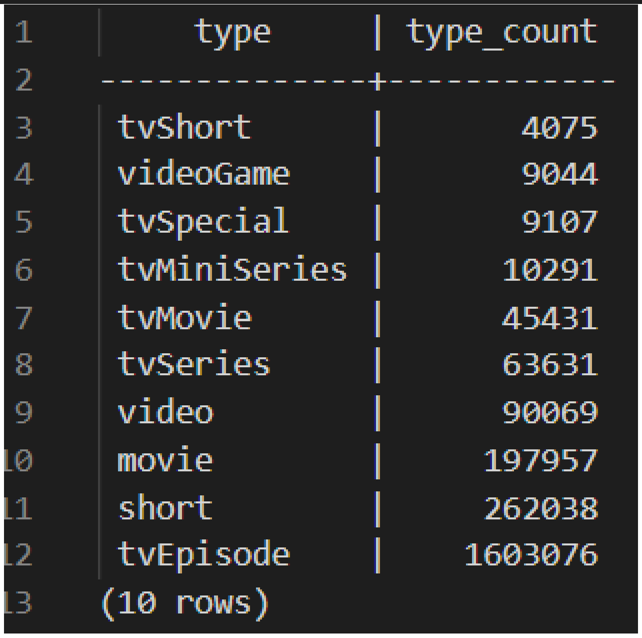


```
     type     |                  primary_title                   | runtime_minutes
--------------+---------------------------------------------+-----------------
 movie        | Logistics                                    |           51420
 short        | Kuriocity                                    |             461
 tvEpisode    | Téléthon 2012                                |            1800
 tvMiniSeries | Kôya no yôjinbô                              |            1755
 tvMovie      | ArtQuench Presents Spirit Art                |            2112
 tvSeries     | The Sharing Circle                           |            8400
 tvShort      | Paul McCartney Backstage at Super Bowl XXXIX |              60
 tvShort      | The People Next Door                         |              60
 tvSpecial    | Katy Perry Live: Witness World Wide          |            5760
 video        | Midnight Movie Madness: 50 Movie Mega Pack   |            5135
 videoGame    | Flushy Fish VR: Just Squidding Around        |            1500
(11 rows)
```

Query 2 - Count of each movie type

Query: In one query, for each movie type, retrieve the number of that type. The results set should contain these attributes: type, title_type. Presentation: order by ascending "number of titles".

```
SELECT type, COUNT(*) AS type_count
FROM titles
GROUP BY type
ORDER BY type_count ASC;
```

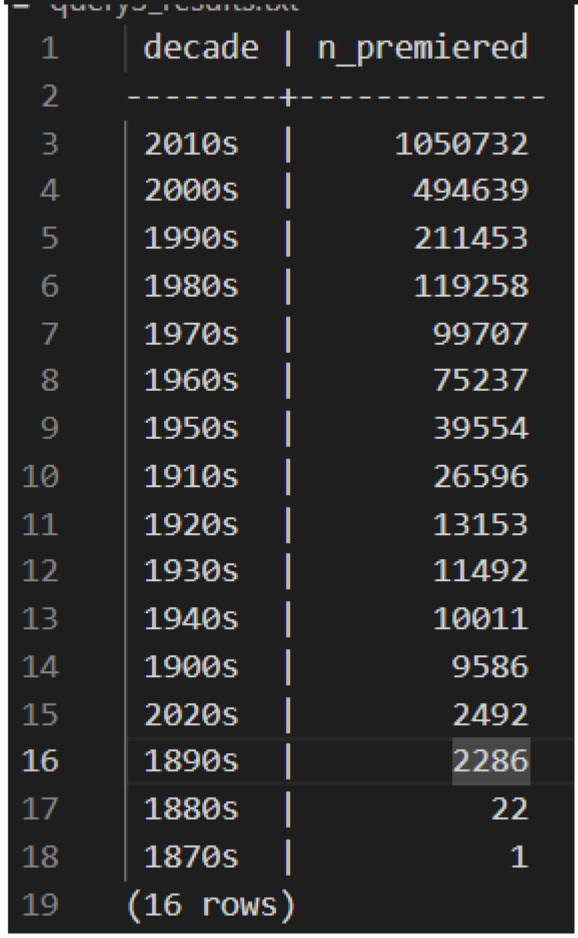


```
     type      | type_count
---------------+------------
 tvShort       |       4075
 videoGame     |       9044
 tvSpecial     |       9107
 tvMiniSeries  |      10291
 tvMovie       |      45431
 tvSeries      |      63631
 video         |      90069
 movie         |     197957
 short         |     262038
 tvEpisode     |    1603076
(10 rows)
```

Query 3 - Titles premiered by decade

Query: For each decade for which data exist in the database, return the number of titles (over all types) that “premiered” in that decade. The result set should contain these attributes: decade, n_premiered. Careful with null! Presentation: values for the “decade” attribute should be in this format: 1960s (note the “s”). The decades should be descending order of “number premiered”.

```sql
SELECT (FLOOR(premiered/10)*10) || 's' AS decade, COUNT(*) AS n_premiered
FROM titles
WHERE premiered IS NOT NULL
GROUP BY decade
ORDER BY n_premiered DESC;
```

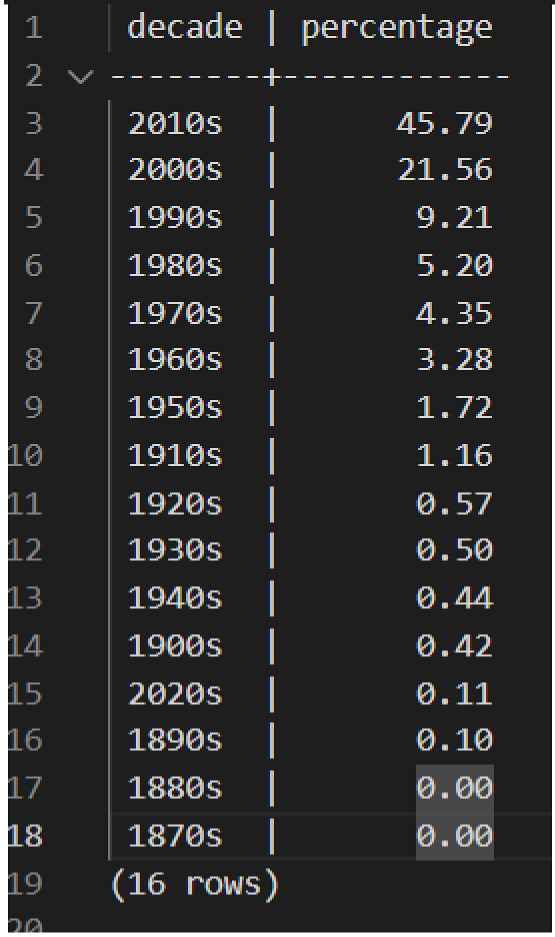


```
 decade | n_premiered
--------+-------------
 2010s  |     1050732
 2000s  |      494639
 1990s  |      211453
 1980s  |      119258
 1970s  |       99707
 1960s  |       75237
 1950s  |       39554
 1910s  |       26596
 1920s  |       13153
 1930s  |       11492
 1940s  |       10011
 1900s  |        9586
 2020s  |        2492
 1890s  |        2286
 1880s  |          22
 1870s  |           1
(16 rows)
```

Query 4 - Percentage of titles premiered by decade

Query: For each decade for which data exist in the database, return the percentage of titles (over all types) that “premiered” in that decade. Define “percentage” as the number of titles divided by the total number of titles. In this query, for the total number of titles, count all titles including ones that have not been premiered. Note: this query is similar to the previous one, so be careful about the differences! The result set should contain these attributes: decade, percentage. Careful with null! Presentation: values for the “decade” attribute should be in this format: 1960s (note the “s”). The decades should be descending order of “number premiered”. Round the percentage to two decimal places using ROUND().

```sql
WITH total AS (SELECT COUNT(*) AS total_count FROM titles)
SELECT (FLOOR(premiered/10)*10) || 's' AS decade,
       ROUND((COUNT(*)*100.0/total.total_count), 2) AS percentage
FROM titles, total
WHERE premiered IS NOT NULL
GROUP BY decade, total.total_count
ORDER BY COUNT(*) DESC;
```

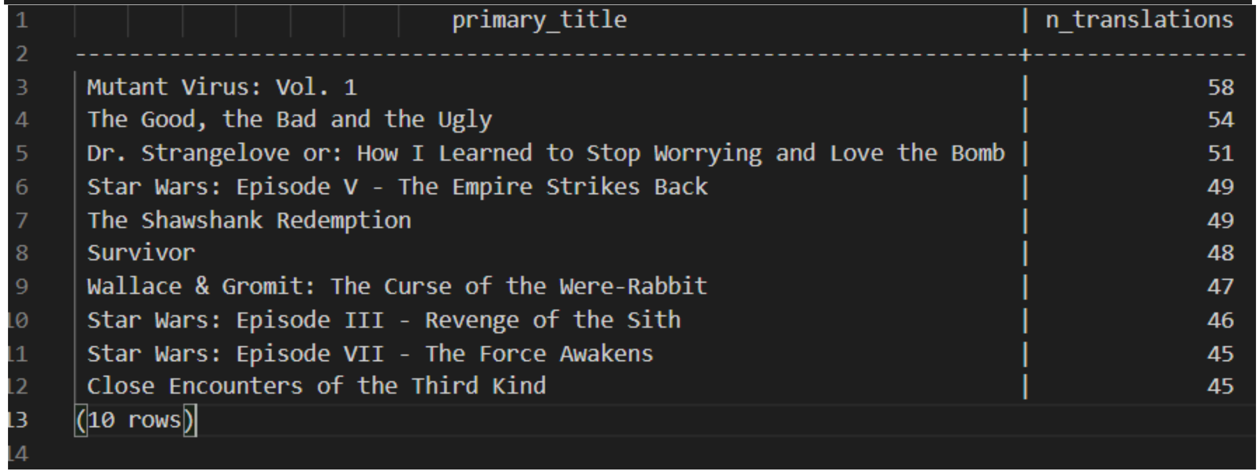


```
 decade | percentage
--------+------------
 2010s  |      45.79
 2000s  |      21.56
 1990s  |       9.21
 1980s  |       5.20
 1970s  |       4.35
 1960s  |       3.28
 1950s  |       1.72
 1910s  |       1.16
 1920s  |       0.57
 1930s  |       0.50
 1940s  |       0.44
 1900s  |       0.42
 2020s  |       0.11
 1890s  |       0.10
 1880s  |       0.00
 1870s  |       0.00
(16 rows)
```

Query 5 - Number or translations per title

Query: For each title in the database, return the number of “translations” that were made of that title. The result set should contain these attributes: primary_title, n_translations. Presentation: the result set should contain only the top ten tuples, ordered by descending “number of translations”.

```sql
SELECT t.primary_title, COUNT(DISTINCT a.title) AS n_translations
FROM titles t
JOIN akas a ON t.title_id = a.title_id
GROUP BY t.title_id, t.primary_title
ORDER BY n_translations DESC
LIMIT 10;
```

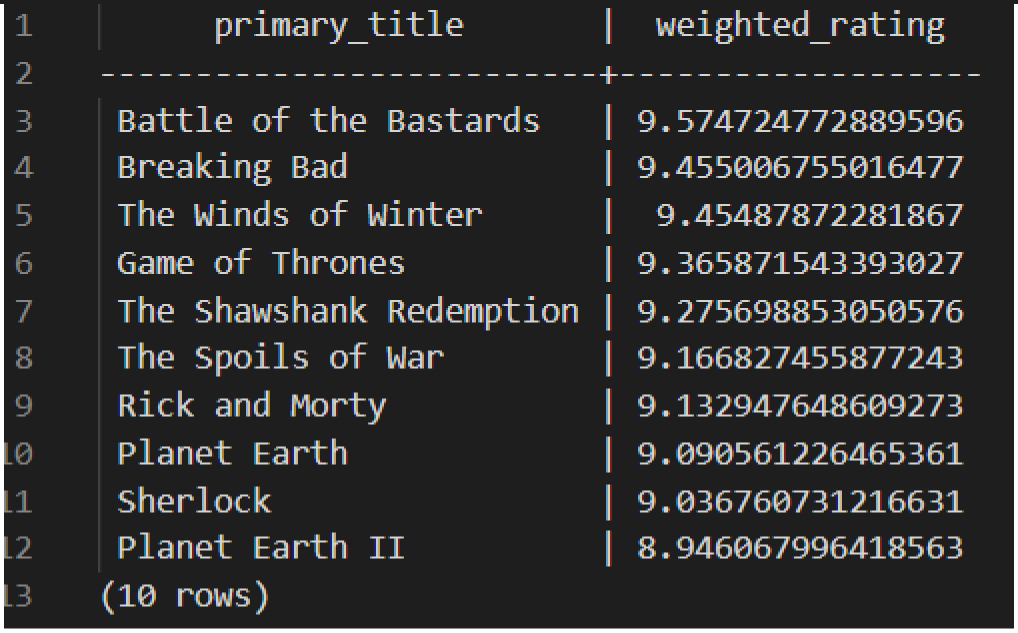


```
                         primary_title                            | n_translations
------------------------------------------------------------------+----------------
 Mutant Virus: Vol. 1                                             |             58
 The Good, the Bad and the Ugly                                   |             54
 Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb |         51
 Star Wars: Episode V - The Empire Strikes Back                   |             49
 The Shawshank Redemption                                         |             49
 Survivor                                                         |             48
 Wallace & Gromit: The Curse of the Were-Rabbit                   |             47
 Star Wars: Episode III - Revenge of the Sith                     |             46
 Star Wars: Episode VII - The Force Awakens                       |             45
 Close Encounters of the Third Kind                               |             45
(10 rows)
```

Query 6 - Weighted ratings for titles

Query: For each title in the database, return the “weighted rating” of that title (details below). The result set should contain these attributes: primary_title, weighted_rating. Presentation: the result set should contain only the top ten tuples ordered by descending “weighted ratings”. Do not round the value of the weighted rating.

```sql
WITH c AS (
    SELECT SUM(rating * votes) / SUM(votes) AS avg_rating
    FROM ratings
)
SELECT t.primary_title,
       ((r.votes::float / (r.votes + 25000)) * r.rating +
        (25000::float / (r.votes + 25000)) * c.avg_rating) AS weighted_rating
FROM titles t
JOIN ratings r ON t.title_id = r.title_id
CROSS JOIN c
ORDER BY weighted_rating DESC
LIMIT 10;
```

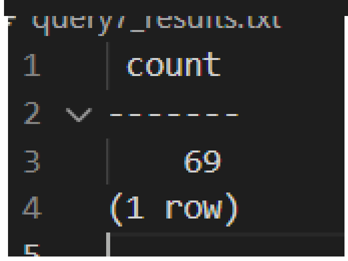


```
       primary_title         |  weighted_rating
-----------------------------+--------------------
 Battle of the Bastards      | 9.574724772889596
 Breaking Bad                | 9.455006755016477
 The Winds of Winter         |  9.45487872281867
 Game of Thrones             | 9.365871543393027
 The Shawshank Redemption    | 9.275698853050576
 The Spoils of War           | 9.166827455877243
 Rick and Morty              | 9.132947648609273
 Planet Earth                | 9.090561226465361
 Sherlock                    | 9.036760731216631
 Planet Earth II             | 8.946067996418563
(10 rows)
```

Query 7 - Number of actor/actresses that were in a movie with Ian McKellen

Query: Return the number of actors or actresses who appeared in any title in the database with Ian McKellen.

```sql
WITH ian_movies AS (
    SELECT DISTINCT c.title_id
    FROM people p
    JOIN crew c ON p.person_id = c.person_id
    WHERE p.name = 'Ian McKellen'
    AND c.category IN ('actor', 'actress')
)
SELECT COUNT(DISTINCT p.person_id) AS count
FROM crew c
JOIN people p ON c.person_id = p.person_id
JOIN ian_movies im ON c.title_id = im.title_id
WHERE c.category IN ('actor', 'actress');
```

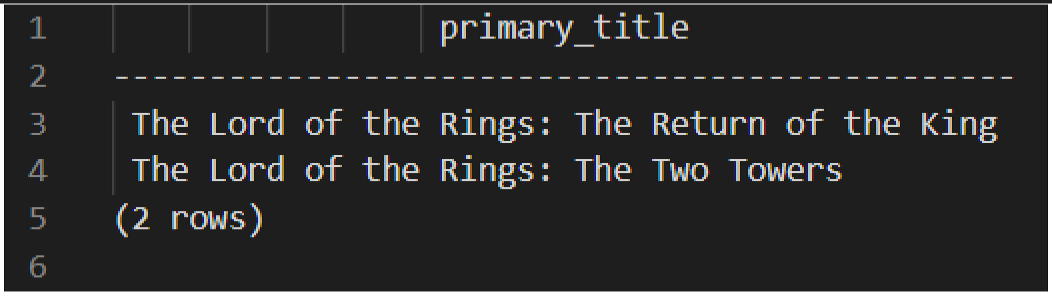


```
 count
-------
    69
(1 row)
```

Query 8 - Movies with Orlando Bloom and Ian McKellen

Query: Return the movies whose cast includes both “Ian McKellen” and “Orlando Bloom”. The result set should include primary_title. Presentation: order the tuples by ascending “primary title”.

```sql
WITH ian_movies AS (
    SELECT DISTINCT c.title_id
    FROM people p
    JOIN crew c ON p.person_id = c.person_id
    WHERE p.name = 'Ian McKellen'
    AND c.category IN ('actor', 'actress')
),
orlando_movies AS (
    SELECT DISTINCT c.title_id
    FROM people p
    JOIN crew c ON p.person_id = c.person_id
    WHERE p.name = 'Orlando Bloom'
    AND c.category IN ('actor', 'actress')
)
SELECT DISTINCT t.primary_title
FROM titles t
JOIN ian_movies i ON t.title_id = i.title_id
JOIN orlando_movies o ON t.title_id = o.title_id
ORDER BY t.primary_title ASC;
```

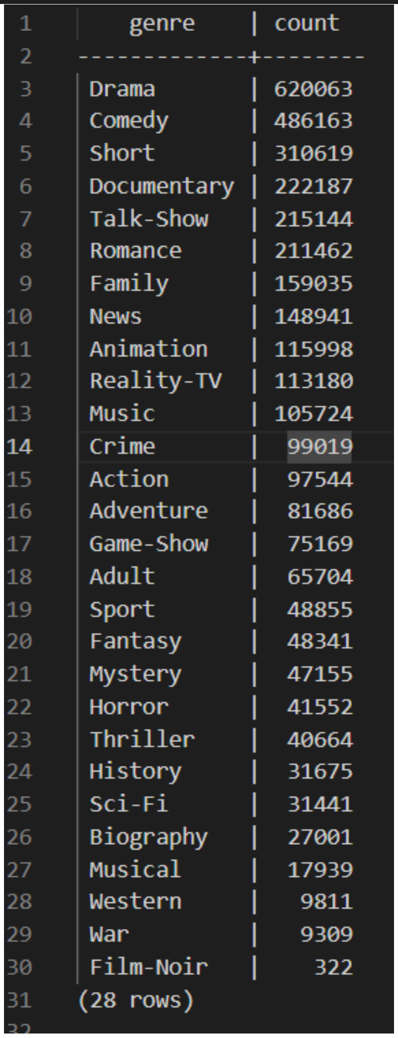


```
                    primary_title
-----------------------------------------------
 The Lord of the Rings: The Return of the King
 The Lord of the Rings: The Two Towers
(2 rows)
```

Query 9 - Count of each genre

Query: Referring to the “genres” attribute associated with titles, return all distinct genres and the number of titles associated with them. The result set contains genre, count. Presentation: order the tuples by descending “count” values.

```sql
WITH split_genres AS (
    SELECT unnest(string_to_array(genres, ',')) AS genre
    FROM titles
    WHERE genres IS NOT NULL AND genres != '\N'
)
SELECT genre, COUNT(*) AS count
FROM split_genres
GROUP BY genre
ORDER BY count DESC;
```



```
    genre     | count
--------------+--------
 Drama        | 620063
 Comedy       | 486163
 Short        | 310619
 Documentary  | 222187
 Talk-Show    | 215144
 Romance      | 211462
 Family       | 159035
 News         | 148941
 Animation    | 115998
 Reality-TV   | 113180
 Music        | 105724
 Crime        |  99019
 Action       |  97544
 Adventure    |  81686
 Game-Show    |  75169
 Adult        |  65704
 Sport        |  48855
 Fantasy      |  48341
 Mystery      |  47155
 Horror       |  41552
 Thriller     |  40664
 History      |  31675
 Sci-Fi       |  31441
 Biography    |  27001
 Musical      |  17939
 Western      |   9811
 War          |   9309
 Film-Noir    |    322
(28 rows)
```