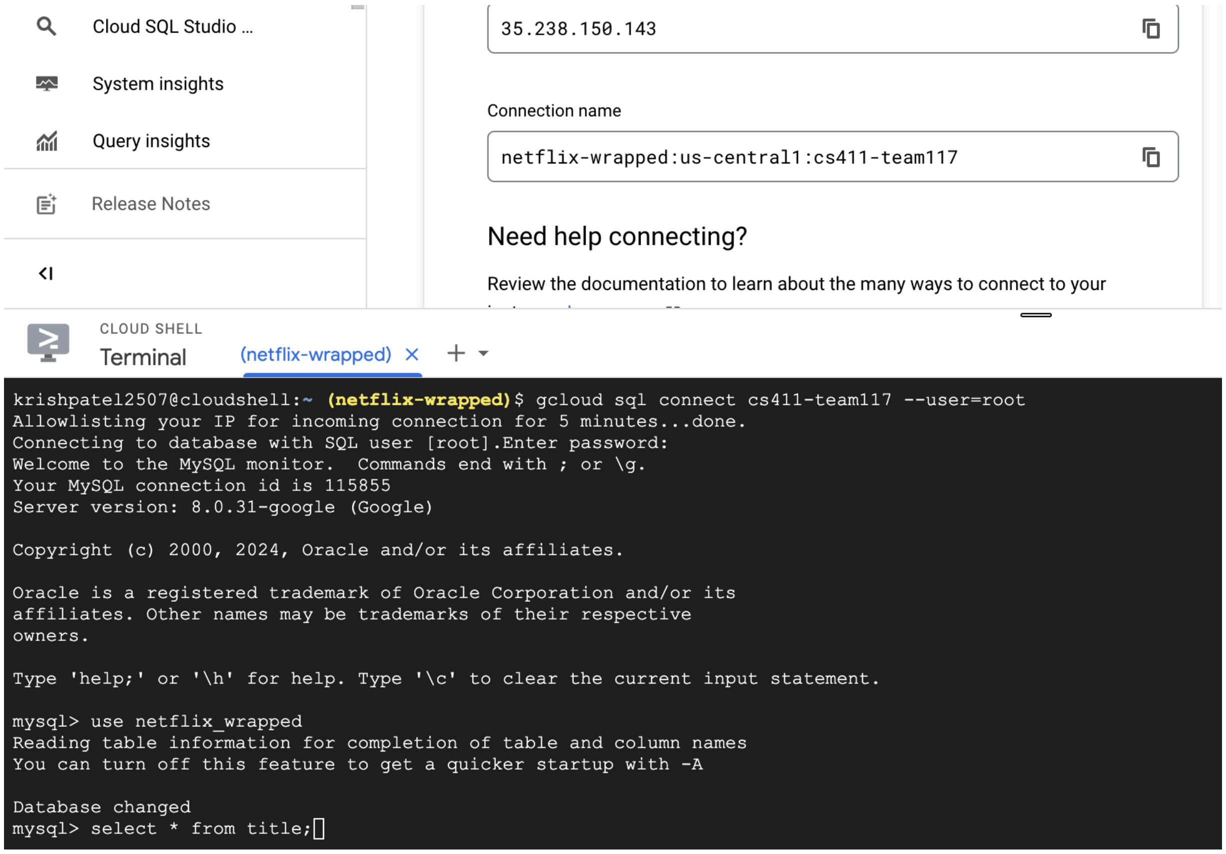
TEAM 117 - NETFLIX WRAPPED
Stage 3:
Database Implementation and Indexing



35.238.150.143

netflix-wrapped:us-central1:cs411-team117

Review the documentation to learn about the many ways to connect to your

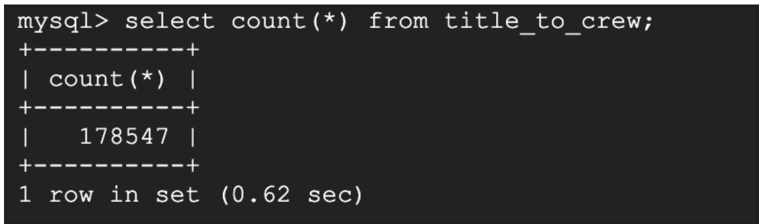
```
krishpatel2507@cloudshell:~ (netflix-wrapped)$ gcloud sql connect cs411-team117 --user=root
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 115855
Server version: 8.0.31-google (Google)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use netflix_wrapped
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from title;
```

Figure 1: Google cloud platform connection

Part 1: Data Definition Language (DDL) commands

Title_to_crew

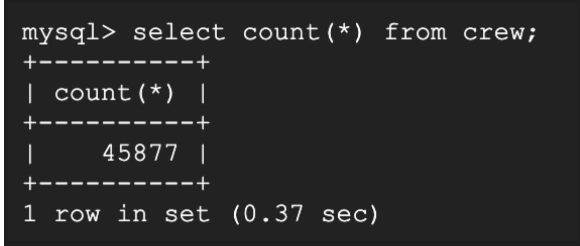
```
CREATE TABLE title_to_crew (
    titleconst VARCHAR(10),
    directors VARCHAR(20),
    PRIMARY KEY (titleconst, directors),
    FOREIGN KEY (titleconst) REFERENCES title (tconst),
    FOREIGN KEY (directors) REFERENCES crew (nconst)
);
```



```
mysql> select count(*) from title_to_crew;
+----------+
| count(*) |
+----------+
|   178547 |
+----------+
1 row in set (0.62 sec)
```

Crew

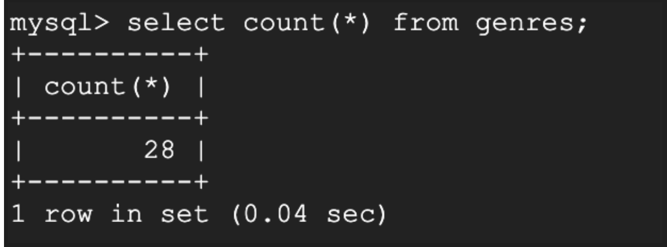
```
CREATE TABLE crew (
   nconst VARCHAR(20),
   primaryName  VARCHAR(255),
   knownForTitles VARCHAR(255),
   PRIMARY KEY (nconst)
);
```



```
mysql> select count(*) from crew;
+----------+
| count(*) |
+----------+
|    45877 |
+----------+
1 row in set (0.37 sec)
```

Genres

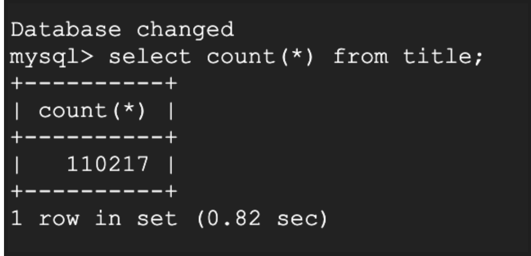
```
CREATE TABLE  genres (
   generes VARCHAR(225),
   id INT
   primary key (id)
);
```



```
mysql> select count(*) from genres;
+----------+
| count(*) |
+----------+
|       28 |
+----------+
1 row in set (0.04 sec)
```

Title

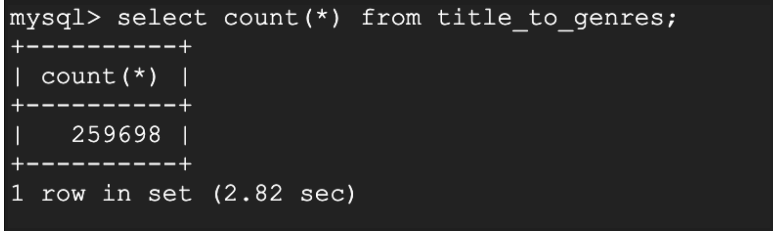
```
CREATE TABLE title (
    tconst VARCHAR(10) PRIMARY KEY,
    titleType VARCHAR(50),
    primaryTitle VARCHAR(255),
    originalTitle VARCHAR(225),
    isAdult INT,
    startYear INT,
    runtimeMinutes INT,
    averageRating FLOAT,
    numVotes INT
);
```



```
Database changed
mysql> select count(*) from title;
+----------+
| count(*) |
+----------+
|   110217 |
+----------+
1 row in set (0.82 sec)
```

Title_to_genres

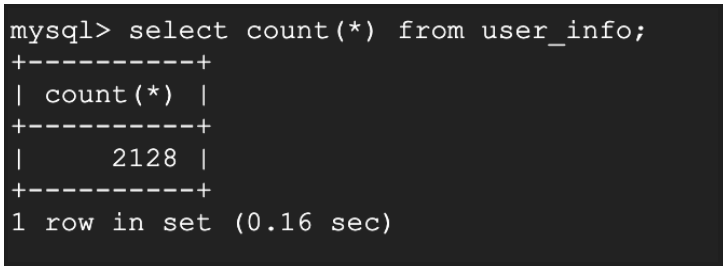
```
CREATE TABLE title_to_genres (
    tconst VARCHAR(20),
    id INT,
    PRIMARY KEY (tconst, id),
    FOREIGN KEY (tconst) REFERENCES titles(tconst),
    FOREIGN KEY (id) REFERENCES genres(id)
);
```



```
mysql> select count(*) from title_to_genres;
+----------+
| count(*) |
+----------+
|   259698 |
+----------+
1 row in set (2.82 sec)
```

User_info

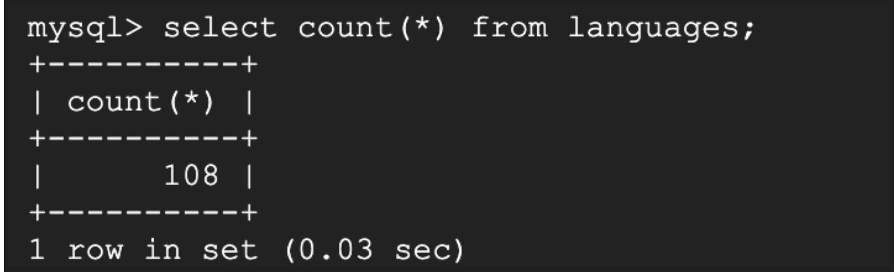
```
CREATE TABLE user_info (
    Title VARCHAR(255),
    Date VARCHAR(50),
    Season VARCHAR(50),
    Episode VARCHAR(50),
    id INT,
    titleType VARCHAR(50),
    PRIMARY KEY (id)
 );
```



```
mysql> select count(*) from user_info;
+----------+
| count(*) |
+----------+
|     2128 |
+----------+
1 row in set (0.16 sec)
```

languages

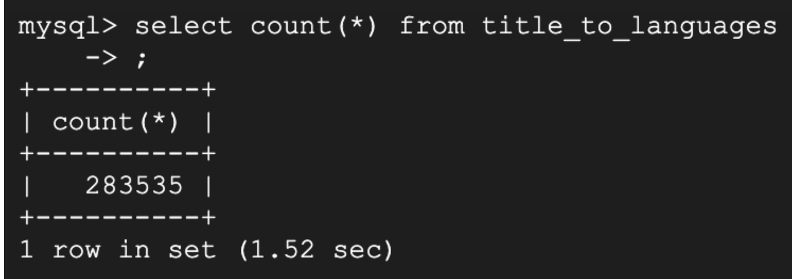
```
CREATE TABLE languages (
  language VARCHAR(20),
  id  INT,
  PRIMARY KEY (id)
);
```



```
mysql> select count(*) from languages;
+----------+
| count(*) |
+----------+
|      108 |
+----------+
1 row in set (0.03 sec)
```

title_to_languages

```
CREATE TABLE title_to_languages (
  titleId VARCHAR(20),
  id  INT,
  PRIMARY KEY (id, titleId),
  FOREIGN KEY (titleId) REFERENCES titles(tconst),
  FOREIGN KEY (id) REFERENCES languages(id)
);
```

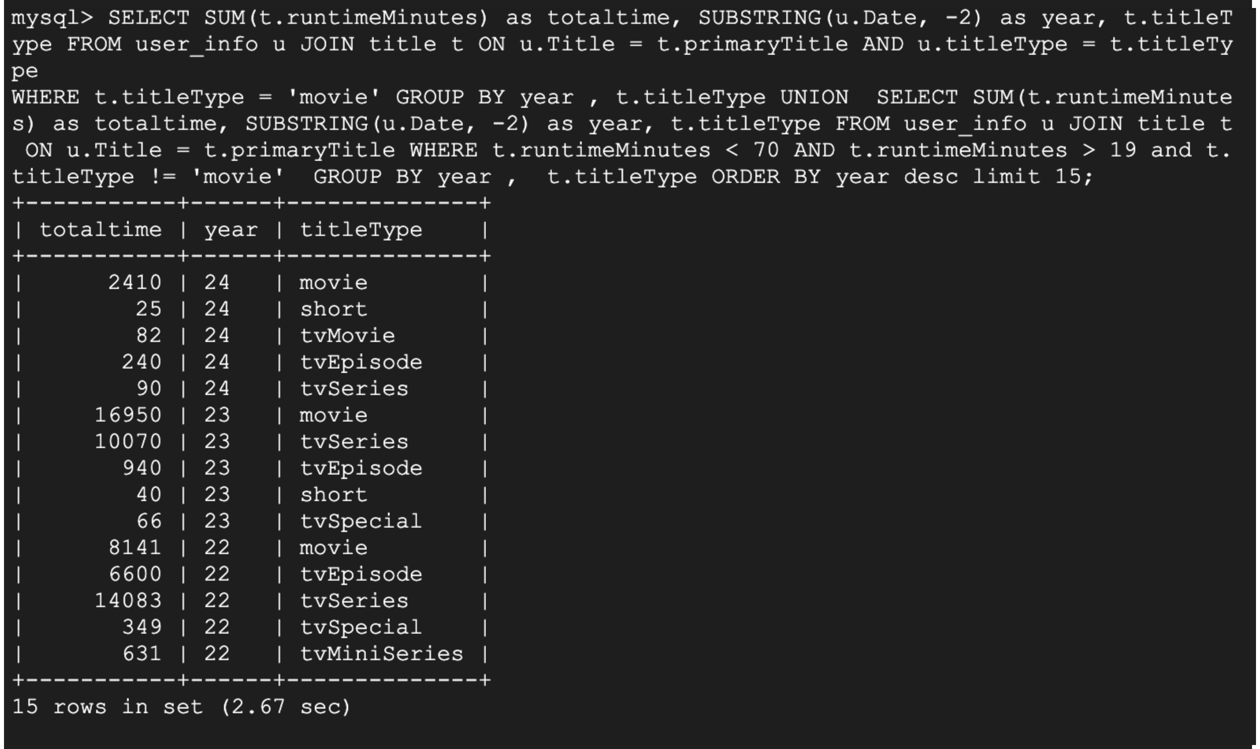


```
mysql> select count(*) from title_to_languages
    -> ;
+----------+
| count(*) |
+----------+
|   283535 |
+----------+
1 row in set (1.52 sec)
```

Part 2. Advanced Queries

Query 1:

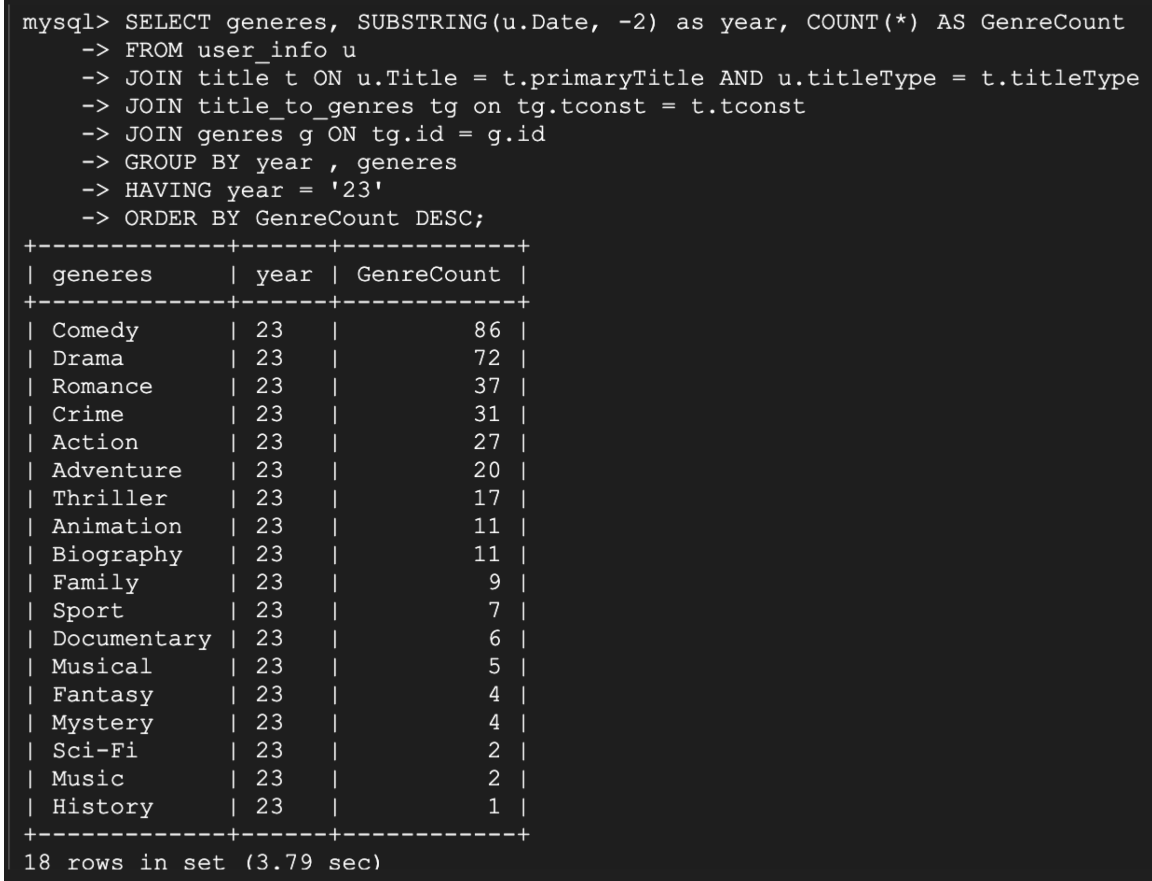
```
Total minutes watched ordered on year

SELECT SUM(t.runtimeMinutes) as totaltime, SUBSTRING(u.Date, -2) as year, t.titleType
FROM user_info u
JOIN title t ON u.Title = t.primaryTitle AND u.titleType = t.titleType
WHERE t.titleType = 'movie'
GROUP BY year , t.titleType

UNION

SELECT SUM(t.runtimeMinutes) as totaltime, SUBSTRING(u.Date, -2) as year, t.titleType
FROM user_info u
JOIN title t ON u.Title = t.primaryTitle
WHERE t.runtimeMinutes < 70 AND t.runtimeMinutes > 19 and t.titleType != 'movie'
GROUP BY year , t.titleType
ORDER BY year desc;
```



```
mysql> SELECT SUM(t.runtimeMinutes) as totaltime, SUBSTRING(u.Date, -2) as year, t.titleT
ype FROM user_info u JOIN title t ON u.Title = t.primaryTitle AND u.titleType = t.titleTy
pe
WHERE t.titleType = 'movie' GROUP BY year , t.titleType UNION  SELECT SUM(t.runtimeMinute
s) as totaltime, SUBSTRING(u.Date, -2) as year, t.titleType FROM user_info u JOIN title t
 ON u.Title = t.primaryTitle WHERE t.runtimeMinutes < 70 AND t.runtimeMinutes > 19 and t.
titleType != 'movie'  GROUP BY year ,  t.titleType ORDER BY year desc limit 15;
+-----------+------+--------------+
| totaltime | year | titleType    |
+-----------+------+--------------+
|      2410 | 24   | movie        |
|        25 | 24   | short        |
|        82 | 24   | tvMovie      |
|       240 | 24   | tvEpisode    |
|        90 | 24   | tvSeries     |
|     16950 | 23   | movie        |
|     10070 | 23   | tvSeries     |
|       940 | 23   | tvEpisode    |
|        40 | 23   | short        |
|        66 | 23   | tvSpecial    |
|      8141 | 22   | movie        |
|      6600 | 22   | tvEpisode    |
|     14083 | 22   | tvSeries     |
|       349 | 22   | tvSpecial    |
|       631 | 22   | tvMiniSeries |
+-----------+------+--------------+
15 rows in set (2.67 sec)
```

Query 2:

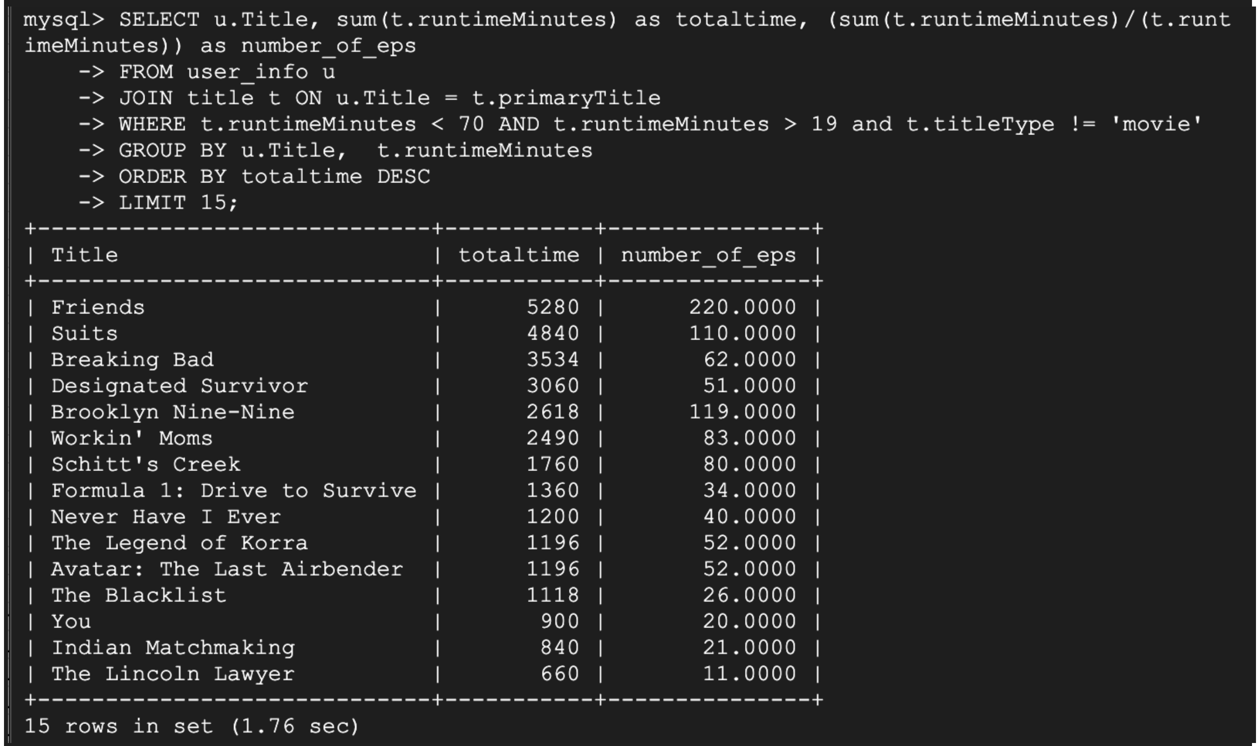
```
Most watched genre based on year watched


SELECT generes, SUBSTRING(u.Date, -2) as year, COUNT(*) AS GenreCount
FROM user_info u
JOIN title t ON u.Title = t.primaryTitle AND u.titleType = t.titleType
JOIN title_to_genres tg on tg.tconst = t.tconst
JOIN genres g ON tg.id = g.id
GROUP BY year , generes
HAVING year = '23'
ORDER BY GenreCount DESC;
```



```
mysql> SELECT generes, SUBSTRING(u.Date, -2) as year, COUNT(*) AS GenreCount
    -> FROM user_info u
    -> JOIN title t ON u.Title = t.primaryTitle AND u.titleType = t.titleType
    -> JOIN title_to_genres tg on tg.tconst = t.tconst
    -> JOIN genres g ON tg.id = g.id
    -> GROUP BY year , generes
    -> HAVING year = '23'
    -> ORDER BY GenreCount DESC;
+-------------+------+------------+
| generes     | year | GenreCount |
+-------------+------+------------+
| Comedy      | 23   |         86 |
| Drama       | 23   |         72 |
| Romance     | 23   |         37 |
| Crime       | 23   |         31 |
| Action      | 23   |         27 |
| Adventure   | 23   |         20 |
| Thriller    | 23   |         17 |
| Animation   | 23   |         11 |
| Biography   | 23   |         11 |
| Family      | 23   |          9 |
| Sport       | 23   |          7 |
| Documentary | 23   |          6 |
| Musical     | 23   |          5 |
| Fantasy     | 23   |          4 |
| Mystery     | 23   |          4 |
| Sci-Fi      | 23   |          2 |
| Music       | 23   |          2 |
| History     | 23   |          1 |
+-------------+------+------------+
18 rows in set (3.79 sec)
```

Query 3:

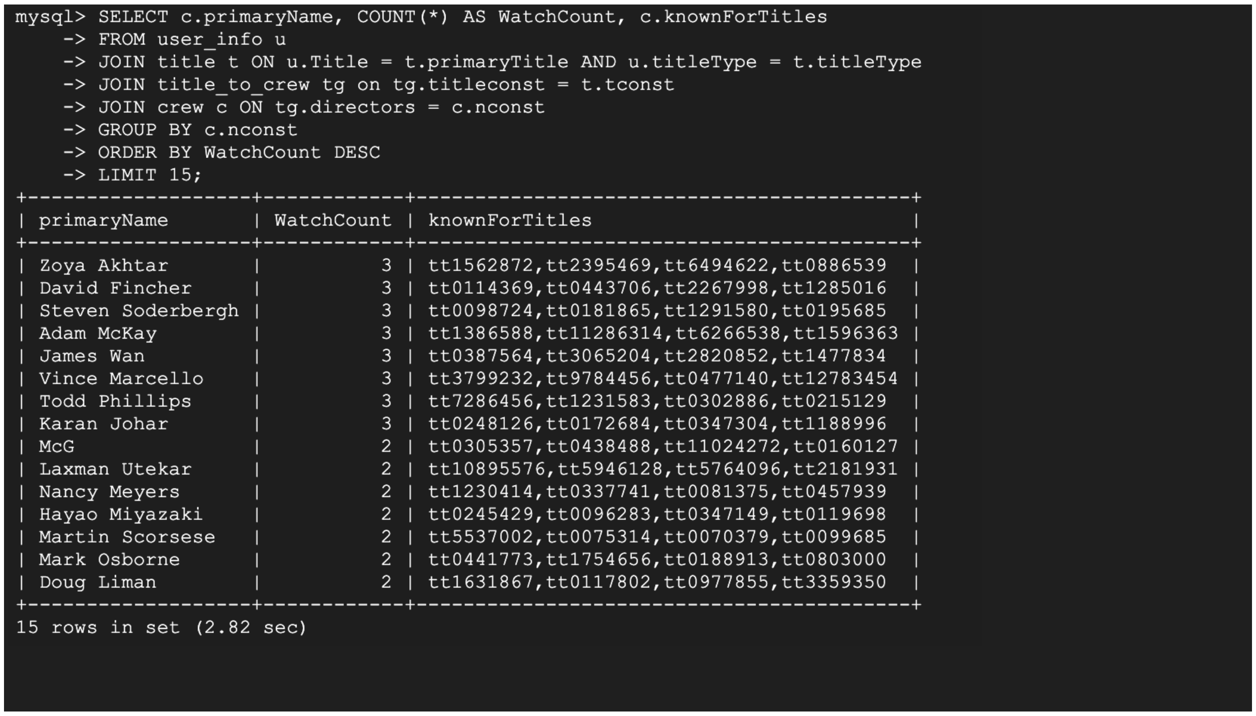
```
most binged tv show


SELECT u.Title, sum(t.runtimeMinutes) as totaltime,
(sum(t.runtimeMinutes)/(t.runtimeMinutes)) as number_of_eps
FROM user_info u
JOIN title t ON u.Title = t.primaryTitle
WHERE t.runtimeMinutes < 70 AND t.runtimeMinutes > 19 and t.titleType != 'movie'
GROUP BY u.Title,  t.runtimeMinutes
ORDER BY totaltime DESC
LIMIT 15;
```



```
mysql> SELECT u.Title, sum(t.runtimeMinutes) as totaltime, (sum(t.runtimeMinutes)/(t.runt
imeMinutes)) as number_of_eps
    -> FROM user_info u
    -> JOIN title t ON u.Title = t.primaryTitle
    -> WHERE t.runtimeMinutes < 70 AND t.runtimeMinutes > 19 and t.titleType != 'movie'
    -> GROUP BY u.Title,  t.runtimeMinutes
    -> ORDER BY totaltime DESC
    -> LIMIT 15;
+-----------------------------+-----------+---------------+
| Title                       | totaltime | number_of_eps |
+-----------------------------+-----------+---------------+
| Friends                     |      5280 |      220.0000 |
| Suits                       |      4840 |      110.0000 |
| Breaking Bad                |      3534 |       62.0000 |
| Designated Survivor         |      3060 |       51.0000 |
| Brooklyn Nine-Nine          |      2618 |      119.0000 |
| Workin' Moms                |      2490 |       83.0000 |
| Schitt's Creek              |      1760 |       80.0000 |
| Formula 1: Drive to Survive |      1360 |       34.0000 |
| Never Have I Ever           |      1200 |       40.0000 |
| The Legend of Korra         |      1196 |       52.0000 |
| Avatar: The Last Airbender  |      1196 |       52.0000 |
| The Blacklist               |      1118 |       26.0000 |
| You                         |       900 |       20.0000 |
| Indian Matchmaking          |       840 |       21.0000 |
| The Lincoln Lawyer          |       660 |       11.0000 |
+-----------------------------+-----------+---------------+
15 rows in set (1.76 sec)
```

Query 4:

```
Favourite movie director or TV show director


SELECT c.primaryName, COUNT(*) AS WatchCount, c.knownForTitles
FROM user_info u
JOIN title t ON u.Title = t.primaryTitle AND u.titleType = t.titleType
JOIN title_to_crew tg on tg.titleconst = t.tconst
JOIN crew c ON tg.directors = c.nconst
GROUP BY c.nconst
ORDER BY WatchCount DESC
LIMIT 15;
```

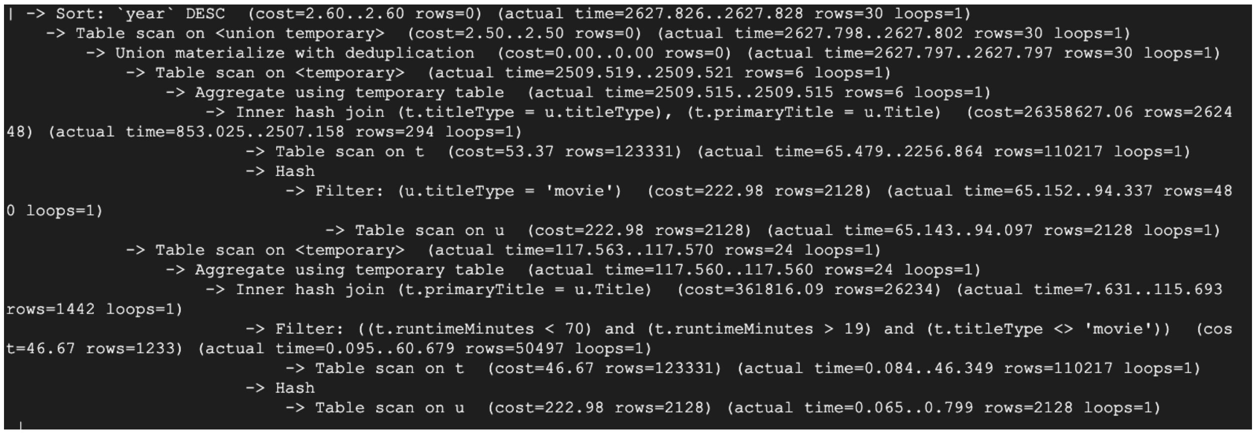


```
mysql> SELECT c.primaryName, COUNT(*) AS WatchCount, c.knownForTitles
    -> FROM user_info u
    -> JOIN title t ON u.Title = t.primaryTitle AND u.titleType = t.titleType
    -> JOIN title_to_crew tg on tg.titleconst = t.tconst
    -> JOIN crew c ON tg.directors = c.nconst
    -> GROUP BY c.nconst
    -> ORDER BY WatchCount DESC
    -> LIMIT 15;
+-------------------+------------+-----------------------------------------+
| primaryName       | WatchCount | knownForTitles                          |
+-------------------+------------+-----------------------------------------+
| Zoya Akhtar       |          3 | tt1562872,tt2395469,tt6494622,tt0886539  |
| David Fincher     |          3 | tt0114369,tt0443706,tt2267998,tt1285016  |
| Steven Soderbergh |          3 | tt0098724,tt0181865,tt1291580,tt0195685  |
| Adam McKay        |          3 | tt1386588,tt11286314,tt6266538,tt1596363 |
| James Wan         |          3 | tt0387564,tt3065204,tt2820852,tt1477834  |
| Vince Marcello    |          3 | tt3799232,tt9784456,tt0477140,tt12783454 |
| Todd Phillips     |          3 | tt7286456,tt1231583,tt0302886,tt0215129  |
| Karan Johar       |          3 | tt0248126,tt0172684,tt0347304,tt1188996  |
| McG               |          2 | tt0305357,tt0438488,tt11024272,tt0160127 |
| Laxman Utekar     |          2 | tt10895576,tt5946128,tt5764096,tt2181931 |
| Nancy Meyers      |          2 | tt1230414,tt0337741,tt0081375,tt0457939  |
| Hayao Miyazaki    |          2 | tt0245429,tt0096283,tt0347149,tt0119698  |
| Martin Scorsese   |          2 | tt5537002,tt0075314,tt0070379,tt0099685  |
| Mark Osborne      |          2 | tt0441773,tt1754656,tt0188913,tt0803000  |
| Doug Liman        |          2 | tt1631867,tt0117802,tt0977855,tt3359350  |
+-------------------+------------+-----------------------------------------+
15 rows in set (2.82 sec)
```

<u>Part 3: Indexing Analysis</u>

Query 1(Time watched**):**

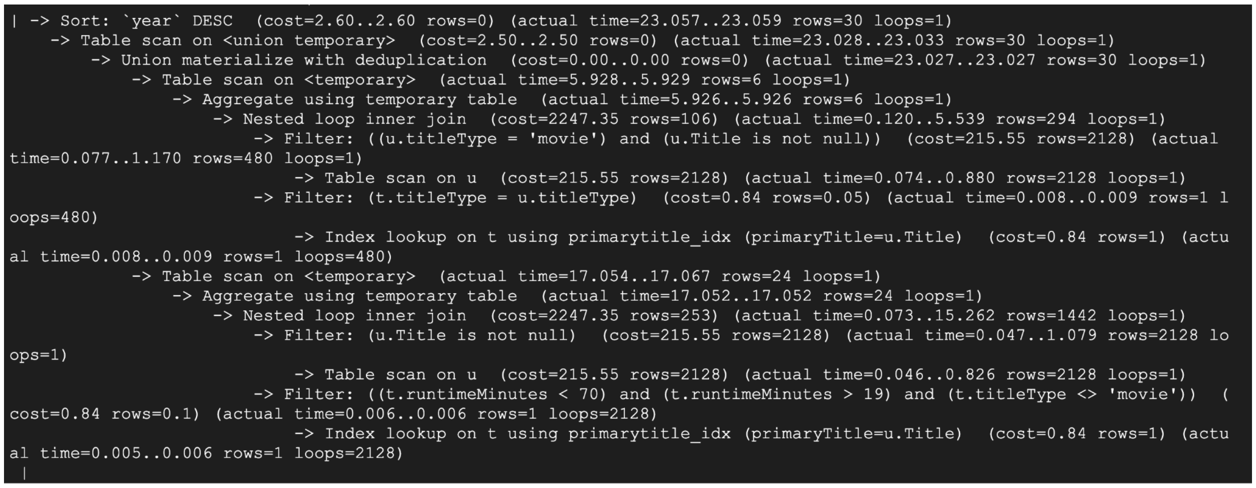
Explain analyze on basic query



```
| -> Sort: `year` DESC  (cost=2.60..2.60 rows=0) (actual time=2627.826..2627.828 rows=30 loops=1)
    -> Table scan on <union temporary>  (cost=2.50..2.50 rows=0) (actual time=2627.798..2627.802 rows=30 loops=1)
        -> Union materialize with deduplication  (cost=0.00..0.00 rows=0) (actual time=2627.797..2627.797 rows=30 loops=1)
            -> Table scan on <temporary>  (actual time=2509.519..2509.521 rows=6 loops=1)
                -> Aggregate using temporary table  (actual time=2509.515..2509.515 rows=6 loops=1)
                    -> Inner hash join (t.titleType = u.titleType), (t.primaryTitle = u.Title)  (cost=26358627.06 rows=2624
48) (actual time=853.025..2507.158 rows=294 loops=1)
                        -> Table scan on t  (cost=53.37 rows=123331) (actual time=65.479..2256.864 rows=110217 loops=1)
                        -> Hash
                            -> Filter: (u.titleType = 'movie')  (cost=222.98 rows=2128) (actual time=65.152..94.337 rows=48
0 loops=1)
                                -> Table scan on u  (cost=222.98 rows=2128) (actual time=65.143..94.097 rows=2128 loops=1)
            -> Table scan on <temporary>  (actual time=117.563..117.570 rows=24 loops=1)
                -> Aggregate using temporary table  (actual time=117.560..117.560 rows=24 loops=1)
                    -> Inner hash join (t.primaryTitle = u.Title)  (cost=361816.09 rows=26234) (actual time=7.631..115.693
rows=1442 loops=1)
                        -> Filter: ((t.runtimeMinutes < 70) and (t.runtimeMinutes > 19) and (t.titleType <> 'movie'))  (cos
t=46.67 rows=1233) (actual time=0.095..60.679 rows=50497 loops=1)
                            -> Table scan on t  (cost=46.67 rows=123331) (actual time=0.084..46.349 rows=110217 loops=1)
                        -> Hash
                            -> Table scan on u  (cost=222.98 rows=2128) (actual time=0.065..0.799 rows=2128 loops=1)
 |
```

Cost baseline: 26358627.06

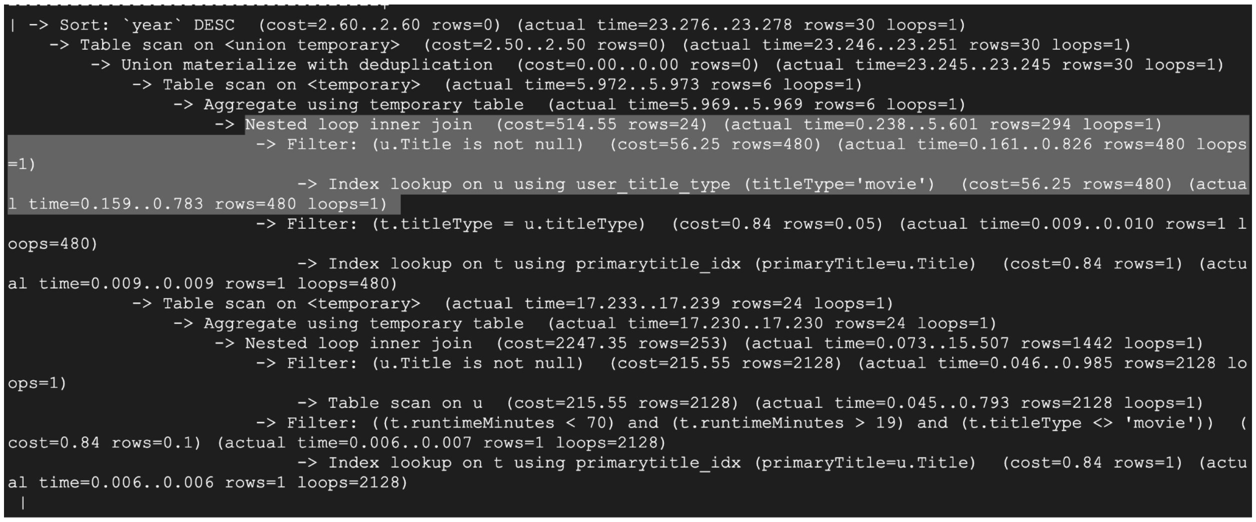
Index 1:
create index primarytitle_idx on title(primaryTitle);



```
| -> Sort: `year` DESC  (cost=2.60..2.60 rows=0) (actual time=23.057..23.059 rows=30 loops=1)
    -> Table scan on <union temporary>  (cost=2.50..2.50 rows=0) (actual time=23.028..23.033 rows=30 loops=1)
        -> Union materialize with deduplication  (cost=0.00..0.00 rows=0) (actual time=23.027..23.027 rows=30 loops=1)
            -> Table scan on <temporary>  (actual time=5.928..5.929 rows=6 loops=1)
                -> Aggregate using temporary table  (actual time=5.926..5.926 rows=6 loops=1)
                    -> Nested loop inner join  (cost=2247.35 rows=106) (actual time=0.120..5.539 rows=294 loops=1)
                        -> Filter: ((u.titleType = 'movie') and (u.Title is not null))  (cost=215.55 rows=2128) (actual
time=0.077..1.170 rows=480 loops=1)
                            -> Table scan on u  (cost=215.55 rows=2128) (actual time=0.074..0.880 rows=2128 loops=1)
                        -> Filter: (t.titleType = u.titleType)  (cost=0.84 rows=0.05) (actual time=0.008..0.009 rows=1 l
oops=480)
                            -> Index lookup on t using primarytitle_idx (primaryTitle=u.Title)  (cost=0.84 rows=1) (actu
al time=0.008..0.009 rows=1 loops=480)
            -> Table scan on <temporary>  (actual time=17.054..17.067 rows=24 loops=1)
                -> Aggregate using temporary table  (actual time=17.052..17.052 rows=24 loops=1)
                    -> Nested loop inner join  (cost=2247.35 rows=253) (actual time=0.073..15.262 rows=1442 loops=1)
                        -> Filter: (u.Title is not null)  (cost=215.55 rows=2128) (actual time=0.047..1.079 rows=2128 lo
ops=1)
                            -> Table scan on u  (cost=215.55 rows=2128) (actual time=0.046..0.826 rows=2128 loops=1)
                        -> Filter: ((t.runtimeMinutes < 70) and (t.runtimeMinutes > 19) and (t.titleType <> 'movie'))  (
cost=0.84 rows=0.1) (actual time=0.006..0.006 rows=1 loops=2128)
                            -> Index lookup on t using primarytitle_idx (primaryTitle=u.Title)  (cost=0.84 rows=1) (actu
al time=0.005..0.006 rows=1 loops=2128)
 |
```

Cost after index 1: 2247.35

Index 2:
create index user_title_type on user_info(titleType);

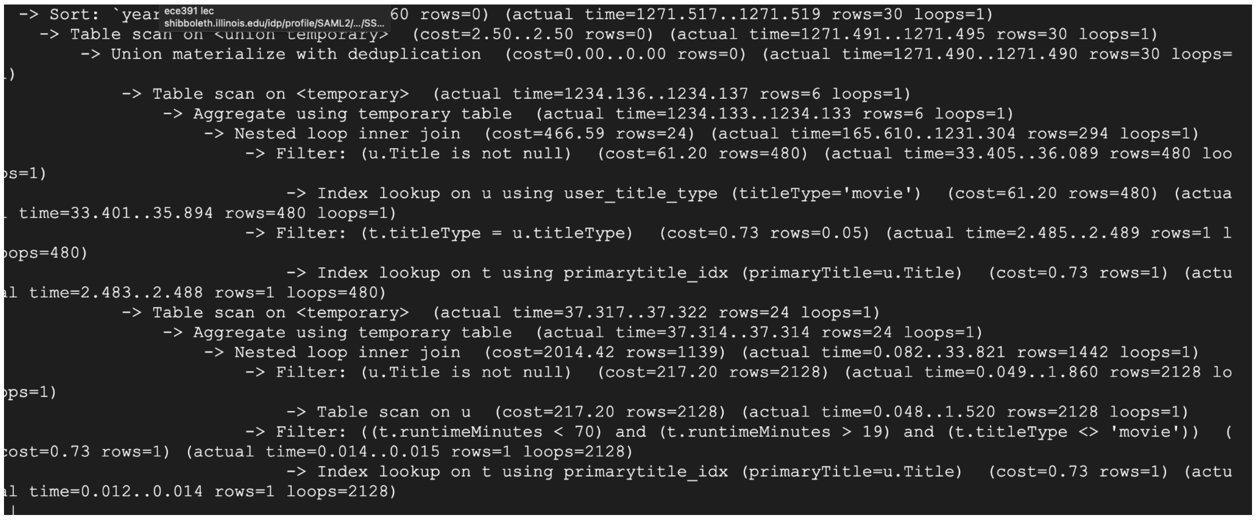


```
| -> Sort: `year` DESC  (cost=2.60..2.60 rows=0) (actual time=23.276..23.278 rows=30 loops=1)
    -> Table scan on <union temporary>  (cost=2.50..2.50 rows=0) (actual time=23.246..23.251 rows=30 loops=1)
        -> Union materialize with deduplication  (cost=0.00..0.00 rows=0) (actual time=23.245..23.245 rows=30 loops=1)
            -> Table scan on <temporary>  (actual time=5.972..5.973 rows=6 loops=1)
                -> Aggregate using temporary table  (actual time=5.969..5.969 rows=6 loops=1)
                    -> Nested loop inner join  (cost=514.55 rows=24) (actual time=0.238..5.601 rows=294 loops=1)
                        -> Filter: (u.Title is not null)  (cost=56.25 rows=480) (actual time=0.161..0.826 rows=480 loops
=1)
                            -> Index lookup on u using user_title_type (titleType='movie')  (cost=56.25 rows=480) (actua
l time=0.159..0.783 rows=480 loops=1)
                        -> Filter: (t.titleType = u.titleType)  (cost=0.84 rows=0.05) (actual time=0.009..0.010 rows=1 l
oops=480)
                            -> Index lookup on t using primarytitle_idx (primaryTitle=u.Title)  (cost=0.84 rows=1) (actu
al time=0.009..0.009 rows=1 loops=480)
            -> Table scan on <temporary>  (actual time=17.233..17.239 rows=24 loops=1)
                -> Aggregate using temporary table  (actual time=17.230..17.230 rows=24 loops=1)
                    -> Nested loop inner join  (cost=2247.35 rows=253) (actual time=0.073..15.507 rows=1442 loops=1)
                        -> Filter: (u.Title is not null)  (cost=215.55 rows=2128) (actual time=0.046..0.985 rows=2128 lo
ops=1)
                            -> Table scan on u  (cost=215.55 rows=2128) (actual time=0.045..0.793 rows=2128 loops=1)
                        -> Filter: ((t.runtimeMinutes < 70) and (t.runtimeMinutes > 19) and (t.titleType <> 'movie'))  (
cost=0.84 rows=0.1) (actual time=0.006..0.007 rows=1 loops=2128)
                            -> Index lookup on t using primarytitle_idx (primaryTitle=u.Title)  (cost=0.84 rows=1) (actu
al time=0.006..0.006 rows=1 loops=2128)
 |
```

Cost after index 2: 2247.55

Note: Cost of first join decreases from 2247 to 514

Index 3:
create index user_title on user_info(Title);



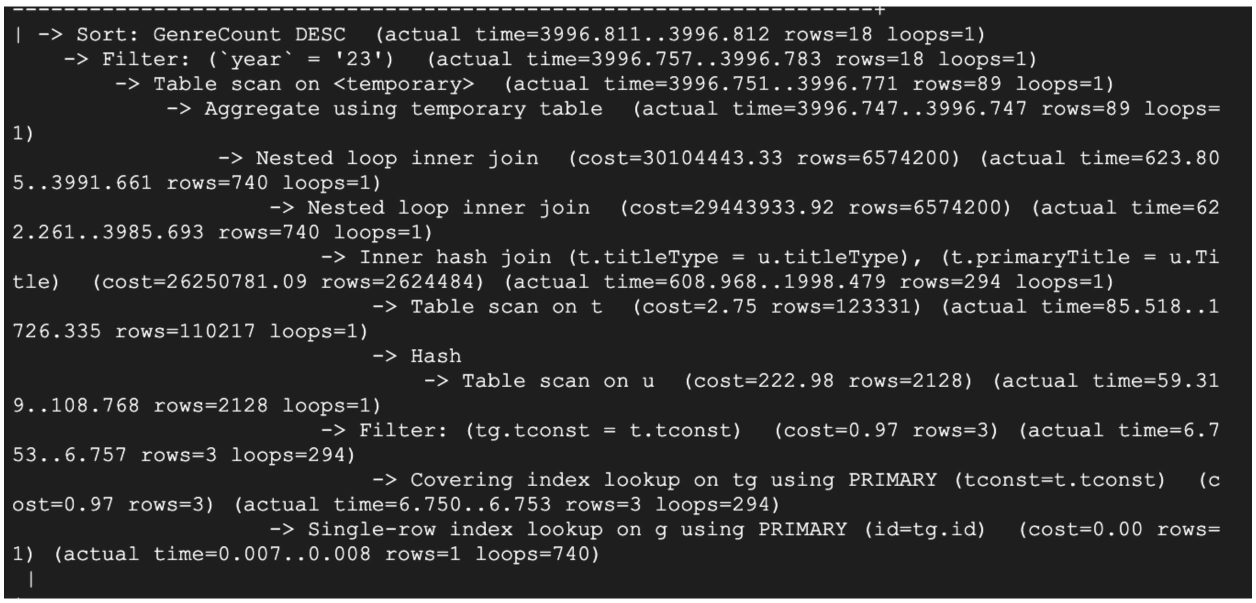
```
 -> Sort: `year ece391 lec shibboleth.illinois.edu/idp/profile/SAML2/.../SS... 60 rows=0) (actual time=1271.517..1271.519 rows=30 loops=1)
   -> Table scan on <union temporary>  (cost=2.50..2.50 rows=0) (actual time=1271.491..1271.495 rows=30 loops=1)
       -> Union materialize with deduplication  (cost=0.00..0.00 rows=0) (actual time=1271.490..1271.490 rows=30 loops=
)
           -> Table scan on <temporary>  (actual time=1234.136..1234.137 rows=6 loops=1)
               -> Aggregate using temporary table  (actual time=1234.133..1234.133 rows=6 loops=1)
                   -> Nested loop inner join  (cost=466.59 rows=24) (actual time=165.610..1231.304 rows=294 loops=1)
                       -> Filter: (u.Title is not null)  (cost=61.20 rows=480) (actual time=33.405..36.089 rows=480 loo
s=1)
                           -> Index lookup on u using user_title_type (titleType='movie')  (cost=61.20 rows=480) (actua
 time=33.401..35.894 rows=480 loops=1)
                       -> Filter: (t.titleType = u.titleType)  (cost=0.73 rows=0.05) (actual time=2.485..2.489 rows=1 l
ops=480)
                           -> Index lookup on t using primarytitle_idx (primaryTitle=u.Title)  (cost=0.73 rows=1) (actu
l time=2.483..2.488 rows=1 loops=480)
           -> Table scan on <temporary>  (actual time=37.317..37.322 rows=24 loops=1)
               -> Aggregate using temporary table  (actual time=37.314..37.314 rows=24 loops=1)
                   -> Nested loop inner join  (cost=2014.42 rows=1139) (actual time=0.082..33.821 rows=1442 loops=1)
                       -> Filter: (u.Title is not null)  (cost=217.20 rows=2128) (actual time=0.049..1.860 rows=2128 lo
ps=1)
                           -> Table scan on u  (cost=217.20 rows=2128) (actual time=0.048..1.520 rows=2128 loops=1)
                       -> Filter: ((t.runtimeMinutes < 70) and (t.runtimeMinutes > 19) and (t.titleType <> 'movie'))  (
ost=0.73 rows=1) (actual time=0.014..0.015 rows=1 loops=2128)
                           -> Index lookup on t using primarytitle_idx (primaryTitle=u.Title)  (cost=0.73 rows=1) (actu
l time=0.012..0.014 rows=1 loops=2128)
```

Cost after index 3: 2014.42

Justification:
When looking at the indexes for query #1, the first index ended up being very similar to query 1. The overall cost after the first index was 2247.35. The output for the second index showed that by using user_title_type, the join cost went down from 2447 to 466.59. This is a result of the index being on the join which can significantly reduce search time and also reduces the overall cost of the join as well. For index 3, the query brought the cost down after the union. However, the index isn't shown as being the optimizer so it might be the wrong choice to use it. This could be happening because the u.Ttile column doesn't have enough values and therefore the query might require a whole scan of the table. Therefore, index 3 might not be the best design for this query.

Query 2 :

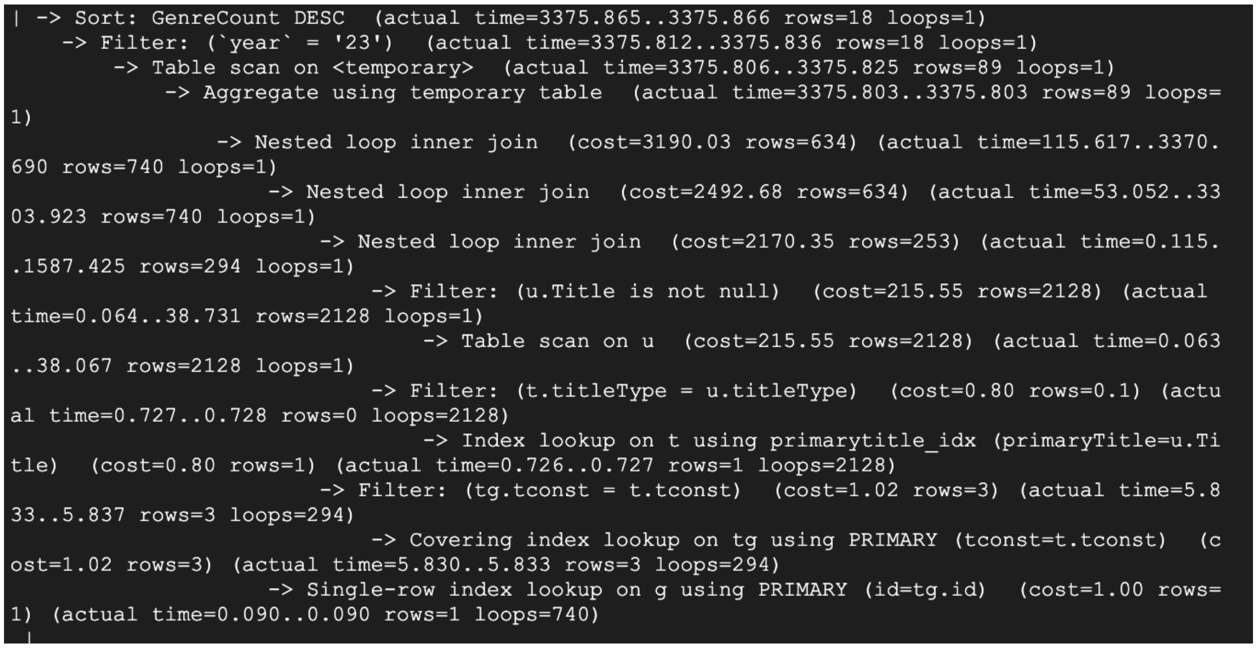
(Genre)
Explain analyze on basic query



```
-----------------------------------------------------------------+
| -> Sort: GenreCount DESC  (actual time=3996.811..3996.812 rows=18 loops=1)
    -> Filter: (`year` = '23')  (actual time=3996.757..3996.783 rows=18 loops=1)
        -> Table scan on <temporary>  (actual time=3996.751..3996.771 rows=89 loops=1)
            -> Aggregate using temporary table  (actual time=3996.747..3996.747 rows=89 loops=
1)
                -> Nested loop inner join  (cost=30104443.33 rows=6574200) (actual time=623.80
5..3991.661 rows=740 loops=1)
                    -> Nested loop inner join  (cost=29443933.92 rows=6574200) (actual time=62
2.261..3985.693 rows=740 loops=1)
                        -> Inner hash join (t.titleType = u.titleType), (t.primaryTitle = u.Ti
tle)  (cost=26250781.09 rows=2624484) (actual time=608.968..1998.479 rows=294 loops=1)
                            -> Table scan on t  (cost=2.75 rows=123331) (actual time=85.518..1
726.335 rows=110217 loops=1)
                            -> Hash
                                -> Table scan on u  (cost=222.98 rows=2128) (actual time=59.31
9..108.768 rows=2128 loops=1)
                        -> Filter: (tg.tconst = t.tconst)  (cost=0.97 rows=3) (actual time=6.7
53..6.757 rows=3 loops=294)
                            -> Covering index lookup on tg using PRIMARY (tconst=t.tconst)  (c
ost=0.97 rows=3) (actual time=6.750..6.753 rows=3 loops=294)
                    -> Single-row index lookup on g using PRIMARY (id=tg.id)  (cost=0.00 rows=
1) (actual time=0.007..0.008 rows=1 loops=740)
 |
```

Cost baseline: 30104443.33

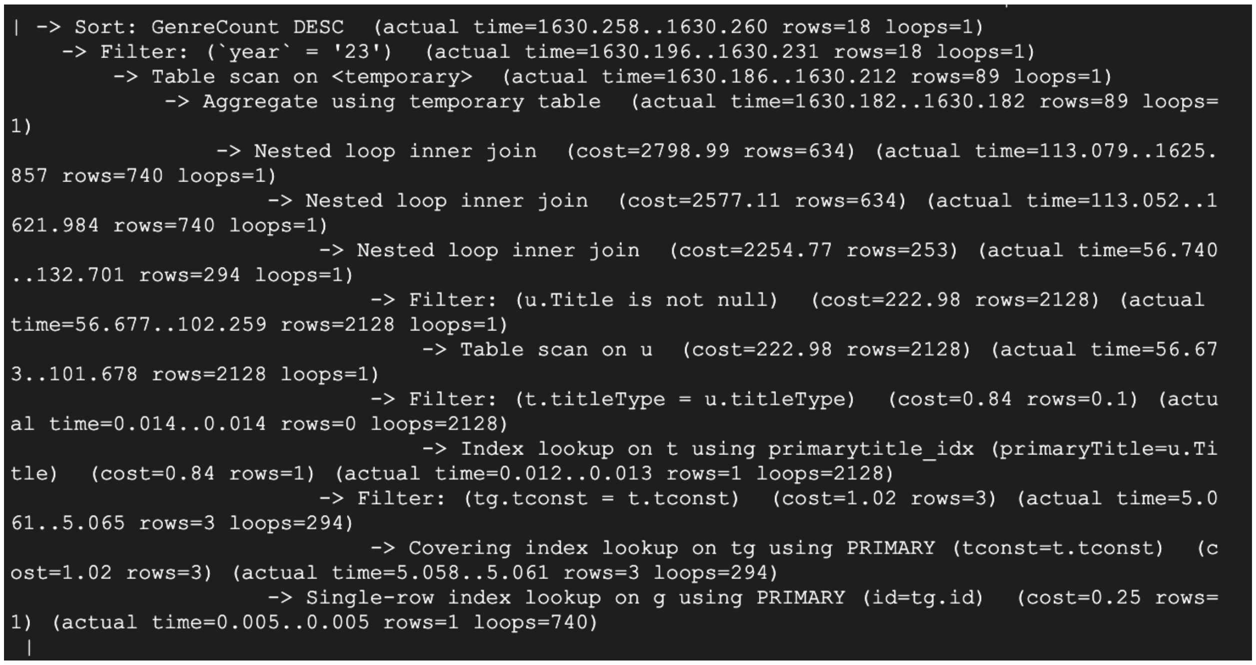
Index 1:
create index primarytitle_idx on title(primaryTitle);



```
| -> Sort: GenreCount DESC  (actual time=3375.865..3375.866 rows=18 loops=1)
    -> Filter: (`year` = '23')  (actual time=3375.812..3375.836 rows=18 loops=1)
        -> Table scan on <temporary>  (actual time=3375.806..3375.825 rows=89 loops=1)
            -> Aggregate using temporary table  (actual time=3375.803..3375.803 rows=89 loops=
1)
                -> Nested loop inner join  (cost=3190.03 rows=634) (actual time=115.617..3370.
690 rows=740 loops=1)
                    -> Nested loop inner join  (cost=2492.68 rows=634) (actual time=53.052..33
03.923 rows=740 loops=1)
                        -> Nested loop inner join  (cost=2170.35 rows=253) (actual time=0.115.
.1587.425 rows=294 loops=1)
                            -> Filter: (u.Title is not null)  (cost=215.55 rows=2128) (actual
time=0.064..38.731 rows=2128 loops=1)
                                -> Table scan on u  (cost=215.55 rows=2128) (actual time=0.063
..38.067 rows=2128 loops=1)
                            -> Filter: (t.titleType = u.titleType)  (cost=0.80 rows=0.1) (actu
al time=0.727..0.728 rows=0 loops=2128)
                                -> Index lookup on t using primarytitle_idx (primaryTitle=u.Ti
tle)  (cost=0.80 rows=1) (actual time=0.726..0.727 rows=1 loops=2128)
                        -> Filter: (tg.tconst = t.tconst)  (cost=1.02 rows=3) (actual time=5.8
33..5.837 rows=3 loops=294)
                            -> Covering index lookup on tg using PRIMARY (tconst=t.tconst)  (c
ost=1.02 rows=3) (actual time=5.830..5.833 rows=3 loops=294)
                    -> Single-row index lookup on g using PRIMARY (id=tg.id)  (cost=1.00 rows=
1) (actual time=0.090..0.090 rows=1 loops=740)
 |
```

Cost after index 1: 3190.03

Index 2:
create index genres_idx on genres(generes);

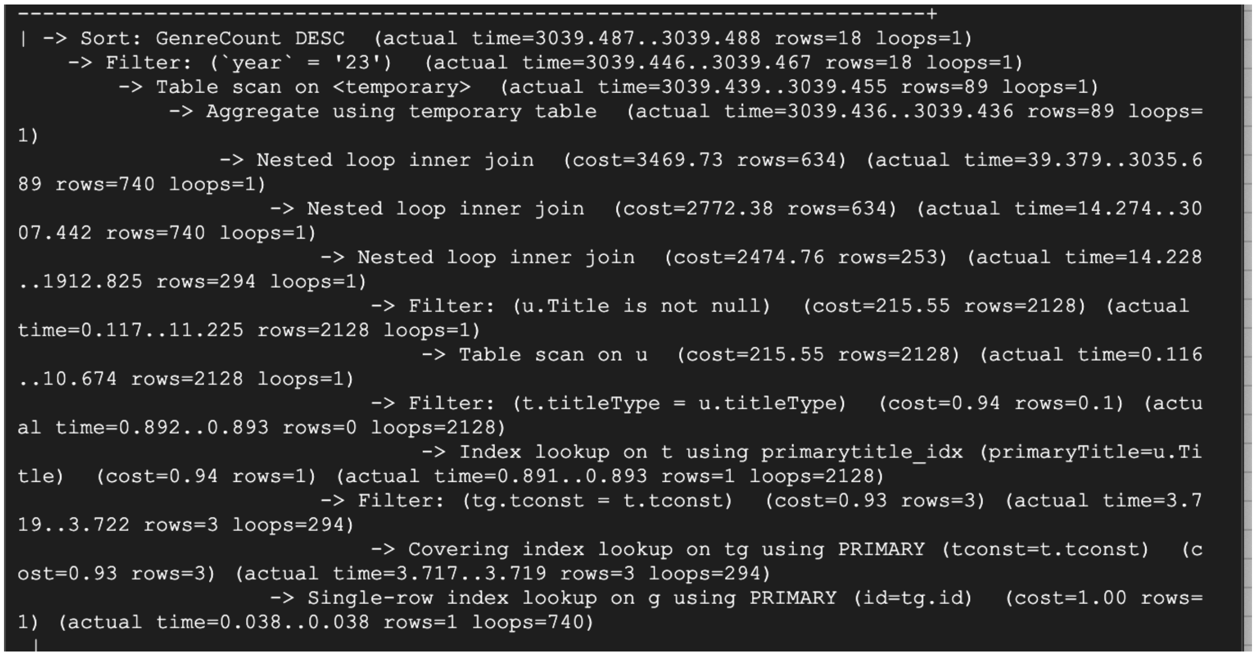


```
| -> Sort: GenreCount DESC  (actual time=1630.258..1630.260 rows=18 loops=1)
    -> Filter: (`year` = '23')  (actual time=1630.196..1630.231 rows=18 loops=1)
        -> Table scan on <temporary>  (actual time=1630.186..1630.212 rows=89 loops=1)
            -> Aggregate using temporary table  (actual time=1630.182..1630.182 rows=89 loops=
1)
                -> Nested loop inner join  (cost=2798.99 rows=634) (actual time=113.079..1625.
857 rows=740 loops=1)
                    -> Nested loop inner join  (cost=2577.11 rows=634) (actual time=113.052..1
621.984 rows=740 loops=1)
                        -> Nested loop inner join  (cost=2254.77 rows=253) (actual time=56.740
..132.701 rows=294 loops=1)
                            -> Filter: (u.Title is not null)  (cost=222.98 rows=2128) (actual
time=56.677..102.259 rows=2128 loops=1)
                                -> Table scan on u  (cost=222.98 rows=2128) (actual time=56.67
3..101.678 rows=2128 loops=1)
                            -> Filter: (t.titleType = u.titleType)  (cost=0.84 rows=0.1) (actu
al time=0.014..0.014 rows=0 loops=2128)
                                -> Index lookup on t using primarytitle_idx (primaryTitle=u.Ti
tle)  (cost=0.84 rows=1) (actual time=0.012..0.013 rows=1 loops=2128)
                        -> Filter: (tg.tconst = t.tconst)  (cost=1.02 rows=3) (actual time=5.0
61..5.065 rows=3 loops=294)
                            -> Covering index lookup on tg using PRIMARY (tconst=t.tconst)  (c
ost=1.02 rows=3) (actual time=5.058..5.061 rows=3 loops=294)
                    -> Single-row index lookup on g using PRIMARY (id=tg.id)  (cost=0.25 rows=
1) (actual time=0.005..0.005 rows=1 loops=740)
 |
```

Cost after index 2 : 2798.99

Index 3:

create index user_Type_idx on user_info(titleType);



```
-----------------------------------------------------------------------+
| -> Sort: GenreCount DESC  (actual time=3039.487..3039.488 rows=18 loops=1)
    -> Filter: (`year` = '23')  (actual time=3039.446..3039.467 rows=18 loops=1)
        -> Table scan on <temporary>  (actual time=3039.439..3039.455 rows=89 loops=1)
            -> Aggregate using temporary table  (actual time=3039.436..3039.436 rows=89 loops=
1)
                -> Nested loop inner join  (cost=3469.73 rows=634) (actual time=39.379..3035.6
89 rows=740 loops=1)
                    -> Nested loop inner join  (cost=2772.38 rows=634) (actual time=14.274..30
07.442 rows=740 loops=1)
                        -> Nested loop inner join  (cost=2474.76 rows=253) (actual time=14.228
..1912.825 rows=294 loops=1)
                            -> Filter: (u.Title is not null)  (cost=215.55 rows=2128) (actual
time=0.117..11.225 rows=2128 loops=1)
                                -> Table scan on u  (cost=215.55 rows=2128) (actual time=0.116
..10.674 rows=2128 loops=1)
                            -> Filter: (t.titleType = u.titleType)  (cost=0.94 rows=0.1) (actu
al time=0.892..0.893 rows=0 loops=2128)
                                -> Index lookup on t using primarytitle_idx (primaryTitle=u.Ti
tle)  (cost=0.94 rows=1) (actual time=0.891..0.893 rows=1 loops=2128)
                        -> Filter: (tg.tconst = t.tconst)  (cost=0.93 rows=3) (actual time=3.7
19..3.722 rows=3 loops=294)
                            -> Covering index lookup on tg using PRIMARY (tconst=t.tconst)  (c
ost=0.93 rows=3) (actual time=3.717..3.719 rows=3 loops=294)
                    -> Single-row index lookup on g using PRIMARY (id=tg.id)  (cost=1.00 rows=
1) (actual time=0.038..0.038 rows=1 loops=740)
 |
```

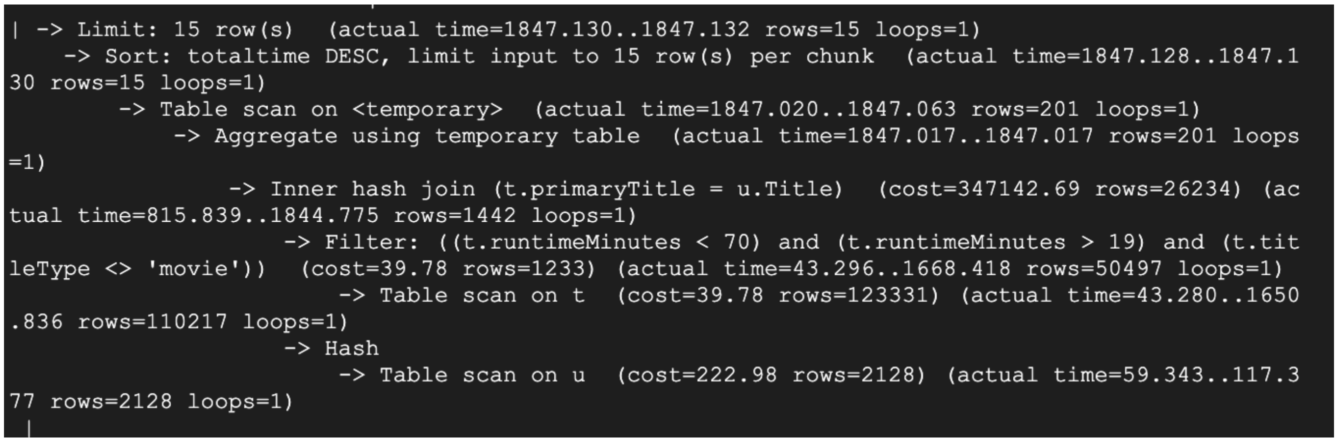
Cost after index 3: 3469.73

Justification:

For query #2, the indexes were tried on title_type and on year but these didn't have the same performance and/or had no effect on the cost. For the first index, the cost of the join was reduced from 26250781 to just 0.94 by doing a join on the primary title and user title. The index on them allows for an easier join. For the second index, we tried u.titleType and t.tytleType but the performance decreased on both of them. The assumption here is that there is low cardinality of the two and the indexing may not have been as effective because of that. For index 2, the cost was only decreased by a little and the optimizer did not choose to use the index. This could have happened because of low cardinality for the index which made it less effective in lowering the cost.

Query 3:

Explain analyze on basic query

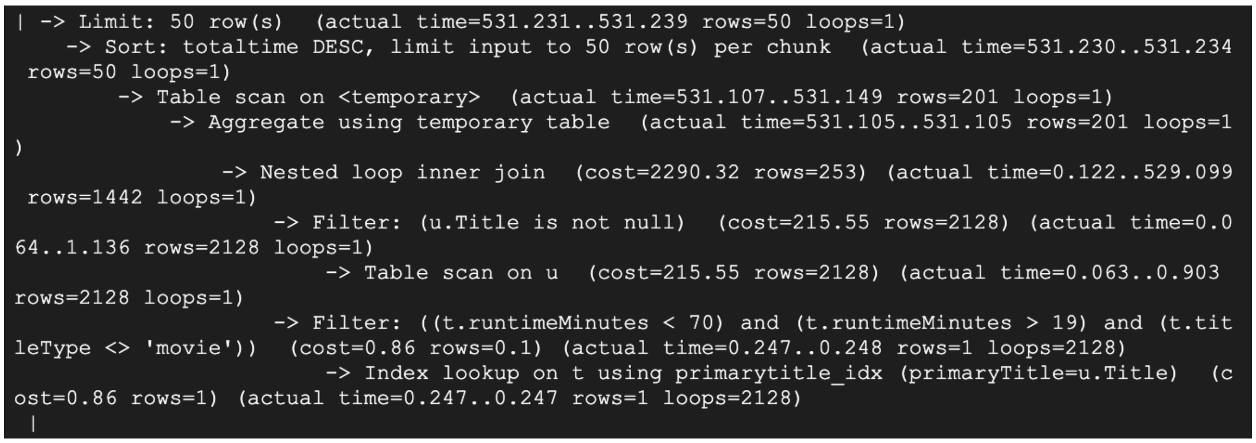


```
| -> Limit: 15 row(s)  (actual time=1847.130..1847.132 rows=15 loops=1)
    -> Sort: totaltime DESC, limit input to 15 row(s) per chunk  (actual time=1847.128..1847.1
30 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=1847.020..1847.063 rows=201 loops=1)
            -> Aggregate using temporary table  (actual time=1847.017..1847.017 rows=201 loops
=1)
                -> Inner hash join (t.primaryTitle = u.Title)  (cost=347142.69 rows=26234) (ac
tual time=815.839..1844.775 rows=1442 loops=1)
                    -> Filter: ((t.runtimeMinutes < 70) and (t.runtimeMinutes > 19) and (t.tit
leType <> 'movie'))  (cost=39.78 rows=1233) (actual time=43.296..1668.418 rows=50497 loops=1)
                        -> Table scan on t  (cost=39.78 rows=123331) (actual time=43.280..1650
.836 rows=110217 loops=1)
                    -> Hash
                        -> Table scan on u  (cost=222.98 rows=2128) (actual time=59.343..117.3
77 rows=2128 loops=1)
 |
```

Cost baseline: 347142.69

Index 1:

create index primarytitle_idx on title(primaryTitle);

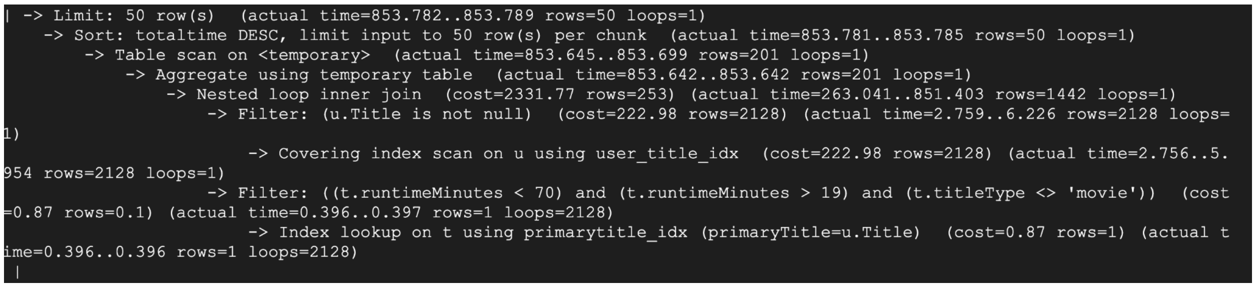


```
| -> Limit: 50 row(s)  (actual time=531.231..531.239 rows=50 loops=1)
    -> Sort: totaltime DESC, limit input to 50 row(s) per chunk  (actual time=531.230..531.234
 rows=50 loops=1)
        -> Table scan on <temporary>  (actual time=531.107..531.149 rows=201 loops=1)
            -> Aggregate using temporary table  (actual time=531.105..531.105 rows=201 loops=1
)
                -> Nested loop inner join  (cost=2290.32 rows=253) (actual time=0.122..529.099
 rows=1442 loops=1)
                    -> Filter: (u.Title is not null)  (cost=215.55 rows=2128) (actual time=0.0
64..1.136 rows=2128 loops=1)
                        -> Table scan on u  (cost=215.55 rows=2128) (actual time=0.063..0.903
rows=2128 loops=1)
                    -> Filter: ((t.runtimeMinutes < 70) and (t.runtimeMinutes > 19) and (t.tit
leType <> 'movie'))  (cost=0.86 rows=0.1) (actual time=0.247..0.248 rows=1 loops=2128)
                        -> Index lookup on t using primarytitle_idx (primaryTitle=u.Title)  (c
ost=0.86 rows=1) (actual time=0.247..0.247 rows=1 loops=2128)
 |
```

Cost after index 1:2290.32

Index 2:

create index user_title_idx on user_info(Title);

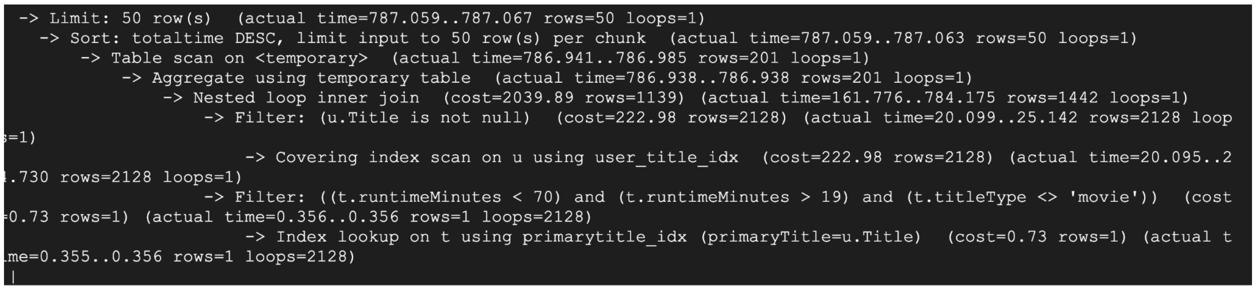


```
| -> Limit: 50 row(s)  (actual time=853.782..853.789 rows=50 loops=1)
    -> Sort: totaltime DESC, limit input to 50 row(s) per chunk  (actual time=853.781..853.785 rows=50 loops=1)
        -> Table scan on <temporary>  (actual time=853.645..853.699 rows=201 loops=1)
            -> Aggregate using temporary table  (actual time=853.642..853.642 rows=201 loops=1)
                -> Nested loop inner join  (cost=2331.77 rows=253) (actual time=263.041..851.403 rows=1442 loops=1)
                    -> Filter: (u.Title is not null)  (cost=222.98 rows=2128) (actual time=2.759..6.226 rows=2128 loops=
1)
                        -> Covering index scan on u using user_title_idx  (cost=222.98 rows=2128) (actual time=2.756..5.
954 rows=2128 loops=1)
                    -> Filter: ((t.runtimeMinutes < 70) and (t.runtimeMinutes > 19) and (t.titleType <> 'movie'))  (cost
=0.87 rows=0.1) (actual time=0.396..0.397 rows=1 loops=2128)
                        -> Index lookup on t using primarytitle_idx (primaryTitle=u.Title)  (cost=0.87 rows=1) (actual t
ime=0.396..0.396 rows=1 loops=2128)
 |
```

Cost after index 2: 2331.77

Index 3:

create index runtime_title_idx on user_info(Title);

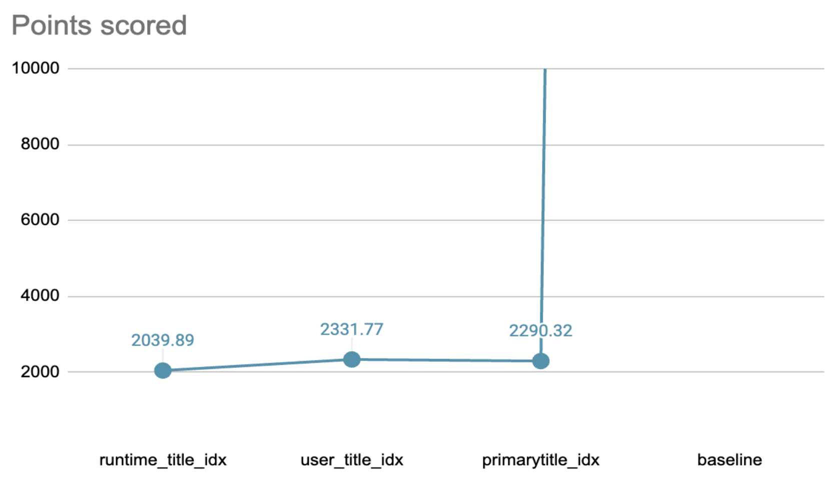


```
 -> Limit: 50 row(s)  (actual time=787.059..787.067 rows=50 loops=1)
    -> Sort: totaltime DESC, limit input to 50 row(s) per chunk  (actual time=787.059..787.063 rows=50 loops=1)
        -> Table scan on <temporary>  (actual time=786.941..786.985 rows=201 loops=1)
            -> Aggregate using temporary table  (actual time=786.938..786.938 rows=201 loops=1)
                -> Nested loop inner join  (cost=2039.89 rows=1139) (actual time=161.776..784.175 rows=1442 loops=1)
                    -> Filter: (u.Title is not null)  (cost=222.98 rows=2128) (actual time=20.099..25.142 rows=2128 loop
s=1)
                        -> Covering index scan on u using user_title_idx  (cost=222.98 rows=2128) (actual time=20.095..2
.730 rows=2128 loops=1)
                    -> Filter: ((t.runtimeMinutes < 70) and (t.runtimeMinutes > 19) and (t.titleType <> 'movie'))  (cost
=0.73 rows=1) (actual time=0.356..0.356 rows=1 loops=2128)
                        -> Index lookup on t using primarytitle_idx (primaryTitle=u.Title)  (cost=0.73 rows=1) (actual t
ime=0.355..0.356 rows=1 loops=2128)
 |
```

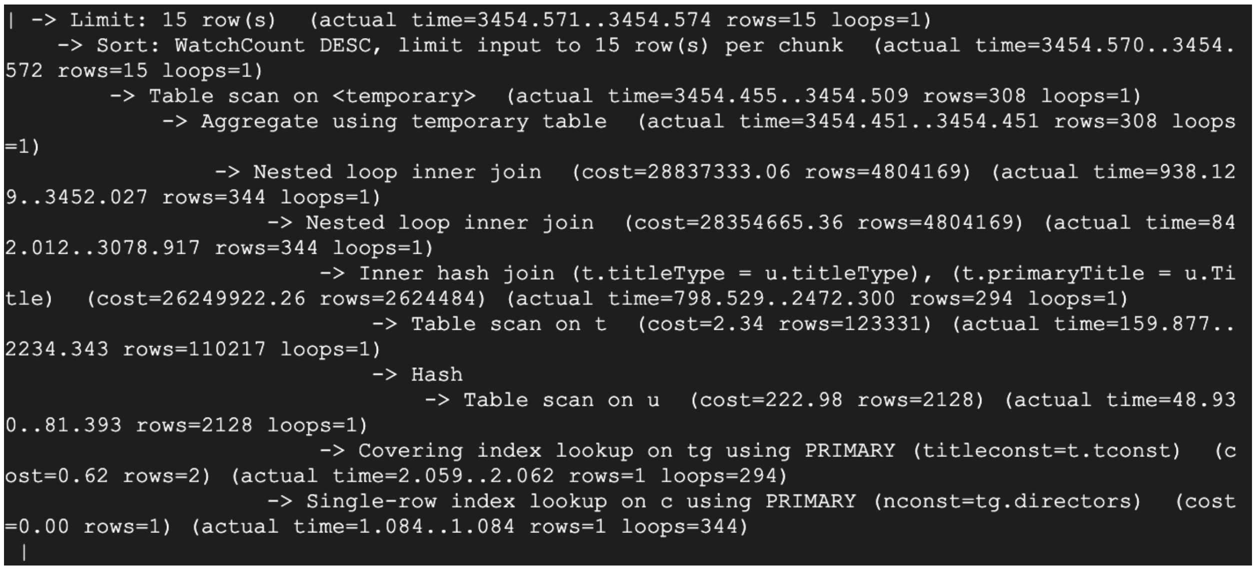
Cost after index 3: 2039.89

Justification:

Index 1 is on the primary title which is used to join the user_info table to title table. This as mentioned before had a huge impact on cost. This is most likely because the join was not on the primary key but in title names and hence each time the title was scanned. Indexing on the title name lowered the cost from 300k to around 3k. Index 2, the user title index is also a part of the join, and indexing on it lowered the cost by 200 points. Although runtimeMuinutes is not directly affecting any joins, it is being grouped to query values. The cost of the the query is reduced after indexing runtimeMinutes but the optimizer hasn't used the index to optimize the query. This might be because of many factors like cardinality and use of attributes in the query.

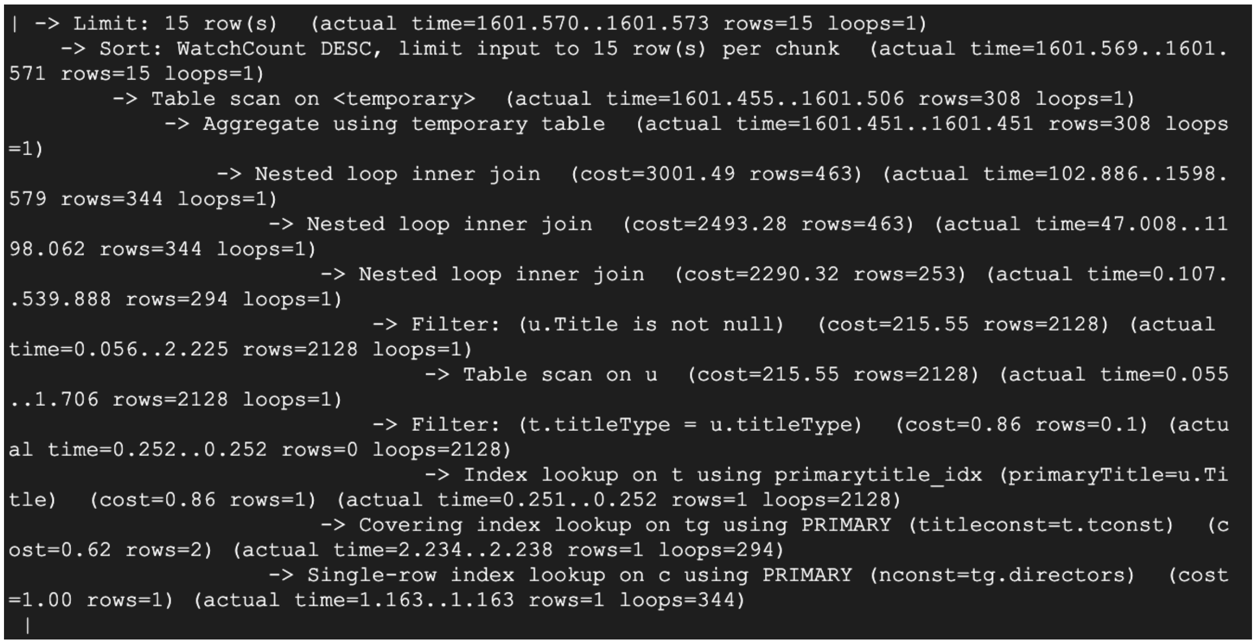


Query 4:
Director
Explain analyze on basic query


```
| -> Limit: 15 row(s)  (actual time=3454.571..3454.574 rows=15 loops=1)
    -> Sort: WatchCount DESC, limit input to 15 row(s) per chunk  (actual time=3454.570..3454.
572 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=3454.455..3454.509 rows=308 loops=1)
            -> Aggregate using temporary table  (actual time=3454.451..3454.451 rows=308 loops
=1)
                -> Nested loop inner join  (cost=28837333.06 rows=4804169) (actual time=938.12
9..3452.027 rows=344 loops=1)
                    -> Nested loop inner join  (cost=28354665.36 rows=4804169) (actual time=84
2.012..3078.917 rows=344 loops=1)
                        -> Inner hash join (t.titleType = u.titleType), (t.primaryTitle = u.Ti
tle)  (cost=26249922.26 rows=2624484) (actual time=798.529..2472.300 rows=294 loops=1)
                            -> Table scan on t  (cost=2.34 rows=123331) (actual time=159.877..
2234.343 rows=110217 loops=1)
                            -> Hash
                                -> Table scan on u  (cost=222.98 rows=2128) (actual time=48.93
0..81.393 rows=2128 loops=1)
                        -> Covering index lookup on tg using PRIMARY (titleconst=t.tconst)  (c
ost=0.62 rows=2) (actual time=2.059..2.062 rows=1 loops=294)
                    -> Single-row index lookup on c using PRIMARY (nconst=tg.directors)  (cost
=0.00 rows=1) (actual time=1.084..1.084 rows=1 loops=344)
 |
```

Cost baseline: 28837333.06

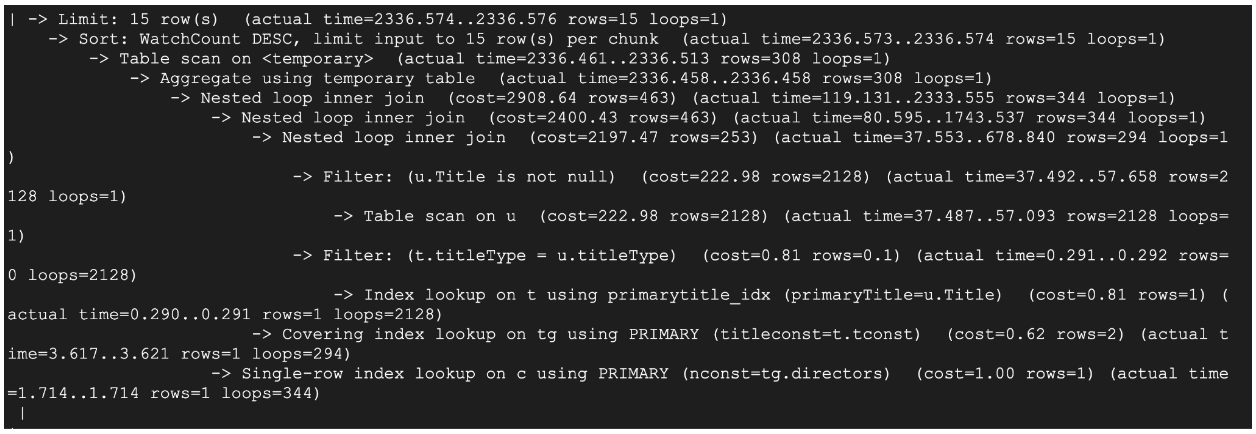
Index 1:
create index primarytitle_idx on title(primaryTitle);


```
| -> Limit: 15 row(s)  (actual time=1601.570..1601.573 rows=15 loops=1)
    -> Sort: WatchCount DESC, limit input to 15 row(s) per chunk  (actual time=1601.569..1601.
571 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=1601.455..1601.506 rows=308 loops=1)
            -> Aggregate using temporary table  (actual time=1601.451..1601.451 rows=308 loops
=1)
                -> Nested loop inner join  (cost=3001.49 rows=463) (actual time=102.886..1598.
579 rows=344 loops=1)
                    -> Nested loop inner join  (cost=2493.28 rows=463) (actual time=47.008..11
98.062 rows=344 loops=1)
                        -> Nested loop inner join  (cost=2290.32 rows=253) (actual time=0.107.
.539.888 rows=294 loops=1)
                            -> Filter: (u.Title is not null)  (cost=215.55 rows=2128) (actual
time=0.056..2.225 rows=2128 loops=1)
                                -> Table scan on u  (cost=215.55 rows=2128) (actual time=0.055
..1.706 rows=2128 loops=1)
                            -> Filter: (t.titleType = u.titleType)  (cost=0.86 rows=0.1) (actu
al time=0.252..0.252 rows=0 loops=2128)
                                -> Index lookup on t using primarytitle_idx (primaryTitle=u.Ti
tle)  (cost=0.86 rows=1) (actual time=0.251..0.252 rows=1 loops=2128)
                        -> Covering index lookup on tg using PRIMARY (titleconst=t.tconst)  (c
ost=0.62 rows=2) (actual time=2.234..2.238 rows=1 loops=294)
                    -> Single-row index lookup on c using PRIMARY (nconst=tg.directors)  (cost
=1.00 rows=1) (actual time=1.163..1.163 rows=1 loops=344)
 |
```

Cost after index 1: 3001

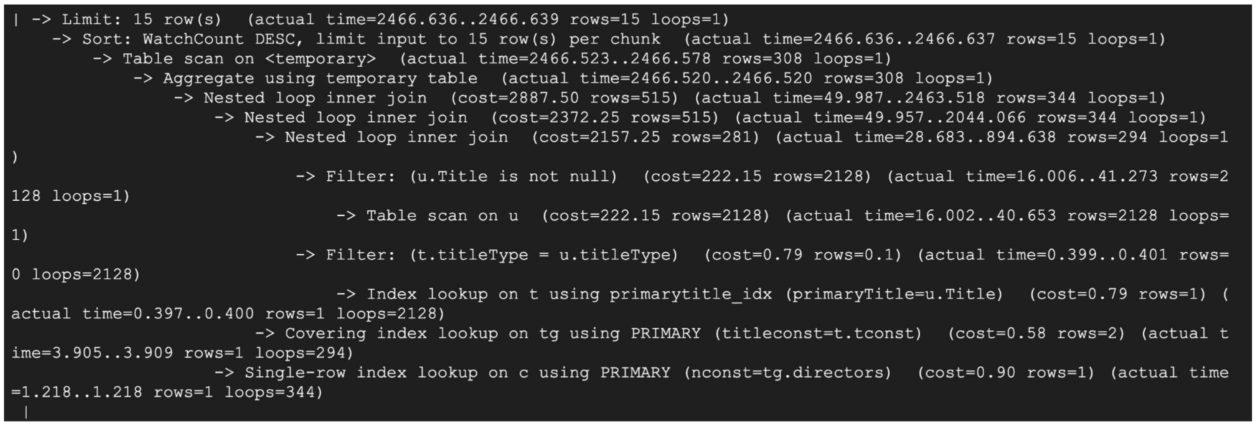
Index 2:

create index user_title on user_info(Title);
Cost after index 2:2908



```
| -> Limit: 15 row(s)  (actual time=2336.574..2336.576 rows=15 loops=1)
    -> Sort: WatchCount DESC, limit input to 15 row(s) per chunk  (actual time=2336.573..2336.574 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=2336.461..2336.513 rows=308 loops=1)
            -> Aggregate using temporary table  (actual time=2336.458..2336.458 rows=308 loops=1)
                -> Nested loop inner join  (cost=2908.64 rows=463) (actual time=119.131..2333.555 rows=344 loops=1)
                    -> Nested loop inner join  (cost=2400.43 rows=463) (actual time=80.595..1743.537 rows=344 loops=1)
                        -> Nested loop inner join  (cost=2197.47 rows=253) (actual time=37.553..678.840 rows=294 loops=1
)
                            -> Filter: (u.Title is not null)  (cost=222.98 rows=2128) (actual time=37.492..57.658 rows=2
128 loops=1)
                                -> Table scan on u  (cost=222.98 rows=2128) (actual time=37.487..57.093 rows=2128 loops=
1)
                            -> Filter: (t.titleType = u.titleType)  (cost=0.81 rows=0.1) (actual time=0.291..0.292 rows=
0 loops=2128)
                                -> Index lookup on t using primarytitle_idx (primaryTitle=u.Title)  (cost=0.81 rows=1) (
actual time=0.290..0.291 rows=1 loops=2128)
                        -> Covering index lookup on tg using PRIMARY (titleconst=t.tconst)  (cost=0.62 rows=2) (actual t
ime=3.617..3.621 rows=1 loops=294)
                    -> Single-row index lookup on c using PRIMARY (nconst=tg.directors)  (cost=1.00 rows=1) (actual time
=1.714..1.714 rows=1 loops=344)
 |
```

Index 3:
create index title_type on title(titleType);



```
| -> Limit: 15 row(s)  (actual time=2466.636..2466.639 rows=15 loops=1)
    -> Sort: WatchCount DESC, limit input to 15 row(s) per chunk  (actual time=2466.636..2466.637 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=2466.523..2466.578 rows=308 loops=1)
            -> Aggregate using temporary table  (actual time=2466.520..2466.520 rows=308 loops=1)
                -> Nested loop inner join  (cost=2887.50 rows=515) (actual time=49.987..2463.518 rows=344 loops=1)
                    -> Nested loop inner join  (cost=2372.25 rows=515) (actual time=49.957..2044.066 rows=344 loops=1)
                        -> Nested loop inner join  (cost=2157.25 rows=281) (actual time=28.683..894.638 rows=294 loops=1
)
                            -> Filter: (u.Title is not null)  (cost=222.15 rows=2128) (actual time=16.006..41.273 rows=2
128 loops=1)
                                -> Table scan on u  (cost=222.15 rows=2128) (actual time=16.002..40.653 rows=2128 loops=
1)
                            -> Filter: (t.titleType = u.titleType)  (cost=0.79 rows=0.1) (actual time=0.399..0.401 rows=
0 loops=2128)
                                -> Index lookup on t using primarytitle_idx (primaryTitle=u.Title)  (cost=0.79 rows=1) (
actual time=0.397..0.400 rows=1 loops=2128)
                        -> Covering index lookup on tg using PRIMARY (titleconst=t.tconst)  (cost=0.58 rows=2) (actual t
ime=3.905..3.909 rows=1 loops=294)
                    -> Single-row index lookup on c using PRIMARY (nconst=tg.directors)  (cost=0.90 rows=1) (actual time
=1.218..1.218 rows=1 loops=344)
 |
```

Cost after index 3: 2887.50

Justification:
Indexing on primaryTitle lowers the cost significantly from around 28 million to around 3000 because, as mentioned before, the join is on the title names and not the primary key. After index 2 the cost does not decrease by a significant amount because the optimizer did not use the index. The attribute is used in a join but due to null values present, the cost did not change. After index 3, the cost also stays relatively the same since the optimizer did not use the index. The attribute is used in a join but likely due to low cardinality, the index might be rejected by the optimizer.

The final configuration of indexes that we will be using are
1) Index on the primary title in the titles column. This index is essential in bringing down the cost of joining the user_info table to the titles table, which is common in every query.
2) Index on user_info titleType. Although this index does not lower the cost for all joins between user_info and title. It helps in lowering the cost for query 3 without increasing the cost for any other query.

3) Genre on genres table. This index is on the names of the genres which especially helps the genre table join with a lower cost than the title table