DBMS - II

Assignment -3

TEAM NO. 17

Report :

Writing various queries For the Relational Tables created in the assignment 2 as per the ER diagram designed in assignment 1

Team Members:

Raviteja Namani - CS18BTECH11032
Geethika Sowmya - ES18BTECH11025
Katravath Rajesh - CS18BTECH11023
Mathangi Jedidiah - CS18BTECH11028

To obtain desired queries, functions used to perform calculations,:

`AVG`
`COUNT`
`DISTINCT`, for getting distinct values without duplicates
`MAX`
`MIN`

There following operations are used in combinations and standalone as needed:

`WHERE`, matching on some strict condition
`LIKE`, matching on substrings for text
`LIMIT`
`GROUP BY`
`ORDER BY`
`INNER JOIN` selects all rows from both tables as long as there is a match between the columns

`JOIN,` combining data from multiple tables

`RIGHT JOIN`, returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join predicate.

The commands mainly used to perform each of the queries are:

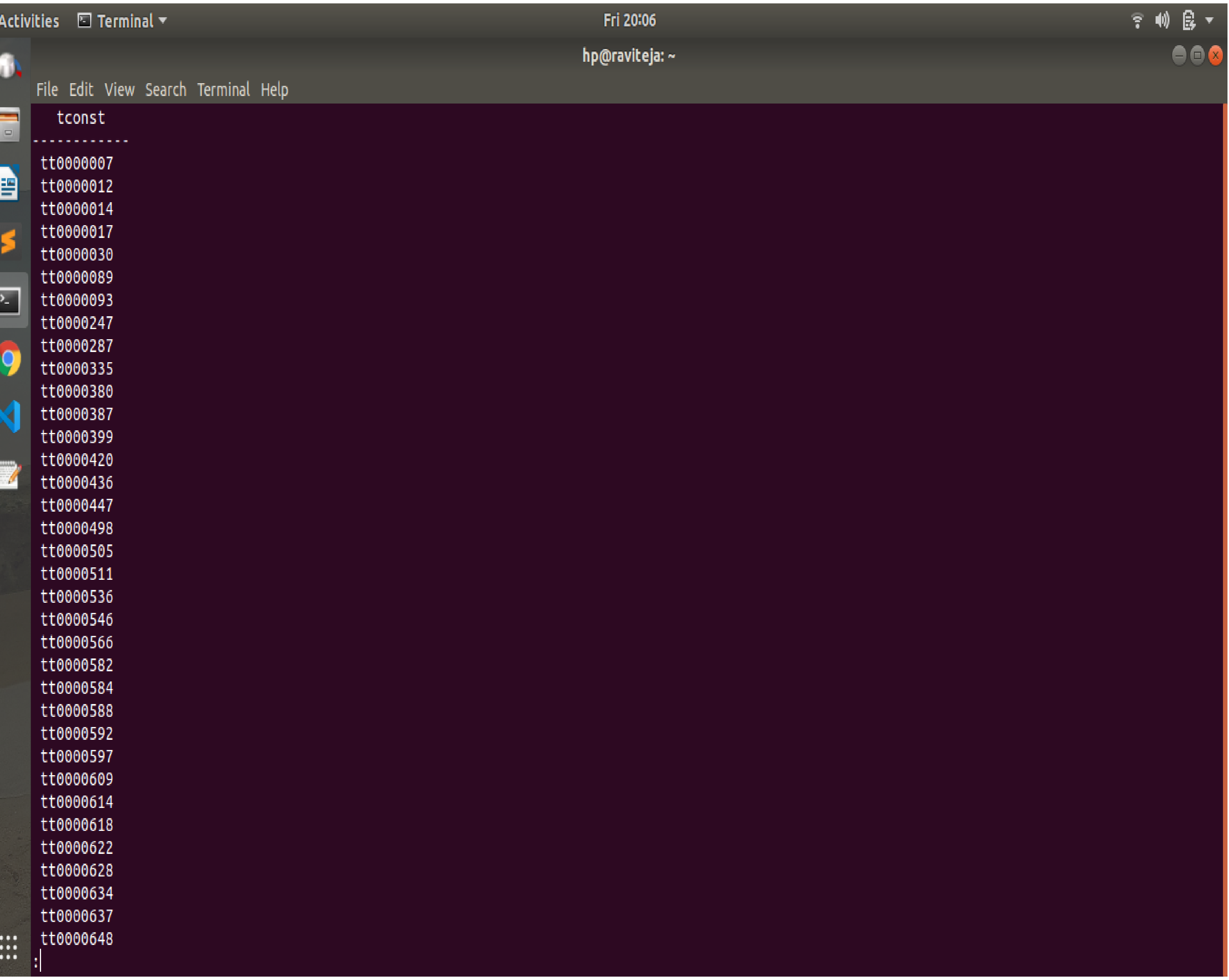
`SELECT:` Along with clauses like
`FROM-` to specify which table to select

`WITH-`to define “statement scoped views”. They are not stored in the database instead, they are only valid in the query they belong to.Makes it possible to improve the structure of a statement without fabricating the global space.

1. Write a query to find the list of movies that are directed by at least 2 directors.

Group the table by titleid and and select those titles having count>=2 and has titletype as movie.

```
select tconst from directors d1 where (select titletype from
originaltitles o1 where o1.tconst=d1.tconst)='movie' group by tconst
having count(tconst)>=2;
```



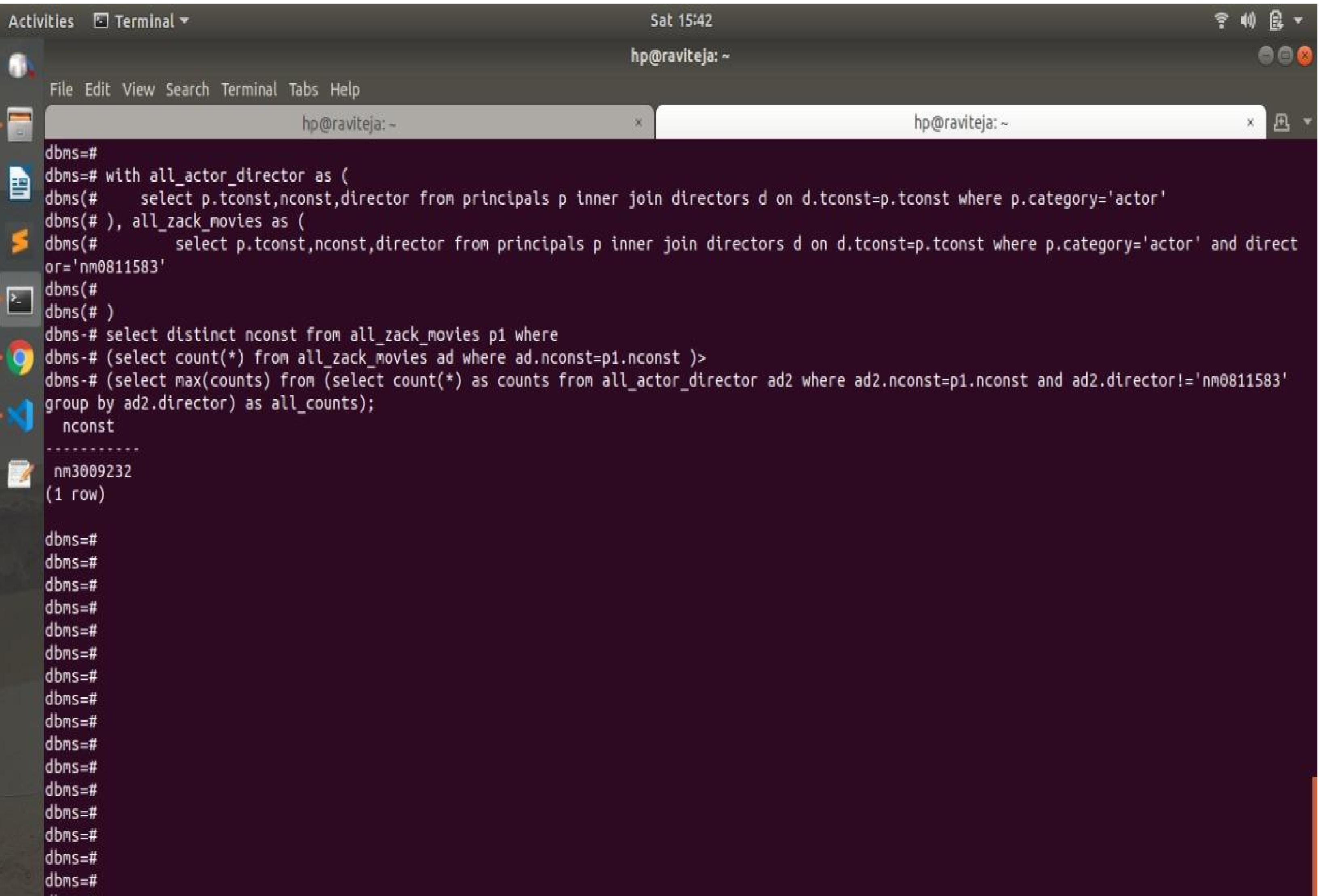
```
   tconst
-----------
 tt0000007
 tt0000012
 tt0000014
 tt0000017
 tt0000030
 tt0000089
 tt0000093
 tt0000247
 tt0000287
 tt0000335
 tt0000380
 tt0000387
 tt0000399
 tt0000420
 tt0000436
 tt0000447
 tt0000498
 tt0000505
 tt0000511
 tt0000536
 tt0000546
 tt0000566
 tt0000582
 tt0000584
 tt0000588
 tt0000592
 tt0000597
 tt0000609
 tt0000614
 tt0000618
 tt0000622
 tt0000628
 tt0000634
 tt0000637
 tt0000648
:
```

Here, tconst is the IDs of the required movies.

2. Find all the actors that made more movies with Zack Snyder than any other director.

The ID of Zack Snyder is "nm0811583". all_actor_director consists of all the pairs of actor-directors. From them, do the needful.
select those distinct actors who have more actor-director pairs with zack snyder.

```
with all_actor_director as (
    select p.tconst,nconst,director from principals p inner join
directors d on d.tconst=p.tconst where p.category='actor'
), all_zack_movies as (
        select p.tconst,nconst,director from principals p inner join
directors d on d.tconst=p.tconst where p.category='actor' and
director='nm0811583'

)
select distinct nconst from all_zack_movies p1 where
(select count(*) from all_zack_movies ad where ad.nconst=p1.nconst )>
(select max(counts) from (select count(*) as counts from
all_actor_director ad2 where ad2.nconst=p1.nconst and
ad2.director!='nm0811583' group by ad2.director) as all_counts);
```



```
dbms=#
dbms=# with all_actor_director as (
dbms(#     select p.tconst,nconst,director from principals p inner join directors d on d.tconst=p.tconst where p.category='actor'
dbms(# ), all_zack_movies as (
dbms(#         select p.tconst,nconst,director from principals p inner join directors d on d.tconst=p.tconst where p.category='actor' and direct
or='nm0811583'
dbms(#
dbms(# )
dbms-# select distinct nconst from all_zack_movies p1 where
dbms-# (select count(*) from all_zack_movies ad where ad.nconst=p1.nconst )>
dbms-# (select max(counts) from (select count(*) as counts from all_actor_director ad2 where ad2.nconst=p1.nconst and ad2.director!='nm0811583'
group by ad2.director) as all_counts);
  nconst
-----------
 nm3009232
(1 row)

dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
```

Here, nconst is the IDs of the required actors.

3. Find the movie that has won fewer than 2 awards.

The table has not been created. So the following are assumed. The table name is awards which is assumed to have variable tconst referencing to originaltitles.tconst.
Group the table awards by titleid and select only those who have count<2

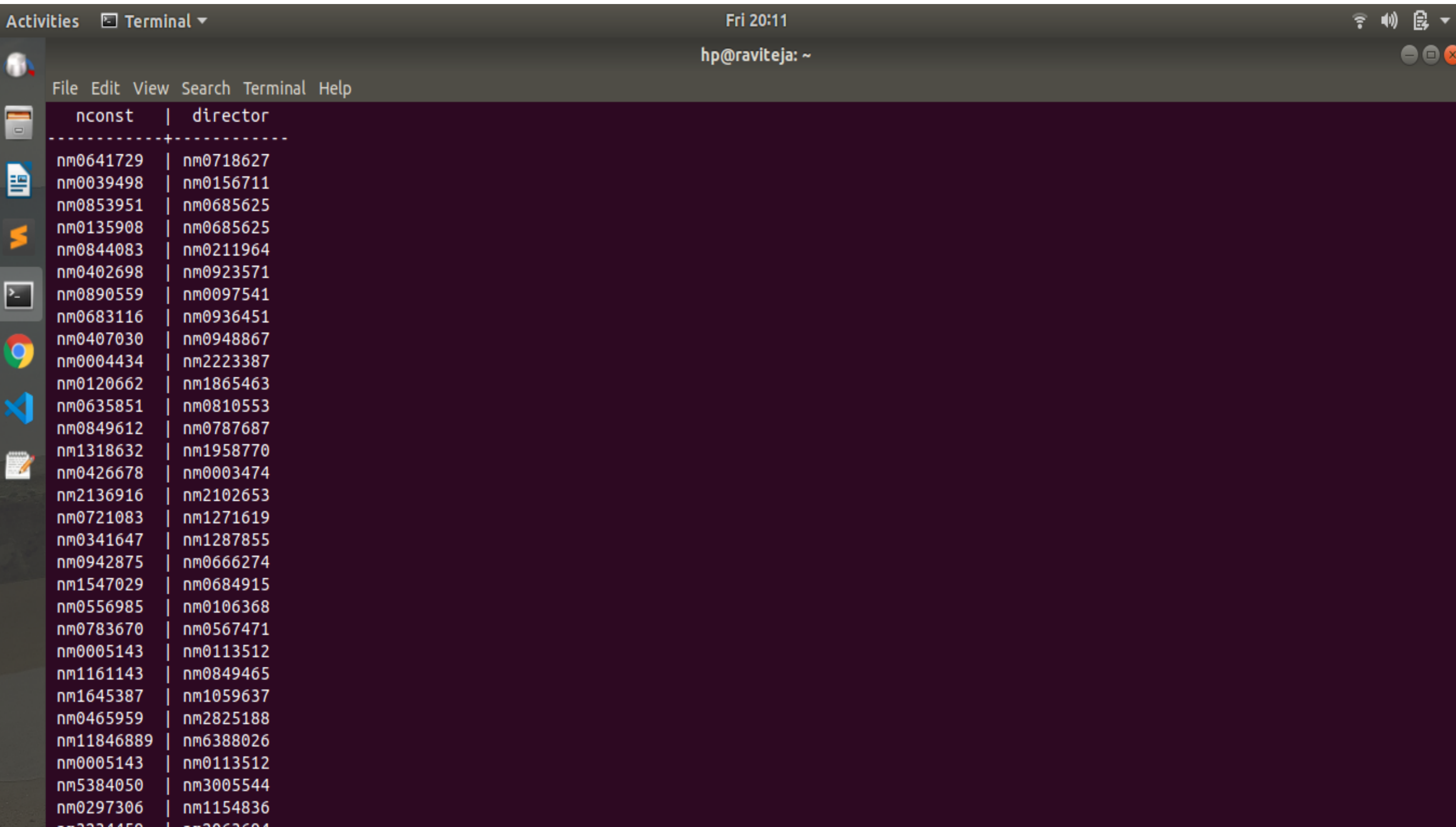
```
select tconst from awards group by tconst having count(tconst)<2;
```

Here, tconst is the IDs of the required movies.

4. Find the pair of actor and movie director, provided that the movie done by them has a rating above 7 and movies done by the pair should be at most 2.

A straight-forward application of inner join. Join the tables principals and directors and select all actor-director pairs with given conditions

```
select foo.nconst,foo.director from
(select p2.tconst,p2.nconst,d2.director from principals p2 inner join directors
d2 on d2.tconst=p2.tconst where
 (p2.nconst,d2.director) in (select nconst,director from principals p1 inner
join directors d1 on d1.tconst=p1.tconst where category='actor' group by
nconst,director having count(p1.tconst)<=2)) as foo inner join originaltitles
 inner join originaltitles o1 on o1.tconst=foo.tconst where o1.rating>7;
```



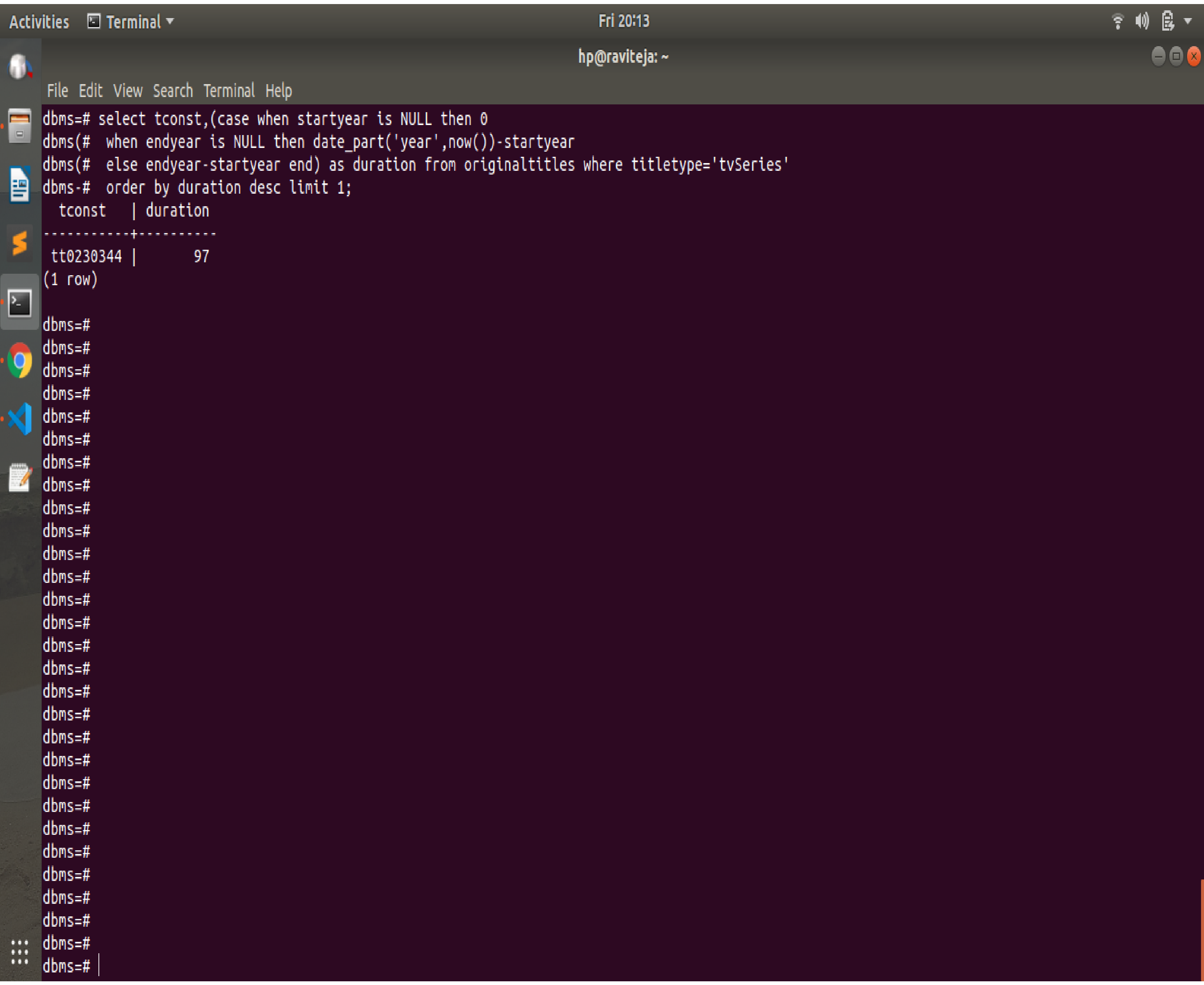
```
  nconst    |  director
------------+------------
 nm0641729  | nm0718627
 nm0039498  | nm0156711
 nm0853951  | nm0685625
 nm0135908  | nm0685625
 nm0844083  | nm0211964
 nm0402698  | nm0923571
 nm0890559  | nm0097541
 nm0683116  | nm0936451
 nm0407030  | nm0948867
 nm0004434  | nm2223387
 nm0120662  | nm1865463
 nm0635851  | nm0810553
 nm0849612  | nm0787687
 nm1318632  | nm1958770
 nm0426678  | nm0003474
 nm2136916  | nm2102653
 nm0721083  | nm1271619
 nm0341647  | nm1287855
 nm0942875  | nm0666274
 nm1547029  | nm0684915
 nm0556985  | nm0106368
 nm0783670  | nm0567471
 nm0005143  | nm0113512
 nm1161143  | nm0849465
 nm1645387  | nm1059637
 nm0465959  | nm2825188
 nm11846889 | nm6388026
 nm0005143  | nm0113512
 nm5384050  | nm3005544
 nm0297306  | nm1154836
```

Here, nconst-director is the IDs of the required actor-director pair.

5.Find the name of the TV series which aired for the longest duration.

To ignore the corrupted/unspecified data, the duration considered is 0. If endyear is null, it is considered that the tvSeries is still running.
Sort the table originaltites according to their duration and limit to 1 for 1st record.

```
select tconst,(case when startyear is NULL then 0
 when endyear is NULL then date_part('year',now())-startyear
 else endyear-startyear end) as duration from originaltitles where
titletype='tvSeries'
 order by duration desc limit 1;
```



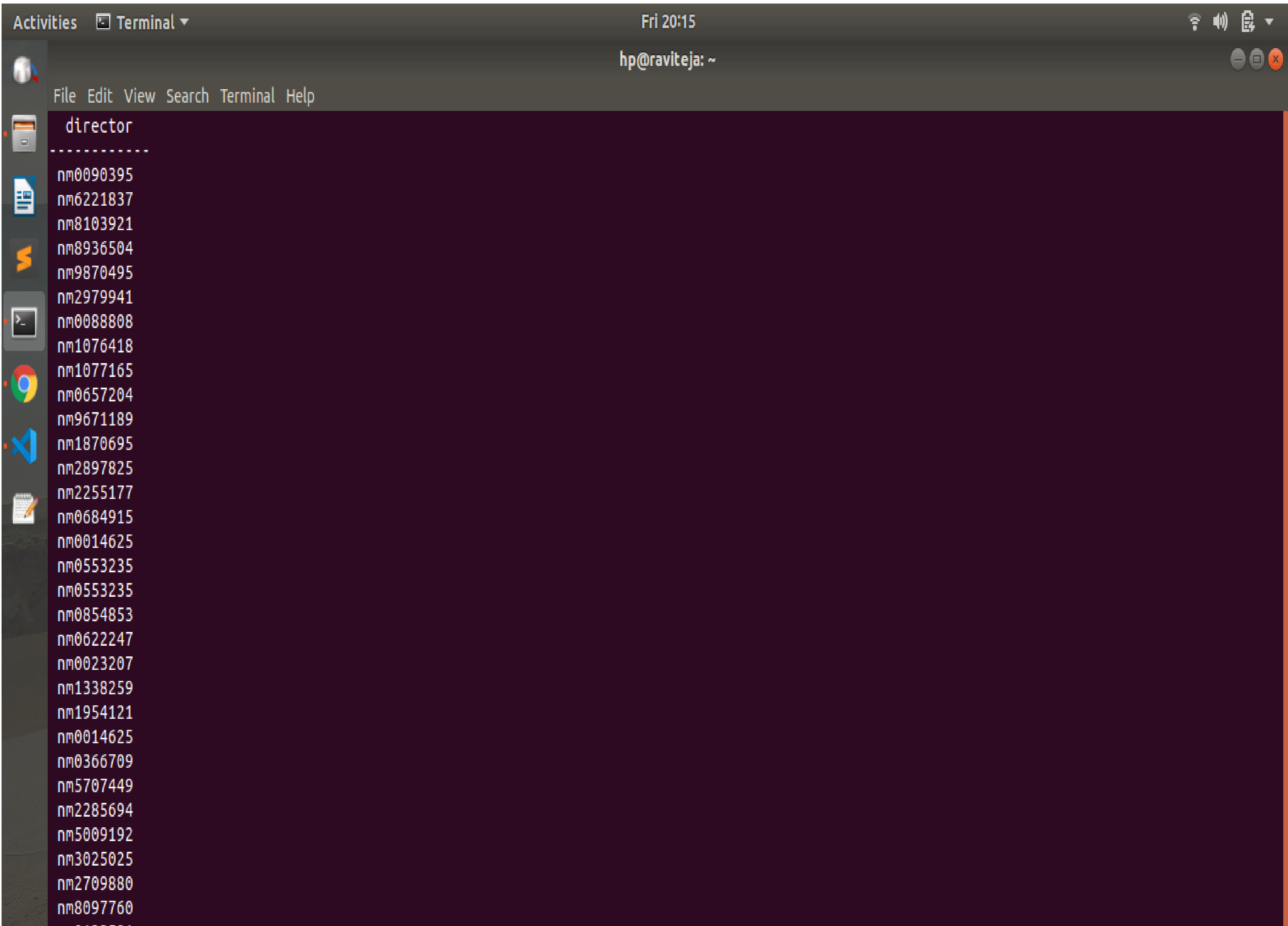
```
dbms=# select tconst,(case when startyear is NULL then 0
dbms(#  when endyear is NULL then date_part('year',now())-startyear
dbms(#  else endyear-startyear end) as duration from originaltitles where titletype='tvSeries'
dbms-#  order by duration desc limit 1;
  tconst   | duration
-----------+----------
 tt0230344 |       97
(1 row)

dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
```

Here, tconst is the ID of the movie and duration of the longest runtime.

6.Find the name of the director who directed the 2nd shortest movie in the year 2020.

Firstly, find the runtime of the second shortest movie. For this, sort by runtime, set offset 1 and limit to 1.
Again, the same assumptions are done to avoid unspecified/corrupted data entries.
Then, search the director titleid from the directors table.

```
select director from directors where tconst in (
select tconst from
originaltitles where runtime=(
    select distinct (case when runtime is NULL then 999999 else
runtime END) from originaltitles where titletype='movie' and
startyear='2020' order by runtime offset 1 limit 1
));
```



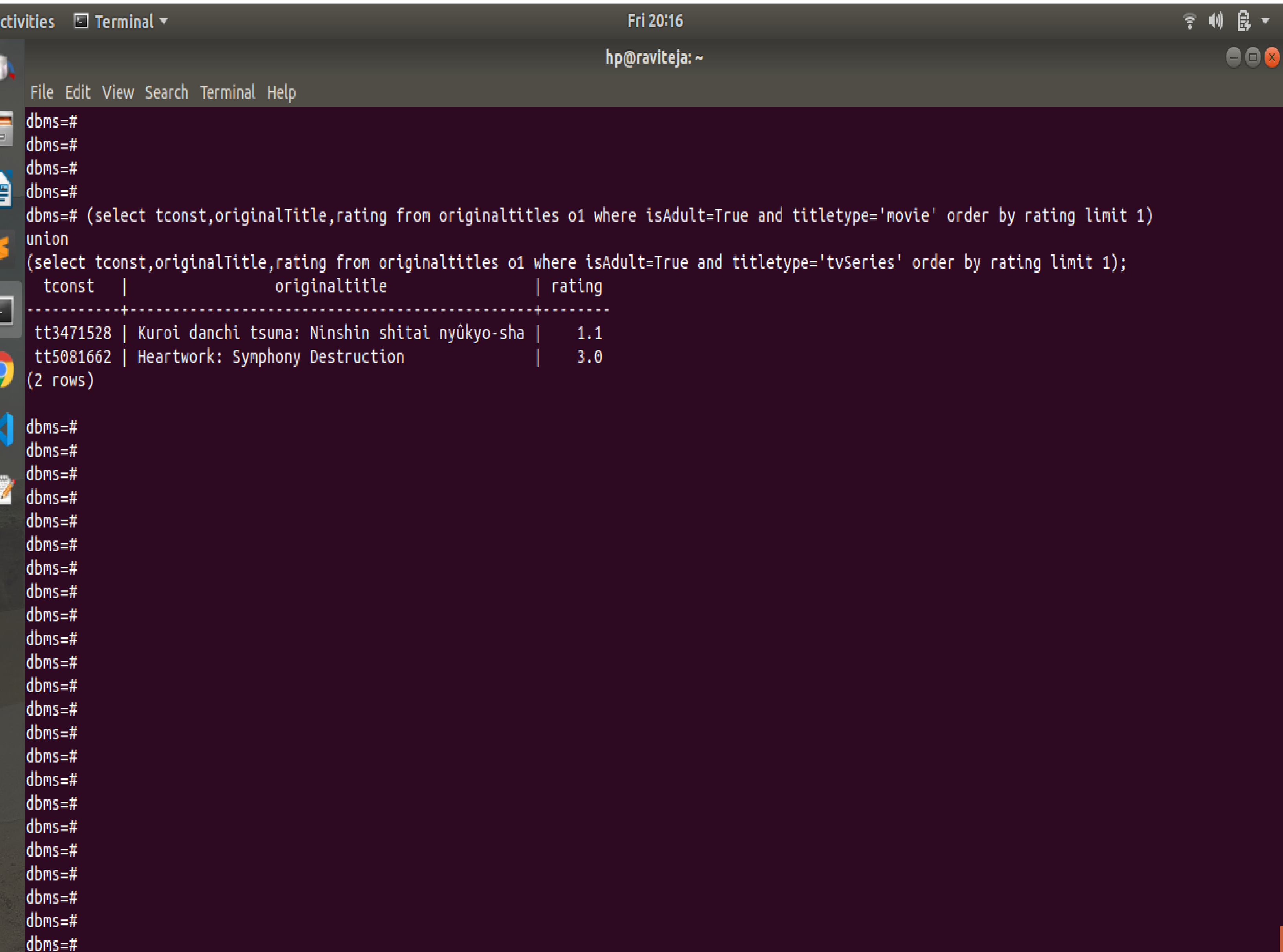
```
 director
-----------
 nm0090395
 nm6221837
 nm8103921
 nm8936504
 nm9870495
 nm2979941
 nm0088808
 nm1076418
 nm1077165
 nm0657204
 nm9671189
 nm1870695
 nm2897825
 nm2255177
 nm0684915
 nm0014625
 nm0553235
 nm0553235
 nm0854853
 nm0622247
 nm0023207
 nm1338259
 nm1954121
 nm0014625
 nm0366709
 nm5707449
 nm2285694
 nm5009192
 nm3025025
 nm2709880
 nm8097760
```

Here, the director column has the IDs of directors for the asked query.

7.Print the adult movie and adult TV series with the lowest average rating.

This is done using ORDER BY, sort by rating and limit the result to 1 to get the lowest
Selecting tconst, the originalTitle and rating from originaltitles table where isAdult value is true and the titletype is a movie.
Similarly for the series where titletype is tvseries. UNION the both for the result

```
(select tconst,originalTitle,rating from originaltitles o1 where
isAdult=True and titletype='movie' order by rating limit 1)
union
(select tconst,originalTitle,rating from originaltitles o1 where
isAdult=True and titletype='tvSeries' order by rating limit 1);
```



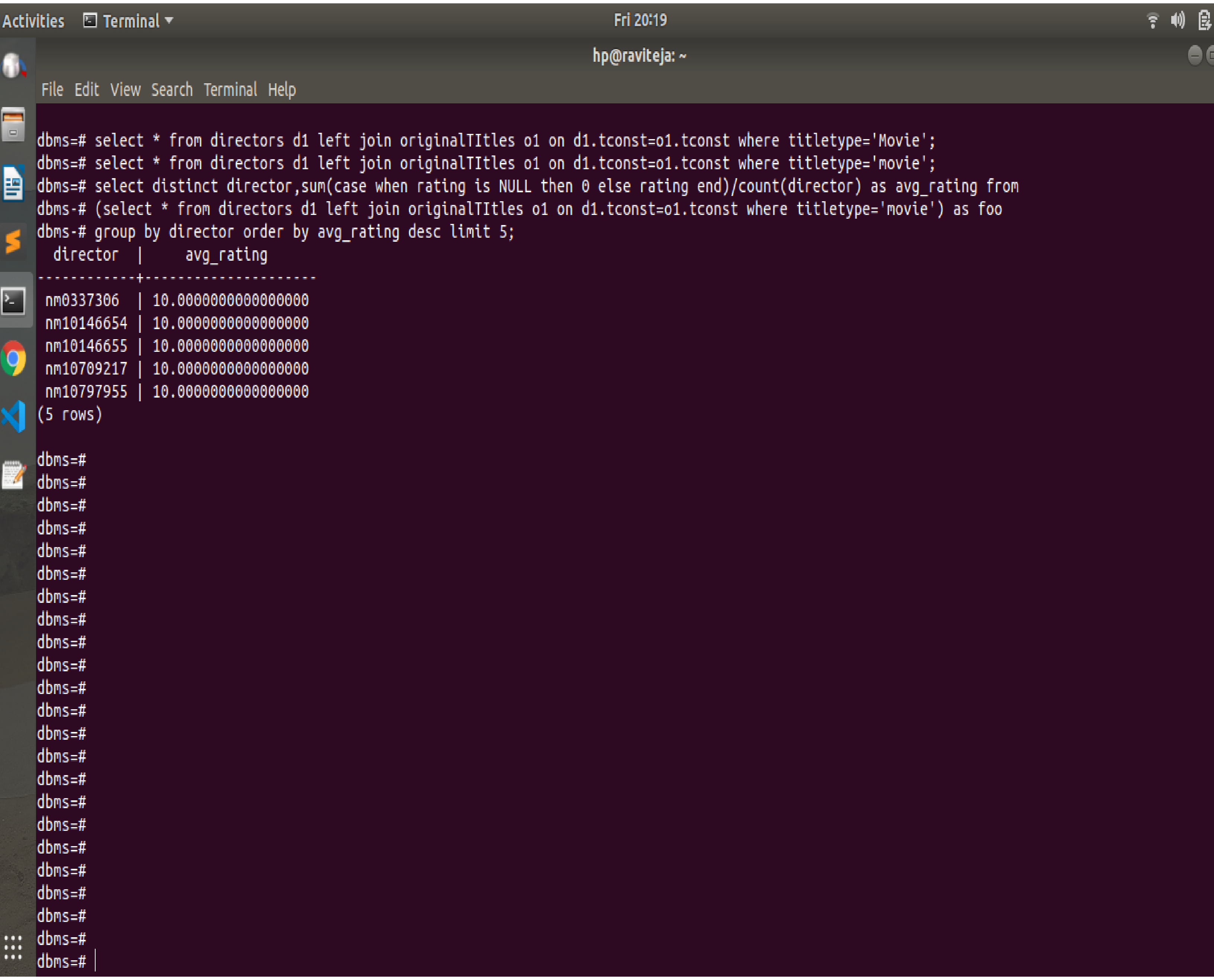
```
dbms=#
dbms=#
dbms=#
dbms=#
dbms=# (select tconst,originalTitle,rating from originaltitles o1 where isAdult=True and titletype='movie' order by rating limit 1)
union
(select tconst,originalTitle,rating from originaltitles o1 where isAdult=True and titletype='tvSeries' order by rating limit 1);
  tconst   |                originaltitle                | rating
-----------+---------------------------------------------+--------
 tt3471528 | Kuroi danchi tsuma: Ninshin shitai nyûkyo-sha |    1.1
 tt5081662 | Heartwork: Symphony Destruction             |    3.0
(2 rows)

dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
```

8.Print the Top 5 directors based on their average rating of all the movies he/she has directed. (In case of equal print all.)

Firstly, LEFT JOIN directors with titles and select only movies with alias as foo. Now, select all the disitnct directors and group them by sorting according to average rating.Limit the results to 5 in descending order..

```
select distinct director,sum(case when rating is NULL then 0 else
rating end)/count(director) as avg_rating from
(select * from directors d1 left join originalTItles o1 on
d1.tconst=o1.tconst where titletype='movie') as foo
group by director order by avg_rating desc limit 5;
```

Here, the director column has IDs of required directors.



```
dbms=# select * from directors d1 left join originalTItles o1 on d1.tconst=o1.tconst where titletype='Movie';
dbms=# select * from directors d1 left join originalTItles o1 on d1.tconst=o1.tconst where titletype='movie';
dbms=# select distinct director,sum(case when rating is NULL then 0 else rating end)/count(director) as avg_rating from
dbms-# (select * from directors d1 left join originalTItles o1 on d1.tconst=o1.tconst where titletype='movie') as foo
dbms-# group by director order by avg_rating desc limit 5;
  director  |     avg_rating
------------+--------------------
 nm0337306  | 10.0000000000000000
 nm10146654 | 10.0000000000000000
 nm10146655 | 10.0000000000000000
 nm10709217 | 10.0000000000000000
 nm10797955 | 10.0000000000000000
(5 rows)

dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
dbms=#
```

9.Print TV series produced by 2 or more production companies and same has been released in at least 3 different countries.

Production companies and released in countries information is not given therefore corresponding tables do not exist in our database and hence assumed in the following way.

production_companies table and locations table have tconst referring to originaltitles.tconst. Firstly, select those originaltitles.tconst for tvSeries then check for the corresponding tconst in both the tables where for an entry there are more than two production companies and more than 3 countries

```
select tconst from originaltitles o1 where titletype="tvSeries"
and (select count(*) from production_companies p1 where
p1.tconst=o1.tconst)>=2 and
(select count(*) from locations l1 where l1.tconst=o1.tconst )>=3;
```

tconst is title IDs of the required titles.

10.Print the name of the actors in the decreasing order of the year of their Oscar wins

The Awards table has not been created. So the following table with fields \is assumed. awardName,issuedYear,nconst which refers to cast_and_Crew.nconst, isWon boolean. Order by issuedYear and select oscars which have been won.

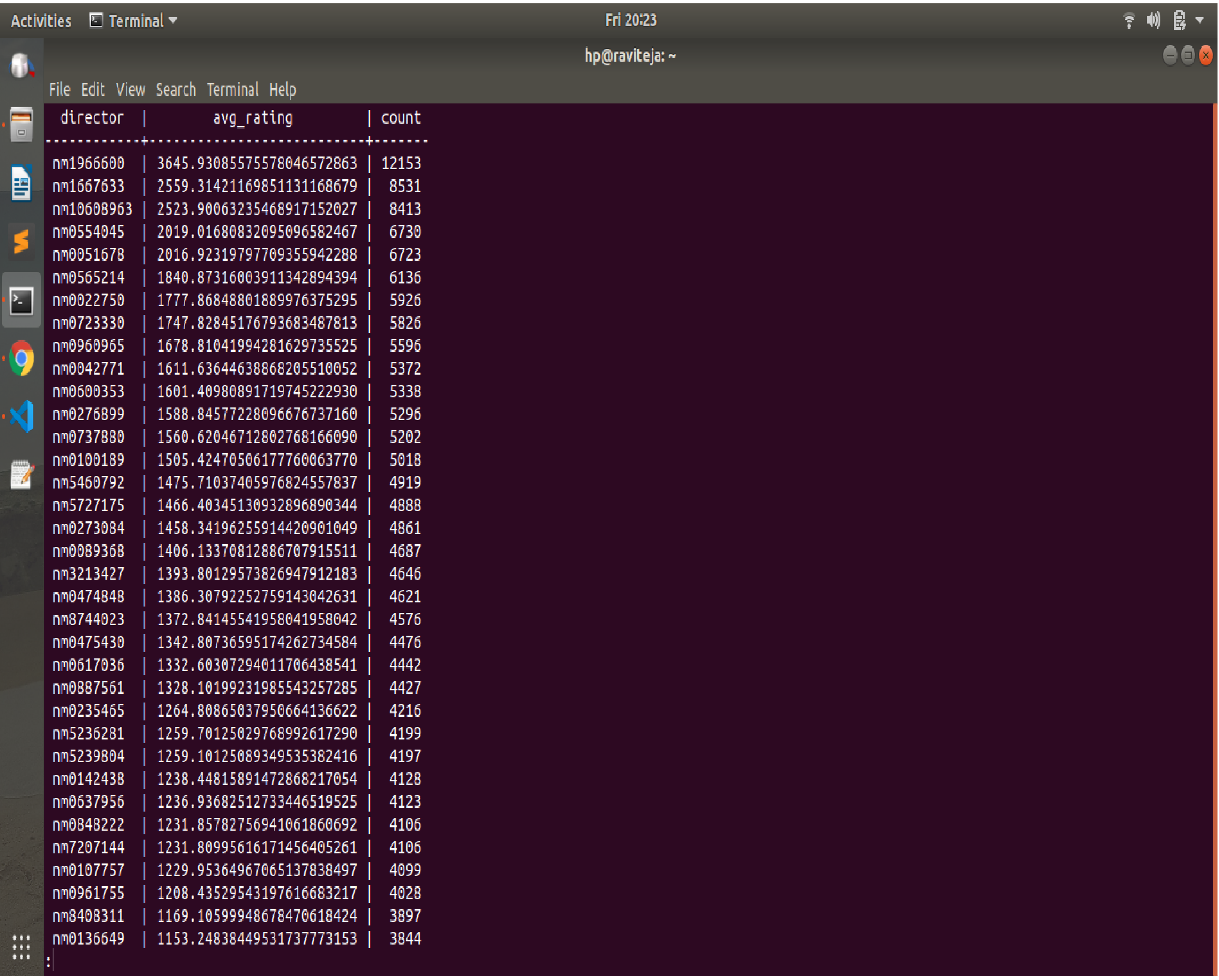
```
select nconst,issuedYear from awards where awardName='Oscars' and
isWon='True' order by issuedYear desc;
```

nconst is the IDs of the required actors.

11.List the directors in descending order of their score based on their experience and average movie ratings. Note the score is defined as Score = 0.3*experience + 0.7*average movie ratings where,
● experience = number of movies for which he/she worked as director or assistant director.
● Average movie ratings = 0.8*average movie ratings(worked as director) + 0.2* average
movie ratings(worked as asst-director).

LEFT JOIN originaltitles with directors and Group the rows by director. Then order by the average rating which is calculated from the expression given

```
select director,((sum(case when rating is NULL then 0 else rating
end)*0.7)/count(director))+count(director)*0.3 as
avg_rating,count(director)  from
(select * from directors d1 left join originalTItles o1 on
d1.tconst=o1.tconst) as foo
group by director order by avg_rating desc ;
```



```
  director  |        avg_rating         | count
------------+---------------------------+-------
 nm1966600  | 3645.93085575578046572863 | 12153
 nm1667633  | 2559.31421169851131168679 |  8531
 nm10608963 | 2523.90063235468917152027 |  8413
 nm0554045  | 2019.01680832095096582467 |  6730
 nm0051678  | 2016.92319797709355942288 |  6723
 nm0565214  | 1840.87316003911342894394 |  6136
 nm0022750  | 1777.86848801889976375295 |  5926
 nm0723330  | 1747.82845176793683487813 |  5826
 nm0960965  | 1678.81041994281629735525 |  5596
 nm0042771  | 1611.63644638868205510052 |  5372
 nm0600353  | 1601.40980891719745222930 |  5338
 nm0276899  | 1588.84577228096676737160 |  5296
 nm0737880  | 1560.62046712802768166090 |  5202
 nm0100189  | 1505.42470506177760063770 |  5018
 nm5460792  | 1475.71037405976824557837 |  4919
 nm5727175  | 1466.40345130932896890344 |  4888
 nm0273084  | 1458.34196255914420901049 |  4861
 nm0089368  | 1406.13370812886707915511 |  4687
 nm3213427  | 1393.80129573826947912183 |  4646
 nm0474848  | 1386.30792252759143042631 |  4621
 nm8744023  | 1372.84145541958041958042 |  4576
 nm0475430  | 1342.80736595174262734584 |  4476
 nm0617036  | 1332.60307294011706438541 |  4442
 nm0887561  | 1328.10199231985543257285 |  4427
 nm0235465  | 1264.80865037950664136622 |  4216
 nm5236281  | 1259.70125029768992617290 |  4199
 nm5239804  | 1259.10125089349535382416 |  4197
 nm0142438  | 1238.44815891472868217054 |  4128
 nm0637956  | 1236.93682512733446519525 |  4123
 nm0848222  | 1231.85782756941061860692 |  4106
 nm7207144  | 1231.80995616171456405261 |  4106
 nm0107757  | 1229.95364967065137838497 |  4099
 nm0961755  | 1208.43529543197616683217 |  4028
 nm8408311  | 1169.10599948678470618424 |  3897
 nm0136649  | 1153.24838449531737773153 |  3844
:
```

12.For each genre, print the top 5 movie names and its director name based on their earnings(box office collection - movie budget).

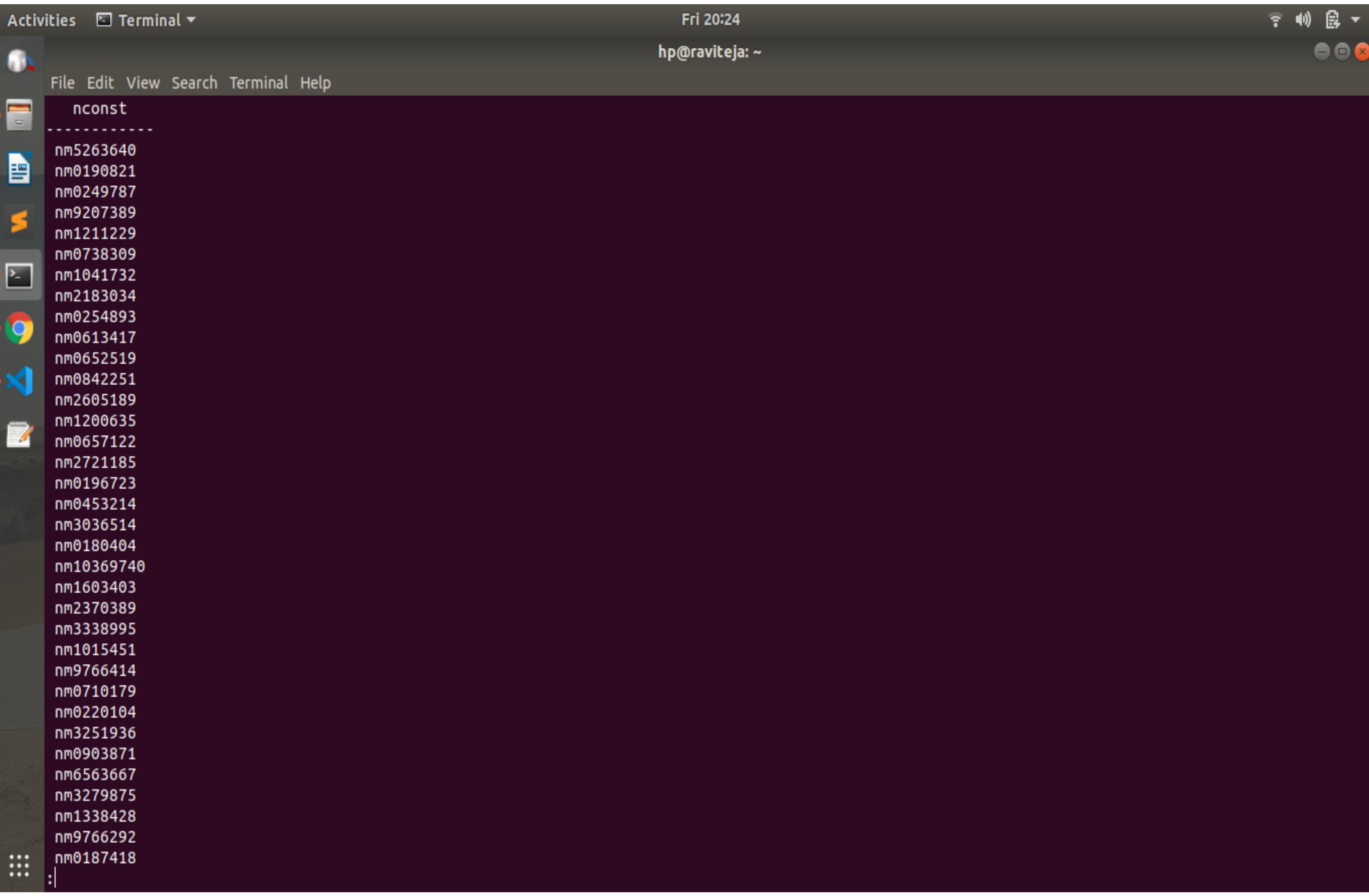
Box office collections and budget have not been recorded in the database. So the following assumptions are made.
OriginalTitles table has the two other columns budget and collections for each title. Just order by collections-budget and limit the results to 5.

```
select originalTitle from originaltitles where titletype='movie'
order by (case when collections is not NULL and budget is not null
then iscollections-budget else 0 end) desc limit 5;
```

13.Print actors who have worked in movies as well as TV series.

INTERSECT is used for the following query. Select only actors from the principals table with category='actor'. Then use the intersection of movie actors and tvSeries actors.

```
with actors as(
    select tconst,nconst from principals where category='actor'
)
select a1.nconst from actors a1 inner join originalTItles o1 on
o1.tconst=a1.tconst
group by a1.nconst,titletype having titletype='movie'
intersect
select a1.nconst from actors a1 inner join originalTItles o1 on
o1.tconst=a1.tconst
group by a1.nconst,titletype having titletype='tvSeries';
```

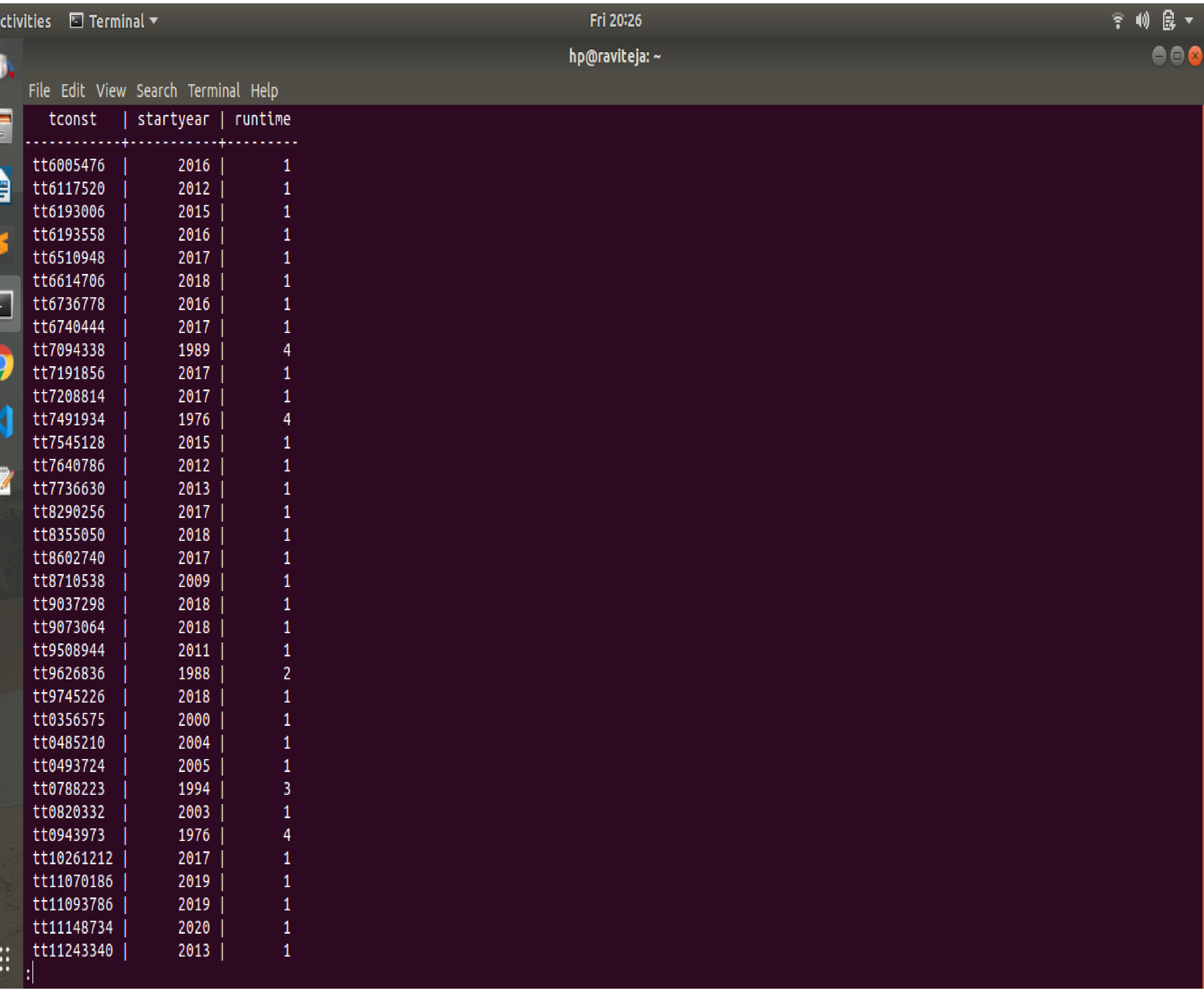


```
   nconst
------------
 nm5263640
 nm0190821
 nm0249787
 nm9207389
 nm1211229
 nm0738309
 nm1041732
 nm2183034
 nm0254893
 nm0613417
 nm0652519
 nm0842251
 nm2605189
 nm1200635
 nm0657122
 nm2721185
 nm0196723
 nm0453214
 nm3036514
 nm0180404
 nm10369740
 nm1603403
 nm2370389
 nm3338995
 nm1015451
 nm9766414
 nm0710179
 nm0220104
 nm3251936
 nm0903871
 nm6563667
 nm3279875
 nm1338428
 nm9766292
 nm0187418
:
```

14.Print the shortest TV episode for each year.

Firstly, consider the min_runtimes temporary table that has a minimum runtime for each year for tvEpisodes. Then select titleids from originaltitles with those values using INNER JOIN.

```
with min_runtimes as (
select startyear,min(runtime) as runtime from originaltitles where
titletype='tvEpisode'
group by startYear)
select tconst,o1.startyear,o1.runtime from originaltitles o1 inner
join min_runtimes m1 on o1.startyear=m1.startyear where
o1.runtime=m1.runtime;
```

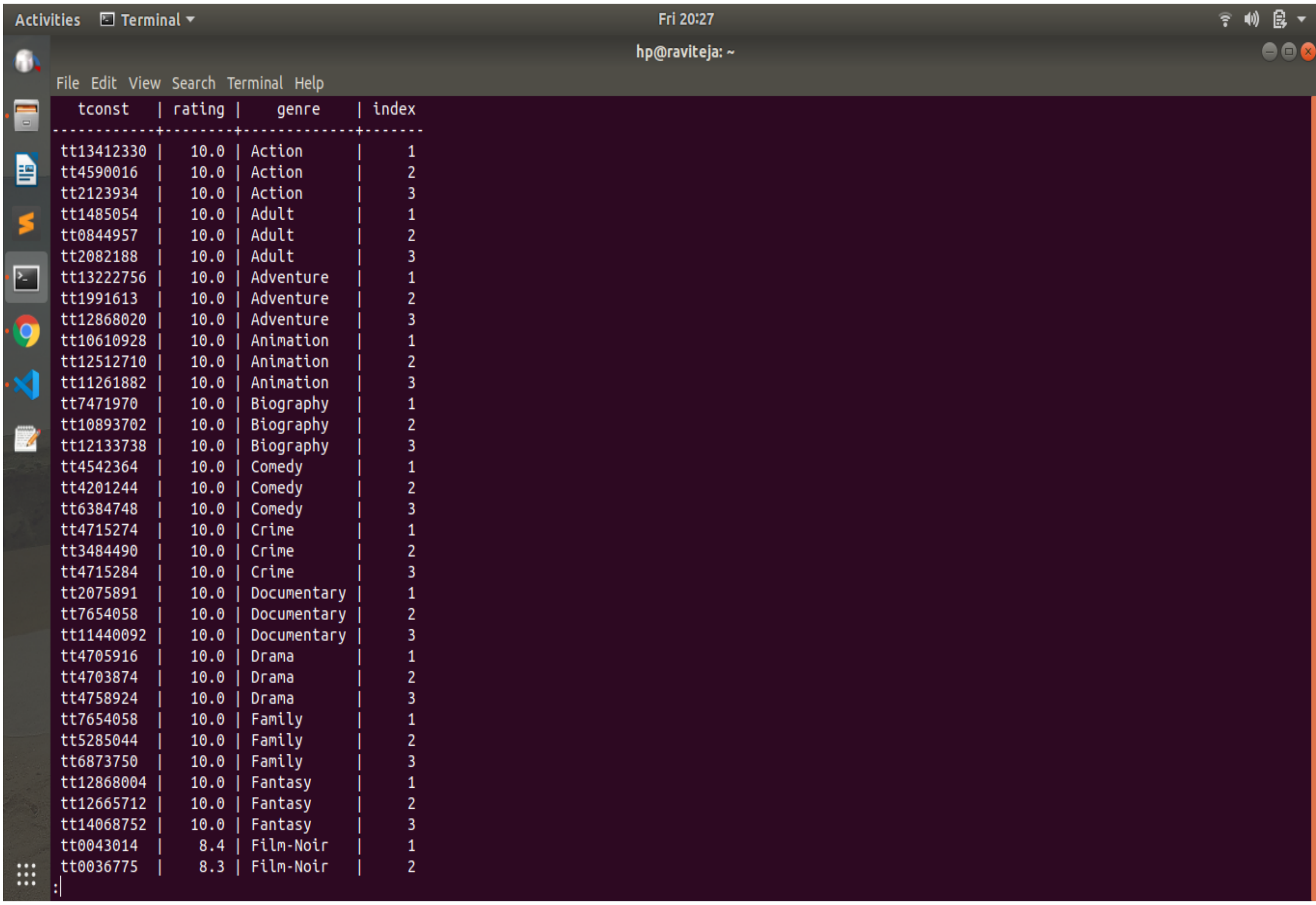


```
   tconst    | startyear | runtime
-------------+-----------+---------
 tt6005476   |      2016 |       1
 tt6117520   |      2012 |       1
 tt6193006   |      2015 |       1
 tt6193558   |      2016 |       1
 tt6510948   |      2017 |       1
 tt6614706   |      2018 |       1
 tt6736778   |      2016 |       1
 tt6740444   |      2017 |       1
 tt7094338   |      1989 |       4
 tt7191856   |      2017 |       1
 tt7208814   |      2017 |       1
 tt7491934   |      1976 |       4
 tt7545128   |      2015 |       1
 tt7640786   |      2012 |       1
 tt7736630   |      2013 |       1
 tt8290256   |      2017 |       1
 tt8355050   |      2018 |       1
 tt8602740   |      2017 |       1
 tt8710538   |      2009 |       1
 tt9037298   |      2018 |       1
 tt9073064   |      2018 |       1
 tt9508944   |      2011 |       1
 tt9626836   |      1988 |       2
 tt9745226   |      2018 |       1
 tt0356575   |      2000 |       1
 tt0485210   |      2004 |       1
 tt0493724   |      2005 |       1
 tt0788223   |      1994 |       3
 tt0820332   |      2003 |       1
 tt0943973   |      1976 |       4
 tt10261212  |      2017 |       1
 tt11070186  |      2019 |       1
 tt11093786  |      2019 |       1
 tt11148734  |      2020 |       1
 tt11243340  |      2013 |       1
:
```

15.You want to suggest some good movies to your friends. Genre wise print the top 3 rated Movies.

give row numbers with following constraints:
partitioned by genres, means each genre has separate numbering to itself.
Order by rating. Meaning, within each genre, index is given by descending order of rating.
Now select only those rows which have index<=3 to get top 3 ratings of each genre.

```
select * from
(select o1.tconst,
rating,
genre,
row_number () over (partition by genre order by (case when rating is
NULL then 0 else rating end) desc) as index
from originaltitles o1 inner JOIN title_genres g1 on
g1.tconst=o1.tconst
) as foo where foo.index<=3;
```



```
   tconst   | rating |    genre    | index
------------+--------+-------------+-------
 tt13412330 |   10.0 | Action      |     1
 tt4590016  |   10.0 | Action      |     2
 tt2123934  |   10.0 | Action      |     3
 tt1485054  |   10.0 | Adult       |     1
 tt0844957  |   10.0 | Adult       |     2
 tt2082188  |   10.0 | Adult       |     3
 tt13222756 |   10.0 | Adventure   |     1
 tt1991613  |   10.0 | Adventure   |     2
 tt12868020 |   10.0 | Adventure   |     3
 tt10610928 |   10.0 | Animation   |     1
 tt12512710 |   10.0 | Animation   |     2
 tt11261882 |   10.0 | Animation   |     3
 tt7471970  |   10.0 | Biography   |     1
 tt10893702 |   10.0 | Biography   |     2
 tt12133738 |   10.0 | Biography   |     3
 tt4542364  |   10.0 | Comedy      |     1
 tt4201244  |   10.0 | Comedy      |     2
 tt6384748  |   10.0 | Comedy      |     3
 tt4715274  |   10.0 | Crime       |     1
 tt3484490  |   10.0 | Crime       |     2
 tt4715284  |   10.0 | Crime       |     3
 tt2075891  |   10.0 | Documentary |     1
 tt7654058  |   10.0 | Documentary |     2
 tt11440092 |   10.0 | Documentary |     3
 tt4705916  |   10.0 | Drama       |     1
 tt4703874  |   10.0 | Drama       |     2
 tt4758924  |   10.0 | Drama       |     3
 tt7654058  |   10.0 | Family      |     1
 tt5285044  |   10.0 | Family      |     2
 tt6873750  |   10.0 | Family      |     3
 tt12868004 |   10.0 | Fantasy     |     1
 tt12665712 |   10.0 | Fantasy     |     2
 tt14068752 |   10.0 | Fantasy     |     3
 tt0043014  |    8.4 | Film-Noir   |     1
 tt0036775  |    8.3 | Film-Noir   |     2
:
```

16.Find the movies and TV series which are filmed in Switzerland. A TV series can be counted as filmed in a country if there exists at least one episode filmed in that country.

title_locations table has not been included in the database .So the following table with fields are assumed:
title_locations table has attributed location_name and tcons that refers to originaltitles.tconst).
Select those titles of only movies and tvSeries which have entries with Switzerland in title_locaitons.

```
select tl.tconst from title_locations tl where
location_name='Switzerland' and
((select titletype from originalTItles o where
o.tconst=tl.tconst)='movie' or
(select titletype from originalTItles o where
o.tconst=tl.tconst)='tvSeries');
```

17.List all movies who have A certificate in the same location in the year 1995.

The question is a little ambiguous at "Same location in year" part. So it is assumed that the location name is "Switzerland" and carried out with the query.

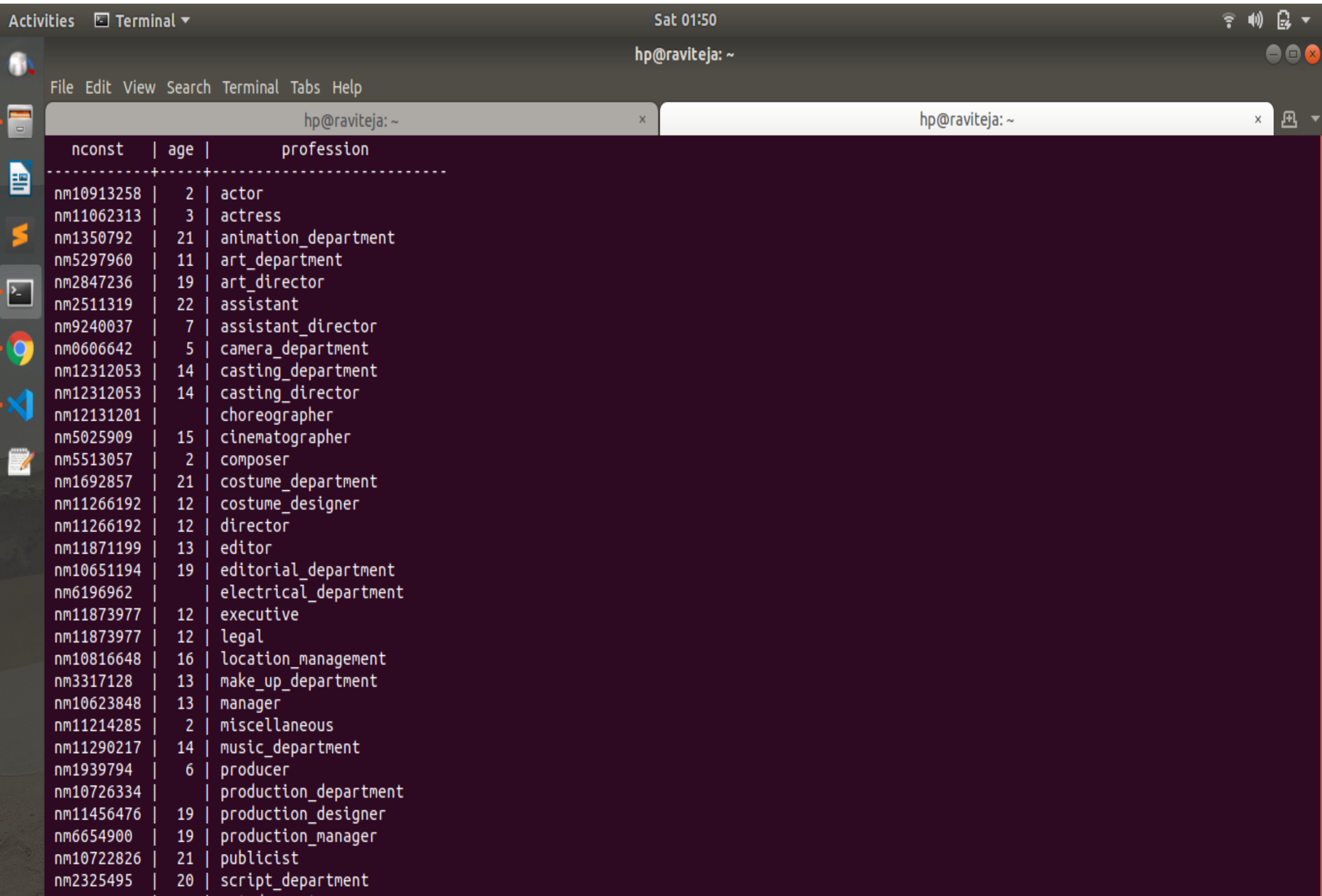
title_locations is assumed to be having the earlier described attributes along with certificate attributes.
Select those titles which are movies and has certificate type A in 1995.
Since the question about the same location is not clear, it is assumed as the continuation of the previous question so the Switzerland location is used.

```
select tl.tconst from title_locations tl inner join originalTItles o1
on o1.tconst=tl.tconst
where tl.certificate='A' and titletype='Movie' AND (startYear>=1995
and endYear<=1995) and location_name='Switzerland';
```

18.For each profession print the youngest one.

min_ages contains nconst, age,profession and row_number partitioned by profession and ordered by age.
To ignore corrupted and unspecified data, only positive ages will be considered. Now, select only those rows with index 1 for the youngest of the profession.

```
with min_ages as (
    select p1.nconst,age,profession,
    row_number () over (partition by profession order by (case when
(age is NULL or age<=1)  then 10000 else age end)) as index
    from castAndCrewProfession p1 inner join cast_and_crew c1
    on c1.nconst=p1.nconst
)

select nconst,age,profession from min_ages where index=1;
```



```
  nconst    | age |        profession
------------+-----+-------------------------
 nm10913258 |   2 | actor
 nm11062313 |   3 | actress
 nm1350792  |  21 | animation_department
 nm5297960  |  11 | art_department
 nm2847236  |  19 | art_director
 nm2511319  |  22 | assistant
 nm9240037  |   7 | assistant_director
 nm0606642  |   5 | camera_department
 nm12312053 |  14 | casting_department
 nm12312053 |  14 | casting_director
 nm12131201 |     | choreographer
 nm5025909  |  15 | cinematographer
 nm5513057  |   2 | composer
 nm1692857  |  21 | costume_department
 nm11266192 |  12 | costume_designer
 nm11266192 |  12 | director
 nm11871199 |  13 | editor
 nm10651194 |  19 | editorial_department
 nm6196962  |     | electrical_department
 nm11873977 |  12 | executive
 nm11873977 |  12 | legal
 nm10816648 |  16 | location_management
 nm3317128  |  13 | make_up_department
 nm10623848 |  13 | manager
 nm11214285 |   2 | miscellaneous
 nm11290217 |  14 | music_department
 nm1939794  |   6 | producer
 nm10726334 |     | production_department
 nm11456476 |  19 | production_designer
 nm6654900  |  19 | production_manager
 nm10722826 |  21 | publicist
 nm2325495  |  20 | script_department
```

19.Print all the music technicians(soundtrack producers) who have worked for at least 5 movies.

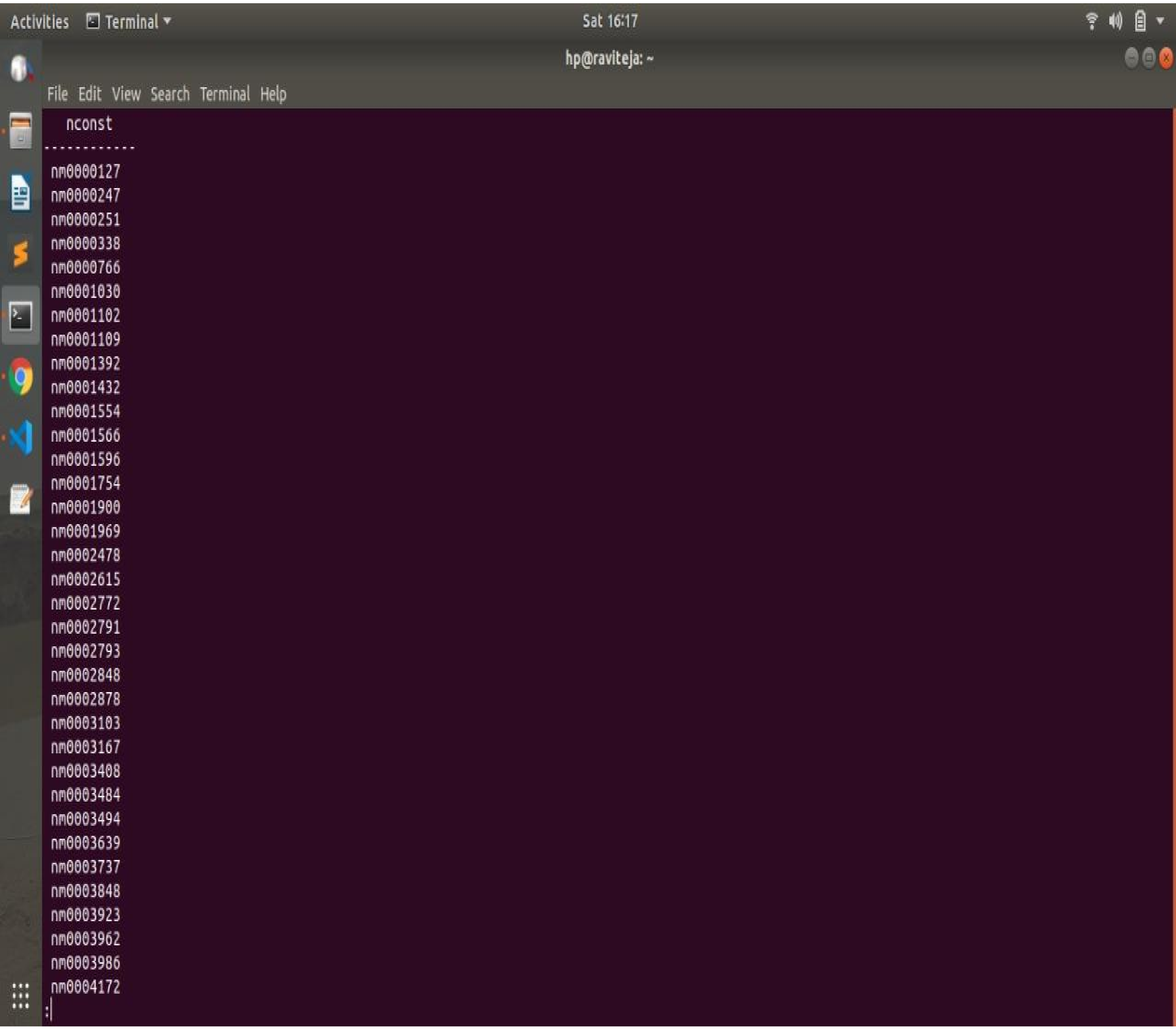
The soundtrack table is not created. So the following are the assumptions:
it has attributed producer_name and tconst(referring to originalTitles.tconst). Self-explanatory.
sound_producers has soundtracks only for movies. Select those producer_names who have count>5 in sound_rpoducers.

```
with sound_producers as (
    select producer_name from soundtrack s1 inner join originalTitles
o1 on o1.tconst=s1.tconst
where titletype='Movie'
)
select distinct producer_name from sound_producers s1
where (select count(s1.producer_name) from sound_producers)>=5 ;
```

20.Print the actor's name who has worked in as many movies as the number of crew members in the movie titled: ‘tt0000003’.

tconst considered here is 'tt0000003' has 4 crew members.
Now out of all actors from principals table, select those whose count = count(crew of tt0000003)

```
select nconst from principals p1 where category='actor' group by
nconst having count()=(select count() from principals p2 where
p2.tconst='tt0000003');
```



```
   nconst
-----------
 nm0000127
 nm0000247
 nm0000251
 nm0000338
 nm0000766
 nm0001030
 nm0001102
 nm0001109
 nm0001392
 nm0001432
 nm0001554
 nm0001566
 nm0001596
 nm0001754
 nm0001900
 nm0001969
 nm0002478
 nm0002615
 nm0002772
 nm0002791
 nm0002793
 nm0002848
 nm0002878
 nm0003103
 nm0003167
 nm0003408
 nm0003484
 nm0003494
 nm0003639
 nm0003737
 nm0003848
 nm0003923
 nm0003962
 nm0003986
 nm0004172
:
```