DBMS : CS3563 : Assignment 3 Report

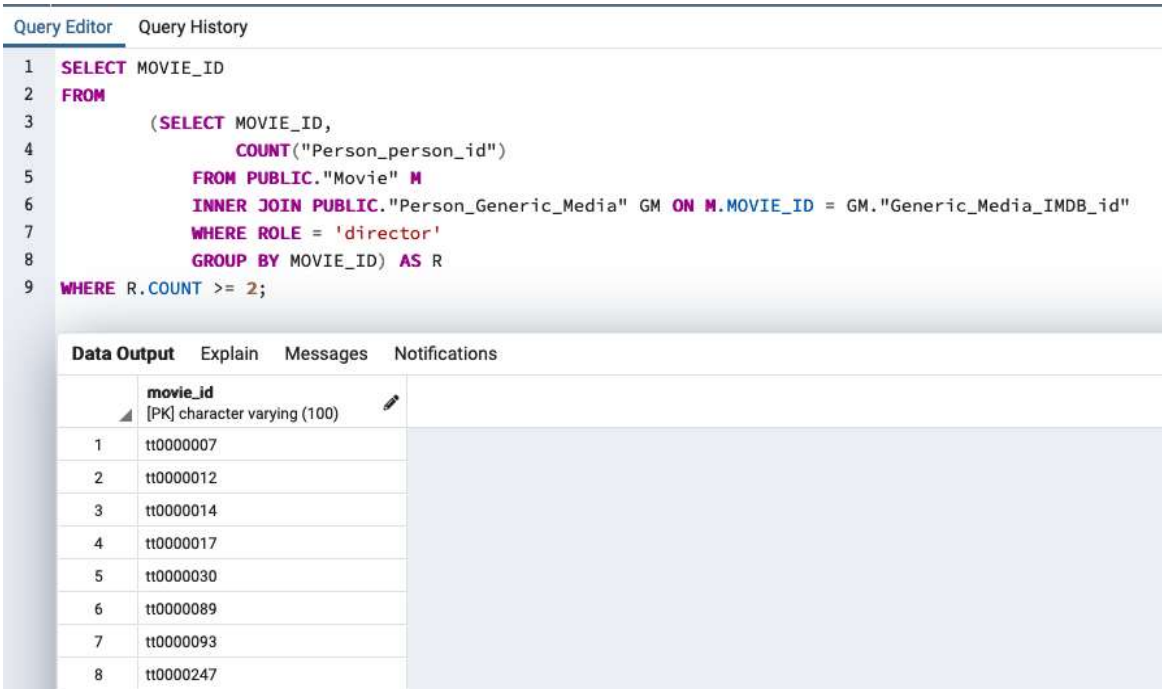
GROUP 4

Name	**Roll no.**
Raj Patil	CS18BTECH11039
Vedant Singh	CS18BTECH11047
Karan Bhukar	CS18BTECH11021
Himanshu Bishnoi	CS16BTECH11018

Describing Deliverables

```
4_assgn3.zip
     |
     |---DBMS A3 Report Grp4.pdf
     |
     |---DB_A3_GRP4_Queries.sql
```

1.

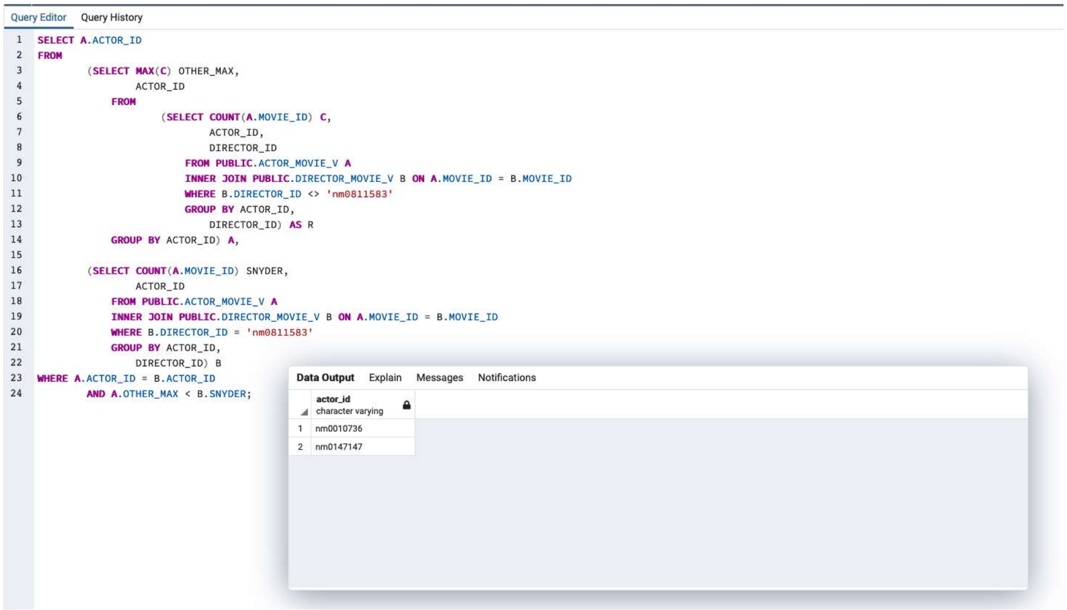


```sql
SELECT MOVIE_ID
FROM
        (SELECT MOVIE_ID,
                COUNT("Person_person_id")
            FROM PUBLIC."Movie" M
            INNER JOIN PUBLIC."Person_Generic_Media" GM ON M.MOVIE_ID = GM."Generic_Media_IMDB_id"
            WHERE ROLE = 'director'
            GROUP BY MOVIE_ID) AS R
WHERE R.COUNT >= 2;
```

	movie_id [PK] character varying (100)
1	tt0000007
2	tt0000012
3	tt0000014
4	tt0000017
5	tt0000030
6	tt0000089
7	tt0000093
8	tt0000247

Firstly we find the directors for each movie using the Person_Generic_Media table. We then aggregate using the movie_id to count the number of directors for each of them. Finally, only those movie_id's are displayed where the count is at least 2.

2.



```sql
SELECT A.ACTOR_ID
FROM
        (SELECT MAX(C) OTHER_MAX,
                ACTOR_ID
            FROM
                    (SELECT COUNT(A.MOVIE_ID) C,
                            ACTOR_ID,
                            DIRECTOR_ID
                        FROM PUBLIC.ACTOR_MOVIE_V A
                        INNER JOIN PUBLIC.DIRECTOR_MOVIE_V B ON A.MOVIE_ID = B.MOVIE_ID
                        WHERE B.DIRECTOR_ID <> 'nm0811583'
                        GROUP BY ACTOR_ID,
                            DIRECTOR_ID) AS R
            GROUP BY ACTOR_ID) A,

        (SELECT COUNT(A.MOVIE_ID) SNYDER,
                ACTOR_ID
            FROM PUBLIC.ACTOR_MOVIE_V A
            INNER JOIN PUBLIC.DIRECTOR_MOVIE_V B ON A.MOVIE_ID = B.MOVIE_ID
            WHERE B.DIRECTOR_ID = 'nm0811583'
            GROUP BY ACTOR_ID,
                DIRECTOR_ID) B
WHERE A.ACTOR_ID = B.ACTOR_ID
        AND A.OTHER_MAX < B.SNYDER;
```

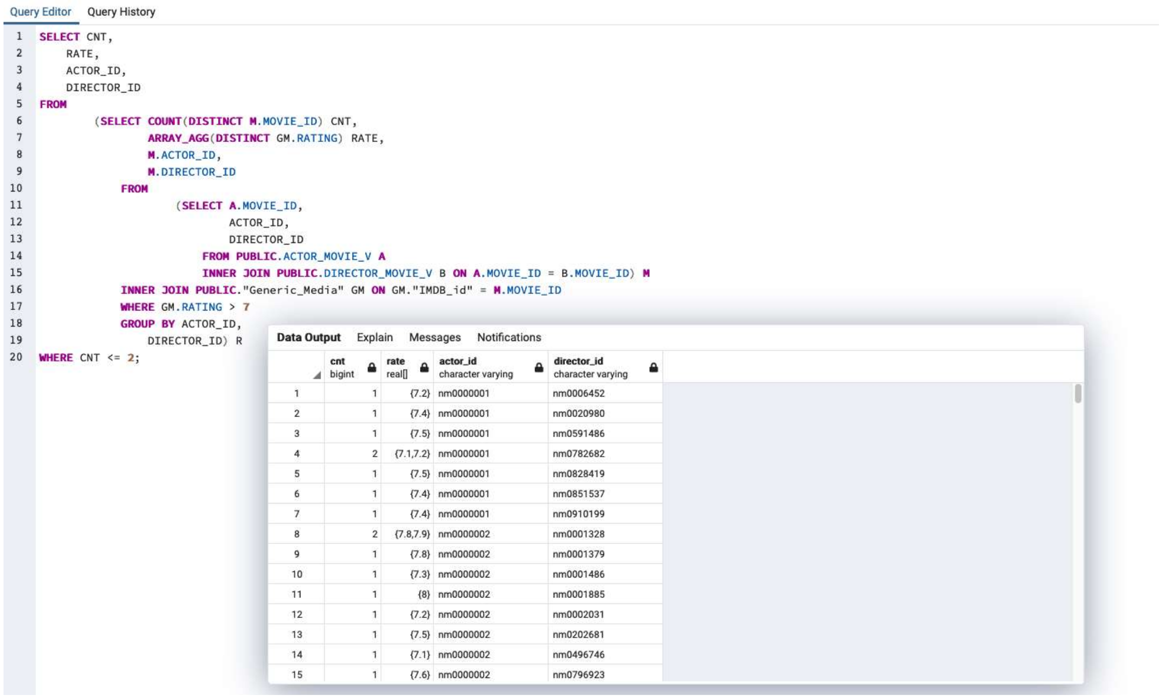
	actor_id character varying
1	nm0010736
2	nm0147147

Two convenience views are created which contain the director and the actor info for each movie. For each actor director pair we count the number of movies done by them. This is done in two different places, one where the director is 'Zack Snyder' and the other where the director is anyone but 'Zack Snyder'. From the second part, the maximum is filtered out for each actor and is compared to the first aggregate (which contain the number of movies a certain actor has done with 'Zack Snyder'). If the second quantity is bigger, we print the corresponding actor_id.

3.

According to our ERD, we had a table which contained the nominations for each movie. From this we filter out those nominations which ultimately resulted in the award being given out. These nominations are then grouped according to the movie_id and those movies are selected whose award count is at least 2. Finally, we select all the movies except those selected above.

4.



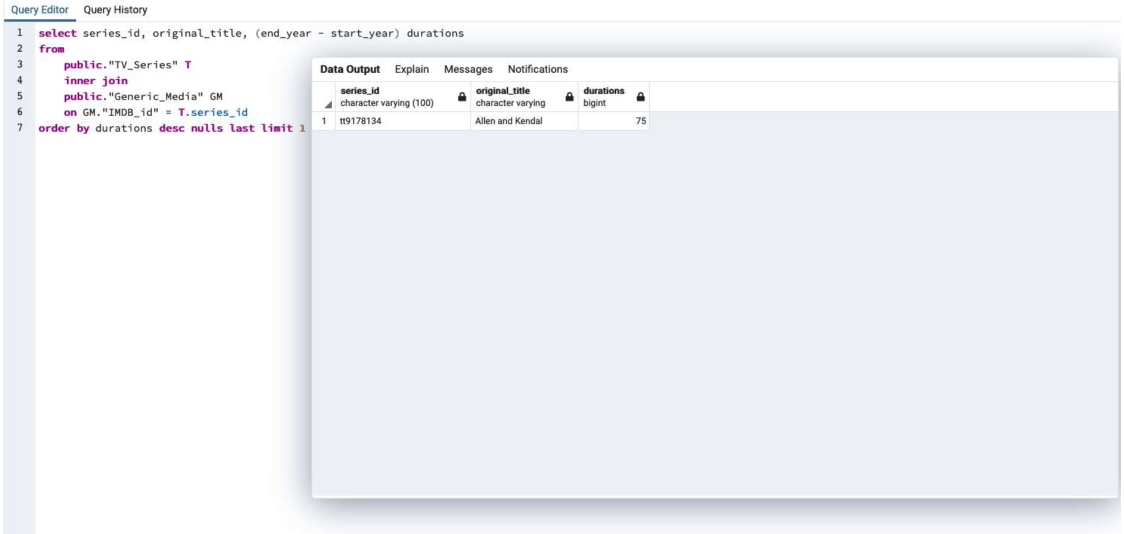
```
SELECT CNT,
    RATE,
    ACTOR_ID,
    DIRECTOR_ID
FROM
        (SELECT COUNT(DISTINCT M.MOVIE_ID) CNT,
                ARRAY_AGG(DISTINCT GM.RATING) RATE,
                M.ACTOR_ID,
                M.DIRECTOR_ID
            FROM
                    (SELECT A.MOVIE_ID,
                            ACTOR_ID,
                            DIRECTOR_ID
                        FROM PUBLIC.ACTOR_MOVIE_V A
                        INNER JOIN PUBLIC.DIRECTOR_MOVIE_V B ON A.MOVIE_ID = B.MOVIE_ID) M
            INNER JOIN PUBLIC."Generic_Media" GM ON GM."IMDB_id" = M.MOVIE_ID
            WHERE GM.RATING > 7
            GROUP BY ACTOR_ID,
                DIRECTOR_ID) R
WHERE CNT <= 2;
```

	cnt bigint	rate real[]	actor_id character varying	director_id character varying
1	1	{7.2}	nm0000001	nm0006452
2	1	{7.4}	nm0000001	nm0020980
3	1	{7.5}	nm0000001	nm0591486
4	2	{7.1,7.2}	nm0000001	nm0782682
5	1	{7.5}	nm0000001	nm0828419
6	1	{7.4}	nm0000001	nm0851537
7	1	{7.4}	nm0000001	nm0910199
8	2	{7.8,7.9}	nm0000002	nm0001328
9	1	{7.8}	nm0000002	nm0001379
10	1	{7.3}	nm0000002	nm0001486
11	1	{8}	nm0000002	nm0001885
12	1	{7.2}	nm0000002	nm0002031
13	1	{7.5}	nm0000002	nm0202681
14	1	{7.1}	nm0000002	nm0496746
15	1	{7.6}	nm0000002	nm0796923

For this we use the same views created in the second question. First we find all the actor director pairs and the ratings of the movies done by them. We now aggregate by grouping according to actor and director pairs, and select the maximum rating of any movie done by a pair along with the count of the movies done by this pair. If the maximum rating is greater than 7, this means that there is at least one movie done by this pair which is rated higher than 7. The count filter is used to ensure that the number of movies done by this pair should be at most 2.

Note - There is small ambiguity in the question. We are not exactly sure what is meant by *'the movie done by them has a rating above 7'*. Should all the movies done by the pair have a rating greater than 7 or any one. We have done the latter. If it is the former, changing MAX to MIN in the query would be sufficient.

5.

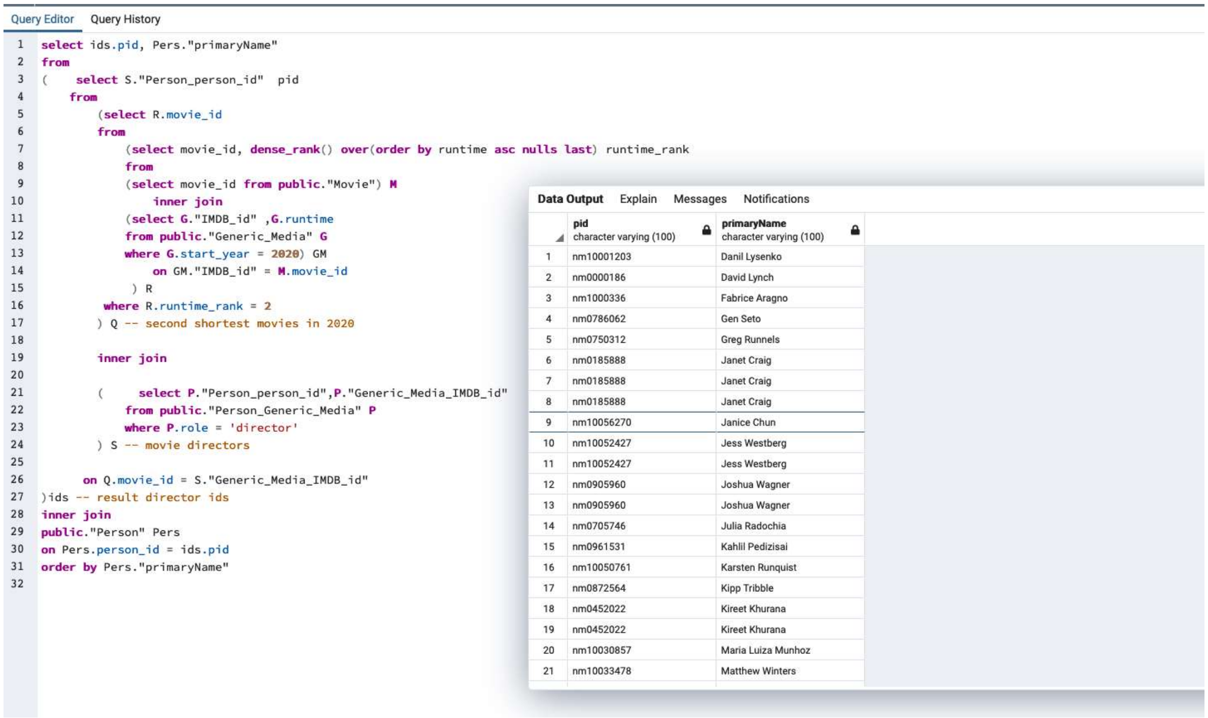


```sql
select series_id, original_title, (end_year - start_year) durations
from
    public."TV_Series" T
    inner join
    public."Generic_Media" GM
    on GM."IMDB_id" = T.series_id
order by durations desc nulls last limit 1
```

	series_id character varying (100)	original_title character varying	durations bigint
1	tt9178134	Allen and Kendal	75

We assimilate the duration info by subtracting the start year from the end year. Series are then arranged in descending order according to this and the first entry is selected.

6.

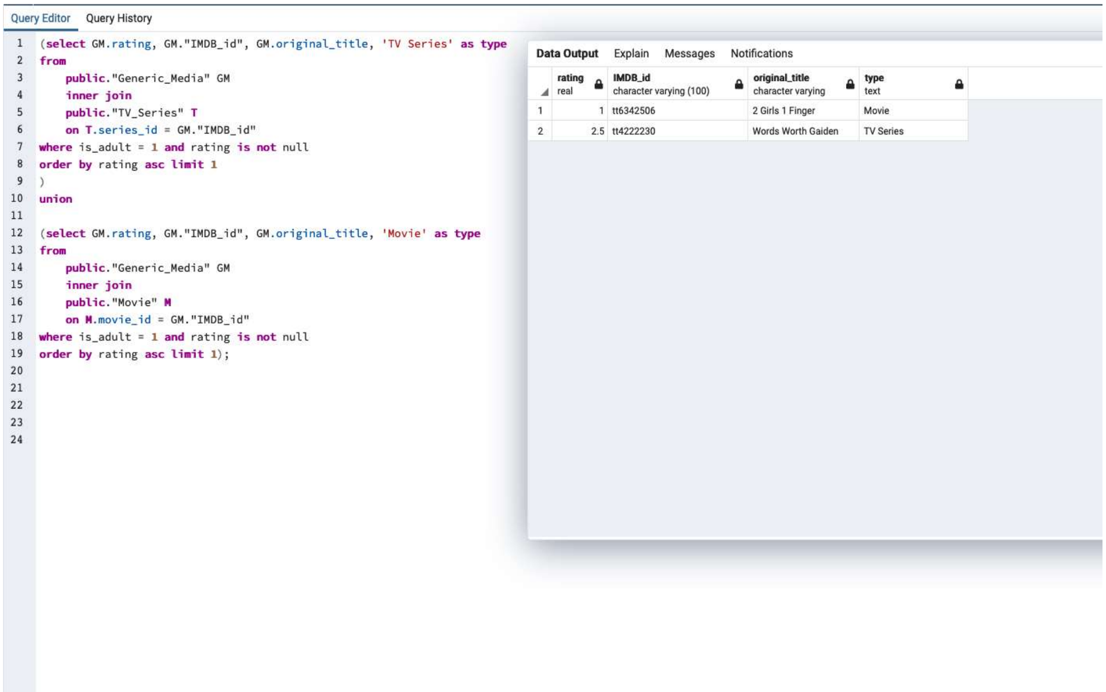


```sql
select ids.pid, Pers."primaryName"
from
(    select S."Person_person_id"  pid
    from
        (select R.movie_id
        from
            (select movie_id, dense_rank() over(order by runtime asc nulls last) runtime_rank
            from
            (select movie_id from public."Movie") M
                inner join
            (select G."IMDB_id" ,G.runtime
            from public."Generic_Media" G
            where G.start_year = 2020) GM
                on GM."IMDB_id" = M.movie_id
             ) R
         where R.runtime_rank = 2
        ) Q -- second shortest movies in 2020

        inner join

        (     select P."Person_person_id",P."Generic_Media_IMDB_id"
            from public."Person_Generic_Media" P
            where P.role = 'director'
        ) S -- movie directors

      on Q.movie_id = S."Generic_Media_IMDB_id"
)ids -- result director ids
inner join
public."Person" Pers
on Pers.person_id = ids.pid
order by Pers."primaryName"
```

	pid character varying (100)	primaryName character varying (100)
1	nm10001203	Danil Lysenko
2	nm0000186	David Lynch
3	nm1000336	Fabrice Aragno
4	nm0786062	Gen Seto
5	nm0750312	Greg Runnels
6	nm0185888	Janet Craig
7	nm0185888	Janet Craig
8	nm0185888	Janet Craig
9	nm10056270	Janice Chun
10	nm10052427	Jess Westberg
11	nm10052427	Jess Westberg
12	nm0905960	Joshua Wagner
13	nm0905960	Joshua Wagner
14	nm0705746	Julia Radochia
15	nm0961531	Kahlil Pedizisai
16	nm10050761	Karsten Runquist
17	nm0872564	Kipp Tribble
18	nm0452022	Kireet Khurana
19	nm0452022	Kireet Khurana
20	nm10030857	Maria Luiza Munhoz
21	nm10033478	Matthew Winters

We use dense ranking to rank the movies according to their runtime and select the movies with the second shortest runtime. Directors of these movies are found by doing an inner join with Person_Generic_Media table which contains data about the crew of each generic media.

7.

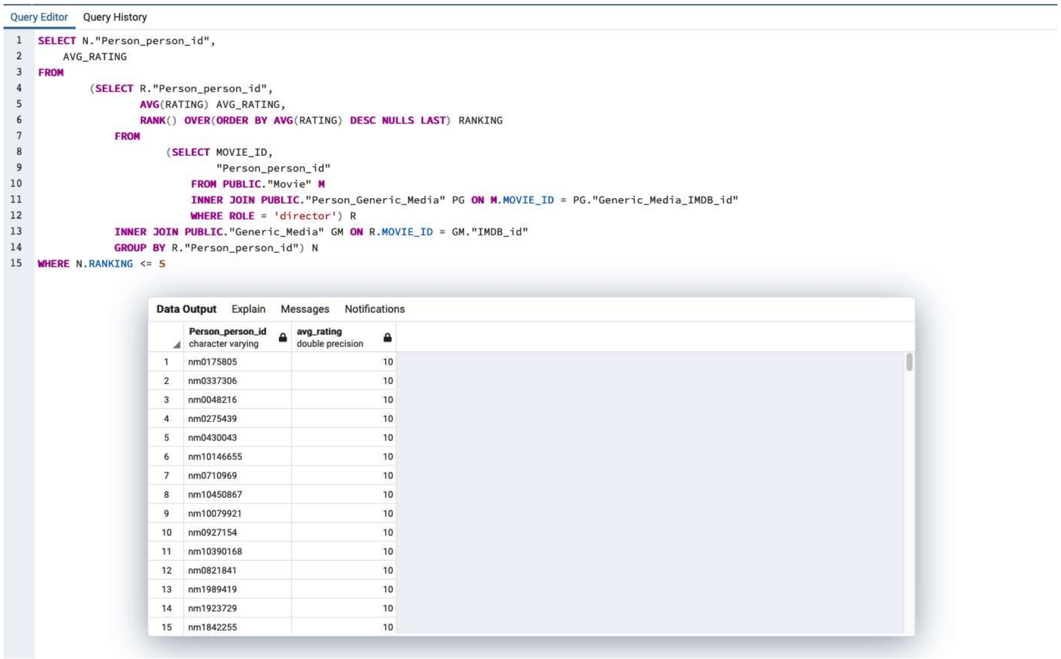


```sql
(select GM.rating, GM."IMDB_id", GM.original_title, 'TV Series' as type
from
    public."Generic_Media" GM
    inner join
    public."TV_Series" T
    on T.series_id = GM."IMDB_id"
where is_adult = 1 and rating is not null
order by rating asc limit 1
)
union

(select GM.rating, GM."IMDB_id", GM.original_title, 'Movie' as type
from
    public."Generic_Media" GM
    inner join
    public."Movie" M
    on M.movie_id = GM."IMDB_id"
where is_adult = 1 and rating is not null
order by rating asc limit 1);
```

	rating real	IMDB_id character varying (100)	original_title character varying	type text
1	1	tt6342506	2 Girls 1 Finger	Movie
2	2.5	tt4222230	Words Worth Gaiden	TV Series

We find out the info about each movie and TV series by doing an inner join with Generic_Media for each of them. We use this info to filter out the non-adult movies and finally order them according to their ratings and finally select the one with the lowest rating.

8.



```sql
SELECT N."Person_person_id",
    AVG_RATING
FROM
        (SELECT R."Person_person_id",
                AVG(RATING) AVG_RATING,
                RANK() OVER(ORDER BY AVG(RATING) DESC NULLS LAST) RANKING
            FROM
                    (SELECT MOVIE_ID,
                            "Person_person_id"
                        FROM PUBLIC."Movie" M
                        INNER JOIN PUBLIC."Person_Generic_Media" PG ON M.MOVIE_ID = PG."Generic_Media_IMDB_id"
                        WHERE ROLE = 'director') R
            INNER JOIN PUBLIC."Generic_Media" GM ON R.MOVIE_ID = GM."IMDB_id"
            GROUP BY R."Person_person_id") N
WHERE N.RANKING <= 5
```

	Person_person_id character varying	avg_rating double precision
1	nm0175805	10
2	nm0337306	10
3	nm0048216	10
4	nm0275439	10
5	nm0430043	10
6	nm10146655	10
7	nm0710969	10
8	nm10450867	10
9	nm10079921	10
10	nm0927154	10
11	nm10390168	10
12	nm0821841	10
13	nm1989419	10
14	nm1923729	10
15	nm1842255	10

We do inner join on Person_Generic_Media and Generic_Media to get the rating data and director data for all the movies. We then group by director id and calculate the average rating of all the movies done by each director. A sparse ranking is done on the average rating and ranks <= 5 are selected. The final results contain ~200 directors all with the same average rating.

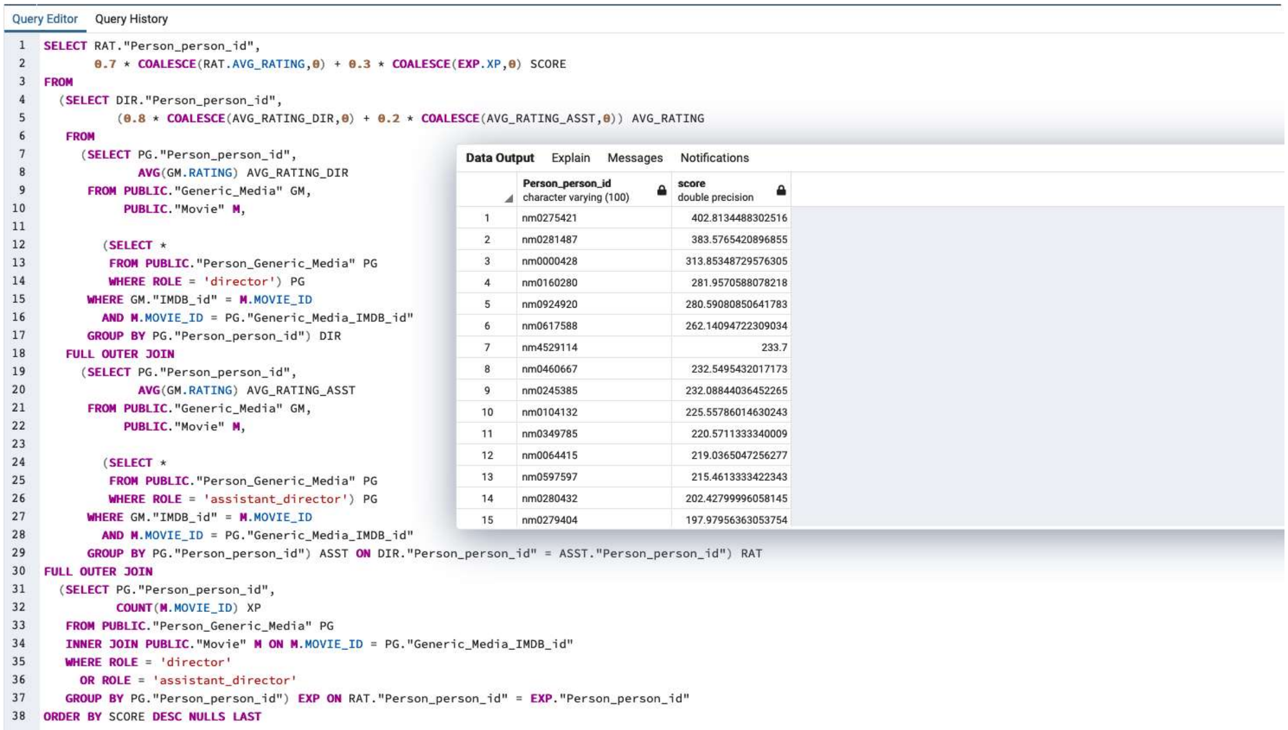
9.

Firstly we calculate all the TV series which have been aired in at least 3 locations. We then calculate all the series which have been produced by at least 2 production companies. Selecting the intersection of these two sets gives us the desired result.

10.

We first filter out all the Oscar awards nominations which were actually awarded from the nominations table. This is possible because we have created awards as entities with award name as an attribute. Then these nominations are paired with the person who received this nomination and this is then ordered according to the award year.

11.

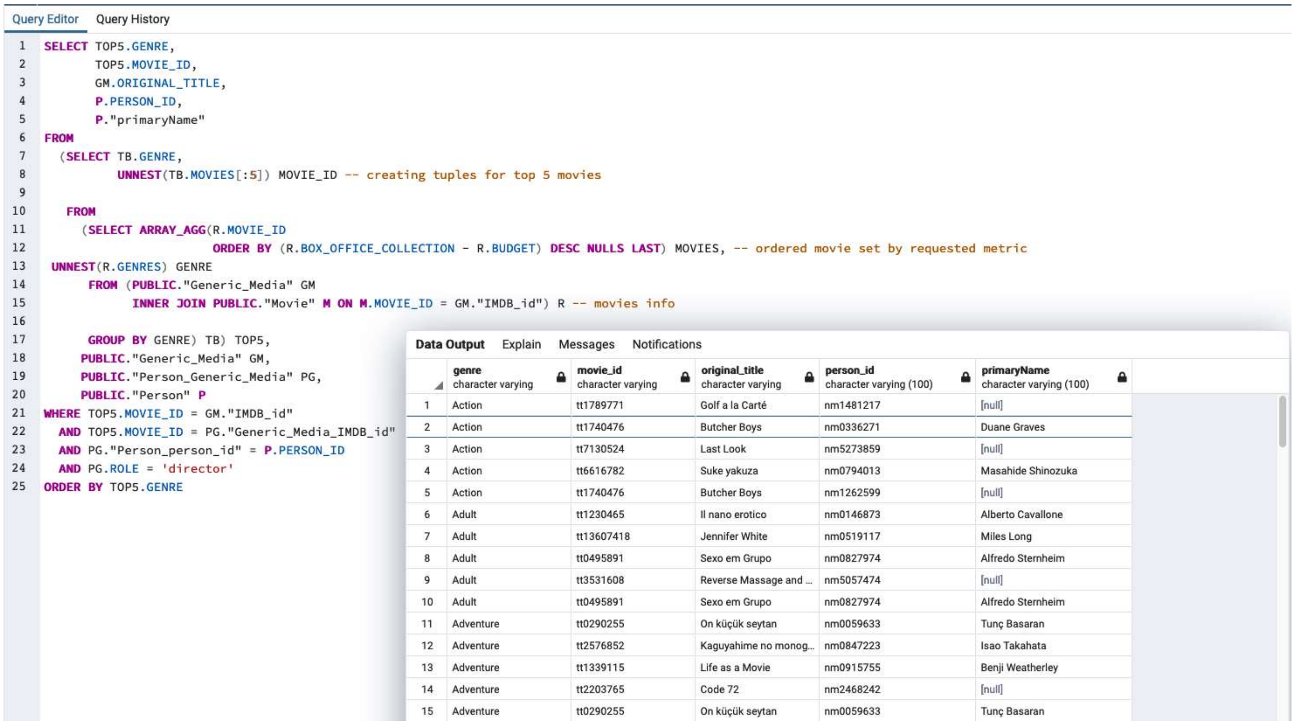


```sql
SELECT RAT."Person_person_id",
       0.7 * COALESCE(RAT.AVG_RATING,0) + 0.3 * COALESCE(EXP.XP,0) SCORE
FROM
  (SELECT DIR."Person_person_id",
          (0.8 * COALESCE(AVG_RATING_DIR,0) + 0.2 * COALESCE(AVG_RATING_ASST,0)) AVG_RATING
   FROM
     (SELECT PG."Person_person_id",
             AVG(GM.RATING) AVG_RATING_DIR
      FROM PUBLIC."Generic_Media" GM,
           PUBLIC."Movie" M,

        (SELECT *
         FROM PUBLIC."Person_Generic_Media" PG
         WHERE ROLE = 'director') PG
      WHERE GM."IMDB_id" = M.MOVIE_ID
        AND M.MOVIE_ID = PG."Generic_Media_IMDB_id"
      GROUP BY PG."Person_person_id") DIR
   FULL OUTER JOIN
     (SELECT PG."Person_person_id",
             AVG(GM.RATING) AVG_RATING_ASST
      FROM PUBLIC."Generic_Media" GM,
           PUBLIC."Movie" M,

        (SELECT *
         FROM PUBLIC."Person_Generic_Media" PG
         WHERE ROLE = 'assistant_director') PG
      WHERE GM."IMDB_id" = M.MOVIE_ID
        AND M.MOVIE_ID = PG."Generic_Media_IMDB_id"
      GROUP BY PG."Person_person_id") ASST ON DIR."Person_person_id" = ASST."Person_person_id") RAT
FULL OUTER JOIN
  (SELECT PG."Person_person_id",
          COUNT(M.MOVIE_ID) XP
   FROM PUBLIC."Person_Generic_Media" PG
   INNER JOIN PUBLIC."Movie" M ON M.MOVIE_ID = PG."Generic_Media_IMDB_id"
   WHERE ROLE = 'director'
     OR ROLE = 'assistant_director'
   GROUP BY PG."Person_person_id") EXP ON RAT."Person_person_id" = EXP."Person_person_id"
ORDER BY SCORE DESC NULLS LAST
```

	Person_person_id character varying (100)	score double precision
1	nm0275421	402.8134488302516
2	nm0281487	383.5765420896855
3	nm0000428	313.85348729576305
4	nm0160280	281.9570588078218
5	nm0924920	280.59080850641783
6	nm0617588	262.14094722309034
7	nm4529114	233.7
8	nm0460667	232.5495432017173
9	nm0245385	232.08844036452265
10	nm0104132	225.55786014630243
11	nm0349785	220.5711333340009
12	nm0064415	219.0365047256277
13	nm0597597	215.4613333422343
14	nm0280432	202.42799996058145
15	nm0279404	197.97956363053754

In one part, we calculate the number of movies done by each person either as a director or an assistant director. In another part, the average ratings of all the movies done by a person are calculated in both the above mentioned roles. These ratings are then combined according to the mathematical expression given in the question. Finally, the experience and the calculated average ratings are combined together to give the final score for each person. We use the coalesce function in this query to ensure that NULL is considered as 0 while doing mathematical calculations.

12.

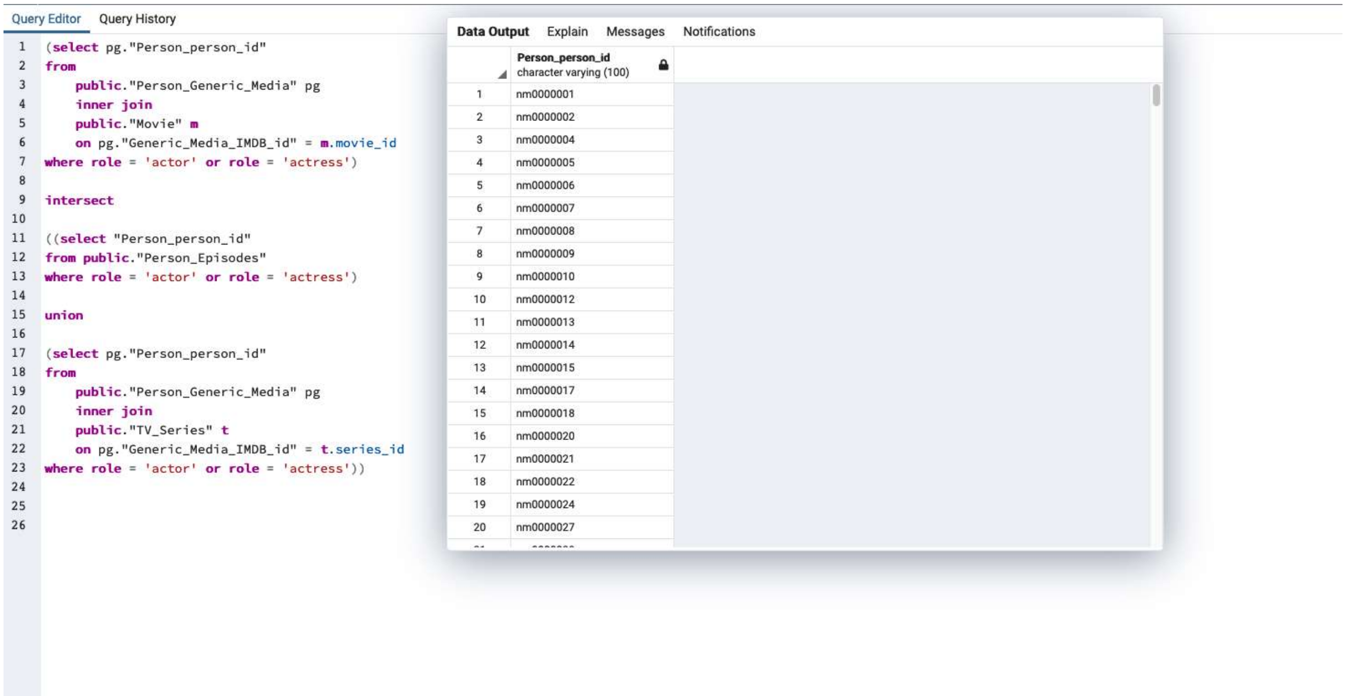


```sql
SELECT TOP5.GENRE,
       TOP5.MOVIE_ID,
       GM.ORIGINAL_TITLE,
       P.PERSON_ID,
       P."primaryName"
FROM
  (SELECT TB.GENRE,
          UNNEST(TB.MOVIES[:5]) MOVIE_ID -- creating tuples for top 5 movies

   FROM
     (SELECT ARRAY_AGG(R.MOVIE_ID
                       ORDER BY (R.BOX_OFFICE_COLLECTION - R.BUDGET) DESC NULLS LAST) MOVIES, -- ordered movie set by requested metric
 UNNEST(R.GENRES) GENRE
      FROM (PUBLIC."Generic_Media" GM
            INNER JOIN PUBLIC."Movie" M ON M.MOVIE_ID = GM."IMDB_id") R -- movies info

      GROUP BY GENRE) TB) TOP5,
     PUBLIC."Generic_Media" GM,
     PUBLIC."Person_Generic_Media" PG,
     PUBLIC."Person" P
WHERE TOP5.MOVIE_ID = GM."IMDB_id"
  AND TOP5.MOVIE_ID = PG."Generic_Media_IMDB_id"
  AND PG."Person_person_id" = P.PERSON_ID
  AND PG.ROLE = 'director'
ORDER BY TOP5.GENRE
```

	genre character varying	movie_id character varying	original_title character varying	person_id character varying (100)	primaryName character varying (100)
1	Action	tt1789771	Golf a la Carté	nm1481217	[null]
2	Action	tt1740476	Butcher Boys	nm0336271	Duane Graves
3	Action	tt7130524	Last Look	nm5273859	[null]
4	Action	tt6616782	Suke yakuza	nm0794013	Masahide Shinozuka
5	Action	tt1740476	Butcher Boys	nm1262599	[null]
6	Adult	tt1230465	Il nano erotico	nm0146873	Alberto Cavallone
7	Adult	tt13607418	Jennifer White	nm0519117	Miles Long
8	Adult	tt0495891	Sexo em Grupo	nm0827974	Alfredo Sternheim
9	Adult	tt3531608	Reverse Massage and ...	nm5057474	[null]
10	Adult	tt0495891	Sexo em Grupo	nm0827974	Alfredo Sternheim
11	Adventure	tt0290255	On küçük seytan	nm0059633	Tunç Basaran
12	Adventure	tt2576852	Kaguyahime no monog...	nm0847223	Isao Takahata
13	Adventure	tt1339115	Life as a Movie	nm0915755	Benji Weatherley
14	Adventure	tt2203765	Code 72	nm2468242	[null]
15	Adventure	tt0290255	On küçük seytan	nm0059633	Tunç Basaran

We have stored the genres in an array for all the movies. For this question, we use the unnest function to get multiple tuples for each movie associated with just one genre. Since postgres allows us to use order by in window function, we make use of this to aggregate genre wise the top 5 movies ordered by their earnings. We now again use unnest to create a different tuple for all these movies and extract their directors.

13.

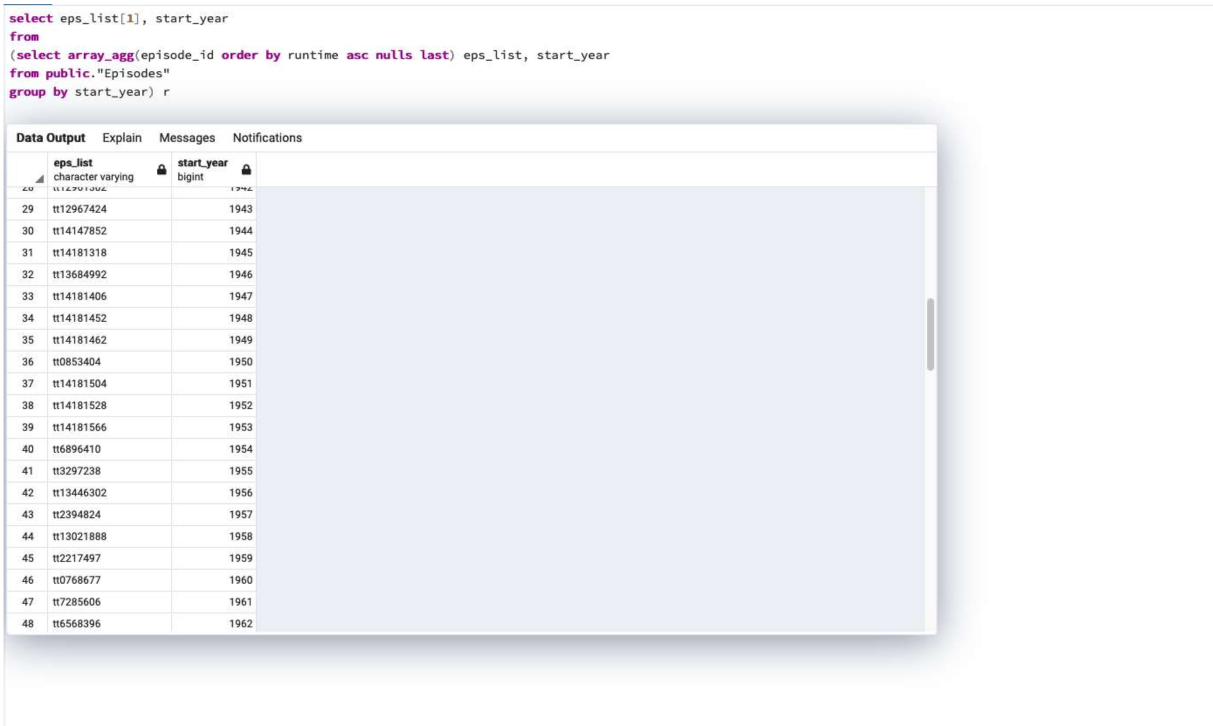


```
(select pg."Person_person_id"
from
    public."Person_Generic_Media" pg
    inner join
    public."Movie" m
    on pg."Generic_Media_IMDB_id" = m.movie_id
where role = 'actor' or role = 'actress')

intersect

((select "Person_person_id"
from public."Person_Episodes"
where role = 'actor' or role = 'actress')

union

(select pg."Person_person_id"
from
    public."Person_Generic_Media" pg
    inner join
    public."TV_Series" t
    on pg."Generic_Media_IMDB_id" = t.series_id
where role = 'actor' or role = 'actress'))
```

	Person_person_id character varying (100)
1	nm0000001
2	nm0000002
3	nm0000004
4	nm0000005
5	nm0000006
6	nm0000007
7	nm0000008
8	nm0000009
9	nm0000010
10	nm0000012
11	nm0000013
12	nm0000014
13	nm0000015
14	nm0000017
15	nm0000018
16	nm0000020
17	nm0000021
18	nm0000022
19	nm0000024
20	nm0000027

In this problem, one part extracts all the actors which have acted in a movie whereas the other part extracts all the actors who have acted in a TV series. For the second part, we take the union of all the actors who have acted in any episode or in any series. This is done to handle the case where acting info of a person is given with respect to an episode but not with the series as a whole. Finally, we take the intersection of both these sets.

14.

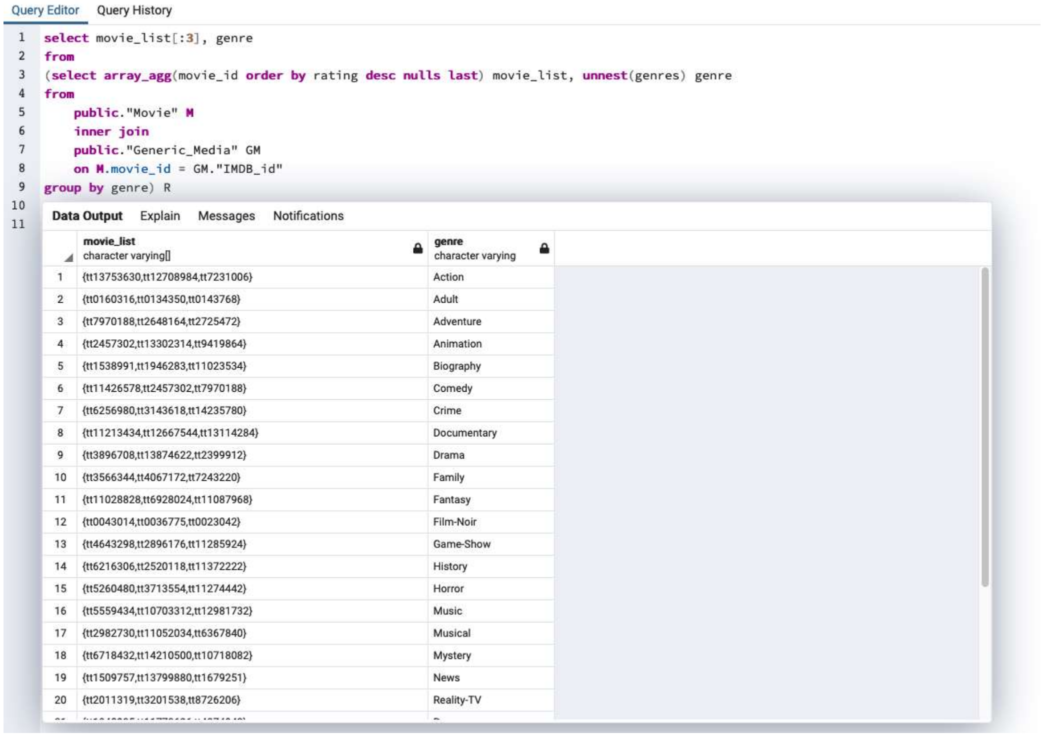


```
select eps_list[1], start_year
from
(select array_agg(episode_id order by runtime asc nulls last) eps_list, start_year
from public."Episodes"
group by start_year) r
```

	eps_list character varying	start_year bigint
28	tt12901302	1942
29	tt12967424	1943
30	tt14147852	1944
31	tt14181318	1945
32	tt13684992	1946
33	tt14181406	1947
34	tt14181452	1948
35	tt14181462	1949
36	tt0853404	1950
37	tt14181504	1951
38	tt14181528	1952
39	tt14181566	1953
40	tt6896410	1954
41	tt3297238	1955
42	tt13446302	1956
43	tt2394824	1957
44	tt13021888	1958
45	tt2217497	1959
46	tt0768677	1960
47	tt7285606	1961
48	tt6568396	1962

In this question we use the order by clause along with the array aggregation function. We group the episodes according to their air year and aggregate all the episodes ordered by their runtime. For each year we get an array and we finally select the first element of this array.

15.

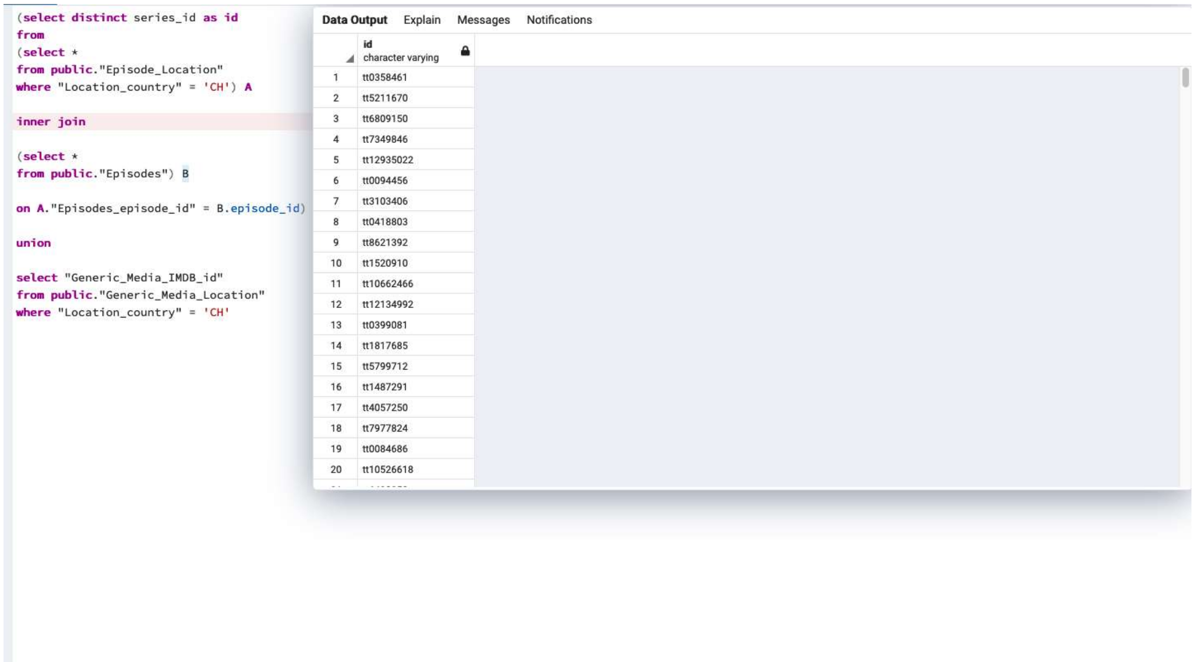


```
select movie_list[:3], genre
from
(select array_agg(movie_id order by rating desc nulls last) movie_list, unnest(genres) genre
from
    public."Movie" M
    inner join
    public."Generic_Media" GM
    on M.movie_id = GM."IMDB_id"
group by genre) R
```

	movie_list character varying[]	genre character varying
1	{tt13753630,tt12708984,tt7231006}	Action
2	{tt0160316,tt0134350,tt0143768}	Adult
3	{tt7970188,tt2648164,tt2725472}	Adventure
4	{tt2457302,tt13302314,tt9419864}	Animation
5	{tt1538991,tt1946283,tt11023534}	Biography
6	{tt11426578,tt2457302,tt7970188}	Comedy
7	{tt6256980,tt3143618,tt14235780}	Crime
8	{tt11213434,tt12667544,tt13114284}	Documentary
9	{tt3896708,tt13874622,tt2399912}	Drama
10	{tt3566344,tt4067172,tt7243220}	Family
11	{tt11028828,tt6928024,tt11087968}	Fantasy
12	{tt0043014,tt0036775,tt0023042}	Film-Noir
13	{tt4643298,tt2896176,tt11285924}	Game-Show
14	{tt6216306,tt2520118,tt11372222}	History
15	{tt5260480,tt3713554,tt11274442}	Horror
16	{tt5559434,tt10703312,tt12981732}	Music
17	{tt2982730,tt11052034,tt6367840}	Musical
18	{tt6718432,tt14210500,tt10718082}	Mystery
19	{tt1509757,tt13799880,tt1679251}	News
20	{tt2011319,tt3201538,tt8726206}	Reality-TV

Similar to the last question, we use the array aggregation function along with the order by clause. We group the movies by genre and then aggregate them ordered by their rating. We then select the first 3 values from each array corresponding to the top 3 movies from each genre.

16.

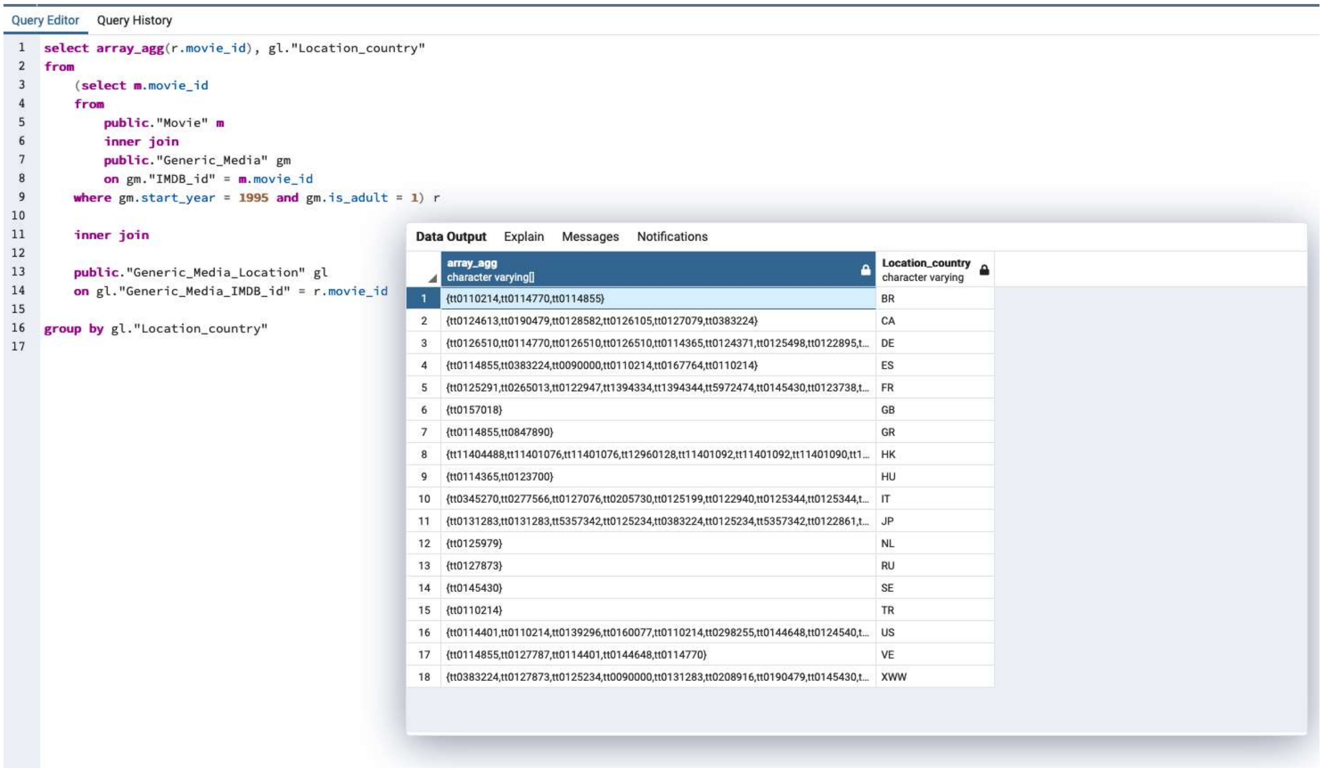


```
(select distinct series_id as id
from
(select *
from public."Episode_Location"
where "Location_country" = 'CH') A

inner join

(select *
from public."Episodes") B

on A."Episodes_episode_id" = B.episode_id)

union

select "Generic_Media_IMDB_id"
from public."Generic_Media_Location"
where "Location_country" = 'CH'
```

	id character varying
1	tt0358461
2	tt5211670
3	tt6809150
4	tt7349846
5	tt12935022
6	tt0094456
7	tt3103406
8	tt0418803
9	tt8621392
10	tt1520910
11	tt10662466
12	tt12134992
13	tt0399081
14	tt1817685
15	tt5799712
16	tt1487291
17	tt4057250
18	tt7977824
19	tt0084686
20	tt10526618

We select the series which have been filmed in Switzerland by first selecting the episodes filmed in Switzerland and then selecting their parent series. We then select all the Generic Media that have been filmed in Switzerland. Our Generic Media table contains only information about TV series and movies. We then take the union of these two sets to get the final result.

17.



```sql
select array_agg(r.movie_id), gl."Location_country"
from
    (select m.movie_id
    from
        public."Movie" m
        inner join
        public."Generic_Media" gm
        on gm."IMDB_id" = m.movie_id
    where gm.start_year = 1995 and gm.is_adult = 1) r

    inner join

    public."Generic_Media_Location" gl
    on gl."Generic_Media_IMDB_id" = r.movie_id

group by gl."Location_country"
```

	array_agg character varying[]	Location_country character varying
1	{tt0110214,tt0114770,tt0114855}	BR
2	{tt0124613,tt0190479,tt0128582,tt0126105,tt0127079,tt0383224}	CA
3	{tt0126510,tt0114770,tt0126510,tt0126510,tt0114365,tt0124371,tt0125498,tt0122895,t...	DE
4	{tt0114855,tt0383224,tt0090000,tt0110214,tt0167764,tt0110214}	ES
5	{tt0125291,tt0265013,tt0122947,tt1394334,tt1394344,tt5972474,tt0145430,tt0123738,t...	FR
6	{tt0157018}	GB
7	{tt0114855,tt0847890}	GR
8	{tt11404488,tt11401076,tt11401076,tt12960128,tt11401092,tt11401092,tt11401090,tt1...	HK
9	{tt0114365,tt0123700}	HU
10	{tt0345270,tt0277566,tt0127076,tt0205730,tt0125199,tt0122940,tt0125344,tt0125344,t...	IT
11	{tt0131283,tt0131283,tt5357342,tt0125234,tt0383224,tt0125234,tt5357342,tt0122861,t...	JP
12	{tt0125979}	NL
13	{tt0127873}	RU
14	{tt0145430}	SE
15	{tt0110214}	TR
16	{tt0114401,tt0110214,tt0139296,tt0160077,tt0110214,tt0298255,tt0144648,tt0124540,t...	US
17	{tt0114855,tt0127787,tt0114401,tt0144648,tt0114770}	VE
18	{tt0383224,tt0127873,tt0125234,tt0090000,tt0131283,tt0208916,tt0190479,tt0145430,t...	XWW

We first filtered out the adult movies of 1995 , followed by gathering the corresponding location information to finally aggregate these into sets grouped by their country of release as requested

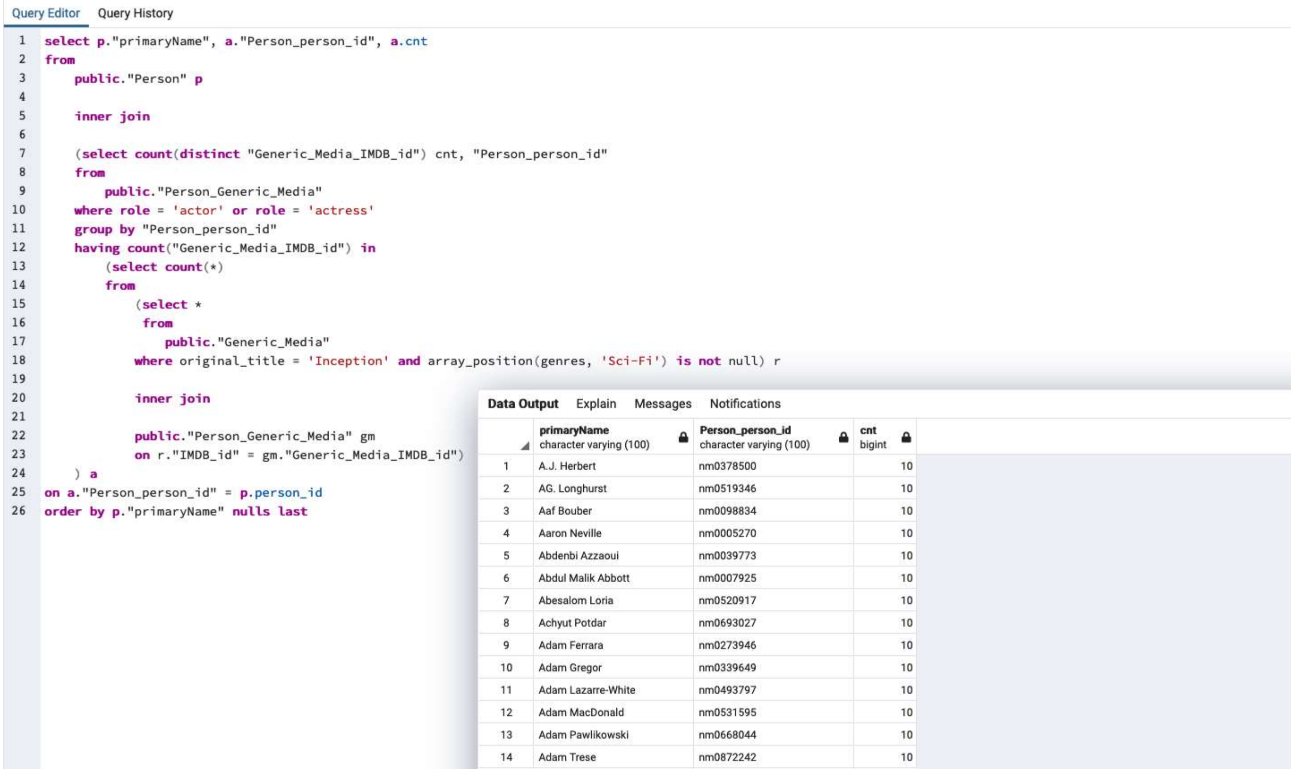
18.

We aggregated the Person_Ids w.r.t the separated (via unnest) professions, keeping the requested order in mind, in the same subquery which were then used to extract the youngest person by accessing the first index of the accumulated array.

19.

Firstly, we extract the music producer for each IMDB title. We then filter out the titles which are not movies. Finally, we group by the producer and count the number of people who have worked in and filter out the ones who have worked in more than 5 movies.

20.



```sql
select p."primaryName", a."Person_person_id", a.cnt
from
    public."Person" p

    inner join

    (select count(distinct "Generic_Media_IMDB_id") cnt, "Person_person_id"
    from
        public."Person_Generic_Media"
    where role = 'actor' or role = 'actress'
    group by "Person_person_id"
    having count("Generic_Media_IMDB_id") in
        (select count(*)
        from
            (select *
             from
                public."Generic_Media"
            where original_title = 'Inception' and array_position(genres, 'Sci-Fi') is not null) r

            inner join

            public."Person_Generic_Media" gm
            on r."IMDB_id" = gm."Generic_Media_IMDB_id")
    ) a
on a."Person_person_id" = p.person_id
order by p."primaryName" nulls last
```

	primaryName character varying (100)	Person_person_id character varying (100)	cnt bigint
1	A.J. Herbert	nm0378500	10
2	AG. Longhurst	nm0519346	10
3	Aaf Bouber	nm0098834	10
4	Aaron Neville	nm0005270	10
5	Abdenbi Azzaoui	nm0039773	10
6	Abdul Malik Abbott	nm0007925	10
7	Abesalom Loria	nm0520917	10
8	Achyut Potdar	nm0693027	10
9	Adam Ferrara	nm0273946	10
10	Adam Gregor	nm0339649	10
11	Adam Lazarre-White	nm0493797	10
12	Adam MacDonald	nm0531595	10
13	Adam Pawlikowski	nm0668044	10
14	Adam Trese	nm0872242	10

We chose the movie Inception(2010) directed by C Nolan. We needed the Sci-Fi filter as there is another movie under the same name (directed by Danial Hajibarat (2014) et al). We then extracted the crew strength of that movie and queried for the actor that has worked in these many movies.