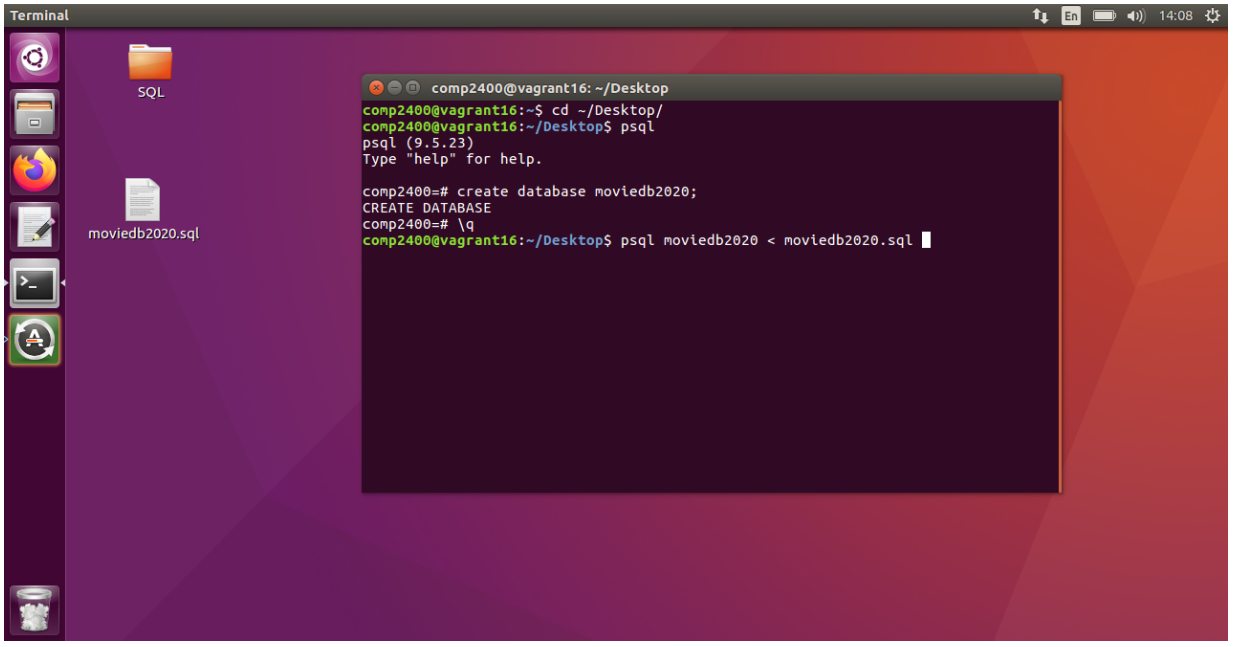
COMP6240 - Relational Databases

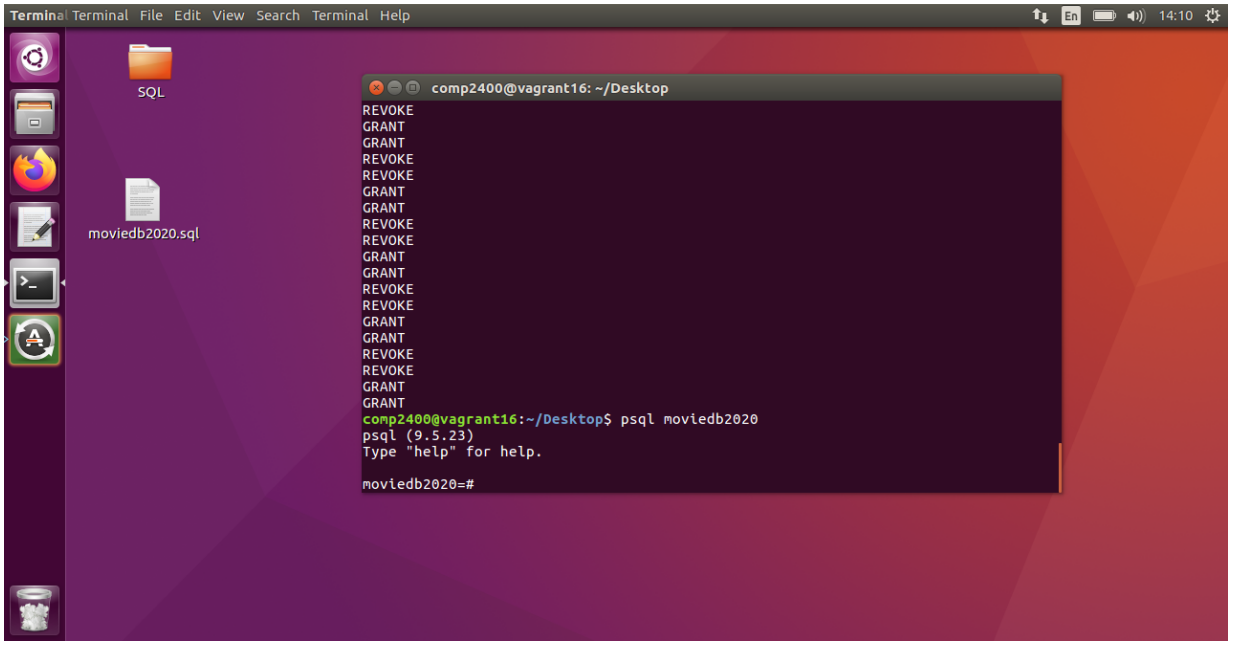
Assignment 1 (SQL) (Solutions)

Please use the `moviedb2020` database to check your query against the following sample solutions. The `moviedb2020` database is available on the ANU RSCS virtual environment (Option 2). You should first open a command shell and then connect to the `moviedb2020` database by entering

```
psql moviedb2020
```

If you are using the COMP2400 virtual machine (Option 1), you can download a copy of the `moviedb2020` database from the sample solution folder on Wattle and set up this database on the COMP2400 virtual machine. Download and save `moviedb2020.sql` file on the Desktop and refer to the following screenshots for instructions.





Question 1 **15 Marks**

The relational database `moviedb` has the following database schema:

MOVIE(title, production_year, country, run_time, major_genre)
`primary key` : {title, production_year}

PERSON(id, first_name, last_name, year_born)
`primary key` : {id}

AWARD(award_name, institution, country)
`primary key` : {award_name}

RESTRICTION_CATEGORY(description, country)
`primary key` : {description, country}

DIRECTOR(id, title, production_year)
`primary key` : {title, production_year}
`foreign keys` : [title, production_year] ⊆ MOVIE[title, production_year]
[id] ⊆ PERSON[id]

WRITER(id, title, production_year, credits)
`primary key` : {id, title, production_year}
`foreign keys` : [title, production_year] ⊆ MOVIE[title, production_year]
[id] ⊆ PERSON[id]

CREW(id, title, production_year, contribution)
`primary key` : {id, title, production_year}
`foreign keys` : [title, production_year] ⊆ MOVIE[title, production_year]
[id] ⊆ PERSON[id]

SCENE(title, production_year, scene_no, description)
`primary key` : {title, production_year, scene_no}
`foreign keys` : [title, production_year] ⊆ MOVIE[title, production_year]

ROLE(id, title, production_year, description, credits)
`primary key` : {title, production_year, description}
`foreign keys` : [title, production_year] ⊆ MOVIE[title, production_year]
[id] ⊆ PERSON[id]

RESTRICTION(title, production_year, description, country)
`primary key` : {title, production_year, description, country}
`foreign keys` : [title, production_year] ⊆ MOVIE[title, production_year]
[description, country] ⊆ RESTRICTION_CATEGORY[description, country]

APPEARANCE(title, production_year, description, scene_no)
`primary key` : {title, production_year, description, scene_no}
`foreign keys` : [title, production_year, scene_no]⊆SCENE[title, production_year, scene_no]
[title, production_year, description]⊆ROLE[title, production_year, description]

MOVIE_AWARD(title, production_year, award_name, year_of_award,category, result)if the
`primary key` : {title, production_year, award_name, year_of_award, category}
`foreign keys` : [title, production_year] ⊆ MOVIE[title, production_year]
[award_name] ⊆ AWARD[award_name]

CREW_AWARD(id, title, production_year, award_name, year_of_award, category, result)
`primary key` : {id, title, production_year, award_name, year_of_award, category}
`foreign keys` : [id, title, production_year] ⊆ CREW[id, title, production_year]
[award_name] ⊆ AWARD[award_name]

DIRECTOR_AWARD(title, production_year, award_name, year_of_award, category, result)
`primary key` : {title, production_year, award_name, year_of_award, category}
`foreign keys` : [title, production_year] ⊆ DIRECTOR[title, production_year]
[award_name] ⊆ AWARD[award_name]

WRITER_AWARD(id, title, production_year, award_name, year_of_award, category, result)
`primary key` : {id, title, production_year, award_name, year_of_award, category}
`foreign keys` : [id, title, production_year] ⊆ WRITER[id, title, production_year]
[award_name] ⊆ AWARD[award_name]

ACTOR_AWARD(title, production_year, description, award_name, year_of_award,category,result)
`primary key` : {title, production_year, description, award_name, year_of_award, category}
`foreign keys` : [award_name] ⊆ AWARD[award_name]
[title,production_year,description]⊆ROLE[title,production_year,description]

There are five different categories of awards: movie awards, crew awards, director awards, writer awards and actor awards. A movie can only win an award after being nominated for the award.

Please refer to the sample solutions to the following questions as well as the common issues summarized during our marking.

1.1 How many *comedy* movies (i.e., the major genre of the movie is comedy) were produced in 1994? List that number. (1.5 Mark)

```
SELECT COUNT(*)
FROM movie
WHERE production_year = 1994 and major_genre = 'comedy';
```

Common issue 1: Forgot to check whether the production_year is 1994.

Common issue 2: Forgot to check whether the major_genre is comedy.

1.2 How many directors have directed at least one movie written by themselves? List that number. (1.5 Mark)

```
SELECT COUNT(DISTINCT id)
FROM director NATURAL JOIN writer;
```

Common issue 1: Compared only the id and title (without production_year) when using Cartesian product or INNER JOIN to join director and writer. You need to compare id, title and production_year. Two different movies may have the same title but different production years. Hence, the primary key of the movie table is {title, production_year}.
Example: Consider that director with id '00000621' has directed the movie titled 'Gladiator' in year 2002 and has written the movie titled 'Gladiator' in year 2000. If you compare only the id and title when using Cartesian product or INNER JOIN to join director and writer, then you will get director 00000621 in the final result which is incorrect. This is because there are two different movies with the same title ('Gladiator') but produced in two different years (2000 and 2002).

Common issue 2: Forgot to remove the duplicate IDs in the result.
Example: There can be directors who have directed more than one movies written by themselves. For example, director with id '0000001' has directed and written the movies 'Titanic' (1997) and 'Aliens'

(1986). Similarly, director with id '00000841' has directed and written the movies 'The Sixth Sense' (1999) and 'Unbreakable' (2000). In such cases, if we do not remove the duplicate ids (using DISTINCT), the count will be incorrect.

1.3 Who played two or more roles in the same movie? List their ids, the titles and production years of the corresponding movies. (1.5 Mark)

```
SELECT id, title, production_year
FROM role
GROUP BY id, title, production_year
HAVING COUNT(*)>1;
```

Common issue 1: Use of only the id and title (without production_year) in GROUP BY.
Example: There are two different movies 'Gladiator' (produced in 2000) and 'Gladiator' (produced in 2002). They have the same title but different production years. The person with id '00000623' has played a role in both the movies 'Gladiator' (2000) and 'Gladiator' (2002). If you GROUP BY only the id and title (without grouping by id, title and production year), the id '00000623' and title 'Gladiator' will get a count of 2 and will be included in the result. This is incorrect as those are two different movies even though they have the same title.

Common issue 2: Forgot to include the HAVING clause with COUNT(*)>1. "Two or more" means that the count should be greater than 1 (OR greater than or equal to 2 (>= 2)).

1.4 Which movies had the 'PG' restriction in at least two countries? List their titles, production years and the corresponding number of countries with the 'PG' restriction. (1.5 Mark)

```
SELECT title, production_year, COUNT(*)
FROM restriction
WHERE description='PG'
GROUP BY title, production_year
HAVING COUNT(*)>1;
```

Common issue 1: Use of only the title (without production_year) in GROUP BY.
Example: There are two movies 'The Birds' 1963 and 'The Birds' 1966. They have the same title but different production years. The movie 'The Birds' 1963 has a PG restriction in Australia whereas the the movie 'The Birds' 1966 has a PG restriction in New Zealand. If you GROUP BY only the title (without grouping by both title and production year), the title 'The Birds' will get a count of 2 and will be included in the result. This is wrong as those are two different movies even though they have the same title.

Common issue 2: Forgot to include the HAVING clause with COUNT(*)>1. "At least two" means that the count should be greater than 1 (OR greater than or equal to 2 (>= 2)).

1.5 Who have written exactly two American movies (*i.e.*, the production country is USA)? List their ids, first and last names. Order your results in the ascending order of their ids. (1.5 Mark)

```
SELECT id, first_name, last_name
FROM writer NATURAL JOIN movie NATURAL JOIN person
WHERE lower(country) = 'usa'
GROUP BY id,  first_name, last_name
HAVING COUNT(*) = 2
ORDER BY id ASC;
```

Common issue 1: Compared only the title (without production_year) when using Cartesian product or INNER JOIN to join writer and movie. You need to compare both title and production_year. Two different movies may have the same title but different production years. Hence, the primary key of the movie table is {title, production_year}.
Example: Consider the writers Joseph Stefano and Robert Bloch. They both have written the two American movies titled 'Psycho' (1960) and 'Psycho' (1998). These two movies have the same title but different production years. If you compare only the title (without comparing both title and production_year) when using Cartesian product or INNER JOIN to join writer and movie, then it will result in 4 records each for Joseph Stefano and Robert Bloch. Hence, they will not be considered in the result as their count (equal to 4) is not equal to 2.

Common issue 2: Forgot to include the HAVING clause with COUNT(*)=2. "Exactly two" means the count should be equal to 2.

Common issue 3: Forgot to order the final results in the ascending order of the ids.

1.6 How many directors have never played any roles in movies directed by themselves? List that number. (1.5 Mark)

```
SELECT COUNT(*)
FROM (SELECT id
      FROM director
      EXCEPT
      SELECT id
      FROM director NATURAL JOIN role) dna;
```

Common issue 1: Forgot to join director and role in the SELECT statement after EXCEPT, to get the directors who have played roles in movies directed by themselves.
Example: There are can be directors who have directed movies, but have played roles in other movies. For example, director with id '00001038' has directed the movie 'Exotica' (1994) and has played a role in the movie 'Alien 3' (1992). Similarly, director with id '00001051' has directed the movies 'Gandhi' (1982) and 'Chaplin' (1992) and has played a role in the movie 'Jurassic Park' (1993). If you do not join director and role in the SELECT statement after EXCEPT, then the id of such directors will be removed from the final result as they have played roles in other movies.

Common issue 2: Compared only the id and title when using Cartesian product or INNER JOIN to join director and role. You need to compare id, title and production_year. Two different movies may have the same title but different production years. Hence, the primary key of the movie table is {title, production_year}.
Example: The director with id '00000621' has directed the movie 'Gladiator' produced in year 2002 and has played a role in the movie 'Gladiator' produced in year 2000. If you compare only the id and title (without comparing id, title and production_year) when using Cartesian product or INNER JOIN to join director and role, then director '00000621' will be included in the result of directors who have directed and played roles in the same movie. This is incorrect as director '00000621' has directed and played roles in two different movies, even though their titles are the same.

1.7 Who won at least one director award and at least one writer award in the same year? List their ids and the corresponding year of award. (1.5 Mark)

```
SELECT id, year_of_award
FROM director_award NATURAL JOIN director
WHERE lower(result)='won'
INTERSECT
SELECT id, year_of_award
```

```
FROM writer_award
WHERE lower(result)='won';
```

Common issue 1: Compared only the id and title (without production_year) when using Cartesian product or INNER JOIN to join director and role. You should compare id, title and production_year. Two different movies may have the same title but different production years.

Common issue 2: Forgot to check whether the result is 'won' for the director awards or writer awards.

1.8 Which crew member(s) worked on the greatest number of movies? List their id(s), first and last names. (1.5 Mark)

```
WITH CrewNumMovie AS (SELECT id, COUNT(*) AS CrewNum
                      FROM crew
                      GROUP BY id)
SELECT id, first_name, last_name
FROM CrewNumMovie mc1 NATURAL JOIN person
WHERE mc1.CrewNum = (SELECT MAX(mc2.CrewNum)
                     FROM CrewNumMovie mc2);
```

Common issue 1: Use of id and title/production_year in GROUP BY. You need to consider the crew members and the number of different movies they worked on. Hence, you should GROUP BY only the id.

1.9 Who received the greatest number of nominations for a director award but never won? List their id(s). (1.5 Mark)

```
WITH DirectorNominationNum
AS (SELECT id, COUNT(*) AS NominationNum
    FROM director_award NATURAL JOIN director
    WHERE id IN (SELECT id
                 FROM director_award NATURAL JOIN director
                 WHERE lower(result) <> 'won'
                 EXCEPT
                 SELECT id
                 FROM director_award NATURAL JOIN director
                 WHERE lower(result) = 'won')
    GROUP BY id)
SELECT DNN1.id
FROM DirectorNominationNum DNN1
WHERE NominationNum IN (SELECT MAX(DNN2.NominationNum)
                FROM DirectorNominationNum DNN2);
```

Common issue 1: Checked whether the result is 'nominated' only. Please refer to the clarification in Week 5 Online Classroom. "Never won director awards" means "did not win any director awards at all". *Example*: Assume that director with id '00000621' has 3 nominations and won once, whereas directors with id '00001005' has 2 nominations and has never won. The correct answer should be director '00001005', not director '00000621' (even though director '00000621' has the maximum number of nominations).

Common issue 2: Try to fix **Common issue 1** by removing those writers who have won awards before. *Example*: Consider directors with ids '00000621' and '00001005'. Director '00000621' has been nominated for 3 times for director awards and has won once. Director '00001005' has been nominated 2 times for director awards but has never won any. However, if you straightaway take the the maximum nominations count as 3 (for director '00000621') and check for the directors who have never won, then you will get an empty result. The desired answer should be director '00001005'.

1.10 Find all the pairs of crew members who won a crew award at the same age. List the pairs of their ids. Note that the result should not contain duplicated pairs of ids, *e.g.*, (id_1, id_2) and (id_2, id_1) are considered as duplicated pairs and your query should only produce one of them in the result. Hint: if Emily (born in 1960) won a crew award in 1995 and Tom (born in 1955) won a crew award in 1990, they are considered as a pair of crew members who won a crew award at the same age (=35). (1.5 Mark)

```
WITH crew_age as (SELECT id, year_of_award - year_born AS age
                  FROM crew_award NATURAL JOIN person
                  WHERE lower(result) = 'won')
SELECT DISTINCT c1.id, c2.id
FROM crew_age c1 INNER JOIN crew_age c2
                 ON c1.id>c2.id AND c1.age = c2.age;
```

Common issue 1: Forgot to remove the duplicate pairs (id1, id2) in the result.
Example: Crew members can win multiple awards in the same year. For example, crew member with id '00000011' (born in year 1956) has won an Oscar award and an Eddie award in the year 1998. Crew member with id '00000009' (born in year 1956) has won an Oscar award in the year 1998. When you do the self join, then you will get 2 id pairs for '00000011' and '00000009'. You should remove the duplicate pairs.

Common issue 2: Forget to remove the duplicated pairs of ids such as (id_1, id_2) and (id_2, id_1). You should use the condition c1.id>c2.id instead of c1.id! =c2.id.
Example: When you do a self join, there can be duplicate pairs such as ('00000349', '00000207') and ('00000207', '00000349'). We should remove the duplicate ones and retain only one pair.

Common issue 3: Forget to check whether the result is 'won' for crew awards.

+++++