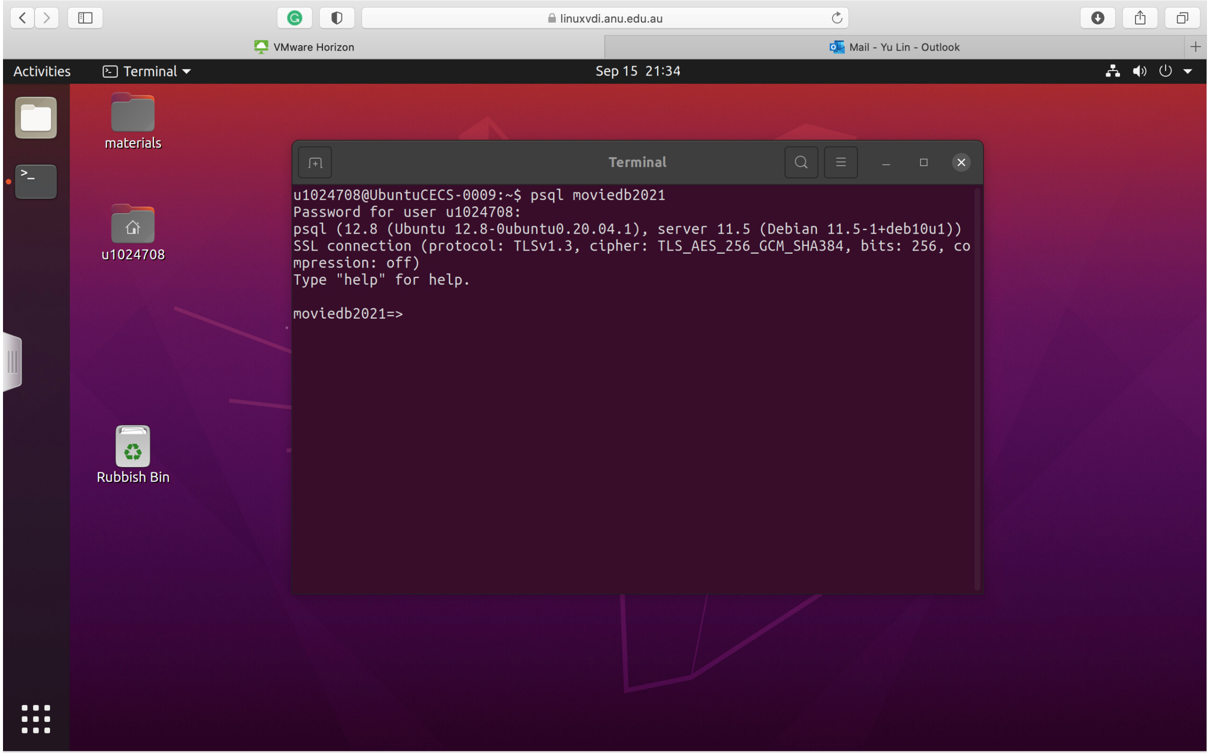
ANU School of Computing
COMP2400 Assignment 1 (SQL)
S2 2021

Sample Solutions and Common Issues

General Information:

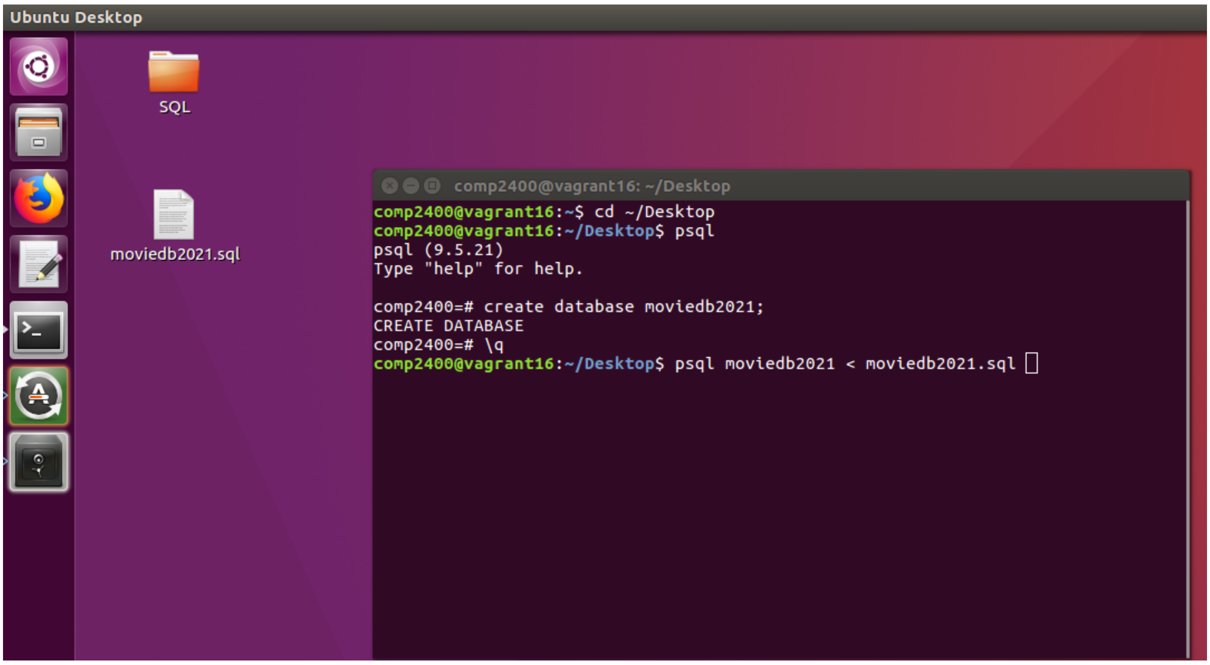
- To reveal common issues of incorrect solutions, we have provided an example database moviedb2021 database for you to compare your query against the following sample solutions. Note that the information in moviedb2021 does not correspond   movies in our real world.

- The moviedb2021 database is available on the ANU LinuxVDI (Option 1) and you can first open a terminal and then connect to the moviedb2021 database by entering "psql moviedb2021". (refer to the following screenshot)

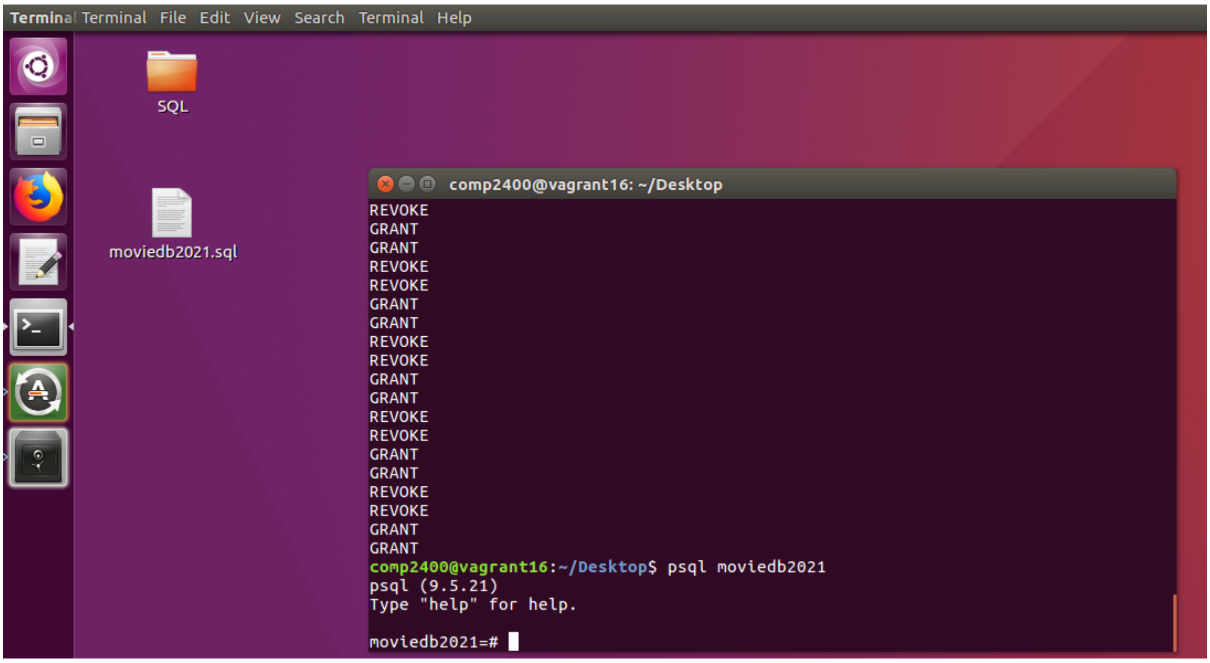


```
u1024708@UbuntuCECS-0009:~$ psql moviedb2021
Password for user u1024708:
psql (12.8 (Ubuntu 12.8-0ubuntu0.20.04.1), server 11.5 (Debian 11.5-1+deb10u1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, co
mpression: off)
Type "help" for help.

moviedb2021=>
```

- If you are using the local VM (Option 2), you can download a copy of the moviedb2021.sql file from the sample solution folder on Wattle and set up this database on your local VM. You can download Refer to the following screenshots for detailed instructions (assume that you have downloaded the moviedb2021.sql on the desktop).



```
comp2400@vagrant16:~$ cd ~/Desktop
comp2400@vagrant16:~/Desktop$ psql
psql (9.5.21)
Type "help" for help.

comp2400=# create database moviedb2021;
CREATE DATABASE
comp2400=# \q
comp2400@vagrant16:~/Desktop$ psql moviedb2021 < moviedb2021.sql
```



Terminal Terminal File Edit View Search Terminal Help
SQL
moviedb2021.sql
comp2400@vagrant16: ~/Desktop

```
REVOKE
GRANT
GRANT
REVOKE
REVOKE
GRANT
GRANT
REVOKE
REVOKE
GRANT
GRANT
REVOKE
REVOKE
GRANT
GRANT
REVOKE
REVOKE
GRANT
GRANT
comp2400@vagrant16:~/Desktop$ psql moviedb2021
psql (9.5.21)
Type "help" for help.

moviedb2021=#
```

- Please refer to the following sample solutions (with common issues) as well as the personalised feedback file (feedback.txt on Wattle). If you still need any further clarification or have any concerns about the marking, you are welcome to join any of the following drop-in sessions in Week 7:
    - Monday, September 20, 3-4 pm
    - Tuesday, September 21, 3-4 pm
    - Wednesday, September 22, 7-8 pm
    - Thursday, September 23, 3-4 pm
    - Friday, September 24, 3-4 pm

Q1. How many comedy movies (i.e., the major genre of the movie is comedy) are produced in Australia? List that number.

```
SELECT COUNT(*)
FROM movie
WHERE major_genre = 'comedy' AND country = 'Australia';
```

Common issue 1: Forgot to check whether the major genre is comedy.

Common issue 2: Forgot to check whether the country is Australia.

Q2. Which movies are classified as “S” in Finland? List their titles and production years and order them in the ascending order of their production years.

```
SELECT title, production_year
FROM restriction
WHERE description = 'S' AND country = 'Finland'
ORDER BY production_year ASC;
```

Common issue 1: Forgot to order the results in the ascending order of the production year.

Common issue 2: Forgot to check the conditions of description and country. Note that we should refer to country in the restriction table, but not the movie table.

Q3. Which countries have at least three restriction categories? List these countries along with the corresponding number of restriction categories.

```
SELECT country, COUNT(*)
FROM restriction_category
GROUP BY country
HAVING COUNT(*) > 2;
```

Common issue 1: Forgot to include the HAVING clause with COUNT(*) > 2. “At least three” means that the count should be greater than or equal to 3 (≥ 3).

Common issue 2: Used the wrong table (“restriction”) in the query.

Q4. Which directors have directed both drama and romance movies? List their ids.

```
SELECT id
FROM director NATURAL JOIN movie
WHERE major_genre = 'drama'
INTERSECT
SELECT id
FROM director NATURAL JOIN movie
WHERE major_genre = 'romance';
```

Common issue 1: Forgot to check whether the major genre of movie is drama or romance.

Common issue 2: Used “UNION” or, used “OR” in the query. In this question, “both drama and romance” indicates that “INTERSECT” should be used in correct query.

Common issue 3: Compared only the title (without production year) when using Cartesian product or INNER JOIN to join director and movie. You need make sure that both title and production year match in this case. Two different movies may have the same title but different production years. Note that the primary key of the movie table is {title, production year}.

Q5. Which movies have won at least one director or writer award? List their titles and production years.

```
SELECT title, production_year FROM director_award
WHERE result = 'won'
UNION
SELECT title, production_year FROM writer_award
WHERE result = 'won';
```

Common issue 1: Forgot to check whether the result of director award and writer award is “won”.

Common issue 2: Used “INTERSECT” in the query. In this question, “one director or writer award” indicates that “UNION” or ”OR” should be used in the query.

Q6. For all movies produced in Germany, list their title, production year and the age of the director when the movie was produced. (Hint: if a movie is produced in 1994 and its director was born in 1940, then the age of the director when this movie was produced is 1994 - 1940 = 54)

```
SELECT title, production_year, production_year-year_born AS age
FROM movie NATURAL JOIN director NATURAL JOIN person
WHERE country='Germany';
```

Common issue 1: Compared only the title (without production year) when using Cartesian product or INNER JOIN to join director and movie. You need make sure that both title and production year match in this case. Note that the primary key of the movie table is {title, production year}. Hence, two different movies may have the same title but different production years.

Example: The director “00000441” has directed a movie Traffic in year 2000. The director “00000321” has directed a different movie Traffic in year 2001. These two Germany movies have the same title but different production years and different directors. If you compare only the title (without comparing both title and production year) when using Cartesian product or INNER JOIN to join director and movie, then it will result in 4 records for these two movies.

Q7. What is the largest number of crew members working in a single movie? List that number.

```
SELECT MAX(cnum)
FROM (SELECT title, production_year, COUNT(*) AS cnum
      FROM crew
      GROUP BY title, production_year) AS count;
```

Common issue 1: GROUP BY only the title (without production year). You need to GROUP BY both title and production year. The primary key of the movie table is {title, production year}. Hence, two different movies may have the same title but different production years.

Example: There are two movies “Titanic” 1997 and “Titanic” 1998. They have the same title but different production years. The movie “Titanic” 1997 has 9 crew members, whereas the movie “Titanic” 1998 has only 1 crew member. If you GROUP BY only the title (without grouping by both title and production year), the title “Titanic” will get a count of 10. This is wrong as those are two different movies even though they have the same title.

Q8. Which directors have collaborated with at least two different writers? List the ids, first and last names of these directors, and order them in the ascending order of their last names. (Hint: a director is considered to collaborate with a writer if the director directed a movie written by the writer).

```
WITH d2w AS (
   SELECT d.id
   FROM director d INNER JOIN writer w
   ON d.title = w.title AND d.production_year = w.production_year
        AND d.id != w.id
   GROUP BY d.id
   HAVING COUNT(DISTINCT w.id) > 1)
SELECT id, first_name, last_name
FROM person NATURAL JOIN d2w
ORDER BY last_name ASC;
```

Common issue 1: Forgot to remove duplicates in HAVING COUNT(w.id). In the question, “two different writers” means that if a director collaborated with the same writer twice, then this writer should only be counted once. Hence, we need to use HAVING COUNT(DISTINCT w.id) to remove the duplicate writers.

Example: The director “00000021” work with writer “00000022” on two movies: “Shakespeare in Love (1998)” and “Shakespeare in Love (1999)”. Hence, the writer “00000022” appear twice when GROUP BY d.id. We need to remove the duplicated writers by using COUNT(DISTINCT w.id).

Common issue 2: Forgot to include the condition to make sure two different writers do not include the director themselves (Refer to the clarification in the workshop in Week 5. e.g., missing a condition like “director.id != writer.id”).

Common issue 3: Compared only the title (without production year) when using Cartesian product or INNER JOIN to join director and writer. You need make sure that both title and production year match in this case. The primary key of the movie table is {title, production year}. Hence, two different movies may have the same title but different production years.

Q9. Among those directors who have never won any director award, who directed the largest number of movies? List their ids.

```
WITH never_won AS (
    SELECT DISTINCT id
    FROM director
    EXCEPT
    SELECT DISTINCT id
    FROM director_award NATURAL JOIN director
    WHERE lower(result)='won'),
never_won_count AS (
     SELECT count(*) AS num, id
     FROM never_won NATURAL JOIN director
     GROUP BY id)
SELECT id
FROM never_won_count
WHERE num=(SELECT max(num)
           FROM never_won_count);
```

[The intuition behind the above query]: You need to discard those directors who have won director award by using "EXCEPT". Now we have the directors who never won any director award in table "never_won". We then count how many movies they have directed in table "never_won_count". And, at last, select the director(s) who have directed the largest number of movies from in table "never_won_count".

Common issue 1: Here is a typical incorrect query:

```
WITH dnwc AS (
   SELECT id, COUNT(*) AS num
   FROM director d
   WHERE NOT EXISTS (SELECT *
                     FROM director_award da
                     WHERE d.title = da.title
                     AND d.production_year = da.production_year
                     AND lower(result) = 'won')
   GROUP BY id)
SELECT id
FROM dnwc
WHERE num = (SELECT max(num)
             FROM dnwc);
```

The above incorrect query failed to filter some directors who has won some director award. For example, the director "00001005" has won director award in "Bullets Over Broad- way (1994)". This same director "00001005" has also directed three other movies ("Bullets Over Broadway (1995)"; "Alice (1990)"; "Manhattan Murder Mystery (1993)"), but none of these three movies won any director award. According to the question description, "00001005" has won the director award once and thus should not be included in the output. But the above incorrect query fails to filter director “00001005” and may include this director in the final result.

Common issue 2: Compared only the title (without production year) when using Cartesian product or INNER JOIN to join director and director award. You need make sure that both title and production year match in this case. The primary key of the movie table is {title, production year}. Hence, two different movies may have the same title but different production years.

Example: “Fight Club (2000)” is directed by director “00000222”. “00000222” won director award in year 2001. “Fight Club (1999)” is directed by director “00000401”. “00000401” never won any director award. However, if we only compared the title (without production year) when using Cartesian product or INNER JOIN to join director and director award. “00000401” will be wrongly classified as those who have won a director award because of the same title of the two different movies “Fight Club (2000)” and “Fight Club (1999)”.

Q10. How many writers have never co-written movies with other writers? List that number. (Hint: a movie may have multiple writers but every movie written by such a writer has only one writer)

```
WITH multi_writer_movies AS (
        SELECT title, production_year
        FROM writer
        GROUP BY title, production_year
        HAVING count(*)>1)
SELECT count(*)
FROM (SELECT id
          FROM writer
          EXCEPT
          SELECT id
          FROM multi_writer_movies NATURAL JOIN writer) AS temp;
```

[The intuition behind the above query]: You need to first list those movies with multiple writers in table “multi_writer_movies”. Then we use set operation “EXCEPT” to find out the writers who have never co-written any movies in table “multi_writer_movies”.

Common issue 1: Here is a typical incorrect query (resulting a total count of 82 instead of 80 on moviedb2021)

```
WITH mwm AS (
   SELECT title, production_year
   FROM writer
   GROUP BY title, production_year
   HAVING count(*) > 1)
SELECT COUNT (DISTINCT id)
FROM writer w
WHERE NOT EXISTS (SELECT * FROM mwm
              WHERE w.title = mwm.title
              AND w.production_year = mwm.production_year);
```

```
-- A Toy Counter example
-- Table name : writer
--    id    |   title   | production_year | credits
------------+-----------+-----------------+---------
-- 00000001 | Titanic   | 1997 |
-- 00000001 | Aliens    | 1986 |
-- 00000001 | Gladiator | 2002 |
-- 00000542 | Gladiator | 2002 |
-- 00000542 | Alien     | 1979 |
```

It is clear from the above toy table that “00000001” and “00000542” co-write on Gladiator (2002). Hence the correct output for the above toy example should be 0. But the above incorrect query will output 2. The incorrect logic is similar to the Common Issue 1 in Q1.9. In short, this incorrect query failed to filter some writers who have co-written movies with other writers.

Common issue 2: GROUP BY only the title (without production year). You need to GROUP BY both title and production year. The primary key of the movie table is {title, production year}. Hence, two different movies may have the same title but different production years.