Principles of Database Systems
Assignment #4 - Structured Query Language 2

He Tianyang, 3022001441

October 9, 2024

1 Execute the SQL in Slides

1.1 Preparation

1.1.1 Create Table

```sql
CREATE TABLE movieexec
(
    name     CHAR(30),
    address  VARCHAR(255),
    cert     INT,
    networth INT
);
CREATE TABLE moviestar
(
    name      CHAR(30),
    address   VARCHAR(255),
    gender    CHAR(1),
    birthdate date
);
CREATE TABLE starsin
(
    movietitle CHAR(100),
    movieyear  INT,
    starname   CHAR(30)
);
CREATE TABLE Movies
(
    title      CHAR(100),
    year       INT,
```

```
    length      INT,
    genre       CHAR(10),
    studioName  CHAR(30),
    producerC   INT,
    PRIMARY KEY (title, year)
);
```

The execution results are shown in the Fig. 1.

Figure 1: Create Tables

1.1.2 Insert Sample Data `movies`

```
INSERT INTO movies VALUES ('Logan''s run', 1976, NULL, 'sciFi', 'MGM',
   123);
INSERT INTO movies VALUES ('Star Wars', 1977, 124, 'sciFi', 'Fox', 555);
INSERT INTO movies VALUES ('Empire Strikes Back', 1980, 111, 'fantasy',
   'Fox', 555);
INSERT INTO movies VALUES ('Star Trek', 1979, 132, 'sciFi', 'Paramount',
    345);
INSERT INTO movies VALUES ('Star Trek: Nemesis', 2002, 116, 'sciFi', '
   Paramount', 345);
INSERT INTO movies VALUES ('Terms of Endearment', 1983, 132, 'romance',
   'MGM', 123);
INSERT INTO movies VALUES ('The Usual Suspects', 1995, 106, 'crime', '
   MGM', 456);
INSERT INTO movies VALUES ('Gone With the Wind', 1938, 238, 'drama', '
   MGM', 123);
INSERT INTO movies VALUES ('Wayne''s World', 1992, 95, 'comedy', '
   Paramount', 123);
```

```sql
INSERT INTO movies VALUES ('King Kong', 2005, 187, 'drama', 'Universal',
    789);
INSERT INTO movies VALUES ('King Kong', 1976, 134, 'drama', 'Paramount',
    666);
INSERT INTO movies VALUES ('King Kong', 1933, 100, 'drama', 'Universal',
    345);
INSERT INTO movies VALUES ('Pretty Woman', 1990, 119, 'comedy', 'Disney'
   , 999);
```

the execution results are shown in the Fig. 2.

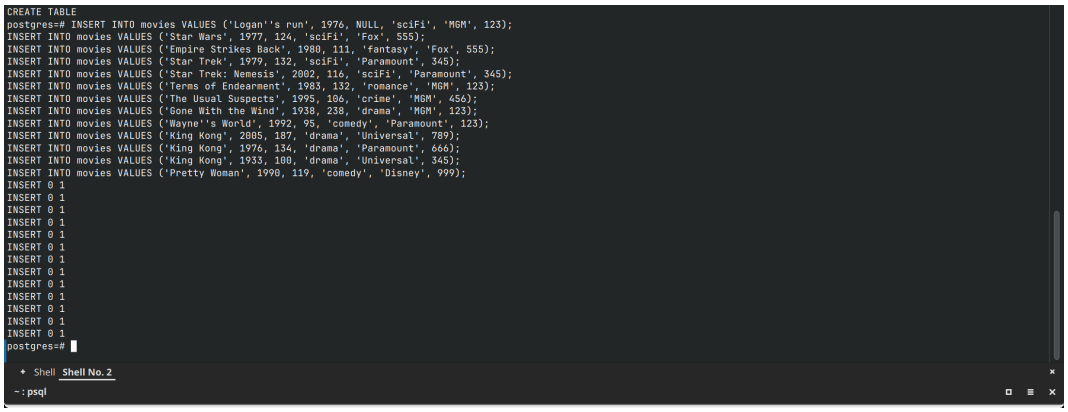


Figure 2: Insert Movies

1.1.3 Insert Sample Data `movieexec`

```sql
INSERT INTO movieexec VALUES ('George Lucas', 'Oak Rd.', 555, 200000000)
   ;
INSERT INTO movieexec VALUES ('Ted Turner', 'Turner Av.', 333,
   125000000);
INSERT INTO movieexec VALUES ('Stephen Spielberg', '123 ET road', 222,
   100000000);
INSERT INTO movieexec VALUES ('Merv Griffin', 'Riot Rd.', 199,
   112000000);
INSERT INTO movieexec VALUES ('Calvin Coolidge', 'Fast Lane', 123,
   20000000);
INSERT INTO movieexec VALUES ('Garry Marshall', 'First Street', 999,
   50000000);
INSERT INTO movieexec VALUES ('J.J. Abrams', 'High Road', 345, 45000000)
   ;
INSERT INTO movieexec VALUES ('Bryan Singer', 'Downtown', 456, 70000000)
   ;
INSERT INTO movieexec VALUES ('George Roy Hill', 'Baldwin Av.', 789,
   20000000);
INSERT INTO movieexec VALUES ('Dino De Laurentiis', ' Beverly Hills',
   666, 120000000);
```

```sql
INSERT INTO movieexec VALUES ('AAA', ' Beverly Hills', 666, 120000000);
```

the execution results are shown in the Fig. 3.

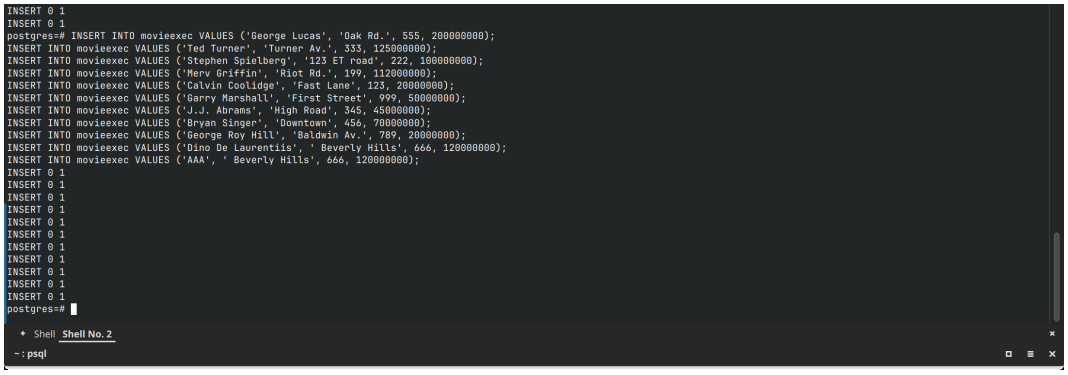


Figure 3: Insert Movie Executives

1.1.4 Insert Sample Data `moviestar`

```sql
INSERT INTO moviestar VALUES ('Jane Fonda', 'Turner Av.', 'F', '
   1977-07-07');
INSERT INTO moviestar VALUES ('Alec Baldwin', 'Baldwin Av.', 'M', '
   1977-06-07');
INSERT INTO moviestar VALUES ('Kim Basinger', 'Baldwin Av.', 'F', '
   1979-05-07');
INSERT INTO moviestar VALUES ('Harrison Ford', 'Beverly Hills', 'M', '
   1977-07-07');
INSERT INTO moviestar VALUES ('Carrie Fisher', '123 Maple St.', 'F', '
   1999-09-09');
INSERT INTO moviestar VALUES ('Mark Hamill', '456 Oak Rd.', 'M', '
   1988-08-08');
INSERT INTO moviestar VALUES ('Debra Winger', 'A way', 'F', '1978-05-06'
   );
INSERT INTO moviestar VALUES ('Jack Nicholson', 'X path', 'M', '
   1949-05-05');
INSERT INTO moviestar VALUES ('Kevin Spacey', 'New York Av.', 'F', '
   1937-12-21');
INSERT INTO moviestar VALUES ('AAA', 'New York Av.', 'F', '1937-12-21');
```

the execution results are shown in the Fig. 4.

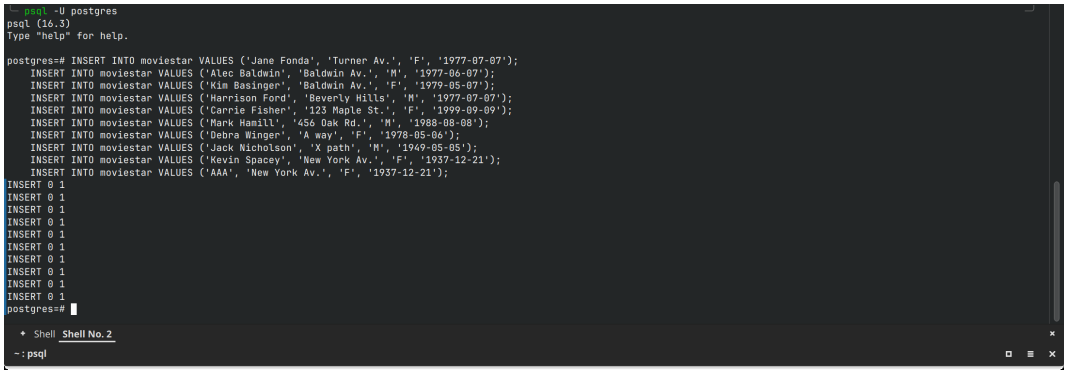


Figure 4: Insert Movie Stars

1.1.5 Insert Sample Data `starsin`

```
INSERT INTO starsin VALUES ('Star Wars', 1977, 'Carrie Fisher');
INSERT INTO starsin VALUES ('Star Wars', 1977, 'Mark Hamill');
INSERT INTO starsin VALUES ('Star Wars', 1977, 'Harrison Ford');
INSERT INTO starsin VALUES ('Empire Strikes Back', 1980, 'Harrison Ford'
    );
INSERT INTO starsin VALUES ('The Usual Suspects', 1995, 'Kevin Spacey');
INSERT INTO starsin VALUES ('Terms of Endearment', 1983, 'Debra Winger')
    ;
INSERT INTO starsin VALUES ('Terms of Endearment', 1983, 'Jack Nicholson
    ');
```

the execution results are shown in the Fig. 5.

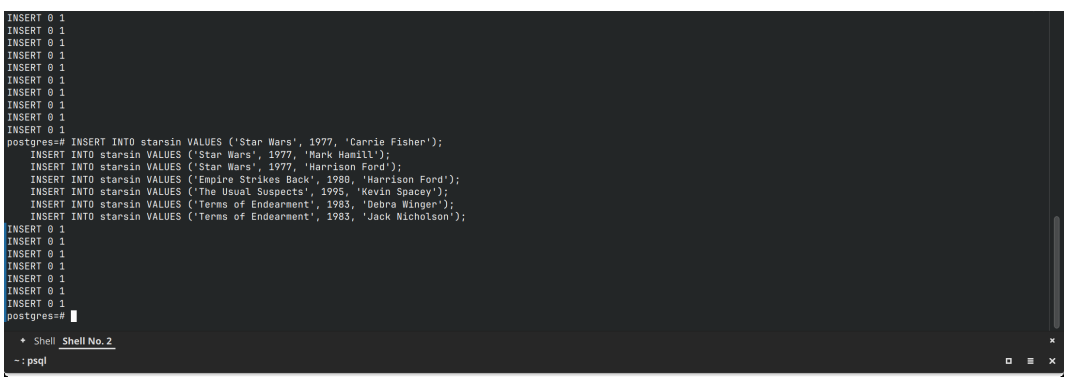


Figure 5: Insert Stars In

1.2 Queries

1.2.1 subquery

```sql
SELECT name
FROM MovieExec
WHERE cert =
        (SELECT producerC
        FROM Movies
        WHERE title = 'Star Wars');

SELECT name
FROM MovieExec
WHERE cert =
        (SELECT *
        FROM Movies
        WHERE title = 'Star Wars');
```

the execution results are shown in the Fig. 6.

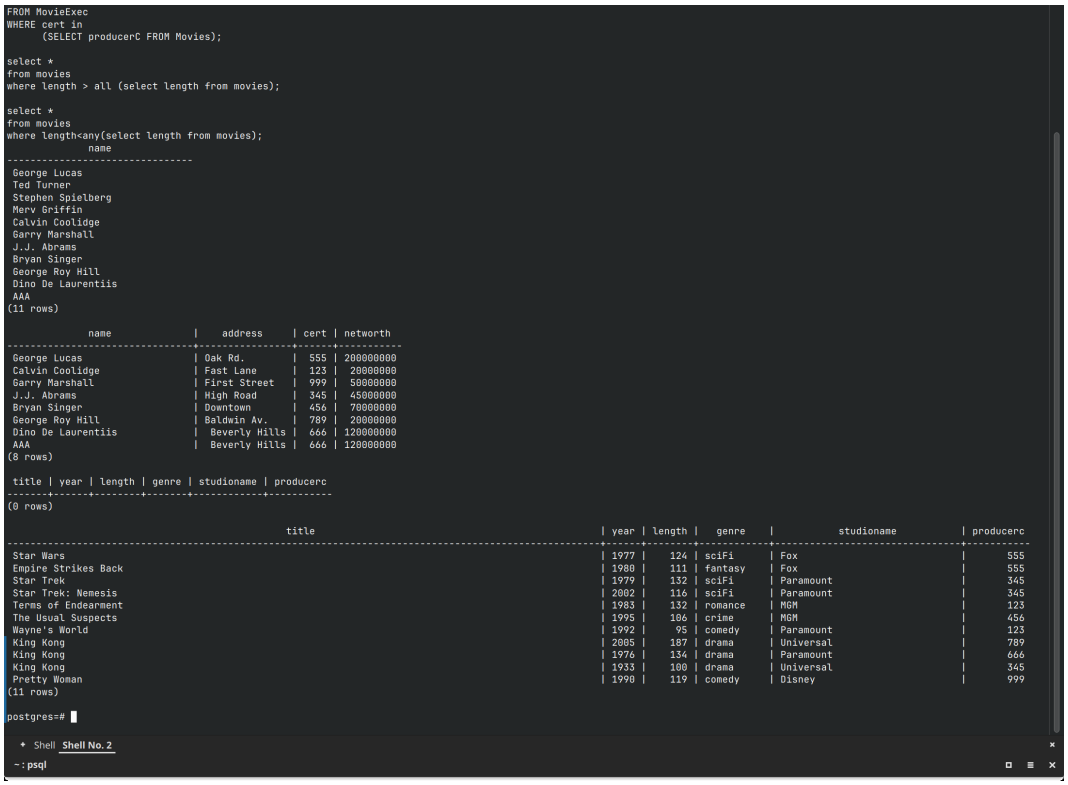
Figure 6: Subquery

1.2.2 conditions Involve Relations

```sql
SELECT name
FROM MovieExec
WHERE Exists
            (SELECT producerC
            FROM Movies
            WHERE title like 'Star%');
SELECT *
FROM MovieExec
WHERE cert in
        (SELECT producerC FROM Movies);

SELECT *
FROM movies
WHERE length > all (SELECT length FROM movies);

SELECT *
```

```
FROM movies
WHERE length<any(SELECT length FROM movies);
```

the execution results are shown in the Fig. 7.



```
FROM MovieExec
WHERE cert in
      (SELECT producerC FROM Movies);

select *
from movies
where length > all (select length from movies);

select *
from movies
where length<any(select length from movies);
              name
--------------------------------
 George Lucas
 Ted Turner
 Stephen Spielberg
 Merv Griffin
 Calvin Coolidge
 Garry Marshall
 J.J. Abrams
 Bryan Singer
 George Roy Hill
 Dino De Laurentiis
 AAA
(11 rows)

               name               |    address     | cert | networth
--------------------------------+----------------+------+-----------
 George Lucas                     | Oak Rd.        |  555 | 200000000
 Calvin Coolidge                  | Fast Lane      |  123 |  20000000
 Garry Marshall                   | First Street   |  999 |  50000000
 J.J. Abrams                      | High Road      |  345 |  45000000
 Bryan Singer                     | Downtown       |  456 |  70000000
 George Roy Hill                  | Baldwin Av.    |  789 |  20000000
 Dino De Laurentiis               |  Beverly Hills |  666 | 120000000
 AAA                              |  Beverly Hills |  666 | 120000000
(8 rows)

 title | year | length | genre | studioname | producerc
-------+------+--------+-------+------------+-----------
(0 rows)

                                          title                                          | year | length |   genre    |           studioname           | producerc
-----------------------------------------------------------------------------------------+------+--------+------------+--------------------------------+-----------
 Star Wars                                                                               | 1977 |    124 | sciFi      | Fox                            |       555
 Empire Strikes Back                                                                     | 1980 |    111 | fantasy    | Fox                            |       555
 Star Trek                                                                               | 1979 |    132 | sciFi      | Paramount                      |       345
 Star Trek: Nemesis                                                                      | 2002 |    116 | sciFi      | Paramount                      |       345
 Terms of Endearment                                                                     | 1983 |    132 | romance    | MGM                            |       123
 The Usual Suspects                                                                      | 1995 |    106 | crime      | MGM                            |       456
 Wayne's World                                                                           | 1992 |     95 | comedy     | Paramount                      |       123
 King Kong                                                                               | 2005 |    187 | drama      | Universal                      |       789
 King Kong                                                                               | 1976 |    134 | drama      | Paramount                      |       666
 King Kong                                                                               | 1933 |    100 | drama      | Universal                      |       345
 Pretty Woman                                                                            | 1990 |    119 | comedy     | Disney                         |       999
(11 rows)

postgres=#
```

Figure 7: Conditions Involve Relations

```
SELECT name
FROM MovieExec
WHERE cert IN
      (SELECT producerC
       FROM Movies
       WHERE (title, year) IN
             (SELECT movieTitle, movieYear
              FROM StarsIn
              WHERE starName = 'Harrison Ford'));
```

the execution results are shown in the Fig. 8.

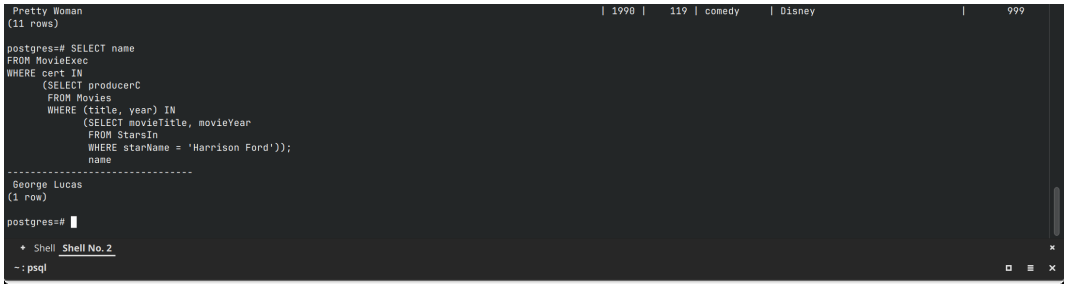


Figure 8: Conditions Involve Relations

1.2.3 Correlated Subqueries

```
SELECT *
FROM Movies Old
WHERE year < ANY
      (SELECT year
       FROM Movies
       WHERE title = Old.title);

SELECT *
FROM MovieExec,
     (SELECT producerC
      FROM Movies,
           StarsIn
      WHERE title = movieTitle
        AND year = movieYear
        AND starName = 'Harrison Ford') Prod
WHERE cert = Prod.producerC;
```

the execution results are shown in the Fig. 9.

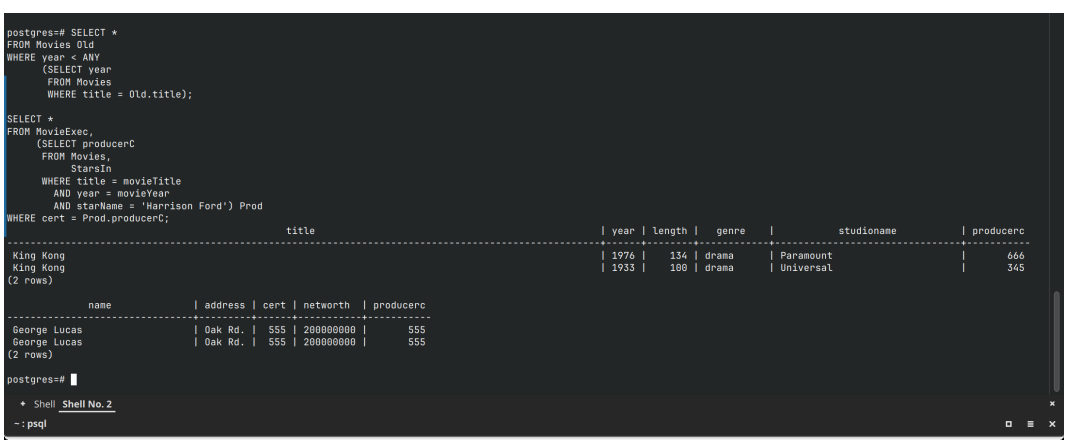


Figure 9: Correlated Subqueries

1.2.4 Join Expression

```
SELECT *
FROM Movies
            JOIN StarsIn ON
    title = movieTitle AND year = movieYear;

SELECT *
FROM Movies
            JOIN StarsIn ON
    title = movieTitle AND year = movieYear
        AND starName = 'Harrison Ford';
```

the execution results are shown in the Fig. 10.

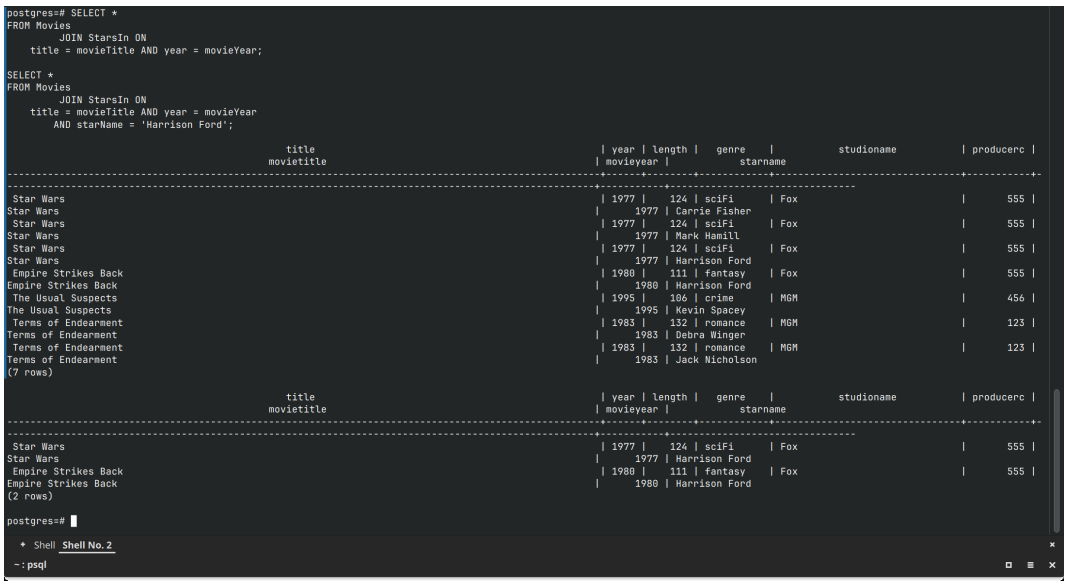


Figure 10: Join Expression

1.2.5 Join Types

```
SELECT *
from MovieStar
            NATURAL JOIN MovieExec;

SELECT *
From Movies
            NATURAL full outer JOIN StarsIn;

SELECT *
From Movies
            NATURAL left outer JOIN StarsIn;
```

```sql
SELECT *
From Movies
            NATURAL right outer JOIN StarsIn;
```

the execution results are shown in the Fig. 11-14.

Figure 11: Join Types

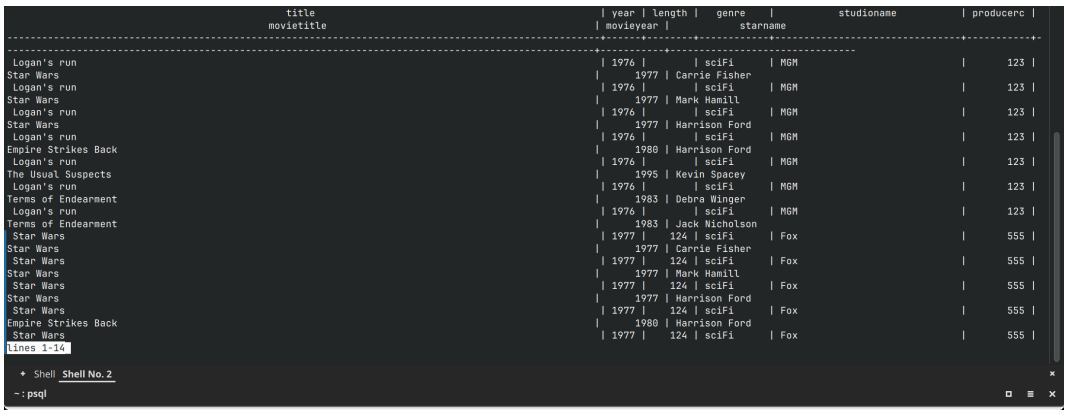


Figure 12: Join Types

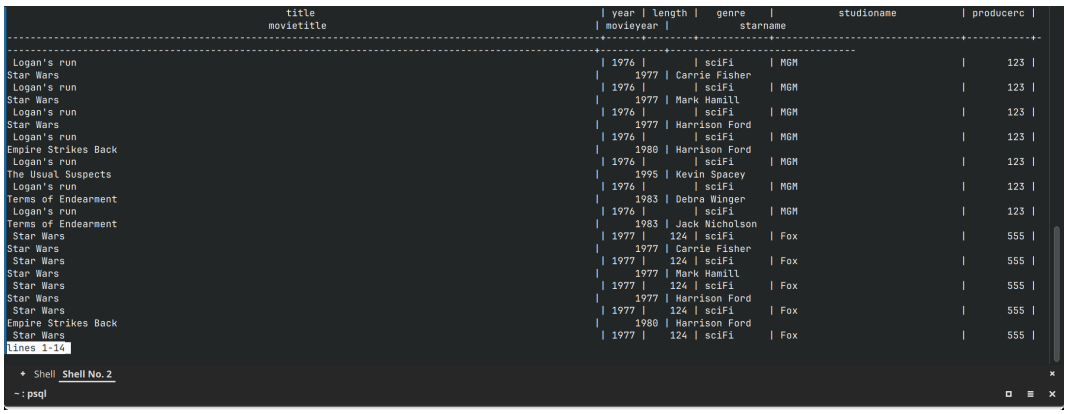
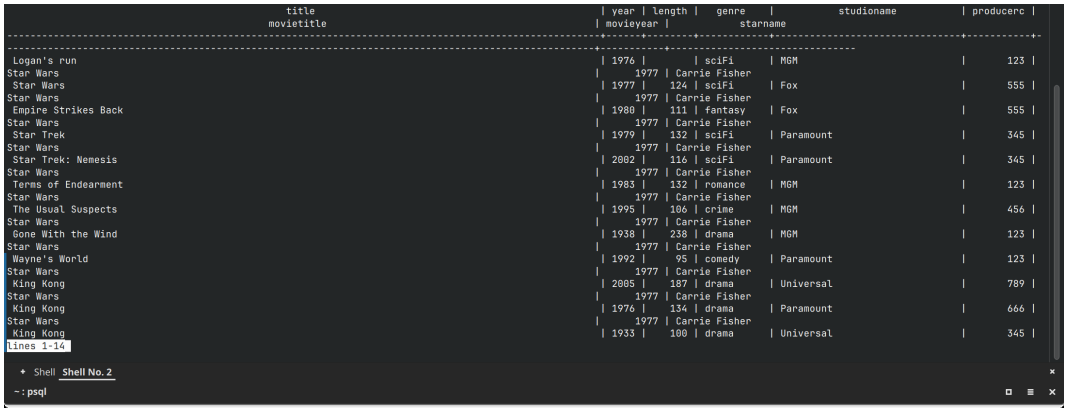


Figure 13: Join Types



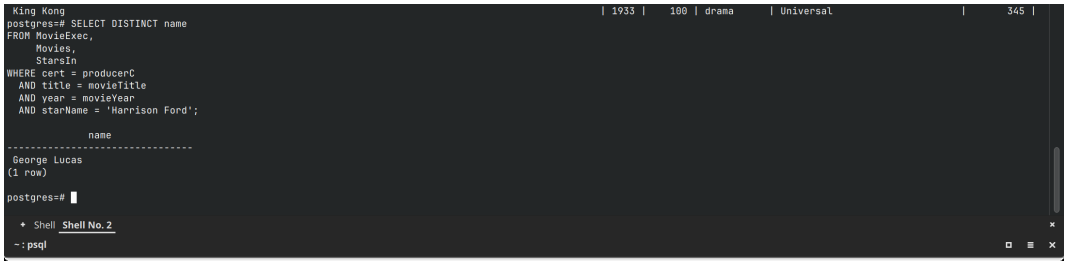
```
                                title                                  | year | length |   genre    |          studioname          | producerc |
                             movietitle                                | movieyear |           starname
-----------------------------------------------------------------------+------+--------+------------+------------------------------+-----------+-
-----------------------------------------------------------------------+-----------+-----------------------------
 Logan's run                                                           | 1976 |        | sciFi      | MGM                          |       123 |
Star Wars                                                              |      1977 | Carrie Fisher
 Star Wars                                                             | 1977 |    124 | sciFi      | Fox                          |       555 |
Star Wars                                                              |      1977 | Carrie Fisher
 Empire Strikes Back                                                   | 1980 |    111 | fantasy    | Fox                          |       555 |
Star Wars                                                              |      1977 | Carrie Fisher
 Star Trek                                                             | 1979 |    132 | sciFi      | Paramount                    |       345 |
Star Wars                                                              |      1977 | Carrie Fisher
 Star Trek: Nemesis                                                    | 2002 |    116 | sciFi      | Paramount                    |       345 |
Star Wars                                                              |      1977 | Carrie Fisher
 Terms of Endearment                                                   | 1983 |    132 | romance    | MGM                          |       123 |
Star Wars                                                              |      1977 | Carrie Fisher
 The Usual Suspects                                                    | 1995 |    106 | crime      | MGM                          |       456 |
Star Wars                                                              |      1977 | Carrie Fisher
 Gone With the Wind                                                    | 1938 |    238 | drama      | MGM                          |       123 |
Star Wars                                                              |      1977 | Carrie Fisher
 Wayne's World                                                         | 1992 |     95 | comedy     | Paramount                    |       123 |
Star Wars                                                              |      1977 | Carrie Fisher
 King Kong                                                             | 2005 |    187 | drama      | Universal                    |       789 |
Star Wars                                                              |      1977 | Carrie Fisher
 King Kong                                                             | 1976 |    134 | drama      | Paramount                    |       666 |
Star Wars                                                              |      1977 | Carrie Fisher
 King Kong                                                             | 1933 |    100 | drama      | Universal                    |       345 |
lines 1-14
```

Figure 14: Join Types

1.2.6 Eliminating Duplicates

```sql
SELECT DISTINCT name
FROM MovieExec,
        Movies,
        StarsIn
WHERE cert = producerC
    AND title = movieTitle
    AND year = movieYear
    AND starName = 'Harrison Ford';
```

the execution results are shown in the Fig. 15.



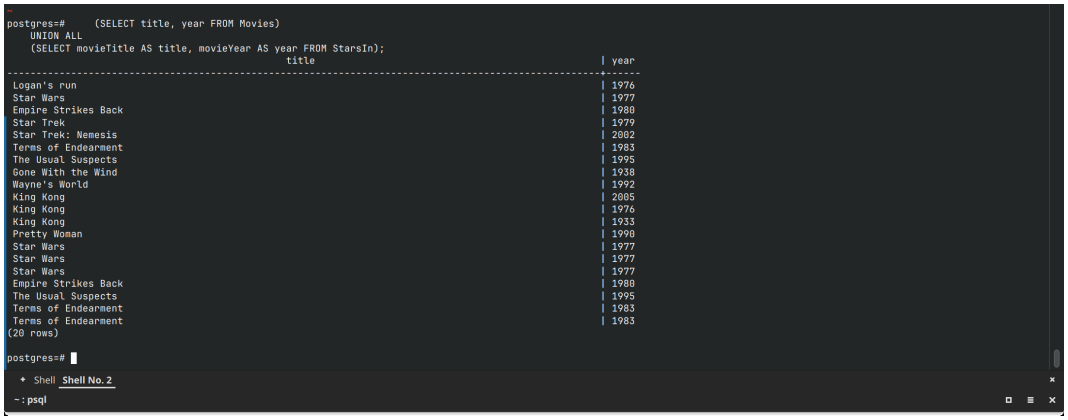
```
 King Kong                                                             | 1933 |    100 | drama      | Universal                    |       345 |
postgres=# SELECT DISTINCT name
FROM MovieExec,
     Movies,
     StarsIn
WHERE cert = producerC
  AND title = movieTitle
  AND year = movieYear
  AND starName = 'Harrison Ford';

              name
--------------------------------
 George Lucas
(1 row)

postgres=#
```

Figure 15: Eliminating Duplicates

1.2.7 Duplicates in Unions, Intersections, and Differences

```sql
(SELECT title, year FROM Movies)
UNION ALL
(SELECT movieTitle AS title, movieYear AS year FROM StarsIn);
```

the execution results are shown in the Fig. 16.



```
postgres=#     (SELECT title, year FROM Movies)
    UNION ALL
    (SELECT movieTitle AS title, movieYear AS year FROM StarsIn);
                                        title                                        | year
-----------------------------------------------------------------------------------------+------
 Logan's run                                                                            | 1976
 Star Wars                                                                              | 1977
 Empire Strikes Back                                                                    | 1980
 Star Trek                                                                              | 1979
 Star Trek: Nemesis                                                                     | 2002
 Terms of Endearment                                                                    | 1983
 The Usual Suspects                                                                     | 1995
 Gone With the Wind                                                                     | 1938
 Wayne's World                                                                          | 1992
 King Kong                                                                              | 2005
 King Kong                                                                              | 1976
 King Kong                                                                              | 1933
 Pretty Woman                                                                           | 1990
 Star Wars                                                                              | 1977
 Star Wars                                                                              | 1977
 Star Wars                                                                              | 1977
 Empire Strikes Back                                                                    | 1980
 The Usual Suspects                                                                     | 1995
 Terms of Endearment                                                                    | 1983
 Terms of Endearment                                                                    | 1983
(20 rows)

postgres=#
```

Figure 16: Duplicates in Unions

2 Excerise 6.3.2(a-d)

Write the following queries, based on the database schema

- Classes(class, type, country, numGuns, bore, displacement)
- Ships(name, class, launched)
- Battles(name, date)
- Outcomes(ship, battle, result)

You should use at least one subquery in each of your answers and write each query in two significantly different ways (e.g., using different sets of the operators EXISTS, IN, ALL, and ANY).

a) Find the countries whose ships had the largest number of guns.

b) Find the classes of ships, at least one of which was sunk in a battle.

c) Find the names of the ships with a 16-inch bore.

d) Find the battles in which ships of the Kongo class participated.

2.1 Solutions

a. Find the countries whose ships had the largest number of guns.

```sql
SELECT country
FROM Classes
WHERE numGuns = (SELECT MAX(numGuns) FROM Classes);
```

b. Find the classes of ships, at least one of which was sunk in a battle.

```sql
SELECT class
FROM Classes
WHERE class IN (
    SELECT class FROM Ships WHERE name IN (
        SELECT ship FROM Outcomes WHERE result = 'sunk'));
```

c. Find the names of the ships with a 16-inch bore.

```sql
SELECT name
FROM Ships
WHERE class IN (
    SELECT class FROM Classes WHERE bore = 16);
```

d. Find the battles in which ships of the Kongo class participated.

```sql
SELECT name
FROM Battles
WHERE name IN (
    SELECT battle FROM Outcomes WHERE ship IN (
        SELECT name FROM Ships WHERE class = 'Kongo'));
```

3 Excerise 6.3.7

For these relations from our running movie database schema

- StarsIn(movieTitle, movieYear, starName)
- MovieStar(name, address, gender, birthdate)
- MovieExec(name, address, cert#, netWorth)

- Studio(name, address, presC#)

describe the tuples that would appear in the following SQL expressions:

a) `Studio CROSS JOIN MovieExec;`

b) `StarsIn NATURAL FULL OUTER JOIN MovieStar;`

c) `StarsIn FULL OUTER JOIN MovieStar ON name = starName;`

3.1 Solutions

a. `Studio CROSS JOIN MovieExec;`

```
(name, address, presC#, name, address, cert#, netWorth)
```

b. `StarsIn NATURAL FULL OUTER JOIN MovieStar;`

```
(movieTitle, movieYear, starName, name, address, gender, birthdate)
```

c. `StarsIn FULL OUTER JOIN MovieStar ON name = starName;`

```
(movieTitle, movieYear, starName, name, address, gender, birthdate)
```

4 Excerise 6.3.9

Using the two relations

- Classes(class, type, country, numGuns, bore, displacement)
- Ships(name, class, launched)

from our database schema of Exercise 2.4.3, write a SQL query that will produce all available information about ships, including that information available in the `Classes` relation. You need not produce information about classes if there are no ships of that class mentioned in `Ships`.

4.1 Solutions

```sql
SELECT * FROM Ships
LEFT OUTER JOIN Classes ON Ships.class = Classes.class;
```