Principles of Database Systems
Assignment #3 - Structured Query Language 1

He Tianyang, 3022001441

October 17, 2024

1 Execute the SQL in Slides

1.1 Preparation

1.1.1 Table Creation

```
CREATE TABLE Movies (
    title        CHAR(100),
    year         INT,
    length       INT,
    genre        CHAR(10),
    studioName   CHAR(30),
    producerC    INT,
    PRIMARY KEY (title, year)
);

\easyfigure{hw4-1.png}{Table Creation Results}{fig:table-creation}
```

The execute results is shown in Figure 1.

1.1.2 Sample Data in `Movies` Table

```
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);
```

Figure 1: Table Creation Results

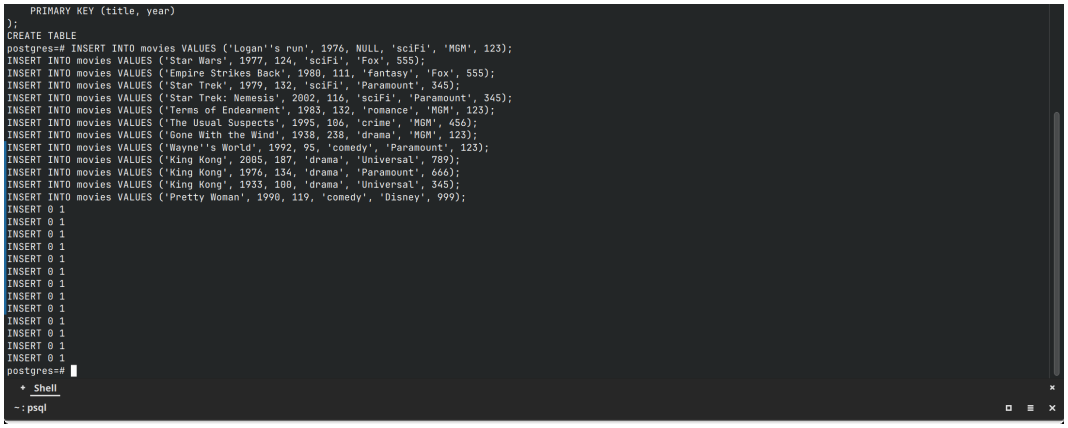
```
INSERT INTO movies VALUES ('Star Trek', 1979, 132, 'sciFi', 'Paramount',
    345);
INSERT INTO movies VALUES ('Star Trek: Nemesis', 2002, 116, 'sciFi', '
   Paramount', 345);
INSERT INTO movies VALUES ('Terms of Endearment', 1983, 132, 'romance',
   'MGM', 123);
INSERT INTO movies VALUES ('The Usual 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);
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 execute results is shown in Figure 2.

1.2 Queries

1.2.1 Simple Query

```
SELECT * FROM movies where studioname='Disney' AND year=1990;
SELECT title, length FROM movies where studioname='Disney' AND year
   =1990;
SELECT title AS name, length AS duration FROM movies where studioname='
   Disney' AND year=1990;
```

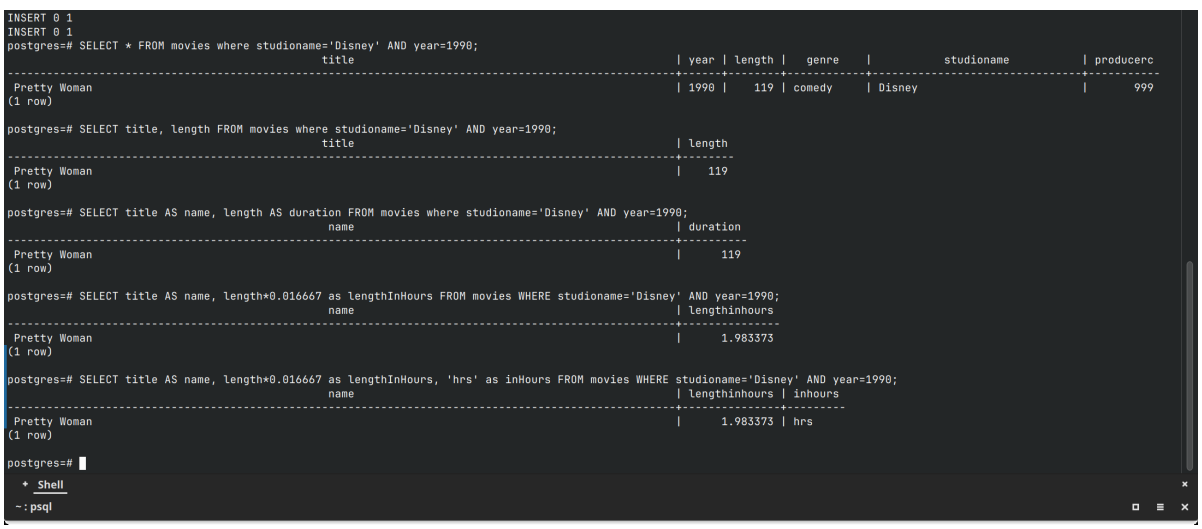


```
   PRIMARY KEY (title, year)
);
CREATE TABLE
postgres=# INSERT INTO movies VALUES ('Logan''s run', 1976, NULL, 'sciFi', 'MGM', 123);
INSERT INTO movies VALUES ('Star Wars', 1977, 124, 'sciFi', 'Fox', 555);
INSERT INTO movies VALUES ('Empire Strikes Back', 1980, 111, 'fantasy', 'Fox', 555);
INSERT 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);
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);
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
postgres=#
```

Figure 2: Sample Data Insertion Results

```sql
SELECT title AS name, length*0.016667 as lengthInHours FROM movies WHERE
    studioname='Disney' AND year=1990;
SELECT title AS name, length*0.016667 as lengthInHours, 'hrs' as inHours
    FROM movies WHERE studioname='Disney' AND year=1990;
```

The execute results is shown in Figure 3.



```
INSERT 0 1
INSERT 0 1
postgres=# SELECT * FROM movies where studioname='Disney' AND year=1990;
                                        title                                         | year | length |   genre    |           studioname           | producerc
--------------------------------------------------------------------------------------+------+--------+------------+--------------------------------+-----------
 Pretty Woman                                                                         | 1990 |    119 | comedy     | Disney                         |       999
(1 row)

postgres=# SELECT title, length FROM movies where studioname='Disney' AND year=1990;
                                        title                                         | length
--------------------------------------------------------------------------------------+--------
 Pretty Woman                                                                         |    119
(1 row)

postgres=# SELECT title AS name, length AS duration FROM movies where studioname='Disney' AND year=1990;
                                        name                                          | duration
--------------------------------------------------------------------------------------+----------
 Pretty Woman                                                                         |      119
(1 row)

postgres=# SELECT title AS name, length*0.016667 as lengthInHours FROM movies WHERE studioname='Disney' AND year=1990;
                                        name                                          | lengthinhours
--------------------------------------------------------------------------------------+---------------
 Pretty Woman                                                                         |      1.983373
(1 row)

postgres=# SELECT title AS name, length*0.016667 as lengthInHours, 'hrs' as inHours FROM movies WHERE studioname='Disney' AND year=1990;
                                        name                                          | lengthinhours | inhours
--------------------------------------------------------------------------------------+---------------+---------
 Pretty Woman                                                                         |      1.983373 | hrs
(1 row)

postgres=#
```

Figure 3: Simple Query Results

1.2.2 Query with Complex WHERE

```sql
SELECT title FROM movies WHERE (year > 1970 OR  length < 90) AND
   studioname='MGM';
SELECT title FROM movies WHERE year between 1970 and 2000;
SELECT title FROM movies WHERE title LIKE '%Star W__s%';
SELECT title FROM movies WHERE title LIKE '%''s%';
```

The execute results is shown in Figure 4.

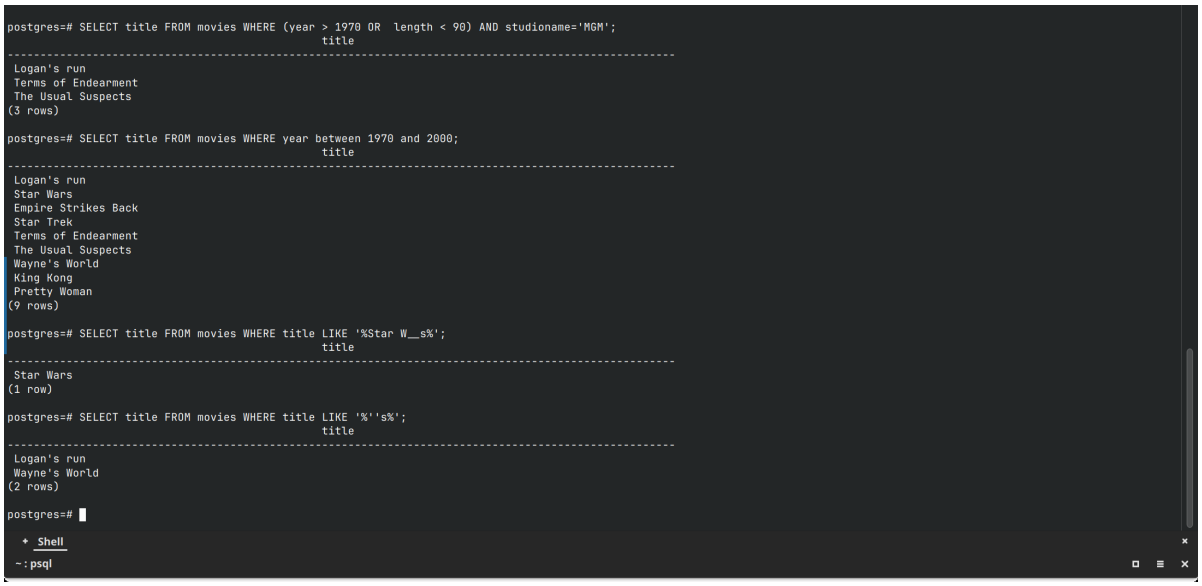


Figure 4: Query with Complex WHERE Results

1.2.3 Query with NULL Values

```
INSERT INTO movies VALUES ('ABC', 1976, NULL, 'sciFi', 'MGM', NULL);

SELECT title, movies.length=3 FROM Movies;
SELECT title, movies.length+3 FROM Movies;
```

The execute results is shown in Figure 5.

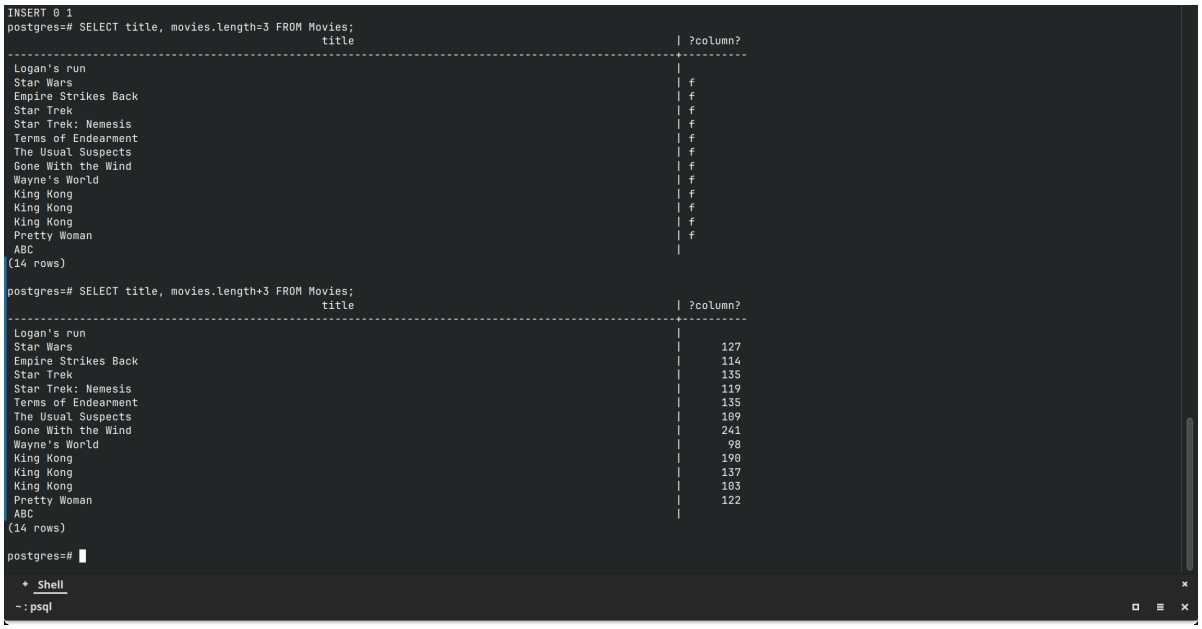
```
SELECT * FROM movies WHERE movies.length==NULL;
SELECT * FROM movies WHERE movies.length!=NULL;
SELECT * FROM movies WHERE movies.length is NULL;
SELECT * FROM movies WHERE movies.length is not NULL;
```

The execute results is shown in Figure 6.

```
SELECT * FROM movies WHERE length<=120 OR length>120;
```

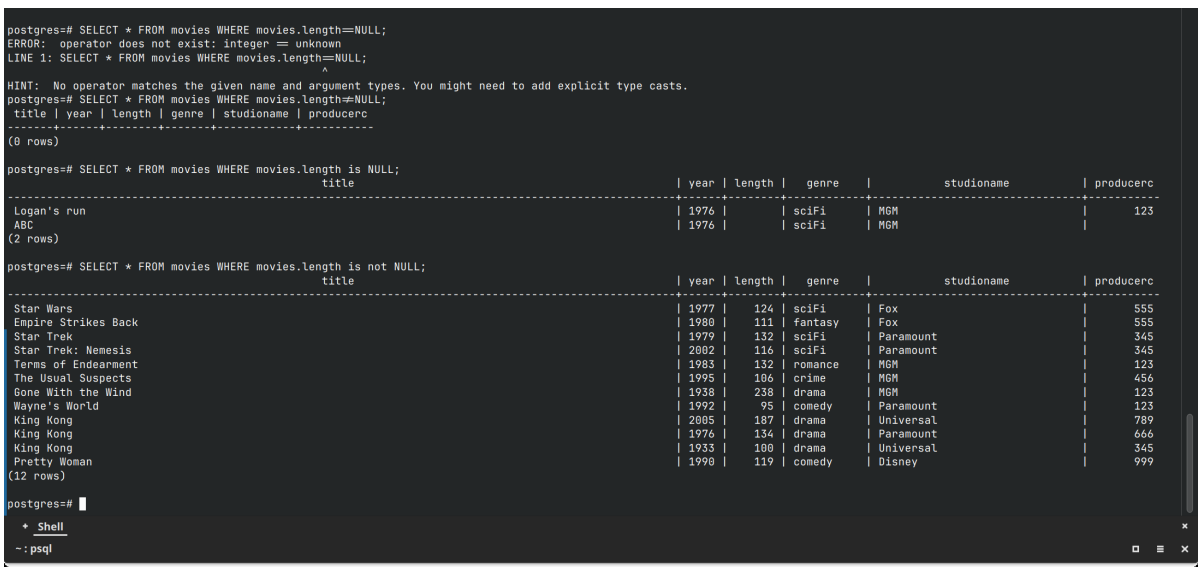
The execute results is shown in Figure 7.

1.2.4 Query with Ordering



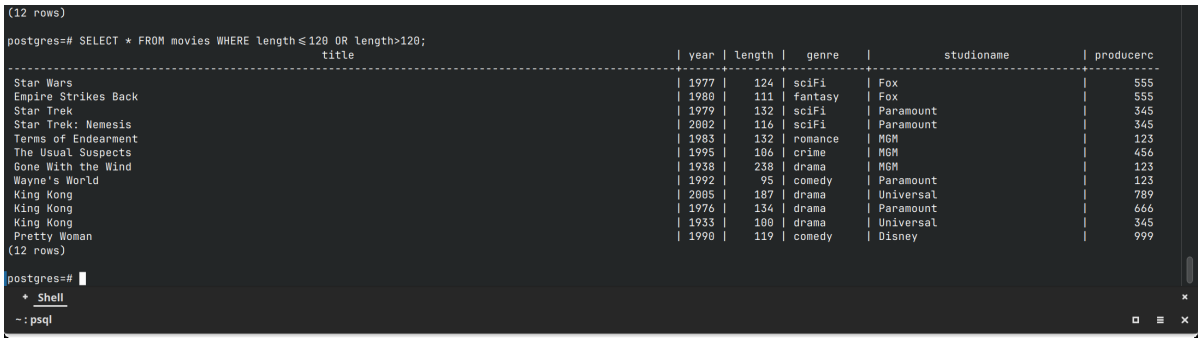
```
INSERT 0 1
postgres=# SELECT title, movies.length=3 FROM Movies;
                                        title                                        | ?column?
-------------------------------------------------------------------------------------+----------
 Logan's run                                                                         |
 Star Wars                                                                           | f
 Empire Strikes Back                                                                 | f
 Star Trek                                                                           | f
 Star Trek: Nemesis                                                                  | f
 Terms of Endearment                                                                 | f
 The Usual Suspects                                                                  | f
 Gone With the Wind                                                                  | f
 Wayne's World                                                                       | f
 King Kong                                                                           | f
 King Kong                                                                           | f
 King Kong                                                                           | f
 Pretty Woman                                                                        | f
 ABC                                                                                 |
(14 rows)

postgres=# SELECT title, movies.length+3 FROM Movies;
                                        title                                        | ?column?
-------------------------------------------------------------------------------------+----------
 Logan's run                                                                         |
 Star Wars                                                                           |      127
 Empire Strikes Back                                                                 |      114
 Star Trek                                                                           |      135
 Star Trek: Nemesis                                                                  |      119
 Terms of Endearment                                                                 |      135
 The Usual Suspects                                                                  |      109
 Gone With the Wind                                                                  |      241
 Wayne's World                                                                       |       98
 King Kong                                                                           |      190
 King Kong                                                                           |      137
 King Kong                                                                           |      103
 Pretty Woman                                                                        |      122
 ABC                                                                                 |
(14 rows)

postgres=#
```

Figure 5: Query with `NULL` Values Results



```
postgres=# SELECT * FROM movies WHERE movies.length==NULL;
ERROR:  operator does not exist: integer == unknown
LINE 1: SELECT * FROM movies WHERE movies.length==NULL;
                                                 ^
HINT:  No operator matches the given name and argument types. You might need to add explicit type casts.
postgres=# SELECT * FROM movies WHERE movies.length=NULL;
 title | year | length | genre | studioname | producerc
-------+------+--------+-------+------------+-----------
(0 rows)

postgres=# SELECT * FROM movies WHERE movies.length is NULL;
                                        title                                        | year | length |   genre    |           studioname           | producerc
-------------------------------------------------------------------------------------+------+--------+------------+--------------------------------+-----------
 Logan's run                                                                         | 1976 |        | sciFi      | MGM                            |       123
 ABC                                                                                 | 1976 |        | sciFi      | MGM                            |
(2 rows)

postgres=# SELECT * FROM movies WHERE movies.length is not NULL;
                                        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
 Gone With the Wind                                                                  | 1938 |    238 | drama      | MGM                            |       123
 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
(12 rows)

postgres=#
```

Figure 6: Query with `NULL` Values Results

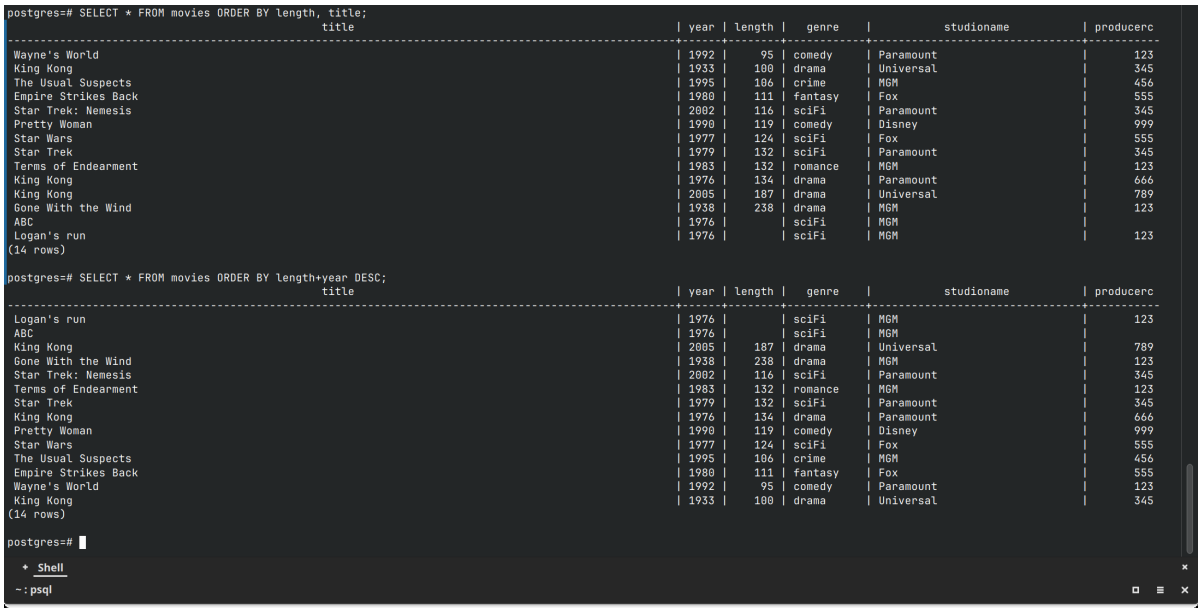


```
(12 rows)

postgres=# SELECT * FROM movies WHERE length≤120 OR length>120;
                                        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
 Gone With the Wind                                                                  | 1938 |    238 | drama      | MGM                            |       123
 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
(12 rows)

postgres=#
```

Figure 7: Query with `NULL` Values Results

```
SELECT * FROM movies ORDER BY length, title;
SELECT * FROM movies ORDER BY length+year DESC;
```

The execute results is shown in Figure 8.



```
postgres=# SELECT * FROM movies ORDER BY length, title;
                                        title                                        | year | length |   genre    |          studioname          | producerc
-------------------------------------------------------------------------------------+------+--------+------------+------------------------------+-----------
 Wayne's World                                                                       | 1992 |     95 | comedy     | Paramount                    |       123
 King Kong                                                                           | 1933 |    100 | drama      | Universal                    |       345
 The Usual Suspects                                                                  | 1995 |    106 | crime      | MGM                          |       456
 Empire Strikes Back                                                                 | 1980 |    111 | fantasy    | Fox                          |       555
 Star Trek: Nemesis                                                                  | 2002 |    116 | sciFi      | Paramount                    |       345
 Pretty Woman                                                                        | 1990 |    119 | comedy     | Disney                       |       999
 Star Wars                                                                           | 1977 |    124 | sciFi      | Fox                          |       555
 Star Trek                                                                           | 1979 |    132 | sciFi      | Paramount                    |       345
 Terms of Endearment                                                                 | 1983 |    132 | romance    | MGM                          |       123
 King Kong                                                                           | 1976 |    134 | drama      | Paramount                    |       666
 King Kong                                                                           | 2005 |    187 | drama      | Universal                    |       789
 Gone With the Wind                                                                  | 1938 |    238 | drama      | MGM                          |       123
 ABC                                                                                 | 1976 |        | sciFi      | MGM                          |
 Logan's run                                                                         | 1976 |        | sciFi      | MGM                          |       123
(14 rows)

postgres=# SELECT * FROM movies ORDER BY length+year DESC;
                                        title                                        | year | length |   genre    |          studioname          | producerc
-------------------------------------------------------------------------------------+------+--------+------------+------------------------------+-----------
 Logan's run                                                                         | 1976 |        | sciFi      | MGM                          |       123
 ABC                                                                                 | 1976 |        | sciFi      | MGM                          |
 King Kong                                                                           | 2005 |    187 | drama      | Universal                    |       789
 Gone With the Wind                                                                  | 1938 |    238 | drama      | MGM                          |       123
 Star Trek: Nemesis                                                                  | 2002 |    116 | sciFi      | Paramount                    |       345
 Terms of Endearment                                                                 | 1983 |    132 | romance    | MGM                          |       123
 Star Trek                                                                           | 1979 |    132 | sciFi      | Paramount                    |       345
 King Kong                                                                           | 1976 |    134 | drama      | Paramount                    |       666
 Pretty Woman                                                                        | 1990 |    119 | comedy     | Disney                       |       999
 Star Wars                                                                           | 1977 |    124 | sciFi      | Fox                          |       555
 The Usual Suspects                                                                  | 1995 |    106 | crime      | MGM                          |       456
 Empire Strikes Back                                                                 | 1980 |    111 | fantasy    | Fox                          |       555
 Wayne's World                                                                       | 1992 |     95 | comedy     | Paramount                    |       123
 King Kong                                                                           | 1933 |    100 | drama      | Universal                    |       345
(14 rows)

postgres=#
```

Figure 8: Query with Ordering Results

1.3 Queries on More Than One Table

1.3.1 Preparation

```
drop table movies;

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
);
```

```sql
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 execute results is shown in Figure 9.

Figure 9: Queries on More Than One Table Results

1.3.2 Example Data

```sql
-- table 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);
```

```sql
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);
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);

-- table movieexec
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);
INSERT INTO movieexec VALUES ('AAA', 'Beverly Hills', 666, 120000000);

-- table moviestar
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');
```

```sql
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');

-- table 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
   ');
```

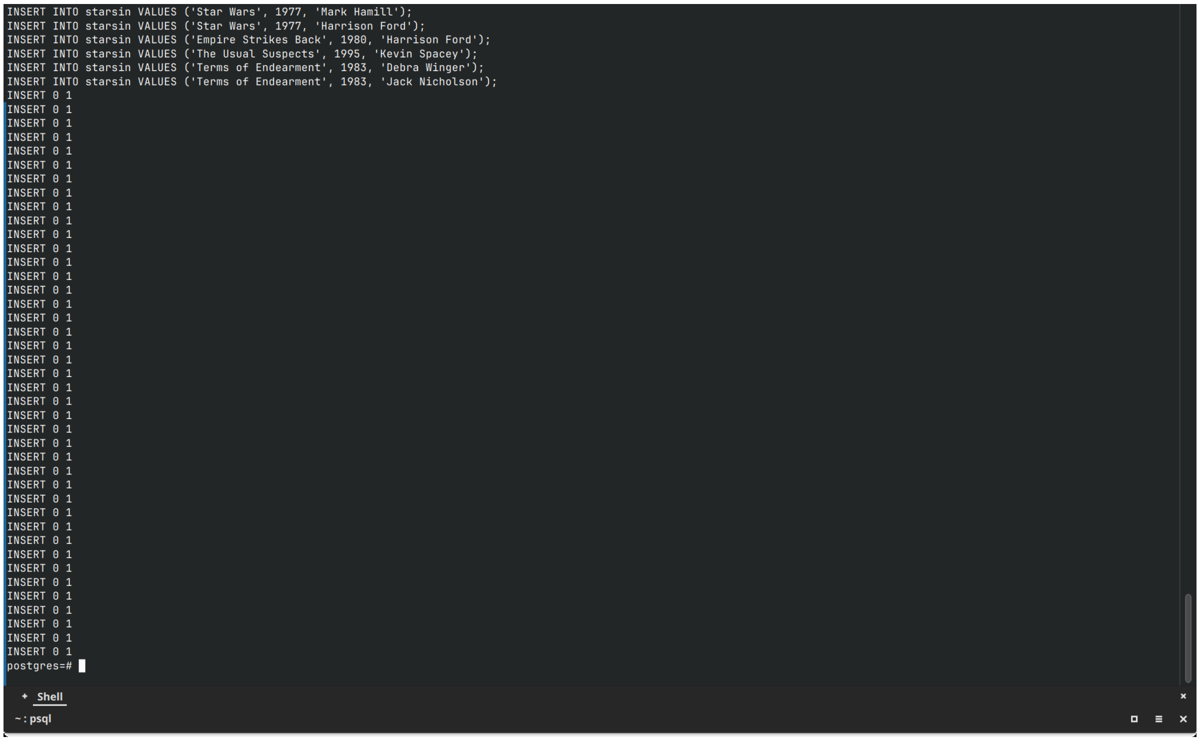
the execute results is shown in Figure 10.

1.3.3 Products Query

```sql
SELECT moviestar.name, movieexec.name
FROM moviestar,
        movieexec
WHERE moviestar.address = movieexec.address;

SELECT Star1.name, Star2.name
FROM moviestar Star1,
        moviestar Star2
WHERE Star1.address = Star2.address
    and Star1.name < Star2.name;
```

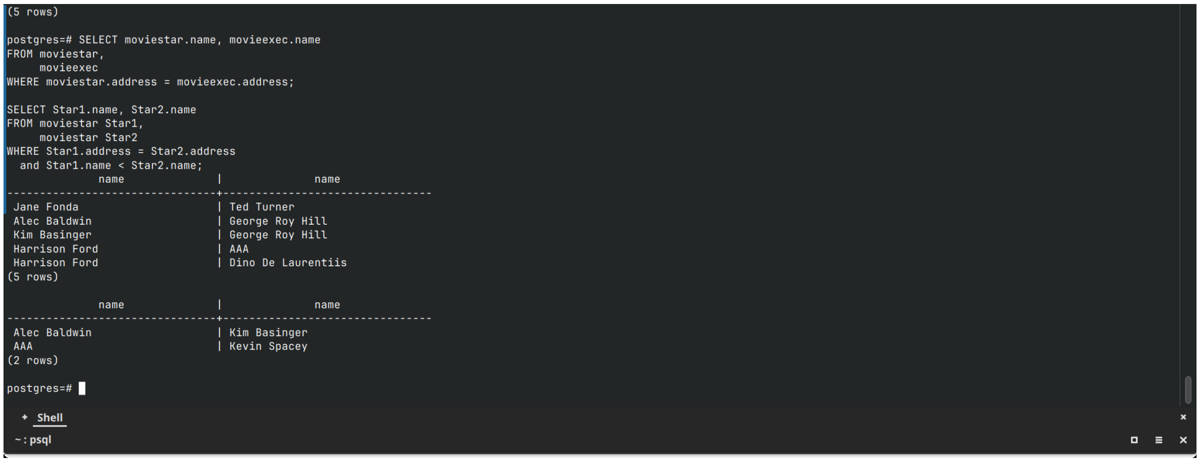
the execute results is shown in Figure 11.

1.3.4 Union, Intersection and Difference



```
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');
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
postgres=#
```

Figure 10: Queries on More Than One Table Results



```
(5 rows)

postgres=# SELECT moviestar.name, movieexec.name
FROM moviestar,
     movieexec
WHERE moviestar.address = movieexec.address;

SELECT Star1.name, Star2.name
FROM moviestar Star1,
     moviestar Star2
WHERE Star1.address = Star2.address
  and Star1.name < Star2.name;
             name              |             name
-------------------------------+-------------------------------
 Jane Fonda                    | Ted Turner
 Alec Baldwin                  | George Roy Hill
 Kim Basinger                  | George Roy Hill
 Harrison Ford                 | AAA
 Harrison Ford                 | Dino De Laurentiis
(5 rows)

             name              |             name
-------------------------------+-------------------------------
 Alec Baldwin                  | Kim Basinger
 AAA                           | Kevin Spacey
(2 rows)

postgres=#
```

Figure 11: Queries on More Than One Table Results

```sql
(SELECT name, address
 FROM MovieStar
 WHERE gender = 'F')
INTERSECT
(SELECT name, address
 FROM MovieExec
 WHERE netWorth > 10000000);

(SELECT name, address FROM MovieStar)
EXCEPT
(SELECT name, address FROM MovieExec);

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

the execute results is shown in Figure 12.

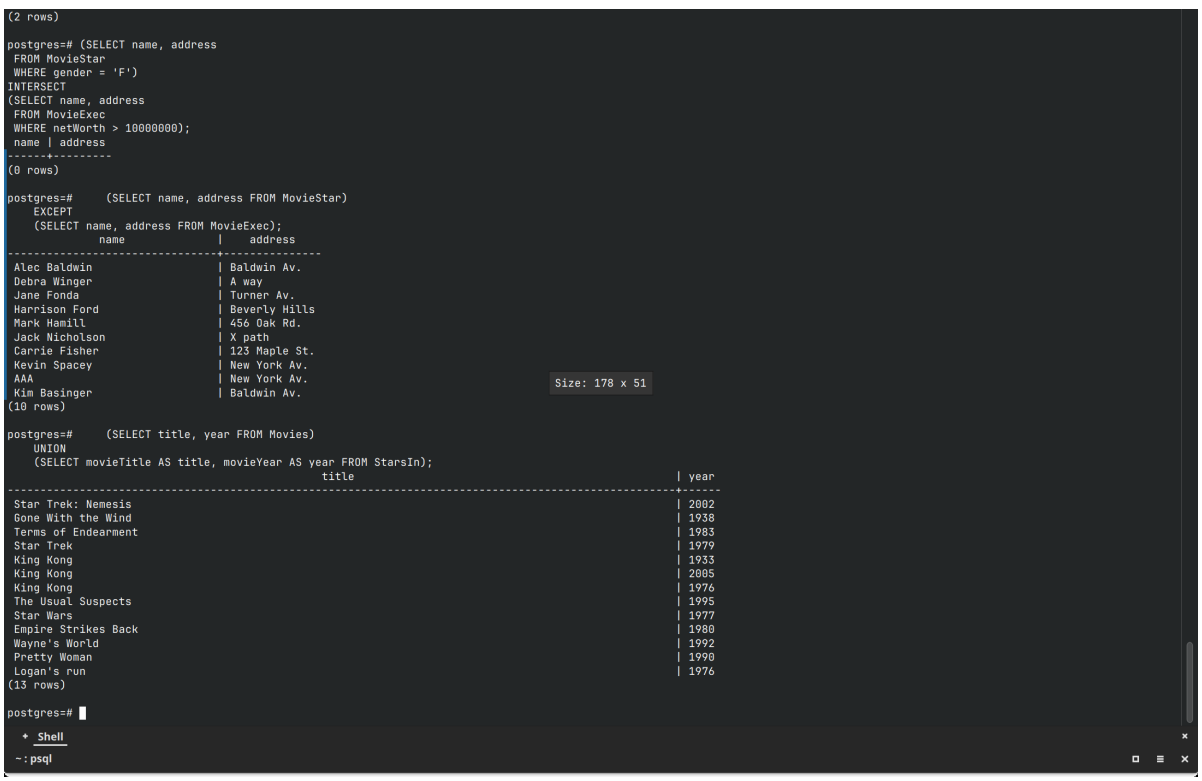


Figure 12: Union, Intersection and Difference Results

2 Exercise 6.1.4

Exercise 6.1.4: Write the following queries based on the database schema of Exercise 2.4.3:

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

and show the result of your query on the data of Exercise 2.4.3.

a) Find the class name and country for all classes with at least 10 guns.

b) Find the names of all ships launched prior to 1918, but call the resulting column `shipName`.

c) Find the names of ships sunk in battle and the name of the battle in which they were sunk.

d) Find all ships that have the same name as their class.

e) Find the names of all ships that begin with the letter "R."

f) Find the names of all ships whose name consists of three or more words (e.g., *King George V*).

2.1 Solutions

a. Find the class name and country for all classes with at least 10 guns.

```
SELECT class, country FROM Classes WHERE numGuns >= 10;
```

b. Find the names of all ships launched prior to 1918, but call the resulting column `shipName`.

```
SELECT name AS shipName FROM Ships WHERE launched < 1918;
```

c. Find the names of ships sunk in battle and the name of the battle in which they were sunk.

```
SELECT ship, battle FROM Outcomes WHERE result = 'sunk';
```

d. Find all ships that have the same name as their class.

```
SELECT Ships.name FROM Ships, Classes WHERE Ships.class = Classes.class;

-- or using join keywords
SELECT Ships.name FROM Ships
JOIN Classes ON Ships.class = Classes.class;
```

e. Find the names of all ships that begin with the letter `R`

```
SELECT name FROM Ships WHERE name LIKE 'R%';
```

f. Find the names of all ships whose name consists of three or more words (e.g., *King George V*).

```
SELECT name FROM Ships WHERE name LIKE '% % %';
```

3 Exercise 6.2.1

Exercise 6.2.1: Using the database schema of our running movie example

- `Movies(title, year, length, genre, studioName, producerC#)`
- `StarsIn(movieTitle, movieYear, starName)`
- `MovieStar(name, address, gender, birthdate)`
- `MovieExec(name, address, cert#, netWorth)`
- `Studio(name, address, presC#)`

write the following queries in SQL.

a) Who were the male stars in *Titanic*?

b) Which stars appeared in movies produced by MGM in 1995?

c) Who is the president of MGM studios?

d) Which movies are longer than *Gone With the Wind*?

e) Which executives are worth more than Merv Griffin?

3.1 Solutions

a. Who were the male stars in *Titanic*?

```
SELECT starName FROM starsin WHERE movieTitle='Titanic' AND starName IN
   (SELECT name FROM moviestar WHERE gender='M');

-- or using join keywords
SELECT starName FROM starsin
JOIN moviestar ON starsin.starName = moviestar.name
WHERE movieTitle='Titanic' AND gender='M';
```

b. Which stars appeared in movies produced by MGM in 1995?

```
-- Using subquery
SELECT starName FROM starsin
WHERE (movieTitle, movieYear) IN (
    SELECT title, year FROM movies WHERE studioName = 'MGM' AND year =
   1995
);

-- Using join keywords
SELECT starName FROM starsin
JOIN movies ON starsin.movieTitle = movies.title AND starsin.movieYear =
    movies.year
WHERE movies.studioName = 'MGM' AND movies.year = 1995;
```

c. Who is the president of MGM studios?

```
-- Using subquery
SELECT name FROM movieexec
WHERE "cert#" = (SELECT "presC#" FROM studio WHERE name = 'MGM');
```

```sql
-- Using JOIN
SELECT movieexec.name FROM movieexec
JOIN studio ON movieexec.cert# = studio.presC#
WHERE studio.name = 'MGM';
```

d. Which movies are longer than *Gone With the Wind*?

```sql
-- Ensure uniqueness by specifying the year if known
SELECT title FROM movies WHERE length > (
    SELECT length FROM movies WHERE title = 'Gone With the Wind'
);

-- Corrected JOIN version
SELECT movies.title FROM movies
JOIN movies GWTW ON GWTW.title = 'Gone With the Wind'
WHERE movies.length > GWTW.length;
```

e. Which executives are worth more than Merv Griffin?

```sql
SELECT name FROM movieexec WHERE netWorth > (SELECT netWorth FROM
   movieexec WHERE name='Merv Griffin');

-- or using join keywords
SELECT movieexec.name FROM movieexec
JOIN movieexec MG ON MG.name = 'Merv Griffin'
WHERE movieexec.netWorth > MG.netWorth;
```