Q0. Restore the dvdrental database from the tar file.

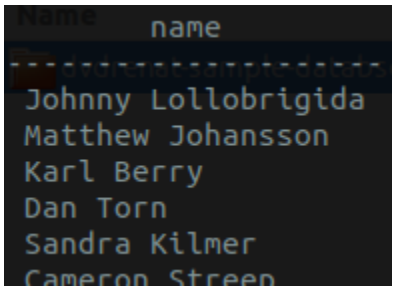
```
112001051@iitpkdd55:~/Desktop/dbmslab/dvdrenat-sample-databse-master$ pg_restore -U postgres -d dvdrental dvdrental.tar
Password:
112001051@iitpkdd55:~/Desktop/dbmslab/dvdrenat-sample-databse-master$ psql -U postgres
Password for user postgres:
psql (10.23 (Ubuntu 10.23-0ubuntu0.18.04.1))
Type "help" for help.

postgres=# \l
                                  List of databases
   Name    |  Owner   | Encoding | Collate | Ctype |   Access privileges
-----------+----------+----------+---------+-------+-----------------------
 dbms1     | postgres | UTF8     | en_IN   | en_IN |
 dvdrental | postgres | UTF8     | en_IN   | en_IN |
 postgres  | postgres | UTF8     | en_IN   | en_IN |
 template0 | postgres | UTF8     | en_IN   | en_IN | =c/postgres          +
           |          |          |         |       | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_IN   | en_IN | =c/postgres          +
           |          |          |         |       | postgres=CTc/postgres
(5 rows)

postgres=# \d
Did not find any relations.
postgres=# \c dvdrental
You are now connected to database "dvdrental" as user "postgres".
```

Q1. Write a query to find all the actors who have acted in the movie "Shawshank Bubble".

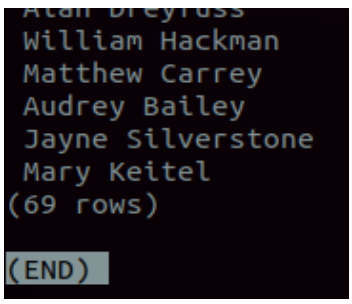
```
dvdrental=# SELECT actor.first_name || ' ' || actor.last_name as Name FROM actor, film_actor, film WHERE
actor.actor_id=film_actor.actor_id AND film_actor.film_id=film.film_id AND film.title='Shawshank Bubble';

        name
--------------------
 Bob Fawcett
 Val Bolger
 Kenneth Pesci
 Sylvester Dern
 Renee Tracy
 Scarlett Bening
 Angela Witherspoon
 Ian Tandy
 Christopher West
 Julia Fawcett
(10 rows)
```

Q2. Write a query to find the actors whose name has ‘a’ in the first 2 positions or in the last 2 positions.

```
dvdrental=# SELECT first_name || ' ' || last_name as name FROM actor WHERE first_name LIKE 'A%' O
R  first_name LIKE '_a%' OR last_name LIKE '%a' OR last_name LIKE '%a_';
```



…



Q3.Display the email id of five customers by changing the domain to “.com” .

```
dvdrental=# SELECT REPLACE(email, '.org', '.com') FROM customer LIMIT 5;
              replace
-------------------------------------
 jared.ely@sakilacustomer.com
 mary.smith@sakilacustomer.com
 patricia.johnson@sakilacustomer.com
 linda.williams@sakilacustomer.com
 barbara.jones@sakilacustomer.com
(5 rows)
```

Description

From these commands, we find that all emails end with ‘.org’.

```
dvdrental=# SELECT count(*) FROM customer;
 count
-------
   599
(1 row)

dvdrental=# SELECT count(*) FROM customer WHERE email LIKE '%.org';
 count
-------
   599
(1 row)
```

So we can use the REPLACE command which searches and replaces a substring (‘.org’) with a new substring(‘.com’) in a string.

Q4. Write a query to concatenate first and last name of actors according to alphabetical order.

```
dvdrental=# SELECT first_name || ' ' || last_name as name FROM actor ORDER BY first_name, last_name;
```

```
      name
--------------------
 Adam Grant
 Adam Hopper
 Al Garland
 Alan Dreyfuss
 Albert Johansson
 Albert Nolte
 Alec Wayne
 Angela Hudson
```

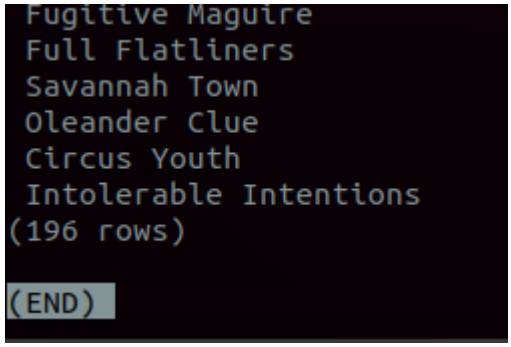
…

```
 Will Wilson
 William Hackman
 Woody Hoffman
 Woody Jolie
 Zero Cage
(200 rows)

(END)
```

Q5. **How many movies are offered in store1 but not in store2.**
Display the name of the movies offered at store 1.

This was run to get the names of films offered in store 1 but not in store 2.
'except' command takes the set difference. I found the set difference of the titles of movies offered in store 1, with the titles of movies offered in store 2.

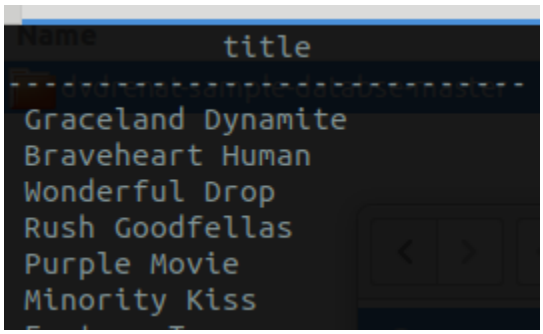
```
dvdrental=# (SELECT film.title FROM film, inventory WHERE film.film_id=inventory.film_id AND inve
ntory.store_id=1) except (SELECT film.title FROM film, inventory WHERE film.film_id=inventory.fil
m_id AND inventory.store_id=2);
```



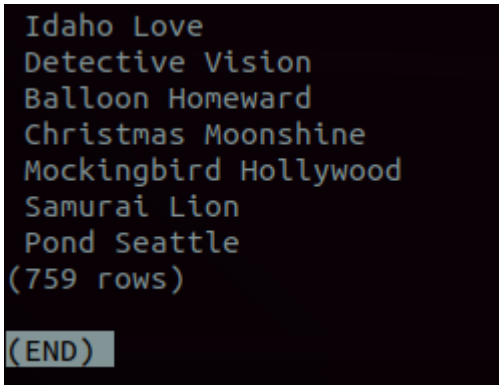
It is 196.
This works as except automatically handles DISTINCT.
This is also the answer for "*Display the name of the movies offered **only** at store 1.*"

Q5.1. **Display the name of the movies offered at store 1.**

```
dvdrental=# (SELECT DISTINCT(film.title) FROM film, inventory WHERE film.film_id=inventory.film_i
d AND inventory.store_id=1);
```



...



DISTINCT was used to remove duplicates.

Q6. Display all the pairs of actors along with their IDs whose

a. first name matches

```
dvdrental=# SELECT (a1.first_name || ' ' || a1.last_name) as actor1name, a1.actor_id, (a2.first_name || ' ' || a2.last_name) as actor2name, a2.actor_id FROM actor AS a1, actor AS a2 WHERE a1.first_name=a2.first_name AND a1.actor_id != a2.actor_id;
```

actor1name	actor_id	actor2name	actor_id
Penelope Guiness	1	Penelope Monroe	120
Penelope Guiness	1	Penelope Cronyn	104
Penelope Guiness	1	Penelope Pinkett	54
Nick Wahlberg	2	Nick Degeneres	166
Nick Wahlberg	2	Nick Stallone	44
Ed Chase	3	Ed Guiness	179
Ed Chase	3	Ed Mansfield	136
Johnny Lollobrigida	5	Johnny Cage	40
Matthew Johansson	8	Matthew Carrey	181

:::

Meryl Allen	194	Meryl Gibson	154
Jayne Silverstone	195	Jayne Nolte	150
Jayne Silverstone	195	Jayne Neeson	62
Reese West	197	Reese Kilmer	45
Mary Keitel	198	Mary Tandy	66
Julia Fawcett	199	Julia Zellweger	186
Julia Fawcett	199	Julia Barrymore	47
Julia Fawcett	199	Julia Mcqueen	27

(190 rows)

(END)

b. last name matches

```
dvdrental=# SELECT (a1.first_name || ' ' || a1.last_name) as actor1name, a1.actor_id, (a2.first_name || ' ' || a2.last_name) as actor2name, a2.actor_id FROM actor AS a1, actor AS a2 WHERE a1.last_name=a2.last_name AND a1.actor_id != a2.actor_id;
```

actor1name	actor_id	actor2name	actor_id
Penelope Guiness	1	Ed Guiness	179
Penelope Guiness	1	Sean Guiness	90
Nick Wahlberg	2	Daryl Wahlberg	95
Ed Chase	3	Jon Chase	176
Jennifer Davis	4	Susan Davis	110
Jennifer Davis	4	Susan Davis	101
Grace Mostel	7	Jim Mostel	99
Matthew Johansson	8	Albert Johansson	146
Matthew Johansson	8	Ray Johansson	64
Zero Cage	11	Johnny Cage	40
Karl Berry	12	Christopher Berry	91
Karl Berry	12	Henry Berry	60

:::

Meryl Allen	194	Kim Allen	145
Meryl Allen	194	Cuba Allen	118
Jayne Silverstone	195	Jeff Silverstone	180
Reese West	197	Christopher West	163
Mary Keitel	198	Greta Keitel	130
Mary Keitel	198	Milla Keitel	74
Julia Fawcett	199	Bob Fawcett	19
Thora Temple	200	Burt Temple	193
Thora Temple	200	Russell Temple	149
Thora Temple	200	Mena Temple	53

(216 rows)

(END)

c. full name matches.

```
dvdrental=# SELECT (a1.first_name || ' ' || a1.last_name) as actor1name, a1.actor_id, (a2.first_n
ame || ' ' || a2.last_name) as actor2name, a2.actor_id FROM actor AS a1, actor AS a2 WHERE a1.fir
st_name=a2.first_name AND a1.last_name=a2.last_name AND a1.actor_id != a2.actor_id;
 actor1name  | actor_id | actor2name  | actor_id
-------------+----------+-------------+----------
 Susan Davis |      101 | Susan Davis |      110
 Susan Davis |      110 | Susan Davis |      101
(2 rows)
```

Q7. Write a query to find the title of films in which "Karl Berry" and “Spencer Depp” worked together.

```
dvdrental=# (SELECT film.title FROM film, film_actor, actor WHERE actor.actor_id=film_actor.actor
_id AND film_actor.film_id=film.film_id AND actor.first_name='Karl' and actor.last_name='Berry')
intersect (SELECT film.title FROM film, film_actor, actor WHERE actor.actor_id=film_actor.actor_i
d AND film_actor.film_id=film.film_id AND actor.first_name='Spencer' and actor.last_name='Depp');

       title
--------------------
 Leathernecks Dwarfs
 Alone Trip
(2 rows)
```

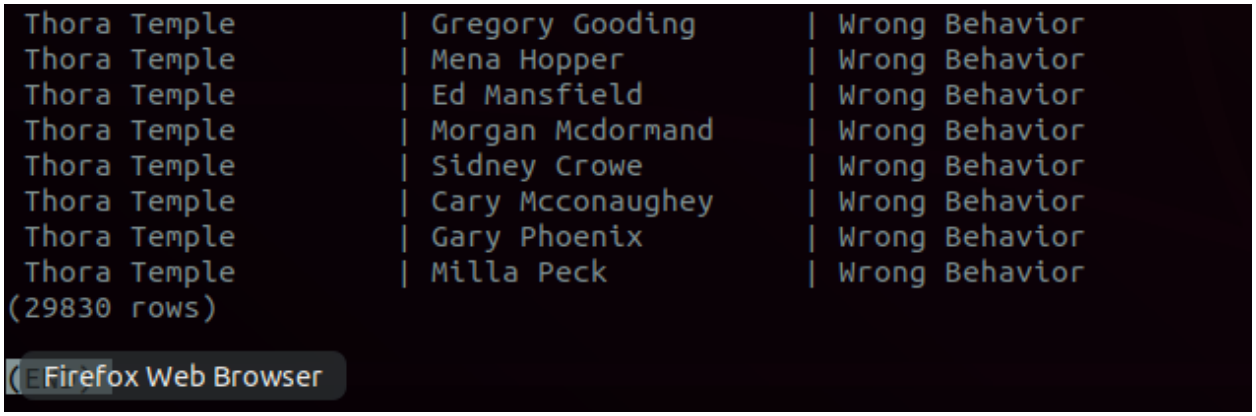
Took all films where Karl Berry worked, took all films where Spencer Depp worked, and took the intersection of both. This works because the intersection will have the list of movies where both Karl Berry worked and Spencer Depp worked.

Q8. Write a query to return all the pairs of actors who have acted in the same movie. Output should look like *(Full name Actor 1, Full name Actor 2, Movie Name)*

```
dvdrental=# SELECT (a1.first_name || ' ' || a1.last_name) as actor1FullName, (a2.first_name || '
' || a2.last_name) as actor2FullName, film.title as movieName FROM actor as a1, actor as a2, film
, film_actor as fa1, film_actor as fa2 WHERE a1.actor_id=fa1.actor_id AND a2.actor_id=fa2.actor_i
d AND fa1.film_id=film.film_id AND fa2.film_id=film.film_id AND a1.actor_id!=a2.actor_id;
```

actor1fullname	actor2fullname	moviename
Penelope Guiness	Mary Keitel	Academy Dinosaur
Penelope Guiness	Rock Dukakis	Academy Dinosaur
Penelope Guiness	Oprah Kilmer	Academy Dinosaur
Penelope Guiness	Warren Nolte	Academy Dinosaur
Penelope Guiness	Mena Temple	Academy Dinosaur
Penelope Guiness	Johnny Cage	Academy Dinosaur
Penelope Guiness	Sandra Peck	Academy Dinosaur
Penelope Guiness	Lucille Tracy	Academy Dinosaur
Penelope Guiness	Christian Gable	Academy Dinosaur
Penelope Guiness	Humphrey Willis	Anaconda Confessions
Penelope Guiness	Jayne Nolte	Anaconda Confessions
Penelope Guiness	Elvis Marx	Anaconda Confessions
Penelope Guiness	Jennifer Davis	Anaconda Confessions

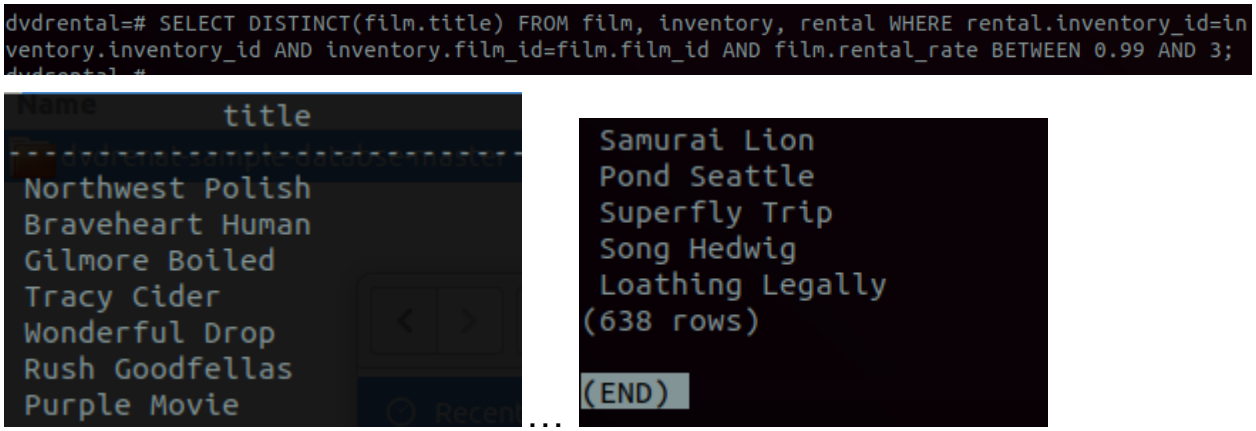
:::



We have to alias both ‘actor’ as a1,a2 and ‘film_actor’ as fa1,fa2. We search for actors whose ids are different and list the same films corresponding to them.

…

Q9. Write a query to find the films which have been rented so far and have rental rates from $0.99 to $3.



…

BETWEEN … AND query is used to select values in a specific range. We used queries on multiple tables, with the condition that the rental’s payment amount is between 0.99 to 3. DISTINCT was used to remove duplicates. We used rented films only because we used rental table in the WHERE clause.

Q10. Write a query to find the title of films offered for rent in store_id = 1 whose film category is “Comedy” and sort these films in descending order.

```
dvdrental=# SELECT DISTINCT(film.title) FROM film_category, film, inventory, category WHERE categ
ory.category_id=film_category.category_id AND film_category.film_id=film.film_id AND film.film_id
=inventory.film_id AND inventory.store_id=1 AND category.name='Comedy' ORDER BY film.title DESC;
```

```
      title
----------------------
 Zorro Ark
 Wisdom Worker
 Velvet Terminator
 Valley Packer
 Tramp Others
 Trainspotting Strangers
 Sweden Shining
 Strictly Scarface
 Snatch Slipper
 Searchers Wait
```

…

```
 Control Anthem
 Connection Microcosmos
 Closer Bang
 Cat Coneheads
 Caper Motions
 Bringing Hysterical
 Anthem Luke
 Airplane Sierra
(49 rows)

(END)
```

We used queries on multiple tables. We had to use DISTINCT so as to remove duplicates. ORDER BY … DESC is the command used to give output in descending order.

…

Q11. Try to delete an actor record with **actor** ID 117, if possible. If not, mention why? Alter the **film_actor** table in such a way that the above operation gets executed. (Hint: Use CASCADE)

```
dvdrental=# DELETE FROM actor WHERE actor_id=117;
ERROR:  update or delete on table "actor" violates foreign key constraint "film_actor_actor_id_fk
ey" on table "film_actor"
DETAIL:  Key (actor_id)=(117) is still referenced from table "film_actor".
```

It is not possible to delete the actor record with actor_id=117. Because it violates foreign key constraint. The table film_actor has its foreign key as actor table's actor_id. Foreign key constraint says that any value in film_actor table's foreign key should be present in actor table as actor_id.

```
dvdrental=# SELECT * FROM FILM_ACTOR WHERE ACTOR_ID=117;
 actor_id | film_id |     last_update
----------+---------+---------------------
      117 |      10 | 2006-02-15 10:05:03
      117 |      15 | 2006-02-15 10:05:03
      117 |     864 | 2006-02-15 10:05:03
      117 |     887 | 2006-02-15 10:05:03
      117 |     926 | 2006-02-15 10:05:03
(33 rows)
```

We see above that actor_id=117 is used at least once in film_actor. Since actor_id is foreign key referring to actor(actor_id), it is not possible to delete actor_id =117 from actor table.

```
dvdrental=# ALTER TABLE film_actor DROP CONSTRAINT "film_actor_actor_id_fkey";
ALTER TABLE
dvdrental=# ALTER TABLE film_actor ADD CONSTRAINT "film_actor_actor_id_fkey"
dvdrental-# FOREIGN KEY (actor_id) REFERENCES actor(actor_id) ON UPDATE CASCADE ON DELETE CASCADE
;
ALTER TABLE
```

Table altered successfully. Now the foreign key will ON DELETE CASCADE.

```
dvdrental=# DELETE FROM actor WHERE actor_id=117;
DELETE 1
```

Delete worked.