

AV3-b

A1) elaborar e fornecer comandos SQL que utilizem os recursos abaixo.

a) consultas aninhadas

----- SELECIONAR OS TÍTULOS DOS FILMES QUE A ATRIZ PENELOPE GUINESS ATUOU

```
SELECT title FROM film
WHERE film.film_id in ( SELECT film_id FROM film_actor
                        WHERE actor_id in (SELECT actor_id FROM actor
                                           WHERE first_name like "PENELOPE" AND last_name like
"GUINESS")
                        );
```

```
mysql> SELECT title FROM film
-> WHERE film.film_id in ( SELECT film_id FROM film_actor
->                        WHERE actor_id in (SELECT actor_id FROM actor
->                                           WHERE first_name like "PENELOPE" AND last_name like "GUINESS")
->                        );
+-----+
| title |
+-----+
| ACADEMY DINOSAUR |
| ANACONDA CONFESSIONS |
| ANGELS LIFE |
| BULWORTH COMMANDMENTS |
| CHEAPER CLYDE |
| COLOR PHILADELPHIA |
| ELEPHANT TROJAN |
| GLEAMING JAWBREAKER |
| HUMAN GRAFFITI |
| KING EVOLUTION |
| LADY STAGE |
| LANGUAGE COWBOY |
| MULHOLLAND BEAST |
| OKLAHOMA JUMANJI |
| RULES HUMAN |
| SPLASH GUMP |
| VERTIGO NORTHWEST |
| WESTWARD SEABISCUIT |
| WIZARD COLDBLOODED |
+-----+
19 rows in set (0,01 sec)

mysql>
```

b) consultas aninhadas correlacionadas

----- SELECIONAR OS ATORES CUJA PRIMEIRA LETRA DO NOME NÃO PERTENCE A PRIMEIRA LETRA DO NOME DE NENHUM PAÍS

```
SELECT *
FROM actor
WHERE NOT EXISTS
      (SELECT country from country
       WHERE LEFT(country,1) = LEFT(first_name,1)
      );
```

```
mysql> SELECT *
-> FROM actor
-> WHERE NOT EXISTS
->     (SELECT country from country
->      WHERE LEFT(country,1) = LEFT(first_name,1)
-> );
```

actor_id	first_name	last_name	last_update
28	WOODY	HOFFMAN	2006-02-15 04:34:33
82	WOODY	JOLIE	2006-02-15 04:34:33
102	WALTER	TORN	2006-02-15 04:34:33
108	WARREN	NOLTE	2006-02-15 04:34:33
119	WARREN	JACKMAN	2006-02-15 04:34:33
140	WHOOPI	HURT	2006-02-15 04:34:33
168	WILL	WILSON	2006-02-15 04:34:33
175	WILLIAM	HACKMAN	2006-02-15 04:34:33

8 rows in set (0,01 sec)

```
mysql>
```

c) exists

----- SELECIONAR ATORES QUE TEM NOMES IGUAIS A NOMES DE CLIENTES

```
SELECT * FROM actor
WHERE EXISTS
  (SELECT first_name, last_name FROM customer
   WHERE first_name=actor.first_name AND last_name=actor.last_name
  );
```

```
mysql>
mysql> SELECT * FROM actor
-> WHERE EXISTS
->     (SELECT first_name, last_name FROM customer
->      WHERE first_name=actor.first_name AND last_name=actor.last_name
-> );
```

actor_id	first_name	last_name	last_update
4	JENNIFER	DAVIS	2006-02-15 04:34:33

1 row in set (0,02 sec)

```
mysql>
```

d) unique ou distinct

----- TABELA TEMPORÁRIA COM OS 10 FILMES MAIS ALUGADOS

```
CREATE TEMPORARY TABLE filmes_mais_alugados
```

```
AS SELECT inventory.film_id
```

```
FROM rental JOIN inventory on rental.inventory_id=inventory.inventory_id
```

```
GROUP BY inventory.film_id
ORDER BY COUNT(*) DESC LIMIT 10;
```

----- NOMES ORDENADOS DOS ATORES QUE ESTAVAM NOS 10 FILMES MAIS ALUGADOS

```
SELECT DISTINCT actor.first_name, actor.last_name
FROM actor, film_actor
WHERE actor.actor_id = film_actor.actor_id AND
      film_actor.film_id in (SELECT * from filmes_mais_alugados) ORDER BY first_name
ASC;
```

```
mysql> CREATE TEMPORARY TABLE filmes_mais_alugados
-> AS SELECT inventory.film_id
-> FROM rental JOIN inventory on rental.inventory_id=inventory.inventory_id
-> GROUP BY inventory.film_id
-> ORDER BY COUNT(*) DESC LIMIT 10;
Query OK, 10 rows affected (0,04 sec)
Records: 10 Duplicates: 0 Warnings: 0

mysql> SELECT DISTINCT actor.first_name, actor.last_name
-> FROM actor, film_actor
-> WHERE actor.actor_id = film_actor.actor_id AND
->      film_actor.film_id in (SELECT * from filmes_mais_alugados) ORDER BY first_name ASC;
+-----+-----+
| first_name | last_name |
+-----+-----+
| ANGELA     | HUDSON    |
| BURT       | TEMPLE    |
| CAMERON    | ZELLWEGER |
| CHARLIZE   | DENCH     |
| CHRIS      | BRIDGES   |
| CHRIS      | DEPP      |
| CHRISTIAN  | NEESON    |
| CUBA       | ALLEN     |
| DARYL      | CRAWFORD  |
| DUSTIN     | TAUTOU    |
| ED         | GUINNESS  |
| ELVIS      | MARX      |
| EWAN       | GOODING   |
| GARY       | PHOENIX   |
| GEOFFREY   | HESTON    |
| GREG       | CHAPLIN   |
| GROUCHO    | SINATRA   |
| IAN        | TANDY     |
| JADA       | RYDER     |
| JAYNE      | SILVERSTONE |
| JOHNNY     | LOLLOBRIGIDA |
| JUDE       | CRUISE    |
| JUDY       | DEAN      |
| JULIA      | FAWCETT   |
| JULIANNE   | DENCH     |
| KEVIN      | GARLAND   |
| KIRSTEN    | AKROYD    |
| LAURA     | BRODY     |
| LISA       | MONROE    |
| MATTHEW    | LEIGH     |
| MERYL      | GIBSON    |
| MERYL      | ALLEN     |
| MICHAEL    | BOLGER    |
| MILLA      | PECK      |
| MORGAN     | WILLIAMS  |
| NICK       | DEGENERES |
| PENELOPE   | MONROE    |
| PENELOPE   | PINKETT   |
| REESE      | KILMER    |
| REESE      | WEST      |
+-----+-----+
```

```

| RENEE      | TRACY      |
| RIP        | CRAWFORD   |
| RIP        | WINSLET    |
| SALMA      | NOLTE      |
| SEAN       | WILLIAMS   |
| TIM        | HACKMAN    |
| TOM        | MIRANDA    |
| VIVIEN     | BERGEN     |
| WALTER     | TORN       |
| WARREN     | JACKMAN    |
| WHOOPI    | HURT       |
+-----+-----+
51 rows in set (0,00 sec)

mysql>

```

e) junções naturais

----- SELECIONAR TODOS OS ATORES DE LADY STAGE

```

SELECT film_actor.film_id, film.title,
       film_actor.actor_id, actor.first_name, actor.last_name
from (actor join film_actor on actor.actor_id=film_actor.actor_id) join film on
film.film_id=film_actor.film_id
WHERE film.title like "LADY STAGE";

```

```

mysql> SELECT film_actor.film_id, film.title,
->       film_actor.actor_id, actor.first_name, actor.last_name
-> from (actor join film_actor on actor.actor_id=film_actor.actor_id) join film on film.film_id=film_actor
.film_id
-> WHERE film.title like "LADY STAGE";
+-----+-----+-----+-----+-----+
| film_id | title      | actor_id | first_name | last_name |
+-----+-----+-----+-----+-----+
| 506     | LADY STAGE | 1        | PENELOPE   | GUINNESS  |
| 506     | LADY STAGE | 48       | FRANCES    | DAY-LEWIS |
| 506     | LADY STAGE | 49       | ANNE       | CRONYN    |
| 506     | LADY STAGE | 64       | RAY        | JOHANSSON |
| 506     | LADY STAGE | 104      | PENELOPE   | CRONYN    |
| 506     | LADY STAGE | 115      | HARRISON   | BALE      |
| 506     | LADY STAGE | 180      | JEFF       | SILVERSTONE |
| 506     | LADY STAGE | 188      | ROCK       | DUKAKIS   |
+-----+-----+-----+-----+-----+
8 rows in set (0,00 sec)

mysql>

```

f) junção left ou right

----- QUANTIDADE DE FILMES QUE NÃO TEM NA LOJA 2

```

SELECT COUNT(DISTINCT film.film_id) as qtd_filmes
FROM film LEFT JOIN inventory on inventory.film_id = film.film_id AND inventory.store_id=2
WHERE inventory.store_id IS NULL;

```

```
mysql> SELECT COUNT(DISTINCT film.film_id) as qtd_filmes
-> FROM film LEFT JOIN inventory on inventory.film_id = film.film_id AND inventory.store_id=2
-> WHERE inventory.store_id IS NULL;
+-----+
| qtd_filmes |
+-----+
|          238 |
+-----+
1 row in set (0,01 sec)

mysql>
```

g) agregação

----- SELECIONAR O NÚMERO DE COMPRAS E O VALOR MÉDIO PAGO PELOS
CLIENTES NA LOJA 1

```
SELECT COUNT(amount) as num_compras , AVG(amount) as valor_medio_pago
FROM payment, staff
WHERE payment.staff_id=staff.staff_id AND staff.store_id=1;
```

```
mysql> SELECT COUNT(amount) as num_compras , AVG(amount) as valor_medio_pago
-> FROM payment, staff
-> WHERE payment.staff_id=staff.staff_id AND staff.store_id=1;
+-----+-----+
| num_compras | valor_medio_pago |
+-----+-----+
|          8054 |          4.157251 |
+-----+-----+
1 row in set (0,02 sec)

mysql>
```

h) group by

----- EXIBIR OS FILMES DE MAIOR DURAÇÃO DE CADA CATEGORIA

```
SELECT category.name, film.title, film.length
FROM (category JOIN film_category on category.category_id=film_category.category_id)
JOIN film on film.film_id=film_category.film_id
WHERE (category.name, film.length) in
(SELECT category.name, MAX(film.length)
FROM (category JOIN film_category on
category.category_id=film_category.category_id)
JOIN film on film.film_id=film_category.film_id
GROUP BY category.name);
```

```
mysql> SELECT category.name, film.title, film.length
-> FROM (category JOIN film_category on category.category_id=film_category.category_id)
-> JOIN film on film.film_id=film_category.film_id
-> WHERE (category.name, film.length) in
-> (SELECT category.name, MAX(film.length)
-> FROM (category JOIN film_category on category.category_id=film_category.category_id)
-> JOIN film on film.film_id=film_category.film_id
-> GROUP BY category.name);
```

name	title	length
Action	DARN FORRESTER	185
Action	WORST BANGER	185
Animation	GANGS PRIDE	185
Animation	POND SEATTLE	185
Children	FURY MURDER	178
Children	WRONG BEHAVIOR	178
Classics	CONSPIRACY SPIRIT	184
Comedy	CONTROL ANTHEM	185
Documentary	WIFE TURN	183
Documentary	YOUNG LANGUAGE	183
Drama	JACKET FRISCO	181
Family	KING EVOLUTION	184
Foreign	CRYSTAL BREAKING	184
Foreign	SORORITY QUEEN	184
Games	CHICAGO NORTH	185
Horror	ANALYZE HOOSIERS	181
Horror	LOVE SUICIDES	181
Music	HOME PITY	185
New	FRONTIER CABIN	183
Sci-Fi	SOLDIERS EVOLUTION	185
Sports	SMOOCHY CONTROL	184
Travel	MUSCLE BRIGHT	185
Travel	SWEET BROTHERHOOD	185

```
23 rows in set (0,03 sec)

mysql>
```

A2) fornecer o comando de criação de uma visão;

----- VIEW COM A QUANTIDADE DE CLIENTES DE CADA PAÍS DO MUNDO

```
CREATE VIEW cliente_por_pais
AS SELECT country.country, COUNT(*) as num_clientes
FROM (customer JOIN address on customer.address_id = address.address_id)
JOIN city on address.city_id=city.city_id
JOIN country on country.country_id=city.country_id
GROUP BY country.country_id;
```

```

mysql> CREATE VIEW cliente_por_pais
-> AS SELECT country.country, COUNT(*) as num_clientes
-> FROM (customer JOIN address on customer.address_id = address.address_id)
->      JOIN city on address.city_id=city.city_id
->      JOIN country on country.country_id=city.country_id
-> GROUP BY country.country_id;
Query OK, 0 rows affected (0,01 sec)

mysql> SELECT * FROM CLIENTE_POR_PAIS;
ERROR 1146 (42S02): Table 'sakila.CLIENTE_POR_PAIS' doesn't exist
mysql> DROP VIEW cliente_por_pais
-> ;
Query OK, 0 rows affected (0,02 sec)

mysql> CREATE VIEW cliente_por_pais
-> AS SELECT country.country, COUNT(*) as num_clientes
-> FROM (customer JOIN address on customer.address_id = address.address_id)
->      JOIN city on address.city_id=city.city_id
->      JOIN country on country.country_id=city.country_id
-> GROUP BY country.country_id;
Query OK, 0 rows affected (0,01 sec)

mysql> SELECT * FROM cliente_por_pais;
+-----+-----+
| country                                | num_clientes |
+-----+-----+
| Afghanistan                           | 1            |
| Algeria                               | 3            |
| American Samoa                        | 1            |
| Angola                                | 2            |
| Anguilla                              | 1            |
| Argentina                             | 13           |
| Armenia                               | 1            |
| Austria                               | 3            |
| Azerbaijan                            | 2            |
| Bahrain                               | 1            |
| Bangladesh                            | 3            |
| Belarus                               | 2            |
| Bolivia                               | 2            |
| Brazil                                | 28           |
| Brunei                                | 1            |
| Bulgaria                              | 2            |
| Cambodia                              | 2            |
| Cameroon                              | 2            |
| Canada                                | 5            |
| Chad                                  | 1            |
| Chile                                 | 3            |
| China                                 | 53           |
| Colombia                              | 6            |
| Congo, The Democratic Republic of the | 2            |
| Czech Republic                        | 1            |
| Dominican Republic                    | 3            |
| Ecuador                               | 3            |
| Egypt                                 | 6            |
| Estonia                               | 1            |

```

A3) elaborar um programa (sugestão: em python) que percorra os registros obtidos por um dos comandos SQL's que foi criado nos itens "a" a "h".

#Percorrer registros obtidos pelo comando SQL do item e

```
import mysql.connector
```

```
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
```

```

        password="Iamon_14",
        database="sakila"
    )

mycursor = mydb.cursor()

mycursor.execute("SELECT film_actor.film_id, film.title, \
        film_actor.actor_id, actor.first_name, actor.last_name \
from (actor join film_actor on actor.actor_id=film_actor.actor_id) join film on \
film.film_id=film_actor.film_id \
WHERE film.title like 'LADY STAGE'")

myresult = mycursor.fetchall()

print("ATORES DE LADY STAGE: \n")
for x in myresult:
    print("- ", x[3], x[4])

```

```

#Percorrer registros obtidos pelo comando SQL do item e
import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Iamon_14",
    database="sakila"
)

mycursor = mydb.cursor()

mycursor.execute("SELECT film_actor.film_id, film.title, \
        film_actor.actor_id, actor.first_name, actor.last_name \
from (actor join film_actor on actor.actor_id=film_actor.actor_id) join film on film.film_id=film_actor.film_id \
WHERE film.title like 'LADY STAGE'")

myresult = mycursor.fetchall()

print("ATORES DE LADY STAGE: \n")
for x in myresult:
    print("- ", x[3], x[4])
monique@monique-Aspire:~/Documentos/BD$ python3 programa.py
ATORES DE LADY STAGE:
- PENELOPE GUINNESS
- FRANCES DAY-LEWIS
- ANNE CRONYN
- RAY JOHANSSON
- PENELOPE CRONYN
- HARRISON BALE
- JEFF SILVERSTONE
- ROCK DUKAKIS
monique@monique-Aspire:~/Documentos/BD$

```

FIM