CURSO TÉCNICO EM INFORMÁTICA
INTEGRADO AO ENSINO MÉDIO

TRABALHO DE BDD
Grupo H – 4ºBim

GEOVANNA DA SILVA LIMA - SP3029034
GIULIA SANTANA DOS ANJOS – SP3025918
IGOR DOMINGOS DA SILVA MOZETIC - SP3027422
JULIA ROMUALDO PEREIRA – SP3023061

SÃO PAULO
2021

1. Lista de todos os dados de todos os filmes

```
SELECT *
FROM film
```

2. Lista do título de todos os filmes

```
SELECT title
FROM film
```

3. Lista de todos os dados dos filmes com duração menor do que 60 minutos

```
SELECT *
FROM film
WHERE length < 60
```

4. Quantidade total de filmes

```
SELECT COUNT(film_id)
AS QntDeFilme
FROM film;
```

5. Duração média dos filmes

```
SELECT AVG (length)
FROM film;
```

6. Lista de todos os dados dos clientes inativos

```
SELECT *
FROM customer
WHERE active = 0
```

7. Lista com o primeiro, último nome e endereço dos clientes ativos (PODE usar join)

```
SELECT first_name, last_name, address
FROM customer c
LEFT JOIN address a
on c.customer_id = a.address_id
WHERE active = 1
```

8. Lista com o primeiro, último nome, endereço e país dos clientes residentes no Brasil. (PODE usar join)

```
SELECT first_name, last_name, address, country
FROM country t, customer c
```

```
LEFT JOIN address a
ON c.customer_id = a.address_id
WHERE t.country = "Brazil"
```

9. Relação de filmes e atores que atuaram no filme, contendo nome do filme, primeiro e último nome dos atores. (DEVE usar join)

```
SELECT f.title, a.first_name, a.last_name
FROM film f
RIGHT JOIN film_actor fa ON f.film_id = fa.film_id
JOIN actor a on a.actor_id = fa.actor_id
```

10. Relação de filmes e atores que atuaram no filme, contendo nome do filme, primeiro e último nome dos atores, ordenada por filme. (DEVE usar join)

```
SELECT f.title, a.first_name, a.last_name
FROM film f
RIGHT JOIN film_actor fa ON f.film_id = fa.film_id
JOIN actor a on a.actor_id = fa.actor_id
ORDER by title
```

11. Relação de filmes e atores que atuaram no filme, contendo nome do filme, primeiro e último nome dos atores, ordenada por ator. (DEVE usar join)

```
SELECT f.title, a.first_name, a.last_name
FROM film f
RIGHT JOIN film_actor fa ON f.film_id = fa.film_id
JOIN actor a on a.actor_id = fa.actor_id
ORDER by first_name
```

12. Relação de filmes com participação de um ator específico.

```
SELECT f.title, a.first_name, a.last_name
FROM film f
RIGHT JOIN film_actor fa ON f.film_id = fa.film_id
JOIN actor a on a.actor_id = fa.actor_id
WHERE a.first_name = "ED" and a.last_name = "CHASE"
```

13. Lista de filmes por categoria contendo o nome do filme e da categoria

```
SELECT f.title, c.name
FROM film f
RIGHT JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c on c.category_id = fc.category_id
```

14. Quantidade de filmes por categoria contendo o nome da categoria e a quantidade.

```
SELECT COUNT(f.title) AS "Quantidade", c.name AS "Categoria"
```

```
FROM film f
RIGHT JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c on c.category_id = fc.category_id
GROUP BY c.name
```

15. Duração média dos filmes por categoria contendo o nome da categoria e a média.

```
SELECT c.name AS "Categoria", AVG (f.length) AS "Média"
FROM film f
RIGHT JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c on c.category_id = fc.category_id
GROUP BY c.name
```

16. Quantidade de filmes por categoria das categorias com menos de 57 filmes, contendo o nome da categoria e a quantidade.

```
SELECT COUNT(f.film_id) as "Qnt de Filmes/Categoria", c.name as "Nome da Categoria"
FROM film f
RIGHT join film_category fc on f.film_id = fc.film_id
JOIN category c on c.category_id = fc.category_id
GROUP by c.category_id
HAVING COUNT(f.film_id)< 57
```

17. Duração média dos filmes por categoria das categorias com menos de 57 filmes, contendo o nome da categoria, a quantidade e a média

```
SELECT COUNT(f.film_id) as "Qnt de Filmes/Categoria", c.name as "Nome da Categoria", AVG (f.length) AS "Média"
FROM film f
RIGHT join film_category fc on f.film_id = fc.film_id
JOIN category c on c.category_id = fc.category_id
GROUP by c.category_id
HAVING COUNT(f.film_id)< 57
```

18. Quantidade de filmes alugados por cliente, contendo o primeiro e último nome do cliente e a contagem.

```
SELECT c.first_name AS "Primeiro Nome", c.last_name AS "Último Nome", COUNT(f.film_id) AS "Qnt Filmes Alugados"
FROM film f
JOIN inventory i ON i.film_id = f.film_id
JOIN rental r ON r.inventory_id = i.inventory_id
JOIN customer c ON c.customer_id = r.customer_id
GROUP BY first_name
```

19. Quantidade de filmes alugados por cliente em ordem decrescente de quantidade de filmes alugados, contendo o primeiro e último nome do cliente e a contagem.

```
SELECT c.first_name AS "Primeiro Nome", c.last_name AS "Último Nome", COUNT(f.film_id) AS "Qnt Filmes Alugados"
FROM film f
JOIN inventory i ON i.film_id = f.film_id
JOIN rental r ON r.inventory_id = i.inventory_id
JOIN customer c ON c.customer_id = r.customer_id
GROUP BY f.film_id
ORDER BY 3 DESC
```

20. Relação do primeiro e último nome dos clientes que possuem um filme alugado no momento

```
SELECT c.first_name AS "Primeiro Nome", c.last_name AS "Último Nome"
FROM customer c
JOIN rental r on r.customer_id = c.customer_id
WHERE r.return_date IS NULL
GROUP BY first_name
```

21. Relação do primeiro e último nome dos clientes que não possuem um filme alugado no momento

```
SELECT c.first_name AS "Primeiro Nome", c.last_name AS "Último Nome"
FROM customer c
WHERE NOT EXISTS (
        SELECT 1
        FROM rental r
WHERE c.customer_id = r.customer_id AND r.return_date IS NOT NULL);
```

22. Quantidade de clientes que moram no mesmo distrito, contendo o nome do distrito e a quantidade

```
SELECT COUNT(c.customer_id) AS "Qnt Clientes", a.district as "Nome Distito"
FROM customer c
LEFT JOIN address a on c.address_id = a.address_id
GROUP BY a.district
```

23. Relação com os nomes das línguas que não tem filmes ligados à ela

```
SELECT l.name AS "Língua"
FROM language l
WHERE NOT EXISTS
(SELECT f.language_id
 FROM film f
 WHERE f.language_id = l.language_id);
```

24. Crie uma view chamada “film_replacement_cost”, que mostre todos os dados dos filmes que tem um custo de reposição maior ou igual a 20 dólares.

```
CREATE VIEW film_replacement_cost AS
SELECT *
FROM film
WHERE replacement_cost >= 20
```

25. Crie uma procedure chamada “film_rental_rate_adjustment”, que receba como parâmetro um valor numérico e aumente o valor do aluguel dos filmes em um percentual baseado no valor recebido.

```
DELIMITER $$
CREATE PROCEDURE film_rental_rate_adjustment(IN aumento INT)
BEGIN
	UPDATE film
	SET film.rental_rate = film.rental_rate +( film.rental_rate*(aumento/100));
END $$
DELIMITER ;
```

```
CALL film_rental_rate_adjustment(100)
```

26. Crie uma trigger chamada “costumer_rental_count” onde sempre que um registro for adicionado na tabela “rental” será acrescido +1 à coluna “total_rentals” na tupla referente ao cliente (customer). Para isso você deverá criar a coluna “total_rentals” na tabela costumer. Essa coluna deverá armazenar um valor inteiro.
* nesse exercício você deve mandar o comando de criação da coluna e da Trigger.

```
ALTER TABLE customer
ADD COLUMN total_rentals int(5)
```

```
UPDATE customer SET total_rentals = 0;
```

```
DELIMITER $$
CREATE TRIGGER customer_rental_count
AFTER INSERT ON rental
FOR EACH ROW
BEGIN
	UPDATE customer SET total_rentals = total_rentals + 1
	WHERE customer_id = new.customer_id;
END $$
DELIMITER ;
```

27. Crie uma view ou um procedure que realize uma consulta que retorne os nomes e sobrenomes dos atores da seguinte maneira:

Nome	Sobrenome	Nome Completo
GUSTAVO	ABREU	gUstAvO AbrEU
MANE	GARRINCHA	mAnE gArrInchA
STEVE	ROGERS	stEvE rOgErs

```
CREATE VIEW desafio AS
SELECT first_name AS "Nome", last_name AS "Sobrenome",
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(CONCAT(LOWER(first_name),
' ', LOWER(last_name)),'a','A'),'e','E'),'i','I'),'o','O'),'u','U')  AS "Nome Completo"
FROM actor;
```

```
//*Comando com as tabelas Nome e Sobrenome separado*//
CREATE VIEW desafio AS
SELECT first_name AS "Nome", last_name AS "Sobrenome",
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE((LOWER(first_name)),
'a','A'),'e','E'),'i','I'),'o','O'),'u','U')  AS "Nome",
REPLACE(REPLACE(REPLACE(REPLACE(REPLACE((LOWER(last_name)),'
a','A'),'e','E'),'i','I'),'o','O'),'u','U')  AS "Sobrenome"
FROM actor;
```