Considere as seguintes afirmações sobre restrições de integridade em bancos de dados relacionais utilizando o MySQL:

I. A restrição `PRIMARY KEY` garante que uma coluna ou conjunto de colunas tenha valores únicos e não nulos.

II. A restrição `FOREIGN KEY` garante que um valor em uma coluna de uma tabela referencie um valor válido em uma coluna correspondente de outra tabela.

III. A restrição `UNIQUE` garante que todos os valores em uma coluna sejam únicos e não permite valores nulos.

IV. A restrição `CHECK` permite definir condições específicas que os valores de uma coluna devem satisfazer.

Está(ão) correta(s) apenas a(s) afirmação(ões):

- ○ I, II e III
- ○ III
- ○ II e III
- ○ I, II e IV
- ○ I e II

ALTERNATIVA: D

2 Múltipla resposta 1 ponto

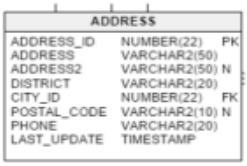
São expressões válidas da álgebra relacional (assinale todas as que forem válidas)

- ☐ π cpf (π nome,cpf (Empregado))
- ☐ π nome,salario (σ idade>40 (Empregado))
- ☐ π dt_nascimento,nome (π data_nascimento (Cliente))
- ☐ π nome,salario (σ salario>10000 (Empregado))
- ☐ σ cpf=099.099.099-10 (π nome,salario(Empregado))

ALTERNATIVAS: A, B, D

3 Múltipla escolha 1 ponto

Considere a seguinte relação:



ADDRESS		
ADDRESS_ID	NUMBER(22)	PK
ADDRESS	VARCHAR2(50)	
ADDRESS2	VARCHAR2(50)	N
DISTRICT	VARCHAR2(20)	
CITY_ID	NUMBER(22)	FK
POSTAL_CODE	VARCHAR2(10)	N
PHONE	VARCHAR2(20)	
LAST_UPDATE	TIMESTAMP	

Assinale a alternativa correta com relação ao registro de telefones (PHONE) nessa relação:

- ◯ Para que o registro fique correto basta alterar para que não permita o registro de múltiplos telefones para um endereço no campo PHONE da relação ADDRESS
- ◯ Para que o registro fique correto basta alterar para que permita o registro de múltiplos telefones para um endereço no campo PHONE da relação ADDRESS
- ◯ Para permitir o registro de múltiplos telefones, deveremos excluir o campo PHONE da tabela ADDRESS e adicionar o CAMPO PHONE_ID como chave estrangeira com o ID de uma nova tabela PHONE
- ◯ O registro não está correto porque permite o registro de múltiplos telefones para um endereço no campo PHONE da relação ADDRESS
- ◯ Para permitir o registro de múltiplos telefones, deveremos excluir o campo PHONE da tabela ADDRESS e adicionar uma tabela PHONE com uma chave estrangeira ADDRESS_ID com o id do endereço que é relacionado ao telefone
- ◯ O registro está correto e permite o registro de múltiplos telefones para um endereço no campo PHONE da relação ADDRESS

ALTERNATIVA: D

4 Múltipla escolha 1 ponto

O seguinte modelo foi desenvolvido para uma loja para registro de itens dos pedidos:

IdItemPedido (PK)	IdPedido (FK)	IdProduto (FK)	Qtde	PrecoItem	TotalItem

Considerando as especificações da 1ªFN, 2ªFN e 3ªFN é possível afirmar:

I. Que a tabela está de acordo com a 1ªFN

II. Que a tabela está de acordo com a 2ªFN

III. Que a tabela está de acordo com a 3ªFN

IV. Que a tabela não atende nem a 1ªFN, nem a 2ªFN e nem 3ªFN

Indique a alternativa que apresenta a(s) afirmação(ões) correta(s):

- ◯ IV
- ◯ I e III
- ◯ II e III
- ◯ I e II
- ◯ I, II e III

ALTERNATIVA: D

5 Múltipla escolha 1 ponto

Sobre as formas normais (NF) em um banco de dados relacional, considere as seguintes afirmações:

I. Uma tabela está na 1ª Forma Normal (1NF) quando todos os atributos contêm valores atômicos.

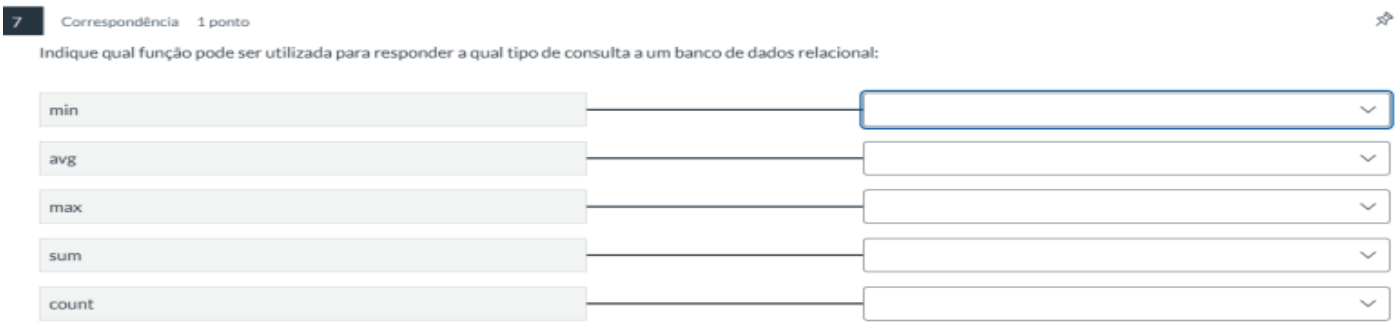
II. Uma tabela está na 2ª Forma Normal (2NF) quando não possui dependências parciais.

III. Uma tabela está na 3ª Forma Normal (3NF) quando todos os atributos não chave dependem da chave primária inteira e não dependem transitivamente de outros atributos não chave.

Está(ão) correta(s) apenas a(s) afirmação(ões):

- ◯ II e III
- ◯ III
- ◯ I, II e III
- ◯ I e II
- ◯ II

ALTERNATIVA: C



MIN –---- MENOR VALOR

AVG –---- MÉDIA

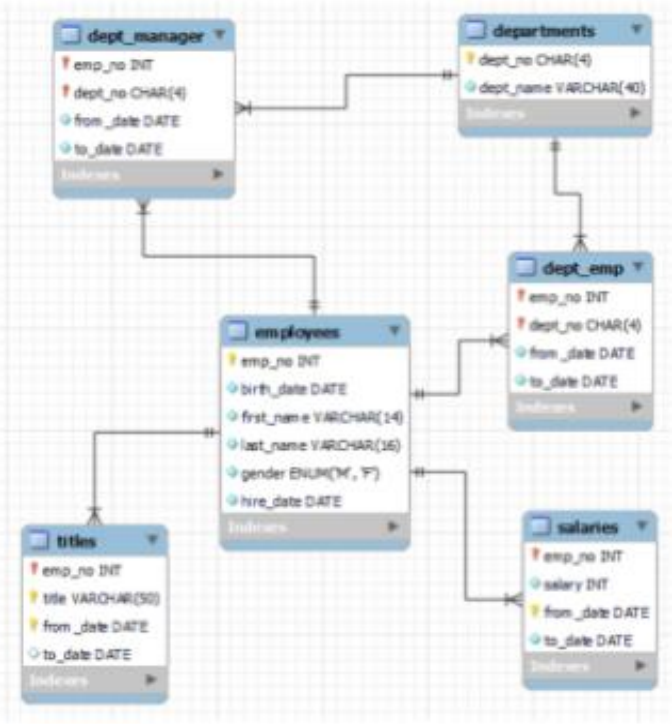
MAX –---- MAIOR VALOR

SUM –---- SOMA

COUNT –---- CONTAGEM DE LINHAS

6 Múltipla escolha 1 ponto

Na empresa Igualdade S/A, é feito anualmente o levantamento dos salários pagos por função e gênero. O modelo de dados utilizado na empresa para registro das informações sobre posição e pagamentos encontra-se abaixo, em que os salários são registrados anualmente:



Qual a instrução SQL adequada para:

Relacionar as médias salariais pagas por posição e gênero no ano de 2020 do maior para o menor?

- select t.title, e.gender, avg(s.salary) as media_salarial
  from salaries s
  INNER JOIN employees e ON s.emp_no = e.emp_no
  INNER JOIN titles t ON t.emp_no = e.emp_no
  WHERE s.from_date > "2020-01-01" and s.to_date < "2020-12-31"
  group by t.title,e.gender order by media_salarial desc;
- select e.name, e.gender, sum(s.salary) as media_salarial
  from salaries s
  INNER JOIN employees e ON s.emp_no = e.emp_no
  INNER JOIN titles t ON t.emp_no = e.emp_no
  WHERE s.from_date > "2020-01-01" and s.to_date < "2020-12-31"
  group by t.title,e.gender order by media_salarial desc;
- select t.title, e.gender, sum(s.salary) as media_salarial
  from salaries s
  INNER JOIN employees e ON s.emp_no = e.emp_no
  INNER JOIN titles t ON t.emp_no = e.emp_no
  WHERE s.from_date > "2020-01-01" and s.to_date < "2020-12-31"
  group by t.title,e.gender order by t.title,media_salarial desc;
- select t.title, e.gender, avg(s.salary) as media_salarial
  from salaries s
  INNER JOIN employees e ON s.emp_no = e.emp_no
  INNER JOIN titles t ON t.emp_no = e.emp_no
  WHERE s.from_date > "2020-01-01" and s.to_date < "2020-12-31"
  group by t.title,e.gender order by media_salarial asc;
- select t.title, e.gender, avg(s.salary) as media_salarial
  from salaries s
  INNER JOIN employees e ON s.emp_no = e.emp_no
  INNER JOIN titles t ON t.emp_no = e.emp_no
  WHERE s.from_date > "2020-01-01" and s.to_date < "2020-12-31"
  group by t.title,e.gender order by t.title,media_salarial desc;

ALTERNATIVA: A

8 Múltipla escolha 1 ponto

Um dos principais objetivos da modelagem conceitual é evitar redundância e inconsistência no banco de dados. Considerando os princípios da modelagem conceitual, avalie as seguintes afirmativas:

I. **Normalização:** A normalização é um processo de dividir as relações em várias menores, de modo a eliminar redundância e anomalias de atualização.

II. **Chave Primária:** A chave primária é um atributo ou conjunto de atributos que identifica unicamente cada tupla em uma relação.

III. **Chave Estrangeira:** A chave estrangeira é um atributo ou conjunto de atributos que referencia a chave primária de outra relação.

IV. **Integridade Referencial:** A integridade referencial garante que cada valor de chave estrangeira esteja presente na chave primária da relação referenciada.

Com base na análise das afirmativas, assinale a alternativa que apresenta os princípios que garantem maior **qualidade** para a modelagem conceitual:

- ◯ I, II e IV, apenas.
- ◯ I, II e III, apenas.
- ◯ II, III e IV, apenas.
- ◯ Nenhuma das afirmativas garante a qualidade da modelagem conceitual.
- ◯ I, II, III e IV

ALTERNATIVA: E

9 Múltipla escolha 1 ponto

Considere a seguinte relação:

idItem	idProduto	qtde	valorItem	totalItem
902091	109201	3	27,00	81,00
902093	453092	1	109,00	109,00

Considerando essa relação, é correto afirmar:

- ◯ Ela não atende a segunda forma normal.
- ◯ Para que ela atenda a primeira forma normal, basta excluir o atributo quantidade (qtde) que é desnecessário.
- ◯ Ela não atende a terceira forma normal.
- ◯ Ela atende a primeira, segunda e terceira formas normais.
- ◯ Ela não atende a primeira forma normal.

ALTERNATIVA: C

10 Múltipla resposta 1 ponto

Para responder à questão, considere a seguinte tabela:

Assinale **todas** as afirmações corretas:

- [ ] SELECT t1.title, t2.title, t1.length FROM film t1 INNER JOIN film t2 ON t1.film_id <> t2.film_id AND t1.title = t2.title;

  Seleciona filmes diferentes com o mesmo título.
- [ ] SELECT t1.title, t2.title, t1.replacement_cost FROM film t1 INNER JOIN film t2 ON t1.film_id <> t2.film_id AND t1.replacement_cost = t2.replacement_cost;

  Seleciona filmes diferentes com o mesmo preço de reposição.
- [ ] A instrução SQL:

  SELECT t1.title, t2.title, t1.length FROM film t1 INNER JOIN film t2 ON t1.title = t2.title AND t1.length <> t2.length;

  Seleciona filmes com mesmo título e durações diferentes.
- [ ] SELECT t1.title, t2.title, t1.length FROM film t1 INNER JOIN film t2 ON t1.film_id <> t2.film_id AND t1.length = t2.length;

  Seleciona filmes com mesmo título e mesma duração

ALTERNATIVAS: A, B, C

11 Múltipla resposta 1 ponto

Considere a seguinte instrução SQL:

```
SELECT idProduto, nomeProduto, nomeCategoria
FROM Produtos
INNER JOIN Categorias USING(idCategoria)
```

Para que ela possa ser utilizada, são condições: (assinale todas que se aplicarem)

- [ ] Que a relação Categorias possua uma chave primária idCategoria
- [ ] Que as chaves primária e estrangeira tenham o mesmo nome em ambas as tabelas
- [ ] Que a relação Produtos possua uma chave estrangeira idCategoria
- [ ] Que não haja um campo idCategoria na relação Produtos
- [ ] Que haja um atributo nomeCategoria na relação Produtos

ALTERNATIVAS: A, B, C

12 Múltipla escolha 1 ponto

Considere a seguinte relação:

idCliente	NomeCompleto	CPF	DataCadastro	Documentos
12	Carla Silva	022.022.022-11	12/01/2023	12.090.901-1, OAB-560.989-3

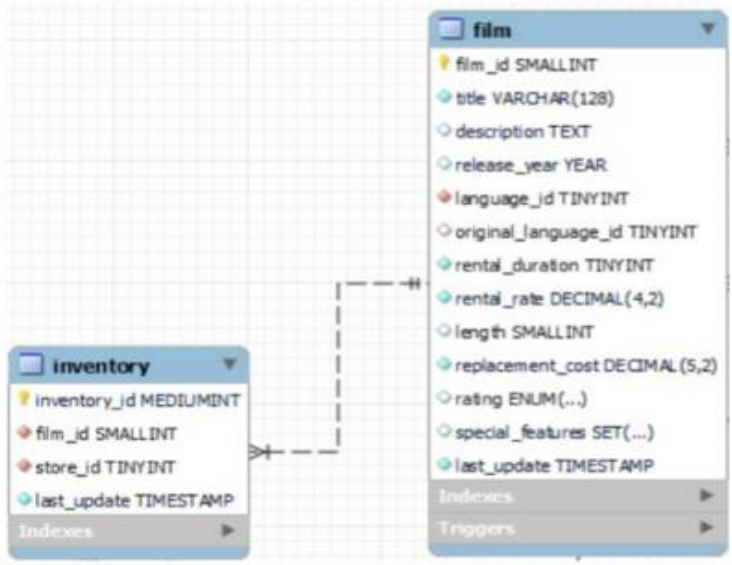
- ( ) Para que seja normalizada, sem perda de informação, basta excluir a coluna documento.
- ( ) Ela está de acordo com a 2FN (atende a 2ª forma normal)
- ( ) Para que seja normalizada, sem perda de informação, deve ser criada uma tabela documentos, com chave estrangeira idCliente e com a inclusão de um registro para tipo de documento e número de documento.
- ( ) Ela está de acordo com a 3FN (atende a 3ª forma normal)
- ( ) Ela está de acordo com a 1FN (atende a 1ª forma normal)

Anterior Próximo

ALTERNATIVA: C

13 Múltipla resposta 1 ponto

Utilizando as relações indicadas a seguir:



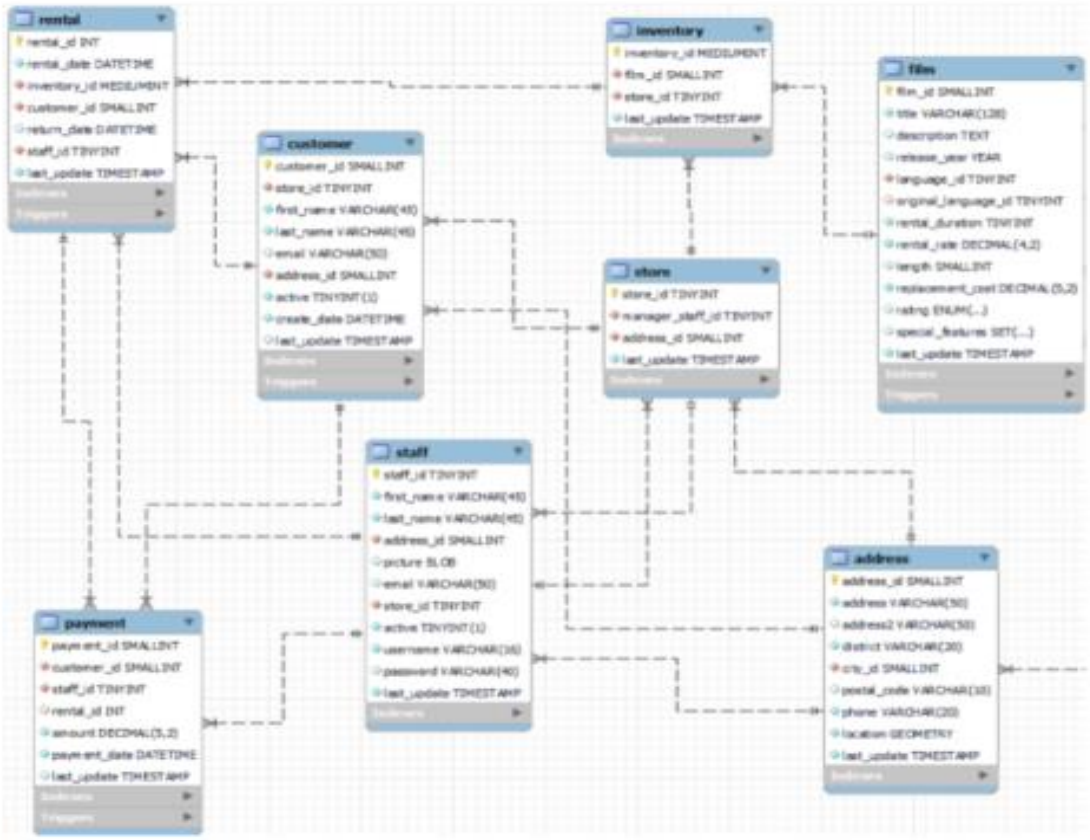
Qual(is) a(s) instrução(ões) adequada(s) para encontrar filmes sem estoque?

- [ ] SELECT film.film_id, title, inventory_id FROM film LEFT JOIN inventory ON inventory.film_id = film.film_id WHERE inventory.film_id IS NULL ORDER BY title
- [ ] SELECT film.film_id, title, inventory_id FROM inventory RIGHT JOIN film ON inventory.film_id = film.film_id WHERE film.film_id IS NULL ORDER BY title
- [ ] SELECT film.film_id, title, inventory_id FROM film LEFT JOIN inventory ON inventory.film_id = film.film_id WHERE film.film_id IS NULL ORDER BY title
- [ ] SELECT film.film_id, title, inventory_id FROM inventory RIGHT JOIN film ON inventory.film_id = film.film_id WHERE inventory.film_id IS NULL ORDER BY title

ALTERNATIVA: A

14 Múltipla escolha 1 ponto

Considere as seguintes tabelas do modelo de banco de dados sakila:



Qual das alternativas apresenta a instrução SQL correta para:

- Encontrar todos os clientes que fizeram locações na loja com endereço na "Rua Antonio José da Silva, 33".

◯
```
SELECT DISTINCT c.first_name, c.last_name
FROM customer c
JOIN rental r ON c.customer_id = r.customer_id
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN store s ON i.store_id = s.store_id
JOIN address a ON s.address_id = a.address_id
WHERE a.address = 'Rua Antonio José da Silva, 33';
```

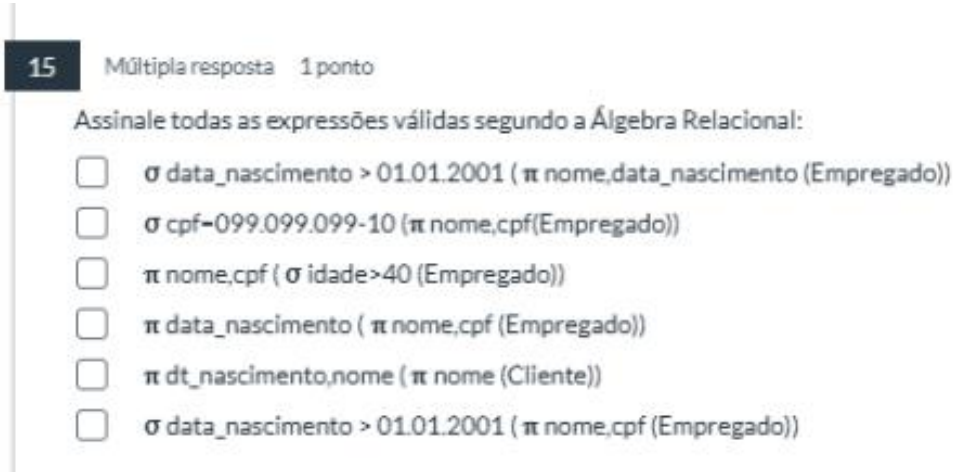
◯
```
SELECT first_name, last_name
FROM customer
JOIN store ON customer.store_id = store.store_id
JOIN address ON store.address_id = address.address_id
WHERE address = 'Rua Antonio José da Silva, 33';
```

◯
```
SELECT c.first_name, c.last_name
FROM customer c
JOIN rental r ON c.customer_id = r.customer_id
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN store s ON i.store_id = s.store_id
JOIN address a ON s.address_id = a.address_id
WHERE a.address = 'Rua Antonio José da Silva, 33';
```

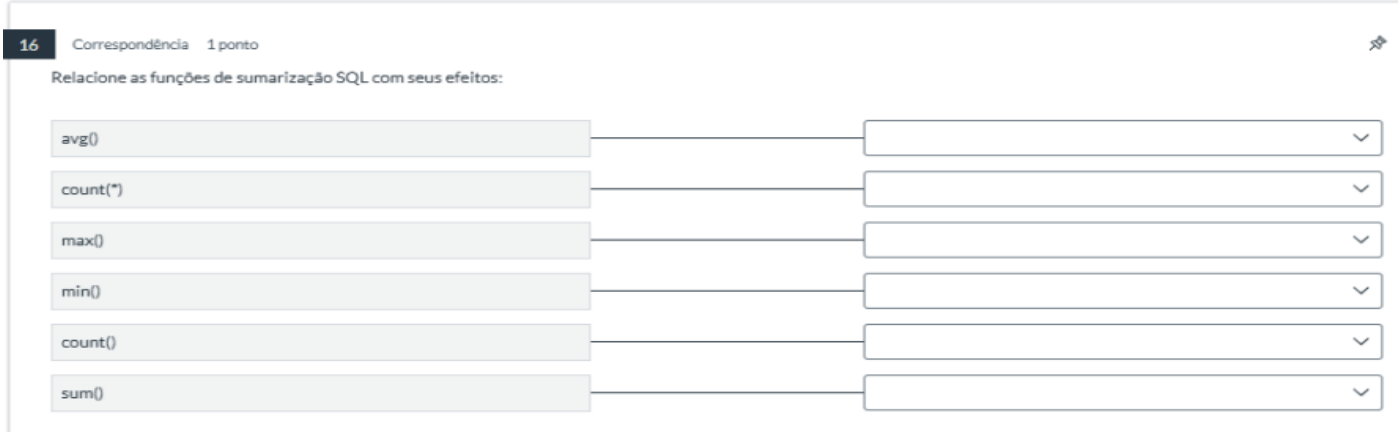
◯
```
SELECT DISTINCT c.first_name, c.last_name
FROM customer c
JOIN rental r c.customer_id = r.customer_id  -- Erro de sintaxe: Falta o ON
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN store s ON i.store_id = s.store_id
JOIN address a ON s.address_id = a.address_id
WHERE a.address = 'Rua Antonio José da Silva, 33';
```

◯
```
SELECT first_name, last_name
FROM customer
JOIN address ON customer.address_id = address.address_id
WHERE address = 'Rua Antonio José da Silva, 33';
```

ALTERNATIVA:A



ALTERNATIVA: A, B, C



AVG ---- MÉDIA

COUNT(*) ---- CONTAGEM

MAX ---- MAIOR VALOR

MIN ---- MENOR VALOR

COUNT() ---- CONTAGEM

SUM ---- SOMA

17 Múltipla escolha 1 ponto

Sobre comandos SQL para modificar a estrutura de uma tabela, considere as seguintes afirmações:

I. O comando UPDATE é usado para modificar os dados existentes em uma tabela.

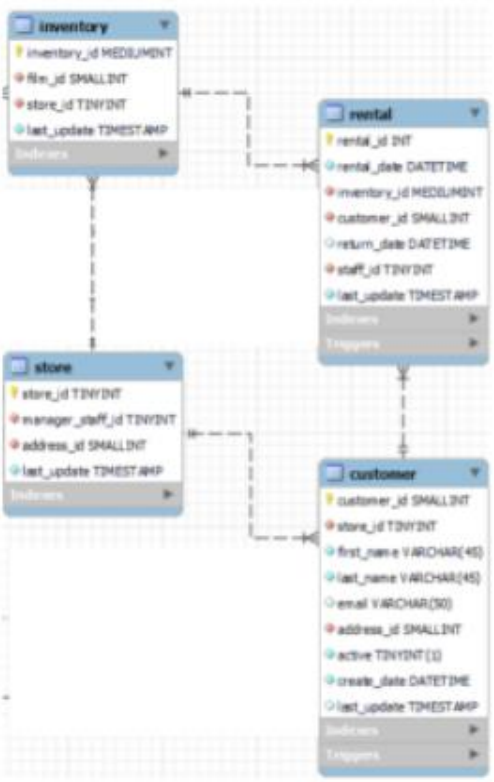
II. O comando ALTER é usado para modificar a estrutura de uma tabela.

III. O comando MODIFY é usado para alterar os valores dos dados registrados em uma coluna específica.

Está(ão) correta(s) apenas a(s) afirmação(ões):

- ◯ I e III
- ◯ I, II e III
- ◯ I e II
- ◯ II e III
- ◯ II

ALTERNATIVA: C



Qual o comando SQL para identificar quais os ids dos filmes (film_id) que foram alugados por um cliente com email joao@email.com.br na loja em que o gerente possui manager_staff_id 2?

- ○
```
SELECT r.inventory_id
FROM rental r
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN customer c ON r.customer_id = c.customer_id
JOIN store s ON i.store_id = s.store_id
WHERE c.email = 'joao@email.com.br'
 AND s.manager_staff_id = 2;
```
- ○
```
SELECT i.film_id
FROM inventory i
JOIN rental r ON i.inventory_id = r.inventory_id
JOIN customer c ON r.customer_id = c.customer_id
JOIN store s ON c.store_id = s.store_id
WHERE c.email = 'joao@email.com.br'
 AND s.manager_staff_id = 2;
```
- ○
```
SELECT r.inventory_id
FROM inventory i
JOIN rental r ON i.inventory_id = r.inventory_id
JOIN customer c ON r.customer_id = c.customer_id
JOIN store s ON r.store_id = s.store_id
WHERE c.email = 'joao@email.com.br'
 AND s.manager_staff_id = 2;
```
- ○
```
SELECT i.film_id
FROM inventory i
JOIN rental r ON i.inventory_id = r.inventory_id
JOIN customer c ON r.customer_id = c.customer_id
JOIN store s ON i.store_id = s.store_id
WHERE c.email = 'joao@email.com.br'
 AND s.manager_staff_id = 2;
```

ALTERNATIVA: D

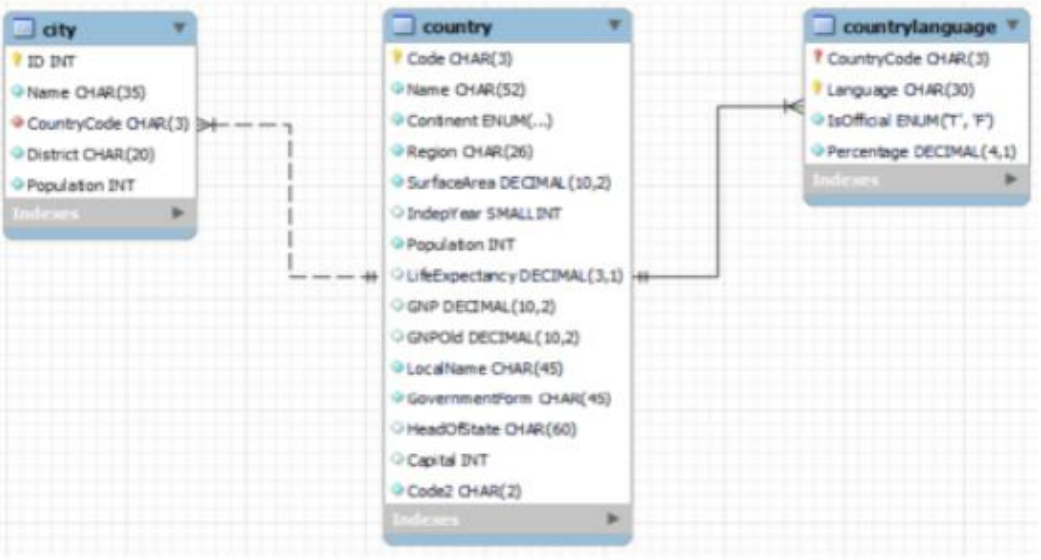
19 Ordenação 1 ponto

Considerando as relações a seguir:

Coloque os componentes da instrução em ordem de forma a que a instrução SQL possa ser adequadamente executada para obter o valor médio das locações cuja soma total seja acima de 100.00 pagas por usuário em ordem decrescente de valor médio:

desc | ORDER by valor | select c.first_name, c.last_name, avg(p.amount) as valor from | GROUP BY c.first_name, c.last_name | HAVING sum(p.amount) > 100.00 | customer c INNER JOIN rental r USING(customer_id) | INNER JOIN payment p USING(rental_id)

- `select c.first_name, c.last_name, avg(p.amount) as valor from`
- `customer c INNER JOIN rental r USING(customer_id)`
- `INNER JOIN payment p USING(rental_id)`
- `GROUP BY c.first_name, c.last_name`
- `HAVING sum(p.amount) > 100.00`
- `ORDER BY valor`
- `desc`



Qual das alternativas apresenta a instrução SQL correta para apresentar o nome da cidade com a maior populacao, a população da cidade, o nome do país, seu idioma e o valor do Produto Nacional Bruto do país que tem o maior Produto Nacional Bruto?

○
```
SELECT ci.Name, ci.Population, co.Name, cl.Language, co.GNP
FROM
city ci
INNER JOIN country co ON co.code = ci.CountryCode
INNER JOIN countryLanguage cl ON co.Code = cl.CountryCode
WHERE co.GNP = (select max(GNP) from country)
AND cl.IsOfficial = 'T'
ORDER BY ci.Population DESC
```

○
```
SELECT ci.Name, ci.Population, co.Name, cl.Language, co.GNP
FROM
city ci
INNER JOIN country co ON co.code = ci.CountryCode
INNER JOIN countryLanguage cl ON co.Code = cl.CountryCode
WHERE co.GNP = (select max(GNP) from country)
AND cl.IsOfficial = 'T'
ORDER BY ci.Population DESC
LIMIT 1
```

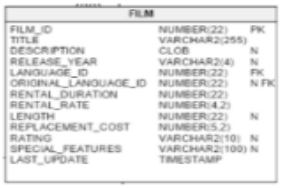
○
```
SELECT ci.Name, MAX(ci.Population), co.Name, cl.Language, co.GNP
FROM
city ci
INNER JOIN country co ON co.code = ci.CountryCode
INNER JOIN countryLanguage cl ON co.Code = cl.CountryCode
WHERE co.GNP = (SELECT MAX(GNP) FROM country)
AND cl.IsOfficial = 'T'
```

○
```
SELECT ci.Name, ci.Population, co.Name, cl.Language, max(co.GNP)
FROM
city ci
INNER JOIN country co ON co.code = ci.CountryCode
INNER JOIN countryLanguage cl ON co.Code = cl.CountryCode
WHERE cl.IsOfficial = 'T'
ORDER BY ci.Population DESC
LIMIT 1
```

ALTERNATIVA: B

21 Múltipla escolha 1 ponto

Considere a seguinte relação:



FILM		
FILM_ID	NUMBER(22)	PK
TITLE	VARCHAR2(255)	
DESCRIPTION	CLOB	N
RELEASE_YEAR	VARCHAR2(4)	N
LANGUAGE_ID	NUMBER(22)	FK
ORIGINAL_LANGUAGE_ID	NUMBER(22)	N FK
RENTAL_DURATION	NUMBER(22)	
RENTAL_RATE	NUMBER(4,2)	
LENGTH	NUMBER(22)	N
REPLACEMENT_COST	NUMBER(5,2)	
RATING	VARCHAR2(10)	N
SPECIAL_FEATURES	VARCHAR2(100)	N
LAST_UPDATE	TIMESTAMP	

Com base nessa relação, avalie as seguintes afirmações:

I. Para identificar os filmes mais recentes, podemos utilizar o seguinte SQL: SELECT FILM_ID, TITLE, DESCRIPTION, RELEASE_YEAR FROM FILM WHERE RELEASE_YEAR = (SELECT MAX(RELEASE_YEAR) FROM FILM)

II. Para relacionar os filmes mais longos, podemos utilizar o seguinte SQL: SELECT FILM_ID, TITLE, DESCRIPTION, MAX(LENGTH) FROM FILM

III. Para relacionar os filmes em ordem decrescente de custo de reposição é possível utilizar o seguinte SQL: SELECT FILM_ID, TITLE, DESCRIPTION, REPLACEMENT_COST FROM FILM ORDER BY REPLACEMENT_COST

Indique a alternativa que assinala qual(is) a(s) afirmação(ões) correta(s):

- ◯ I e III
- ◯ II e III
- ◯ III
- ◯ II
- ◯ I

ALTERNATIVA: E

Identifique qual o enunciado da álgebra relacional corresponde a qual instrução SQL.

π cpf,nome(<Cadastro>)	select cpf, nome from Cadastro
σ cpf ="099.099.099-01"∧nome="Ana"(Empregado)	select * from Empregado where cpf = "099.099.099-01" and
Cadastro(cpf, nome) x Endereco(logradouro,numero,cidade,estado)	select Cadastro cross join Endereco
π nome,salario(<Empregado>)	select nome,salario from Empregado
σ nome ="Carla"∧ano_nascimento>1997(Empregado)	select * from Empregado where nome = "Carla" and ano_nas
π nome,salario (σ idade>40 (Empregado))	select nome, salario from Empregado where idade > 40

23 Múltipla escolha 1 ponto

Considere as seguintes afirmações sobre comandos SQL:

I. O comando DELETE remove todos os registros de uma tabela, mas mantém a estrutura da tabela.

II. O comando DROP remove todos os registros e a estrutura da tabela do banco de dados.

III. O comando TRUNCATE remove todos os registros de uma tabela, mas mantém a estrutura da tabela.

Está(ão) correta(s) apenas a(s) afirmação(ões):

- ◯ II
- ◯ I , II e III
- ◯ I e III
- ◯ I
- ◯ III

ALTERNATIVA: B

24 Múltipla escolha 1 ponto

Considere a seguinte relação:

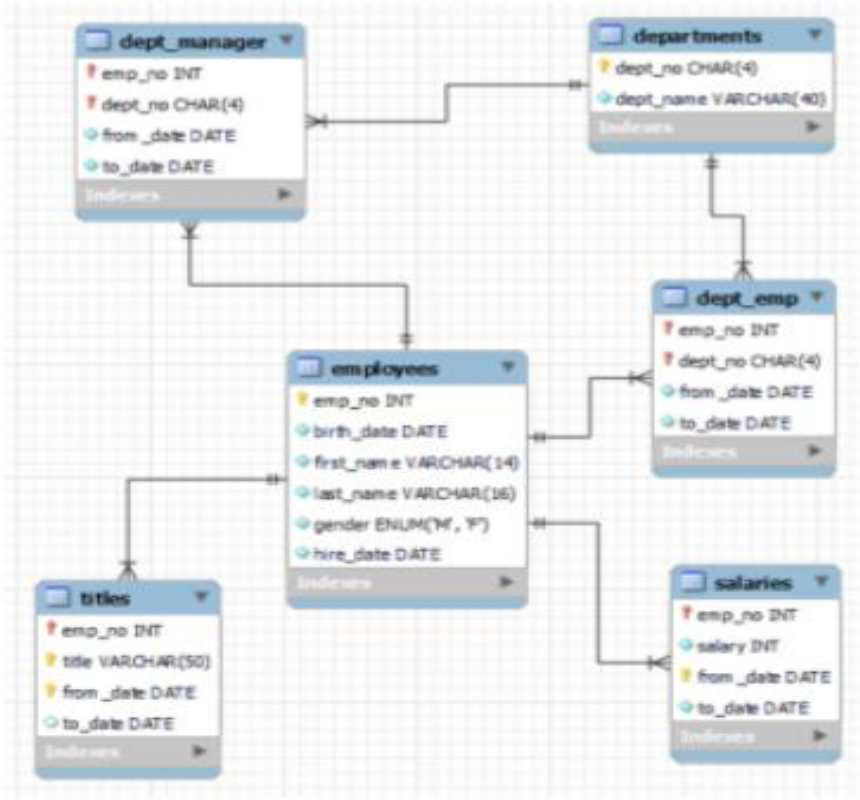
IdCliente	NomeCompleto	CPF	DataCadastro	Telefones
11	Ana Lima	033.033.330-12	12/01/2023	11-90909-0909, 11-98980-9898

Considerando essa relação, é possível afirmar:

- ◯ Para que seja normalizada, sem perda de informação, deve ser criada uma tabela telefones, com chave estrangeira idCliente e com a inclusão de um registro para cada telefone.
- ◯ Para que seja normalizada, sem perda de informação, basta excluir a tabela telefone.
- ◯ Ela está de acordo com a 3FN (atende a 3ª forma normal)
- ◯ Ela está de acordo com a 1FN (atende a 1ª forma normal)
- ◯ Ela está de acordo com a 2FN (atende a 2ª forma normal)

ALTERNATIVA: A

No seguinte modelo de dados, os salários são registrados anualmente, desde a contratação ou promoção:



Considerando esse modelo, qual a instrução SQL adequada para:

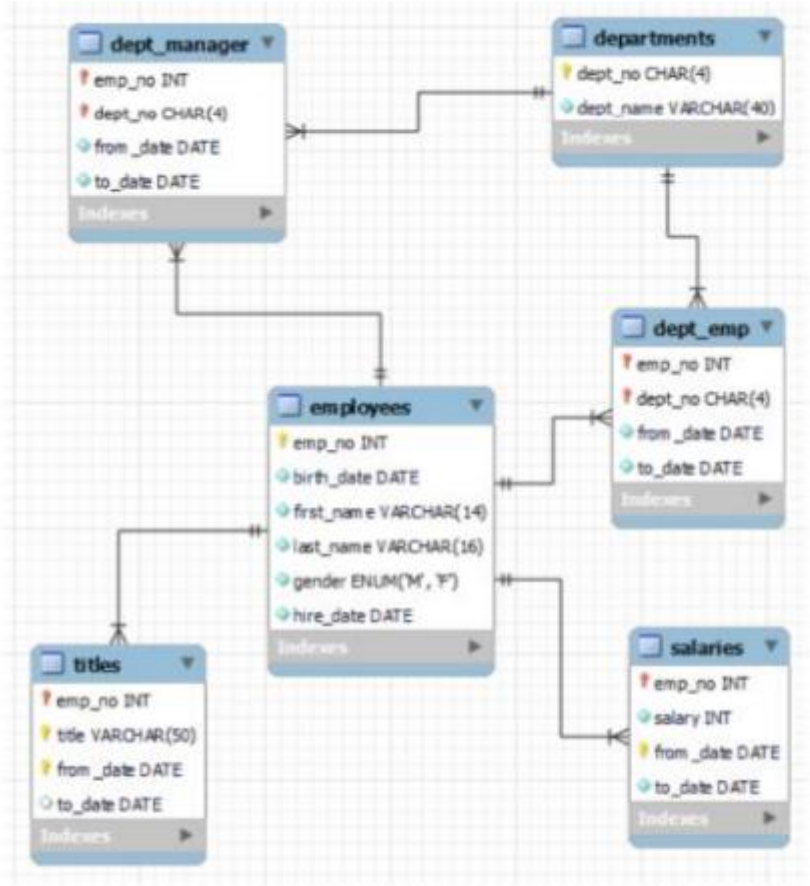
Identificar os salários pagos por departamento no ano de 2023.

- ◯
```
select d.dept_name, s.salary
from salaries s
INNER JOIN employees e ON s.emp_no = e.emp_no
INNER JOIN dept_emp de ON de.emp_no = e.emp_no
INNER JOIN departments d ON d.dept_no = de.dept_no
WHERE s.from_date > "2023-01-01" and s.to_date < "2023-12-31"
having d.dept_name;
```
- ◯
```
select d.dept_name, s.salary
from salaries s
INNER JOIN employees e ON s.emp_no = e.emp_no
INNER JOIN dept_emp de ON de.emp_no = e.emp_no
INNER JOIN departments d ON d.dept_no = de.dept_no
WHERE s.from_date > "2023-01-01" and s.to_date < "2023-12-31"
group by d.dept_name;
```
- ◯
```
select d.dept_name, sum(s.salary)
from salaries s
INNER JOIN employees e ON s.emp_no = e.emp_no
INNER JOIN dept_emp de ON de.emp_no = e.emp_no
INNER JOIN departments d ON d.dept_no = de.dept_no
WHERE s.from_date > "2023-01-01" and s.to_date < "2023-12-31"
group by d.dept_name;
```
- ◯
```
select d.dept_name, sum(s.salary)
from salaries s
INNER JOIN employees e ON s.emp_no = e.emp_no
INNER JOIN dept_emp de ON de.emp_no = e.emp_no
INNER JOIN departments d ON d.dept_no = de.dept_no
WHERE s.from_date > "2023-01-01" and s.to_date < "2023-12-31";
```

ALTERNATIVA: C

26 Múltipla escolha 1 ponto

Observe o modelo:



Avalie de acordo com o modelo qual(is) das seguintes instruções pode(m) ser utilizada(s) para identificar a(s) contratação(ões) mais recente(s):

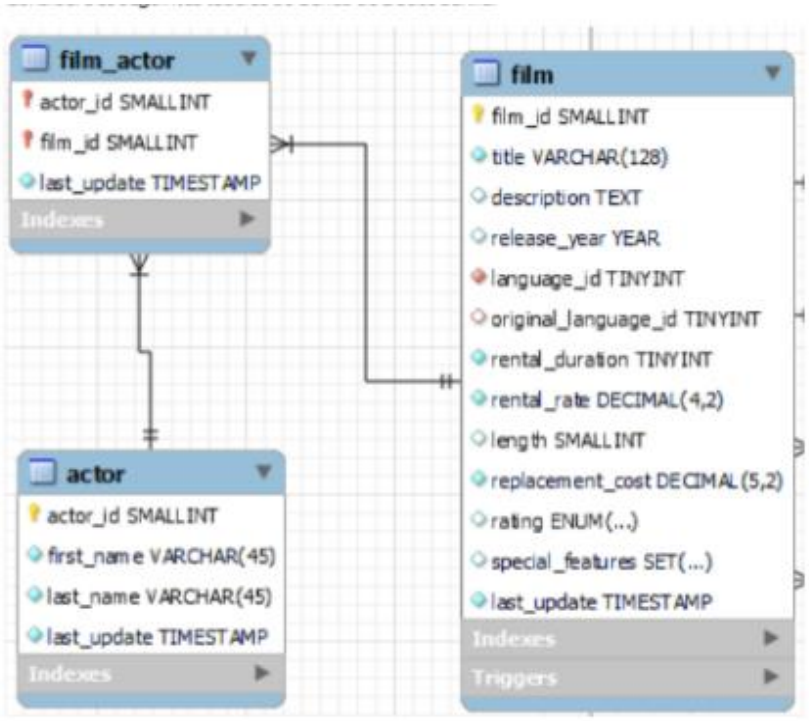
I. SELECT first_name, last_name, hire_data FROM employees WHERE hire_date = (SELECT min(hire_date) FROM employees)

II. SELECT first_name, last_name FROM employees ORDER BY hire_date desc LIMIT 1

III. SELECT first_name, last_name FROM employees WHERE hire_date = (SELECT hire_date from employees order by hire_date desc LIMIT 1)

- ◯ II e III
- ◯ I e III
- ◯ II
- ◯ I
- ◯ III

ALTERNATIVA: A



Quais das seguintes instruções SQL apresenta instruções válidas para relacionar os filmes e os atores que fazem parte do elenco:

I. SELECT title, first_name, last_name FROM film JOIN film_actor ON film.film_id = film_actor.film_id JOIN actor ON film_actor.actor_id = actor.actor_id;

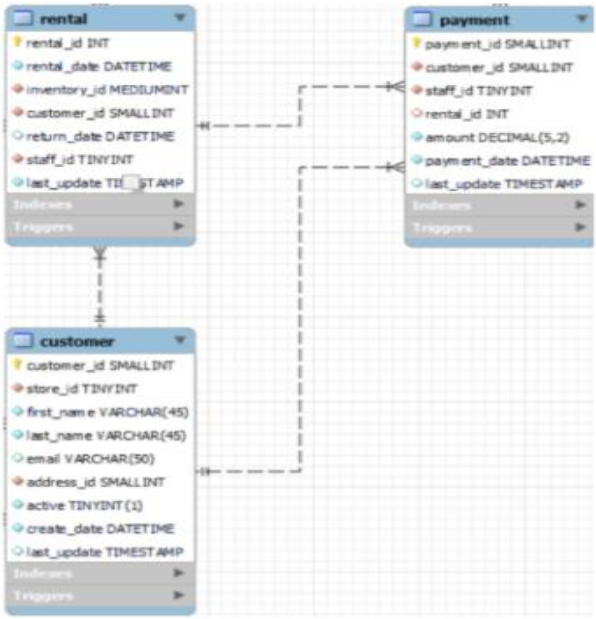
II. SELECT title, first_name, last_name FROM film_actor JOIN film ON film_actor.film_id = film.film_id JOIN actor ON film_actor.actor_id = actor.actor_id;

III. SELECT title, first_name, last_name FROM actor JOIN film_actor ON actor.actor_id = film_actor.actor_id JOIN film ON film_actor.film_id = film.film_id;

IV. SELECT title, first_name, last_name FROM film JOIN actor ON actor.actor_id = film.actor_id;

- ◯ I e IV
- ◯ III
- ◯ I, II e III
- ◯ II e III
- ◯ II, III e IV

ALTERNATIVA: C



Coloque os componentes da instrução em ordem de forma a que a instrução SQL possa ser adequadamente executada para obter o total de locações acima de 100.00 pagas por usuário em ordem decrescente do valor total de locações:

select c.first_name, c.last_name, sum(p.amount) as valor from

HAVING sum(p.amount) > 100.00

GROUP BY c.first_name, c.last_name

INNER JOIN payment p USING(rental_id)

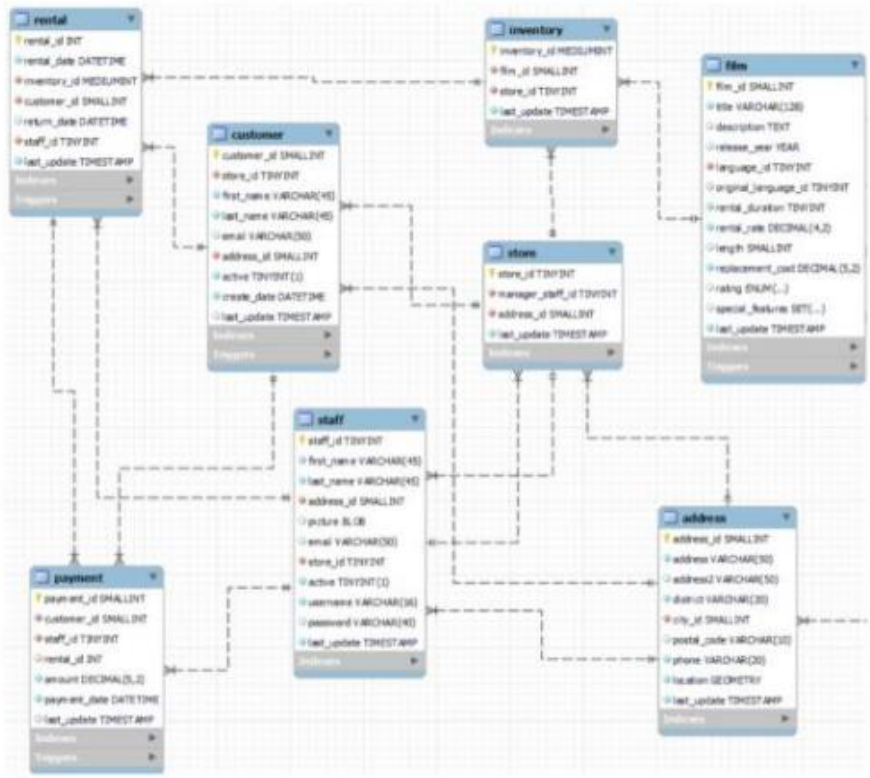
desc

customer c INNER JOIN rental r USING(customer_id)

ORDER by valor

- `select c.first_name, c.last_name, sum(p.amount) as valor from`
- `customer c INNER JOIN rental r USING(customer_id)`
- `INNER JOIN payment p USING(rental_id)`
- `GROUP BY c.first_name, c.last_name`
- `HAVING sum(p.amount) > 100.00`
- `ORDER BY valor`
- `desc`

Considere o seguinte conjunto de tabelas:



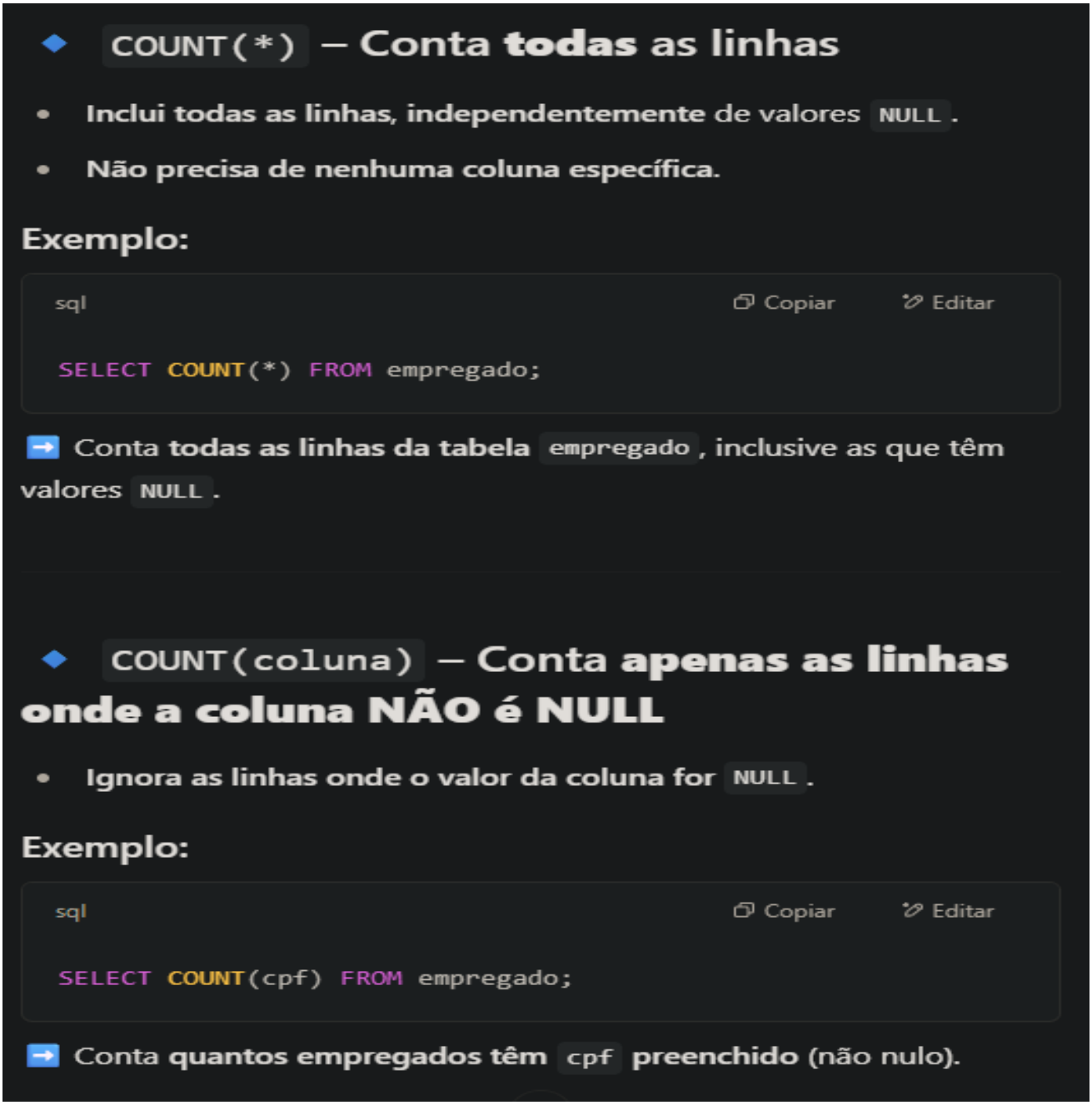
Considerando esse modelo, é correto afirmar:

- ◯ Uma locação possui, obrigatoriamente, um ou mais pagamentos e pode possuir ou não um cliente, um filme em estoque e um funcionário.
- ◯ Uma locação possui, obrigatoriamente, um filme em estoque, um cliente e um funcionário e um ou mais pagamentos.
- ◯ Uma locação possui, obrigatoriamente, um filme em estoque (inventory), um cliente (customer) e um funcionário (staff) e pode possuir um ou mais pagamentos (payment).
- ◯ Uma locação pode ser registrada apenas com a data de logcação e a data de retorno do filme.
- ◯ Uma locação possui, obrigatoriamente, um pagamento.

ALTERNATIVA: C

Símbolos da Álgebra Relacional:

Símbolo	Nome	Função
σ	Seleção (select)	Filtra linhas (registros) com base em uma condição.
π	Projeção (project)	Seleciona colunas (atributos) específicos da relação.
ρ	Renomeação (rename)	Renomeia o nome da tabela ou de colunas.
∪	União (union)	Une duas relações (sem duplicatas).
∩	Interseção (intersection)	Retorna os registros comuns entre duas relações.
−	Diferença (difference)	Retorna os registros da primeira relação que não estão na segunda.
×	Produto cartesiano	Junta todas as combinações de tuplas entre duas relações.
⋈	Junção (join)	Junta tabelas com base em condições.



1. Introdução e Linguagem de Definição de Dados (DDL)

A DDL é usada para criar e modificar a estrutura dos objetos do banco de dados.

- **`CREATE DATABASE nome_do_banco;`**
  - Utilizado para criar um novo banco de dados.
- **`CREATE TABLE nome_tabela ( coluna1 tipo_dados, coluna2 tipo_dados, ... );`**

    - Utilizado para criar uma nova tabela dentro de um banco de dados, especificando suas colunas e os tipos de dados que cada coluna armazenará.
- **`ALTER TABLE nome_tabela ACAO;`**
    - Utilizado para modificar uma tabela existente. Ações comuns incluem:
        - `ADD nome_coluna tipo_dados;` (Adicionar uma nova coluna)
        - `DROP COLUMN nome_coluna;` (Excluir uma coluna)
        - `MODIFY COLUMN nome_coluna novo_tipo_dados;` (Alterar o tipo de dados de uma coluna)

2. Normalização de Banco de Dados

A normalização é o processo de organização dos dados em um banco de dados para reduzir a redundância e melhorar a integridade dos dados. Segue algumas regras chamadas "Formas Normais".

- **O que é Normalização?**
    - É o processo de organizar as colunas e tabelas de um banco de dados relacional para minimizar a redundância de dados e melhorar a integridade dos dados.
- **Primeira Forma Normal (1FN):**
    - Elimina grupos repetidos: cada célula da tabela deve conter um único valor e cada registro deve ser único.
    - Cria-se uma tabela separada para cada conjunto de dados relacionados.
    - Identifica-se cada conjunto de dados com uma chave primária.
- **Segunda Forma Normal (2FN):**
    - Requer que a tabela esteja na 1FN.
    - Remove dados redundantes: todos os atributos não chave devem depender funcionalmente da chave primária completa. Se uma tabela tem uma chave primária composta, os atributos não chave devem depender de toda a chave, e não de parte dela.
    - Criam-se tabelas separadas para conjuntos de valores que se aplicam a múltiplos registros, relacionando-as com uma chave estrangeira.

- **Terceira Forma Normal (3FN):**
  - Requer que a tabela esteja na 2FN.
  - Elimina dependências transitivas: os atributos não chave não devem depender de outros atributos não chave. Ou seja, elimina campos que não dependem diretamente da chave primária.
- **Exceção:** A adesão estrita à terceira forma normal pode, em alguns casos, introduzir complexidade que não compensa o benefício prático, especialmente em bancos de dados NoSQL, que podem aplicar a 3FN com certa flexibilidade ("folguinha") priorizando outros aspectos como escalabilidade e desempenho para dados que mudam com frequência.

3. Linguagem de Manipulação de Dados (DML)

A DML é usada para gerenciar os dados dentro dos objetos de esquema.

- **`INSERT INTO nome_tabela (coluna1, coluna2, ...) VALUES (valor1, valor2, ...);`**
  - Utilizado para inserir novos registros (linhas) em uma tabela.
- **`UPDATE nome_tabela SET coluna1 = valor1, coluna2 = valor2, ... WHERE condicao;`**
  - Utilizado para modificar registros existentes em uma tabela. A cláusula `WHERE` é crucial para especificar quais registros serão atualizados.
- **`DELETE FROM nome_tabela WHERE condicao;`**
  - Utilizado para excluir registros de uma tabela. A cláusula `WHERE` especifica quais registros serão excluídos. Se omitida, todos os registros da tabela serão excluídos.

4. Linguagem de Consulta de Dados (DQL) - O Comando SELECT

A DQL é usada para consultar os dados contidos nos objetos do esquema.

Estrutura Básica:

- **`SELECT coluna1, coluna2, ... FROM nome_tabela WHERE condicao ORDER BY coluna [ASC|DESC] LIMIT n;`**
  - `SELECT`: Especifica as colunas a serem retornadas.

        - `FROM`: Especifica a tabela da qual recuperar os dados.
        - `WHERE`: Filtra os registros com base em uma condição.
        - `ORDER BY`: Ordena os resultados.
        - `LIMIT`: Restringe o número de linhas retornadas.
- **`SELECT DISTINCT coluna;`**
    - Retorna apenas valores distintos (únicos) para a coluna especificada.
- **Operadores de Filtragem:**
    - **`LIKE` / `NOT LIKE`**: Usados na cláusula `WHERE` para filtrar dados de texto com base em padrões.
        - `WHERE coluna LIKE 'A%';` (Começa com 'A')
        - `WHERE coluna LIKE '%x';` (Termina com 'x')
        - `WHERE coluna LIKE '%termo%';` (Contém 'termo' em qualquer posição)
        - `WHERE coluna LIKE '_a_';` (O caractere _ substitui um único caractere. Ex: 'casa', 'bala')
        - `NOT LIKE` é usado para excluir padrões.

5. Junções (JOINs)

As junções são usadas para combinar linhas de duas ou mais tabelas com base em uma coluna relacionada entre elas.

- **`INNER JOIN` (ou `JOIN`)**
    - Retorna registros que têm valores correspondentes em ambas as tabelas. Se não houver correspondência, o registro não é retornado.
    - **Sintaxe:**

SQL
```
SELECT tabelaA.coluna1, tabelaB.coluna2
FROM tabelaA
INNER JOIN tabelaB ON tabelaA.chave_comum =
tabelaB.chave_comum;
```

    - (Visualização: Interseção de dois conjuntos)
- **`LEFT JOIN` (ou `LEFT OUTER JOIN`)**
    - Retorna todos os registros da tabela da esquerda (`tabelaA`) e os registros correspondentes da tabela da direita (`tabelaB`).

Se não houver correspondência na tabela da direita, retorna `NULL` para as colunas da tabela da direita.

- **Sintaxe:**

```
SQL
SELECT tabelaA.coluna1, tabelaB.coluna2
FROM tabelaA
LEFT JOIN tabelaB ON tabelaA.chave_comum =
tabelaB.chave_comum;
```

- (Visualização: Todo o conjunto da esquerda mais a interseção)

- **`RIGHT JOIN` (ou `RIGHT OUTER JOIN`)**
  - Retorna todos os registros da tabela da direita (`tabelaB`) e os registros correspondentes da tabela da esquerda (`tabelaA`). Se não houver correspondência na tabela da esquerda, retorna `NULL` para as colunas da tabela da esquerda.
  - **Sintaxe:**

```
SQL
SELECT tabelaA.coluna1, tabelaB.coluna2
FROM tabelaA
RIGHT JOIN tabelaB ON tabelaA.chave_comum =
tabelaB.chave_comum;
```

  - (Visualização: Todo o conjunto da direita mais a interseção)
- **`FULL JOIN` (ou `FULL OUTER JOIN`)**
  - Retorna todos os registros quando há uma correspondência em uma das tabelas (esquerda ou direita). Ou seja, retorna todos os registros de ambas as tabelas e preenche com `NULL` onde não há correspondência.
  - **Sintaxe:**

```
SQL
SELECT tabelaA.coluna1, tabelaB.coluna2
FROM tabelaA
FULL OUTER JOIN tabelaB ON tabelaA.chave_comum =
tabelaB.chave_comum;
```

  - (Visualização: União de ambos os conjuntos)

6. Agrupamento de Dados e Funções de Agregação

- **`GROUP BY`**
    - Agrupa linhas que têm os mesmos valores em colunas especificadas em linhas de resumo. É frequentemente usado com funções de agregação para calcular métricas para cada grupo.
    - **Exemplo:** Contar quantos produtos existem em cada categoria.

SQL
```
SELECT categoria_id, COUNT(produto_id) AS
total_produtos
FROM produtos
GROUP BY categoria_id;
```

- **Funções de Agregação:** Realizam um cálculo em um conjunto de valores e retornam um único valor.
    - **`COUNT(coluna_ou_*)`**: Retorna o número de linhas.
        - `COUNT(*)`: Conta todas as linhas.
        - `COUNT(coluna)`: Conta as linhas onde a coluna não é `NULL`.
        - Sintaxe: `SELECT COUNT(coluna) FROM tabela WHERE condicao;`
    - **`SUM(coluna)`**: Retorna a soma dos valores de uma coluna numérica.
        - Sintaxe: `SELECT SUM(coluna) FROM tabela WHERE condicao;`
    - **`AVG(coluna)`**: Retorna a média dos valores de uma coluna numérica.
        - Sintaxe: `SELECT AVG(coluna) FROM tabela WHERE condicao;`
    - **`MIN(coluna)`**: Retorna o menor valor de uma coluna.
        - Sintaxe: `SELECT MIN(coluna) FROM tabela WHERE condicao;`
    - **`MAX(coluna)`**: Retorna o maior valor de uma coluna.

        - Sintaxe: `SELECT MAX(coluna) FROM tabela WHERE condicao;`
    - **`ISNULL(coluna, valor_se_nulo)`**: (Função auxiliar, não de agregação pura, mas mencionada) Verifica se uma expressão é NULL e permite substituí-la por outro valor. A sintaxe exata pode variar entre SGBDs (ex: `COALESCE`, `IFNULL`).
- **`HAVING`**
    - Usada para filtrar grupos criados pela cláusula `GROUP BY`. Enquanto `WHERE` filtra linhas *antes* do agrupamento, `HAVING` filtra grupos *após* o agrupamento e após as funções de agregação serem calculadas.
    - **Exemplo:** Selecionar tipos de produto onde a quantidade em estoque agrupada é maior que 200.

SQL

```
SELECT tipo, SUM(quantidade) AS quantidade_total
FROM produtos
GROUP BY tipo
HAVING SUM(quantidade) > 200;
```