Lista 2 (exercícios) - Amanda Vicente e Daniella Maleski

-------------------------------------------------------------------------------------------------------------------------------

Questao 1:

A locadora deseja fazer um levantamento para saber quais títulos de filmes dublados são pouco locados. Entende-se por pouco locados um total inferior a 3 locações.   
Repare que, em geral, os filmes possuem cópias dubladas e legendadas. Assim sendo, é preciso contabilizar as locações de cópias dubladas por filme;

SQL:

SELECT f.titulo, COUNT(\*) AS qtda\_locacoes FROM locacoes l, filmes f

WHERE l.filme = f.id AND EXISTS (

SELECT \* FROM copias c

WHERE c.tipo = 'dublado' AND l.id = c.id AND l.filme = c.filme)

GROUP BY f.titulo

SELECT cp.id, cp.filme, titulo FROM filmes f JOIN Copias cp ON f.id=cp.filme

WHERE tipo='dublado' AND (

SELECT COUNT(\*) FROM locacoes l WHERE l.filme=cp.filme AND l.id=cp.id) < 3;

-------------------------------------------------------------------------------------------------------------------------------

Questao 2:

Buscar o título e estilo dos filmes mais antigos da locadora. Entende-se por mais antigos àqueles filmes que possuem o menor ano de lançamento;

SQL:

SELECT titulo, e.nome FROM filmes JOIN estilos e ON e.id = estilo

WHERE ano <= ALL (SELECT ano FROM filmes);

SELECT titulo, nome FROM filmes f JOIN estilos e ON f.estilo=e.id

WHERE ano <= ALL(SELECT ano FROM Filmes);

-------------------------------------------------------------------------------------------------------------------------------

Questao 3:

buscar nome e endereço de clientes que tenham locado todos os filmes cadastrados na locadora;

SQL:

SELECT nome, endereco FROM clientes c WHERE NOT EXISTS (

SELECT \* FROM filmes f WHERE NOT EXISTS (

SELECT \* FROM locacoes l WHERE f.id = l.filme AND c.id = l.cliente));

SELECT nome, endereco FROM clientes c WHERE NOT EXISTS(

SELECT \* FROM filmes f WHERE NOT EXISTS (

SELECT \* FROM locacoes l WHERE l.cliente=c.id AND l.filme=f.id));

-------------------------------------------------------------------------------------------------------------------------------

Questao 4:

Buscar nome e endereço de clientes que tenham locado pelo menos 1 filme de cada um dos estilos de filme cadastrados na locadora;

SQL:

SELECT nome, endereco FROM clientes c WHERE c.id = ANY (

SELECT id FROM Locacoes l WHERE l.filme = ANY (

SELECT id FROM Filmes f WHERE f.estilo != ANY(SELECT id FROM estilos)));

SELECT nome, endereco FROM clientes c WHERE NOT EXISTS (

SELECT \* FROM estilos e WHERE NOT EXISTS (

SELECT \* FROM locacoes l WHERE l.cliente=c.id ANDl.filme IN (

SELECT id FROM filmes WHERE estilo=e.id)));

-------------------------------------------------------------------------------------------------------------------------------

Questao 5:

buscar o nome e telefone dos funcionários que registraram locações de mais cópias do que as registradas por funcionários que têm o maior salário cadastrado. Pode haver mais de um funcionário com o maior salário, neste caso considerar o que tem o maior número de locações;

SQL:

-- SELECT nome, telefone FROM funcionarios

-- SELECT id FROM funcionarios WHERE salario >= ALL (

-- SELECT salario FROM funcionarios)

SELECT nome, fone FROM funcionarios f WHERE (

SELECT COUNT(\*) FROM locacoes WHERE funcionario=f.id) > ALL(

SELECT COUNT(\*) FROM locacoes l WHERE funcionario IN(

SELECT id FROM funcionarios WHERE salario=(

SELECT MAX(salario) FROM funcionarios))

GROUP BY funcionario);

-------------------------------------------------------------------------------------------------------------------------------

Questao 6:

buscar os ids, nomes e fones dos clientes que locaram filmes de ação e de suspense. Resolver está questão sem utilizar junções ou produto cartesiano;

SQL:

SELECT id, nome, fone FROM clientes WHERE id IN (

SELECT DISTINCT cliente FROM locacoes WHERE filme IN (

SELECT id FROM filmes WHERE estilo = ANY (

SELECT id FROM estilos WHERE nome = 'ação' OR nome = 'suspense')));

SELECT id, nome, fone FROM clientes WHERE id IN(

SELECT cliente FROM locacoes WHERE filme IN(

SELECT id FROM filmes WHERE estilo IN(

SELECT id FROM estilos WHERE nome IN('ação','suspense'))));

-------------------------------------------------------------------------------------------------------------------------------

Questao 7:

buscar os ids, nomes e fones dos clientes que nunca locaram filmes de romance e de ação;

SQL:

SELECT id, nome, fone FROM clientes WHERE id NOT IN (

SELECT DISTINCT cliente FROM locacoes WHERE filme = ANY (

SELECT id FROM filmes WHERE estilo = ANY (

SELECT id FROM estilos WHERE nome = 'ação' OR nome = 'romance')));

SELECT id, nome, fone FROM clientes WHERE id NOT IN(

SELECT cliente FROM locacoes WHERE filme IN(

SELECT id FROM filmes WHERE estilo IN(

SELECT id FROM estilos WHERE nome IN('romance','ação'))));

-------------------------------------------------------------------------------------------------------------------------------

Questao 8:

Formule uma expressão SQL que modifique o salário de todos aqueles funcionários retornados pela expressão SQL da questão 5. O novo salário deve corresponder a um aumento de 10% sobre o maior valor de salário cadastrado . Reaproveite a expressão da questão 5 para identificar os funcionários que devem receber este aumento.

SQL:

UPDATE funcionarios SET salario=1.1\*(Select max(salario) From funcionarios) WHERE id IN (SELECT id FROM funcionarios f WHERE (Select COUNT(\*) From locacoes Where funcionario=f.id) >   
ALL(Select COUNT(\*) From locacoes l Where funcionario IN(Select id From funcionarios Where salario=(Select max(salario) From funcionarios)) GROUP BY funcionario));

-------------------------------------------------------------------------------------------------------------------------------

Questao 9:

O funcionário com id=1 tornou-se um cliente. Faça uma expressão SQL que cadastre este funcionário como um cliente (ele não terá um responsável);

SQL:

INSERT INTO clientes (id, nome, endereco, fone, cidade, cpf, responsavel)

SELECT \* FROM SELECT MAX(id)+1 FROM clientes,

SELECT nome, endereco, fone, cidade, cpf FROM funcionarios WHERE id = 2,

UPDATE SET responsavel = NULL;

INSERT INTO clientes (id,nome,endereco,fone,cidade,cpf)

(SELECT \* FROM (SELECT MAX(id)+1 FROMclientes) AS a,

(SELECT nome, endereco, fone, cidade, cpf FROM funcionarios WHERE id=1) AS b);

-------------------------------------------------------------------------------------------------------------------------------

Questao 10:

O cliente que fez a última locação (datar mais recente) deseja agora locar uma cópia disponível do filme com título ‘X-Men 3’ para as mesmas datar e datapd de sua última locação. Faça uma expressão SQL que identifique uma cópia disponível para o dia de hoje e cadastre esta locação. O funcionário que irá registrar esta locação é aquele que possui o nome 'Rui Hack'.

SQL:

--INSERT INTO locacoes(id, filme, cliente, datar, datapd, funcionario, datad)

-- SELECT \* FROM (id da copia do filme disponivel),

-- (SELECT id FROM filmes WHERE titulo = 'X-Men 3'),

-- (SELECT cliente, datar, datapd FROM locacoes WHERE datar >= ALL (SELECT datar FROM locacoes)),

-- (SELECT id FROM funcionarios WHERE nome = 'Rui Hack'),

SELECT \* FROM copias WHERE filme = (SELECT id FROM filmes WHERE titulo = 'X-Men 3');

SELECT \* FROM locacoes;

INSERT INTO locacoes (id,filme,cliente,datar,datapd,funcionario)

(SELECT \* FROM (SELECT id,filme FROM copias WHERE filme IN(SELECT id FROM filmes WHERE titulo='X-Men 3') AND(id,filme) NOT IN(SELECT id,filme FROM Locacoes WHERE datapd>=current\_date AND datad IS NULL) LIMIT 1) AS a,

(SELECT cliente, datar, datapd FROM locacoes ORDER BY datar DESC LIMIT 1) AS b,

(Select id From funcionarios Where nome='Rui Hack') as c);

-------------------------------------------------------------------------------------------------------------------------------