***<Usando junção>***

1. Recupere todos os números das notas fiscais, juntamente com a data de emissão onde o CNPJ é igual a 87.027.742/0001-99.

**select nf.nnota, nf.dataemissao**

**from pessoajuridica pf join notafiscal nf on(pf.idpessoa=nf.idpessoajuridica)**

**where pf.cnpj= '87.027.742/0001-99'**

***<Usando junção>***

1. Recupere todos os Produtos vendidos, a descrição e o valor dos produtos que foram vendidos mais de 3 unidades.

**select p.descricao, p.valor**

**from produto p join vendaProduto vp on(p.id=vp.idProduto)**

**where vp.quantidade >= 3**

***<Usando Ordenação>***

1. Recupere o nome e o valor da compra de todas as pessoas que efetuaram um pagamento via cartão de crédito, cujo o valor da compra foi dividido em 4 parcelas. O resultado final deve mostrar o valor de cada compra ordenado de forma descendente.

**select p.nome as NomePessoa, pg.valor as ValorCompra**

**from Pessoa p, Pagamento pg , PagamentoCartao pc**

**where p.id = pg.idPessoa and pg.id=pc.idPagamento and pc.numeroParcelas=4**

**order by pg.valor desc**

**<Aninhada, Ordenação e Intersect>**

**<Necessário adicionar uma nova tupla para satisfazer a condição>**

1. Recupere o ID, o Nome das pessoas que efetuaram um pagamento com cartão de crédito e também um pagamento a vista, a exibição deve ordenar o ID da pessoa de forma ascendente.

**select nome, id**

**from Pessoa**

**where id in(**

**(select pg.idPessoa**

**from Pagamento pg, PagamentoCartao pc**

**where pg.id=pc.idPagamento)**

**Intersect**

**(select pg.idPessoa**

**from pagamento pg, PagamentoAvista pc**

**where pg.id=pc.idPagamento)**

**)**

**order by id**

**<SubString>**

1. Recupere o Nome e o Telefone de todas as pessoas cuja a primeira letra do nome começa com ‘M’.

**select p.nome as NomePessoa, pf.telefone as TelefonePessoa**

**from Pessoa p join PessoaTelefone pf on(p.id=pf.idPessoa )**

**where p.nome like 'M%'**

**-- <Group by, Order by e Having>**

1. Recupere o ID, Nome e a quantidade de telefones de todas as pessoas que tem mais de 1 telefone cadastrado. Deve-se exibir o resultado ordenado pelo ID da pessoa de forma ascendente.

**select p.id, p.nome, count(p.id) as QuantidadeTelefones**

**from Pessoa p join PessoaTelefone pf on(p.id=pf.idPessoa)**

**group by p.id, p.nome**

**having count(\*)>1**

**order by p.id,p.nome**

**--<Order by e Group by>**

1. Crie uma visão que mostre o nome do funcionário e valor do caixa após o fechamento do mesmo. O resultado deve exibir o valor do caixa após fechado em ordem descendente.

**create view FechamentoCaixa as**

**select f.nome, c.valorFechamento**

**from funcionario f join caixa c on (f.nctps=c.nctps)**

**group by c.valorFechamento, f.nome**

**order by c.valorFechamento desc**

**<Comparação com valor Nulo>**

1. Recupere o id, a data e a descrição do produto onde o número da nota seja NULO e o valor da venda seja maior que 250 reais.

**select v.id, p.descricao, v.data**

**from venda v, vendaProduto vp, produto p**

**where v.id= vp.idVenda and vp.idProduto = p.id and v.nnota is null and v.valor >250**

**<Comparação com valor Nulo>**

1. Recupere o número da nota fiscal, o nome da pessoa Jurídica e a razão social do mesmo onde na venda o número da nota fiscal seja diferente de NULO e o valor da venda seja menor que 999 reais. Deve-se exibir o nome da pessoa jurídica em ordem descendente.

**select p.nome as NomePessoaJuridica, pj.razaosocial, nf.nnota as NúmeroNotaFiscal**

**from Pessoa p , PessoaJuridica pj, NotaFiscal nf, Venda v**

**where p.id=pj.idPessoa and v.nnota=nf.nnota and nf.idPessoajuridica=pj.idPessoa and v.nnota is not null and v.valor < 999**

**order by p.nome desc**

**<SubString>**

1. Recupere a descrição do produto onde a venda seja do tipo ‘consumidor’ e a descrição do produto tenha a letra ‘c’ em sua descrição.

**select p.descricao**

**from Produto p, vendaProduto vp , venda v**

**where p.id=vp.idProduto and vp.idVenda=v.id and v.tipoVenda = 'Consumidor' and p.descricao like 'A%'**

**<Função Agregada>**

1. Recupere o menor salário, o maior salário e a média salarial e valor total do salário de todos os empregados onde a data de admissão e igual a ‘2018-09-01’.

**select**

**Max(salario) as MaiorSalario,**

**Min(Salario) as MenorSalario,**

**AVG(Salario) as MediaSalarial,**

**SUM(Salario) as Total**

**from Funcionario**

**where dataAdmissao = '2018-09-01'**

**<Função agregada>**

1. Recupere o nome e o salário dos funcionários que ganham acima de 1000 reais. O resultado deve ser ordenado pelo salário de forma descendente.

**select nome, salario**

**from Funcionario**

**group by nome, salario**

**having (MIN(salario)>1000)**

**order by salario desc**

**<Operação com Conjuntos>**

1. Recupere todos os ID e valor dos pagamentos feitos a prazo aonde o valor do mesmo seja maior que 280.

**(SELECT idpagamento, p.valor FROM Pagamentocheque pc, Pagamento p where p.id = pc.idpagamento and valor > 280)**

**union**

**(SELECT IDPagamento, p.valor FROM PagamentoCartao pca, Pagamento p where p.id = pca.idpagamento and valor > 280)**

**<Aninhada Correlacionada>**

1. Recupere todos os nomes dos funcionários que não possuem telefone.

**SELECT Nome FROM funcionario f**

**WHERE NOT EXISTS**

**(SELECT \* FROM funcionarioTelefone FT**

**WHERE F.NCTPS = FT.NCTPS)**

**<Aninhada Correlacionada>**

1. Recupere o ID e o número de Parcelas dos Pagamentos feitos no cartão.

**SELECT pc.IdPagamento, pc.NumeroParcelas FROM pagamentoCartao pc**

**WHERE EXISTS**

**(SELECT \* FROM pagamento p)**

**<Aninhada Correlacionada>**

1. Recupere o ID e o número do cheque dos Pagamentos feitos no cheque.

**SELECT pc.IdPagamento, pc.NCheque FROM pagamentoCheque pc**

**WHERE EXISTS**

**(SELECT \* FROM pagamento p)**

**<Aninhada>**

1. Recupere o nome e o valor do pagamento de todas as pessoas, cujo valor do mesmo foi maior do que o pagamento de 'David' ou 'Joana'. O resultado deve ser ordenado pelo valor do pagamento em ordem descendente.

**select p.nome, pg.valor <Consulta Aninhada>**

**from Pessoa p join Pagamento pg on(p.id=pg.idPessoa)**

**where pg.valor > ALL**

**(Select pg.valor**

**from Pessoa p join Pagamento pg on(p.id=pg.idPessoa)**

**where p.nome= 'David' or p.nome= 'Joana')**

1. Crie um gatilho para conceder um desconto de 3% para todas as vendas acima de 300 reais.

**CREATE OR REPLACE FUNCTION descontoConcedido()**

**RETURNS TRIGGER AS $$**

**DECLARE**

**desconto real = 300;**

**BEGIN**

**SELECT INTO desconto FROM Venda**

**WHERE desconto > NEW.ID**

**GROUP BY ID;**

**UPDATE Venda SET valor = valor \* 0.97 WHERE id = NEW.ID;**

**RETURN NULL;**

**END**

**$$**

**LANGUAGE PLPGSQL;**

**CREATE TRIGGER desconta**

**AFTER INSERT ON Venda**

**FOR EACH ROW**

**EXECUTE PROCEDURE descontoConcedido();**

**--Teste do gatilho.**

**INSERT INTO Venda (ID, Data, TipoVenda, Valor, NNota, NCupom)VALUES(16, '2018-12-21','Empresa', 890, 8, null);**

1. Crie um gatilho para conceder um desconto de 1% para todos os pagamentos feitos a vista, cujo o valor esteja acima de 100 reais.

**CREATE OR REPLACE FUNCTION descontoPagamentoCartao()**

**RETURNS TRIGGER AS $$**

**DECLARE**

**desconto real = 100;**

**BEGIN**

**SELECT INTO desconto FROM PagamentosCartao**

**WHERE desconto > NEW.ID;**

**UPDATE Pagamento SET valor = valor \* 0.99 WHERE id = NEW.ID;**

**RETURN NULL;**

**END**

**$$**

**LANGUAGE PLPGSQL;**

**CREATE TRIGGER descontoCartao**

**AFTER INSERT ON Pagamento**

**FOR EACH ROW**

**EXECUTE PROCEDURE descontoPagamentoCartao();**

**--INSERT INTO Pagamento(ID, data, valor, idpessoa, ncupom)VALUES(13, '2018-07-10', 300, 5, 8)**

**--INSERT INTO PagamentoCartao(idpagamento, numeroparcelas)Values(13, 4);**

1. Crie um procedimento armazenado para retornar o total do valor do caixa operado por um determinado funcionario na data ‘X’ onde o funcionario e a data será passado por parâmetro.

**CREATE OR REPLACE FUNCTION totalCaixaFuncionarioData (DATE, INTEGER) RETURNS Caixa.ValorFechamento%type**

**AS $$**

**DECLARE**

**dia ALIAS FOR $1;**

**nCarteira ALIAS FOR $2;**

**soma Caixa.ValorFechamento%type := 0;**

**valor Caixa.ValorFechamento%type;**

**curs REFCURSOR;**

**BEGIN**

**OPEN curs FOR SELECT ValorFechamento FROM Caixa WHERE DataFechamento = dia and NCTPS = nCarteira;**

**FETCH curs INTO valor;**

**WHILE FOUND LOOP**

**soma := soma + valor;**

**FETCH curs INTO valor;**

**END LOOP;**

**CLOSE curs;**

**RETURN soma;**

**END**

**$$**

**LANGUAGE PLPGSQL;**

**--Testando o procedimento!**

**select totalCaixaFuncionarioData('2016-05-19', 5)**

1. Crie um Procedimento armazenado para retorna o somatório de todas as vendas realizadas em uma data passado por parâmetro, no mercado independente do tipo da venda se a mesma e consumidor ou empresa.

**CREATE OR REPLACE FUNCTION totalDasVendas (DATE) RETURNS Venda.valor%type**

**AS $$**

**DECLARE**

**dia ALIAS FOR $1;**

**totalVendas Venda.Valor%type := 0;**

**valoresVenda Venda.Valor%type;**

**ponteiro REFCURSOR;**

**BEGIN**

**OPEN ponteiro FOR**

**SELECT valor FROM Venda WHERE Data=dia;**

**FETCH ponteiro INTO valoresVenda;**

**WHILE FOUND LOOP**

**totalVendas := totalVendas + valoresVenda;**

**FETCH ponteiro INTO valoresVenda;**

**END LOOP;**

**CLOSE ponteiro;**

**RETURN totalVendas;**

**END**

**$$**

**LANGUAGE PLPGSQL;**

**--Testando o procedimento.**

**select totalDasVendas('2018-12-21')**

1. Crie uma visão que mostre a ficha da pessoa que fizer uma compra via cheque, a visão deve mostrar o Nome da pessoa, o CPF, a Rua onde mora, o Número da residência, a Cidade, o Bairro, o Estado, a Data do pagamento, o Valor da venda e o Número do Cheque.

**Create view fichaCliente as**

**select pf.cpf, p.nome,p.uf as Estado, p.cidade,p.bairro,p.rua, p.numero as NumeroCasa, pg.data as DataPagamento,pg.valor as ValorCompra, pc.ncheque**

**from Pessoa p, PessoaFisica pf, Pagamento pg, PagamentoCheque pc**

**where p.id=pf.idPessoa and pg.idPessoa=p.id and pg.id=pc.idPagamento**

1. Crie uma visão que mostre os dados de pessoa que fizer um pagamento avista, a visão deve mostrar o nome da pessoa, a data do pagamento, o valor do pagamento, e a data de efetivação do pagamento.

**create view pagamentosAvista as**

**select p.nome, pg.data as dataCompra, pg.valor as ValorCompra, pa.dataEfetivacao**

**from Pessoa p, Pagamento pg, pagamentoAvista pa**

**where p.id=pg.idPessoa and pg.id=pa.idPagamento**

1. Crie um gatilho para conceder um desconto de 2% para todos os pagamentos feitos avista cujo o valor esteja acima de 100 reais.

**CREATE OR REPLACE FUNCTION descontoAvistaPagamento()**

**RETURNS TRIGGER AS $$**

**DECLARE**

**desconto real = 100;**

**BEGIN**

**SELECT INTO desconto FROM PagamentosAvista**

**WHERE desconto > NEW.ID;**

**UPDATE Pagamento SET valor = valor \* 0.98 WHERE id = NEW.ID;**

**RETURN NULL;**

**END**

**$$**

**LANGUAGE PLPGSQL;**

**CREATE TRIGGER descontoAvista**

**AFTER INSERT ON Pagamento**

**FOR EACH ROW**

**EXECUTE PROCEDURE descontoAvistaPagamento();**

**--INSERT INTO Pagamento(ID, data, valor, idpessoa, ncupom)VALUES(12, '2018-07-10', 298, 5, 8)**

**--INSERT INTO PagamentoAvista(idpagamento, dataefetivacao)VALUES(12,'2016-03-19')**

1. Crie uma visão que mostre os dados da pessoa que fizer um pagamento via Cartão de crédito, a visão deve mostrar o nome da pessoa, a data do pagamento, o valor do pagamento.

**create view pagamentoscartao as**

**SELECT p.nome, pg.data as datapagamento, pg.valor as valorcompra**

**FROM pessoa p, pagamento pg, pagamentocartao pc**

**WHERE p.id = pg.idpessoa and pg.id = pc.idpagamento**

1. Crie uma visão que mostre todos os dados dos funcionários e do caixa.

**Create view public.caixafuncionario as**

**SELECT DISTINCT \***

**FROM caixa c NATURAL JOIN funcionario f**