| MySQL |

Documentação- https://dev.mysql.com/doc/refman/8.0/en/functions.html
https://www.w3schools.com/sql/default.asp

1 - Importando banco de dados

Acessando banco de Dados

use sucos_vendas;

2 - Conhecendo Banco de Dados Ao abrir o menu tables, notaremos que há cinco tabelas: "itens_notas_fiscais", "notas_fiscais", "tabela_de_clientes", "tabela_de_produtos" e "tabela_de_vendedores". Pelos próprios nomes, já temos uma boa noção do que elas tratam: "tabelas_de_vendedores" é a lista de vendodores, "tabela_de_produtos" são os produtos vendidos pela empresa de sucos, "tabela_de_clientes" é a lista de clientes, "notas_fiscais" e "itens_notas_fiscais" são referentes às notas fiscais.

Para gerar esse esquema visual, vamos até a barra superior do programa e selecionamos "Database > Reverse Engineer...". Em outras palavras, faremos uma engenharia reversa no nosso banco que já existe. Na caixa de diálogo que será aberta, selecionamos a nossa conexão usual, clicamos em "Next" duas vezes, selecionamos a base "sucos_vendas", clicamos em "Next" mais duas vezes e, em seguida, em "Execute". Com isso, será gerado um diagrama que representa as tabelas e suas relações. Você pode dar zoom, arrastar e organizar os elementos do esquema da maneira que preferir.

3 - Verificando as Chaves, PK FK

4 - Abri o Diagrama de Relacionamentos

5 - Condições

Se uma das expressoes for verdadeira a expressão completa sera verdadeira. Se todas as expressoes forem verdadeias a expressão completa será verdadeira. OR: Pode ser verdadeira ou falsa AND: Pode ser verdadeira ou falsa

NOT: inverte o resultado da consulta. (De falso para verdadeiro ou verdadeiro para falso)

EX: EXPRESSÃO

(NOT ((3 > 2) OR (4 >= 5)) AND (5 > 4)) OR (9 > 0) (NOT ((Verdadeiro) OR (Falso)) AND (Verdadeiro)) OR (Verdadeiro) (NOT (Verdadeiro) AND (Verdadeiro)) OR (Verdadeiro) (Falso AND Verdadeiro) OR (Verdadeiro) (Falso) OR (Verdadeiro) Verdadeiro

Sintax Condições:

SELECT * FROM tabela_de_produtos WHERE SABOR = "Manga" OR TAMANHO = '470 ml'; SELECT * FROM tabela_de_produtos WHERE SABOR = "Manga" AND TAMANHO = '470 ml'; SELECT * FROM tabela_de_produtos WHERE NOT SABOR = "Manga" AND TAMANHO = '470 ml';

SELECT * FROM tabela_de_produtos WHERE NOT SABOR = "Manga" OR TAMANHO = '470 ml'; SELECT * FROM tabela_de_produtos WHERE SABOR = "Manga" AND NOT (TAMANHO = '470 ml');

SELECT * FROM tabela_de_produtos WHERE SABOR IN ('Laranja', 'Manga');

SELECT * FROM tabela_de_produtos WHERE SABOR = 'Laranja' OR SABOR = 'Manga'; SELECT * FROM tabela_de_clientes WHERE CIDADE IN ('Rio de Janeiro', 'São Paulo') AND IDADE >= 20;

SELECT * FROM tabela_de_clientes WHERE CIDADE IN ('Rio de Janeiro', 'São Paulo') AND (IDADE >= 20 AND IDADE <=22);

USANDO LIKE

SELECT * FROM tabela_de_produto WHERE SABOR LIKE '%Maça%';

SELECT * FROM tabela_de_produtos WHERE SABOR LIKE '%Maça%' AND EMBALAGEM = 'pet'; SELECT * FROM tabela_de_clientes WHERE NOME LIKE '%Mattos';

USANDO DISTINCT

(Valores que não se repetem)

SELECT DISTINCT EMBALAGE, TAMANHO FROM tabela_de_prdutos; SELECT DISTINCT EMBALAGE, TAMANHO FROM tabela_de_prdutos;

USANDO LIMIT

SELECT * FROM taleba_de_produtos LIMIT 5; (Os cinco primeiros registro) SELECT * FROM taleba_de_produtos LIMIT 2,3; (Retorna 3 registros aparti do 2º) SELECT * FROM notas_fiscais WHERE DATA_VENDA = '2017-01-01' LIMIT 10; USANDO ORDE BY (ORDENAR) SELECT * FROM tabela_de_produtos ORDER BY NOME_DO_PRODUTO; SELECT * FROM tabela_de_produtos ORDER BY NOME_DO_PRODUTO DESC; SELECT * FROM tabela_de_produtos ORDER BY NOME_DO_PRODUTO ASC; SELECT * FROM tabela_de_produtos ORDER BY EMBALAGEM DESC, NOME_DO_PRODUTO ASC;

USANDO GROUP BY (AGRUPAR)

SELECT ESTADO, SUM(LIMITE_DE_CREDITO) AS LIMITE_TOTAL FROM tabela_de_clientes GROUP BY ESTADO;

SELECT EMBALAGEM, MAX(PRECO_DE_LISTA) AS MAIOR_PRECO FROM tabela_de_produtos GROUP BY EMBALAGEM; SELECT EMBALAGEM, COUNT*() AS CONTADOR FROM tabela_de_produtos GROUP BY EMBALAGEM; SELECT BAIRRO, SUM(LIMITE_DE_CREDITO) AS LIMITE FROM tabela_de_clientes GROUP BY BAIRRO; SELECT BAIRRO, SUM(LIMITE_DE_CREDITO) AS LIMITE FROM tabela_de_clientes WHERE CIDADE = 'Rio de Janeiro 'GROUP BY BAIRRO; SELECT ESTADO, BAIRRO, SUM(LIMITE_DE_CREDITO) AS LIMITE FROM tabela_de_clientes WHERE CIDADE = 'Rio de Janeiro ' GROUP BY ESTADO, BAIRRO ORDER BY BAIRRO ASC; SELECT MAX(QUANTIDADE) as 'MAIOR QUANTIDADE' FROM itens_notas_fiscais WHERE CODIGO_DO_PRODUTO = '1101035' ; SELECT COUNT*()* FROM itens_notas_fiscais WHERE codigo_do_produto = '1101035' AND QUANTIDADE = 99;

USANADO O HAVING

Having é uma condição de (Filtro) que se aplica ao resultado de uma agregação.

SELECT ESTADO, SUM(LIMITE_DE_CREDITO) AS SOMA_LIMITE FROM

tabela_de_clientes GROUP BY ESTADO HAVING SUM(LIMITE_DE_CREDITO) > 900000; SELECT EMBALAGEM, MAX(PRECO_DE_LISTA) AS MAIOR_PRECO , MIN(PRECO_DE_LISTA) AS MENOR_PRECO FROM tabela_de_produtos GROUP BY EMBALAGEM SELECT CPF, COUNT*() FROM notas_fiscais WHERE YEAR(DATA_VENDA) = 2016 GROUP BY CPF HAVING COUNT()* > 2000

USANDO CASE

Case - Fazemos um teste em um ou mais campos e, dependendo do resultado, teremos um ou outro valor.

SELECT NOME_DO_PRODUTO, PRECO_DE_LISTA, CASE WHEN PRECO_DE_LISTA >= 12 THEN 'PRODUTO CARO' WHEN PRECO_DE_LISTA >= 7 AND PRECO_DE_LISTA < 12 THEN 'PRODUTO EM CONTA' ELSE 'PRODUTO BARATO END AS STATUS_PRECO FROM tabela_de_produtos;

SELECT EMBALAGEM, CASE WHEN PRECO_DE_LISTA >= 12 THEN 'PRODUTO CARO' WHEN PRECO_DE_LISTA >= 7 AND PRECO_DE_LISTA < 12 THEN 'PRODUTO EM CONTA' ELSE 'PRODUTO BARATO END AS STATUS_PRECO, AVG(PRECO_DE_LISTA) AS PRECO_MEDIO FROM tabela_de_produtos;

SELECT EMBALAGEM, CASE WHEN PRECO_DE_LISTA >= 12 THEN 'PRODUTO CARO' WHEN PRECO_DE_LISTA >= 7 AND PRECO_DE_LISTA < 12 THEN 'PRODUTO EM CONTA' ELSE 'PRODUTO BARATO' END AS STATUS_PRECO, AVG(PRECO_DE_LISTA) AS PRECO_MEDIO FROM tabela_de_produtos WHERE sabor = 'Manga' GROUP BY EMBALAGEM, CASE WHEN PRECO_DE_LISTA >= 12 THEN 'PRODUTO CARO' WHEN PRECO_DE_LISTA >= 7 AND PRECO_DE_LISTA < 12 THEN 'PRODUTO EM CONTA' ELSE 'PRODUTO BARATO' END ORDER BY EMBALAGEM;

6 - JOINs INNER JOIN *= Traz somentes os correspondentes das daus tabelas LEFT JOIN = Retorna todos da tabela da esquerda e somente os correspondentes na tabela da direita RIGHT JOIN = Retorna todos da tabela da direita e somente os correspondetes da tabelada esquerda. FULL JOIN = Retorna todos os registros de todas as tabelas, quem não tiver registros ficara como NULL CROSS JOIN = Faz o produto cartesiano das duas tabelas.*

EX 1: SELECT * FROM tabela_de_vendedores A INNER JOIN notas_fiscais B ON A.MATRICULA = B.MATRICULA; -- Campos em comum entre as tabelas EX 2: (Obter o total de notas fiscais por vendedores) SELECT A.MATRICULA, A.NOME, COUNT(*) FROM tabela_de_vendedores A INNER JOIN notas_fiscais B ON A.MATRICULA =

B.MATRICULA -- Campos em comum entre as tabelas GROUP BY A.MATRICULA, A.NOME; EX 3:(Obtenha o faturamento anual da empresa. Leve em consideração que o valor financeiro das vendas consiste em multiplicar a quantidade pelo preço.) SELECT YEAR(DATA_VENDA), SUM(QUANTIDADE * PRECO) AS FATURAMENTO FROM notas_fiscais NF INNER JOIN itens_notas_fiscais INF ON NF.NUMERO = INF.NUMERO GROUP BY YEAR(DATA_VENDA) --Agrupando pelo Ano de venda

EX 4: (LEFT) --Traz todos os campos da tabela da esquerda e correspodente da direita SELECT DISTINCT A.CPF, A.NOME, B.CPF FROM tabela_de_clientes A LEFT JOIN notas_fiscais B ON A.CPF = B.CPF;
--Traz somente os campos NULL(CPF nulo) correspondentes da tabela da direita

SELECT DISTINCT A.CPF, A.NOME, B.CPF FROM tabela_de_clientes A LEFT JOIN notas_fiscais B ON A.CPF = B.CPF; WHERE B.CPF IS NULL SELECT DISTINCT A.CPF, A.NOME, B.CPF FROM tabela_de_clientes A LEFT JOIN notas_fiscais B ON A.CPF = B.CPF WHERE B.CPF IS NULL AND YEAR(B.DATA_VENDA) = 2015;

FULL JOIN e CROSS JOIN - ***Retorna clientes no mesmo bairro que os vendores tem escirotrio.***

SELECT * FROM tabela_de_vendedores INNER JOIN tabela_de_clientes ON tabela_de_vendedores.BAIRRO = tabela_de_clientes.BAIRRO; SELECT tabela_de_vendedores.BAIRRO, tabela_de_vendedores.NOME, tabela_de_clientes.BAIRRO, tabela_de_clientes.NOME FROM tabela_de_vendedores INNER JOIN tabela_de_clientes ON tabela_de_vendedores.BAIRRO = tabela_de_clientes.BAIRRO; -- Vai apresentar um erro, o Mysql não da suporte ao Full Join SELECT tabela_de_vendedores.BAIRRO, tabela_de_vendedores.NOME,DE_FERIAS, tabela_de_clientes.BAIRRO, tabela_de_clientes.BAIRRO, tabela_de_clientes.NOME

FROM tabela_de_vendedores FULL JOIN tabela_de_clientes ON tabela_de_vendedores.BAIRRO = tabela_de_clientes.BAIRRO;

-- Cross Join

SELECT tabela_de_vendedores.BAIRRO, tabela_de_vendedores.NOME, DE_FERIAS, tabela_de_clientes.BAIRRO, tabela_de_clientes.NOME FROM tabela_de_vendedores, tabela_de_clientes;

UNION - ***Faz união de duas ou mais tabelas. (Mesmo tipo e correspodnencias entre as tabelas)***

SELECT tabela_de_vendedores.BAIRRO, tabela_de_vendedores.NOME, DE_FERIAS, tabela_de_clientes.BAIRRO, tabela_de_clientes.NOME

FROM tabela_de_vendedores LEFT JOIN tabela_de_clientes ON tabela_de_vendedores.BAIRRO = tabela_de_clientes.BAIRRO UNION SELECT tabela_de_vendedores.BAIRRO, tabela_de_vendedores.NOME, DE_FERIAS, tabela_de_clientes.BAIRRO, tabela_de_clientes.NOME

FROM tabela_de_vendedores RIGHT JOIN tabela_de_clientes ON tabela_de_vendedores.BAIRRO = tabela_de_clientes.BAIRRO;

SUBCONSULTA

SELECT X.EMBALAGEM, X.PRECO_MAXIMO FROM (SELECT EMBALAGEM, MAX(PRECO_DE_LISTA) AS PRECO_MAXIMO FROM tabela_de_produtos GROUP BY EMBALAGEM) X WHERE X.PRECO_MAXIMO >= 10;

SELECT * FROM tabela_de_clientes WHERE BAIRRO IN (SELECT DISTINCT BAIRRO FROM tabela_de_vendedores);

Consulta:

SELECT CPF, COUNT*() FROM notas_fiscais WHERE YEAR(DATA_VENDA) = 2016 GROUP BY CPF HAVING COUNT()* > 2000

Subconsulta:

SELECT X.CPF, X.CONTADOR FROM (SELECT CPF, COUNT(*) AS CONTADOR FROM notas_fiscais WHERE YEAR(DATA_VENDA) = 2016 GROUP BY CPF) X WHERE X.CONTADOR > 2000

7 - VIEWS

É padrão usar as letras "VW" no início dos nomes das views, essa prática facilita a identificação das visões na hora de programar

CREATE VIEW 'VW_MAIORES_EMBALAGENS' AS SELECT EMBALAGEM, MAX(PRECO_DE_LISTA) AS MAIOR_PRECO FROM tabela_de_produtos GROUP BY EMBALAGEM

Também temos a opção de fazer, por exemplo, um JOIN com informações da tabela de produtos e dados da visão: Trata-se de um INNER JOIN entre uma tabela

e a view

SELECT A.NOME_DO_PRODUTO, A.EMBALAGEM, A.PRECO_DE_LISTA, X.MAIOR_PRECO FROM tabela_de_produtos A INNER JOIN vw_maiores_embalagens X ON A.EMBALAGEM = X.EMBALAGEM;

8 - FUNÇÕES (https://www.w3schools.com/mysql/mysql_ref_functions.asp)

SELECT NOME, CONCAT(ENDERECO_1, ' ', BAIRRO, ' ', CIDADE, ' - ', ESTADO) AS COMPLETO FROM tabela_de_clientes SELECT DISTINCT DATA_VENDA FROM NOTAS_FISCAIS; SELECT DISTINCT DATA_VENDA,DAYNAME(DATA_VENDA) AS DIA, MONTHNAME(DATA_VENDA) AS MES, YEAR(DATA_VENDA) AS ANO FROM NOTAS_FISCAIS; SELECT DAYNAME("2017-06-15") SELECT CURDATE(); SELECT YEAR(CURRENT_TIMESTAMP()); SELECT DAY(CURRENT_TIMESTAMP()); SELECT (23+((25-2)/2)*45) AS RESULTADO; SELECT CEILING(12.33333232323) AS RESULTADO; SELECT ROUND(12.33333232323) AS RESULTADO; SELECT ROUND(12.7777232323) AS RESULTADO; SELECT FLOOR(12.7777232323) AS RESULTADO; SELECT YEAR(DATA_VENDA), FLOOR(SUM(IMPOSTO * (QUANTIDADE * PRECO))) FROM notas_fiscais NF INNER JOIN itens_notas_fiscais INF ON NF.NUMERO = INF.NUMERO WHERE YEAR(DATA_VENDA) = 2016 GROUP BY YEAR(DATA_VENDA) SELECT CONCAT('O dia de hoje é: ', CURRENT_TIMESTAMP()) AS RESULTADO; SELECT CONCAT('O dia de hoje é: ', DATE_FORMAT(CURRENT_TIMESTAMP(),'%Y')) AS RESULTADO; SELECT CONCAT('O dia de hoje é: ', DATE_FORMAT(CURRENT_TIMESTAMP(),'%W, %d/%m/%Y')) AS RESULTADO; SELECT CONCAT('O dia de hoje é: ', DATE_FORMAT(CURRENT_TIMESTAMP(),'%d/%m/%Y - %U')) AS RESULTADO;

Queremos construir um SQL cujo resultado seja, para cada cliente: "O cliente João da Silva faturou 120000 no ano de 2016". Somente para o ano de 2016.

SELECT CONCAT('O cliente ', TC.NOME, ' faturou ', CAST(SUM(INF.QUANTIDADE * INF.preco) AS char (20)) , ' no ano ', CAST(YEAR(NF.DATA_VENDA) AS char (20))) AS SENTENCA FROM notas_fiscais NF INNER JOIN itens_notas_fiscais INF ON NF.NUMERO = INF.NUMERO INNER JOIN tabela_de_clientes TC ON NF.CPF = TC.CPF WHERE YEAR(DATA_VENDA) = 2016 GROUP BY TC.NOME, YEAR(DATA_VENDA);

| *Consulta com Vendas de Clientes Por Mês* |

SELECT NF.CPF, DATE_FORMAT(NF.DATA_VENDA, '%Y-%m') AS MES_ANO, SUM(INF.QUANTIDADE) AS QUANTIDADE_VENDAS FROM NOTAS_FISCAIS NF INNER JOIN ITENS_NOTAS_FISCAIS INF ON NF.NUMERO = INF.NUMERO GROUP BY NF.CPF, DATE_FORMAT(NF.DATA_VENDA, '%Y-%m'); |Limite de compra Por Cliente| SELECT TC.CPF, TC.NOME, TC.VOLUME_DE_COMPRA AS QUANTIDADE_LIMITE FROM TABELA_DE_CLIENTES TC;

Primeiro montamos uma seleção que determina se as vendas mensais por cliente são válidas ou não. Consideramos válidas vendas abaixo da quantidade limite e não válidas acima da quantidade limite existente no cadastro do cliente. A consulta é a mostrada abaixo

SELECT X.CPF, X.NOME, X.MES_ANO, X.QUANTIDADE_VENDAS, X.QUANTIDADE_LIMITE, CASE WHEN (X.QUANTIDADE_LIMITE - X.QUANTIDADE_VENDAS) < 0 THEN 'INVÁLIDA' ELSE 'VÁLIDA' END AS STATUS_VENDA FROM (SELECT NF.CPF, TC.NOME, DATE_FORMAT(NF.DATA_VENDA, '%Y-%m') AS MES_ANO, SUM(INF.QUANTIDADE) AS QUANTIDADE_VENDAS , MAX(TC.VOLUME_DE_COMPRA) AS QUANTIDADE_LIMITE FROM NOTAS_FISCAIS NF INNER JOIN ITENS_NOTAS_FISCAIS INF ON NF.NUMERO = INF.NUMERO INNER JOIN TABELA_DE_CLIENTES TC ON TC.CPF = NF.CPF GROUP BY NF.CPF, TC.NOME, DATE_FORMAT(NF.DATA_VENDA, '%Y-%m')) X;

/* QUANTIDADE VENDIDA POR SABOR ANO 2016 */

SELECT TP.SABOR, YEAR(NF.DATA_VENDA) AS ANO, SUM(INF.QUANTIDADE) AS QUANTIDADE FROM TABELA_DE_PRODUTOS TP INNER JOIN ITENS_NOTAS_FISCAIS INF ON TP.CODIGO_DO_PRODUTO = INF.CODIGO_DO_PRODUTO INNER JOIN NOTAS_FISCAIS NF ON NF.NUMERO = INF.NUMERO WHERE YEAR(NF.DATA_VENDA) = 2016 GROUP BY TP.SABOR, YEAR(NF.DATA_VENDA) ORDER BY SUM(INF.QUANTIDADE) DESC;

criaremos mais um relatório a pedido da área comercial da empresa de sucos de frutas. A solicitação é um acompanhamento sobre as vendas do ano de 2016 por sabores e eles gostariam de ver as informações no seguinte modelo

```
SELECT VENDA_SABOR.SABOR, VENDA_SABOR.ANO, VENDA_SABOR.QUANTIDADE,
ROUND((VENDA_SABOR.QUANTIDADE/VENDA_TOTAL.QUANTIDADE) * 100, 2) AS
PARTICIPACAO FROM (SELECT TP.SABOR, YEAR(NF.DATA_VENDA) AS ANO,
SUM(INF.QUANTIDADE) AS QUANTIDADE FROM TABELA_DE_PRODUTOS TP INNER
JOIN ITENS_NOTAS_FISCAIS INF ON TP.CODIGO_DO_PRODUTO =
INF.CODIGO_DO_PRODUTO INNER JOIN NOTAS_FISCAIS NF ON NF.NUMERO =
INF.NUMERO WHERE YEAR(NF.DATA_VENDA) = 2016 GROUP BY TP.SABOR,
YEAR(NF.DATA_VENDA)) AS VENDA_SABOR INNER JOIN (SELECT
YEAR(NF.DATA_VENDA) AS ANO, SUM(INF.QUANTIDADE) AS QUANTIDADE FROM
TABELA_DE_PRODUTOS TP INNER JOIN ITENS_NOTAS_FISCAIS INF ON
TP.CODIGO_DO_PRODUTO = INF.CODIGO_DO_PRODUTO INNER JOIN
NOTAS_FISCAIS NF ON NF.NUMERO = INF.NUMERO WHERE YEAR(NF.DATA_VENDA) =
2016 GROUP BY YEAR(NF.DATA_VENDA)) AS VENDA_TOTAL ON VENDA_SABOR.ANO =
VENDA_TOTAL.ANO ORDER BY VENDA_SABOR.QUANTIDADE DESC;
```

| MySQL |

Documentação- https://dev.mysql.com/doc/refman/8.0/en/functions.html
https://www.w3schools.com/sql/default.asp