Lucas Santos – GUIA SQL

CONSIDERAÇÕES IMPORTANTES

Este guia foi feito com base no uso da linguagem SQL no SQL SERVER

SQL (Structured Query Language) é uma linguagem usada para se comunicar com bancos de dados relacionais. Com ela, podemos criar bancos, tabelas, inserir dados, atualizar, buscar, deletar e muito mais.

OBS -> O SQL pode ser case sensitivy ou não dependendo da IDE e da forma que a mesma foi instalada (este guia utiliza tudo em minúsculo).

OBS.2-> A tabela utilizada nos exemplos é:

```
create table alunos (
  matricula INT,
  nome VARCHAR(100),
  cpf VARCHAR(20),
  rg VARCHAR(20),
  uf_rg VARCHAR(2),
  idade INT,
  curso VARCHAR(100)
);
```

Os nomes utilizados nos exemplos podem ser alterados para o contexto de uso do programador e de suas tabelas/banco de dados.

Categorias de Comandos SQL

Os comandos em SQL são separados em 4 categorias principais.

DDL -> Data Definition Language. Cria ou altera estruturas de dados.
Exemplo: Create, alter, drop.
DML -> Data Manipulation Language. Manipula os dados.
Exemplo: Insert, Update, delete.
DQL -> Data Query Language. Consulta os dados.
Exemplo: Select.
DCL -> Data Control Language. Controla permições.
Exemplo: Grant, Revoke.

COMANDOS SQL

Comandos de Criação: DDL

1) Constraints

São restrições que garantem a qualidade e consistência dos dados nas tabelas.

primary key -> Define que um campo da tabela é a chave primaria.
not null -> Impede que um campo receba valores nulos.
unique -> Garante que os valores não se repitam na tabela.
identity -> Gera automaticamente um valor pro campo. Sua incrementação pode ter o valor definido praviamente.
Exemplo: identity(valor_inicial, incremento)
Identity(1, 1) -> começa em 1 e soma 1 a cada chamada.
default -> Define um valor padrão para um campo.
check -> Impede valores fora de uma regra.
foreign key -> Define que um campo da tabela é uma chave estrangeira vinda de outra tabela.

Exemplos:

```
create table Clientes{
        id int primary key, //chave primaria
        nome varchar(50) not null, //campo não nulo
        cpf char(11) unique, //valor único no campo
        idade int check (idade >= 18), //idade deve ser maior que 18
        cidadeID int, // chave estrangeira
        constraint FK_cidade foreign key (cidadeID) //foreign key nomeada
        references Cidades(id)
}

Create table Cidades{
        id int primary key identity(1, 1), //identity
        nome varchar(100) notnull
}
```

2) Create

São responsáveis pela criação de tabelas e bancos de dados.

create database Nome_banco -> Cria um banco de dados.
Exemplo: **create database banco1 ->** Cria um banco de dados chamado banco1.

create table nome_tabela -> Cria uma tabela no banco de dados.
É recomendado que todos os campos da tabela sejam criados nesse momento.

Exemplo: **create table alunos{**
id int identity(1,1) primary key,
nome varchar(50) not null,
cpf varchar(20) not null,
rg varchar(20),
idade int not null,
curso varchar(100) not null
}

Comandos de Seleção: DQL

São utilizados para consultar dados de tabelas. É possível retonar todos os campos de uma tabela de uma vez, só alguns campos, campos com restrições e demais outras variedades de consultas.

* -> O operador "*" é um coringa utilizado para referenciar todas as colunas de uma tabela.

select * from alunos -> Retorna todas as colunas da tabela Alunos.

select campo1, campo2 from tabela -> É possível retornar so alguns campos da da tabela.
Exemplo: select id, nome, curso from alunos -> retorna 3 campos da tabela alunos.

Where – friltragem de dados

O comando where perm Permite impor restrições as consulas à tabela utilizando funções ou operadores presentes no SQL.

Operadores de comparação

São utilziados para realizar comparações de valores entre diferentes valores e campos presentes nas tabelas.

= -> Utilizado para validar se o valor do campo é igual ao requisitado.
Exemplo: select id, nome from alunos where curso = 'ADS'
Retorna o id e nome dos alunos que cursam ADS.

!= -> Utilizado para validar se o valor do campo é diferente ao requisitado.
Exemplo: select id, nome, curso from alunos where nome != 'Lucas'
Retorna o id,nome e curso dos alunos cujo nome não é Lucas.
OBS-> também pode ser representado por: <>

> -> Utilizado para validar se o valor do campo é maior que o requisitado.
Exemplo: select id, nome, curso from alunos where idade > 18
Retorna o id,nome e curso dos alunos cujo nome não é Lucas.

< -> Utilizado para validar se o valor do campo é menor que o requisitado.
Exemplo: select id, nome, curso from alunos where idade < 50
Retorna o id,nome e curso dos alunos cujo nome não é Lucas.

>= -> Utilizado para validar se o valor do campo é maior ou igual ao requisitado.
Exemplo: select id, nome, curso from alunos where id >= 30
Retorna o id,nome e curso dos alunos cujo nome não é Lucas.

<= -> Utilizado para validar se o valor do campo é menor ou igual ao requisitado.
Exemplo: select id, nome, curso from alunos where idade >= 18
Retorna o id,nome e curso dos alunos cujo nome não é Lucas.

Operadores Lógicos

São utilizados para realizar combinações de múltiplas condições difentes dentro de um mesma propriedade where.

and -> Operador “E”. Utizado para validar se todas as condições passadas são verdadeiras.

Exemplo: select id, nome, cpf from alunos where idade >= 18 and curso = ‘ADS’

Verifica se a idade do aluno é maior ou igual a 18 E se o curso do aluno é ADS. Caso as duas condições sejam verdadeiras, ele retorna a linha.

or -> Operador “OU”. Utilizado para validar se qualquer uma das condições é verdadeira.

Exemplo: select id, nome, cpf from alunos where curso = “Letras” or idade < 50.

Verifica se o curos do aluno é Letras ou a idade dele é menor que 50. Caso qualquer uma das condições seja verdadeira, ele retorna a linha.

not -> Operador “Negação”. Utilizado para validar a negação de uma condição.

Exemplo: select nome, curso from alunos where not(idade < 18)

Verifica se a idade do aluno é menor que 18. Caso não, ele retorna o aluno, que terá idade >= 18.

Operadores de Padrão e Faixa

São operadores especiais que trabalham com padrões, listas ou intervalos. Também aparecem geralmente em clausulas where.

between -> Operador "entre". Utilizado para verificar se o valor está dentro de um intervalo.
Exemplo: select * from alunos where id between 1 and 20
Verifica se o id do aluno está entre 1 e 20. Caso sim, retorna a linha.

in -> Operador "Dentro". Verifica se o valor está dentro de uma lista.
Exemplo: select id, nome, curso from alunos where curso in('ADS', 'Letras', 'Fisica')
Verifica se o aluno está em um dos 3 cursos. Caso sim, retorna a linha.
OBS: Alternativa para não usar excessivamente os operadores and e or.

is null -> Utilizado para verificar se o valor do elemento é nulo.
Exemplo: select * from alunos where rg is null
Verifica se o campo rg é nulo. Caso sim, retorna a linha

is not null -> Utilizado para verificar se o valor do elemento não é nulo.
Exemplo select * from alunos where rg is not null
Verifica se o campo rg não é nulo. Caso não seja, ele retorna a linha.

like -> Utilizado para comparar string com padrões utilizando os curingas _ e %.
_ -> Cada _ substitui um caractere na string que se deseja avaliar.
% -> Cada % substitui qualquer sequência de caracteres(incluindo espaços).

Exemplos de _:
select * from alunos where nome like 'A__'
Exibe todas as linhas cujos nomes tem 3 letras e começam com 'A'(Ana, Ari, Ada etc).

select * from alunos where nome like '_____'
Exibe todas as linhas cujos nomes tem 5 letras(Jorge, Lucia, Lucas etc).

select * from alunos where nome like '_u_a_'
Exibe todas as linhas cujos nomes tem 5 letras e a segunda e quarta letra são u e a. (Lucas).

Exemplos de %:

select * from alunos where nome like 'L%'

Exibe todas as linhas cujos nomes começam com L (Lucas, Lucia, Laura etc).

select * from alunos where nome like '%s'

Exibe todas as linhas cujos nomes terminam com s (Lucas, Agnes, Marcos etc).

select * from alunos where nome like '%a%'

Exibe todas as linhas cujos nomes possuam a letra a (Paula, Lucas, Angela etc).

select * from alunos where nome like '%S_t_o_%'

Exibe todas as linhas cujos nomes possuan a sequencia S_t_o_ de caracteres
(Lucas Santos, Silvio Santos, Alberto Santos Dummnt etc).

Clausula order by

Responsavel por ordenar os resultados por um ou mais campos da tabela, podendo definir a ordem da tabela (crescente ou decrescente).

asc -> Faz com que a consulta retorne os valores em ordem ascendente, ou seja, do menor para o maior.

Exemplo: select * from alunos order by idade asc

Exibe todas as linhas da tabela alunos de forma ascendente em idade (da menor idade em primeiro até a maior no fim).

desc -> Faz com que a consulta retorne os valores em ordem decrescente, ou seja, do maior para o menor.

Exemplo: select * from alunos order by idade desc

Exibe todas as linhas da tabela alunos de forma decrescente em idade (da maior idade em primeiro até a menor no fim).

Comandos de Manipulação: DML

Comandos de manipulação DML(Data Manipulation Language) são utilizados para manipular os dados dentro de tabelas. São separados em 3 comandos principais: Insert, Update e Detele.

1) Insert

O comando insert é utilizado para adicionar nova linha(com seus dados) em uma tabela pré existente.

insert into nm_tabela (coluna1, coluna2, ...) values (val1, val2, ...)
Exemplo:
insert into alunos (nome, cpf, rg, uf_rg, idade, curso)
values ('Lucas Silva', '12345678900', '1234567', 'SP', 20, 'ADS')

OBS: se os valores forem inseridos na ordem de criação das tabelas, os campos podem ser omitidos
Ex: insert into alunos values ('Lucas Silva', '12345678900', '1234567', 'SP', 20, 'ADS')

2) Update

O comando update é utilizado para modificar valores presentes em uma tabela pré existente em uma ou mais linhas.

update nome_tabela set coluna1 = valor1 where condição
Ex: uptdade idade set curso = 'Engenharia' where matrícula = 1
Obs: Sem o where o comando altera TODAS as linhas da tabela.
Ex: update idade set curso = 'Engenharia'

3) Delete

O comando delete é utilizado para deletar valores em uma ou mais linhas de uma tabela pré existem

delete from nome_tabela where condição
Ex: delete from alunos where matrícula = 2
Obs: Sem o where o comando apaga TODAS as linhas da tabela.
Ex: delete from alunos --apaga tudo

Funções de agregação

As funções de agragação são utilizadas para realizar cálculos em conjuntos de linahs e retornar **um único** valor como resutado.

1) count()

Utilizado para contar o numero de linhas onde uma determinada especificação aparece.

Exemplos básicos:
select count(*) from tabela
Conta todas as linhas da tabela.
select count(email) from alunos
Conta quantos alunos tem o campo email preenchido.
select count(distinct curso) from alunos
Conta quantos cursos diferentes existem na tabela.

Exemplo maior:
select count(*) as total_maiores_21 from alunos where idade > 20
Conta quantos alunos são maiores que 21 anos
OBS: select count(idade) as total_maiores_21 from alunos where idade > 20
Isso tbm funciona igual.

2) sum()

Utilizado para somar os valores de uma coluna numérica.

Exemplo básico:
select sum(idade) from alunos
Soma todas as idades dos alunos

Exemplo maior:
select sum(valor) as Total_credito from vendas where forma_pagamento = 'crédito'
Soma todos os valores do campo valor onde a forma de pagamento é credito.

3)avg()

Calcula a média de valores de uma coluna numérica.

Exemplo básico:
select avg(idade) from alunos
Retorna a média aritmética da idade dos alunos.

Exemplo maior:
select avg(idade) as media_ADS from alunos where curso = 'ADS'
Retorna a média das idades dos alunos de ADS.

4)max

Retorna o maior valor de uma coluna.

Exemplo básico:
select max(idade) from alunos
Retorna a maior idade da tabela alunos.

Exemplo maior:
select nome, preco from produtos
where preco = (select max(preco) from alunos)
Retorna o nome e o preço do produto com o maior preço.

4)min

Retorna o menor valor de uma coluna.

Exemplo básico:
select max(idade) from alunos
Retorna a menor idade da tabela alunos.

Exemplos maior:
select nome, preco from produtos
where preco = (select min(preco) from alunos)
Retorna o nome e o preço do produto com o menor preço.

select min(salario) as menor_salario from funcionários where cargo = 'TI'
Retorna o menor valor entre os que possuem cargo igual a TI

Group BY e having

Group by

Group by serve para agrupar os dados por uma ou mais colunas, permitindo a utilização de funções de agregação (como count, avg, sum, etc) em cada grupo de valores semelhantes.

Exemplo simples:
select curso, count(*) as total_alunos from alunos order by curso
Agrupa os alunos por curso e conta quantos há em cada curso.

Having

O having serve para filtrar os resultados agregados depois do group by. Ele é como o where, só que usado para filtrar **os grupos**, não as linhas da individuais**.**

Exemplo simples:
select curso, count(*) as total_alunos from alunos having count(*) > 10
Retorna apenas os cursos com mais de 10 alunos.

Exemplos group by com Funções de Agregação

1)Count

select curso, count(email) as e-mails-preenchidos from alunos
group by curso
having count(email) > 5
Conta quantos alunos tem email preenchido em cada curso e mostra só os cursos coim mais de 5.

2) Sum

select forma_pagamento, sum(valor) as total_vendas from vendas
group by forma_pagamento
having sum(valor) > 1000
Soma os valores de vendas por formato de pagamento. Mostra só as formas com mais de R$ 1000 em vendas.

3) Avg
select curso, avg(idade) as media_idade from alunos
group by curso
having avg(idade) > 22
Calcula a média de idade por curso. Mostra só os cursos com média de idade maior que 22 anos.

4) Max
select curso, max(idade) as maior_idade from alunos
group by curso
having max(idade) >= 30
Encontra a maior idade por curso. Mostra só os cursos em que alguém possua 30 anos ou mais.

5) Min
select curso, min(idade) as menor_idade from alunos
group by curso
having min(idade) < 18
Encontra a menor idade por curso. Mostra só os cursos com alunos menores de idade.

Estrutura de Condição Case

Utilizado para gerar e validar condições dentro de uma consulta SQL. Funciona como um if – else, podendo atribuir valores diferentes a diferentes condições

Exemplo simples:
select nome, idade
case idade >=18 then 'maior de idade'
else 'menor de idade'
End as situacao from alunos
Verifica a idade e retorna "maior de idade" ou "menor de idade" para cada aluno.

1) Mais de uma clausula

select nome, curso
case
when curso = 'ADS' then 'Tecnologia'
when curso = 'Direito' then 'Humanas'
else 'Outros'
end as area from alunos
Define a área com base no curso.

2) Exemplo Manipulando faixa de idade

```
select nome, idade
case
 when idade < 18 then 'Menor de idade'
 when idade <= 30 then 'adulto joven'
else 'Adulto'
end as faixa from alunos
```

3) Exemplo com Funções de agregação

```
select curso, count(*) as total,
sum(case when idade >=18 then 1 else 0 end) as maiores_idade
from alunos group by curso
```

Conta quantos alunos há por curso e quantos são adultos. Tudo na mesma consulta.

Join – Junção de tabelas

O join é utilizado para combinar duas ou mais tabelas com base em uma coluna em comum entre as tabelas.

1)Inner join
Retorna os registros que existem nas duas tabelas.
Exemplo simples:
select alunos.nome, cursos.nome as nome_curso from alunos
inner join cursos on alunos.id_curso = cursos.id
Retorna os alunos que têm um curso correspondente cadastrado.

Exemplos maior:
select alunos.nome, cursos.nome as nome_cursofrom alunos
inner join cursos on alunos.id_curso = cursos.id
where cursos.nome = 'ads'
Mostra apenas os alunos do curso ADS.

2) Left join
Retorna todos os registros da tabela da esquerda(após o from) mesmo os que não possuem correspondência na tabela de baixo.
Exemplo simples:
select alunos.nome, cursos.nome as nome_curso from alunos
left join cursos on alunos.id_curso = cursos.id;
Retorna todos os alunos, mesmo que não tenham curso. Se não tiver, o campo nome_curso fica NULL.

Exemplo maior:
select clientes.nome, pedidos.id as pedido_id from clientes
left join pedidos on clientes.id = pedidos.id_cliente
where pedidos.id is null
Mostra apenas os clientes que ainda não têm pedido (pedidos nulos).

3) Right join
Parecido com o LEFT JOIN, mas retorna todos os registros da tabela da direita, mesmo que não tenham correspondência na da esquerda.
Exemplo simples:
select alunos.nome, cursos.nome as nome_curso from alunos
right join cursos on alunos.id_curso = cursos.id
Mostra todos os cursos, mesmo os que não têm alunos matriculados.

Exemplo maior:
select alunos.nome, cursos.nome as nome_curso from alunos
right join cursos on alunos.id_curso = cursos.id
where alunos.id is null
Mostra os cursos que não têm nenhum aluno matriculado.

Mais de um JOIN

É possível utilizar junções de mais de um join para realizar consultas entre tabelas.

1)Inner join com 3 tabelas
Exemplo:
select alunos.nome as aluno, cursos.nome as curso, professores.nome as professor from alunos
inner join cursos on alunos.id_curso = cursos.id
inner join professores on cursos.id_professor = professores.id
Traz o nome do aluno, o curso que ele faz e o nome do professor responsável por esse curso.

2) Left join + inner join
Exemplo:
select clientes.nome, pedidos.id as pedido_id, pagamentos.status from clientes
left join pedidos on clientes.id = pedidos.id_cliente
inner join pagamentos on pedidos.id = pagamentos.id_pedido
Traz todos os clientes com seus pedidos e o status de pagamento, mas só aparece pagamento se existir (INNER JOIN com pagamentos).

3) Inner join + mais de um critério ON
Exemplo:

```
select produtos.nome, categorias.nome as categoria, fornecedores.nome as
fornecedor from produtos
inner join categorias on produtos.id_categoria = categorias.id
inner join fornecedores
   on produtos.id_fornecedor = fornecedores.id
  and fornecedores.ativo = 1
```

Traz nome do produto, categoria e fornecedor, mas só se o fornecedor estiver ativo (condição no ON).

4) Join + where
Exemplo:

```
select a.nome as aluno, c.nome as curso, p.nome as professor
from alunos a
inner join cursos c on a.id_curso = c.id
inner join professores p on c.id_professor = p.id
where c.nome = 'engenharia';
```

Mostra os alunos de Engenharia e quem são os professores deles.

Subquery

Subqueries são consultas dentro de uma outra consulta. Elas são usadas para retornar valores que serão utilizados em cláusulas como where, from, select etc.

1) Subquery no WHERE
Serve para usar o resultado da subquery como critério de filtro.
Exemplo simples:

```
select nome from alunos where idade > (select avg(idade) from alunos);
```

Retorna os alunos com idade acima da média.

Exemplo maior:

```
select nome, preco from produtos where preco = (select max(preco) from produtos);
```

Retorna o(s) produto(s) mais caro(s).

2) Subquery na cláusula FROM

Usada para criar uma “tabela temporária” com base em outra consulta.

Exemplo simples:

select media_idade from (select avg(idade) as media_idade from alunos) as resultado;

Retorna a média da idade dos alunos. A subquery é tratada como uma tabela chamada "resultado".

Exemplo maior:

```
select curso, media_idade
from (select curso, avg(idade) as media_idade from alunos
group by curso) as medias
where media_idade > 25;
```

Retorna os cursos cuja média de idade dos alunos é maior que 25.

3) Subquery no Select

Serve para mostrar o resultado da subquery como coluna.

Exemplo simples:

select nome, (select avg(idade) from alunos) as media_geral from alunos

EXERCICIOS FEITOS NA DISCIPLINA

Banco de Dados II - Exercício AULA 01
Crie consultas que retornem os seguintes dados

j) todas as colunas dos alunos do curso de computação e que tenham 20 anos
Resposta: select * from alunos where curso = "Computação" and idade = 20

k) nome, curso, cpf e rg dos alunos de direito cujo rg foi emitido em São Paulo
Resposta: select nome, curso, cpf, rg from alunos where curso = "Direito" and uf_rg = "SP"

l) nome, curso, cpf e rg dos alunos de biologia cujo rg foi emitido no Rio de Janeir
Resposta: select nome, curso, cpf, rg from alunos where curso = "Biologia" and uf_rg = "RJ"

m) todas as colunas dos alunos cujo curso seja direito ou biologia
Resposta: select * from alunos where curso = "Biologia" or curso = "Direito"

n) todas as colunas dos alunos cujo curso seja direito ou biologia e tenham 20 anos
Resposta: select * from alunos where ((curso = "Biologia" or curso = "Direito") and idade = 20)

A qual categoria de comandos SQL pertencem os seguintes comandos já vistos (DDL,DML,DCL):

a) Create Database dbAcademico _____
Resposta: DDL
b) Insert into Alunos values (10,'657843901-98','2345678','SP',22,'Computação')___
Resposta: DML
c) Alter Table Alunos Drop Column UF_RG _____
Resposta: DDL
d) Select * from Alunos _____
Resposta: DQL
e) Drop Table Alunos _____
Resposta: DDL

Banco de Dados II - Exercicios Consulta Funções
Construa consultas que retornem:

d) registros cujos ceps sejam maiores ou iguais a 11538-000 e menores que 11540-000
R: select * from CepBaixada where cep between 11538-000 and 11539-999

e) exibir o campo EnderecoCompleto(que será a concatenação dos campos tipo, titulo e nome)
R: select tipo + ' ' + titulo + ' ' + nome as EnderecoCompleto from CepBaixada

g) todos os registros cujo campo nome se inicia com "B", "C" ou "D"
R: select * from CepBaixada where Nome like 'B%' or Nome like 'C%' or Nome like 'D%'

h) todos os registros cujo campo nome se inicia com "B", "E", "F"
R: select * from CepBaixada where Nome like 'B%' or Nome like 'E%' or Nome like 'F%'

i) todos os registros cuja segunda letra seja "e" e a quarta letra seja "n"
R: select * from CepBaixada where nome like '_e_n%'

j) todos os registros em cujo nome tenha a palavra "Campo" não importando a posição
R: select * from CepBaixada where nome like '%Campo%'

k) todos os registros cujo nome se inicia com a palavra "Campo"
R: select * from CepBaixada where nome like 'Campo%'

l) todos os registros cujas cidades não sejam Santos ou Cubatão
R: select * from CepBaixada where Cidade not in ('Santos', 'Cubatão')

m) todos os registros cuja cidade seja Santos ou São Vicente e que possua a string "centr" no nome do logradouro
R: select * from CepBaixada where Cidade in('Santos', 'São Vicente') and logradouro like '%centr%'

n) data de hoje
R: select getdate()

o) mês corrente
R: select month(getdate())

p) ano corrente
R: select year(getdate())

q) diferença em dias entre seu nascimento e a data de hoje
R: select datediff(day, '2004-11-19', getdate())

r) diferença em semanas entre seu nascimento e a data de hoje
R: select datediff(week, '2004-11-19', getdate())

s) diferença em minutos entre seu nascimento e a data de hoje
R: select datediff(minute, '2004-11-19', getdate())

t) Trinta e quatro dias atrás
R: select dateadd(day, -34, getdate())

u) Cento e Vinte Dias após a data de hoje
R: select dateadd(day, 120, getdate())

v) Quatro meses após data de hoje
R: select dateadd(month, 4, getdate())

w) Dois elevando a Oito
R: select power(2, 8)

x) Circunferência de um círculo de raio 10 cm
R: select 2 * pi() * 10

y) Área de um círculo de raio 10 cm
R: select pi() * power(10, 2)

z) Altere todos os registros dos bairros Centro e Boqueirão da cidade de Santos para que
sejam grafados em maiúsculos.

R: update CepBaixada set bairro = UPPER(bairro) where cidade = 'Santos' and bairro IN ('Centro', 'Boqueirão')

Trabalho Prático 16 de Abril

3) Atualize com apenas um UPDATE a coluna ds_situacao com o seguinte critério:
R: update notas set ds_situacao =

```
CASE
        when vl_media >= 6 then 'Aprovado'
        when vl_media < 3 then 'Reprovado'
        else 'Exame'
end
```

4) Execute uma cláusula para arredondar as notas dos alunos seguindo o critério:
R: update notas set vl_media =

```
CASE
        when vl_media - floor(vl_media) <= 0.25 then floor(vl_media)
        when vl_media - floor(vl_media) >= 0.75 then floor(vl_media) + 1
        else floor(vl_media) + 0.5
end
```

5) Obtenha a quantidade de alunos, a maior nota, a menor nota e a média geral de todos os alunos
R:select

```
COUNT (nm_aluno) as Quantidade_Alunos,
max(vl_media) as Maior_nota,
min(vl_media) as Menor_nota,
avg(vl_media) as Media_Geral
from notas
```

7) Execute uma cláusula para mostrar os dados da tabela em ordem decrescente de média e crescentede nome
R:select * from notas order by vl_media desc, nm_aluno asc

8) Obtenha a lista (nome e média) dos 10 alunos com melhor desempenho (10 melhores médias)
R: select top 10 nm_aluno, vl_media from notas order by vl_media desc

9) Obtenha a lista (nome e média) dos 10 alunos com pior desempenho (10 piores médias)
R: select top 10 nm_aluno, vl_media from notas order by vl_media

10) Obtenha uma lista com o valor da média e número de alunos que obtiveram aquela média

R: select floor(avg(vl_media)) as Media, count(*) as numero_alunos from notas
where vl_media = (select avg(vl_media) from notas)

11) A partir da tabela notas crie duas outras tabelas:
aprovados (com todos os alunos cuja média seja igual ou superior a 6.0)
exame (com todos os alunos cuja média seja inferior a 6.0)

R:select * into aprovados from notas
where vl_media >= 6

select * into exame from notas
where vl_media < 6

12) Montar uma consulta que retorne uma msg que um sistema de e-mail enviará a todos os alunos. A msg deverá ter o seguinte formato. Caro(a) nome do aluno(a) – Sua Situação é: situação - exibir ainda após a situação as seguintes mensagens: - Parabéns (para os alunos aprovados), - Você deverá comparecer no dia 30/04 para a prova final (para os alunos de exame) e Infelizmente você deverá refazer a disciplina no próximo semestre (para os alunos reprovados).

R:select 'Caro(a) ' + nm_aluno + ' sua situação é: ' + ds_situacao + ' | ' +
case
when ds_situacao = 'Aprovado' then ' Parabéns'
when ds_situacao = 'Exame' then ' Você deverá comparecer no dia 30/04 para a prova final'
else ' Infelizmente você deverá refazer a disciplina no próximo semestre'
end as mensagem
from notas

12) Listar toda a tabela de vendas
R: select * from vendas

13) qual o valor total de vendas?
R: select sum(vl_venda) from vendas

14) qual o valor das vendas do vendedor 1?
R: select sum(vl_venda) from vendas where cd_vendedor = 1

15) qual a venda com valor mínimo?
R:select * from vendas where vl_venda = (select min(vl_venda) from vendas)

16) qual a venda com valor máximo?
R:select * from vendas
where vl_venda = (select max(vl_venda) from vendas)

17) qual a média de vendas?
R: select avg(vl_venda) as Media_Vendas from vendas

18) quantas vendas foram feitas?
R: select count(vl_venda) as Numero_de_vendas from vendas

19) quantas vendas o vendedor 1 fez?
R: select count(*) as Quantidade_Vendas from vendas
where cd_vendedor = 1

20) qual o valor médio de vendas do vendedor 1?
R: select avg(vl_venda) from vendas
where cd_vendedor = 1

21) monte uma consulta que retorne o vendedor e a soma total de suas vendas
R: select cd_vendedor, sum(vl_venda) from vendas
group by cd_vendedor
22) monte uma consulta que em cada linha retorne o código do vendedor,
a soma de suas vendas, quantas vendas ele fez, a média de suas vendas, sua menor venda e sua venda máxima
R: select cd_vendedor,
sum(vl_venda) as Soma_Vendas,
count(vl_venda) as Numero_Vendas,
avg(vl_venda) as Media_vendas,
min(vl_venda) as Menor_Venda,
max(vl_venda) as Maior_venda from vendas
group by cd_vendedor

23) Quantos vendedores distintos temos nessa tabela?
R: select count(distinct cd_vendedor) as Quantidade_de_Vendedores from vendas

EXERCICIOS JOIN

Exercicio 1

TP – Join, PK, Relacionamento. Funções Agregadas

01) Sobre CHAVE PRIMARIA (PRIMARY KEY) podemos afirmar:
(X) a. É uma coluna em sua tabela que faz com que cada registro seja único
( ) b. Os valores da chave primaria não podem ser alterados
(X) c. Um valor deve ser atribuído na chave primária sempre que um registro for inserido em sua tabela
( ) d. Pode ser do tipo NULL

02) Sobre ORDER BY podemos afirmar que:
( ) a. Ordena a consulta baseado em uma única coluna
(X) b. Podem ser usadas várias colunas na ordenação
( ) c. A ordenação default é descendente. Caso queira pode ser mudada usando a palavra chave ASC
( ) d. Não podemos usar colunas de diferentes tipos na mesma declaração ORDER BY

03) Sobre o comando IDENTITY (SQL Server):
( ) a. Preencha automaticamente a coluna que é PRIMARY KEY com um valor INT que se auto-numera de 1 em 1
(X) b. Em outra variação do comando, pode se auto-numerar com valores definidos pelo usuário
(X) c. Na cláusula INSERT, devemos omitir nome do campo e valor a ser inserido
( ) d. Preenche automaticamente a coluna que é PRIMARY KEY com um valor DEFAULT(pré-gravado)

04) Sobre cláusulas SQL:
(X) a. A função AVG soma os valores da coluna e depois divide pelo número de registros
( ) b. A função GROUP BY usada com a função COUNT soma os valores da coluna agrupando a soma por grupos de valores.
( ) c. A função COUNT é utilizada para retornar os maiores valores de uma coluna
( ) d. A função DISTINCT usada juntamente com o DESC, retorna os dados da consulta invertidos em sua ordem

05. Sobre cláusulas SQL
( ) a. Select top 10 from clientes order by Cd_Cliente ASC retorna os 10 primeiros clientes da tabela clientes
(X) b. Select nome,case when sexo = 0 then ‘Masculino’ else ‘Feminino' end, Cpf from clientes – retorna os
registros substituindo o valor 0 do campo sexo pelo valor ‘Masculino’ e ‘Feminino’ em caso contrario.
(X) c. Select * from clientes into clienteNovo – cria uma nova tabela chamada clienteNovo, com todos os
campos e registros da tabela cliente.
(X) d. Select count(distinct bairro) from clientes where UF = ‘SP’ retorna a quantidade de bairros distintos da
tabela clientes cuja UF (Unidade da Federação’ seja ‘SP’

USANDO O RELACIONAMENTO DAS TABELAS ABAIXO RESPONDA AS QUESTÕES

06, 07 e 08.

06) A consulta acima é resultado de que cláusula abaixo:
(X) a. select e.cd_cliente,e.nome_empresa,e.cargo,c.nome_conjuge from conjuge c right join empresa e on c.cd_cliente = e.cd_cliente
( ) b. select e.cd_cliente,e.nome_empresa,e.cargo,c.nome_conjuge from conjuge c left join empresa e on c.cd_cliente = e.cd_cliente
( ) c. select e.cd_cliente,e.nome_empresa,e.cargo,c.nome_conjuge from conjuge c inner join empresa e on c.cd_cliente = e.cd_cliente
( ) d. select e.cd_cliente,e.nome_empresa,e.cargo,c.nome_conjuge from empresa e inner join conjuge c on c.cd_cliente = e.cd_cliente

07 – Monte uma consulta que retorne os dados dos cônjuges dos clientes que residem nos seguintes bairros:
(Vila Belmiro, Campo Grande, Centro)
R: select conjuge.* from clientes inner join conjuge on ccliente.cd_cliente = conjuge.cd_cliente
where cliente.bairro IN('Vila Belmiro', 'Campo Grande', 'Centro')

08 – Monte uma consulta que retorne o nome do bairro e quantas ocorrências existem desse bairro. Essa
consulta deverá vir ordenada da maior quantidade para a menor, conforme exemplo abaixo.

R: select bairro, count(*) as quantidade
from(
select bairro from empresa
union all
select bairro from cliente
) as todos_bairros
where bairro is not null
group by bairro
order by quantidade desc

EXERCICIO 2

4) Crie as tabelas relacionadas no diagrama abaixo (ATENÇÃO: a estrutura está na forma normal, portanto
defina chaves primarias e estrangeiras adequadamente)
R: create table fornecedor
(

cd_fornecedor int,
nm_fornecedor varchar(50),
primary key (cd_fornecedor)

)

create table material
(

cd_material int,
nm_material varchar(50),
vl_unitario money,
primary key (cd_material)

)

```
create table ordemcompra
(

        cd_ordemcompra int,
        dt_ordemcompra smalldatetime,
        cd_fornecedor int,
        primary key (cd_ordemcompra),
        foreign key (cd_fornecedor) references fornecedor (cd_fornecedor)

)

create table item
(

        cd_ordemcompra int,
        cd_material int,
        qt_material int,
        primary key (cd_ordemcompra, cd_material),
        foreign key (cd_ordemcompra) references ordemcompra(cd_ordemcompra),
        foreign key (cd_material) references material (cd_material)

)
```

5) Execute a seguinte clausula:
insert ordemcompra values (1,'2008-05-14',1) ;
Porque a execução desta clausula provoca uma mensagem de erro?

R: O erro ocorreu porque o comando insert tentou inserir um valor na tabela ordemcompra, mas como
ordemcompra possui uma chave estrangeira(cd_fornecedor) na tabela fornecedor que não possue esse valor
a compilação resulta em erro para manter a integridade do banco de dados.

7) Escreva uma clausula que liste todas as ordens de compras realizadas incluindo os seguintes dados: código do fornecedor, nome do fornecedor, código da ordem de compra, data da ordem de compra.
R: select o.cd_fornecedor as codigo_fornecedor,
f.nm_fornecedor as Nome_fornecedor,
o.cd_ordemcompra as codigo_ordemCompra,
o.dt_ordemcompra as dataCompra
from ordemcompra as o
left join fornecedor as f on o.cd_fornecedor = f.cd_fornecedor

8) Escreva uma clausula para listar as colunas código do material, nome do material, valor unitário, código da
ordem de compra, quantidade solicitada dos materiais cujo código esteja na lista: 1, 2, 3, 4, 5. Deverão ser listados os dados de todas as ordens de compra para estes materiais. Notar que mesmo se os materiais não possuírem nenhuma ordem de compra eles também deverão ser listados.
R: select
m.cd_material as codigo_material,
m.nm_material as nome_material,
m.vl_unitario as valor_unitario,
o.cd_ordemcompra as codigo_ordemCompra,
i.qt_material as quantidade_solicitada
from material as m
left join item as i on m.cd_material = i.cd_material
left join ordemcompra as o on i.cd_ordemcompra = o.cd_ordemcompra
where m.cd_material in (1, 2, 3, 4, 5);

9) Apague a tabela de fornecedores (DROP TABLE). A tabela foi apagada? Qual a causa deste comportamento?
R: o comando resultou em erro. E isso ocorreu porque existe referência a campos da tabela fornecedor
em outras tabelas, por isso o compilador impede que a tabela seja apagada para manter a integridade do banco de dados

EXERCICIOS SUBQUERY

Utilizando o Banco de Dados, Tabelas e dados do exercício anterior cuja estrutura relembramos abaixo:

1) Escreva uma clausula que liste todos os códigos de ordem de compra dos itens onde a quantidade solicitada foi superior a 10. (use a tabela
item)
R: select cd_ordemcompra from item where qt_material > 10

2) Escreva uma clausula que liste código do fornecedor, nome do fornecedor e código de ordem de compra de todas as ordens de compra.
(junção das tabelas fornecedor e ordemcompra)
R: select f.cd_fornecedor, f.nm_fornecedor, o.cd_ordemcompra
from ordemcompra as o
inner join fornecedor as f on o.cd_fornecedor = f.cd_fornecedor

3) Combine as clausulas das 2 questões anteriores de maneira a criar uma clausula que liste código do fornecedor, nome do fornecedor e
código de ordem de compra de todas as ordens de compra onde há itens cuja quantidade solicitada é superior a 10. (use a subquery na
condição da query)
R: select f.cd_fornecedor, f.nm_fornecedor, o.cd_ordemcompra from ordemcompra as o
inner join fornecedor as f on o.cd_fornecedor = f.cd_fornecedor
where o.cd_ordemcompra in(select cd_ordemcompra from item where qt_material> 10)

4) Elabore uma clausula para obter o valor total da ordem de compra de código 1. (use SUM (vl_unitario * qt_material) na junção das tabelas
item e material.)
R: select sum(m.vl_unitario * i.qt_material) as ValorTotal from material as m
inner join item as i on m.cd_material = i.cd_material
where i.cd_ordemcompra = 1

5) Elabore uma clausula para listar código do fornecedor, nome do fornecedor, código da ordem de compra de todas as ordens de compra.
(junção das tabelas fornecedor e ordemcompra)
R: select f.cd_fornecedor, f.nm_fornecedor, o.cd_ordemcompra
from ordemcompra as o
inner join fornecedor as f on o.cd_fornecedor = f.cd_fornecedor

6) Combine as clausulas das 2 questões anteriores de maneira a listar código do fornecedor, nome do fornecedor, código da ordem de compra
e valor total (calculado) de todas as ordens de compra. (use a subquery como uma coluna da outer query.)
R: select f.cd_fornecedor, f.nm_fornecedor, o.cd_ordemcompra, (select sum(m.vl_unitario * i.qt_material) from material as m
inner join item as i on m.cd_material = i.cd_material
where i.cd_ordemcompra = o.cd_ordemcompra) as ValorTotal
from ordemcompra as o
inner join fornecedor as f on o.cd_fornecedor = f.cd_fornecedor