Programação Cunha

Índice

[SQL 1](#_Toc105848055)

[1. Criar um banco de dados: 1](#_Toc105848056)

[2. Apagar um banco de dados: 1](#_Toc105848057)

[3. SELECT 1](#_Toc105848058)

[4. DISTINCT 1](#_Toc105848059)

[5. Sinalização 1](#_Toc105848060)

[6. WHERE 2](#_Toc105848061)

[7. COUNT 2](#_Toc105848062)

[8. TOP 3](#_Toc105848063)

[9. AS 3](#_Toc105848064)

[10. order by 3](#_Toc105848065)

[11. BETWEEN 3](#_Toc105848066)

[12. IN 4](#_Toc105848067)

[13. LIKE 4](#_Toc105848068)

[14. FUNÇÕES: MIN MAX SUM AVG 5](#_Toc105848069)

[15. GROUP BY 6](#_Toc105848070)

[16. HAVING 7](#_Toc105848071)

[17. INNER JOIN 8](#_Toc105848072)

[18. TIPOS DE JOINS: 9](#_Toc105848073)

[19. UNION 11](#_Toc105848074)

[20. DATEPART 12](#_Toc105848075)

[21. MANIPULAÇÃO DE STRINGS 12](#_Toc105848076)

[22. FUNÇÕES MATEMATICAS 14](#_Toc105848077)

[23. SUBQUERY (SUBSELECT) 15](#_Toc105848078)

[MUDAMOS DE TABELA E COMEÇAMOS A TRABALHAR COM A NORTHWIND 17](#_Toc105848079)

[24. SELF JOIN 17](#_Toc105848080)

[TIPOS DE DADOS: 17](#_Toc105848081)

[25. Chave Primária e Estrangeira 19](#_Toc105848082)

[26. CREATE TABLE: 20](#_Toc105848083)

[27. INSERT 22](#_Toc105848084)

[28. UPDATE 24](#_Toc105848085)

[29. DELETE 25](#_Toc105848086)

[30. alter table 26](#_Toc105848087)

[31. drop table 27](#_Toc105848088)

[32. CHECK CONSTRAINT 28](#_Toc105848089)

[33. NOT NULL CONSTRAINT 28](#_Toc105848090)

[34. UNIQUE CONSTRAINT 29](#_Toc105848091)

[35. VIEWS 29](#_Toc105848092)

[DESAFIOS 30](#_Toc105848093)

[MUDAMOS DE TABELA E COMEÇAMOS A TRABALHAR COM A NORTHWIND 39](#_Toc105848094)

# SQL

### Criar um banco de dados:

New Query- create database joao; (carregar f5)

### Apagar um banco de dados:

drop database joao; (carregar f5)

1. SELECT (o que queremos, ex:) SELECT firsName,lastName

FROM (a tabela que contem o que queremos buscar, ex:) FROM person.person;

1. DISTINCT-> é usado quando queremos omitir os dados duplicados de uma tabela.

ex: Se quisermos só o primeiro nome dos nossos contactos, sem repetição de contactos

SELECT DISTINCT FirstName

FROM person.person

### Sinalização

-- ou /\* ou != -> serve para COMENTAR as coisas no SQL

= -> igual

< -> menor que

> -> maior que

<= -> menor ou igual que

>= -> maior ou igual que

<> -> diferente de

and -> e

or -> ou

as -> dar nome às tabelas

1. WHERE-> Mostra especificamente o que queremos procurar, ex:

SELECT \* (\* -> significa TUDO)

FROM person.person

WHERE LastName = 'miller' and FirstName = 'anna' -------> tem que ser com '' e não com ""

OU

SELECT \*

FROM production.product

WHERE ListPrice > 1500 and ListPrice < 5000 ISTO MOSTRA NOS TODOS OS PRODUTOS QUE CUSTAM ENTRE 1500-5000

SE QUISERMOS TODOS OS PRODUTOS MENOS OS VERMELHOS:

SELECT \*

FROM production.Product

WHERE color <> "red"

1. COUNT -> Serve para nos dar o numero a que corresponde cada criterio

ex: QUANTAS MARIAS EXISTEM NA NOSSA BASE DE DADOS DE CLIENTES:

SELECT count(\*)

From person.Person

where FirstName='Marie'

1. TOP -> limita o select (metemos a quantidade que queremos)

ex: Retornar só as primeiras 10 linhas da tabela production.product.

SELECT TOP 10 \*

FROM production.product

1. AS -> Para dar ou mudar o nome da nossa coluna.
2. order by -> ordenar os resultados de uma coluna por ordem crescente ou decrescente (asc/desc)

ex: ordenar o primeiro nome das pessoas por ordem crescente e o ultimo por ordem decrescente.

SELECT FirstName, LastName

FROM person.person

ORDER BY FirstName asc ,LastName desc

1. BETWEEN -> É usado para encontrar um valor ENTRE um valor min e um valor max. É o mesmo que: X >= y <=Z

EX: todos os empregados que foram contratados no ano 2009

SELECT \*

FROM HumanResources.Employee

where HireDate BETWEEN '2009/01/01' and '2010/01/01'

order by HireDate

EX: SE EU QUISER O OPOSTO, (OS QUE FORAM CONTRATADOS NOS OUTROS ANOS TODOS:)

SELECT \*

FROM HumanResources.Employee

where HireDate NOT BETWEEN '2009/01/01' and '2010/01/01'

order by HireDate

1. IN -> Usado juntamente com o WHERE, este operador (in) serve para verificar se um valor corresponde a qualquer valor passado na lista de valores.

EX: Quero encontrar todos as pessoas que têm o BusinesssEntityID de 2 7 e 13

SELECT \*

FROM person.person

WHERE BusinessEntityID IN (2,7,13)

EX: Quero encontrar todos as pessoas que não têm o BusinesssEntityID de 2 7 e 13

SELECT \*

FROM person.person

WHERE BusinessEntityID NOT IN (2,7,13)

1. LIKE -> para encontrarmos algo sem saber o nome em concreto.

EX: nós queremos encontrar uma pessoa mas só sabemos que começa em "ma"

entao:

SELECT \*

FROM person.person

WHERE FirstName like 'Ma%'

EX: nós queremos encontrar uma pessoa mas só sabemos que acaba em "to"

SELECT \*

FROM person.person

WHERE FirstName like '%To'

EX: nós queremos encontrar uma pessoa mas só sabemos que no meio do nome tem "essa"

SELECT \*

FROM person.person

WHERE FirstName like '%essa%'

EX: SE SOUBERMOS QUE FALTA SO UMA LETRA/CARACTER DEPOIS DAS LETRAS "ro"

SELECT \*

FROM person.person

WHERE FirstName like '%ro\_'

1. FUNÇÕES: MIN MAX SUM AVG -> As funções de agregação basicamente agregam ou combinam dados de uma tabela num resultado só.

Para dar um nome à nossa coluna, fazemos no fim: AS "nome da coluna"

Ex: Informações sobre vendas, min, max sum, avg:

MIN(minimo): VER O VALOR MINIMO

SELECT TOP 10 MIN(LINEtotal) AS "Min"

FROM Sales.SalesOrderDetail

MAX(maximo): VER O VALOR MAXIMO

SELECT TOP 10 MAX(LINEtotal) AS "TOTAL"

FROM Sales.SalesOrderDetail

SUM(soma): VER A SOMA DOS VALORES

SELECT TOP 10 SUM(LINEtotal) AS "SOMA"

FROM Sales.SalesOrderDetail

AVG(media): VER A MEDIA DOS VALORES

SELECT TOP 10 AVG(LINEtotal) AS "AVG(media)"

FROM Sales.SalesOrderDetail

1. GROUP BY -> Basicamente divide o resultado da nossa pesquisa em grupos

-para cada grupo pode aplicar uma função de agregação, por exemplo:

-Calcular a soma de itens

-Contar o numero de itens naquele grupo

ex: Ordenar todas as compras que cada ID FEZ

SELECT \*

FROM Sales.SalesOrderDetail

SELECT SpecialOfferID, SUM(UnitPrice) AS "SOMA DE CADA ID"

FROM Sales.SalesOrderDetail

GROUP BY SpecialOfferID

OU

ex: Queremos saber quantos de cada produto foram vendidos ate hoje

SELECT \*

FROM Sales.SalesOrderDetail

SELECT ProductID, COUNT(ProductID) AS "CONTAGEM"

FROM Sales.SalesOrderDetail

GROUP BY ProductID

1. HAVING -> Basicamente é usado em junção com GROUP BY para filtrar os resultados de um agrupamento. De uma forma mais simples, eu gosto de olhar para ele como um WHERE para dados agrupados.

HAVING usa-se depois dos dados agrupados

WHERE usa-se antes dos dados agrupados

ex: QUEREMOS SABER QUAIS OS NOMES NO SISTEMA QUE TEM UMA OCORRENCIA MAIOR QUE 10 VEZES

SELECT FirstName, COUNT(FirstName)

FROM person.person

GROUP BY FirstName

HAVING COUNT(firstname) > 10

1. INNER JOIN-> Junta informções de outras tabelas, com uma COLUNA em COMUM

EX: NÓS QUEREMOS O BUSINESSENTITYID, FIRSTNAME, LASTNAME,EMAILADRESS

SELECT \*

FROM Person.Person

-AQUI TEMOS UMA PARTE DAS COISAS QUE QUEREMOS: BusinessEntityID,FirstName, LastName

SELECT \*

FROM Person.EmailAddress

-AQUI TEMOS: O EmailAddress, que tambem queremos!

AGORA, PRECISAMOS DESSAS DUAS TABELAS E PARA FAZER A LIGAÇÃO, PRECISAMOS DE SABER A COLUNA EM COMUM

QUE É: BusinessEntityID e liga-la com o INNER JOIN Person.emailadress(Coluna que queremos para saber o email) as PEA ON, e dizer que: pp.BusinessEntityID = pea.BusinessEntityID

Entao fazemos:

SELECT pp.BusinessEntityID, pp.FirstName, pp.LastName, pea.EmailAddress

FROM Person.Person as PP

INNER JOIN Person.EmailAddress as PEA ON pp.BusinessEntityID = pea.BusinessEntityID

PARA SABER:

pp -> Apelido que dei, vem de Person.Person, para sabermos de que tabela fomos buscar aquela coluna

pea -> Apelido que dei, vem de Person.EmailAddress, para sabermos de que tabela fomos buscar aquela coluna

### TIPOS DE JOINS:

-Inner join-> Tem uma ou mais colunas em comum e é apartir daí que ligamos tudo

ex: table 1 contém nome, idade e nif | table 2 email, Estado\_Civil e nif

SELECT t1.name, t1.age, t2.email, t2.estado\_civil, t2.nif

FROM table1 as t1

INNER JOIN table2 as t2 ON t1.nif = t2.nif

----> Vamos ter o resultado desses dados QUANDO o NIF for o mesmo:

TABELA1 TABELA2

NAME IDADE NIF Email Estado\_civil NIF

jg 21 12345 picas@gmail.com Casado 12345

lois 14 54321 anao@gmail.com Casado 54321

caldas 8 NULL burrodroga@gmail.com Solteiro NULL <----

-Full outer join-> retorna todos os registos correspondestes na tabela A e B quando são iguais. Alem disso, se nao houver valores correspondentes, ele vai preencher os dados com "null"

ex: table 1 contém id, nome | table 2 id, name

SELECT \*

FROM table1 as t1

FULL OUTER JOIN table2 as t2 on t1.nome = t2.name

----> Vamos ter TODOS os resultado desses dados E quando nao existir nenhum dado na tabela, o valor aparece NULL:

t1: t2 ID NAME ID NAME

id nome id nome 1 cc 2 cc

1 cc 1 esp 2 hell NULL NULL

2 hell 2 cc 3 bea 4 bea

3 bea 3 mario 4 ola NULL NULL

4 ola 4 bea NULL NULL 1 Esp

NULL NULL 3 mario <----

LEFT OUTER JOIN-> Retorna um conjunto de todos os registos da tabela1 e alem disso, os registos correspondentes(quando disponiveis) na tabela2. Se não houver registos correspondentes, ele preenche com "NULL":

t1: t2 ID NAME ID NAME

id nome id nome 1 cc 2 cc

1 cc 1 esp 2 hell NULL NULL

2 hell 2 cc 3 bea 4 be a

3 bea 3 mario 4 ola NULL NULL

4 ola 4 bea

RIGHT OUTER JOIN-> Retorna um conjunto de todos os registos da tabela2 e alem disso, os registos correspondentes(quando disponiveis) na tabela1. Se não houver registos correspondentes, ele preenche com "NULL":

t1: t2 ID NAME ID NAME

id nome id nome 1 esp 2 hell

1 cc 1 esp 2 cc NULL NULL

2 hell 2 cc 3 mario 4 ola

3 bea 3 mario 4 bea NULL NULL

4 ola 4 bea

1. UNION -> Ajuda a combinar dois ou mais resultados de um select apenas num resultado sem resultados duplicados.

Se quisermos Dados duplicados, temos que fazer UNION ALL

EX: Queremos saber todas as pessoas que tenham o titulo de "Mr." e nome do meio "a"

SELECT FirstName,Title,MiddleName

FROM person.person

WHERE title = 'Mr.'

UNION

SELECT FirstName,Title,MiddleName

FROM person.person

WHERE MiddleName = 'a'

1. DATEPART-> Trabalhar com datas, por exemplo extrair mes de X coisa:

ex: o mes que foram feitas as compras

SELECT salesorderid, DATEPART(month,OrderDate) as "month"

from Sales.SalesOrderHeader

ex: Qual foi a media de valor ganho por mes?

SELECT AVG(TotalDue) as "media", DATEPART(month,OrderDate) as "month"

FROM Sales.SalesOrderHeader

GROUP BY DATEPART(month, orderdate)

order by month

### MANIPULAÇÃO DE STRINGS

NOME DE EXEMPLO: ClaudioCunha

CONCAT-> concatena as strings, ou seja, JUNTA TUDO. ex: SELECT CONCAT (FirstName, Lastname)

vai juntar tudo tipo: ClaudioCunha

se eu quiser separar, tambem posso fazer. ex: SELECT CONCAT (FirstName, ' ',LastName)

LEN -> Mostra o tamanho da nossa string(caracteres) ex: SELECT LEN(FirstName)

7

UPPER-> transforma os nossos resultados todos em maiusculo. ex: SELECT UPPER(LastName)

CUNHA

LOWER-> transforma os nossos resultados todos em minisculo. ex: SELECT LOWER(FirstName)

claudio

SUBSTRING-> extrai um bocado de uma string. ex:

imaginemos que temos esta lista de nomes:

-bea

-joana

-ana

-kim

-cat

-gui

-claudio

Se quisermos só os nomes com 3 letras, usamos uma substring. ex: SELECT SUBSTRING(FirstName,1,3)

bea

joa

ana

kim

cat

gui

cla

PS: 1,3 -> significa que estámos a pedir letras do FirstName que vá de 1(primeira letra)-3(terceira letra)

REPLACE-> Substitui o que queremos pelo que nós quisermos. ex: SELECT REPLACE(ProducNumber, '-', '#')

FROM Production.Product

ORIGINAL: AR-5381

MODIFICADA: AR#5381

ou seja, trocamos o " - " pelo " # "

### FUNÇÕES MATEMATICAS

funções ja vistas:

MIN- valor minimo

MAX- valor maximo

SUM- soma dos valores

AVG- media dos valores

Funções novas:

ROUND-> Arredonda os valores

ex: Estamos a pedir para arredondar os numeros a partir do 2 numero(dps da virgula)

SELECT ROUND(LineTotal,2), Linetotal

2024.990000 | 2024.994000

6074.980000 | 6074.982000

2024.990000 | 2024.994000

2039.990000 | 2039.994000

4079.990000 | 4079.988000

SQRT-> Raíz Quadrada dos valores

SELECT SQRT(Linetotal)

From Sales.SalesOrderDetail

1. SUBQUERY (SUBSELECT) -> é um Select Dentro de outro select

ex: Monte um relatório para mim de todos os produtos registados que contenham um preço de venda acima da média

Nós podiamos fazer isto:

SELECT AVG(ListPrice)

FROM Production.Product

SELECT \*

FROM Production.Product

WHERE ListPrice > 438.66

MAS SE HOUVER MUDANÇA NOS VALORES, ISTO VAI SER INVÁLIDO. ENTÃO TEMOS QUE FAZER UM SUBQUERY:

SELECT \*

FROM PRODUCTION.PRODUCT

WHERE ListPrice > (SELECT AVG(ListPrice) FROM Production.Product)

ex: --QUERO SABER O PRIMEIRO E ULTIMO NOME DOS FUNCIONÁRIOS QUE TÊM COMO CARGO "Design Engineer"

--COM UM JOIN

SELECT pp.FirstName,pp.LastName, he.jobtitle

FROM Person.Person as pp

left JOIN HumanResources.Employee as he on pp.businessEntityID = he.BusinessEntityID

WHERE JobTitle = 'Design Engineer'

--COM UM SUBQUERY

SELECT FirstName,LastName

FROM Person.Person

WHERE BusinessEntityID IN (SELECT BusinessEntityID FROM HumanResources.Employee

Where JobTitle = 'Design Engineer')

### MUDAMOS DE TABELA E COMEÇAMOS A TRABALHAR COM A NORTHWIND

1. SELF JOIN-> Agrupar ou ordenar dados dentro da mesma tabela, usamos o self join quando queremos filtrar informação a usar a mesma tabela

temos que usar o " AS " para usar o SELF JOIN

ex: Queremos todos os clientes que moram na mesma região

SELECT A.contactName, A.Region, B.ContactName, B.Region

FROM Customers A, Customers B

Where A.region = B.Region

ex: Os funcionários que foram contratados no mesmo ano

SELECT a.FirstName, a.HireDate, b.FirstName, b.HireDate

FROM dbo.Employees as a, Employees as b

WHERE DATEPART(YEAR, a.Hiredate) = DATEPART(YEAR, b.HireDate)

### TIPOS DE DADOS:

1. Boleanos -> Por padrão ele é inicializado com nulo, e pode receber tanto 1 ou 0

2. Caracteres: .tamanho fixo -> Char: Permite inserir até uma quantidade fixa de caracteres e ocupa sempre todo o espaço reservado 10/50

.tamanhos variaveis -> Varchar ou Nvarchar: Permite inserir até uma quantidade que for definida, porem só usa o espaço que for preenchido 10/50

3. Numeros:

Valores Exatos:

-TinyINT -> não tem parte valor fracionados(ex: 1.43, 24,23)

-SmallINT -> mesma coisa porem limite maior

-INT -> mesma coisa porem limite maior

-BigINT -> mesma coisa porem limite maior

-Numeric ou Decimal -> Valores exatos, porem permite ter parte fracionados, que também pode ser especificado a precisão e escala (escala é o numero de digitos na parte fracional) ex: Numeric (5,2) 113,44

Valores Aproximados:

-Real -> Tem precisão aproximado de ate 15 digitos

-Float -> Mesmo conceito de REAL

4. Temporais:

-Date -> Armazena data no formato aaaa/mm/dd

-DateTime -> Armazena data e horas no formato aaaa/mm/dd:hh:mm:ss

-DateTime2 -> Data e horas com adição de milissegundos no formato aaaa/mm/dd:hh:mm:ms(milisegundos)

-SmallDateTime -> Data e hora respeitando o limite entre '1900-01-01:00:00:00' até '2079-06-06:23:59:59'

-Time -> Horas, minutos, segundos e milissegundos respeitando o limite de '00:00:00.000000' to '23:59:59.9999999'

-DateTimeOffset -> Permite armazenar informações de data e horas incluindo o fuso horário

### Chave Primária e Estrangeira

Primary Key and Foreign Key

Primary Key / Chave Primária -> É basicamente uma coluna ou grupo de colunas, usada para identificar unicamente uma linha numa tabela

Consegues criar essas chaves primárias através de restrições (CONSTRAINTS em ingles), que são regras que defines quando estás a criar uma coluna

Assim, quando fazes isso, estás a criar um índice único para aquela coluna ou grupo de colunas

Para identificar a PRIMARY KEY numa tabela sql, tem um simbolo de uma chave virada para a esquerda com os dentes para baixo e diz (PK, ....) no fim

EX de como criar uma PrimaryKey:

CREATE TABLE nome\_databela (

nomeColuna TipoDeDados PRIMARY KEY

nomeColuna TipoDeDados ...

)

Foreign Key / Chave Estrangeira -> É uma coluna ou grupo de colunas numa tabela que identifica unicamente uma linha noutra tabela.

ou seja, uma chave estrangeira é definida numa tabela onde é apenas uma referencia e não contem todos os dados ali

Então, uma chave estrangeira é simplesmente uma coluna ou grupo de colunas que é uma chave primária noutra tabela

A tabela que contem a chave estrangeira é chamada de tabela referenciadora ou tabela filho. e a tabela na qual a chave estrangeira é referenciada é chamada de tabela referenciada ou tabela pai.

Uma tabela pode ter mais do que uma chave estrangeira dependendo do seu relacionamento com as outras tabelas

Para identificar a FOREIGN KEY numa tabela sql, tem um simbolo de uma chave virada para a direita com os dentes para cima e diz (FK, ....) no fim

Regras basicas: No SQL Server defines uma chave estrangeira através de um "Foreign Key Constraint" ou restrição de chave estrangeira. Uma Restrição de Chave Estrangeira (Foreign Key) indica que os valores numa coluna ou grupo de colunas na tabela filho, correspondem aos valores na tabela pai. Nos podemos entender que uma chave estrangeira mantem a "integridade referencial"

ex: de onde seria usado uma Foreign Key

tabela produtos:

Chave|ProductID

|ProductName

|SupplierID Tabela Categories:

|CategoryID -------------------- Chave|CategoryID

|QuantityPerUnit |CategoryName

|UnitPrice |Description

### CREATE TABLE:

ex: como criar uma tabela

Create TABLE nometabela (

coluna1 tipo restriçãoDaColuna,

coluna2 tipo,

coluna3 tipo,

...

);

Restrições que podem ser aplicadas no SQL:

-NOT NULL -> Não permite nulos

-UNIQUE -> Força que todos os valores numa coluna sejam diferentes

-PRIMARY KEY -> uma junção de NOT NULL e UNIQUE

-FOREIGN KEY -> identifica únicamente uma linha noutra tabela

-CHECK -> Força uma condição específica numa coluna

-DEFAULT -> Força um valor padrão quando nenhum valor é passado

EX: CRIRAR ESTAS 2 TABLEAS:

Uma imagem com texto

Descrição gerada automaticamente

create table CANAL(

CanalId int primary Key

Nome varchar(150)

ContagemInscritos int default 0

DataCriacao datetime not null

);

create table VIDEO(

VideoId int primary Key

Nome varchar(150) not null

Visualizações int default 0

likes int default 0

dislikes int default 0

duracao int not null

canalId Foreign Key references canal(CanalId)

);

1. INSERT-> Inserir dados nas nossas tabelas

create table CANAL(

CanalId int primary Key,

Nome varchar(150)

);

INSERT INTO nome\_Da\_tabela(nomeDoCampo, NomeDoCampo)

VALUES(valordocampo,'algo escrito se quisermos')

EX:

INSERT INTO canal(CanalId,Nome)

VALUES(123456789,'RicFazeres')

SE QUISERMOS PREENCHER UMA TABELA COM MUITAS LINHAS:

INSERT INTO canal(CanalId,Nome)

VALUES

(123456789,'RicFazeres'),

(987654321,'RicFazeres'),

(987651234,'RicFazeres'),

(123459875,'RicFazeres'),

(198273645,'RicFazeres');

TAMBEM PODEMOS COPIAR OS DADOS DE UMA TABELA PARA OUTRA TABELA:

ex: Vamos passar os dados da tabela canal, para uma com o nome de "COPIACANAL"

select \* into CopiaCanal FROM CANAL

1. UPDATE-> serve para atualizar linhas do nosso banco de dados

TEMOS QUE UTILIZAR O "WHERE" QUANDO QUEREMOS ALTERAR SOMENTE UMA DETERMINADA LINHA, SE NAO FIZERMOS ISTO, VAMOS ALTERAR TODAS AS LINHAS DO NOSSO BANCO DE DADOS.

EX:

canal

ID NOME

(123456789,'RicFazeres'),

(987654321,'RicFazeres'),

(987651234,'RicFazeres'),

(123459875,'RicFazeres'),

(198273645,'RicFazeres');

se nós fizermos:

UPDATE canal

SET NOME = 'TiagoPaiva'

isto fica:

canal

ID NOME

(123456789,'TiagoPaiva'),

(987654321,'TiagoPaiva'),

(987651234,'TiagoPaiva'),

(123459875,'TiagoPaiva'),

(198273645,'TiagoPaiva');

se nós fizermos:

UPDATE canal

SET NOME = 'Antonio'

WHERE ID = 123459876

isto fica:

canal

ID NOME

(123456789,'TiagoPaiva'),

(987654321,'TiagoPaiva'),

(987651234,'TiagoPaiva'),

(123459875,'Antonio'),

(198273645,'TiagoPaiva');

1. DELETE-> serve para apagar

TEMOS QUE UTILIZAR O "WHERE" QUANDO QUEREMOS APAGAR SOMENTE UMA DETERMINADA LINHA, SE NAO FIZERMOS ISTO, VAMOS APAGAR TODAS AS LINHAS DO NOSSO BANCO DE DADOS.

DELETE FROM canal

where nome = 'Antonio'

isto fica:

canal

ID NOME

(123456789,'TiagoPaiva'),

(987654321,'TiagoPaiva'),

(987651234,'TiagoPaiva'),

(198273645,'TiagoPaiva');

1. alter table-> Alterar a estrutura de uma tabela.

Nós podemos fazer varias coisas com o alter table:

- Add, remover, alterar uma coluna

- set valores padrões para uma coluna

- Add ou remover restrições de colunas

- Renomear uma tablea

ex: add uma coluna à nossa tabela canal

Alter table canal

add ativo bit bit-> é do tipo boleano (true or false), vai servir para saber se o canal está ativo ou não

AGORA a nossa tabela canal tem estas colunas:

ID NOME ATIVO

ex: alterar uma restrição de uma coluna. O nome tem varchar(150), vamos alterar para 300 caracteres

ALTER TABLE canal

ALTER COLUMN nome varchar(300) not null

ex: mudar o nome de uma coluna

exec sp\_rename 'canal.id' , 'NIF' , 'COLUMN'

ex: se quisermos mudar o nome da tabela

exec sp\_rename 'canal' , 'Conta'

1. drop table -> serve para apagarmos uma tabela

atenção que só conseguimos apagar tabelas que não são referenciadas por outras tabelas, ou seja, que nao tenha, ex: foreign key references canal[canalid]

é so fazer:

drop table CANAL

Como esvaziar uma tabela sem eliminar a tabela:

temos que usar o TRUNCATE TABLE

ex:

truncate table canal

continuamos com o ID e NOME, mas tudo sem estar preenchido.

1. CHECK CONSTRAINT-> cria restrições sobre valores. Podemos fazer isto quando estivermos a inserir o valor ou a alterar o valor, ele vai impor um tipo de restrição.

ex: Tirar a carta de condução

CREATE TABLE CartaDeConducao(

id int not null,

nome varchar(255) not null,

Idade int CHECK (Idade >= 18)

);

insert into CartaDeConducao(id,nome,idade)

Values

(1,'joao',18),

(1,'pedro',19),

(1,'afonso',20);

SE O NUMERO DA IDADE FOR POR EX 17, VAI DAR-NOS UM ERRO

1. NOT NULL CONSTRAINT-> Obriga a que uma coluna seja preenchida. Fazemos isso com o NOT NULL

ex:

CREATE TABLE CartaDeConducao(

id int not null,

nome varchar(255) not null,

Idade int CHECK (Idade >= 18)

);

Aqui somos obrigados a preencher o id e nome

1. UNIQUE CONSTRAINT-> obriga a que um campo seja UNICO, ou seja, que não haja outro repetido

ex:

create table unico(

id int not null unique,

nome varchar(255) not null,

idade int

);

1. VIEWS-> tabela criada para consultar dados de outras tabelas, baseado num select com alguns dados que queremos

TEMOS QUE USAR O AS, E CRIAR UM SELECT COM ALGUNS DADOS, DEPOIS DO AS.

ex:

CREATE VIEW[pessoas simplificado] AS

SELECT FirstName,MiddleName,LastName

FROM Person.Person

WHERE Title = 'Ms.'

### DESAFIOS

1- A EQUIPA DE MARKETING PRECISA DE FAZER UMA PESQUISA SOBRE OS NOMES MAIS COMUNS DOS SEUS CLIENTES.

PRECISA DO PRIMEIRO E ULTIMO NOME DE TODOS OS CLIENTES QUE ESTÃO REGISTADOS NO SISTEMA.

SELECT FirstName, LastName

FROM person.person;

2- QUANTOS ULTIMOS NOMES UNICOS TEMOS NA NOSSA TABELA PERSON.PERSON?

SELECT DISTINCT lastName

FROM person.person;

3- A EQUIPA DE PRODUÇÃO DE PRODUTOS PRECISA DO NOME DE TODAS AS PEÇAS QUE PESAM MAISS DO QUE 500KG MAS NÃO MAIS DO QUE 700KG PARA INSPEÇÃO

SELECT Name

FROM Production.Product

WHERE Weight > 500 and Weight < 700

4- FOI PEDIDO PELO MARKETING UMA RELAÇÃO DE TODDOS OS EMPREGADOS (EMPLOYEES) QUE SÃO CASADOS (MARRIED) E SAO ASSALARIADOS (SALARIED)

SELECT \*

FROM HumanResources.Employee

WHERE MaritalStatus='m' and SalariedFlag=1

5- UM USUARIO CHAMADO PETER KREBS ESTÁ A DEVER UM PAGAMENTO, CONSIGA O EMAIL DELE PARA QUE POSSAMOS ENVIAR UMA COBRANÇA!

(VAI TER QUE USAR A TABELA PERSON.PERSON E DEPOIS A TABELA PERSON.EMAILADRESS)

SELECT \*

FROM person.person

Where FirstName='Peter' and LastName='Krebs'

(vou buscar o ID " BUSINESSENTITYID=26)

SELECT EmailAddress

FROM person.EmailAddress

WHERE businessEntityID=26

OU

select e.EmailAddress

from person.person p

left outer join person.EmailAddress e on (e.BusinessEntityID=p.BusinessEntityID)

where p.FirstName='peter' and p.lastname='krebs'

6- QUERO SABER QUANTOS PRODUTOS TEMOS REGISTADOS NA NOSSA TABELA DE PRODUTOS (production.product)

select count(\*)

from Production.Product

7- QUERO SABER QUANTOS TAMANHOS DE PRODUTOS TEMOS REGISTADOS NA NOSSA TABELA (production.product)

select count(size)

from Production.Product

8- QUERO SABER QUANTOS TAMANHOS DIFERENTES DE PRODUTOS TEMOS REGISTADOS NA NOSSA TABELA (production.product)

select count(DISTINCT size)

from Production.Product

9- OBTER O PRODUCTID DOS 10 PRODUTOS MAIS CAROS REGISTADOS NO SISTEMA, LISTA DO MAIS CARO PARA O MAIS BARATO

select top 10 ProductID

from Production.Product

order by ListPrice desc

10- OBTER O NOME E NUMERO DO PRODUTO DOS PRODUTOS QUE TEM O PRODUCTID ENTRE 1-4

SELECT TOP 4 Name, ProductNumber

FROM Production.Product

order by ProductID asc

11- QUANTOS PRODUTOS TEMOS REGISTADOS NO SISTEMA QUE CUSTAM MAIS QUE 1500 DOL

SELECT count(\*)

FROM Production.Product

Where ListPrice > 1500

12- QUANTAS PESSOAS TEMOS COM ULTIMO NOME QUE COMEÇA COM A LETRA P ?

SELECT count(LastName) SELECT count(\*)

FROM person.Person OU FROM person.Person

WHERE LastName like 'P%' WHERE LastName like 'P%'

13- EM QUANTAS CIDADES UNICAS ESTÃO REGISTADOS OS NOSSOS CLIENTES?

SELECT COUNT (DISTINCT City)

FROM Person.Address

14- QUAIS SÃO AS CIDADES UNICAS QUE TEMOS REGISTADAS NO NOSSO SISTEMA?

SELECT DISTINCT City

FROM Person.Address

15- QUANTOS PRODUTOS VERMELHOS TEM O PREÇO ENTRE 500 A 1000 DOL

SELECT COUNT (\*)

FROM Production.Product

WHERE color = 'red' and ListPrice BETWEEN 500 and 1000

16- QUANTOS PRODUTOS REGISTADOS TEM A PALAVRA "ROAD" NO NOME DELES?

SELECT COUNT(\*)

FROM production.product

Where name like '%road%'

17- QUANTAS PESSOAS TÊM O MESMO MIDDLENAME AGRUPADAS POR O MIDDLENAME

SELECT MiddleName, COUNT(FirstName)

FROM Person.Person

GROUP BY MiddleName

18- QUEREMOS SABER EM MEDIA QUAL É A QUANTIDADE QUE CADA PRODUTO É VENDIDO NA LOJA

SELECT salesOrderID, AVG(OrderQty)

From Sales.SalesOrderDetail

GROUP BY SalesOrderID

19- Quais foram as 10 vendas que no total tiveram os maiores valores de venda(line total) por produto do maior para o menor

SELECT TOP 10 productid,SUM(LineTotal)

From Sales.SalesOrderDetail

GROUP BY ProductID

ORDER BY SUM(linetotal) DESC

20- QUANTOS PRODUTOS E QUAL A QUANTIDADE MEDIA DE PRODUTOS TEMOS CADASTRADOS NAS NOSSAS ORDEM DE SERVIÇOS, AGRUPADOS POR PORDUCTID

SELECT productid, COUNT(ProductID) as PRODUTOS, AVG(OrderQTY) as "MEDIA"

FROM Production.WorkOrder

GROUP BY ProductID

21- QUEREMOS IDENTIFICAR AS PROVINCIAS(STATEPROVINCEID) COM O MAIOR NUMERO DE REGISTOS NO NOSSO SISTEMA, ENTÃO É PRECISO ENCONTRAR QUAIS PROVÍNCIAS(STATEPROVINCEID) ESTÃO REGISTADAS NO BANCO DE DADOS MAIS DO QUE 1000 VEZES

SELECT stateprovinceid, COUNT(stateprovinceid)

FROM Person.Address

GROUP BY StateProvinceID

HAVING COUNT(stateprovinceid) > 1000

22- SENDO QUE SE TRATA DE UMA MULTINACIONAL, OS GERENTES QUEREM SABER QUAIS OS PRODUTOS (PRODUCTID) NÃO ESTÃO A TRAZER EM MÉDIA NO MINIMO 1 MILHÃO EM TOTAL DE VENDAS (LINETOTAL)

select productid, avg(linetotal)

FROM sales.SalesOrderDetail

group by ProductID

HAVING avg(linetotal) < 1000000

23- NESTAS DUAS TABELAS: SELECT TOP 10\* SELECT top 10\*

FROM Person.PhoneNumberType FROM Person.personPhone

Queremos usando o join: BusinessEntityId, Name, PhoneNumbertypeId, PhoneNumber

SELECT TOP 10 \*

FROM Person.PhoneNumberType

SELECT TOP 10 \*

FROM Person.PersonPhone

SELECT ppp.BusinessEntityID,ppnt.PhoneNumberTypeId, ppp.PhoneNumber, ppnt.Name

FROM Person.PersonPhone as ppp

INNER JOIN Person.PhoneNumberType as ppnt ON ppp.PhoneNumberTypeID = ppnt.PhoneNumberTypeID

24- NESTAS DUAS TABELAS: SELECT TOP 10 \* SELECT TOP 10 \*

FROM person.stateprovince FROM person.address

QUEREMOS: AddressId, City, StateProvinceId, Nome do estado

SELECT pa.AddressID, pa.city, psp.StateProvinceID, psp.Name

FROM Person.StateProvince as psp

INNER JOIN person.Address as pa ON psp.StateProvinceID = pa.stateprovinceID

25- Encontrar qualquer tabela no banco e juntar duas colunas usando UNION com um filtro WHERE

SELECT BusinessEntityID, BirthDate, Gender

FROM HumanResources.Employee

WHERE JobTitle like '%chief%'

UNION

SELECT BusinessEntityID,BirthDate, Gender

FROM HumanResources.Employee

WHERE JobTitle like '%President%'

26- Encontrar qualquer outra tabela que contenha uma data e extrair mes e ano dessa coluna usando o DATEPART

SELECT NAME, ProductNumber, DATEPART(MONTH, SellStartDate) as "Mês de fabrico"

FROM Production.Product

SELECT NAME, ProductNumber, DATEPART(YEAR, SellStartDate) as "Ano de fabrico"

FROM Production.Product

27- Encontrar uma tabela e executar todos os comandos que aprendemos na manipulação de strings:

CONCAT

SELECT CONCAT(BusinessentityID, JobTitle)

FROM HumanResources.Employee

LEN

SELECT LEN(JobTitle)

FROM HumanResources.Employee

LOWER

SELECT LOWER(JobTitle)

FROM HumanResources.Employee

UPPER

SELECT UPPER(JobTitle)

FROM HumanResources.Employee

SUBSTRING

SELECT SUBSTRING(JobTitle,1,6)

FROM HumanResources.Employee

REPLACE

SELECT REPLACE(NationalIDNumber,'4','quatro')

FROM HumanResources.Employee

28- Usar a função ROUND numa outra tabela.

SELECT ROUND(UnitPrice,2)

FROM Sales.SalesOrderDetail

29- Encontre todos os endereços que estão na cidade de "toronto". Pode trazer todas informações.

select \*

from Person.Address as pa

inner join person.stateprovince as ps on ps.StateProvinceID = pa.StateProvinceID

where city = 'toronto'

SELECT \*

FROM person.Address

WHERE StateProvinceID in (SELECT StateProvinceID FROM person.StateProvince

WHERE city = 'toronto')

#### MUDAMOS DE TABELA E COMEÇAMOS A TRABALHAR COM A NORTHWIND

30- Quero saber na tabela detalhe do pedido [Order Details] Quais produtos têm o mesmo percentual de desconto

SELECT um.ProductID, um.Discount, dois.ProductID, dois.Discount

FROM [Order Details] as um, [Order Details] as dois

where um.Discount = dois.Discount

31- encontrar todos os exemplos de foreign key e primary key

32- Encontre duas coisas que podem tornar tabelas e crie 2 tabelas que tem no minimo 1 relacionamento com a outra

create table TEAMS( create table NAVI( create table G2(

Name varchar(150) PrimaryKey, NamePlayers varchar(999) Primary Key, NamePlayers varchar(999) Primary Key,

Players INT NOT NULL, coach int default 0, coach int default 0,

FANS INT NOT NULL, CoachName varchar(999), CoachName varchar(999),

TorneiosImportantes INT NOT NULL); Name varchar(150) not null foreign key REFERENCES Teams(Name) Name varchar(150) not null foreign key REFERENCES Teams(Name)

33- INSERIR DADOS NA NOSSA TABELA E CRIAR UMA TABELA NOVA COM OS DADOS COPIADOS DE OUTRA

insert into Teams(Name,Players,fans,TorneiosImportantesGanhos)

VALUES

('NAVI', 5, 2000000, 18),

('G2',5,167000000,13);

select \*

from Teams

select \* into copiateams FROM Teams

select \*

from copiateams

34-Mude 2 linhas da sua tabela

UPDATE Teams

SET Players = 6

where Name = 'G2'

UPDATE Teams

SET fans = 189000000

where Name = 'G2'

35- 1) criar uma tabela nova com 3 colunas

2) alterar o tipo de uma coluna!

3) Renomear o nome de uma coluna!

4) Renomear o nome da table que criamos

1)

create table teste(

id int primary key,

nome varchar(150) unique not null,

cor\_preferida varchar(150) not null

);

2)

alter table teste

alter column nome varchar(300) not null

3)

exec sp\_rename 'teste.nome' , 'NomeCanal', 'column'

4)

exec sp\_rename 'teste' , 'FEITO'

36- Criar 2 tabelas e apaga-las

create table apagar1(

id int primary key,

nome varchar(150) unique not null,

cor\_preferida varchar(150) not null

);

create table apagar2(

id int primary key,

nome varchar(150) unique not null,

cor\_preferida varchar(150) not null

);

drop table apagar1

drop table apagar2

37-Criar 1 tabela nova e duas restrições para ela

create table apagar2(

nome varchar(150) unique not null,

cor\_preferida varchar(150) not null,

idade int check (idade >= 18),

nacionalidade varchar(150) not null check (nacionalidade = 'Portuguesa')

);

INSERT INTO apagar2(nome,cor\_preferida,idade,nacionalidade)

VALUES ('joao', 'azul', 19, 'Portuguesa')

38- criar 1 tabela nova e nao permita campos vazios nas colunas

CREATE TABLE TudoObrigatorio(

nome varchar(150) not null,

idade int not null,

id int not null

);

39- cria uma tabela nova e defina 1 ou 2 campos unicos.

create table canalyoutube(

nomeDoCanal varchar(150) not null unique,

idade ínt not null,

id int not null unique,

TipoDeVideo varchar(150)

);