TIPOS DE DADOS

INT - USADO PARA NUMEROS INTEIROS, NÃO PRECISA ESCOLHER A QUANTIDADE DE CARACTERES.

VARCHAR – USADO PARA TEXTO, PRECISA ESCOLHER A QUANTIDADE DE CARACTERES, EXEMPLO:

- NOME VARCHAR (200) NOT NULL (not null quer dizer que o campo não pode ser deixado em branco).

NUMERIC – UTILIZADO QUANDO DESEJA COLOCAR NUMERO REPRESENTANDO VALOR DE ALGUMA COISA, POR EXEMPLO, VALOR DE UM PRODUTO EM REAIS, PRECISA COLOCAR A QUANTIDA DE CASA ANTES DA VIRGULA E DEPOIS DA VIRGULA:

-NUMERIC (5,2) NOT NULL.

AULA 1 - Criação de tabelas

Para criar uma tabela em script no SQL, basta digitar a função CREATE TABLE (com o nome que você quer colocar para essa tabela na frente), por exemplo, CREATE TABLE Alunos, conforme a imagem a baixo:



Em seguida, você abre um parêntese na linha de baixo (função padrão para que o sql saiba que aquilo que esta sendo digitado, é uma função que vai ser executada) e depois basta colocar as colunas dessa tabela, entre esses parênteses.

Toda tabela relacionável vai ter uma chave primaria para sua identificação e para que ela possa ser usada para relacionar uma tabela com a outra. Por exemplo, você vai criar uma tabela de alunos e uma tabela de cursos. Na teoria, um aluno pode estar inscrito em vários cursos, então para não escrever o nome desse aluno varias vezes junto com todos os cursos que ele vai fazer, você apenas usa o ID, que será o numero que ele vai receber por causa dessa chave primaria ou primary key e vice versa.

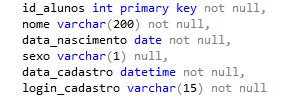
Então a próxima etapa é definir a chave primaria ou primary key, como por exemplo:



Para escrever e definir uma chave primaria, sempre vai ser dessa forma que esta na imagem, o que muda vai ser apenas o nome que você vai dar para ela. Para ficar mais fácil, sempre tente colocar um nome que faça sentido para a tabela, como por exemplo, se a tabela for Alunos, o certo é colocar a chave também como aluno, para não confundir depois.

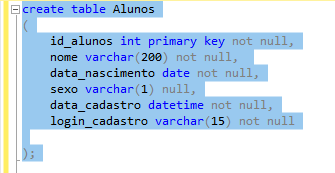
Todas as vezes que for definir a chave, sempre escreva o seguinte código “INT PRIMATY KEY NOT NULL”. NOT NULL, que dizer que esse campo não pode ser em branco, ele tem que ser preenchido, pois quando for buscar o ID do aluno X, vai ser por essa chave primaria que você iria buscar todas as informações do mesmo.

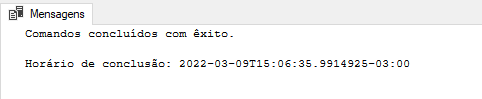
Depois de criar a chave primaria, comece definindo quais serão os campos ou colunas que vão existir na sua tabela, por exemplo:



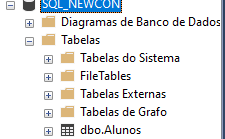
Como pode ver, essa tabela vai ser composta pelos seguintes campos, NOME, DATA DE NASCIMENTO, SEXO, DATA DE CADASTRO, LOGIN DE CADASTRO. Podem reparar também, que todos esses campos definidos estão com algumas palavras na frente, que nada mais é o que define qual o tipo dessas “variáveis”. Por exemplo, o campo nome esta com a palavra VARCHAR (200) na frente, isso quer dizer que a variável NOME, vai ser uma variável escrita com 200 caracteres. E a questão de definir caracteres, é saber se aquele campo vai ser definido para escrita ou para numero, que no caso em vez de ser VARCHAR, vai ser int para números inteiro (exemplo própria chave primaria). A diferença entre as variaves DATE e DATETIME é que a DATE só aparece a data e DATETIME pode ser preenchida com a data e hora.

Depois de definir as colunas existentes na tabela, você terá que fechar essa tabela com um fecha parênteses e selecionar todas as linhas e dar um EXECUTAR ou F5 para criar a tabela. Em seguida aparece a mensagem se deu certo ou não.





Para ter certeza de que a tabela foi realmente feita, é simples, bastar ir nas pastas ao lado esquerdo da tela, no banco de dados que você esta utilizando, apertar nele com o botão direito, ir em atualizar, e depois abrir os subcampos do bando de dados, ir até em tabelas e verificar se a tabelas está lá.



Para excluir essa tabela, basta escrever DROP TABLE e o nome da mesma tabela, no caso DROP TABLE Alunos; e apertar um F5 ou executar.

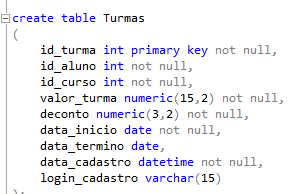
Aula 2 – CHAVES ESTRANGERIAS

Chave estrangeira é nada mais nada menos que do você usar uma chave primaria, ou, primary key, dentro de outra coluna. Por exemplo, em vez de cadastrar todos os alunos novamente na tabela TURMAS, sabendo que eles já estão cadastrados na tabela ALUNOS, eu simplesmente pego a CHAVE PRIMARIA da tabela ALUNOS e cadastro um campo para essa chave dentro da tabela TURMAS. Sendo assim, na hora de eu procurar o aluno pelo nome dentro da tabela TURMA, eu apenas coloca o ID desse aluno que vai estar cadastrado como chave estrangeira para pegar todas as informações dentro da tabela aluno e me mostrar na tabela TURMA.

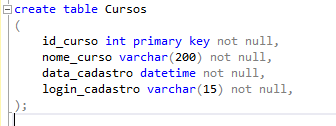
Em outras palavras, a chave estrangeira serve para relacionar uma tabela a outra, aluno com turma, turma com curso, para evitar a repetição de dados, fazendo com que fique com o sistema mais limpo e funcionando.

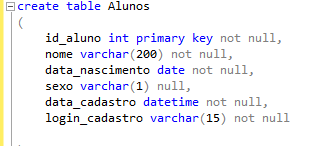
Para criar uma chave estrangeira, você deve criar todas as tabelas que serão utilizadas, nisso, se atentar em quais vão estar cada quais vão ser as chaves secundarias.

Por exemplo:



Como vocês podem ver, na tabela TURMAS existe uma chave primaria “ID\_TURMA”, identificada como primary key (chave primaria), e duas chaves estrangeiras, “ID\_ALUNO” e “ID\_CURSO”. Da para saber, pois existem as tabelas CURSO com a chave primaria “ID\_CURSO” e a tabela “ALUNOS”, com a chave primaria “ID\_ALUNO”.





Então é de extrema importância saber na hora que esta criando uma tabela, quais serão relacionadas, se vão ser relacionas, quais as chaves primarias.

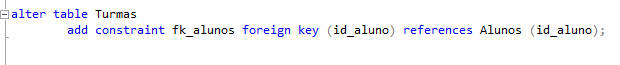
Depois de criar e colocar as chaves estrangeiras dentro de cada tabela, você terá que relacionar essas tabelas definitivamente, por enquanto, elas não estão relacionadas diretamente, então o sistema não sabe o que fazer e qual tabela relacionar apenas com a criação da tabela e suas chaves.

Para relacionar as chaves estrangeiras com a suas chaves primarias, basta alterar a tabela já criada, com o seguinte comando:

“ALTER TABLE (Nome da tabela que contem uma chave estrangeira)” e na linha de baixo escrever a seguinte frase “ADD CONSTRAINT FK\_(NAME) FOREIGN KEY (ID\_NAME) REFERENCES “NAME TABLE” (ID\_NAME)”

O que isso quer dizer? Que você esta pedindo para alterar a tabela X, adicionando uma chave secundaria com o nome fk\_name, que esta relacionada ao id\_name que se referece a tabela name com o id name.

Exemplo:



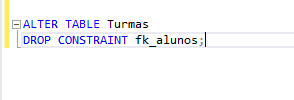
“*Altere a tabela turmas, acrescentando uma constraint chamada fk\_alunos, que será uma chave estrangeira do (id\_aluno) encontrada na tabela turmas, que é a chave primaria da tabela Alunos, identificada como id\_aluno.*

Caso relacione uma chave estrangeira de forma errada, ou com outra tabela, basta fazer a seguinte modificação:

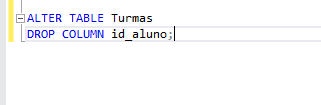


No caso você esta alterando a tabela Turmas, excluindo a chave estrangeira que você criou para relacionar a chave primaria da tabela aluno (ID\_ALUNO) com a chave estrangeira (ID\_ALUNO) na tabela Turmas.

Caso você tenha feito algo de errado e queira apagar a sua tabela depois de ter criado a sua CONTRAINT, deve se lembrar que se você tentar apagar, a tabela, você não vai conseguir com o comando DROP TABLE e o nome da tabela caso ela tenha uma relação com outra. Para isso, você terá que fazer o seguinte passo:



Isso quer dizer que você vai esta querendo apagar a CONSTRAINT fk\_alunos que esta dentro da tabela TURMAS e que esta relacionada a tabela ALUNOS. Isso será necessário, pois a tabela turmas tem colunas relacionadas a tabela alunos, então o programa só deixa apagar a tabela depois de fazer a exclusão da chave estrangeira. Esse passo serve também para que você consiga excluir apenas uma coluna, caso essa coluna for a chave estrangeira dentro da tabela relacionada à outra.



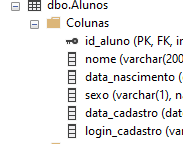
Nesse caso, o comando que esta sendo digitado no script, esta dizendo que a tabela Turmas, vai ser alterada, e que vai ter a exclusão da coluna ID\_ALUNO, que era uma chave estrangeira, mas como houve uma exclusão dessa chave estrangeira antes, então não terá problema em executar essa ação.

A partir disso você conseguira apagar a tabela inteira, pois não terá mais nada relacionada a ela.

LEMBRE-SE, O COMANDO “DROP” SIGNIFICA EXCLUIR.

Aula 3 - INSERT – inserindo dados na tabela

Para inserir os dados na tabela pelo script, existe o comando INSERT INTO. Mas antes de tudo isso, você tem que entrar na sua tabela e ver quais são as colunas criadas para depois começar a fazer a inserção.



O código “INSERT INTO” serve para inserir ou preencher os dados necessários conforme aquela tabela foi criada, seguindo na linha de baixo de um abre e fecha parênteses “()“ escrevendo todos os nomes das colunas que existem nessa tabela e que vão receber os dados a serem inseridos, por exemplo:

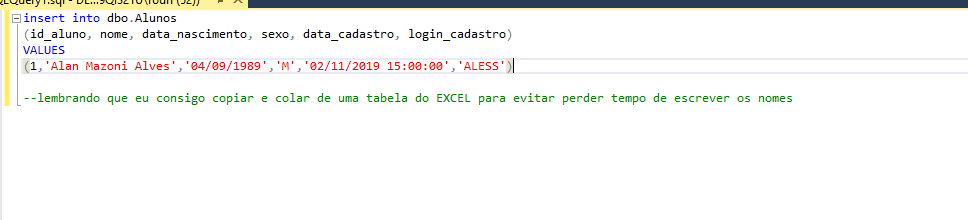
INSERT INTO dbo.Alunos

(id\_aluno, nome, data\_nascimento, sexo, data\_cadastro, login\_cadastro)

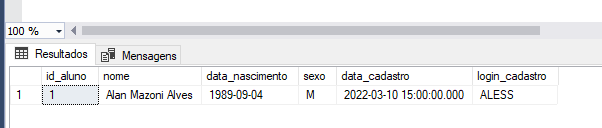
Isso quer dizer que eu quero inserir ou preencher alguns dados dentro da tabela dbo.Alunos, e as colunas que vão ser preenchidas são a id\_aluno, nome do aluno, data de nascimento, sexo... e assim por diante.

Em seguida, depois de mencionar qual será a tabela e quais serão os campos que eu vou preencher, vou acrescentar outro comando na linha de baixo chamado “VALUES”. Logo após eu começo a escrever entre parênteses, conforme foi mencionado na linha onde estão os campos a serem preenchidos, na ordem, lembrando sempre de colocar a numeração de cada primary key, que nesse caso será o ID\_ALUNO, como mostra a imagem a baixo.

Lembrar também de colocar sempre as informações dentro de aspas simples separadas por virgula , para que as informações digitadas possam ser inseridas dentro das colunas em ordem que você colocou tanto quando a tabela foi feita, quanto escrita no script na hora do INSERT INTO.



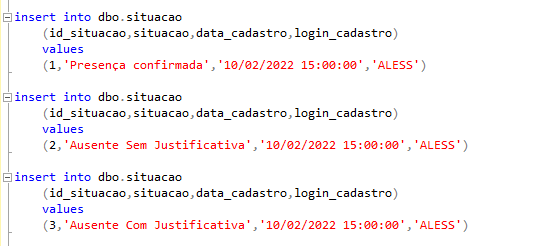
Em seguida, selecione todo o código scrpit e de um F5, e verifique se apareceu a seguinte frase “(1 linha afetada)”, quer dizer que foi preenchido uma linha das colunas. Para tirar a prova se deu certo ou não, faça uma busca com o comando SELECT \* FROM + nome da tabela que esta sendo editada e aperte F5 ou executar, exemplo:



Podem ver que deu certo a inserção dos dados digitados no scritp.

OBS: NÃO A NECESSIDADE DE ESCREVER OS COMANDOS E CODIGOS EM LETRAS MAIUSCULAS, TUDO UMA QUESTAO DE LAYOUT. MAS NUNCA ESCREVER AS COISAS NA MESMA LINHA, POIIS NA HORA DE EXECUTAR, PODE DAR ERRO A MISTURA.

Não tem como ir fazendo varias linhas no script, pois quando for executar o código, ele vai executar todas as linhas e vai dar um erro dizendo que aquelas informações já foram registradas. Então pra cada cadastro de informação pelo INSERT INTO, terá que fazer uma linha nova com o código escrito novamente, pode copiar e colar sem problemas nenhum, mas o que não pode esquecer é de trocar o numero da CHAVE PRIMARIA DE CADA TABELA, exemplo:

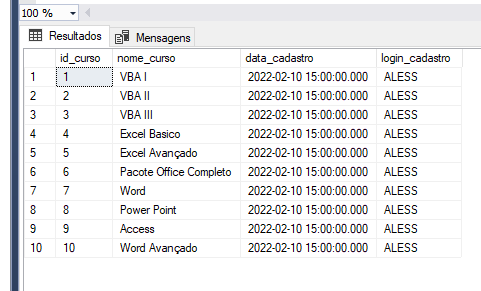


Pode ver que foram feitas três INSERT INTO, e nas três teve alteração do ID\_SITUAÇÃO, que seria a chave primaria da tabela Situação, pois se tentar usar o mesmo numero para outro registro, dará erro. Isso se faz, pois na hora de puxar a situação de cada aluno, se ele compareceu, se ele faltou sem justificativa, se ele deu justificativa pela falta, tudo ira se resumir apenas colocando o numero 1 para compareceu, 2 para falta sem justificativa e 3 com justificativa, em vez de ficar escrevendo uma por uma.

OBS: CASO QUEIRA COLOCAR A DATA ATUAL UTILIZE O COMANDO “GETDATE()” NO SCRIPT.

AULA 4 – SELECT

O comando select, nada mais é que você puxar todas as informações, ou algumas, de uma tabela já criada. No caso seria você tirar a prova limpa para ver se esta tudo certo com a tabela que você acabou de criar, ou para analisar se as informações inseridas estão corretas, ou simplesmente apenas para analisar ou olhar a tabela desejada. Para realizar esse comando, basta digitar “SELECT \* FROM nome da tabela desejada”:



Depois de apertar F5 para executar o comando, note que na parte de baixo da tela, vai aparecer a tabela desejada com todas as informações que existem nela. Caso você escreva dois SELECT no mesmo script, quando apertar para executar, as duas tabelas vão ser exibidas na parte de baixo da tela. Caso você queira apenas que uma dessas tabelas apareça, basta selecionar o comando inteiro desejado e apertar F5 ou o botão EXECUTAR, que assim, aparecera apenas uma.

Bom lembrar que cada tabela tem o seu WALNER antes do seu nome, por exemplo:



Note que no comando a tabela cursos esta escrita com uma DBO. antes, isso facilita saber de onde aquela tabela faz parte, caso você tenha vários outros departamentos. Por exemplo, se você mexe em uma empresa que tem vários departamentos, e todos eles mexem com banco de dados, você pode classificar ou ordenar as suas tabelas mudando essa nomenclatura que vem antes do nome da tabela, por isso, caso tenha vários outras tabelas com o mesmo nome “CURSOS”, o bom é sempre colocar a sua walner antes, para não evitar erros. Então a dica é, quando for criar uma tabela, sempre indique qual é a walner dessa tabela, para depois na hora de dar um SELECT, nao executar a tabela errada.

Existe também outra forma de se selecionar uma tabela, como por exemplo:

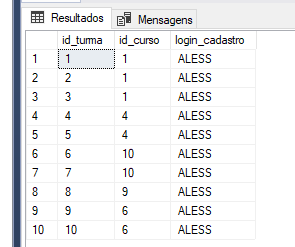


Nesse caso, estamos selecionando a tabela usando o próprio nome dela, nessa caso se explica assim “SELECIONE A TABELA CURSOS E SELECIONE TAMBEM TODOS OS CAMPOS DA TABELA CURSO”. Basicamente é isso que o comando esta dizendo, selecione e me mostre a tabela e tudo o que tem nessa tabela. O ponto \* significa isso por exemplo.

Além de buscar todas as informações que existe na tabela, a função SELECT, também consegue mostrar apenas algumas colunas ou ate uma coluna desejada, por exemplo:



Nesse caso estamos pedindo que o sistema nos mostre apenas a coluna ID\_TURMA, ID\_CURSO e LOGIN\_CADASTRO, que depois de executado, ficara assim:

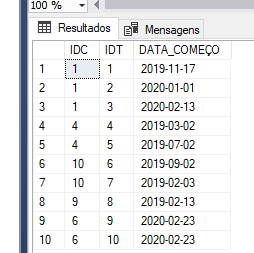
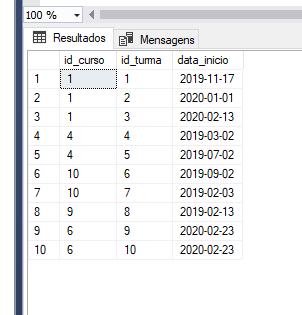


Note que depois de executado, ele só mostrou exatamente o que foi pedido no comando. SELECIONE A COLUNA ID\_TURMA, ID\_CURSO, LOGIN\_CADASTRO DA TABELA TURMAS – “SELECT ID\_TURMA, ID\_CURSO, LOGIN\_CADASTRO FROM DBO.TURMAS”.

Da para fazer varias coisas na hora de pesquisar ou selecionar uma tabela, por exemplo, mudar o nome das colunas que você seleciona. Por exemplo, suponhamos que você precisa apresentar essa tabela para um cliente da empresa, só que, você utilizou alguns nomes nas colunas que só você sabe o que quer dizer, então para facilitar na hora de selecionar e mostrar para o cliente, você quer apresentar ela com algumas alterações, sem mudar de fato o que cada coluna é, apenas para ter mais facilidade na hora do cliente olhar. O que podemos fazer é acrescentar um “APELIDO” em cada coluna, ou, umas ALIAS, que são a mesma coisa:



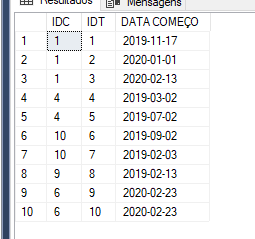
O que esse comando essa dizendo é o seguinte, QUERO ALTERAR O NOME DA COLUNA ID\_CURSOS PARA IDC, ALTERAR A COLUNA ID\_TURMA PARA IDT, ALTERAR DATA\_INICIO PARA DATA\_COMEÇO E DEPOIS ME MOSTRAR, TUDO ISSO DA TABELA TURMAS.



São as mesmas colunas da tabela TURMA, o que mudou foi que na segunda foto, em vez de digitar o comando para apresentar as colunas com os nomes que foi cadastro, utilizamos o código para que apresentasse as mesmas colunas só que com os nomes alterados, apenas para apresentação.

“OBS, DA PARA ESCREVER OS NOMES DAS COLUNAS COM ESPAÇOS E SEM “\_”, BASTA VOCE NA HORA DE DIGITAR, COLOCAR NO SCRIPT O NOME DA TABELA ENTRE ASPAS DUPLAS:





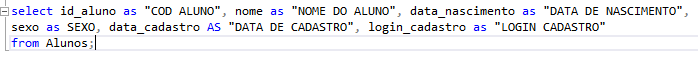
PODE VER QUE A ULTIMA COLUNA FICOU COM ESPAÇAMENTO ENTRE AS PALAVRAS. ISSO SERVE PARA FACILITAR E DEIXAR AINDA MAIS BONITO NA HORA DE APRESENTAR.

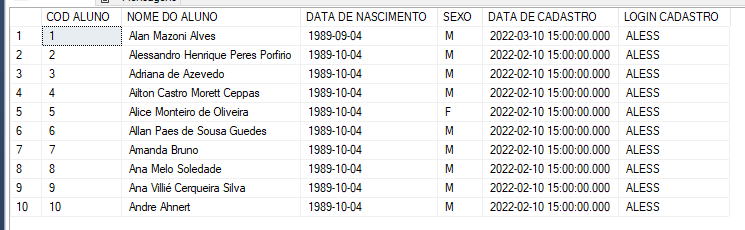
Segue alguns exemplos:





Selecionei a tabela de alunos para visualizar os nomes de cada coluna, para tentar modicar usando o código AS, para ficar de fácil entendimento ou para melhorar na hora de apresentação.

Depois usei o código em quase cada coluna e modiquei usando o código, e ficou assim:



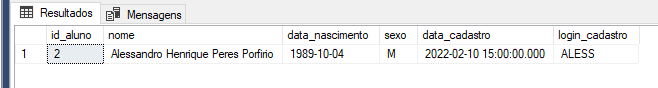
Note que tem a mesma quantidade de colunas, as mesmas informações em cada coluna, então apenas modiquei o nome de cada uma delas.

Bom ressaltar que depois que você faz essa alteração, elas não ficam como definitivas, basta você dar outro SELECT e notar que tudo volta ao normal de acordo de como cada tabela foi criada, então basicamente isso serve para facilitar em analises, apresentações e outros casos.

Para mudar definitivamente as colunas, você deve utilizar o comando ALTER TABLE.

Voltando a parte do SELECT, agora vamos aprender sobre WHERE, que é um comando que busca determinadas informações, exemplo, um nome, uma cidade, um telefone, uma chave primaria:

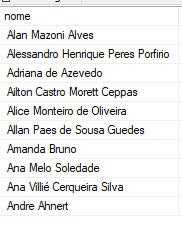
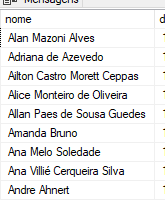




Notem que o comando é o mesmo quando vamos buscar as informações de uma tabela, só que acrescentamos a palavra WHERE para buscar uma determinada informação, que nesse caso, foi o nome ‘ALESSANDRO HENRIQUE PERES PORFIRIO’. Então basicamente se le assim o código digitado

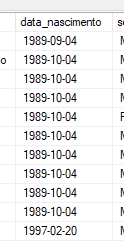
‘SELECIONE NA TABELA ALUNOS, NA COLUNA NOME, O NOME ALESSANDRO HENRIQUE PERES PORFIRIO’.

O sinal de igual, seria uma das coisas principal também na hora de apresentar ou buscar alguma informação, nesse caso a cima, estamos buscando um nome IGUAL a ALESSANDRO, caso você mude ele para um sinal de DIFERENTE <>, o sistema vai mostrar tudo o que esta na coluna nome, exceto, o nome ALESSANDRO:

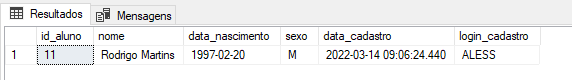


Na primeira imagem foi feito um SELECT padrão, que mostra tudo. Já na segunda imagem, foi feito conforme dito no exemplo, SELECT com WHERE <> ‘ALESSANDRO’, então o sistema entende que é apenas para mostrar todos os nomes da coluna exceto o nome de ALESSANDRO.

Outro exemplo seria puxar as informações com outra coluna, por exemplo, coluna de DATA DE NASCIMENTO na tabela ALUNOS:





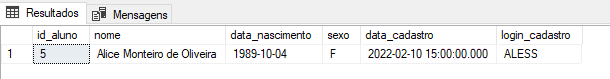


A primeira imagem são todas as datas cadastradas na tabela, e em seguida, foi feito o seguinte comando “SELECIONE NA TABELA ALUNOS, COLUNA DATA NASCIMNETO, TODAS AS DATAS QUE SEJAM MAIORES OU IGUAIS A 01/01/1997”.

E na segunda imagem, da para ver que ele retirou todas as outras informações e buscou somente a que estava no comando script, e isso pode ser em qualquer tipo de comparação, maior, menor, igual, maior ou igual, menor ou igual, diferente, igual, e assim vai.

Outro exemplo:

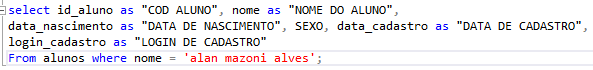


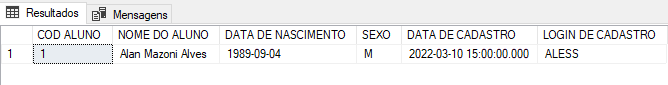


Nesse exemplo, o comando foi feito para buscar as pessoas que são do sexo FEMININO na tabela alunos.

Com isso da para perceber que ele puxa qualquer tipo de informação que a pessoa deseja, basta escrever o script de forma correta.

O que da para fazer agora, é tentar ir juntando as coisas uma com as outras, por exemplo, o sistema permite que a gente mude a nomenclatura de cada coluna, na hora de darmos um WHERE, exemplo:



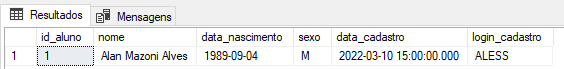


Aparentemente parece ser difícil, mas é simples quando se pratica. Esse código grande, nada mais é que a junção da função de apelidos “AS”, com a de WHERE.

“SELECIONE E ALTERE A COLUNA” ID\_ALUNO PARA COD ALUNO, SELECIONE E ALTERE A COLUNA NOME PARA NOME DO ALUNO, SELECIONE E ALTERE A COLUNA DATA\_NASCIMENTO PARA DATA DE NASCIMENTO, SELECIONE A COLUNA SEXO, SELECIONE E ALTERE A COLUNA DATA\_CADASTRO PARA DATA DE CADASTRO E DEPOIS ME MOSTRE APENAS AS INFORMAÇÕES DA PESSOA QUE SE CHAMA ‘ALAN MAZONI ALVES’ NA COLUNA NOME.

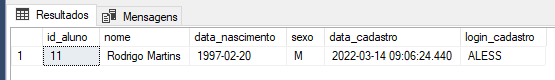
Se esse comando for feito de forma simples, ficaria basicamente assim:





Que seria a mesma coisa, apenas foi alterado o jeito de como mostrar o resultado. Isso é bom para apresentações, caso a nomenclatura das colunas fossem mais difíceis de compreender entre pessoas leigas. Então você pode mudar tudo antes de apresentar.

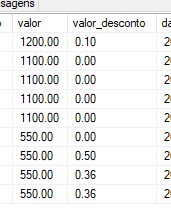
Podemos também usar mais de uma pesquisa dentro de um comando, por exemplo, além de buscar apenas algumas pessoas entre uma data de nascimento, podemos buscar essa data de nascimento, e as pessoas que tem o sexo M. outro exemplo:



“SELECIONE NA TABELA ALUNOS, AS PESSOAS QUE TEM O SEXO M E COM A DATA DE NASCIMENTO MAIOR OU IGUAL 01/01/1997”.

Isso acontece por causa do “AND” que esta escrita no comando script, ele significa E, então quer dizer, busque tal informação e outra. E no mesmo comando pode ter vários “AND”, basta digitar certinho qual você deseja apresentar e apertar para executar.

OBS = DA PARA REALIZAR CONTAS DENTRO DO PROPRIO SELECT E FAZER COM QUE ELE MOSTRE NO FINAL O RESULTADO:

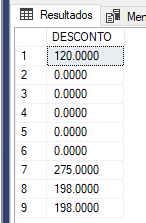




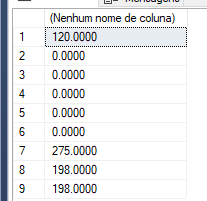
Se dermos o comando SELECT \* FROM ALUNOSXTURMAS, vai aparecer a tabela inteira, e como podemos notar, na segunda imagem, existe os campos VALOR e VALOR\_DESCONTO. O que queremos agora, é somar ou multiplicar essas colunas para saber qual será o resultado e a quantidade de desconto que esses alunos terão. Pra isso, basta digitar o seguinte comando:

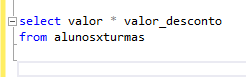


“SELECIONE E MULTIPLIQUE NA TABELA ALUNOSXTURMAS, AS COLUNAS VALOR E VALOR DESCONTO, E DEPOIS MOSTRE O RESULTADO EM UMA COLUNA CHAMADA ‘DESCONTO’”.



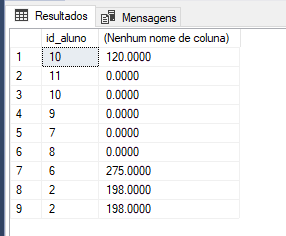
O sinal de \* dentro desse comando, esta se referindo à operação de multiplicação, e em seguida estamos dando um nome para a coluna que ele ira apresentar esse resultado, pois se não dermos o nome dela, ela aparecera da seguinte forma:





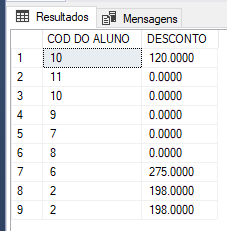
Notem que no final do da primeira linha do código, não tem o comando ‘AS’ e nem o nome da coluna de resultado, por isso que aparece na segunda imagem, a coluna “NENHUM NOME DE COLUNA”. Então a forma correta seria colocar sempre o comando do SELECT, as colunas que vão ser multiplicadas e em seguida colocar o nome da coluna com comando AS.

Para ficar ainda mais informativo, com mais detalhe dentro da tabela de resultado, podemos também acrescentar mais colunas:



Nesse caso, foi escrito o comando de forma simples, sem nada de alteração, apenas para mostrar que podemos fazer dessa forma mais simples. Só que como podemos ver na segunda coluna, ela esta sem nome, e assim fica difícil de entender o que se trata aquelas informações. Então podemos fazer algumas alterações nela:



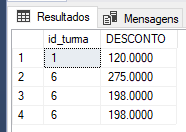
Lembra-se da questão de apelidos? Então, aqui podemos também mudar o nome da coluna de id\_alunos e darmos o nome da coluna de resultado da operação: 

Notem que o que era ID\_ALUNOS passou a ser COD DO ALUNO e a coluna de resultado agora tem um nome, DESCONTO. Isso melhora muito na hora de uma apresentação.

Para filtrar ainda mais, vamos tirar os valores que estão com o desconto zerados, pois queremos apenas que o sistema nos mostre os valores que tem desconto:



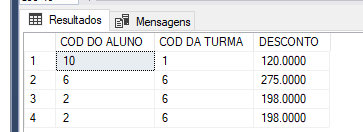
“SELECIONE E MOSTRE DA TABELA ALUNOSXTURMAS A COLUNA ID\_TURMA, DEPOIS MULTIPLIQUE A COLUNA VALOR COM A VALOR\_DESCONTO, MOSTRANDO NA COLUNA DE RESULTADO DESCONTO, SOMENTE QUANDO O VALOR\_DESCONTO FOR MAIOR QUE ZERO”



O resultado do comando ficara assim. Caso você queira incrementar mais ainda, para deixar mais explicativo, podemos modificar algumas coisas, mudando os apelidos:



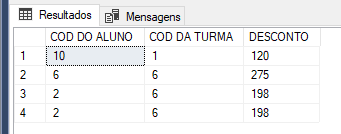
Nesse comando foi acrescentada a coluna ID\_ALUNO mudando a nomenclatura dela para COD DO ALUNO. Deixamos a coluna ID\_TURMA e alteramos para COD DA TURMA e deixamos a coluna DESCONTO:



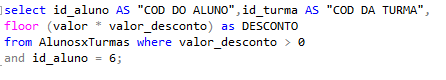
Aqui já da pra saber quem é o aluno, que turma ele esta e qual serão o valor de desconto dele. Pra deixar mais limpo ainda, que tal tirar as casas decimais dos valores de desconto?

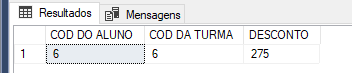
Para isso, você terá que acrescentar o comando FLOOR antes da operação que vai ser realizada, nesse caso, será a operação de multiplicação das colunas VALOR E VALOR\_DESCONTO. É mais simples do que parece, apenas escreva FLOOR (VALOR \* VALOR\_DESCONTO)... Isso significa que você deseja que os números decimais da operação de multiplicação entre as colunas valor e valor\_desconto não apareçam:





Ficando assim no final. E você pode também acrescentar alguns AND juntos a esses outros comandos:





Nesse exemplo foi filtrado também os ID\_ALUNO, então basicamente eu quero que apareça o valor de desconto sem as casas decimais, com o valor de desconto maior que zero, e dentro deles apenas o aluno que tem o código 6.

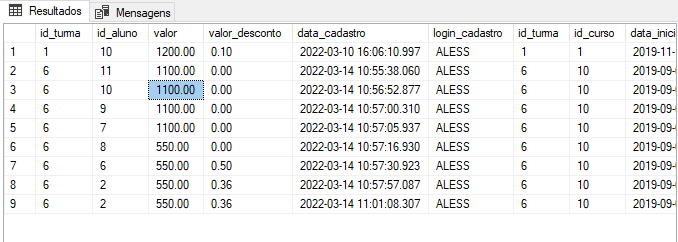
Caso você queira puxar e saber o nome do curso, o nome do aluno de outras tabelas basta fazer alguns comandos. Primeiro, lembre-se das ALIAS, que é os apelidos que você da para cada coluna.

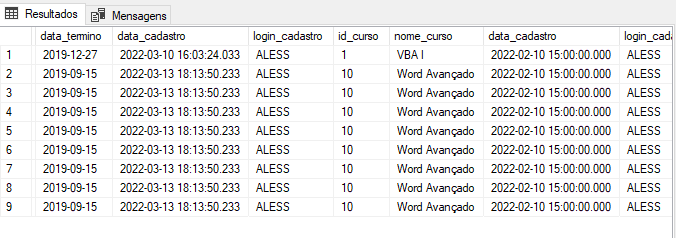


Nesse caso eu apelidei a tabela ALUNOSXTURMAS com AT, a tabela TURMAS com T e a tabela CURSOS com C. Esse comando esta querendo dizer o seguinte:

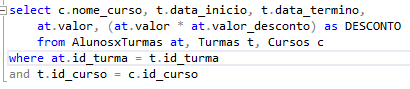
“SELECIONE A TABELA ALUNOSXTURMAS, A TABELA TURMAS E A TABELA CURSOS, E ME MOSTRE TODOS OS CAMPOS DELAS, MAS NÃO SE ESQUEÇA QUE O ID\_TURMA NA TABELA ALUNOSXTURMAS É UMA CHAVE ESTRANGEIRA DA TABELA TURMA, E, O ID\_CURSO NA TABELA ALUNOXTURMAS É UMA CHAVE ESTRANGEIRA DA TABELA CURSO”

No resultado o sistema vai trazer todas tabelas mencionas, tudo na sequencia.

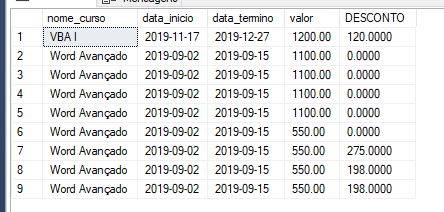




Agora que no comando eu já disse para o sistema qual será as tabelas e a suas relações, vamos então filtrar alguns campos dessas tabelas:



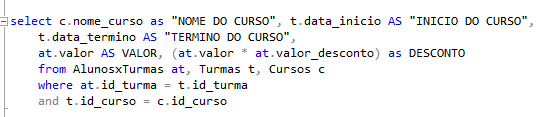
O comando digitado significa que: “SELECIONE NA TABELA CURSOS A COLUNA NOME\_CURSO, DEPOIS NA TABELA TURMAS A COLUNA DATA\_INICIO, DEPOIS NA TABELA TURMAS DATA\_TERMINO. EM SEGUIDA, ME MOSTRE O RESULTADO DA OPERAÇÃO DE MULTIPLICAÇÃO ENTRE AS COLUNAS VALOR E VALOR\_DESCONTO DA TABELA ALUNOXTURMAS, MODIFIQUE O NOME DESSA COLUNA DE RESULTADO PARA DESCONTO, LEMBRANDO SEMPRE QUE A TABELA ALUNOSXTURMA É AT, TURMAS É T E CURSOS É C. E TAMBEM QUE A COLUNA ID\_TURMA NA TABELA ALUNOXTURMAS É UMA CHAVE ESTRANGEIRA DA TABELA TURMAS ID\_TURMA, E, A COLUNA ID \_CURSO NA TABELA TURMA É UMA CHAVE ESTRANGEIRA DA TABELA CURSO”.

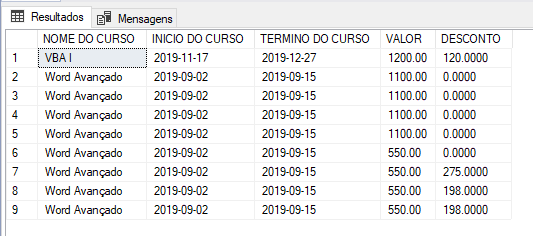


Como pode ver, o sistema puxo as seguintes informações, NOME DO CURSO, DATA DE INICIO, DATA DE TERMINO, VALOR E DESCONTO, como esta escrito no comando no script.

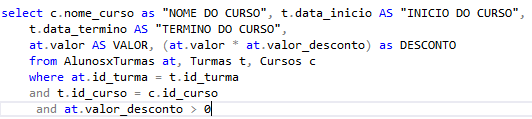
OBS: SE FOR BUSCAR MAIS DE UMA TABELA, E ELAS TEREM RELAÇÃO ENTRE SI, NO CASO, TER UMA CHAVE ESTRANGEIRA DENTRO DELAS, TERA QUE DIZER ISSO NO COMANDO WHERE, LEMBRANDO SEMPRE DE COLOCAR ALIAS PARA FACILITAR NA HORA DO COMANDO.

Podemos também acrescentar algumas modificações dentro dessas colunas, como mudar as alias delas sem modificar o resultado final:

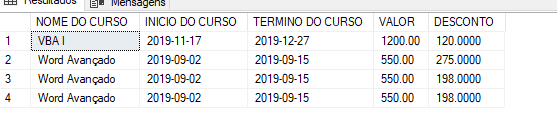
Nesse comando apenas vai mudar os nomes das colunas, para ficar mais fácil de entender do que se trata, ou quais tipos de informações contem cada coluna.



Podemos ir mais a fundo ainda, eliminando as linhas aonde o desconto vai dar zero:

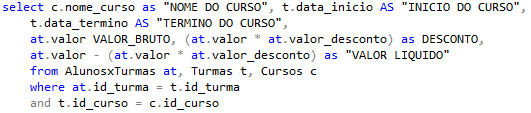


Nesse caso foi acrescentado mais um filtro com a opção AND, dizendo que é para aparecer todos os outros campos, exceto aqueles que a coluna de VALOR DE DESCONTO é menor que zero:

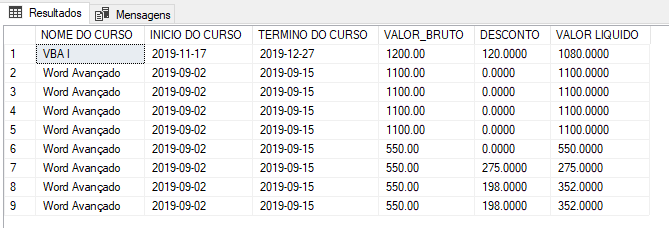


E nisso ele mostrou apenas quatro linhas.

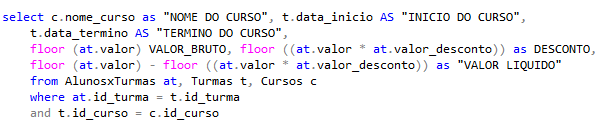
Podemos também acrescentar mais uma coluna, para mostrar o valor liquido que cada aluno vai pagar:

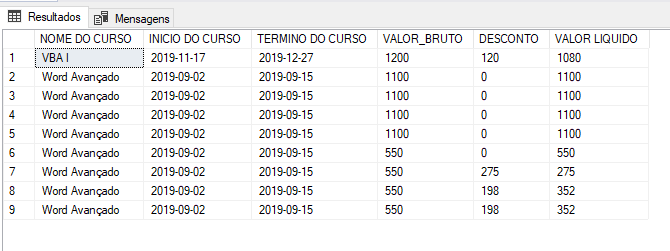


Repare que foi acrescentado apenas mais uma linha dentro do script, que nada mais é que a coluna VALOR DA TABELA ALUNOxTURMA – (VALOR DA TABELA ALUNOxTURMA \* VALOR DO DESCONTO DA TABELA ALUNOxTURMA).



Podemos também tirar as casas decimais, para ficarem mais limpo os valores em dinheiro, acrescentando o comando FLOOR na frente de cada operação, lembrando-se de colocar entre parênteses sempre.



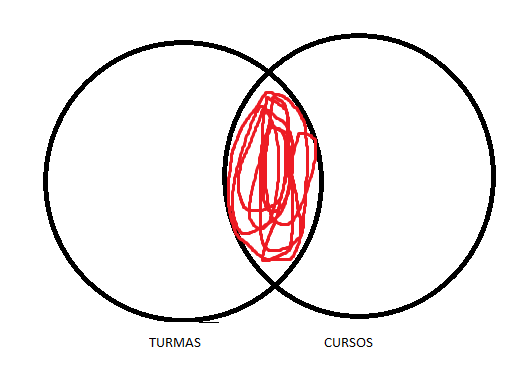


AULA 5 – JOIN (JUNÇÃO DE TABELAS)

A função join pode ser explicada como uma função para unir e apresentar mais de uma tabela. Essa junção dependera das chaves primarias e chaves estrangeiras existentes em cada tabela. Por isso é de boa importância saber quais são essas chaves e como relaciona-las entre si.

Pra ficar mais fácil de entender como escrever os comandos de INNER JOIN, RIGHT JOIN OU LEFT JOIN, decidi colocar como desenho no paint.

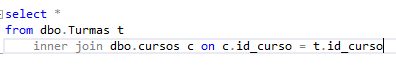
Então a primeira imagem a baixo, seria o INNER JOIN, todas as informações que existem e se relacionam nas duas tabelas, excluindo as informações que não são necessários por enquanto, que no caso, são as que não se relacionam.



Tudo o que estiver no meio dessas informações, vao ser apresentadas no comando de INNER JOIN, caso ao contrario, não apareceram no resultado. Então para que o comando funcione, lembre-se que para dar o JOIN em duas ou mais tabelas, deve existir dentro delas a chave de ligação, que seria a chave estrangeira. Sem a chave estrangeira, não será possível o comando JOIN, pois é por ela que as informações vão ter ligamentos ou relação e assim poderão ser mostradas no resultado

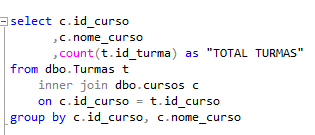
-INNER JOIN = basicamente seria mostrar as informações que cada tabela tem em comum. Basicamente o comando seria assim:

“SELECIONE E MOSTRE AS INFORMAÇÕES DA TABELA ‘A’ E AS INFORMAÇÕES DA TABELA ‘B’ QUE TEM RELAÇÃO ENTRE SI”.



Nesse exemplo acima, queremos saber qual é o total de turmas por curso, então vamos fazer uma junção da tabela turmas com a tabela cursos. Então, SELECT \* FROM DBO.TURMAS T(selecione todas as informações da tabela turmas alis t), INNER JOIN DBO.CURSOS C (juntando as informações da tabela cursos alias c), ON C.ID\_CURSO = T.ID\_CURSO (onde o id\_curso da tabela Curso vai ser o mesmo id\_curso da tabela Turmas)

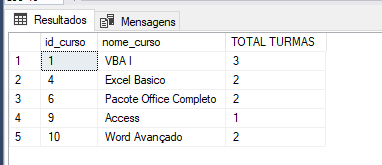
Basicamente já fizemos a estrutura para calcular a quantidade de turmas por curso, basta colocar agora as colunas que você quer que apareça e a função de agregação COUNT para contar e nos mostrar o resultado com outra coluna de resultado, lembrando sempre em colocar o nome da ALIAS:



A função COUNT serve para somar ou para mostrar um resultado de soma dentro de uma coluna de resposta.

Notem que agora estamos pedindo para que o comando nos mostre a coluna ID\_CURSO da tabela CURSOS, a coluna NOME\_CURSO da tabela CURSOS, depois, quer que ele conte quantos ID\_TURMA existem na tabela TURMAS usando a função COUNT e que me mostre isso em uma coluna de repostas chamada TOTAL TURMAS.

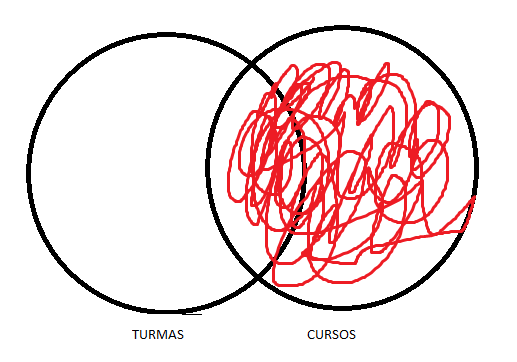
Como a gente pediu uma agregação, uma função e agregação, devemos então agrupar tudo o que vem antes dessa função, que seria as colunas C.ID\_CURSO, C.NOME\_CURSO, pois se não da erro na hora de apresentar as informações.

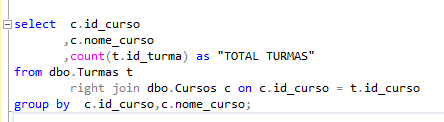


Então basicamente ficara assim, com o INNER JOIN vai ser apresentado apenas o que tem RELAÇÃO ENTRE AS DUAS TABELAS, AS INFORMAÇÕES QUE CONTEM NA NAS MESMAS. Ele vai me mostrar como pedido a coluna id\_curso, nome\_curso e total de turmas que foi uma função COUNT.

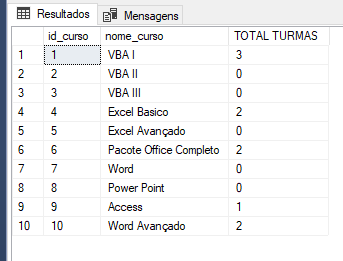
-RIGHT JOIN = Seria outra variável de JOIN, de junção. Nesse caso, ele apenas me mostra entre as tabelas relacionadas, o que esta na tabela da DIREITA, que sempre é tabela depois do comando JOIN, e as informações que coincidem com a tabela da esquerda.

Então na imagem a baixo, o que podemos observar é que, o comando vai pedir para que me mostre todas as informações da tabela Cursos e as que tem relação com a tabela Turmas



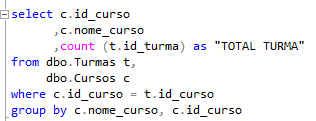


Nesse caso pedimos para o sistema nos mostrar a coluna ID\_CURSO da tabela CURSOS, NOME\_CURSO da tabela CURSOS, depois pedimos para ele contar e mostrar em uma coluna de resultado o total de ID\_TURMA na tabela turma, independente se há turmas ou não, ele vai mostrar todos os resultados de cursos e tudo que tem relação com a tabela cursos.

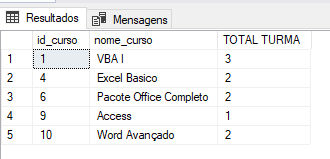


Podem ver que aqui temos turmas zeradas, pois no comando pedimos pra ele mostras todos os cursos independentes se tem turma ou não.

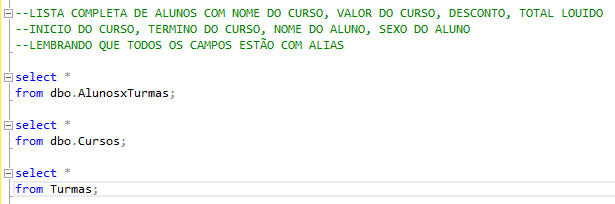
Da para fazer um INNER JOIN utilizando o comando WHERE:

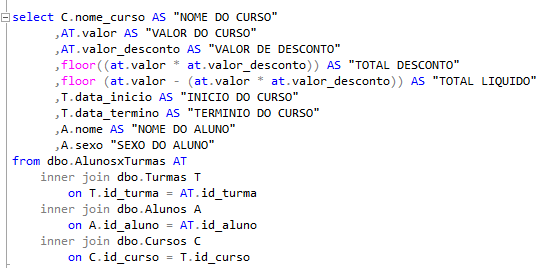


Basicamente aqui fizemos a mesma coisa do que utilizar um INNER JOIN, só que em vez de relacionar depois com a outra tabela, colocamos as duas que tem relação na parte do FROM, e utilizamos a função WHERE para relacionar as duas chaves que relacionam elas ,e mostrar o que estamos pedindo no select, que é listar os cursos que tem turma e contar cada um deles.

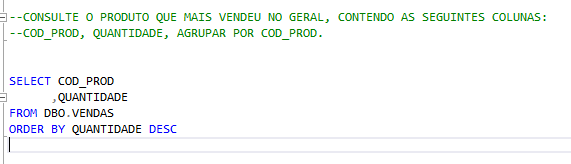


Fizemos outro exemplo para praticar:





Nesse caso, fizemos uma lista completa de aluno, contendo as seguintes informações: NOME DO CURSO, VALOR DO CURSO, DESCONTO, TOTAL LIQUIDO, INICIO DO CURSO, TERMINO DO CURSO, NOME DO ALUNO, SEXO DO ALUNO, COLOCANDO ALIAS EM TODOS OS CAMPOS.

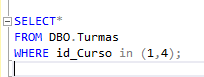
OBS: 

Nesse exercício o enunciado estava pedido para trazer o produto que mais vendeu e depois mostrar junto com ele o código daquele produto. Então, fizemos o seguinte comando:

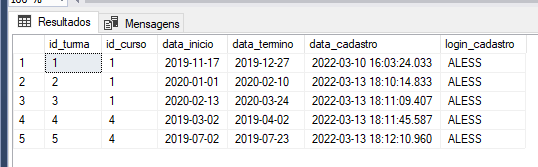
SELECIONE COD\_PRODUTO, QUANTIDADE DA TABELA DBO.VENDAS E ORDENE A QUANTIDADE EM ORDEM DESCRECENTE. Pois ai ele vai mostrar junto com o código do produto, a quantidade, e com o comando DESC, ele vai mostrar em ordem DECRESCENTE, mostrando qual o que mais vendeu.

AULA 6 – UTILIZANDO CLAUSULA IN

O comando ‘IN’ seria basicamente você pedir para o SQL mostrar o resultado que tem algumas informações, por exemplo:



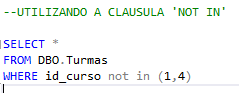
Nesse comando estamos pedindo para que o SQL nos mostre todos os resultados que estão com o id\_curso igual a 1 e 4:



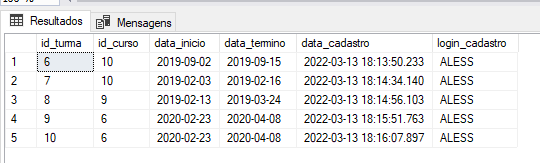
Notem que ele mostrou todos as colunas que tem esse id\_curso 1 e 4, como solicitado no script.

- CLAUSULA NOT IN

Seria basicamente o comando contrario da clausula IN, nesse comando, o sistema nos mostrara todas as informações que NÃO estão dentro daquele código. Por exemplo:



Resumindo, estamos pedindo para que o sistema no mostre todas as informações que existem dentro da tabela turmas, que NÃO TEM O ID\_CURSO IGUAL A 1 E 4.



Podemos ver que ele excluiu todas as informações que continham o ID\_CURSO igual a 1 e 4.

-FUNÇÃO DATEPART

A função DATEPART é um comando do SQL que extrai parte de uma data, ela pode trazer o ano, o dia ou o mês. Para que ele de certo, você deve sempre colocar os valores em ordem, e para facilitar ainda mais, o próprio SQL mostra qual é essa sequencia, por exemplo:

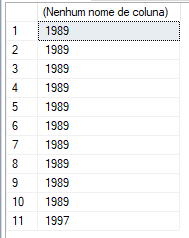


No próprio comando, vai aparecer a ordem de informações que você deve colocar, para que ele retorne o resultado esperado:

INTERVALO – QUAL O INTERVALO QUE EU QUERO DE INFOMAÇÃO, nesse caso vamos c colocar o YEAR, que será o ano de nascimento do aluno na tabela alunos.

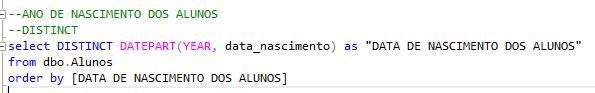
CAMPO – QUAL CAMPO VOCE VAI PEGAR ESSA INFORMAÇÃO, nesse caso vai buscar a informação no campo DATA\_NASCIMENTO na tabela alunos.

Depois de fazer o passo a passo do comando, lembre-se de colocar a TABELA que você vai querer pegar todas essa informações. Nesse exemplo estamos utilizando a tabela ALUNOS.



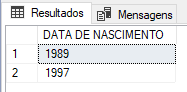


Notem na imagem à direita, que ele retornou os anos, só que, tem vários repetidos, e caso você pegue uma tabela com vários anos, isso pode atrapalhar um pouco na hora de contar ou analisar. Para mudar isso, fazendo com que ele mostre apenas um ano de cada, basta fazer o seguinte comando:

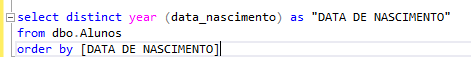


DISTINCT você poderá usar para que o comando mostre apenas um ano de cada, sem que ele mostre todos repetidamente.

No exemplo acima estamos basicamente dizendo assim, “ME MOSTRE O ANO DE NASCIMENTO DOS ALUNOS NA TABELA ALUNOS, TIRANDO OS ANOS REPETIDOS, APENAS COLOCANDO UM ANO DE CADA”. Então a função DISTINCT faz com que o resultado não fique tão poluído, filtrando as informações desnecessárias.



Vale lembrar também que existe outra função igual a DATEPART, que seria selecionar o que você busca dentro daquela coluna:



Nesse exemplo, colocamos o YEAR depois do select distinct para buscar apenas o ano da tabela ALUNOS na coluna DATA\_NASCIMENTO.

AULA 7 – FUNÇÕES DE AGREGAÇÕES

SUM = a função SUM, é uma função que soma os valores de determinada coluna. Basta colocar a função primeira e depois colocar a coluna que você deseja somar entre parênteses:



Nesse caso, ele vai trazer todos os valores da coluna VALOR da tabela ALUNOxTURMA somados, lembrando sempre em colocas um AS para o resultado dessa soma.

ROUND = a função ROUND, é uma função de agregação que mostra os valores decimais de uma determinada coluna, caso ela tenha esses valores. Basta colocar a função antes da coluna que você deseja realizar a função entre parênteses, lembrando que ela pode ser adicionada junta com a função SUM, colocando entre parênteses e depois colocar a quantidade de casa que você quer que apareça:



Nesse exemplo, estamos somando o resultado da operação de duas colunas. Primeiro, fizemos a operação de vezes entre o valor do curso na tabela ALUNOxTURMA e depois somamos o resultado com a função SUM. Em seguida, utilizamos a função ROUND para NÃO MOSTRAR o as casas decimais depois da virgula, basta olhar pelo “0” que está no final do comando, pois é esse valor no final que vai falar quantas casas depois da virgula vai ser exibida.