

SQL Para Data Science

Conteúdo Programático

- | | | | |
|----|---|----|--|
| 01 | Introdução | 04 | Junção de Tabelas |
| 02 | Consulta, Filtro, Ordenação e Operadores | 05 | Agregando Dados Para Análise |
| 03 | Categorização, Codificação e Binarização de Variáveis com SQL | 06 | Window Functions, Subqueries e Tratamento de Datas |

SQL Para Data Science

Conteúdo Programático

- | | | | |
|----|--|----|-----------------------------------|
| 07 | Estudo de Caso 1 - Análise Exploratória de Dados com SQL | 10 | Programação em SQL |
| 08 | Limpeza e Processamento de Dados com SQL | 11 | Otimização de Consultas SQL |
| 09 | Análise de Dados com SQL | 12 | Módulo Bônus - Google Data Studio |

O Que é Linguagem SQL?

- SQL é uma linguagem padrão para acessar e manipular dados.
- SQL significa Structured Query Language (Linguagem de Consulta Estruturada).
- SQL se tornou um padrão do American National Standards Institute (ANSI) em 1986 e da International Organization for Standardization (ISO) em 1987.
- Embora SQL seja um padrão ANSI / ISO, existem diferentes versões da linguagem SQL.
- No entanto, para serem compatíveis com o padrão ANSI, todas as versões suportam pelo menos os comandos principais (como SELECT, UPDATE, DELETE, INSERT, WHERE) de maneira semelhante.



Quais softwares iremos utilizar neste curso?

- Iremos utilizar o SGBD (Sistema Gerenciador de banco de dados) O SQL.
- E com o Google DataStudio no último Capítulo.

2 - Instalação.

terça-feira, 25 de outubro de 2022 22:06

Para efeito de início, fizemos o download dos executáveis diretamente na página do SQL.

 mysql-workbench-community-8.0.31-win64.msi
 mysql-installer-community-8.0.31.0.msi

Foram eles o Instalador do SGBD e o Workbench.

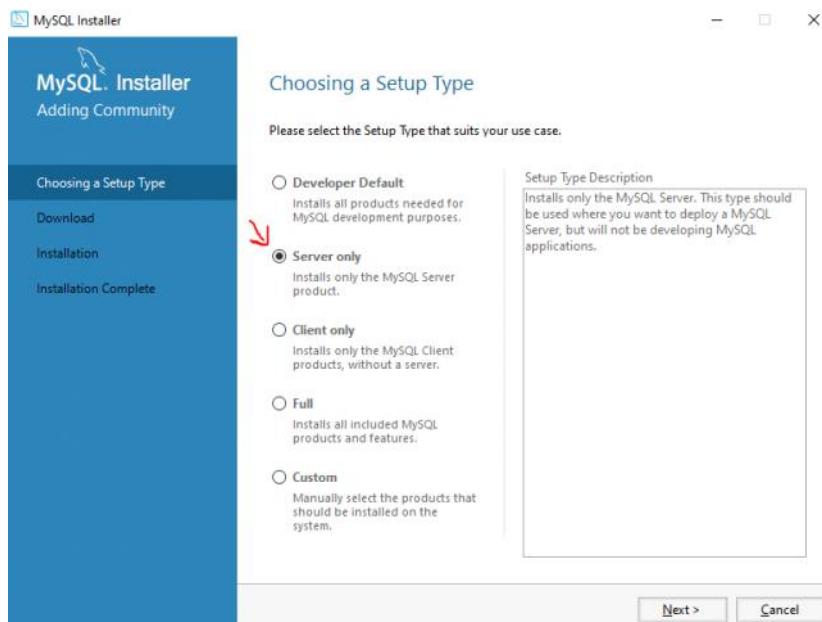
Qual a função do Workbench do MySql?

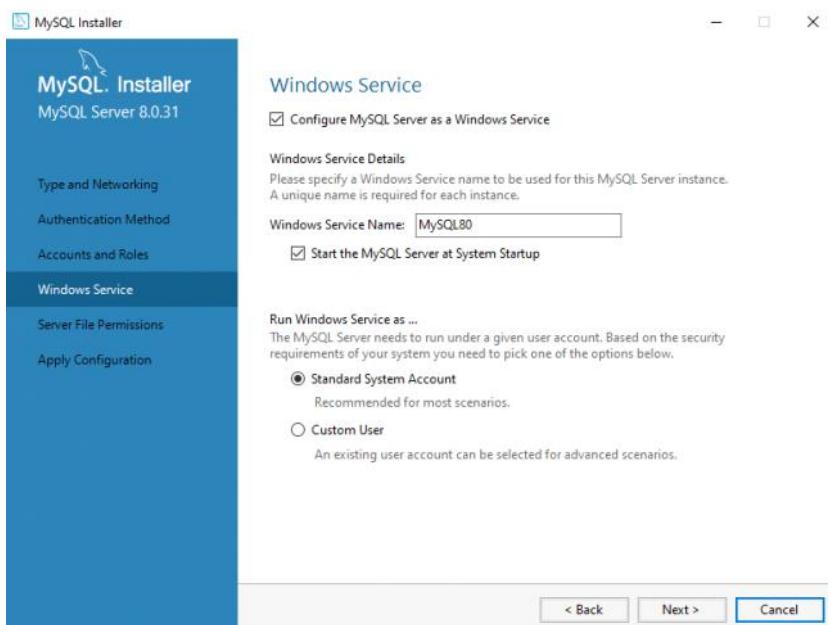
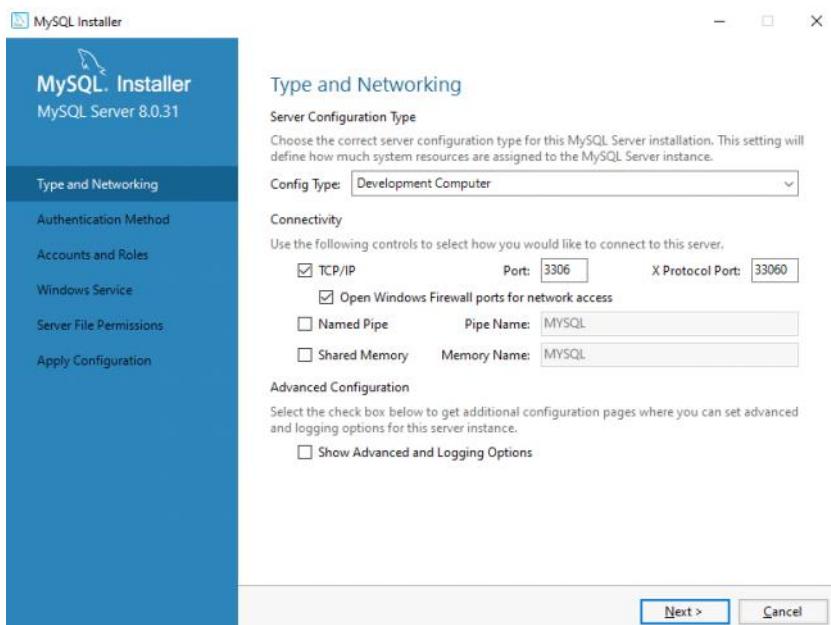
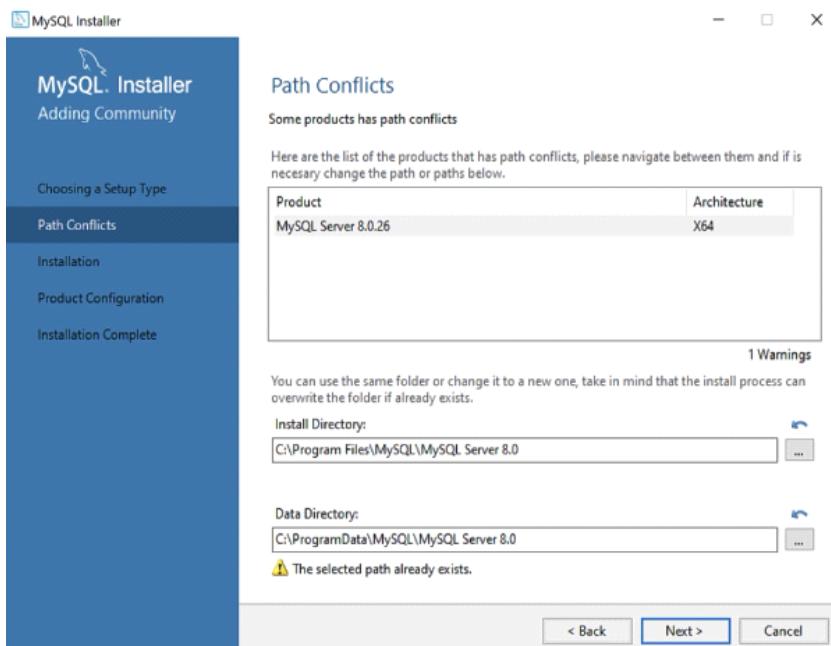
Através do MySQL Workbench, pode-se executar consultas SQL, administrar o sistema e modelar, criar e manter a base de dados através de um ambiente integrado. O MySQL Workbench está disponível para Windows, Linux e Mac OS.

Instalando o SGBD.

Conforme a imagem abaixo, seguimos o seguinte passo:

- Server Only >> Clicar em next.
- Escolha seu diretório de instalação >> Next.
- Deixe o padrão de instalação, com o config type: Development Computer>> Next.
- Na página de senhas digite sua senha forte, e não esqueça esta senha de jeito algum, No meu caso, escolhi a senha: "Rosanapf123!">> Clicar em next.
- Marque a opção de rodar como um serviço, pra não precisar ficar inicializando o SQL.





3 - Introdução ao Banco de dados e Modelos relacionais

quinta-feira, 27 de outubro de 2022 10:24

Terminologia

SGBD – Sistema Gerenciador de Banco de Dados

Banco de Dados – Arquivos que armazenam os dados

Banco de Dados Relacional – Arquivos que armazenam dados de forma relacional, onde os dados são distribuídos em tabelas que se relacionam

Modelo Relacional Lógico – Modelo lógico que determina como os dados se relacionam

Modelo Relacional Físico – Instruções SQL que criam o modelo no banco de dados relacional

Pode

4 - Carregando primeira extração de dados

quinta-feira, 27 de outubro de 2022 22:06

Começamos pelo CAP02.

Retiramos os dados do site do próprio governo, com base em uma inspeção de navios.

Na imagem abaixo, podemos ver que abrimos o nosso SQL Workbench, e criamos na aba Schemas ao lado da aba administration, definimos como "cap02" o nome do schema (imagem1), e abriu o quadro de review do SQL (imagem2). Veja que na imagem 2 ele mostra uma instrução DDL, que é para criar um objeto no banco de dados.

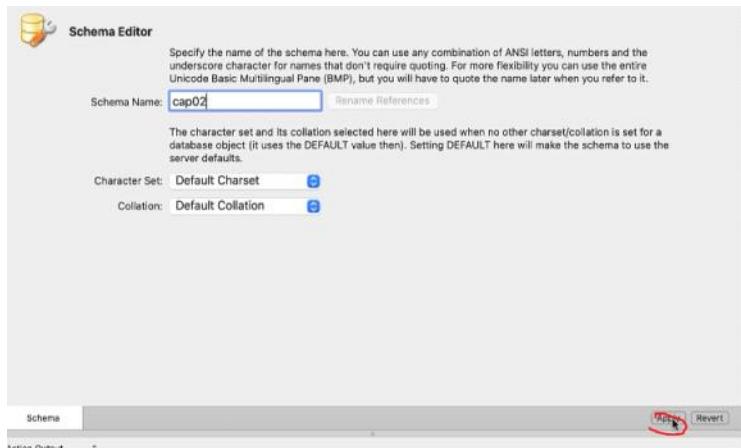


Imagen1

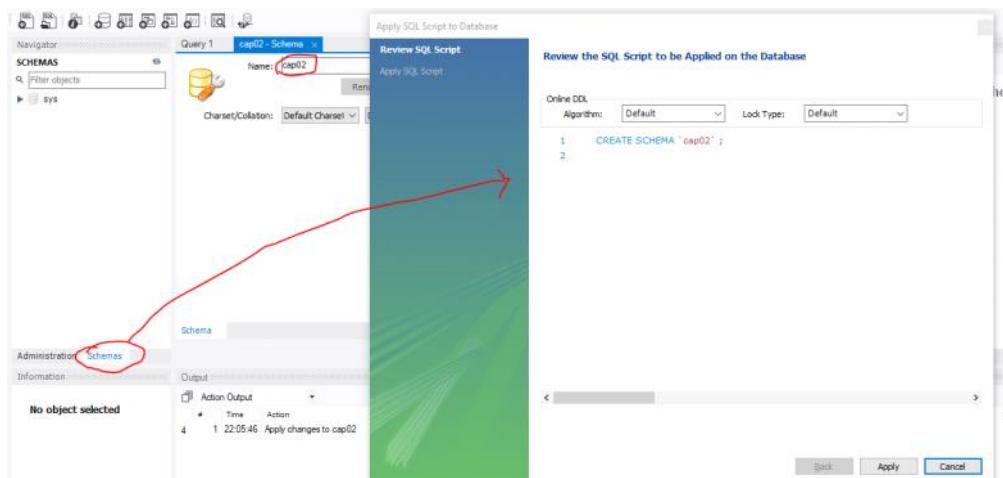


Imagen2

Feito isso,

Seguimos o passo a passo de criar uma tabela.

Bastou clicar em cima de Table com o botão direito > Create Table... > Nomeamos a tabela como tb_navios.

Conforme imagem3

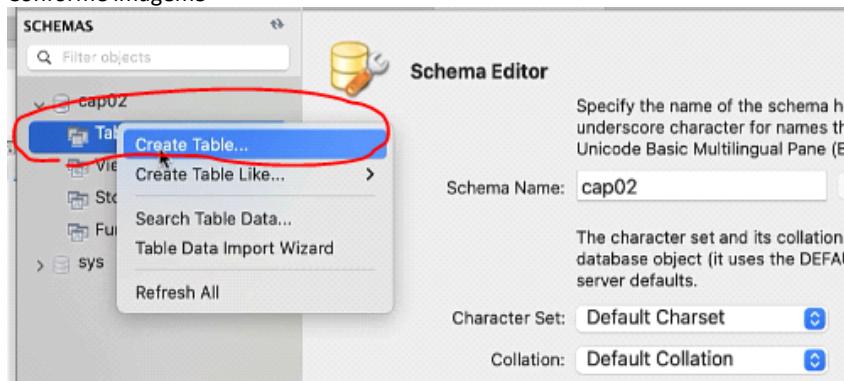


Imagen3

Criando as colunas:

Podemos ver na imagem abaixo, bem no centro da imagem a parte de Column name e datatype logo ao lado. Porém, antes de tudo temos a Table name que representa o nome da tabela, que no caso deixamos setados como tb_navios.

Na parte central da imagem4, (Column Name | Datatype, etc...)

Começamos a criar cada coluna com base nas colunas que estavam no banco de dados.

The screenshot shows the MySQL Workbench interface. In the center, there's a table definition for 'tb_navios' under the schema 'cap02'. The table has five columns: 'NOME_NAVIO' (VARCHAR(50)), 'MES_ANO_ABERTURA' (VARCHAR(10)), 'CLASSIFICACAO_RISCO' (VARCHAR(15)), 'INDICE_CONFORMIDADE' (VARCHAR(15)), and 'PONTUACAO_RISCO' (INT). Below the table definition, there's a preview of the SQL script being generated:

```
CREATE TABLE `cap02`.`tb_navios` (
  `NOME_NAVIO` VARCHAR(50) NULL,
  `MES_ANO_ABERTURA` VARCHAR(10) NULL,
  `CLASSIFICACAO_RISCO` VARCHAR(15) NULL,
  `INDICE_CONFORMIDADE` VARCHAR(15) NULL,
  `PONTUACAO_RISCO` INT NULL,
  `TEMPORADA` VARCHAR(200) NULL
)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

On the right side of the interface, there's a 'Review SQL Script' panel with the heading 'Review the SQL Script to be Applied on the Database'. It shows the generated DDL code. At the bottom of the interface, there's a taskbar with various icons and a system tray showing the date and time.

Imagen4

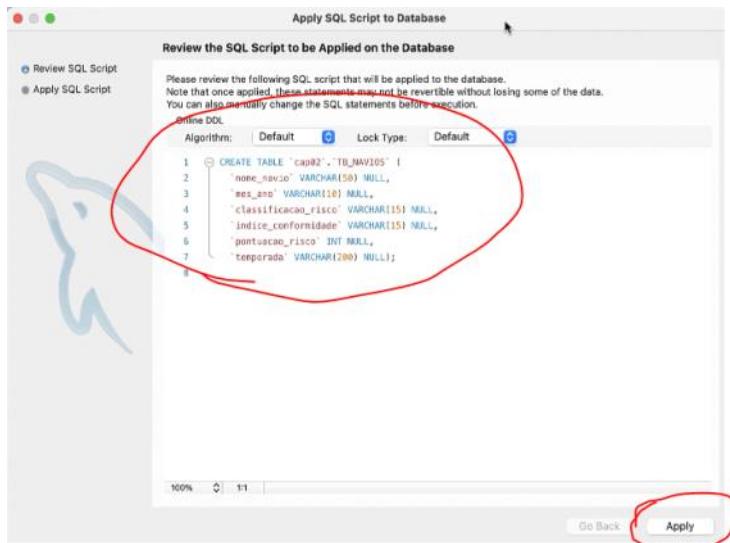
Ao lado de cada uma, no Datatype, podemos especificar cada tipo de dado que vamos inserir.

No caso do exemplo abaixo, definimos quase tudo com VARCHAR(XX) no caso o XX é um número, onde limita a quantidade de caracteres que vão ser permitidos a inserção.

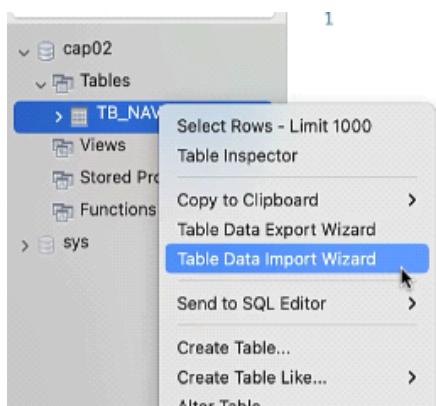
Também tem o tipo INT, que é voltado a números.

This screenshot shows the MySQL Workbench interface again, but this time with a red circle drawn around the column names 'NOME_NAVIO', 'MES_ANO_ABERTURA', 'CLASSIFICACAO_RISCO', 'INDICE_CONFORMIDADE', and 'PONTUACAO_RISCO' in the table definition. Below the table, a 'DADOS_ABERTOS_INSPECACAO_NAVIO.csv - Bloco de Notas' window is open, displaying the data to be inserted: 'NOME_NAVIO;MES_ANO_ABERTURA;CLASSIFICACAO_RISCO;INDICE_CONFORMIDADE;PONTUACAO_RISCO;TEMPORADA'. The data is separated by semicolons.

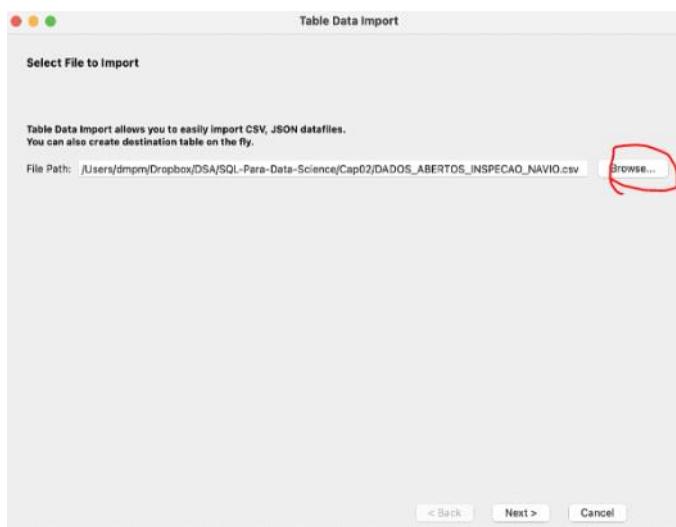
Por fim, basta verificar como os dados estão sendo inseridos corretamente conforme você configurou. E dar o famoso Apply.



Depois de aplicado, simplesmente clicamos com o botão direito em cima da tb_navios > table data import wizard. Conforme a imagem abaixo:

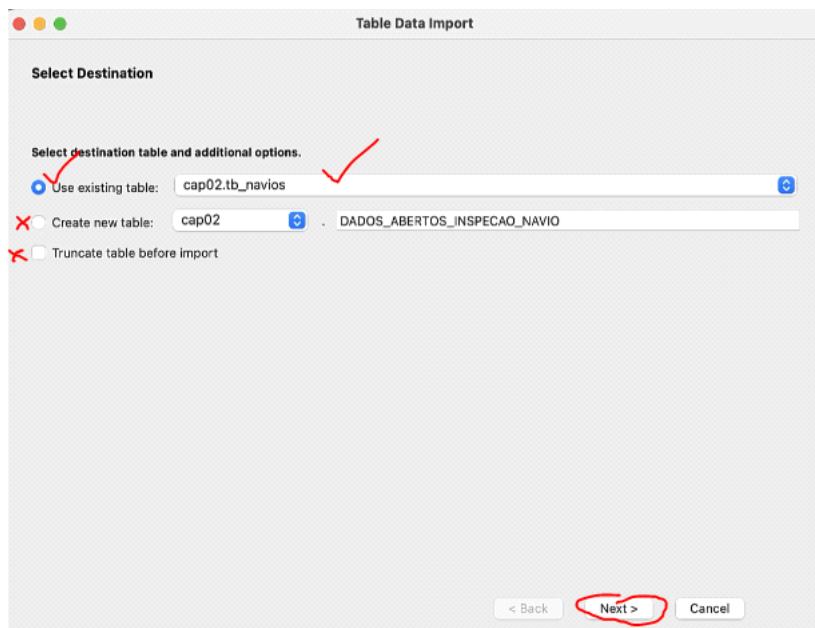


Procuramos o arquivo dos dados, no diretório o qual foi deixado.



Selecionamos nosso arquivo de dados dentro do path, logo em seguida apareceu a imagem abaixo. Clicamos em use existing table, que no caso seria pra usar a table do cap02(tb_navios) Não selecionamos nenhuma outra opção, se não as que estão configuradas na imagem a seguir.

Por fim clicamos em next.

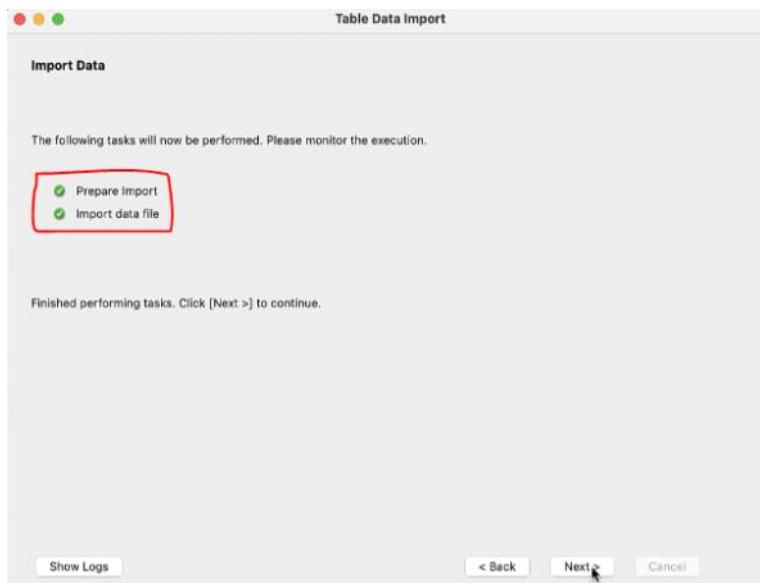


Vemos também, que as nossas configurações de importações deram certo.
Cada coluna fonte, foi encontrada na nossa extração de dados.
O delimitador que separa um dado do outro é o ponto e vírgula.

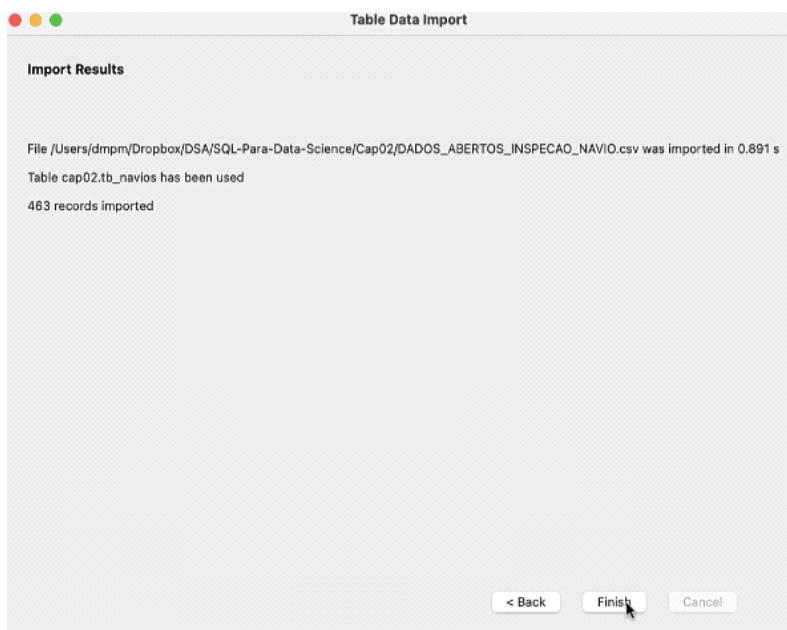
The screenshot shows the 'Configure Import Settings' step of the Table Data Import process. It displays a list of source columns and their corresponding destination columns. The source columns are: NOME_NAVIO, MES_ANO_ABERTURA, CLASSIFICACAO_RISCO, INDICE_CONFORMIDADE, and PONTUACAO_RISCO. The destination columns are: nome_navio, mes_ano, classificacao_risco, indice_conformidade, and pontuacao_risco. All mappings have checkboxes checked. Red arrows point from the source column names to their respective destination column dropdown menus. Below this, a preview table shows five rows of data with columns labeled NOME_NAVIO, MES_ANO_ABERTURA, CLASSIFICA..., INDICE_CO..., PONTUACA..., and TEMPORADA. The data rows are: HORIZON..., 12/2010, C, 90,76, 310, Programa...; SPLENDIDO..., 12/2012, C, 91,71, 415, Programa...; AZAMAR..., 02/2012, A, 97,98, 80, Programa... . At the bottom right are buttons for '< Back', 'Next >', and 'Cancel'.

NOME_NAVIO	MES_ANO_ABERTURA	CLASSIFICA...	INDICE_CO...	PONTUACA...	TEMPORADA
HORIZON...	12/2010	C	90,76	310	Programa...
SPLENDIDO...	12/2012	C	91,71	415	Programa...
AZAMAR...	02/2012	A	97,98	80	Programa...

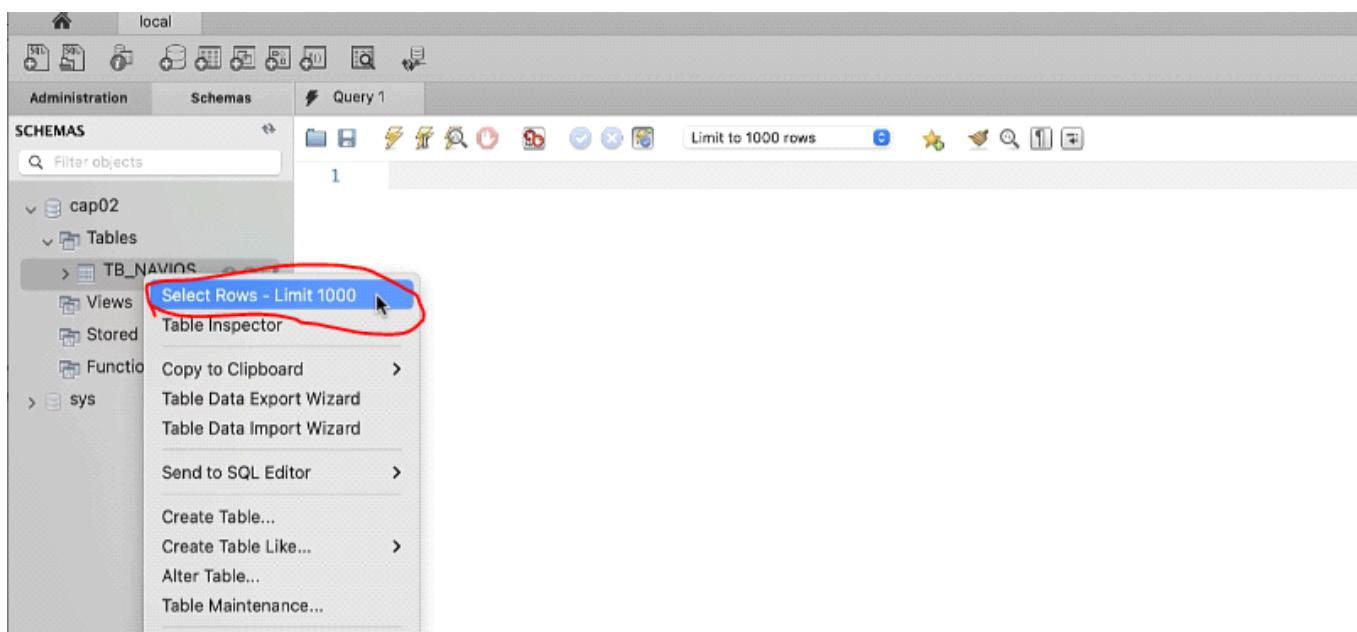
Abaixo, a confirmação que nossos dados deram certo.



Dando tudo ok, clicamos em finish.



Abrimos o power query executando os passos da ilustração abaixo
Clicando em: Tables > TB_NAVIOS (botão direito) > Select Rows - Limit 1000



E tivemos o resultado abaixo.

The screenshot shows the MySQL Workbench interface. On the left, the schema browser displays 'cap02' with 'Tables' expanded, showing 'TB_NAVIOS'. The main pane shows a query editor with the command 'SELECT * FROM cap02.TB_NAVIOS;'. Below it is a result grid displaying data from the table. The columns are: nome_navio, mes_ano, classificacao_risco, indice_conformidade, pontuacao_risco, and temporada. The data includes various cruise ship names like 'AZAMARA JOURNEY', 'BOUDICCA', 'NATIONAL GEOGRAPHIC', etc., along with their respective details. A status bar at the bottom indicates 463 rows returned in 0.00076 sec / 0.0004...

Utilizando um pouco da Power Query.

O Código abaixo:

```
1 •  SELECT * FROM cap02.tb_navios;
```

É uma linha de comando executável da query, inclusive o SELECT é um dos comandos que mais se utiliza no SQL.

SELECT * FROM cap-2.tb_navios;

A função select precede o símbolo *, este caractere, retorna no painel todas as colunas da tabela.

Faremos um exemplo, com uma outra linha de código, para a visualização de como o retorno ficará.

Utilizamos SELECT "nome da coluna", (vírgula caso eu queria incluir mais uma coluna) FROM cap02.tb_navios;
Podemos notar que ela filtrou nossa tabela carregada, conforme apontada na linha de comando.

Para executar a linha de comando basta clicar no símbolo que tem uma forma de raio.

The screenshot shows the MySQL Workbench interface with a query 'SELECT NOME_NAVIO, MES_ANO_ABERTURA FROM cap02.tb_navios;'. A red circle highlights the execute button (a lightning bolt icon). A red oval encloses the result grid, which only shows the 'NOME_NAVIO' and 'MES_ANO_ABERTURA' columns for the first few rows. The data includes 'HORIZON (PACIFIC DRE', 'SPLENDOUR OF THE SEA', 'AZAMARA JOURNEY', 'BOUDICCA', 'NATIONAL GEOGRAPHIC', 'COSTA PACIFICA', and 'SEVEN SEAS MARINER'. The status bar at the bottom shows 'tb_navios 6'.

4.1 - Cláusula WHERE e outros operadores lógicos

quinta-feira, 27 de outubro de 2022 23:05

Aplicando Filtro:

A partir do comando: `SELECT NOME_NAVIO, CLASSIFICACAO_RISCO, TEMPORADA FROM cap02.tb_navios WHERE CLASSIFICACAO_RISCO = 'D'`

Conseguimos lê-lo, mais ou menos na seguinte semântica: "SELECIONAR nome_de_uma_coluna, nome_de_outra, nome_de_outra DA TABELA esquema.nome_da_tabela ONDE nome_de_uma_coluna = (contenha) 'D'.

Pois é esse comando que fazemos no SQL.

Vide exemplo da imagem abaixo (com outra linha de comando):

The screenshot shows a MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it, the query editor window has a title bar with 'Query 1', 'cap02 - Schema', 'tb_navios - Table', and 'tb_navios'. The main area contains the following SQL command:

```
1 •    MES_ANO_ABERTURA, CLASSIFICACAO_RISCO FROM cap02.tb_navios WHERE CLASSIFICACAO_RISCO = 'A'
```

Below the command, the results are displayed in a grid:

NOME_NAVIO	TEMPORADA	MES_ANO_ABERTURA	CLASSIFICACAO_RISCO
EXPLORER	Programa de Inspecção Navios de Cruzeiro 2013...	11/2013	A
INSIGNIA	Programa de Inspecção Navios de Cruzeiro 2016...	11/2016	A
COSTA FAVOLOSA	Programa de Inspecção Navios de Cruzeiro 2012...	01/2013	A
SILVER CLOUD	Programa de Inspecção Navios de Cruzeiro 2011...	11/2011	A
SEABOURN QUEST	Programa de Inspecção Navios de Cruzeiro 2015...	05/2016	A
CLUB MED 2	Programa de Inspecção Navios de Cruzeiro 2013...	01/2014	A
SEVEN SEAS MARINER	Programa de Inspecção Navios de Cruzeiro 2014...	04/2015	A
COSTA FORTUNA	Programa de Inspecção Navios de Cruzeiro 2011...	03/2012	A
SPLENDOUR OF THE SEA	Programa de Inspecção Navios de Cruzeiro 2014...	01/2015	A
AURORA	Programa de Inspecção Navios de Cruzeiro 2014...	01/2015	A
MSC DIVINA	Programa de Inspecção Navios de Cruzeiro 2014...	06/2014	A

Com a linguagem intuitiva do SQL, pudemos aplicar uma ordenação por ordem alfabética na coluna NOME_NAVIO. Basicamente, a semântica do SQL seria assim:

- 1 SELECIONAR COLUNA, COLUNA, COLUNA
- 2 DA tabela_02.tb_navios
- 3 ONDE classificacao_risco = 'd'
- 4 ORDENANDO PELA coluna_navio;

Veja exemplo abaixo:

```

1 •  SELECT NOME_NAVIO, CLASSIFICACAO_RISCO, TEMPORADA
2   FROM cap02.tb_navios
3   WHERE CLASSIFICACAO_RISCO = 'D'
4   ORDER BY NOME_NAVIO;

```

NOME_NAVIO	CLASSIFICACAO_RISCO	TEMPORADA
AIDA CARA	D	Programa de Inspecao Navios de Cruzeiro 2010...
ALBATROS	D	Programa de Inspecao Navios de Cruzeiro 2010...
ALBATROS	D	Programa de Inspecao Navios de Cruzeiro 2011...
AMERA (PRINSENDAM)	D	Programa de Inspecao Navios de Cruzeiro 2010...
ASTOR	D	Programa de Inspecao Navios de Cruzeiro 2011...
ASUKA II	D	Programa de Inspecao Navios de Cruzeiro 2011...
AUSTRAL	D	Programa de Inspecao Navios de Cruzeiro 2013...
AZAMARA QUEST	D	Programa de Inspecao Navios de Cruzeiro 2012...
BALMORAL	D	Programa de Inspecao Navios de Cruzeiro 2017...
BLACK WATCH	D	Programa de Inspecao Navios de Cruzeiro 2010...
BLEU DE FRANCE	D	Programa de Inspecao Navios de Cruzeiro 2010...

Ao utilizar o critério **AND**, conforme a imagem abaixo.

Temos que deixar claro, que o **AND** é um operador lógico igual ao E, da lógica computacional.

Neste caso selecionamos a pontuacao_risco > '1000'.

Lembrando que, o AND deve trazer critérios verdadeiros, pra poder retornar resultado final.

Se um dos dois, não for verdadeiro, o resultado não aparece.

Deixarei claro, que se pode usar quantos **AND** for necessário, porém, desde que todos os critérios sejam verdadeiros.

Se, na pontuacao_risco > '1000' tivesse como '3000', possivelmente não traria nenhuma busca, uma vez que um dos critérios não é verdadeiro.

```

1 •  SELECT NOME_NAVIO, CLASSIFICACAO_RISCO, PONTUACAO_RISCO, TEMPORADA
2   FROM cap02.tb_navios
3   WHERE CLASSIFICACAO_RISCO = 'D' AND PONTUACAO_RISCO > '1000'
4   ORDER BY NOME_NAVIO;

```

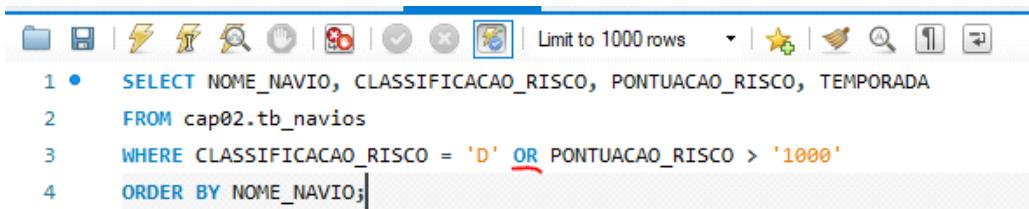
NOME_NAVIO	CLASSIFICACAO_RISCO	PONTUACAO_RISCO	TEMPORADA
ORIENT QUEEN II	D	1065	Programa de Inspecao Navios de Cruzeiro 2013...
SEA EXPLORER	D	1230	Programa de Inspecao Navios de Cruzeiro 2013...
SOVEREIGN (SOBERANO)	D	1035	Programa de Inspecao Navios de Cruzeiro 2012...
STAR PRINCESS	D	1115	Programa de Inspecao Navios de Cruzeiro 2011...

No entanto, temos o operador lógico **OR**, o qual condiciona pelo menos um critério como verdadeiro.

Semanticamente seria

3 **ONDE** CLASSIFICACAO_RISCO FOR 'D' **OU** PONTUACAO_RISCO MAIOR QUE '1000'

Neste caso, em uma situação onde um critério fosse falso, ele traria pelo menos um dos critérios o qual fosse verdadeiro.



```
1 • SELECT NOME_NAVIO, CLASSIFICACAO_RISCO, PONTUACAO_RISCO, TEMPORADA
2 FROM cap02.tb_navios
3 WHERE CLASSIFICACAO_RISCO = 'D' OR PONTUACAO_RISCO > '1000'
4 ORDER BY NOME_NAVIO;
```

Em resumo, o **AND** é utilizado quando se tem certeza absoluta dos dados, ou se quer uma busca mais específica.

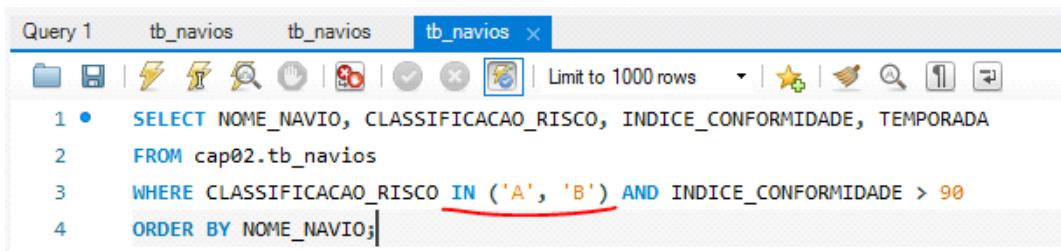
Enquanto o **OR**, permite que você faça uma filtragem mais flexível.

Operador IN.

O operador IN, permite que façamos a inclusão de 2 critérios dentro de um operador, neste caso, na semantica da imagem abaixo, utilizamos o seguinte

ONDE classificacao_risco CONTENDO ('A', 'B') E indice_conformidade maior que 90.

O operador **IN** permite verificar se um valor específico corresponde a algum valor presente em uma lista ou subconsulta, passada como parâmetro a uma cláusula de filtragem **WHERE**.



```
1 • SELECT NOME_NAVIO, CLASSIFICACAO_RISCO, INDICE_CONFORMIDADE, TEMPORADA
2 FROM cap02.tb_navios
3 WHERE CLASSIFICACAO_RISCO IN ('A', 'B') AND INDICE_CONFORMIDADE > 90
4 ORDER BY NOME_NAVIO;
```

A função LIMIT, tem a função de limitar a quantidades de dados retornados.

Porém, ela funciona com base nas definições lógicas que vierem antes dela, e não com base em uma ordem feita pelo próprio SQL. Por isso, vale frisar que LIMIT não é nenhum filtro, e sim um limitador de dados espelhados.

Caso eu tivesse mudado o NOME_NAVIO da coluna abaixo, para indice_conformidade somente, muito provavelmente eu teria 10 itens totalmente diferente.

Query 1 tb_navios tb_navios tb_navios

1 • SELECT NOME_NAVIO, CLASSIFICACAO_RISCO, INDICE_CONFORMIDADE, TEMPORADA
2 FROM cap02.tb_navios
3 WHERE CLASSIFICACAO_RISCO IN ('A', 'B') AND INDICE_CONFORMIDADE > 90
4 ORDER BY INDICE_CONFORMIDADE, NOME_NAVIO
5 LIMIT 10;

Result Grid Filter Rows: Export: Wrap Cell Content: Fetch rows:

	NOME_NAVIO	CLASSIFICACAO_RISCO	INDICE_CONFORMIDADE	TEMPORADA
▶	ADONIA	A	100,00	Programa de Inspecao Navios de Cruzeiro 2012.
	ADONIA	A	100,00	Programa de Inspecao Navios de Cruzeiro 2017.
	ADONIA	A	100,00	Programa de Inspecao Navios de Cruzeiro 2015.
	AIDA CARA	A	100,00	Programa de Inspecao Navios de Cruzeiro 2011.
	AIDA VITA	A	100,00	Programa de Inspecao Navios de Cruzeiro 2012.
	AIDA VITA	A	100,00	Programa de Inspecao Navios de Cruzeiro 2011.
	AIDA VITA	A	100,00	Programa de Inspecao Navios de Cruzeiro 2010.
	AIDA VITA	A	100,00	Programa de Inspecao Navios de Cruzeiro 2011.
	AIDA VITA	A	100,00	Programa de Inspecao Navios de Cruzeiro 2012.
	AI RATROS	A	100,00	Prorrama de Inspecao Navios de Cruzeiro 2013.

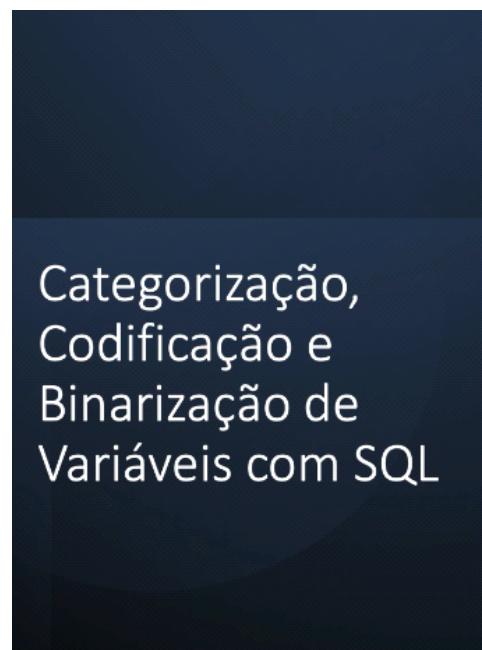
Nota: O operador "IN" geralmente pode nos causar problemas.

5 - Categorização, Codificação e Binarização de Variáveis com SQL

domingo, 13 de novembro de 2022 21:25



O que vamos aprender?



5.1 - Tipos de dados

domingo, 13 de novembro de 2022 22:11

Basicamente trabalharemos com dois tipos de dados no curso em questão, sendo eles "Númericos ou categóricos" e da classificação "Estruturados" conforme imagem abaixo (em sublinhado e negrito).

Tipos de Dados

Temos quatro classificações para os tipos de dados e essa compreensão é importante para todo o ciclo de análise de dados.

- 1- Com relação aos tipos de dados armazenados na memória do computador, eles podem ser numéricos ou categóricos.
- 2- Em relação à fonte os dados podem ser primários ou secundários.
- 3- Em relação à linguagem de programação eles podem ser primitivos ou não primitivos.
- 4- Independente dos itens anteriores, os dados podem ser estruturados, não estruturados ou semi estruturados.



5.2 - Categorização e Codificação

domingo, 13 de novembro de 2022 22:14

Veja a Tabela abaixo:

Podemos notar que foi criada uma categoria de idade, com base na coluna de idade do cliente, onde por exemplo, foi criada uma faixa etária de idade X até a idade Y.

Essa categoria faz sentido aplicar para variáveis numéricas e variáveis categóricas, uma

O Que é Categorização?

Estado do Cliente	Idade do Cliente	Faixa Etária do Cliente
Rio de Janeiro	28	De 21 a 30 Anos
São Paulo	34	De 31 a 40 Anos
Ceará	62	De 61 a 70 Anos
Santa Catarina	46	De 41 a 50 Anos
Amazonas	43	De 41 a 50 Anos
Goiás	67	De 61 a 70 Anos

Esta é uma variável categórica que representa classes ou categorias.

Esta é uma variável numérica (operações matemáticas fazem sentido).

Categorização é converter uma variável para o tipo categórico a fim de obter um novo tipo de informação.

Já na tabela abaixo, podemos ver que na codificação, o cara criou uma coluna chamada Cor_Label, para obter uma representação numérica das cores, onde o Branco=1 e o Verde = 2 por exemplo.

Além disso, não se perde a essência da informação.

Isso (codificação) é muito usado para se trabalhar com Machine Learning, uma vez que não se consegue fazer matemática com o tipo texto.

O Que é Codificação (Encoding)?

Carro	Cor	Cor_Label
Fusca	Branco	1
Passat	Verde	2
Uno	Preto	3
Kadet	Verde	2
Santana	Azul	4
Monza	Preto	3

Esta é uma **variável categórica** que representa classes ou categorias.

Esta é uma **variável categórica** que representa classes ou categorias.

Codificação é transformar uma variável para uma representação numérica, sem perder a essência da informação.

Existem 2 tipos principais de encoding.

Sendo eles: Label Encoding e o One Hot Encoding.

One Hot Encoding

O One-Hot-Encoding, podemos ver na tabela abaixo, que acima de 5 categorias, já não vale a pena usar a Label encoding.

Em virtude disso, utilizamos a One Hot Encoding, criando então uma coluna para cada categoria na coluna "COR", onde o 1 seria um "OK" para a cor contida, e um 0, que seria negativo para a cor.

Exemplo: Cor_Label_Branco, se a coluna "Cor" tiver Branco = 1, se não 0.

O Que é Codificação (Encoding)?

Carro	Cor	Cor_Label_Branco	Cor_Label_Verde	Cor_Label_Preto	Cor_Label_Azul	Cor_Label_Vermelho	Cor_Label_Prata
Fusca	Branco	1	0	0	0	0	0
Passat	Verde	0	1	0	0	0	0
Uno	Preto	0	0	1	0	0	0
Kadet	Verde	0	1	0	0	0	0
Santana	Azul	0	0	0	1	0	0
Monza	Preto	0	0	1	0	0	0
Chevette	Vermelho	0	0	0	0	1	0
Puma	Prata	0	0	0	0	0	1

O exemplo acima chamamos de One-Hot-Encoding, sendo usado para criar variáveis dummy (representações numéricas para cada categoria de uma variável categórica) e normalmente usamos quando a variável categórica tem mais de 5 categorias.

Aplicamos a Codificação às variáveis categóricas e raramente precisamos aplicar a variáveis numéricas.

O objetivo é exatamente converter a variável categórica para uma representação numérica sem perder a essência da informação contida na variável.

5.2.1 - One Hot Encoding

quinta-feira, 17 de novembro de 2022 22:44

Tivemos um exercício durante a formação de analista de dados, o qual era o seguinte: "Aplique One-Hot-Encoding à coluna deg_malig".

Para fazermos uma análise prática do problema, abrimos o excel e fizemos uma rápida passagem sobre o que é o one hot encoding.

Como funciona o One Hot Encoding?

Aplicando **one-hot-encoding**, as categorias se transformaram em colunas (variáveis) onde o número 1 representa o valor afirmativo e o 0 negativo.

Deixarei abaixo alguns exemplos práticos do que é o ONE HOT ENCODING

ID	Color	Shape	blue	green	red	ID	circle	square	triangle
1	green	square	0	1	0	1	0	1	0
2	red	triangle	0	0	1	1	0	0	1
3	red	square	0	0	1	1	0	1	0
4	blue	triangle	1	0	0	1	0	0	1
5	green	circle	0	1	0	1	1	0	0

A screenshot of a Microsoft Word document. At the top, there's a ribbon with 'Arquivo', 'Página Inicial', 'Inserir', and 'Layout da Página'. Below the ribbon, there are standard toolbar icons for file operations like 'Colar' (Paste). The main area shows a table with 6 rows and 4 columns. The columns are labeled 'A', 'B', 'C', and 'D'. The first row contains 'cat1', 'cat2', 'cat3', and an empty cell. Rows 2 through 5 contain binary values: (0, 0, 1), (1, 0, 0), (0, 1, 0), and (1, 0, 0) respectively. Row 6 is entirely empty. A red oval highlights the first four rows of the table.

Para tal, utilizamos o seguinte código:

tb_dados4 tb_dados5

```
1 #[Desafio] Aplique One-Hot-Encoding à coluna deg_malig.
2
3 • SELECT DISTINCT deg_malig FROM cap03.tb_dados4;
4
5 • CREATE TABLE cap03.tb_dados5
6 AS
7 SELECT classe, idade, menopausa, tamanho_tumor, posicao_tumor, node_caps,
8 CASE WHEN deg_malig = 1 THEN 1 ELSE 0 END AS deg_malig_cat1,
9 CASE WHEN deg_malig = 2 THEN 1 ELSE 0 END AS deg_malig_cat2,
10 CASE WHEN deg_malig = 3 THEN 1 ELSE 0 END AS deg_malig_cat3,
11 seio, irradiando
12 FROM cap03.tb_dados4;
```

Utilizamos como sempre Create Table, as, select, selecionando todas as colunas separadas por vírgula, de uma maneira respeitosa. Parando somente quando precisavamos utilizar o CASE WHEN Com as colunas que vamos de fato utilizar.

É possível notar que foram criadas 3 colunas onde o END AS trouxe o nome delas. De maneira semântica precisamos utilizar a seguinte string:

"CASE QUANDO (nome da coluna) for igual a (número) ENTÃO 1 SENÃO 0 POR FIM (nomear coluna)"

Foi possível notar que fizemos CASE WHEN 3x, uma para cada coluna. Porém com a variável distinta diferente

Obtivemos então o seguinte resultado:

Result Grid | Filter Rows: Export: Wrap Cell Content: □

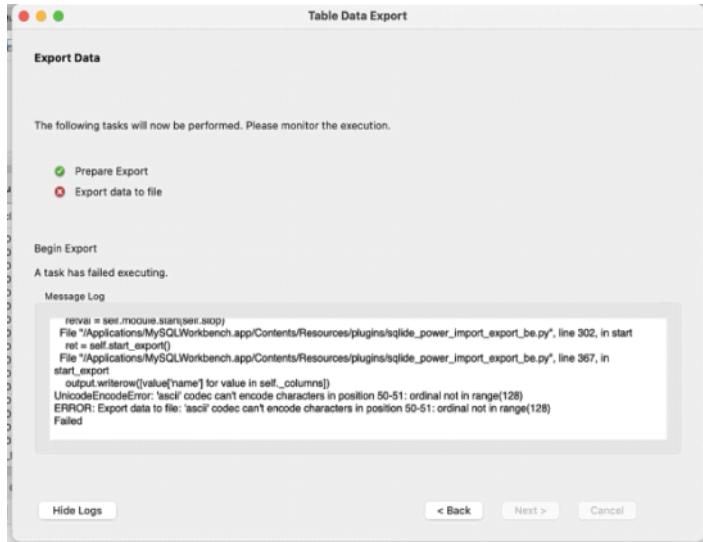
	classe	menopausa	tamanho_tumor	posicao_tumor	node_caps	deg_malig_cat1	deg_malig_cat2	deg_malig_cat3	seio	irradiando
0	30-39	Grande		0-2 no quadrante 1	0	0	0	1	E	0
0	40-49	Medio		0-2 no quadrante 4	0	0	1	0	D	0
0	40-49	Medio		0-2 no quadrante 1	0	0	1	0	E	0
0	60-69	Pequeno		0-2 no quadrante 2	0	0	1	0	D	0
0	40-49	Muito Pequeno		0-2 no quadrante 3	0	0	1	0	D	0

Dúvida: "por que foram utilizado 3x "CASE"?

Aparentemente não há impacto se utilizar um case para 3 WHEN dentro.

Tivemos também o seguinte erro:

Derivado de acentos em coluna, então favor, não colocar acento em nome de coluna.



5.3 - Binarização

segunda-feira, 14 de novembro de 2022 21:53

O que é Binarização?

Binarização, em tese é converter uma variável para sua representação binária, sem perder a essência da informação. Por exemplo na tabela abaixo:

Se na coluna "Paciente com Diabetes" for Sim, então a coluna "Paciente com Diabetes Bin" irá retornar o número 1, caso contrário retornará o número 0.

Em suma, só pode ser 1 ou 0.

O Que é Binarização?

Estado do Paciente	Paciente com Diabetes	Paciente com Diabetes Bin
Rio de Janeiro	Sim	1
São Paulo	Não	0
Ceará	Não	0
Santa Catarina	Não	0
Amazonas	Sim	1
Goiás	Não	0

Esta é uma variável categórica que representa classes ou categorias.

Esta é uma variável categórica que representa classes ou categorias.

Binarização é converter uma variável para sua representação binária sem perder a essência da informação.

5.3.1 - CASE, COUNT, BIN

segunda-feira, 14 de novembro de 2022 21:58

No Exemplo abaixo:

```
SELECT COUNT(*) FROM cap03.tb_dados;
```

Essa é uma query para contar quantas linhas eu tenho na minha base de dados. Vimos que está funcionando tranquilamente.

Queremos converter para 0 ou 1.

O Primeiro passo é contabilizar o total de registro distintos, por exemplo o total de categorias da coluna classe.

Para isso, vamos utilizar o comando DISTINCT, na coluna classe.

Para os rabiscos da imagem abaixo, temos o exemplo do comando DISTINCT, o qual exibiu uma contagem distinta de todo o conteúdo da coluna "classe".

Outro rabisco é até um adendo a minha base de dados que já veio binarizada, o que é estranho. Pois bem, eu desbinarizei a planilha, o argumento então ficou o seguinte:

SELECIONAR DISTINTO

CASO

QUANDO classe for igual a '0' ENTÃO 'no-recurrency-events'

QUANDO classe for igual a '1' ENTÃO 'recurrence-events'

FINALIZAR como classe

DA cap03.tb_dados;

```
1      #binarização da variavel classe (0/1)
2
3 •   SELECT DISTINCT classe FROM cap03.tb_dados;
4
5 •   SELECT DISTINCT
6     CASE
7       WHEN classe = '0' THEN 'no-recurrency-events'
8       WHEN classe = '1' THEN 'recurrence-events'
9     END as classe
10    FROM cap03.tb_dados;
11
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	classe			
	no-recurrency-events			
	recurrence-events			

Na Imagem abaixo, no item nº 1, utilizamos o DISTINCT da coluna IRRADIANDO, para nos dar o retorno que nos dê entendimento das variáveis contidas naquela coluna.

Ou seja, é uma contagem distinta.

Na número 2, após termos descoberto quais variáveis tínhamos na coluna distinct, então fizemos um de para com WHEN and THEN.

Onde as colunas que retornassem um valor negativo teriam o número 0

E assim por diante.

Há uma convenção onde o zero representa algo negativo e o 1 representa algo positivo.

The screenshot shows the MySQL Workbench interface with a query editor titled 'b_dados'. The code is as follows:

```
1 # binarização da variável irradiando (0/1)
2
3 • SELECT DISTINCT IRRADIANDO FROM cap03.tb_dados; 1
4
5 • SELECT
6   CASE
7     WHEN irradiando = 'no' THEN '0'
8     WHEN irradiando = 'yes' THEN '1'
9   END as irradiando
10
11   FROM cap03.tb_dados
12   ORDER BY irradiando
13
```

Annotations: A green box highlights 'SELECT DISTINCT IRRADIANDO' with arrow 1 pointing to it. Another green box highlights the CASE block with arrow 2 pointing to it.

The screenshot shows the MySQL Workbench interface with a results grid titled 'Result Grid'. The data is as follows:

irradiando
0
0
0
0
0
0

Annotations: The table header 'irradiando' has a blue border. The 'Result 12' tab at the bottom is highlighted.

Analisamos um erro na coluna node_caps, pois trata-se de uma coluna com respostas de Sim e Não, porém tivemos alguns caracteres de interrogação (?) dentro desta coluna, tornando mais complicado uma binarização.

The screenshot shows the MySQL Workbench interface with a query editor titled 'b_dados 11'. The code is as follows:

```
1 # binarização da variável node_ caps (0/1)
2
3 • SELECT DISTINCT node_caps FROM cap03.tb_dados;
4
5 • SELECT
6   CASE
7     WHEN node_caps = 'no' THEN 0
8     WHEN node_caps = 'yes' THEN 1
9   END as node_caps
10
11   FROM cap03.tb_dados;
```

Annotations: The entire CASE block is underlined in red. The 'node_caps' in the first WHEN clause is also underlined in red.

Resumindo, quando fizemos a binarização, tivemos "NULL"

	node_caps
1	1
0	0
0	1
1	NULL
0	0

Para tratar isso, iremos codificar, e não mais binarizar.

Precisamos então, incluir o comando ELSE adicionando uma nova variável para a coluna.

Uma nota importante: Para situações como essa, existem muitas tomadas de decisões que precisam ser feitas.

Se for preciso, até acionar o cliente, ou o usuário chave, para tentar entender o que deve ser feito com esse dado.

Neste caso, tomamos a decisão de tudo o que não for "1" ou "0", será 2.

Não importando se a coluna está vazia, com erro de digitação etc.

Veja como o exemplo abaixo:

```
tb_dados x
File | New | Open | Save | Print | Filter Rows | Limit to 1000 rows | 
1  # Para tratar isso, iremos codificar, e não mais binarizar.
2  # Precisamos então, incluir o comando ELSE adicionando uma nova variável para a coluna.
3
4 •  SELECT DISTINCT node_caps FROM cap03.tb_dados;
5
6 •  SELECT
7   CASE
8     WHEN node_caps = 'no' THEN 0
9     WHEN node_caps = 'yes' THEN 1
10    ELSE 2
11  END as node_caps
12  FROM cap03.tb_dados;
13

Result Grid | Filter Rows: [ ] | Export: [ ] | Wrap Cell Content: [ ]
node_caps
1
0
0
1
2
0
```

Exercício --> Faça a categorização da Variável Seio (E/D)

Onde para o left será E, e right será D.

Obtive o resultado abaixo.

```

1      #Categorização da Variavel Seio (E/D)
2
3 •   SELECT DISTINCT seio FROM cap03.tb_dados;
4
5 •   SELECT
6     CASE
7       WHEN seio = 'left' THEN 'E'
8       WHEN seio = 'right' THEN 'D'
9     END as seio
10    FROM cap03.tb_dados
11    ORDER BY seio;

```

Para o exercício 2 --> Fiz o seguinte exemplo abaixo.

O qual estava correto, no entanto, o nosso professor utilizou uma alternativa, que irei contemplar abaixo da minha imagem.

Abaixo meu exemplo:

The screenshot shows the MySQL Workbench interface with a query editor and a result grid.

Query Editor:

```

1      #Categorização da Variavel Tamanho Tumor (6 categorias)
2
3 •   SELECT DISTINCT tamanho_tumor FROM cap03.tb_dados
4     ORDER BY tamanho_tumor;
5
6 •   SELECT DISTINCT
7     CASE
8       WHEN tamanho_tumor IN ('0-4','5-9') THEN 'Muito Pequeno'
9       WHEN tamanho_tumor IN ('10-14','15-19') THEN 'Pequeno'
10      WHEN tamanho_tumor IN ('20-24','25-29') THEN 'Medio'
11      WHEN tamanho_tumor IN ('30-34','35-39') THEN 'Grande'
12      WHEN tamanho_tumor IN ('40-44','45-49') THEN 'Muito Grande'
13      WHEN tamanho_tumor IN ('50-54') THEN 'Tratamento Urgente'
14    END as tamanho_tumor
15    FROM cap03.tb_dados
16    ORDER BY tamanho_tumor;

```

Result Grid:

tamanho_tumor
Grande
Medio
Muito Grande
Muito Pequeno
Pequeno
Tratamento Urgente

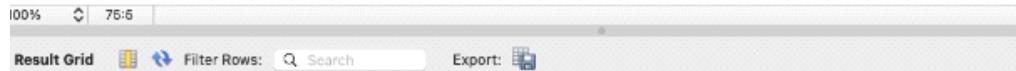
tb_dados 32 Result 33 X

Abaixo exemplo do professor:

```

1  # Categorização da variável tamanho_tumor (6 Categorias)
2 • SELECT DISTINCT tamanho_tumor FROM cap03.TB_DADOS;
3 • SELECT
4     CASE
5         WHEN tamanho_tumor = '0-4' OR tamanho_tumor = '5-9' THEN 'Muito Pequeno'
6         WHEN tamanho_tumor = '10-14' OR tamanho_tumor = '15-19' THEN 'Pequeno'
7         WHEN tamanho_tumor = '20-24' OR tamanho_tumor = '25-29' THEN 'Medio'
8         WHEN tamanho_tumor = '30-34' OR tamanho_tumor = '35-39' THEN 'Grande'
9         WHEN tamanho_tumor = '40-44' OR tamanho_tumor = '45-49' THEN 'Muito Grande'
10        WHEN tamanho_tumor = '50-54' OR tamanho_tumor = '55-59' THEN 'Tratamento Urgente'
11    END as tamanho_tumor
12  FROM cap03.TB_DADOS;

```



Fiz uma pesquisa na internet, mais especificamente pelo Stackoverflow e encontrei o questionamento a seguir:

Tenho as seguintes consultas no MySQL:

1º: Utilizando múltiplos OR

```

SELECT SUM(qtd)
FROM
produtos
WHERE
qtd > 10
and (status = '0' or status = '4' or status = '7')

```

2º: Utilizando IN

```

SELECT SUM(qtd)
FROM
produtos
WHERE
qtd > 10
and status IN (0,4,7)

```

Qual consulta terá melhor performance? Ambos irão me resultar os mesmos dados ou tem diferença?

Obtive a seguinte resposta:

O IN é uma implementação interna e pode ser melhor otimizada. Não deve ter ganho grande, mas é para ser mais rápida para uma lista grande de comparação. Mas em geral a escolha deve ser em função da semântica e não pela performance. O IN procura por um valor em um conjunto de dados, o OR concatena condições e só vale o uso se for em pequena quantidade. Para pequena quantidade a performance não fará diferença. Para grande volume de comparação o OR é virtualmente inviável. O IN é mais poderoso por poder comparar com um conjunto de dados a ser determinado. Quando não faz diferença qual usar, escolha uma forma e seja sempre consistente.

De <<https://pt.stackoverflow.com/questions/218462/utilizar-in-ou-m%C3%BAtiplos-or-qual-possui-melhor-performance>>

6 - Concat, Criando Tabela e corrigindo exercícios

quarta-feira, 16 de novembro de 2022 23:55

Com a query que fizemos, utilizamos alguns argumentos para criar uma nova tabela.

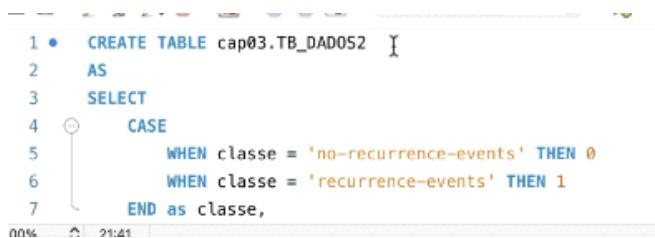
Sendo o seguinte código

```
CREATE TABLE cap03.tb_dados02
```

AS

```
SELECT  
(aqui o resto da query...)
```

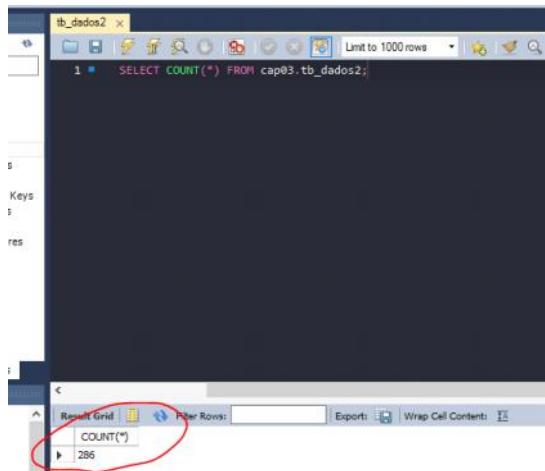
Vide imagem abaixo:



```
1 • CREATE TABLE cap03.TB_DADOS02  
2 AS  
3 SELECT  
4     CASE  
5         WHEN classe = 'no-recurrence-events' THEN 0  
6         WHEN classe = 'recurrence-events' THEN 1  
7     END as classe,
```

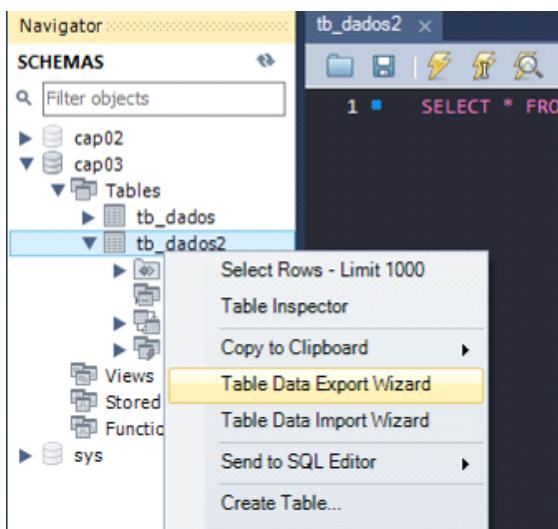
Assim que criamos a tabela, como boa prática, a primeira coisa que fizemos, foi verificar se houve alguma perda de dados.

Inicialmente, tínhamos 286 linhas de dados, mantivemos as 286



Agora, exportando nossa base de dados manipulada em formato .csv:

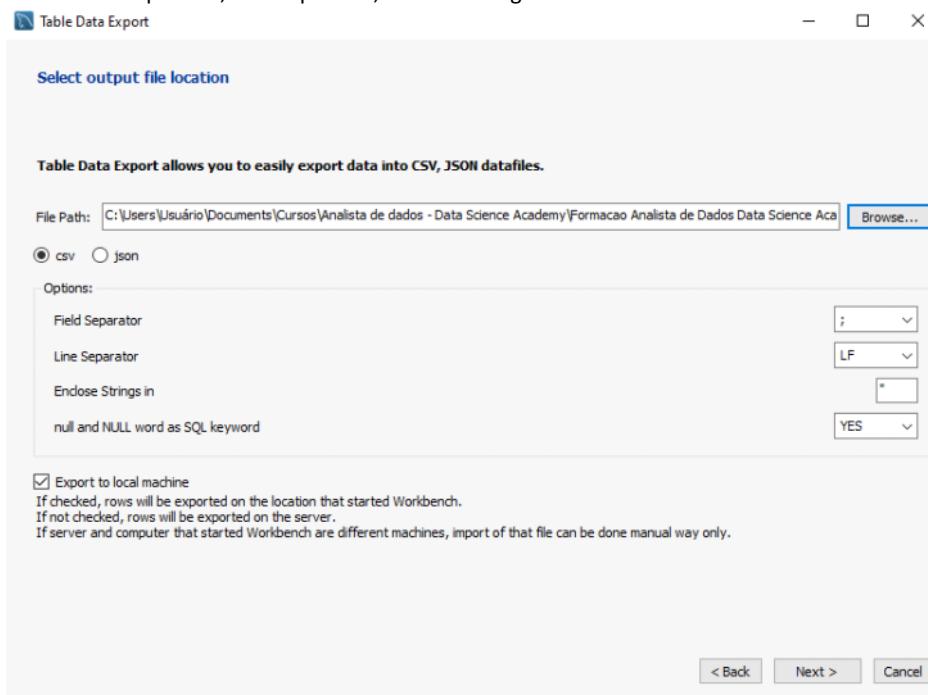
1 - Clicamos com o botão direito em cima da tb_dados2 que no caso foi a planilha criada.



Faça a verificação de tudo que será exportado e em seguida clique em next.



Então, escolha o local path e já mete o dedo no next após analisar as configurações.
Como Field Separator, Line separator, enclose strings e null and NULL.



No exercício, precisávamos fazer o seguinte trampo:

Aplicar uma label encoding, à variável menopausa.
E até aí beleza,

Eu não estava entendendo um seguinte ponto.

Você pode notar que na linha 5 eu abro o comando SELECT e depois faço uma menção a todas as colunas respectivamente.

Eu entro na linha da variável de menopausa, e utilizei CASE, pois foi aplicado uma label encoding na menopausa.

Após aplicar o label encoding, eu simplesmente quando vou fechar com o END as, preciso mencionar a coluna menopausa que estou fechando, e todas as demais colunas.

Até então, não sei a razão, mas ficou conforme imagem abaixo.

```
1 •   SELECT DISTINCT menopausa FROM cap03.tb_dados2;
2     # Aplique label encoding à variável menopausa.
3
4 •   CREATE TABLE cap03.tb_dados3
5   →SELECT
6     classe, idade,
7     CASE
8       WHEN menopausa = 'premeno' THEN 1
9       WHEN menopausa = 'ge40' THEN 2
10      WHEN menopausa = 'lt40' THEN 3
11    END as menopausa, tamanho_tumor, inv_nodes, node_caps,deg_malig,seio,quadrante,irradiando
12    FROM cap03.tb_dados2;
13
14
```

Segue o jeito que usei abaixo para concatenar duas colunas e já de praxe criar uma tabela:

```
1 •   SELECT * FROM cap03.tb_dados5;
2
3 •   [CREATE TABLE cap03.tb_teste
4   SELECT classe,
5   CONCAT(tamanho_tumor, ' aparecem entre as idades ', menopausa, ' |anos') AS teste_concat,
6   node_caps, posicao_tumor, deg_malig_cat1,deg_malig_cat2,deg_malig_cat3,seio,irradiando
7   FROM cap03.tb_dados5;]
```

7 - Relacionamento entre dados. (JOIN)

quinta-feira, 17 de novembro de 2022 23:47

NOTAS:

O INNER JOIN, SÓ TRAZ OS RESULTADOS, SE HOUVEREM CORRESPONDÊNCIAS ENTRE AS TABELAS (SE NÃO TIVER CORRESPONDÊNCIA, NEM NULLO VAI TRAZER).

Agora, se sabe que não há correspondência, e quer retornar algo no relatório (mesmo com nulo) neste caso pode usar right ou left join.

Verificando a tabela abaixo.

Aparentemente não tem problema, mas olhando com olhar analítico, vemos que ela tem muitas dificuldades.

Tabela: TB_PRODUTOS

Repetição da palavra ELETRODOMÉSTICOS

codigo_produto	nome_produto	valor_produto	categoria
1001	TV	2.000,00	Eletrodomésticos
1002	Sofá	1.200,00	Móveis
1003	Banana	15,00	Alimentos
1004	Liquidificador	130,00	Eletrodomésticos
1005	Geladeira	4.500,0	Eletrodomésticos

SELECT categoria, AVG(valor_produto) FROM TB_PRODUTOS GROUP BY categoria

Precisamos atuar então com relacionamento de tabelas, vide exemplo abaixo:

Tabela: TB_PRODUTOS

Tabela: TB_CATEGORIAS

codigo_produto	nome_produto	valor_produto	id_categoria	id	nome_categoria
1001	TV	2.000,00	1	1	Eletrodomésticos
1002	Sofá	1.200,00	2	2	Móveis
1003	Banana	15,00	3	3	Alimentos
1004	Liquidificador	130,00	1		
1005	Geladeira	4.500,0	1		

Reduzindo a variável da categoria, conseguiremos melhorar o desempenho das nossas cargas de dados. Serão mais performáticas.

Tabela: TB_PRODUTOS

codigo_produto	nome_produto	valor_produto	id_categoria
1001	TV	2.000,00	1
1002	Sofá	1.200,00	2
1003	Banana	15,00	3
1004	Liquidificador	130,00	1
1005	Geladeira	4.500,0	1

Tabela: TB_CATEGORIAS

id	nome_categoria
1	Eletrodomésticos
2	Móveis
3	Alimentos

Seleciona variável (nome_produto)

```
SELECT nome_produto
FROM TB_PRODUTOS P, TB_CATEGORIAS C
WHERE P.id_categoria = C.id
AND C.nome_categoria = 'Eletrodomésticos'
```

Nesta imagem no caso, foi selecionado de duas tabelas TB_PRODUTOS_P e também a TB_CATEGORIAS_C

Join (Junção)

Semanticamente falando essa é a explicação da query acima:

SELECT nome_produto (*não precisa especificar de qual tabela ela virá, desde que só tenha essa string com esse nome*) --> que virá da tabela de produtos

FROM TB_PRODUTOS P, TB_CATEGORIAS C --> A PARTIR das strings Produtos e Categorias

WHERE P.id_categoria = C.id --> id_categoria em P for igual id_categoria em C

AND C.nome_categoria = 'eletrodomesticos'.

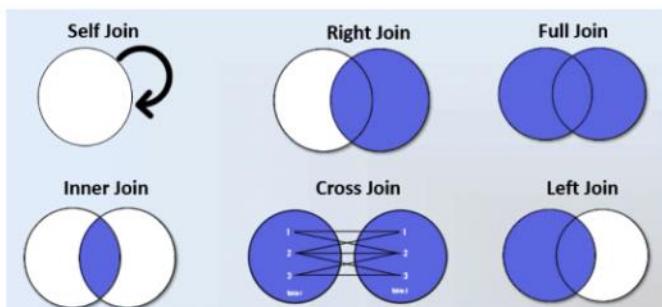
7.1 - Tipos de Junções

sábado, 19 de novembro de 2022 14:56

Na figura abaixo vemos que cada bolinha, cada circulo, eles representam um conjunto.

Pode retornar dados no mesmo conjunto, que estejam num conjunto e não no outro, e assim sucessivamente. Isso representa junção SQL.

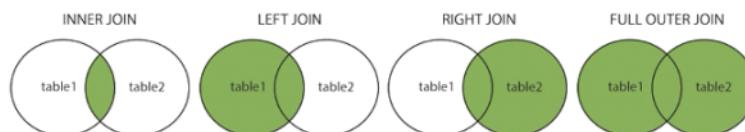
Tipos de Junções



As junções em SQL representam a teoria dos conjuntos!

Aqui estão os diferentes tipos de JOINs em SQL:

- **(INNER) JOIN:** Retorna registros que possuem valores correspondentes em ambas as tabelas.
- **LEFT (OUTER) JOIN:** Retorna todos os registros da tabela da esquerda e os registros correspondentes da tabela da direita.
- **RIGHT (OUTER) JOIN:** Retorna todos os registros da direita e os registros correspondentes da tabela da esquerda.
- **FULL (OUTER) JOIN:** Retorna todos os registros quando há uma correspondência na tabela da esquerda ou da direita.



Nota: A ordem da tabela que você coloca na sua query SQL, tem um impacto gigantesco no retorno dos dados. Devemos tomar cuidado com esse tipo de coisa.

É possível usar a CLAUSULA LEFT JOIN, neste caso, claramente ele usará a tabela da esquerda.

Mas: A ordem das tabelas geram impacto na QUERY SQL.

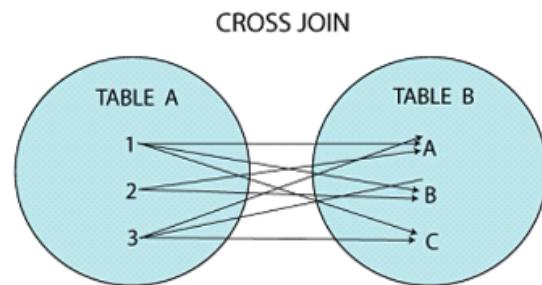
Assim como temos o left, temos também o RIGHT Join, que retorna todos os registros da tabela da direita e os correspondentes da tabela da esquerda.

E o FULL OUTER JOIN: Retorna todos os registros quando há correspondencia em uma tabela da esquerda e da direita (basicamente tudo).

Temos ainda o Cross Join, que não é amplamente usado, porque apresenta muitos problemas de performance, e basicamente retorna associação de todos os registros de ambas as tabelas;

Aqui estão os diferentes tipos de JOINs em SQL:

- **CROSS JOIN**: Retorna a associação de todos os registros de ambas as tabelas.

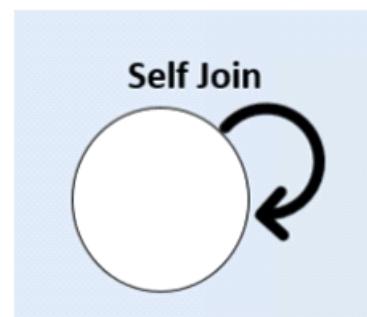


Temos também o Self Join, que retorna os registros da associação de uma tabela com ela mesma.

Aplica recursividade, um conceito muito usado em programação de computadores.

Aqui estão os diferentes tipos de JOINs em SQL:

- **SELF JOIN**: Retorna os registros da associação de uma tabela com ela mesma.

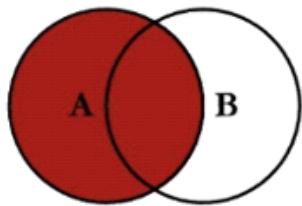


A forma que você constrói uma junção, vai ser completamente modificada de acordo com os filtros que você aplicar. Então nós temos as junções abaixo, e suas pequenas variações.

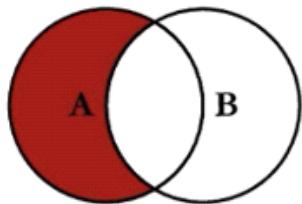
Não precisa decorar nome de junção, basta entender o que você precisa.

Cada tipo de junção tem as suas variações de acordo com os dados que serão retornados na query SQL.

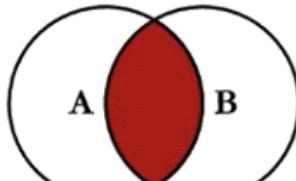
SQL JOINS



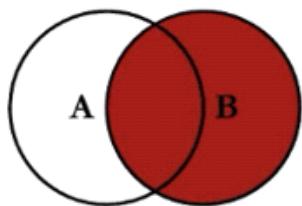
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



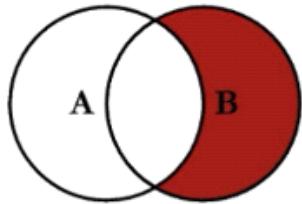
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL.
```



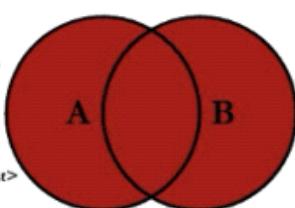
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



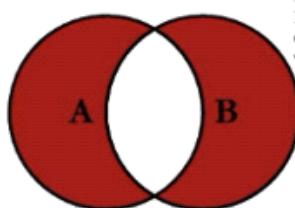
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL.
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL.
```

7.2 - Preparando e Carregando os Dados

terça-feira, 22 de novembro de 2022 23:31

Primeiro precisamos preparar e carregar os dados, contido no script abaixo:

Deixarei algumas anotações em outra coloração.

```
CREATE SCHEMA `cap04`;
```

```
CREATE TABLE [CRIAR A SUA TABELA - Instrução DDL] `cap04`->(nome do schema).`TB_CLIENTES` (nome da tabela) (
`id_cliente`(nome da coluna) INT (nome do tipo de dado) NULL,
`nome_cliente` VARCHAR(50) NULL,
`endereco_cliente` VARCHAR(50) NULL,
`cidade_cliente` VARCHAR(50) NULL,
`estado_cliente` VARCHAR(50) NULL);
```

Info: A query acima, funciona para criar a tabela, nela está contida a argumentação

```
Insert (insert é uma instrução DML de manipulação de dados, ou seja irá alterar dados de alguma forma, serve para inserir na tabela) INTO (onde?) --> `cap04`.`TB_CLIENTES` (quais colunas serão inseridas?) -->(`id_cliente`, `nome_cliente`,
`endereco_cliente`, `cidade_cliente`, `estado_cliente`)
VALUES (1, "Bob Silva", "Rua 67", "Fortaleza", "CE");
```

(Vários dados abaixo são fictícios)

```
INSERT INTO `cap04`.`TB_CLIENTES` (`id_cliente`, `nome_cliente`, `endereco_cliente`, `cidade_cliente`, `estado_cliente`)
VALUES (2, "Ronaldo Azevedo", "Rua 64", "Campinas", "SP");
```

```
INSERT INTO `cap04`.`TB_CLIENTES` (`id_cliente`, `nome_cliente`, `endereco_cliente`, `cidade_cliente`, `estado_cliente`)
VALUES (3, "John Lennon", "Rua 42", "Rio de Janeiro", "RJ");
```

```
INSERT INTO `cap04`.`TB_CLIENTES` (`id_cliente`, `nome_cliente`, `endereco_cliente`, `cidade_cliente`, `estado_cliente`)
VALUES (4, "Billy Joel", "Rua 39", "Campos", "RJ");
```

```
INSERT INTO `cap04`.`TB_CLIENTES` (`id_cliente`, `nome_cliente`, `endereco_cliente`, `cidade_cliente`, `estado_cliente`)
VALUES (5, "Lady Gaga", "Rua 45", "Porto Alegre", "RS");
```

(Novamente será criada uma tabela)

```
CREATE TABLE `cap04`.`TB_PEDIDOS` (
`id_pedido` INT NULL,
`id_cliente` INT NULL,
`id_vendedor` INT NULL,
`data_pedido` DATETIME NULL,
`id_entrega` INT NULL);
```

(perceba como abaixo é feita uma argumentação onde na semântica fica da seguinte maneira abaixo:

INserir NA 'esquema'.nome_da_tabela' ('nome_coluna_1', 'nome_coluna_2', 'nome_coluna3')

VALORES ('valor_coluna_1', 'valor_coluna_2', 'valor_coluna_3')

```
INSERT INTO `cap04`.`TB_PEDIDOS` (`id_pedido`, `id_cliente`, `id_vendedor`, `data_pedido`, `id_entrega`)
VALUES (1001, 1, 5, (pega a hora exata do momento em que os dados foram carregados) -->now(), 23);
```

```
INSERT INTO `cap04`.`TB_PEDIDOS` (`id_pedido`, `id_cliente`, `id_vendedor`, `data_pedido`, `id_entrega`)
VALUES (1002, 1, 7, now(), 24);
```

```
INSERT INTO `cap04`.`TB_PEDIDOS` (`id_pedido`, `id_cliente`, `id_vendedor`, `data_pedido`, `id_entrega`)
VALUES (1003, 2, 5, now(), 23);
```

```
CREATE TABLE `cap04`.`TB_VENDEDOR` (
  `id_vendedor` INT NULL,
  `nome_vendedor` VARCHAR(50) NULL);

INSERT INTO `cap04`.`TB_VENDEDOR`(`id_vendedor`, `nome_vendedor`)
VALUES (1, "Vendedor 1");

INSERT INTO `cap04`.`TB_VENDEDOR`(`id_vendedor`, `nome_vendedor`)
VALUES (2, "Vendedor 2");

INSERT INTO `cap04`.`TB_VENDEDOR`(`id_vendedor`, `nome_vendedor`)
VALUES (3, "Vendedor 3");

INSERT INTO `cap04`.`TB_VENDEDOR`(`id_vendedor`, `nome_vendedor`)
VALUES (4, "Vendedor 4");

INSERT INTO `cap04`.`TB_VENDEDOR`(`id_vendedor`, `nome_vendedor`)
VALUES (5, "Vendedor 5");

INSERT INTO `cap04`.`TB_VENDEDOR`(`id_vendedor`, `nome_vendedor`)
VALUES (6, "Vendedor 6");

INSERT INTO `cap04`.`TB_VENDEDOR`(`id_vendedor`, `nome_vendedor`)
VALUES (7, "Vendedor 7");
```

7.3 - INNER JOIN - Junção de tabela com SQL

terça-feira, 22 de novembro de 2022 23:59

INNER JOIN

A cláusula **INNER JOIN** permite usar um operador de comparação para comparar os valores de colunas provenientes de tabelas associadas. Por meio desta cláusula, os registros de duas tabelas são usados para que sejam gerados os dados relacionados de ambas. Usamos as cláusulas **WHERE** e **FROM** para especificar esse tipo de associação.

No exemplo abaixo, iremos criar as tabelas Cargo e Funcionario, cujas colunas são as seguintes:

	IdCargo	NomeCargo
1	1	Programador Jr.
2	2	Web Designer Pl.
3	3	Programador Pl.
4	4	DBA Jr.
5	5	Programador Sr.

	IdFuncionario	IdCargo	NomeFuncionario	SalarioFuncionario
1	1	2	Tiririca	2500.00
2	2	1	Zé da Pizza	2250.00
3	3	3	Tiozão do Gás	2750.00
4	4	4	Adalberto do Sacolão	2300.00
5	5	1	Marisa da Horta	2500.00

Repare que temos a coluna **IdCargo** nas duas tabelas, porém, ela possui finalidades distintas: enquanto na tabela **Cargo**, ela é chave primária, na tabela **Funcionario** ela é chave estrangeira.

Assim, a associação entre as tabelas é feita pela coluna **IdCargo** e podemos identificar os cargos existentes e o nome dos funcionários que desempenham cada um deles.

Usamos então a cláusula **INNER JOIN** para obtermos os dados relacionados das duas tabelas, para que sejam retornados todos os cargos ocupados pelos funcionários, bem como todos os funcionários que desempenham algum cargo. Veja como isso é feito no script abaixo:

```
SELECTC.NOME[CARGO],F.NOME[FUNCIONÁRIO],F.SALARIO[Salário]FROMCARGO ASC  
INNERJOIN FUNCIONARIO ASF ON C.IDCARGO = F.IDCARGO
```

Nossa consulta terá o seguinte retorno:

CARGO	FUNCIONÁRIO	SALÁRIO
Web Designer Pl.	Tiririca	2500.00
Programador Jr.	Zé da Pizza	2250.00
Programador Pl.	Tiozão do Gás	2750.00
DBA Jr.	Adalberto do Sacolão	2300.00
Programador Jr.	Marisa da Horta	2500.00

Podemos usar também a cláusula **WHERE** e termos o mesmo resultado. O script ficará assim:

```
SELECTC.NOME[CARGO],F.NOME[FUNCIONÁRIO],F.SALARIO[Salário]FROMCARGO  
ASC, FUNCIONARIO ASF  
WHEREC.IDCARGO = F.IDCARGO
```

Independente da cláusula usada, o relacionamento entre as tabelas será feito, como dito anteriormente, por meio da chave primária da tabela **Cargo** e da chave estrangeira da tabela **Funcionario**.

Note que, no retorno de nossas consultas, o único cargo que não é exibido pra nós é o **Programador Sr.** já que não há nenhum funcionário relacionado a este cargo. Para que exibamos também este cargo usamos a cláusula que será explicada a seguir.

Para executar o Inner Join, podemos fazer de duas maneiras.

A primeira maneira é a seguinte:

```
SELECT p.id_pedido, c.nome_cliente  
FROM cap04.tb_pedidos as P  
Inner Join cap04.tb_clientes as C on P.id_cliente= c.id_cliente
```

SQL Por gentileza, retorno id_pedido da tabela P → (P.id_pedido)

```
# Retornar id do pedido e nome do cliente
# Inner Join

SELECT P.id_pedido, C.nome_cliente
FROM cap04.TB_PEDIDOS as P
INNER JOIN cap04.TB_CLIENTES as C
ON P.id_cliente = C.id_cliente;
```

Também retorna nome_cliente, da tabela C → C.nome_cliente

Antes disso, precisamos ligar as tabelas

Com o ON, você indica a ligação entre as tabelas

O Inner Join, só traz relação de dados entre uma tabela e outra.

A segunda maneira é a seguinte:

Selecionar P.id_pedido (SELECIONAR ID_PEDIDO DA TABELA P DO PATH cap04.tb_pedidos como P)

```
SELECT P.id_pedido, C.nome_cliente
FROM cap04.TB_PEDIDOS as P, cap04.TB_CLIENTES as C
WHERE P.id_cliente = C.id_cliente;
```

Selecionar nome_cliente da tabela C do path cap04.tb_clientes como C

Onde a coluna id_cliente da tabela P = coluna cliente contida na tabela C

Abaixo teremos um desafio:

#retornar id do pedido, nome do cliente e nome do vendedor
#inner join com 3 tabelas

Eu utilizei a seguinte query: (estava errado)

```
SELECT P.id_pedido, C.nome_cliente, V.nome_vendedor
FROM cap04.tb_pedidos AS P,
INNER JOIN cap04.tb_clientes AS C
INNER JOIN cap04.tb_vendedor AS V
ON P.id_cliente = C.id_cliente
ON P.id_vendedor = V.id_vendedor;
```

A query correta seria essa:

Podemos notar que seguimos o select da maneira correta.
Porém, quando fui abrir a cláusula FROM, percebi que não utilizei os parenteses.
O correto era abrir 2 parenteses.
Conforme imagem abaixo:

```
SELECT P.id_pedido, C.nome_cliente, V.nome_vendedor
FROM ((cap04.TB_PEDIDOS as P
INNER JOIN cap04.TB_CLIENTES as C ON P.id_cliente = C.id_cliente)
INNER JOIN cap04.TB_VENDEDOR as V ON P.id_vendedor = V.id_vendedor);
```

Agora tentando com a cláusula WHERE com as explicações abaixo:

```

SELECT p.id_pedido, c.nome_cliente, v.nome_vendedor
FROM cap04.tb_pedidos AS P,
cap04.tb_clientes AS C,
cap04.tb_vendedor AS V
WHERE P.id_cliente = C.id_cliente
AND P.id_vendedor = V.id_vendedor;

```

→ SELECIONAR AS COLUNAS QUE EU QUERO QUE APAREÇAM PARA MIM

→ DA SEGUINTE ORIGEM CAP04.TB_PEDIDOS COMO VARIÁVEL P;

DA SEGUINTE ORIGEM CAP04.TB_CLIENTES COMO VARIÁVEL C;

DA SEGUINTE ORIGEM CAP04.TB_VENDEDOR COMO VARIÁVEL V

→ ONDE coluna id_cliente da variável P é equivalente ou possui os mesmos resultados = coluna id_cliente na variável C.

E coluna id_vendedor que estiver contida na variável P seja equivalente a coluna id_vendedor contida na variável V;

Basicamente é o seguinte fluxo:

1. Eu tenho que buscar os campos que eu quero
2. Coloco as tabelas que preciso
3. E por fim crio a relação entre os dados.

Lembrando que a INNER JOIN trabalha com interseção dos dados.

Os dados que nesse caso podem ser retornados a partir de 3 tabelas.

Se você não coloca condição também é possível retornar dados, porém não é Inner Join, aí é full join.

Veja no exemplo abaixo puxando 3 tabelas utilizando a Cláusula INNER JOIN:

```

SELECT P.id_pedido, C.nome_cliente, V.nome_vendedor
FROM ((cap04.TB_PEDIDOS as P
INNER JOIN cap04.TB_CLIENTES as C ON P.id_cliente = C.id_cliente)
INNER JOIN cap04.TB_VENDEDOR as V ON P.id_vendedor = V.id_vendedor);

```

→ SELECT --> id_pedido e já deixo o nome das tabelas lá.

→ Enderço de onde a tabela vai ser puxada

→ Faço o inner join com a tb_clientes, e aí ligo o id_cliente de cada uma das tabelas
Eu então, faço um novo inner join, um novo inner join, agora de tb_pedidos com o tb_vendedor, e faço a ligação de cada uma das tabelas.

Novamente, é mais fácil fazer utilizando a Cláusula WHERE.

Veja no exemplo abaixo:

```

SELECT P.id_pedido, C.nome_cliente, V.nome_vendedor
FROM cap04.TB_PEDIDOS as P, cap04.TB_CLIENTES as C, cap04.TB_VENDEDOR as V
WHERE P.id_cliente = C.id_cliente
AND P.id_vendedor = V.id_vendedor;

```

→ Seleciono os campos que eu quero

→ Das 3 tabelas

→ Uso cláusula WHERE com duas condições
1 Comparando o id_cliente (P com C)
2 id_vendedor (P com V)

7.3.1 INNER JOIN e outras cláusulas.

sábado, 26 de novembro de 2022 21:47

O Inner Join, como já aprendemos é fazer uma **interseção entre as tabelas**.

Podemos considerar somente o Join (para INNER JOIN)

E também, quando o nome da coluna é o mesmo em ambas as tabelas

Podemos utilizar a cláusula **USING** (nome_coluna);

```
# Inner Join quando o nome da coluna é o mesmo em ambas as tabelas
SELECT P.id_pedido, C.nome_cliente
FROM cap04.TB_PEDIDOS AS P
JOIN cap04.TB_CLIENTES as C
USING (id_cliente);
```

Temo também a seguinte query

Ao analisar, podemos ver então o seguinte:

- 💡 **SELECT** p.id_pedido, C.nome_cliente --> A Cláusula **SELECT** vai sempre exibir os campos que você selecionar, quando você coloca esse "p".id_pedido, por exemplo, quer dizer que para a coluna id_pedido, você irá considerar como nome P, para caso mencione futuramente na junção.
- 💡 **FROM** cap04.tb_pedidos AS P --> Neste caso, esse **FROM** é a origem da tabela, que você mencionou o P, note que Após a cláusula FROM, temos uma cláusula **AS** "P" --> Ou seja: SQL por favor, traga a tb_pedidos como P.
- 💡 **USING** (id_cliente), o que quer dizer: "Usando o Id_cliente para o relacionamento entre as tabelas"
- 💡 **WHERE** C.nome_cliente **LIKE** 'Bob%' --> Onde nome do cliente se parecer com 'Bob%', então se tiver um cliente que comece com BOB, e tenha qualquer coisa depois (esse % indica qualquer coisa DEPOIS da palavra) então irá retornar o Bob.
- 💡 **ORDER BY** P.id_pedido **DESC** --> E quero ordenar por id_pedido em ordem **DESC** (Decrescente)

```
# Inner Join com WHERE e ORDER BY
SELECT P.id_pedido, C.nome_cliente
FROM cap04.TB_PEDIDOS AS P
INNER JOIN cap04.TB_CLIENTES as C
USING (id_cliente)
WHERE C.nome_cliente LIKE 'Bob%'
ORDER BY P.id_pedido DESC;
```

7.4 Left Join e Right Join

sábado, 26 de novembro de 2022 22:47

#Retornar todos os clientes, com o seu pedido associado (usando left Join)

#left join - indica que queremos todos os dados da tabela da esquerda, mesmo sem correspondente na tabela da Direita

```
SELECT C.nome_cliente, p.id_pedido --> exibir nome_cliente (considerando C), id_pedido (considerando P)
FROM cap04.tb_clientes AS C --> Da tabela localizada no cap04.tb_clientes identificada COMO C
LEFT JOIN cap04.tb_pedidos AS P --> Fazendo LEFT JOIN a partir da cap04.tb_pedidos como P
ON C.id_cliente = P.id_cliente ONDE o que estiver contido na coluna id_cliente da tabela C (que eu declarei) terá informação contida na coluna id_cliente da coluna P
```

Desafio: #Retornar a data do pedido, o nome do cliente, todos os vendedores, com ou sem pedido associado e ordenar o resultado pelo nome do cliente.

```
#Retornar a data do pedido, o nome do cliente, todos os vendedores, com ou sem pedido associado e ordenar o resultado pelo nome do cliente.

* SELECT p.data_pedido, c.nome_cliente, v.nome_vendedor; → Aqui vai os 3 campos que você quer
  FROM ((cap04.tb_pedidos AS P) → A partir da tb_pedidos eu faço o JOIN
        JOIN cap04.tb_clientes AS C ON p.id_cliente = c.id_cliente) → Faço o Join entre tb_pedidos e tb_clientes | depois do ON, eu coloco a relação das duas
        RIGHT JOIN cap04.tb_vendedor AS V ON p.id_vendedor = v.id_vendedor) → Eu então faço um RIGHT JOIN com a primeira tabela após o FROM
        ORDER BY C.nome_cliente; → Finalizo Ordenando por cliente.
```

Jamais entregue uma query com erros para um analista de negócio,
Veja que a partir daí, temos então algumas instruções para tratar os dados inicialmente

```
1   #Retornar a data do pedido, o nome do cliente, todos os vendedores, com ou sem pedido associado e ordenar o resultado pelo nome do cliente.
2
3   SELECT
4     CASE
5       WHEN p.data_pedido IS NULL THEN 'Sem Pedido'
6       ELSE p.data_pedido
7     END AS data_pedido,
8
9   CASE
10    WHEN c.nome_cliente IS NULL THEN 'Sem Pedido'
11    ELSE c.nome_cliente
12  END AS nome_cliente, v.nome_vendedor
13
14  FROM ((cap04.tb_pedidos AS P
15        JOIN cap04.tb_clientes AS C ON p.id_cliente = c.id_cliente)
16        RIGHT JOIN cap04.tb_vendedor AS V ON p.id_vendedor = v.id_vendedor)
17  ORDER BY V.nome_vendedor;
```

Para inserir condições quando estamos com NULL, note que utilizamos o SELECT normalmente, com os 3 campos a serem exibidos inicialmente. Porém, antes de cada campo, fizemos algumas condições, como por exemplo CASE WHEN No caso da imagem, CASE QUANDO p.data_pedido FOR NULLO ENTÃO 'Sem Pedido' Se não p.data_pedido FINALIZANDO AS data_pedido,

A mesma coisa se aplica na próxima, e após o END AS, finalizamos os outros campo normalmente

7.5 Full Outer Join (UNION and UNION ALL)

segunda-feira, 28 de novembro de 2022 22:54

O que faz o FULL OUTER JOIN?

Basicamente, ele busca os dados, se estiverem presentes em qualquer uma das duas tabelas.

E também, OUTER JOIN serve para **forçar a exibição de todos os itens cadastrados, mesmo que não exista a correspondência em todas as tabelas.** Existem dois tipos de OUTER JOIN, que são o LEFT JOIN e o RIGHT JOIN.

Uma coisa que devemos saber, é que, o FULL JOIN e a clausula FULL OUTER JOIN, não estão permitidas no padrão ANSI e com isso não são aplicáveis ao SGBD My SQL.

Trazendo o Erro abaixo:

```
75 14:44:11 SELECT C.nome_cliente, P.id_pedido FROM cap04.TB_CLIENTES as C FULL JOIN cap04.TB_PEDIDOS AS P... Error Code: 1064. You have an error in your SQL syntax... 0.00030 sec
1 # Full Outer Join (alguns SGBDs não implementam essa junção)
2 • SELECT C.nome_cliente, P.id_pedido
3 FROM cap04.TB_CLIENTES as C
4 FULL JOIN cap04.TB_PEDIDOS AS P → ERRO
5 ON C.id_cliente = P.id_cliente;
6
```

Para tal, teremos que fazer a boa e velha gambiarra.

Veja bem:

Criamos abaixo uma query trazendo a tabela tb_pedidos para a tabela tb_clientes.

E depois fizemos um RIGHT Join, trazendo a tb_pedidos na tb_clientes basicamente identicas. Porém, utilizamos o UNION antes de iniciar a query da RIGHT OUTER JOIN.

Isso é basicamente uma gambiarra manjada KKKKK. (Mas funciona).

Lembrando que no final é tudo uma única QUERY, porém foi dividida para explicação, por isso o ponto e vírgula vem somente no final da query.

```
1 • → QUERY #1 com o LEFT OUTER JOIN
2   SELECT C.nome_cliente, P.id_pedido
3   FROM cap04.TB_CLIENTES as C
4   LEFT OUTER JOIN cap04.TB_PEDIDOS AS P
5   ON C.id_cliente = P.id_cliente
6 UNION → Junção de QUERY → QUERY #2 com o RIGHT OUTER JOIN
7   SELECT C.nome_cliente, P.id_pedido
8   FROM cap04.TB_CLIENTES as C
9   RIGHT OUTER JOIN cap04.TB_PEDIDOS AS P
10  ON C.id_cliente = P.id_cliente;
```

Além do UNION, existe a clausula UNION ALL, que faz um processo de trazer todas as linhas das colunas, incluindo as duplicadas.

Apenas UNION = Retira duplicidade , executa e retorna somente registro unicos.

UNION ALL = Muitas vezes útil quando quer retornar valores duplicados para fazer algum tipo de validação.

Se quero apenas um resultado da query sem duplicidades, então utilizo apenas UNION.

É um de talhe sutil, mas lembre-se, faz toda a diferença na hora de executar a query.

IMPORTANTE

Para que o UNION funcione, precisamos retornar os mesmos campos nas duas query (RIGHT E LEFT) Não precisa ser exatamente tem que ser o mesmo campo, **mas precisa ser o mesmo número de colunas.**

O tipo de dado nas duas querys precisam ser iguais.

Não podemos por exemplo: Trazer tipos diferentes de dados (VAR CHAR com VAR NUM por exemplo).

A ordem utilizada na primeira query tem que ser a mesma da segunda query.

O UNION além disso não apresenta boa performance em muitos casos. (dependendo do volume de dados).

7.6 Self Join

segunda-feira, 28 de novembro de 2022 23:30

O que é SELF JOIN

O que é a SELF JOIN do MySQL?

Uma SELF JOIN é uma junção que é usada para unir uma tabela consigo mesma . Nas seções anteriores, aprendemos sobre a junção da tabela com as outras tabelas usando diferentes JOINS, como INNER, LEFT, RIGHT e CROSS JOIN. No entanto, existe a necessidade de combinar dados com outros dados na própria tabela.

O SELF JOIN no SQL, como o próprio nome indica, é usado para unir uma tabela a ela mesma. Isso significa que cada linha em uma tabela é unida a si mesma e a todas as outras linhas dessa tabela. No entanto, referenciar a mesma tabela mais de uma vez em uma única consulta resultará em erro. Para evitar isso, são usados aliases SQL SELF JOIN.

Basicamente está aqui em baixo.

Não é muito usada, mas serve.

```
1 # Retornar clientes que sejam da mesma cidade
2 • SELECT A.nome_cliente, A.cidade_cliente
3 FROM cap04.TB_CLIENTES A, cap04.TB_CLIENTES B
4 WHERE A.id_cliente <> B.id_cliente
5 AND A.cidade_cliente = B.cidade_cliente;
```

Exemplo de casos de uso

Suponha que você tenha um banco de dados com as seguintes informações (Veja a consulta completa abaixo)

```
DROP SCHEMA IF EXISTS self;
CRIAR ESQUEMA auto;
USAR a si mesmo;
CREATE TABLE users(
    id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(255),
    email VARCHAR(255),
    payment_id INT,
    subscrição INT
);
INSERT INTO users(first_name, email, payment_id, subscription) VALUES
("Valerie G. Phillip", " ValerieGPhillip@teleworm.us ", 10001, 1), ("Sean
R. Storie", " SeanRStorie@rhyta.com ", 10005, 2), ("Bobby S. Newsome", " BobbySNewsome@armyspy.com ", 100010, 5);
```

7.7 - Cross Join

terça-feira, 29 de novembro de 2022 21:59

A seguir ilustramos a sintaxe do SQL Server CROSS JOIN de duas tabelas, sendo elas T1 e Cross Join com a T2 que é nossa tabela 2. Abaixo exemplo:

```
SELECT select_list  
FROM T1  
CROSS JOIN T2;
```

O que o CROSS JOIN fez exatamente?

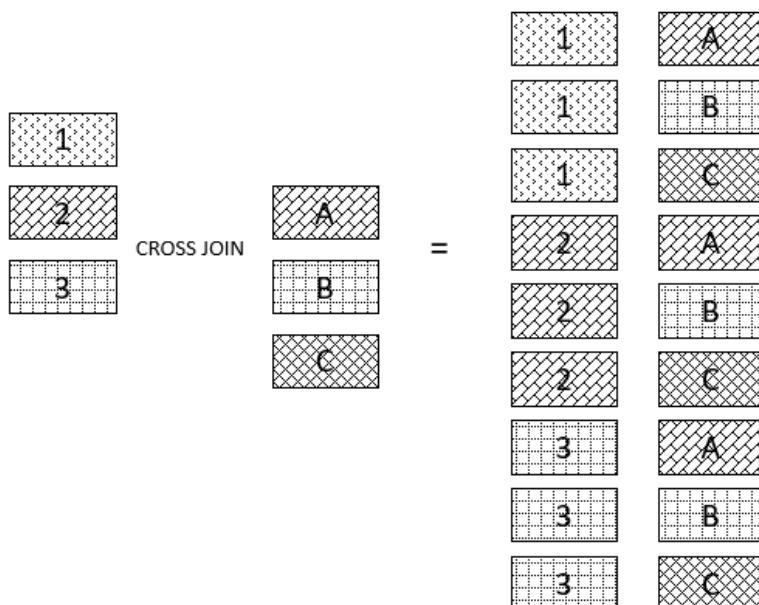
Bom, praticamente a clausula de CROSS JOIN, juntou cada linha da primeira tabela (T1) com cada linha da segunda tabela (T2). Em outras palavras, a junção cruzada retorna um produto cartesiano de linhas de ambas as tabelas.

Ao contrário de [INNER JOIN](#) ou [LEFT JOIN](#), o CROSS JOIN não estabelece um relacionamento entre as tabelas unidas.

Suponha que a tabela T1 contenha três linhas 1, 2 e 3 e a tabela T2 contenha três linhas A, B e C. O CROSS JOIN vai obter uma linha da primeira tabela (T1) e, em seguida, vai criar uma nova linha para cada uma das linhas na segunda tabela (T2).

Em seguida, ele faz o mesmo para a próxima linha na primeira tabela (T1) e assim por diante.

Veja a ilustração abaixo, nela, o CROSS JOIN cria nove linhas no total. Em geral, se a primeira tabela tiver N linhas e a segunda tiver X linhas, a junção cruzada resultará em NX linhas.



Agora veja outro exemplo mais prático abaixo:

```

SELECT
    product_id,
    product_name,
    store_id,
    0 AS quantity
FROM
    production.products
CROSS JOIN sales.stores
ORDER BY
    product_name,
    store_id;

```

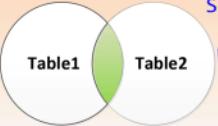
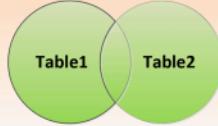
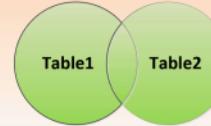
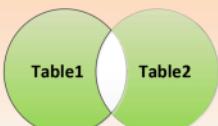
product_id	product_name	store_id	quantity
257	Electra Amsterdam Fashion 3i Ladies' - 2017/2018	1	0
257	Electra Amsterdam Fashion 3i Ladies' - 2017/2018	2	0
257	Electra Amsterdam Fashion 3i Ladies' - 2017/2018	3	0
81	Electra Amsterdam Fashion 7i Ladies' - 2017	1	0
81	Electra Amsterdam Fashion 7i Ladies' - 2017	2	0
81	Electra Amsterdam Fashion 7i Ladies' - 2017	3	0
70	Electra Amsterdam Original 3i - 2015/2017	1	0
70	Electra Amsterdam Original 3i - 2015/2017	2	0
70	Electra Amsterdam Original 3i - 2015/2017	3	0
82	Electra Amsterdam Original 3i Ladies' - 2017	1	0
82	Electra Amsterdam Original 3i Ladies' - 2017	2	0
82	Electra Amsterdam Original 3i Ladies' - 2017	3	0
258	Electra Amsterdam Royal 8i - 2017/2018	1	0
258	Electra Amsterdam Royal 8i - 2017/2018	2	0
258	Electra Amsterdam Royal 8i - 2017/2018	3	0

7.8 - Chart Join types

terça-feira, 29 de novembro de 2022 22:15

TSQL JOIN TYPES

Created by Steve Stedman

 <p>SELECT from two tables</p>	<pre>SELECT * FROM Table1; SELECT * FROM Table2;</pre>	 <p>INNER JOIN</p>	<pre>SELECT * FROM Table1 t1 INNER JOIN Table2 t2 ON t1.fk = t2.id;</pre>
 <p>LEFT OUTER JOIN</p>	<pre>SELECT * FROM Table1 t1 LEFT OUTER JOIN Table2 t2 ON t1.fk = t2.id;</pre>	 <p>RIGHT OUTER JOIN</p>	<pre>SELECT * FROM Table1 t1 RIGHT OUTER JOIN Table2 t2 ON t1.fk = t2.id;</pre>
 <p>SEMI JOIN</p>	<pre>SELECT * FROM Table1 t1 WHERE EXISTS (SELECT 1 FROM Table2 t2 WHERE t1.fk = t2.id);</pre>	 <p>ANTI SEMI JOIN</p>	<pre>SELECT * FROM Table1 t1 WHERE NOT EXISTS (SELECT 1 FROM Table2 t2 WHERE t1.fk = t2.id);</pre>
 <p>LEFT OUTER JOIN with exclusion – replacement for a NOT IN</p>	<pre>SELECT * FROM Table1 t1 LEFT OUTER JOIN Table2 t2 ON t1.fk = t2.id WHERE t2.id IS NULL;</pre>	 <p>RIGHT OUTER JOIN with exclusion – replacement for a NOT IN</p>	<pre>SELECT * FROM Table1 t1 RIGHT OUTER JOIN Table2 t2 ON t1.fk = t2.id WHERE t1.fk IS NULL;</pre>
 <p>FULL OUTER JOIN</p>	<pre>SELECT * FROM Table1 t1 FULL OUTER JOIN Table2 t2 ON t1.fk = t2.id;</pre>	 <p>CROSS JOIN, the Cartesian product</p>	<pre>SELECT * FROM Table1 t1 CROSS JOIN Table2 t2;</pre>
 <p>FULL OUTER JOIN with exclusion</p>	<pre>SELECT * FROM Table1 t1 FULL OUTER JOIN Table2 t2 ON t1.fk = t2.id WHERE t1.fk IS NULL OR t2.id IS NULL;</pre>	 <p>NON-EQUI INNER JOIN</p>	<pre>SELECT * FROM Table1 t1 INNER JOIN Table2 t2 ON t1.fk >= t2.id;</pre>

Created By Steve Stedman <http://SteveStedman.com>
Twitter @SqlEmt <http://linkedin.com/in/stevestedman>

TSQL JOIN TYPES

Created by Steve Stedman

 CROSS APPLY	<pre><code>SELECT * FROM Table1 t1 CROSS APPLY [dbo].[someTVF](t1.fk) AS t;</code></pre>	 OUTER APPLY	<pre><code>SELECT * FROM Table1 t1 OUTER APPLY [dbo].[someTVF](t1.fk) AS t;</code></pre>																																																																												
 Two FULL OUTER JOINS	<pre><code>SELECT * FROM Table1 t1 FULL OUTER JOIN Table2 t2 ON t1.fk = t2.id FULL OUTER JOIN Table3 t3 ON t1.fk_table3 = t3.id;</code></pre>	 Two INNER JOINS	<pre><code>SELECT * FROM Table1 t1 INNER JOIN Table2 t2 ON t1.fk = t2.id INNER JOIN Table3 t3 ON t1.fk_table3 = t3.id;</code></pre>																																																																												
 Two LEFT OUTER JOINS	<pre><code>SELECT * FROM Table1 t1 LEFT OUTER JOIN Table2 t2 ON t1.fk = t2.id LEFT OUTER JOIN Table3 t3 ON t1.fk_table3 = t3.id;</code></pre>	 INNER JOIN and a LEFT OUTER JOIN	<pre><code>SELECT * FROM Table1 t1 INNER JOIN Table2 t2 ON t1.fk = t2.id LEFT OUTER JOIN Table3 t3 ON t1.fk_table3 = t3.id;</code></pre>																																																																												
 EXCEPT	<pre><code>SELECT fk AS id FROM Table1 EXCEPT SELECT ID FROM Table2;</code></pre>	Sample Schema <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="4">Table 1 (People)</th> <th colspan="2">Table 2 (Favorite Colors)</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">id</td> <td style="text-align: center;">Name</td> <td style="text-align: center;">fk</td> <td style="text-align: center;">fk_table3</td> <td style="text-align: center;">id</td> <td style="text-align: center;">FavoriteColor</td> </tr> <tr> <td>1</td> <td>Steve</td> <td>1</td> <td>NULL</td> <td>1</td> <td>red</td> </tr> <tr> <td>2</td> <td>Aaron</td> <td>3</td> <td>NULL</td> <td>2</td> <td>green</td> </tr> <tr> <td>3</td> <td>Mary</td> <td>2</td> <td>NULL</td> <td>3</td> <td>blue</td> </tr> <tr> <td>4</td> <td>Fred</td> <td>1</td> <td>NULL</td> <td>4</td> <td>pink</td> </tr> <tr> <td>5</td> <td>Anne</td> <td>5</td> <td>NULL</td> <td>5</td> <td>purple</td> </tr> <tr> <td>6</td> <td>Beth</td> <td>8</td> <td>1</td> <td>6</td> <td>mauve</td> </tr> <tr> <td>7</td> <td>Johnny</td> <td>NULL</td> <td>1</td> <td>7</td> <td>orange</td> </tr> <tr> <td>8</td> <td>Karen</td> <td>NULL</td> <td>2</td> <td>8</td> <td>yellow</td> </tr> <tr> <td>9</td> <td></td> <td></td> <td></td> <td>9</td> <td>indigo</td> </tr> </tbody> </table> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2">Table 3 (Favorite Foods)</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">id</td> <td style="text-align: center;">dataValue</td> </tr> <tr> <td>1</td> <td>Pizza</td> </tr> <tr> <td>2</td> <td>Burger</td> </tr> <tr> <td>3</td> <td>Sushi</td> </tr> </tbody> </table> <p>Note: Column names are very generic to simplify the sample queries. Foreign keys are Table1.fk -> Table2.id Table2.fk_table3 -> Table3.id</p>		Table 1 (People)				Table 2 (Favorite Colors)		id	Name	fk	fk_table3	id	FavoriteColor	1	Steve	1	NULL	1	red	2	Aaron	3	NULL	2	green	3	Mary	2	NULL	3	blue	4	Fred	1	NULL	4	pink	5	Anne	5	NULL	5	purple	6	Beth	8	1	6	mauve	7	Johnny	NULL	1	7	orange	8	Karen	NULL	2	8	yellow	9				9	indigo	Table 3 (Favorite Foods)		id	dataValue	1	Pizza	2	Burger	3	Sushi
Table 1 (People)				Table 2 (Favorite Colors)																																																																											
id	Name	fk	fk_table3	id	FavoriteColor																																																																										
1	Steve	1	NULL	1	red																																																																										
2	Aaron	3	NULL	2	green																																																																										
3	Mary	2	NULL	3	blue																																																																										
4	Fred	1	NULL	4	pink																																																																										
5	Anne	5	NULL	5	purple																																																																										
6	Beth	8	1	6	mauve																																																																										
7	Johnny	NULL	1	7	orange																																																																										
8	Karen	NULL	2	8	yellow																																																																										
9				9	indigo																																																																										
Table 3 (Favorite Foods)																																																																															
id	dataValue																																																																														
1	Pizza																																																																														
2	Burger																																																																														
3	Sushi																																																																														

Created By Steve Stedman <http://SteveStedman.com>
 Twitter @SqlEmt <http://linkedin.com/in/stevestedman>

8 - Agregando dados para Análise (Group By)

sexta-feira, 2 de dezembro de 2022 00:03

Vamos começar com a cláusula GROUP BY.

Atente-se a explicação abaixo:

```
1 - SELECT COUNT(id_cliente), cidade_cliente
2   FROM cap05.tb_clientes
3   GROUP BY cidade_cliente;
```

Vemos que SELECT COUNT (ID_CLIENTE) retornou o número contado da coluna (id_cliente). Neste caso, 3 clientes localizados em Fortaleza, 1 cliente localizado em campinas e etc.
Porém, isso só foi possível porque utilizamos a Cláusula GROUP BY (nome da coluna). E ai, ele retornou o número de clientes por cidade, conforme exemplo dado.

COUNT(id_cliente)	cidade_cliente
3	Fortaleza
1	Campinas
1	Rio de Janeiro
1	Campos
1	Porto Alegre
1	Natal
2	Ubatuba

Veja só que lógica:

Inclusive, isso é bem comum quando um analista de negócio solicita por exemplo, querer ver quantos clientes por cidade fizeram compras.

```
1 - SELECT *
2   FROM cap05.tb_clientes;
```

É possível ver, que é possível que cidade se repita. Geralmente, quando essa possibilidade existe, podemos então agrupar pelo nome que se repete, mencionando o nome da coluna neste caso seria GROUP BY cidade_cliente;

id_cliente	nome_cliente	endereco_cliente	cidade_cliente	estado_cliente
1	Bob Silva	Rua 67	Fortaleza	CE
2	Ronaldo Azevedo	Rua 64	Campinas	SP
3	John Lennon	Rua 42	Rio de Janeiro	RJ
4	Billy Joel	Rua 39	Campos	RJ
5	Lady Gaga	Rua 45	Porto Alegre	RS
6	Zico Nunes	Rua 67	Fortaleza	CE
7	Maria Aparecida	Rua 61	Natal	RN
8	Elton John	Rua 22	Ubatuba	SP
9	Dario Maravilha	Rua 14	Ubatuba	SP
10	Lebron James	Rua 29	Fortaleza	CE

Outras considerações:

```

1 * SELECT COUNT(id_cliente) AS Contagem, cidade_cliente
2   FROM cap05.tb_clientes
3   GROUP BY cidade_cliente
4   ORDER BY contagem DESC;
5

```

Fizemos o GROUP BY para agrupar por cidade_cliente
Optamos por ordenar de maneira DESC (decrecente)
Alias, por padrão o ORDER BY é CRESCENTE

Contagem	cidade_cliente
3	Fortaleza
2	Ubatuba
1	Campinas
1	Rio de Janeiro
1	Campos
1	Porto Alegre
1	Natal

Fizemos então, a contagem de id_cliente e utilizamos a clausula AS com o nome que escolhemos atribuir.
É valido para quando queremos entregar uma base de dados com boa visualização para os analistas de negócio.

```

1 * SELECT ROUND(AVG(valor_pedido),2) AS Media, c.id_cliente, cidade_cliente
2   FROM cap05.tb_pedidos P, cap05.tb_clientes C
3   WHERE p.id_cliente = c.id_cliente
4   GROUP BY cidade_cliente
5   ORDER BY Media DESC;

```

ROUND = Arredondar
AVG = Media (Average)
valor_pedido = Nome da coluna que estou chamando
,2 = Quantas casas após a vírgula eu quero manter.
NOTA: ROUND vem primeiro e depois podemos ver que são feitos agrupamentos dentro da linha.

Media	id_cliente	cidade_cliente
1290.00	4	Campos
579.20	7	Natal
520.85	5	Porto Alegre
340.00	3	Rio de Janeiro
250.00	2	Campinas
220.41	1	Fortaleza
138.97	9	Ubatuba

Fizemos a inclusão de uma nova linha para podermos fazer alguns testes.

Utilizamos a seguinte query:

```

1 * INSERT INTO cap05.tb_clientes (id_cliente, nome_cliente, endereco_cliente, cidade_cliente, estado_cliente)
2   VALUES (11, "Michael Jordan", "Rua 21", "Palmas", "TO");

```

Agora, veja essas considerações com as palavras do próprio professor

```

1 * SELECT ROUND(AVG(valor_pedido),2) AS Media, cidade_cliente
2   FROM cap05.tb_pedidos P LEFT JOIN cap05.tb_clientes C ?
3   ON p.id_cliente = c.id_cliente
4   GROUP BY cidade_cliente
5   ORDER BY Media DESC;

```

Media	cidade_cliente
1290.00	Campos
579.20	Natal
520.85	Porto Alegre
340.00	Rio de Janeiro
250.00	Campinas
220.41	Fortaleza
138.97	Ubatuba

Para a query acima, eu devo usar o Right ou Left Join onde ta o Inner?

A tabela de pedidos que está do lado esquerdo, na verdade está completa, não há problema com a tabela de pedidos.

A tabela que não tem correspondência é na verdade a tabela de clientes, ou seja, para pedidos eu tenho sempre um cliente associado, já para cliente as vezes não tenho pedido associado.

Neste caso, não pode ser o left join e sim o right join.

Veja que agora usando a Right Join, onde puxamos a tabela faltando correspondência.

Obtivemos o seguinte resultado:

```

1 *   SELECT ROUND(AVG(valor_pedido),2) AS Media, cidade_cliente
2     FROM cap05.tb_pedidos P RIGHT JOIN cap05.tb_clientes C
3       ON p.id_cliente = c.id_cliente
4     GROUP BY cidade_cliente
5     ORDER BY Media DESC

```

Media	cidade_cliente
1290.00	Campos
579.20	Natal
520.85	Porto Alegre
340.00	Rio de Janeiro
250.00	Campinas
220.41	Fortaleza
138.97	Ubatuba
HULL	Palmas

Trouxemos também a seguinte query:

```
1 •  SELECT
2     CASE
3         WHEN ROUND(AVG(valor_pedido),2) IS NULL THEN '0'
4         ELSE ROUND(AVG(valor_pedido),2)
5     END AS Media, cidade_cliente
6     FROM cap05.tb_pedidos P RIGHT JOIN cap05.tb_clientes C
7     ON p.id_cliente = c.id_cliente
8     GROUP BY cidade_cliente
9     ORDER BY Media DESC
```

The screenshot shows a SQL query in a code editor and its results in a grid. Red annotations highlight specific parts of the code and the resulting data.

Annotations:

- A red bracket groups the CASE statement, and a red circle highlights the value '0' in the WHEN clause.
- A red arrow points from the 'Media' column header in the result grid to the 'Media' alias in the SQL query.
- A red circle highlights the value '0' in the first row of the result grid.

Result Grid:

Media	cidade_cliente
579.20	Natal
520.85	Porto Alegre
340.00	Rio de Janeiro
250.00	Campinas
220.41	Fortaleza
138.97	Ubatuba
1290.00	Campos
0	Palmas

8.1 - Funções de Agrupamento (SUM...)

sábado, 3 de dezembro de 2022 01:00

Segue abaixo, a query utilizada para fazer soma (SUM)

```
1 *   SELECT ROUND(SUM(valor_pedido),0) AS Total, C.cidade_cliente  
2     FROM cap05.tb_pedidos P, cap05.tb_clientes C  
3    WHERE p.id_cliente = c.id_cliente  
4    GROUP BY cidade_cliente;
```

Resultado:

Total	cidade_cliente
1322	Fortaleza
250	Campinas
340	Rio de Janeiro
1290	Campos
417	Ubatuba
1042	Porto Alegre
579	Natal

```
1      #Soma total do valor dos pedidos por cidade e estado, com right Join e Case  
2  
3 *  SELECT  
4     CASE  
5       WHEN FLOOR(SUM(valor_pedido)) IS NULL THEN '0'  
6           ELSE FLOOR(SUM(valor_pedido))  
7       END AS total, c.cidade_cliente, c.estado_cliente  
8     FROM (cap05.tb_pedidos AS P  
9        RIGHT JOIN cap05.tb_clientes AS C ON p.id_cliente = c.id_cliente)  
10    GROUP BY cidade_cliente, estado_cliente  
11    ORDER BY total DESC;  
12  
13
```

total	cidade_cliente	estad
579	Natal	RN
416	Ubatuba	SP
340	Rio de Janeiro	RJ
250	Campinas	SP
1322	Fortaleza	CE
1290	Campos	RJ
1041	Porto Alegre	RS
0	Palmas	TO
0	Santos	SP
0	Osasco	SP
0	Barueri	SP

DESAFIO 01

domingo, 13 de novembro de 2022 21:46

Trabalhando com o mesmo dataset usado neste capítulo, crie instruções SQL que respondam às perguntas abaixo:

1- Quais embarcações possuem pontuação de risco igual a 310?

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is:

```
1 •  SELECT * FROM cap02.tb_navios
2 WHERE PONTUACAO_RISCO = 310
```

The result grid displays two rows of data:

	NOME_NAVIO	MES_ANO_ABERTURA	CLASSIFICACAO_RISCO	INDICE_CONFORMIDADE	PONTUACAO_RISCO	TEMPORADA
▶	HORIZON (PACIFIC DRE	12/2010	C	90,76	310	Programa de Inspecção Navios de
	AIDA CARA	11/2013	C	93,52	310	Programa de Inspecção Navios de

2- Quais embarcações têm classificação de risco A e índice de conformidade maior ou igual a 95%?

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is:

```
1 •  SELECT NOME_NAVIO, CLASSIFICACAO_RISCO, INDICE_CONFO
2 WHERE INDICE_CONFORMIDADE >= 95 AND CLASSIFICACAO_RI
3
```

The result grid displays 14 rows of data:

	NOME_NAVIO	CLASSIFICACAO_RISCO	INDICE_CONFORMIDADE
▶	AZAMARA JOURNEY	A	97,98
	BOUDICCA	A	100,00
	NATIONAL GEOGRAPHIC	A	97,43
	COSTA PACIFICA	A	98,20
	SPLENDOUR OF THE SEA	A	99,40
	SOVEREIGN (SOBERANO)	A	98,65
	SOVEREIGN (SOBERANO)	A	97,60
	REGATTA	A	100,00
	RHAPSODY OF THE SEAS	A	98,20
	MSC PREZIOSA	A	99,19
	SEABOURN QUEST	A	98,85
	VIKING SUN	A	100,00
	MAASDAM	A	99,10

3- Quais embarcações têm classificação de risco C ou D e índice de conformidade menor ou igual a 95%?

tb_navios tb_navios tb_navios x tb_navios tb_navios tb_navios

```

1 •  SELECT NOME_NAVIO, CLASSIFICACAO_RISCO, INDICE_CONFORMIDADE FROM capa
2 WHERE CLASSIFICACAO_RISCO IN ('C','D') AND INDICE_CONFORMIDADE <= 95

```

NOME_NAVIO	CLASSIFICACAO_RISCO	INDICE_CONFORMIDADE
VISION OF THE SEAS	C	86,24
COSTA SERENA	C	90,00
GRAND HOLIDAY	D	89,44
COLUMBUS 2	C	88,09
VISION OF THE SEAS	D	90,72
SEVEN SEAS NAVIGATOR	C	91,69
ASTOR	D	86,60
REGATTA	C	91,66
INSIGNIA	D	82,58
GRAND MISTRAL	C	91,91
BRAEMAR	C	86,54
LE CHAMPLAIN	C	92,80
MSC OPERA	D	89,97

4- Quais embarcações têm classificação de risco A ou pontuação de risco igual a 0?

Incorreto, na verdade é solicitado se possui uma ou outra, e não se possui exclusivamente as duas condições juntas.

tb_navios tb_navios tb_navios tb_navios tb_navios tb_navios

```

1 •  SELECT NOME_NAVIO, PONTUACAO_RISCO, CLASSIFICACAO_RISCO FROM cap02.tb_navios
2 WHERE CLASSIFICACAO_RISCO = 'A' AND PONTUACAO_RISCO = '0'

```

NOME_NAVIO	PONTUACAO_RISCO	CLASSIFICACAO_RISCO
SILVER CLOUD	0	A
CRYSTAL SERENITY	0	A
VIKING JUPITER	0	A
AMERA (PRINSENDAM)	0	A
EMPERSS (IMPERATRIZ)	0	A
BRAEMAR	0	A
MSC MUSICA	0	A
CELEBRITY ECLIPSE	0	A
CRYSTAL SERENITY	0	A
AMERA (PRINSENDAM)	0	A
MARCO POLO	0	A
AIDA CARA	0	A
PACIFIC PRINCESS	0	A

5- [DESAFIO] Quais embarcações foram inspecionadas em Dezembro de 2016?

Incorreto pois pesquisei de uma outra coluna, quando o correto era buscar na coluna temporada.

tb_navios tb_navios tb_navios tb_navios tb_navios tb_navios

```

1 •  SELECT NOME_NAVIO, MES_ANO_ABERTURA, TEMPORADA FROM cap02.tb_navios
2 WHERE MES_ANO_ABERTURA = '12/2016'
3 ORDER BY TEMPORADA

```

NOME_NAVIO	MES_ANO_ABERTURA	TEMPORADA
SOVEREIGN (SOBERANO)	12/2016	Programa de Inspecao Navios de Cruzeiro 2016/2017 - Dezembro 2016
MSC PREZIOSA	12/2016	Programa de Inspecao Navios de Cruzeiro 2016/2017 - Dezembro 2016
COSTA PACIFICA	12/2016	Programa de Inspecao Navios de Cruzeiro 2016/2017 - Dezembro 2016
MSC MUSICA	12/2016	Programa de Inspecao Navios de Cruzeiro 2016/2017 - Dezembro 2016
HAMBURG	12/2016	Programa de Inspecao Navios de Cruzeiro 2016/2017 - Dezembro 2016
MARINA	12/2016	Programa de Inspecao Navios de Cruzeiro 2016/2017 - Dezembro 2016
SILVER SPIRIT	12/2016	Programa de Inspecao Navios de Cruzeiro 2016/2017 - Novembro 2016

O correto seria:

The screenshot shows a MySQL Workbench interface. At the top, there are five tabs labeled 'tb_navios' and one tab labeled 'tb_navios' with a red asterisk, indicating unsaved changes. Below the tabs is a toolbar with various icons for database management. The main area displays a SQL query and its results.

```

1 •  SELECT NOME_NAVIO, MES_ANO_ABERTURA, TEMPORADA FROM cap02.tb_navios
2   WHERE TEMPORADA LIKE '%Dezembro 2016'
3   ORDER BY TEMPORADA

```

Result Grid:

NOME_NAVIO	MES_ANO_ABERTURA	TEMPORADA
SOVEREIGN (SOBERANO)	12/2016	Programa de Inspecao Navios de Cruzeiro 2016/2017 - Dezembro 2016
MSC PREZIOSA	12/2016	Programa de Inspecao Navios de Cruzeiro 2016/2017 - Dezembro 2016
COSTA PACIFICA	12/2016	Programa de Inspecao Navios de Cruzeiro 2016/2017 - Dezembro 2016
MSC MUSICA	12/2016	Programa de Inspecao Navios de Cruzeiro 2016/2017 - Dezembro 2016
COSTA FASCINOSA	01/2017	Programa de Inspecao Navios de Cruzeiro 2016/2017 - Dezembro 2016
HAMBURG	12/2016	Programa de Inspecao Navios de Cruzeiro 2016/2017 - Dezembro 2016
MARINA	12/2016	Programa de Inspecao Navios de Cruzeiro 2016/2017 - Dezembro 2016

Note que estamos usando o comando **LIKE**,

O comando LIKE é um importante recurso da linguagem SQL, para auxiliar na seleção de dados específicos em uma tabela. Sua utilização é bastante útil quando queremos encontrar uma correspondência exata de um determinado termo, como palavras iniciadas ou terminadas com uma letra específica.

Geralmente está acompanhado a alguns sinais.

- Como o sinal de porcentagem "%": utilizado para indicar zero, um ou múltiplos caracteres antes o depois do termo pesquisado;
- Caractere de sublinhado "_" : usado para representar um único caractere antes ou após o termo procurado.
- Palavra-chave ESCAPE: utilizada para que seja possível incluir os caracteres curingas (%) e _) ao realizar uma busca.

Material de Referência

segunda-feira, 7 de novembro de 2022 22:56

W3resource é um site que nos trás boas referências de operadores lógicos, para consulta. Tanto de SQL, como front end e back end.



<https://w3resource.com/>

Exercício 3

terça-feira, 29 de novembro de 2022 22:45

1- Quem são os técnicos (coaches) e atletas das equipes de Handball?

Resposta:

```
1 *   SELECT name, discipline
2     FROM tokyo2021.tb_treinadores
3    WHERE discipline = 'Handball';
```

Result Grid | Filter Rows: Export:

name	discipline
ALEKSEEV Alexey	Handball
AXNER Tomas	Handball
BERGE Christian	Handball
BREIVIK Marit	Handball
CADENAS MONTANES Manuel	Handball
CRUZ Filipe	Handball
de OLIVEIRA Marcus Ricardo	Handball
DUENAS de GALARZA Jorge	Handball
ELEK Gabor	Handball
GILLE Guillaume	Handball
GISLASON Alfred	Handball
HALD Simon	Handball

Quem são os técnicos (coaches) dos atletas da Austrália que receberam medalhas de Ouro?

Query utilizada:

```
SELECT Name, T1.NOC,
T2.GOLD
FROM tokyo2021.tb_treinadores AS T1
RIGHT JOIN tokyo2021.tb_medalhas AS T2
ON T1.NOC = T2.`Team/NOC`
WHERE T1.NOC = 'Australia'
ORDER BY Name;
```

```

1 *   SELECT Name, T1.NOC,
2      T2.GOLD
3      FROM tokyo2021.tb_treinadores AS T1
4      RIGHT JOIN tokyo2021.tb_medalhas AS T2
5      ON T1.NOC = T2.`Team/NOC`|
6      WHERE T1.NOC = 'Australia'
7      ORDER BY Name;

```

Resultado Obtido:

ARNOLD Graham	Australia	17
BATCH Colin	Australia	17
BRONDELLO Sandy	Australia	17
CHAMBERS Cheryl	Australia	17
DAVY Jeremy	Australia	17
DYKSTRA Trisha	Australia	17
FATOVIC Elvis	Australia	17
GOORJIAN Brian	Australia	17
GUSTAVSSON Tony	Australia	17
HARROW Laing	Australia	17
JENNINGS Dee	Australia	17
JONES Nathan	Australia	17
KIRKPATRICK Andrew	Australia	17
MANENTI John	Australia	17
MIHAILOVIC Predrag	Australia	17
MONTICO Loredana	Australia	17
MONTICO Loredana	Australia	17
POTTER Anthony	Australia	17
POWELL Katrina	Australia	17
RILLIE John	Australia	17
STANIFORTH Dave	Australia	17
WALSH Tim	Australia	17

3- Quais países tiveram mulheres conquistando medalhas de Ouro?

Query Utilizada

```

SELECT M.NOC, M.GOLD, G.discipline, G.Female
FROM tokyo2021.tb_medalhas AS M, tokyo2021.tb_genero AS G, tokyo2021.tb_times AS T
WHERE T.discipline = G.discipline
AND M.NOC = T.NOC
AND M.GOLD > 0;

```

Quais atletas da Noruega receberam medalhas de Ouro ou Prata?

```
SELECT a.name, a.noc, m.gold, m.silver
```

```
FROM tokyo2021.tb_atletas AS A, tokyo2021.tb_medalhas AS M  
WHERE a.noc = M.noc  
AND a.noc = 'norway';
```

```
1 *   SELECT a.name, a.noc, m.gold, m.silver  
2     FROM tokyo2021.tb_atletas AS A, tokyo2021.tb_medalhas AS M  
3    WHERE a.noc = M.noc  
4    AND a.noc = 'norway';
```

name	noc	gold	silver
AALERUD Katrine	Norway	4	2
ABELVIK ROED Magnus	Norway	4	2
BERGERUD Torbjørn	Norway	4	2
BJOERNSEN Kristian	Norway	4	2
BLUMMENFELT Kristian	Norway	4	2