



Preditiva.ai

Bancos de Dados Relacionais

Parte I

O que você verá neste módulo?

- ❑ O que é SQL?
- ❑ Introdução ao SQL: Comandos essenciais
- ❑ Funções importantes para a Análise de Dados
- ❑ Utilizando múltiplas tabelas
- ❑ SQL em Big Data
- ❑ Bancos de dados não relacionais: NoSQL



Introdução ao SQL: Comandos Essenciais



Extração de Dados

Introdução ao SQL: Comandos essenciais

Nas nossos experimentos vamos utilizar o famoso conjunto de dados Titanic.



Fonte: <https://aventurasnahistoria.uol.com.br/noticias/reportagem/neste-dia-em-1912-o-rms-titanic-colidia-com-um-iceberg-no-mais-impactante-desastre-do-seculo-20.phtml>

Extração de Dados

Introdução ao SQL: Comandos essenciais



Preditiva.ai

Neste conjunto de dados temos diversas informações sobre os passageiros, conforme o metadados apresentado abaixo:

Campo	Descrição	Tipo Campo	Tipo Variável
PassengerId	Código único de identificação do passageiro	Numérico	Qualitativa Nominal
Survived	Indicador de sobrevivência: 1 - sobreviveu, 0 - não sobreviveu	Numérico	Qualitativa Nominal
Pclass	Classe das acomodações no navio: 1, 2 ou 3	Numérico	Qualitativa Ordinal
Name	Nome do passageiro	Texto	Qualitativa Nominal
Sex	Gênero do passageiro	Texto	Qualitativa Nominal
Age	Idade do passageiro	Numérico	Quantitativa Discreta
SibSp	Número de irmãos ou esposa a bordo	Numérico	Quantitativa Discreta
Parch	Número de pais ou filhos a bordo	Numérico	Quantitativa Discreta
Ticket	Número do ticket	Texto	Qualitativa Nominal
Fare	Valor da tarifa	Numérico	Quantitativa Contínua
Cabin	Código da cabine	Texto	Qualitativa Nominal
Embarked	Porto de embarque: C - Cherbourg, Q - Queenstown, S - Southampton	Texto	Qualitativa Nominal

Extração de Dados

Introdução ao SQL: Comandos essenciais



Para iniciar a exploração de uma tabela, podemos selecionar uma amostra das primeiras observações com todas as variáveis.

SELECT: indica quais variáveis serão selecionadas. Para selecionar todas você pode utilizar o '*'.

```
SELECT TOP 10
*
FROM
titanic
```

TOP n: seleciona apenas as primeiras n observações. Útil nas explorações iniciais.

FROM: indica a origem dos dados.

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
	SMALLINT	BIT	TINYINT	NVARCHAR (100)	NVARCHAR (50)	FLOAT	TINYINT	TINYINT	NVARCHAR (50)	FLOAT	NVARCHAR (50)	NVARCHAR (50)
1	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	725	NULL	S
2	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	712.833	C85	C
3	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.925	NULL	S
4	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	531	C123	S
5	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	805	NULL	S
6	6	0	3	Moran, Mr. James	male	NULL	0	0	330877	84.583	NULL	Q
7	7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	518.625	E46	S
8	8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909	21.075	NULL	S
9	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742	111.333	NULL	S
10	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736	300.708	NULL	C

Extração de Dados

Introdução ao SQL: Comandos essenciais



Podemos perceber que a variável **Cabin** apresenta muitas observações com **NULL**. Essa identificação representa, no linguajar de análise de dados, um *missing value*, ou **informação faltante**.
Vamos refazer a query **filtrando** apenas os registros que não possuam valores faltantes na variável **Cabin**.

```
SELECT TOP 10
*
FROM
titanic
WHERE
Cabin is NOT NULL
```

WHERE: filtra as observações baseado em condições

	PassengerId	Survived	Pc		Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
	SMALLINT	BIT	TINYINT	NVARCHAR (100)	NVARCHAR (50)	FLOAT	TINYINT	TINYINT	NVARCHAR (50)	FLOAT	NVARCHAR (50)	NVARCHAR (50)
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	712.833	C85	C
2	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	531	C123	S
3	7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	518.625	E46	S
4	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4	1	1	PP 9549	167	G6	S
5	12	1	1	Bonnell, Miss. Elizabeth	female	58	0	0	113783	2.655	C103	S
6	22	1	2	Beesley, Mr. Lawrence	male	34	0	0	248698	13	D56	S
7	24	1	1	Sloper, Mr. William Thompson	male	28	0	0	113788	355	A6	S
8	28	0	1	Fortune, Mr. Charles Alexander	male	19	3	2	19950	263	C23 C25 C27	S
9	32	1	1	Spencer, Mrs. William Augustus (Marie Eugenie)	female	NULL	1	0	PC 17569	1.465.208	B78	C
10	53	1	1	Harper, Mrs. Henry Sleeper (Myna Haxtun)	female	49	1	0	PC 17572	767.292	D33	C

Extração de Dados

Introdução ao SQL: Comandos essenciais



Preditiva.ai

Ainda estamos apenas no início, mas para evoluirmos com organização é importante seguir alguns padrões de formatação das consultas.

Esses padrões **facilitam** a **utilização** e a **manutenção** das consultas.

```
SELECT TOP 10
  *
FROM
  titanic
WHERE
  Cabin is NOT NULL
```

Formatação

1. **Palavras-chave:** Sempre em letras **maiúsculas**.
2. **Nome de campos e tabelas:** Sempre em letras **minúsculas**.
3. **Espaços em branco:** Evitar usar em nomes de campos e nomes de tabelas.
4. **Indentação:** Utilize para facilitar a leitura e entendimento, e a manutenção.

Extração de Dados

Introdução ao SQL: Comandos essenciais



A seguir vamos nos aprofundar em algumas outras funcionalidades que podem ser muito úteis no nosso dia a dia de análise de dados.

- **WHERE:** Filtra os registros de acordo com as condições especificadas
- **ORDER BY:** Ordena o resultado
- **Operações Aritméticas:** Realiza cálculos simples de Adição, Subtração, Multiplicação e Divisão

Extração de Dados

Introdução ao SQL: Comandos essenciais



WHERE: FILTRAR as observações

A palavra-chave **WHERE** é muito importante por permitir que eliminemos do resultado da consulta as observações não desejadas. Com isso, já realizamos uma primeira etapa de preparação dos dados para a análise.

Operadores lógicos: Podem ser utilizados em campos numéricos ou texto. Nos campos texto é obedecida a ordem alfabética.

Operador	Exemplo Numérico	Exemplo Texto	Descrição
=	WHERE Age = 18	WHERE Embaked = 'C'	igual a
!=	WHERE Age != 18	WHERE Embaked != 'C'	diferente de
>	WHERE Age > 18	WHERE Embaked > 'C'	maior do que
>=	WHERE Age >= 18	WHERE Embaked >= 'C'	maior ou igual a
<	WHERE Age < 18	WHERE Embaked < 'C'	menor do que
<=	WHERE Age <= 18	WHERE Embaked <= 'C'	menor ou igual a

Extração de Dados

Introdução ao SQL: Comandos essenciais



WHERE: FILTRAR as observações

Em campos texto o **LIKE** é o operador que nos permite criar condições mais sofisticadas. Ou seja, podemos filtrar por parte do texto, seja ele o começo, meio ou fim.

Exemplo	Descrição
WHERE Cabin LIKE 'C%'	Código da cabine começa com 'C'
WHERE Ticket LIKE '%7'	Código do ticket termina com '7'
WHERE Name LIKE '%, Dr. %'	Nome do passageiro contém ', Dr. '

Extração de Dados

Introdução ao SQL: Comandos essenciais



WHERE: FILTRAR as observações

Quando temos mais de um valor, numérico ou texto, que queremos utilizar como filtro, podemos utilizar o operador **IN**.

Exemplo	Descrição
WHERE Embarked IN ('C','Q')	Embarque realizado nos portos Cherbourg ou Queenstown
WHERE Pclass IN (2,3)	Cabines de 2ª ou 3ª classe

Extração de Dados

Introdução ao SQL: Comandos essenciais



WHERE: FILTRAR as observações

Para seleccionar um intervalo numérico, um outro operador importante é o **BETWEEN**. Ao invés de utilizar:

```
WHERE Age >= 18 AND Age <= 25
```

Podemos utilizar:

```
WHERE Age BETWEEN 18 AND 25
```


Extração de Dados

Introdução ao SQL: Comandos essenciais



Preditiva.ai

WHERE: FILTRAR as observações

Podemos utilizar os operadores **AND** e **OR** para combinar diferentes condições e construir um critério de seleção mais complexo:

Idade maior ou igual a 18 do sexo feminino:

```
WHERE Age >= 18 AND sex = 'female'
```

Idade maior ou igual a 18 OU sexo feminino:

```
WHERE Age >= 18 OR sex = 'female'
```

Extração de Dados

Introdução ao SQL: Comandos essenciais



WHERE: FILTRAR as observações

Nas situações em que o critério de exclusão é mais simples do que o de inclusão no resultado, podemos utilizar o operador **NOT**. Ele inverte o efeito lógico dos operadores utilizados.

Exemplo	Descrição
WHERE nome NOT LIKE '%, Dr. %'	Nome NÃO contém ', Dr. '
WHERE Embarked NOT IN ('C','Q')	Embarque NÃO realizado nos portos Cherbourg ou Queenstown
WHERE Age NOT BETWEEN 18 and 25	Idade não está entre 18 e 25

Extração de Dados

Introdução ao SQL: Comandos essenciais



Preditiva.ai

ORDER BY: Ordena o resultado da consulta

Caso desejemos ordenar o resultado da consulta, utilizamos o comando **ORDER BY**.

Também é possível ordenar de forma inversa ou decrescente. Nesse caso, utilizamos o ORDER BY seguido do nome do campo e a opção **DESC**.

SELECT TOP 10

FROM

titanic

ORDER BY

PassengerId DESC

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
	SMALLINT	BIT	TINYINT	NVARCHAR (100)	NVARCHAR (50)	FLOAT	TINYINT	TINYINT	NVARCHAR (50)
1	891	0	3	Dooley, Mr. Patrick	male	32	0	0	370376
2	890	1	1	Behr, Mr. Karl Howell	male	26	0	0	111369
3	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NULL	1	2	W./C. 661
4	888	1	1	Graham, Miss. Margaret Edith	female	19	0	0	112053
5	887	0	2	Montvila, Rev. Juozas	male	27	0	0	211536
6	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39	0	5	382652
7	885	0	3	Sutehall, Mr. Henry Jr	male	25	0	0	SOTON/C
8	884	0	2	Banfield, Mr. Frederick James	male	28	0	0	C.A./SOT
9	883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22	0	0	7552
10	882	0	3	Markun, Mr. Johann	male	33	0	0	349257

Extração de Dados

Introdução ao SQL: Comandos essenciais



Preditiva.ai

Uma forma de criarmos novas variáveis a partir das existentes é utilizando o **CASE WHEN**. Vamos criar uma nova variável que indique o título do passageiro.

```
SELECT TOP 10
  Name,
  CASE WHEN Name LIKE '%, Dr. %' THEN 'Doutor'
        WHEN Name LIKE '%, Master. %' THEN 'Mestre'
        WHEN Name LIKE '%, Mr. %' THEN 'Senhor'
        WHEN Name LIKE '%, Miss. %' THEN 'Senhorita'
        WHEN Name LIKE '%, Mrs. %' THEN 'Senhora'
        ELSE 'Outro'
  END AS Titulo
FROM
  titanic
```



	Name	Titulo
	NVARCHAR (100)	VARCHAR (9)
1	Braund, Mr. Owen Harris	Senhor
2	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	Senhor
3	Heikkinen, Miss. Laina	Senhorita
4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	Senhor
5	Allen, Mr. William Henry	Senhor
6	Moran, Mr. James	Senhor
7	McCarthy, Mr. Timothy J	Senhor
8	Palsson, Master. Gosta Leonard	Mestre
9	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	Senhor
10	Nasser, Mrs. Nicholas (Adele Achem)	Senhor

A condição indicada após a palavra chave **WHEN** pode utilizar qualquer uma das técnicas que vimos anteriormente.

Extração de Dados

Introdução ao SQL: Comandos essenciais



Operações Aritméticas: Realiza cálculos simples de Adição, Subtração, Multiplicação e Divisão

Cálculos simples podem ser realizados na própria query. Para isso, basta utilizarmos os operadores:

- '+' para adição
- '-' para subtração
- '*' para multiplicação
- '/' para divisão

```
SELECT TOP 10
    PassengerId,
    Fare,
    Fare * 4.5
FROM
    titanic
```

	PassengerId	Fare	No Column Name
	SMALLINT	FLOAT	FLOAT
1	1	725	2.900
2	2	712.833	2.851.332
3	3	7.925	31.700
4	4	531	2.124
5	5	805	3.220
6	6	84.583	338.332
7	7	518.625	2.074.500
8	8	21.075	84.300
9	9	111.333	445.332
10	10	300.708	1.202.832

Extração de Dados

Introdução ao SQL: Comandos essenciais



Comando	Descrição	Exemplo
SELECT	Seleciona os campos desejados	SELECT PassengerId, Fare...
TOP	Limita o número de registros no resultado	TOP 10
CASE WHEN	Cria nova variável a partir de condições de outras variáveis	CASE WHEN Name LIKE '%, Dr. %'...
FROM	Indica a origem dos dados	FROM titanic
WHERE	Filtra os resultados	WHERE Age = 18
LIKE	Condição como parte de um texto	WHERE Cabin LIKE 'C%'
IN	Condição como uma lista de itens	WHERE Embarked in ('C','Q')
NOT	Inversão da condição	WHERE Embarked NOT in ('C','Q')
AND	Agrupa condições em que todas devem ser verdadeiras	WHERE Sex = 'female' AND Age > 18
OR	Agrupa condições em que pelo menos uma deve ser verdadeira	WHERE Sex = 'female' OR Age > 18
BETWEEN	Condição como um intervalo	WHERE Age between 18 AND 25
ORDER BY	Ordena os resultados	ORDER BY Age

Extração de Dados

Introdução ao SQL: Comandos essenciais



Preditiva.ai



Hands on

Vamos extrair os dados da tabela Titanic e prepara-los para uma análise.

Roteiro:

Construa uma query para extrair:

1. **Colunas:** PassengerId, Pclass, Sex, Age e Survived.
2. Crie uma **nova variável** chamada Titulo com as categorias: Doutor, Mestre, Senhor(a), Senhorita e Reverendo a partir da variável Name.
3. Filtre apenas os registros não **nulos / não preenchidos** em qualquer uma das variáveis.
4. Ordene pela variável **Age** e verifique que tem alguns passageiros com mais de 100 anos. Como isso foi um **erro de importação**, altere a query criando uma nova variável também com o nome Age e nessa nova variável divida a idade apenas desses passageiros por 10.

Funções importantes para a Análise de Dados



Extração de Dados

Funções importantes para a Análise de Dados



Quando possuímos **um volume muito grande de informações** e temos a nossa disposição apenas um computador comum, com processamento e memória bastante limitados, pode ser interessante realizar parte das análises no **servidor de banco de dados**.

A seguir veremos como podemos realizar algumas das etapas de uma **análise exploratória** utilizando comandos SQL:

1. Tabelas de **frequência absoluta e relativa**
2. **Medidas Resumo**
3. Análise Bidimensional: **Information Value**

Extração de Dados

Funções importantes para a Análise de Dados: COUNT

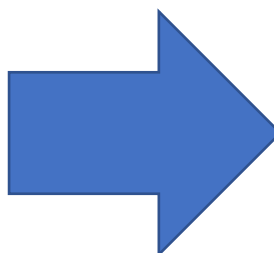


Preditiva.ai

Uma forma de resumir os dados resultantes de uma consulta é utilizar as funções de agregação. Elas possibilitam resumir dados de múltiplas observações.

Suponha que você deseje saber quantos passageiros haviam no Titanic. Para isso utilizaremos a função de agregação **COUNT**:

```
SELECT  
  COUNT (PassengerId)  
FROM  
  titanic
```



	No Column Name
	INT
1	891

Extração de Dados

Funções importantes para a Análise de Dados: COUNT

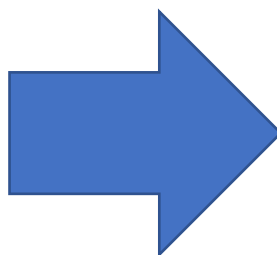


A função **COUNT** pode ser utilizada de duas diferentes formas.

1. Contar o número de registros na tabela: **COUNT(1)**
2. Contar o número de registros com valores não nulos em determinado campo: **COUNT(campo)**

Esta 2ª forma é bastante útil para identificar campos que possuem **NULL**, ou *missing values*.

```
SELECT
    COUNT (1) ,
    COUNT (PassengerId) ,
    COUNT (Age)
FROM
    titanic
```



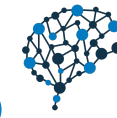
	No Column Name	No Column Name	No Column Name
	INT	INT	INT
1	891	891	714

Com a consulta acima extraímos as seguintes informações:

1. A tabela possui 891 registros
2. O campo PassengerId possui 891 valores válidos (não nulos)
3. O campo Age possui 714 valores válidos, ou seja 177 dos 891 são nulos

Extração de Dados

Funções importantes para a Análise de Dados: **Funções Agregação**



Preditiva.ai

Uma outra informação que pode ser importante nesse conjunto de dados é o valor pago nas passagens. Podemos calcular algumas medidas resumo deste campo utilizando as funções de agregação **MIN**, **MAX**, **SUM**, **AVG** e **STDEV**:

SELECT

COUNT (passengerId) ,

MIN (Fare) ,

MAX (Fare) ,

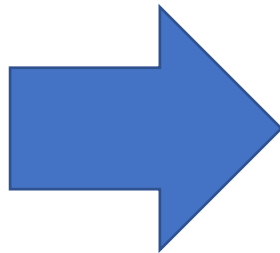
SUM (Fare) ,

AVG (Fare) ,

STDEV (Fare)

FROM

titanic



	No Column Name	No Column Name	No Column Name	No Column Name	No Column Name	No Column Name
	INT	FLOAT	FLOAT	FLOAT	FLOAT	FLOAT
1	891	0	5.123.292	113.745.644	127.660,65544332	411.122,99261142

Apesar do resultado trazer as informações que solicitamos, os nomes das colunas ficaram todos iguais. Vamos aprender como melhorar isso usando **ALIAS**.

Extração de Dados

Funções importantes para a Análise de Dados: **Funções Agregação**



Preditiva.ai

O **ALIAS** serve para renomearmos os campos, ou nesse caso atribuir um nome após a aplicação da função de agregação.

Com isso, a informação fica mais organizada e fácil de se analisar.

SELECT

```
COUNT(passengerId) as n,  
MIN(Fare) as tarifa_min,  
MAX(Fare) as tarifa_max,  
SUM(Fare) as tarifa_total,  
AVG(Fare) as tarifa_media,  
STDEV(Fare) as tarifa_dp
```

FROM

```
titanic
```



	n	tarifa_min	tarifa_max	tarifa_total	tarifa_media	tarifa_dp
	INT	FLOAT	FLOAT	FLOAT	FLOAT	FLOAT
1	891	0	5.123.292	113.745.644	127.660,65544332	411.122,99261142

Podemos perceber que o desvio padrão das tarifas, de aproximadamente 411 mil, é bastante grande quando comparado com a média. Isso pode ser porque estamos calculando essas medidas resumo de todos os passageiros de diferentes classes. Vamos então ver como calcular essas medidas resumo por classe.

Extração de Dados

Funções importantes para a Análise de Dados: **GROUP BY**



O **GROUP BY** serve para realizarmos as agregações agrupadas pelos valores de um ou mais campos. Vamos calcular as mesmas medidas resumo, agora agrupadas e ordenadas por classe:

SELECT

Pclass,

COUNT(passengerId) as n,

MIN(Fare) as tarifa_min,

MAX(Fare) as tarifa_max,

SUM(Fare) as tarifa_total,

AVG(Fare) as tarifa_media,

STDEV(Fare) as tarifa_dp

FROM

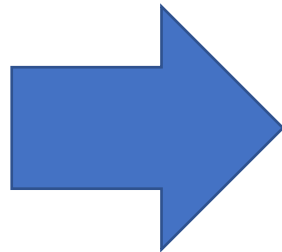
titanic

GROUP BY

Pclass

ORDER BY

Pclass



	Pclass	n	tarifa_min	tarifa_max	tarifa_total	tarifa_media	tarifa_dp
	TINYINT	INT	FLOAT	FLOAT	FLOAT	FLOAT	FLOAT
1	1	216	0	5.123.292	82.946.250	384.010,416666667	766.130,98810115
2	2	184	0	415.792	3.524.792	19.156,47826087	76.227,79571568
3	3	491	0	564.958	27.274.602	55.549,08757637	93.204,37904102

O desvio padrão total que era aproximadamente 411 mil aumentou para 766 mil na 1ª classe e diminuiu para 76 mil e 93 mil na 2ª e 3ª classes, respectivamente.

Extração de Dados

Funções importantes para a Análise de Dados: SUBQUERY



Preditiva.ai

Vamos aproveitar para incluir a frequência relativa utilizando o recurso do SQL chamado de **subquery**. Com ele é possível encadear queries:

A contagem de observações resultantes dessa **subquery** é utilizada como denominador para o cálculo da frequência relativa.

SELECT

```
Pclass,  
COUNT(passengerId) as n,  
CAST(COUNT(passengerId) AS FLOAT) / (SELECT COUNT(1) FROM titanic) as freq_rel,  
MIN(Fare) as tarifa_min,  
MAX(Fare) as tarifa_max,  
SUM(Fare) as tarifa_total,  
AVG(Fare) as tarifa_media,  
STDEV(Fare) as tarifa_dp
```

FROM

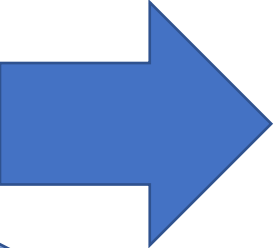
titanic

GROUP BY

Pclass

ORDER BY

Pclass



	Pclass	n	tarifa_min	tarifa_max	tarifa_total	tarifa_media	tarifa_dp
	TINYINT	INT	FLOAT	FLOAT	FLOAT	FLOAT	FLOAT
1	1	216	0	5.123.292	82.946.250	384.010,41666667	766.130,98810115
2	2	184	0	415.792	3.524.792	19.156,47826087	76.227,79571568
3	3	491	0	564.958	27.274.602	55.549,08757637	93.204,37904102

O comando **CAST** serve para converter uma variável para o tipo desejado. Neste caso, queremos converter o número inteiro para um com casas decimais (FLOAT).

Extração de Dados

Funções importantes para a Análise de Dados: SUBQUERY



Preditiva.ai

Um dado interessante é que a tarifa média é maior na 3ª classe do que na 2ª classe. Por que será que isso ocorreu? Vamos incluir o porto de embarque para ver se ele é a resposta.

SELECT

```
Pclass,  
COUNT(passengerId) as n,  
CAST(COUNT(passengerId) AS FLOAT) / (SELECT COUNT(1) FROM titanic) as freq_rel,  
MIN(Fare) as tarifa_min,  
MAX(Fare) as tarifa_max,  
SUM(Fare) as tarifa_total,  
AVG(Fare) as tarifa_media,  
STDEV(Fare) as tarifa_dp
```

FROM

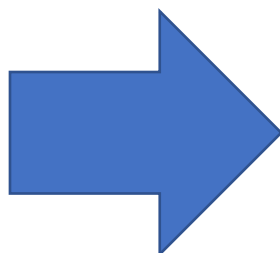
titanic

GROUP BY

Pclass

ORDER BY

Pclass



	Pclass	n	tarifa_min	tarifa_max	tarifa_total	tarifa_media	tarifa_dp
	TINYINT	INT	FLOAT	FLOAT	FLOAT	FLOAT	FLOAT
1	1	216	0	5.123.292	82.946.250	384.010,41666667	766.130,98810115
2	2	184	0	415.792	3.524.792	19.156,47826087	76.227,79571568
3	3	491	0	564.958	27.274.602	55.549,08757637	93.204,37904102

Extração de Dados

Funções importantes para a Análise de Dados: **SUBQUERY**



Preditiva.ai

Incluindo o porto de embarque conseguimos extrair mais informações interessantes:

SELECT

```
Embarked,  
Pclass,  
COUNT(passengerId) as n,  
CAST(COUNT(passengerId) AS FLOAT) / (SELECT COUNT(1) FROM titanic) as freq_rel,  
MIN(Fare) as tarifa_min,  
MAX(Fare) as tarifa_max,  
SUM(Fare) as tarifa_total,  
AVG(Fare) as tarifa_media,  
STDEV(Fare) as tarifa_dp
```

FROM

titanic

GROUP BY

Embarked, Pclass

ORDER BY

Embarked, Pclass

Existem 2 passageiros que não possuem a informação do porto de embarque

	Embarked	Pclass	n	tarifa_min	tarifa_max	tarifa_total	tarifa_media	tarifa_dp
	NVARCHAR(30)	TINYINT	INT	FLOAT	FLOAT	FLOAT	FLOAT	FLOAT
1	NULL	1	2	80	80	160	80	0
2	C	1	85	30	5.123.292	53.605.416	630.651,95294118	1.007.248,090535
3	C	2	17	12	415.792	3.446.107	202.712,17647059	164.295,28141133
4	C	3	66	7.225	223.583	6.620.995	100.318,10606061	63.004,76088974
5	Q	1	2	90	90	180	90	0
6	Q	2	3	1.235	1.235	3.705	1.235	0
7	Q	3	72	155	84.583	1.751.323	24.323,93055556	34.297,01255143
8	S	1	127	0	2.217.792	29.340.494	231.027,51181102	505.846,75825073
9	S	2	164	0	12.525	74.980	457,19512195	1.532,31968936
10	S	3	353	0	564.958	18.902.284	53.547,54674221	102.680,28659409

Extração de Dados

Funções importantes para a Análise de Dados: **SUBQUERY**



Preditiva.ai

Incluindo o porto de embarque conseguimos extrair mais informações interessantes:

SELECT

```
Embarked,  
Pclass,  
COUNT(passengerId) as n,  
CAST(COUNT(passengerId) AS FLOAT) / (SELECT COUNT(1) FROM titanic) as freq_rel,  
MIN(Fare) as tarifa_min,  
MAX(Fare) as tarifa_max,  
SUM(Fare) as tarifa_total,  
AVG(Fare) as tarifa_media,  
STDEV(Fare) as tarifa_dp
```

FROM

titanic

GROUP BY

Embarked, Pclass

ORDER BY

Embarked, Pclass

No porto de Cherbourg a tarifa média por classe parece fazer sentido.

	Embarked	Pclass	n	tarifa_min	tarifa_max	tarifa_total	tarifa_media	tarifa_dp
	NVARCHAR (50)	TINYINT	INT	FLOAT	FLOAT	FLOAT	FLOAT	FLOAT
1	NULL	1	2	80	80	160	80	0
2	C	1	85	30	5.123.292	53.605.416	630.651,95294118	1.007.248,090535
3	C	2	17	12	415.792	3.446.107	202.712,17647059	164.295,28141133
4	C	3	66	7.225	223.583	6.620.995	100.318,10606061	63.004,76088974
5	Q	1	2	90	90	180	90	0
6	Q	2	3	1.235	1.235	3.705	1.235	0
7	Q	3	72	155	84.583	1.751.323	24.323,93055556	34.297,01255143
8	S	1	127	0	2.217.792	29.340.494	231.027,51181102	505.846,75825073
9	S	2	164	0	12.525	74.980	457,19512195	1.532,31968936
10	S	3	353	0	564.958	18.902.284	53.547,54674221	102.680,28659409

Extração de Dados

Funções importantes para a Análise de Dados: **SUBQUERY**



Preditiva.ai

Incluindo o porto de embarque conseguimos extrair mais informações interessantes:

SELECT

Embarked,

Pclass,

COUNT(passengerId) as n,

CAST(COUNT(passengerId) AS FLOAT) / (SELECT COUNT(1) FROM titanic) as freq_rel,

MIN(Fare) as tarifa_min,

MAX(Fare) as tarifa_max,

SUM(Fare) as tarifa_total,

AVG(Fare) as tarifa_media,

STDEV(Fare) as tarifa_dp

FROM

titanic

GROUP BY

Embarked, Pclass

ORDER BY

Embarked, Pclass

No porto de Queenstown os valores da tarifa média estão mais altos na 2ª e 3ª classes, mas o número de passageiros que embarcaram nesta cidade é bastante pequeno.

	Embarked	Pclass	n	tarifa_min	tarifa_max	tarifa_total	tarifa_media	tarifa_dp
	NVARCHAR (50)	TINYINT	INT	FLOAT	FLOAT	FLOAT	FLOAT	FLOAT
1	NULL	1	2	80	80	160	80	0
2	C	1	85	30	5.123.292	53.605.416	630.651,95294118	1.007.248,090535
3	C	2	17	12	415.792	3.446.107	202.712,17647059	164.295,28141133
4	C	3	66	7.225	223.583	6.620.995	100.318,10606061	63.004,76088974
5	Q	1	2	90	90	180	90	0
6	Q	2	3	1.235	1.235	3.705	1.235	0
7	Q	3	72	155	84.583	1.751.323	24.323,93055556	34.297,01255143
8	S	1	127	0	2.217.792	29.340.494	231.027,51181102	505.846,75825073
9	S	2	164	0	12.525	74.980	457,19512195	1.532,31968936
10	S	3	353	0	564.958	18.902.284	53.547,54674221	102.680,28659409

Extração de Dados

Funções importantes para a Análise de Dados: **SUBQUERY**



Preditiva.ai

Incluindo o porto de embarque conseguimos extrair mais informações interessantes:

SELECT

Embarked,

Pclass,

COUNT(passengerId) as n,

CAST(COUNT(passengerId) AS FLOAT) / (SELECT COUNT(1) FROM titanic) as freq_rel,

MIN(Fare) as tarifa_min,

MAX(Fare) as tarifa_max,

SUM(Fare) as tarifa_total,

AVG(Fare) as tarifa_media,

STDEV(Fare) as tarifa_dp

FROM

titanic

GROUP BY

Embarked, Pclass

ORDER BY

Embarked, Pclass

No porto de Southampton é onde está a discrepância! A tarifa média da 2ª classe é muito menor do que a tarifa média da 3ª classe, e é o porto com o maior número de embarques.

	Embarked	Pclass	n	tarifa_min	tarifa_max	tarifa_total	tarifa_media	tarifa_dp
	NVARCHAR (50)	TINYINT	INT	FLOAT	FLOAT	FLOAT	FLOAT	FLOAT
1	NULL	1	2	80	80	160	80	0
2	C	1	85	30	5.123.292	53.605.416	630.651,95294118	1.007.248,090535
3	C	2	17	12	415.792	3.446.107	202.712,17647059	164.295,28141133
4	C	3	66	7.225	223.583	6.620.995	100.318,10606061	63.004,76088974
5	Q	1	2	90	90	180	90	0
6	Q	2	3	1.235	1.235	3.705	1.235	0
7	Q	3	72	155	84.583	1.751.323	24.323,03055556	34.297,01255143
8	S	1	127	0	2.217.792	29.340.494	231.027,51181102	505.846,75825073
9	S	2	164	0	12.525	74.980	457,19512195	1.532,31968936
10	S	3	353	0	564.958	18.902.284	53.547,54674221	102.680,28659409

Extração de Dados

Funções importantes para a Análise de Dados: **HAVING**

Considerando que grupos com menos de 10 observações não produzem uma estimativa robusta, vamos filtrar apenas as combinações de porto de embarque e classe com n maior do que 10 utilizando o **HAVING**.

SELECT

```
Embarked,  
Pclass,  
COUNT(passengerId) as n,  
CAST(COUNT(passengerId) AS FLOAT) / (SELECT COUNT(1) FROM titanic) as freq_rel,  
MIN(Fare) as tarifa_min,  
MAX(Fare) as tarifa_max,  
SUM(Fare) as tarifa_total,  
AVG(Fare) as tarifa_media,  
STDEV(Fare) as tarifa_dp
```

FROM

titanic

GROUP BY

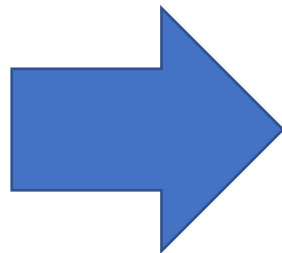
Embarked, Pclass

HAVING

COUNT(passengerId) >= 10

ORDER BY

Embarked, Pclass



	Embarked	Pclass	n	tarifa_min	tarifa_max	tarifa_total	tarifa_media	tarifa_dp
	NVARCHAR (50)	TINYINT	INT	FLOAT	FLOAT	FLOAT	FLOAT	FLOAT
1	C	1	85	30	5.123.292	53.605.416	630.651,95294118	1.007.248,090535
2	C	2	17	12	415.792	3.446.107	202.712,17647059	164.295,28141133
3	C	3	66	7.225	223.583	6.620.995	100.318,10606061	63.004,76088974
4	Q	3	72	155	84.583	1.751.323	24.323,93055556	34.297,01255143
5	S	1	127	0	2.217.792	29.340.494	231.027,51181102	505.846,75825073
6	S	2	164	0	12.525	74.980	457,19512195	1.532,31968936
7	S	3	353	0	564.958	18.902.284	53.547,54674221	102.680,28659409

HAVING: é utilizado para filtrar resultados com condições baseadas em funções de agregação.

Extração de Dados

Funções importantes para a Análise de Dados: IV



Uma técnica de análise bidimensional muito utilizada quando temos uma das variáveis do tipo qualitativa binária é o cálculo **do *Information Value* – IV**. Vamos ver como podemos calcular o IV entre as variáveis **Pclass** e a variável binária **Survived**.

Mas antes, vamos relembrar quais informações necessárias para calcular o IV:

1. Percentual de **Sobreviventes** em cada categoria da variável qualitativa
2. Percentual de **Não Sobreviventes** em cada categoria da variável qualitativa
3. Razão entre Percentual de **Sobreviventes** e **Não Sobreviventes** = Odds
4. Logaritmo Natural do Odds = WOE
5. Diferença entre Percentual de **Sobreviventes** e **Não Sobreviventes** multiplicado pelo WOE
6. Soma dos valores obtidos no item 5

Para facilitar, utilizaremos novamente o conceito de **subquery**.

Extração de Dados

Funções importantes para a Análise de Dados: IV



Vamos iniciar uma consulta com os itens 1 e 2 da nossa lista:

1. Percentual de Sobreviventes em cada categoria da variável qualitativa
2. Percentual de Não Sobreviventes em cada categoria da variável qualitativa

```
SELECT
  Pclass,
  COUNT(1) as n,
  SUM(CAST(Survived AS FLOAT)) / (SELECT COUNT(1) FROM titanic WHERE Survived=1) AS p_sobrev,
  SUM(CAST(1-Survived AS FLOAT)) / (SELECT COUNT(1) FROM titanic WHERE Survived=0) AS p_nsobrev
FROM
  titanic
GROUP BY
  Pclass
```

123 Pclass	123 p_sobrev	123 p_nsobrev
3.00	0.35	0.68
1.00	0.40	0.15
2.00	0.25	0.18

Para calcular o Percentual de Sobreviventes e Não Sobreviventes foi necessário utilizar duas **subqueries** que resultassem na contagem de Sobreviventes e Não Sobreviventes.

Extração de Dados

Funções importantes para a Análise de Dados: IV



Vamos iniciar uma consulta com os itens 1 e 2 da nossa lista:

1. Percentual de Sobreviventes em cada categoria da variável qualitativa
2. Percentual de Não Sobreviventes em cada categoria da variável qualitativa

```
SELECT
  Pclass,
  COUNT(1) as n,
  SUM(CAST(Survived AS FLOAT)) / (SELECT COUNT(1) FROM titanic WHERE Survived=1) AS p_sobrev,
  SUM(CAST(1-Survived AS FLOAT)) / (SELECT COUNT(1) FROM titanic WHERE Survived=0) AS p_nsobrev
FROM
  titanic
GROUP BY
  Pclass
```

123 Pclass 🔒 ⚙️	123 p_sobrev 🔒 ⚙️	123 p_nsobrev 🔒 ⚙️
3.00	0.35	0.68
1.00	0.40	0.15
2.00	0.25	0.18

Além disso, foi necessário também converter o cálculo da variável **Survived**, originalmente do tipo TINYINT, para o tipo FLOAT utilizando o comando **CAST**. Com isso o resultado é exibido corretamente.

Extração de Dados

Funções importantes para a Análise de Dados: IV



Como essa **subquery** será utilizada nos passos seguintes, vamos utilizar o **WITH** para deixar o código da consulta mais organizado. Vamos adicionar também o cálculo do odds, woe e parc_iv, que é o IV parcial. Com essas duas queries chegamos até o item 5, ou seja, basta agora que realizemos a soma da variável **parc_iv** para obter o IV da variável **Pclass**.

```
WITH iv_passo1 AS (  
    SELECT  
        Pclass,  
        COUNT(1) as n,  
        SUM(CAST(Survived AS FLOAT))/(SELECT COUNT(1) FROM titanic WHERE Survived=1) AS p_sobrev,  
        SUM(CAST(1-Survived AS FLOAT))/(SELECT COUNT(1) FROM titanic WHERE Survived=0) AS p_nsobrev  
    FROM  
        titanic  
    GROUP BY  
        Pclass)  
SELECT  
    Pclass,  
    perc_sobrev,  
    perc_naosobrev,  
    perc_sobrev / perc_naosobrev AS odds,  
    LOG(perc_sobrev / perc_naosobrev) AS woe,  
    (perc_sobrev - perc_naosobrev) * LOG(perc_sobrev / perc_naosobrev) AS parc_iv  
FROM  
    iv_passo1
```



123 Pclass 📊 ⬆️	123 p_sobrev 📊 ⬆️	123 p_nsobrev 📊 ⬆️	123 odds 📊 ⬆️	123 woe 📊 ⬆️	123 parc_iv 📊 ⬆️
3.00	0.35	0.68	0.51	-0.67	0.22
1.00	0.40	0.15	2.73	1.00	0.25
2.00	0.25	0.18	1.44	0.36	0.03

Extração de Dados

Funções importantes para a Análise de Dados: IV



Preditiva.ai

Uma outra forma de realizar as queries em etapas é usando as **Tabelas Temporárias**. Vamos utilizar esse recurso para calcular a soma da variável **parc_iv** no final:

```
SELECT
    Pclass,
    COUNT(1) as n,
    SUM(CAST(Survived AS FLOAT))/(SELECT COUNT(1) FROM titanic WHERE Survived=1) AS p_sobrev,
    SUM(CAST(1-Survived AS FLOAT))/(SELECT COUNT(1) FROM titanic WHERE Survived=0) AS p_nsobrev
```

```
INTO #iv_passo1
FROM titanic
GROUP BY
    Pclass
```

Para utilizar novamente uma tabela temporária é necessário excluí-la antes:

```
DROP TABLE #iv_passo1
```

```
SELECT
    Pclass,
    p_sobrev,
    p_nsobrev,
    p_sobrev / p_nsobrev AS odds,
    LOG(p_sobrev / p_nsobrev) AS woe,
    (p_sobrev - p_nsobrev) * LOG(p_sobrev / p_nsobrev) AS parc_iv
```

```
INTO #iv_passo2
FROM #iv_passo1
```

```
SELECT
    SUM(parc_iv)
FROM #iv_passo2
```



	IV
	FLOAT
1	0,50094974

Com esse resultado, podemos considerar que a variável **Pclass** possui um forte poder de separação entre os passageiros que sobreviveram e os que não sobreviveram.

Extração de Dados

Funções importantes para a Análise de Dados



No quadro abaixo temos as principais funções de agregação e um resumo sobre elas.

Função Agregação	Descrição
COUNT	Conta o número de registros ignorando os valores NULL
MIN	Calcula o valor mínimo de um campo numérico
MAX	Calcula o valor máximo de um campo numérico
SUM	Calcula a soma dos valores de um campo numérico
AVG	Calcula a média dos valores de um campo numérico
STDEV	Calcula o desvio padrão dos valores de um campo numérico

Extração de Dados

Funções importantes para a Análise de Dados



Hands on

Vamos realizar algumas análises adicionais na query que construímos anteriormente.

Roteiro:

1. Crie uma nova variável chamada **FaixaEtaria** com intervalos de 10 em 10 anos a partir da variável **Age**. Dica: Use CASE WHEN.
2. Calcule a **taxa de sobrevivência** por: Sex, FaixaEtaria e Titulo.
3. Calcule o **IV** das variáveis Sex, FaixaEtaria e Titulo.

Dica: Ao utilizar tabelas temporárias, inclua seu nome no final para não correr o risco de sobrescrever a tabela de um colega.



Preditiva.ai