



Preditiva.ai

Bancos de Dados Relacionais

Parte III – Tratamento de Dados

O que você verá nesta unidade?

☐ Técnicas de Preparação de Dados (Data Prep)

- ☐ Correção de tipos de dados

- ☐ Transformações em campos:

 - ☐ Texto

 - ☐ Numérico

 - ☐ Data

- ☐ Tratamento de *missing values*

- ☐ Dados duplicados

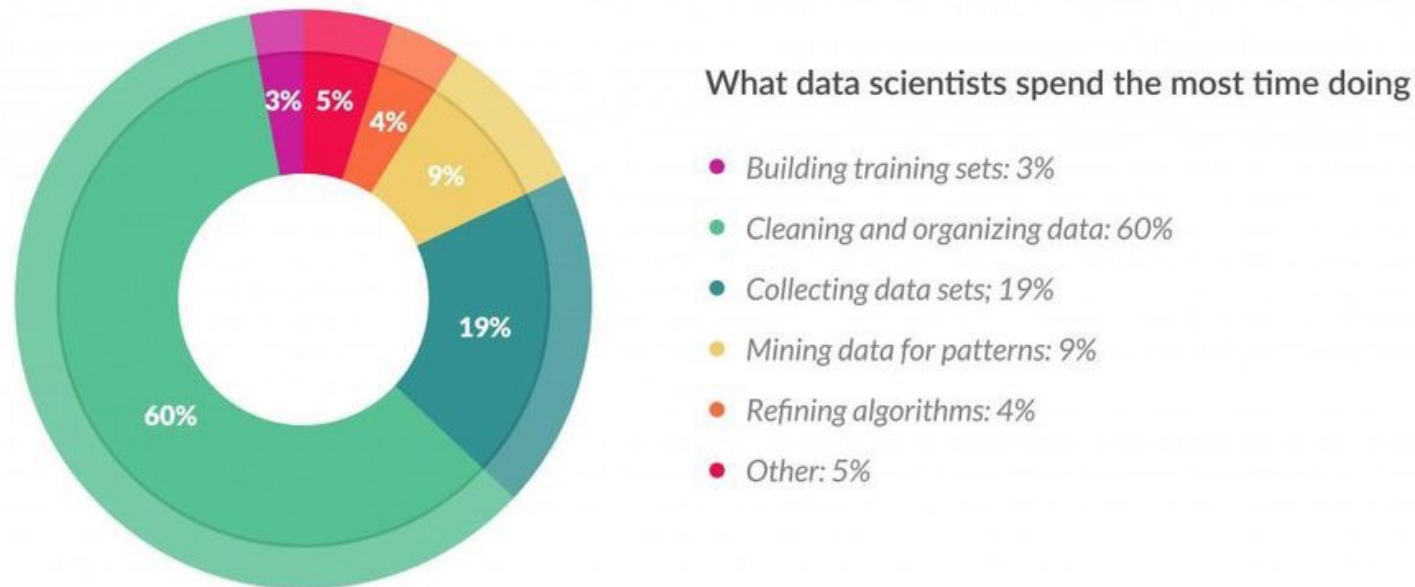


Preparação de Dados

Tempo gasto no Data Prep



A etapa de **Preparação de Dados** é uma das mais trabalhosas e importantes para garantir a qualidade e robustez das análises.



Fonte: <https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says>

A seguir veremos como podemos realizar diversos **tratamentos** e aplicar **transformações** em campos **texto**, **numéricos** e de **data**.

Correção de tipos de dados



Preparação de Dados

Correção de tipos de dados



Na etapa de **Preparação de Dados** é importante garantir que as variáveis sejam extraídas ou criadas estejam com o **tipo adequado**.

Variáveis Numéricas

Tipo de Variável	Valores possíveis	Armazenamento
bit	0, 1, NULL	1 byte
int	-2^{31} a 2^{31}	4 bytes
tinyint	0 a 255	1 byte
smallint	-2^{15} a 2^{15}	2 bytes
bigint	-2^{63} a 2^{63}	8 bytes
decimal(p,s)	-10^{38} a 10^{38}	5 bytes a 17 bytes
float	$-1,79e^{+308}$ a $-2,23e^{-308}$, 0 e $2,23e^{-308}$ a $1,79e^{+308}$	4 bytes a 8 bytes

Preparação de Dados

Correção de tipos de dados



Preditiva.ai

Na etapa de **Preparação de Dados** é importante garantir que as variáveis sejam extraídas ou criadas estejam com o **tipo adequado**.

Variáveis Texto

Tipo de Variável	Valores possíveis	Armazenamento
char(n)	n entre 1 e 8.000	1 a 8.000 bytes
varchar(n max)	n entre 1 e 8.000	1 a 8.000 bytes
nchar(n)	n entre 1 e 4.000	2 a 8.000 bytes
nvarchar(n max)	n entre 1 e 4.000	2 a 8.000 bytes

Preparação de Dados

Correção de tipos de dados



Na etapa de **Preparação de Dados** é importante garantir que as variáveis sejam extraídas ou criadas estejam com o **tipo adequado**.

Variáveis Data

Tipo de Variável	Valores possíveis	Armazenamento
date	0001-01-01 a 9999-12-31	3 bytes
time	00:00:00.0000000 a 23:59:59.9999999	5 bytes
smalldatetime	0000-01-01 a 2079-06-06 e 00:00:00 a 23:59:59	4 bytes
datetime	1753-01-01 a 9999-12-31 e 00:00:00 a 23:59:59.997	8 bytes
datetime2	1753-01-01 a 9999-12-31 e 00:00:00 a 23:59:59.9999999	8 bytes

Preparação de Dados

Correção de tipos de dados



Para converter os tipo de dados podemos usar os comandos **CAST** ou **CONVERT** especificando o tipo de variável que desejamos como resultado:

```
SELECT
  t.Sex,
  COUNT(t.passengerId) AS n,
  (SELECT COUNT(1) FROM titanic) as n_total,
  COUNT(t.passengerId)/(SELECT COUNT(1) FROM titanic) AS freq_rel,
  CAST(COUNT(t.passengerId) AS FLOAT)/(SELECT COUNT(1) FROM titanic) AS freq_rel_float,
  avg(CAST(t.Survived AS INT)) AS taxa_sobrev_int,
  avg(CAST(t.Survived AS FLOAT)) AS taxa_sobrev_float,
  avg(CONVERT(DECIMAL(10,2) , t.Survived)) AS taxa_sobrev_dec
FROM
  titanic t
GROUP BY
  Sex
```



Sex	n	n_total	freq_rel	freq_rel_float	taxa_sobrev_int	taxa_sobrev_float	taxa_sobrev_dec
male	577.00	891.00	0.00	0.65	0.00	0.19	0.19
female	314.00	891.00	0.00	0.35	0.00	0.74	0.74

Transformações em campo Texto



Preparação de Dados

Transformações em campo Texto



Uma das primeiras transformações que fazemos quando começamos a trabalhar com dados em formato texto é converter todas as letras para minúsculo ou maiúsculo. No **SQL** temos a disposição as funções **LOWER** e **UPPER**.

```
SELECT  
  t.Name ,  
  LOWER(t.Name) as nome_minusculas ,  
  UPPER(t.Name) as nome_maiusculas  
FROM  
  titanic t
```



Name	nome_minusculas	nome_maiusculas
Braund, Mr. Owen Harris	braund, mr. owen harris	BRAUND, MR. OWEN HARRIS
Cumings, Mrs. John Bradley (Florence Briggs Thayer)	cumings, mrs. john bradley (florence briggs thayer)	CUMINGS, MRS. JOHN BRADLEY (FLORENCE BRIGGS THAYER)
Heikkinen, Miss. Laina	heikkinen, miss. laina	HEIKKINEN, MISS. LAINA
Futrelle, Mrs. Jacques Heath (Lily May Peel)	futrelle, mrs. jacques heath (lily may peel)	FUTRELLE, MRS. JACQUES HEATH (LILY MAY PEEL)
Allen, Mr. William Henry	allen, mr. william henry	ALLEN, MR. WILLIAM HENRY
Moran, Mr. James	moran, mr. james	MORAN, MR. JAMES
McCarthy, Mr. Timothy J	mccarthy, mr. timothy j	MCCARTHY, MR. TIMOTHY J
Palsson, Master. Gosta Leonard	palsson, master. gosta leonard	PALSSON, MASTER. GOSTA LEONARD

Preparação de Dados

Transformações em campo Texto



Anteriormente criamos a variável **Título** contendo parte da informação contida na variável **Name**, utilizando a função **CASE WHEN**. Uma forma **mais eficiente** é utilizando as funções específicas para tratamento de campos do tipo texto.

Podemos perceber que o título está localizado entre uma vírgula e um ponto final. Logo, o primeiro passo é obter a localização dos sinais de pontuação "," e ".". Para isso vamos utilizar a função **charindex**:

```
SELECT TOP 10
    t.Name,
    CHARINDEX(',', t.Name) pos_virgula,
    CHARINDEX('.', t.Name) pos_ponto
FROM
    titanic
```



Name	pos_virgula	pos_ponto
Braund, Mr. Owen Harris	7.00	11.00
Cumings, Mrs. John Bradley (Florence Briggs Thayer)	8.00	13.00
Heikkinen, Miss. Laina	10.00	16.00
Futrelle, Mrs. Jacques Heath (Lily May Peel)	9.00	14.00
Allen, Mr. William Henry	6.00	10.00
Moran, Mr. James	6.00	10.00
McCarthy, Mr. Timothy J	9.00	13.00
Palsson, Master. Gosta Leonard	8.00	16.00

Na função **charindex** indicamos qual o texto procurado e em qual variável. A função retorna a localização do texto procurado na variável.

Preparação de Dados

Transformações em campo Texto



Para extrair apenas o título que está entre essas duas posições vamos utilizar a função **SUBSTRING**. Nela devemos indicar o **nome da variável** queremos extrair uma parte do texto, a **posição inicial** e o **comprimento**.

```
SELECT TOP 10
    t.Name,
    CHARINDEX(',', t.Name) as pos_virgula,
    CHARINDEX('.', t.Name) as pos_ponto,
    SUBSTRING(t.Name, CHARINDEX(',', t.Name)+2, CHARINDEX('.', t.Name)-CHARINDEX(',', t.Name)-2) as Titulo
FROM
    titanic t
```



	Name	pos_virgula	pos_ponto	Titulo
	NVARCHAR (100)	INT	INT	NVARCHAR (100)
1	Braund, Mr. Owen Harris	7	11	Mr
2	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	8	13	Mrs
3	Heikinen, Miss. Laina	10	16	Miss
4	Futelle, Mrs. Jacques Heath (Lily May Peel)	9	14	Mrs
5	Allen, Mr. William Henry	6	10	Mr
6	Moran, Mr. James	6	10	Mr
7	McCarthy, Mr. Timothy J	9	13	Mr
8	Palsson, Master. Gosta Leonard	8	16	Master
9	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	8	13	Mrs
10	Nasser, Mrs. Nicholas (Adele Achem)	7	12	Mrs

Note que foi necessário realizar alguns ajustes de índice para eliminar o espaço em branco, virgula e ponto final.

Preparação de Dados

Transformações em campo Texto



Para extrair apenas o título que está entre essas duas posições vamos utilizar a função **SUBSTRING**. Nela devemos indicar o **nome da variável** queremos extrair uma parte do texto, a **posição inicial** e o **comprimento**.

```
SELECT TOP 10
  t.Name,
  CHARINDEX(',', t.Name) as pos_virgula,
  CHARINDEX('.', t.Name) as pos_ponto,
  SUBSTRING(t.Name, CHARINDEX(',', t.Name)+2, CHARINDEX('.', t.Name)-CHARINDEX(',', t.Name)-2) as Titulo
FROM
  titanic t
```



Variável: texto de onde desejamos extrair um subtexto.

Posição inicial: localização da vírgula com adição de 2 posições, 1 para eliminar a própria vírgula e outra para eliminar o espaço após a vírgula.

Comprimento: localização do ponto final menos a Posição Inicial, ou seja, a localização da vírgula já ajustada para desconsiderar a vírgula e o espaço subsequente.

	Name	pos_virgula	pos_ponto	Titulo
	NVARCHAR (100)	INT	INT	NVARCHAR (100)
1	Braund, Mr. Owen Harris	7	11	Mr
2	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	8	13	Mrs
3	Heikinen, Miss. Laina	10	16	Miss
4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	9	14	Mrs
5	Allen, Mr. William Henry	6	10	Mr
6	McCarthy, Mr. Timothy J	8	10	Mr
7	McCarthy, Mr. Timothy J	8	13	Mr
8	McCarthy, Mr. Timothy J	8	16	Master
9	McCarthy, Mr. Timothy J	8	13	Mrs
10	McCarthy, Mr. Timothy J	7	12	Mrs

Preparação de Dados

Transformações em campo Texto



Outra função bastante interessante é a **CONCAT_WS**, que é uma variação da função **CONCAT**. Ambas têm o mesmo objetivo: **concatenar strings**. A diferença é que na 1ª você pode especificar um separador enquanto a 2ª concatena apenas as strings indicadas.

```
SELECT TOP 10
  t.Pclass,
  t.Sex,
  CONCAT_WS(' - ', t.Pclass, t.Sex) as pclass_sex
FROM
  titanic t
```



123 Pclass	ABC Sex	ABC pclass_sex
3.00	male	3 - male
1.00	female	1 - female
3.00	female	3 - female
1.00	female	1 - female
3.00	male	3 - male
3.00	male	3 - male
1.00	male	1 - male

Preparação de Dados

Transformações em campo Texto



Uma tarefa muito realizada também é a **comparação de strings** para identificar **erros de digitação**. Para esse objetivo existe a função **SOUNDEX**, que gera um código baseado na **fonética** das strings.

```
SELECT TOP 100
    t.Name,
    SOUNDEX(t.Name)
FROM
    titanic t
ORDER BY
    SOUNDEX(t.Name)
```



ABC Name	ABC
Ayoub, Miss. Banoura	A100
Abbott, Mrs. Stanton (Rosa Hunt)	A130
Abbott, Mr. Rossmore Edward	A130
Abelson, Mr. Samuel	A142
Abelson, Mrs. Samuel (Hannah Witosky)	A142
Appleton, Mrs. Edward Dale (Charlotte Lamson)	A143
Abbing, Mr. Anthony	A152
Aubart, Mme. Leontine Pauline	A163

Transformações Numéricas



Preparação de Dados

Transformações Numéricas



Existem diversas **funções matemáticas** disponíveis no SQL. Como nas aulas anteriores, em alguns momentos transformar as variáveis pode ajudar a **normalizar** os dados.

SELECT

t.Fare,

CEILING(t.Fare/10) as fare_arred_cima,

FLOOR(t.Fare/10) as fare_arred_baixo,

LOG(CAST(t.Fare+1 AS FLOAT)) as fare_ln,

LOG10(CAST(t.Fare+1 AS FLOAT)) as fare_log10,

SQRT(CAST(t.Fare+1 AS FLOAT)) as fare_raiz

FROM

titanic t



	123 Fare T↑	123 fare_arred_cima T↑	123 fare_arred_baixo T↑	123 fare_ln T↑	123 fare_log10 T↑	123 fare_raiz T↑
1	725	73	72	6,5875500148	2,8609366207	26,9443871706
2	712.833	71.284	71.283	13,4770038532	5,852988406	844,2949721513
3	7.925	793	792	8,9779037738	3,899054068	89,0280854562
4	531	54	53	6,2766434893	2,7259116323	23,0651251893
5	805	81	80	6,6920837425	2,9063350418	28,3901391332
6	84.583	8.459	8.458	11,3455004024	4,9272882192	290,8332855778
7	518.625	51.863	51.862	13,1589382858	5,7148542852	720,1569273429
8	21.075	2.108	2.107	9,9558902313	4,3237881899	145,1757555517

Transformações de Data



Preparação de Dados

Transformações de Data



As **variáveis de Datas** possuem muita informação e permitem que criemos **novas variáveis** para indicar tendência, sazonalidade, diferença entre datas e muitas outras. Uma função importante é a **DATEPART**:

SELECT

```
o.order_purchase_timestamp,  
DATEPART(yy, o.order_purchase_timestamp) as ano,  
DATEPART(mm, o.order_purchase_timestamp) as mes,  
DATEPART(dd, o.order_purchase_timestamp) as dia,  
DATEPART(hh, o.order_purchase_timestamp) as hora,  
DATEPART(minute, o.order_purchase_timestamp) as minuto,  
DATEPART(ss, o.order_purchase_timestamp) as segundo,  
DATEPART(dy, o.order_purchase_timestamp) as dia_ano,  
DATEPART(wk, o.order_purchase_timestamp) as semana_ano
```

FROM

```
db_olist.orders o
```



🕒 order_purchase_timestamp 📈	🕒 ano 📈	🕒 mes 📈	🕒 dia 📈	🕒 hora 📈	🕒 minuto 📈	🕒 segundo 📈	🕒 dia_ano 📈	🕒 semana_ano 📈
2017-10-02 10:56:33	2,017.00	10.00	2.00	10.00	56.00	33.00	275.00	40.00
2018-07-24 20:41:37	2,018.00	7.00	24.00	20.00	41.00	37.00	205.00	30.00
2018-08-08 08:38:49	2,018.00	8.00	8.00	8.00	38.00	49.00	220.00	32.00
2017-11-18 19:28:06	2,017.00	11.00	18.00	19.00	28.00	6.00	322.00	46.00
2018-02-13 21:18:39	2,018.00	2.00	13.00	21.00	18.00	39.00	44.00	7.00

Preparação de Dados







Transformações de Data



Uma outra funcionalidade bastante utilizada é o cálculo de **diferença entre datas**. Para isso está disponível a função **DATEDIFF**.

```
SELECT
  o.order_purchase_timestamp,
  o.order_delivered_customer_date,
  DATEDIFF(dd, o.order_purchase_timestamp, o.order_delivered_customer_date) as dias
FROM
  dbolist.orders o
```



 order_purchase_timestamp 	 order_delivered_customer_date 	 123 dias 
2017-12-20 23:45:07	2018-01-09 18:14:02	20.00
2018-04-22 23:23:18	2018-04-30 17:57:25	8.00
2018-08-03 08:59:39	2018-08-17 00:49:41	14.00
2018-05-14 08:35:33	2018-05-18 14:48:38	4.00
2017-11-22 11:32:22	2017-12-28 19:43:00	36.00
2017-03-30 07:50:33	2017-04-10 02:59:52	11.00

Preparação de Dados

Transformações de Data



Também está disponível uma função para **adicionar tempo** em uma variável de data, considerando diferentes partes: **DATEADD**.

```
SELECT
  o.order_purchase_timestamp,
  DATEADD(dd, 10, o.order_purchase_timestamp) as data_prevista_entrega
FROM
  db_olist.orders o
```



order_purchase_timestamp	data_prevista_entrega
2017-10-02 10:56:33	2017-10-12 10:56:33
2018-07-24 20:41:37	2018-08-03 20:41:37
2018-08-08 08:38:49	2018-08-18 08:38:49
2017-11-18 19:28:06	2017-11-28 19:28:06
2018-02-13 21:18:39	2018-02-23 21:18:39
2017-07-09 21:57:05	2017-07-19 21:57:05
2017-04-11 12:22:08	2017-04-21 12:22:08
2017-05-16 13:10:30	2017-05-26 13:10:30

Preparação de Dados

Transformações de Data



Preditiva.ai

Partes que podem ser utilizadas nas funções de **Datas**.

Informação	Função	<i>datepart</i>	Abreviações
Ano	DATEPART / DATEDIFF / DATEADD	year	yy, yyyy
Trimestre	DATEPART / DATEDIFF / DATEADD	quarter	qq, q
Mês	DATEPART / DATEDIFF / DATEADD	month	mm, m
Dia do ano	DATEPART / DATEDIFF / DATEADD	dayofyear	dy, y
Dia	DATEPART / DATEDIFF / DATEADD	day	dd, d
Semana	DATEPART / DATEDIFF / DATEADD	week	wk, ww
Dia da semana	DATEPART / DATEADD	weekday	dw, w
Hora	DATEPART / DATEDIFF / DATEADD	hour	hh
Minuto	DATEPART / DATEDIFF / DATEADD	minute	mi, n
Segundo	DATEPART / DATEDIFF / DATEADD	second	ss, s
Milisegundo	DATEPART / DATEDIFF / DATEADD	millisecond	ms
Microsegundo	DATEPART / DATEDIFF / DATEADD	microsecond	mcs
Nanosegundo	DATEPART / DATEDIFF / DATEADD	nanosecond	ns



Hands on

Visando planejar a produção, o CEO da **Parch and Posey** solicitou um estudo para analisar se os **tipos de papéis** produzidos pela empresa possuem **demandas diferentes ao longo do ano** e em cada região.

Possíveis etapas:

1. Calcule a média da quantidade de papel comprada (db_parchandposey.orders: standard_qty, gloss_qty, poster_qty) por:
 1. Mês (occured_at) – **Dica:** Use a função DatePart()
 2. Região (db_parchandposey.region: name)
2. Exporte os resultados para o Excel e construa um gráfico para realizar as análises.
3. A demanda aparenta ser diferente por mês ou região?

Tratamento de *missing values*



Preparação de Dados

Tratamento de *missing values*



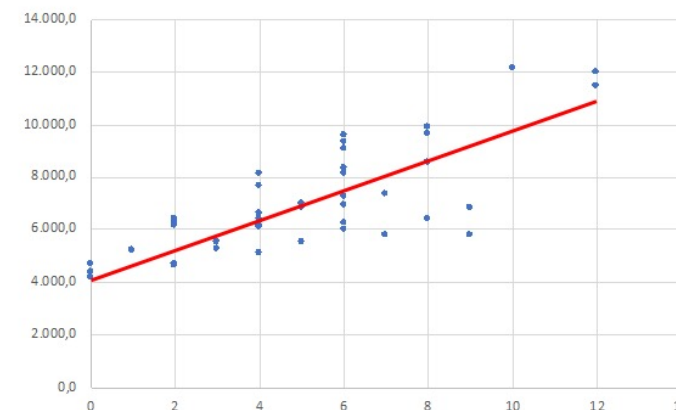
As variáveis que contém ***missing values*** podem ser tratadas para que as informações contidas nas demais variáveis possam ser aproveitadas.

Existe uma área da Estatística chamada **Imputação** que estuda exatamente como fazer o tratamento adequado dos ***missing values***.

Entre os métodos mais simples de **imputação** temos:

- Substituição pela **média geral**
- Substituição pela **média por grupo**

Já entre os métodos mais sofisticados temos a construção de **modelos** para estimar o valor dos ***missing values*** considerando os dados disponíveis, ou dados **não missing**.



Preparação de Dados

Tratamento de *missing values*



O 1º passo é encontrar as variáveis com **missing values**. Para isso vamos calcular a diferença entre o **número total** de linhas e o **número de valores válidos** em cada variável:

```
SELECT
    count(1) as n,
    count(1) - count(t.Survived) n_survived,
    count(1) - count(t.Pclass) n_pclass,
    count(1) - count(t.Name) n_name,
    count(1) - count(t.Sex) n_sex,
    count(1) - count(t.Age) n_age,
    count(1) - count(t.SibSp) n_sibsp,
    count(1) - count(t.Parch) n_parch,
    count(1) - count(t.Ticket) n_ticket,
    count(1) - count(t.Fare) n_fare,
    count(1) - count(t.Cabin) n_cabin,
    count(1) - count(t.Embarked) n_embarked
FROM
    titanic t
```

Verificamos que existem 3 variáveis com ***missing values***:

- Age
- Cabin
- Embarked

Vamos utilizar as técnicas mais simples para fazer a imputação na variável **Age**.

n	n_survived	n_pclass	n_name	n_sex	n_age	n_sibsp	n_parch	n_ticket	n_fare	n_cabin	n_embarked
891.00	0.00	0.00	0.00	0.00	177.00	0.00	0.00	0.00	0.00	687.00	2.00

Preparação de Dados

Tratamento de *missing values*



Para calcular o valor da média geral de idade e utilizá-lo nos registros com *missing values*, vamos utilizar a função de agregação **AVG**, vista anteriormente.

```
DROP TABLE IF EXISTS #idade_media
SELECT
    AVG(t.Age) as idade_media
INTO
    #idade_media
FROM
    titanic t
WHERE
    t.Age IS NOT NULL
SELECT * FROM #idade_media
```

Considerando apenas os valores não nulos, a média de idade é de 38,42 anos.

Para substituir os valores NULL pela idade média, vamos aprender uma nova função: **COALESCE**.

idade_media
38.42

Preparação de Dados

Tratamento de *missing values*



A função **COALESCE** pode ser utilizada como um atalho para a função **CASE WHEN**, dessa forma o código da query fica mais compacto e simples. Após a substituição dos ***missing values***, vamos comparar a média e desvio padrão da **variável Age imputada**.

```
DROP TABLE IF EXISTS #titanic_age_nomissing
SELECT
    t.PassengerId,
    COALESCE(t.Age, (SELECT idade_media FROM #idade_media)) as Age
INTO
    #titanic_age_nomissing
FROM
    titanic t
```

Após a imputação percebemos que a **idade média não foi alterada** e o **desvio padrão diminuiu**.

Com ***missing values***

n	nmiss_age	idade_media	idade_desvpad
891.00	177.00	38.42	56.53

Sem ***missing values***

n	nmiss_age	idade_media	idade_desvpad
891.00	0.00	38.42	50.59

Preparação de Dados

Tratamento de *missing values*



Por ser simples, substituir os *missing values* pela **média geral**, normalmente causa **distorções**. Por esse motivo, vamos fazer a **imputação da média por grupos**, que é uma **evolução** do método que vimos agora.

```
DROP TABLE IF EXISTS #imputacao_grupo
SELECT
    t.Sex,
    t.Survived,
    COUNT(1) as n,
    AVG(t.Age) AS media,
    STDEV(t.Age) AS desvpad
INTO
    #imputacao_grupo
FROM
    titanic t
WHERE
    t.Age IS NOT NULL
GROUP BY
    t.Sex, t.Survived
SELECT * FROM #imputacao_grupo
```

ABC Sex	123 Survived	123 n	123 media	123 desvpad
female	0.00	64.00	31.38	40.19
male	0.00	360.00	45.56	74.23
female	1.00	197.00	31.09	25.63
male	1.00	93.00	31.18	18.74

Preparação de Dados

Tratamento de *missing values*



Por ser simples, substituir os *missing values* pela **média geral**, normalmente causa **distorções**. Por esse motivo, vamos fazer a **imputação da média por grupos**, que é uma **evolução** do método que vimos agora.

```
SELECT
  t.PassengerId,
  t.Sex,
  t.Survived,
  t.Age,
  i.media,
  COALESCE(t.Age, i.media) as new_age
FROM
  titanic t
LEFT JOIN
  #imputacao_grupo i
  ON t.Sex = i.Sex AND
     t.Survived = i.Survived
```

PassengerId	Sex	Survived	Age	media	new_age
1.00	male	0.00	22.00	45.56	22.00
2.00	female	1.00	38.00	31.09	38.00
3.00	female	1.00	26.00	31.09	26.00
4.00	female	1.00	35.00	31.09	35.00
5.00	male	0.00	35.00	45.56	35.00
6.00	male	0.00	[NULL]	45.56	45.56
7.00	male	0.00	54.00	45.56	54.00
8.00	male	0.00	2.00	45.56	2.00

Dados duplicados



Preparação de Dados

Dados duplicados



Ao realizar as queries para extrair os dados uma boa prática é sempre verificar se os dados não estão **duplicados**. Isso porque dependendo da forma como cruzamos 2 tabelas podemos gerar uma **combinação dos campos** fazendo com que o resultado fique replicado.

```
SELECT TOP 1000
  a.name as account_name,
  a.primary_poc
FROM
  db_parchnposey.orders o
  LEFT JOIN db_parchnposey.accounts a
    ON o.account_id = a.id
```



ABC account_name T↑	ABC primary_poc T↑
Walmart	Tamara Tuma
Walmart	Tamara Tuma
Walmart	Tamara Tuma
Walmart	Tamara Tuma
Walmart	Tamara Tuma
Walmart	Tamara Tuma
Walmart	Tamara Tuma

Nesta query selecionamos **todas as compras** com os respectivos **nomes dos clientes** e da **pessoa de contato**.

Preparação de Dados

Dados duplicados



A forma mais simples para eliminar as duplicidades considerando simultaneamente todos os campos é utilizando o comando **DISTINCT**.

```
SELECT DISTINCT TOP 1000
    a.name as account_name,
    a.primary_poc
FROM
    db_parchnposey.orders o
LEFT JOIN db_parchnposey.accounts a
    ON o.account_id = a.id
```



account_name	primary_poc
3M	Orville Leavell
Abbott Laboratories	Seymour Olmedo
AbbVie	Agnus Jenkin
ADP	September Jacqu
Advance Auto Parts	Maxwell Vanderg
AECOM	Della Arceo
AES	Maybell Bascomb

O comando **DISTINCT** fará com que o resultado não contenha **nenhuma linha replicada**, ou seja, o resultado conterà apenas **1 linha por combinação** dos valores das variáveis selecionadas.

Preparação de Dados

Missing Values



Hands on

Utilize o método de imputação das médias para tratar os missings da variável **price** da tabela *db_olist2.order_items*.

Possíveis etapas:

1. Na tabela *db_olist2.order_items* utilize a técnica de imputação dos **missing values** da variável **price** considerando a **média** por categoria do produto.

Dicas:

- Procure a tabela no schema **db_olist** que contenha a categoria de produto.
- Faça duas queries. Uma para calcular a média e outra para imputar os dados faltantes (use Joins).

Juntando tudo

Modelo para Base do Titanic



Hands on

Desenvolva um modelo que dê a probabilidade de sobrevivência dos passageiros do Titanic.

Roteiro:

1. Extraia a base do Titanic do banco de dados SQL.
 - **Dica:** No Knime, utilize os nós “Microsoft SQL Server Connector” para a conexão e depois o nó “DB Query Reader” para a consulta SQL.
2. Lembre-se que a variável Age tem missings. Aplique alguma técnica de imputação nessa variável.
3. Desenvolva um modelo de classificação para a variável target “Survived”. Ela atribui o valor 1 para sobreviventes.
4. Interprete o modelo e calcule sua Matriz de Confusão e curva ROC.
5. Refaça os passos 3 e 4 porém com uma base sem imputação. Neste caso, remova os missings da variável Age. Qual sua conclusão ?



Preditiva.ai