



# SEMANA DO PYTHON DA HASHTAG

## Apostila Completa Aula 2

Aprenda como fazer uma análise de dados  
que vai deixar seu chefe impressionado!  
Impressionador do absoluto zero!



Parte 1

# Introdução

# O que vamos aprender

Na segunda aula do Intensivo do Python você vai aprender a criar um código de análise de dados. No dia a dia das empresas, é muito comum dúvidas sobre os resultados da empresa. Um conceito que cada dia mais cresce nas empresas é o *data driven*. Basicamente, é dizer que ações são tomadas com base nos dados e não em achismos. Aprenda como **fazer uma super análise do zero** com os conceitos abaixo:

Importando dados  
de bases .csv

Tratar dados usando  
a biblioteca Pandas

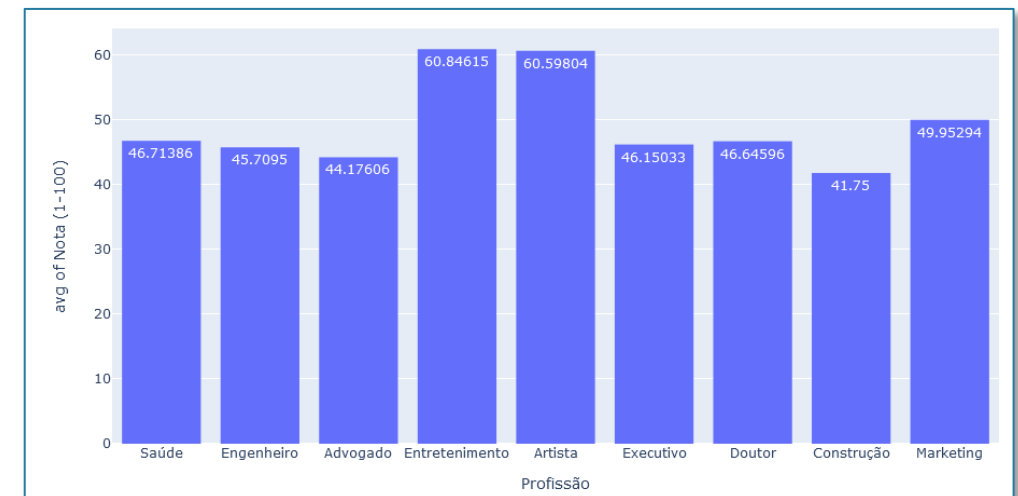
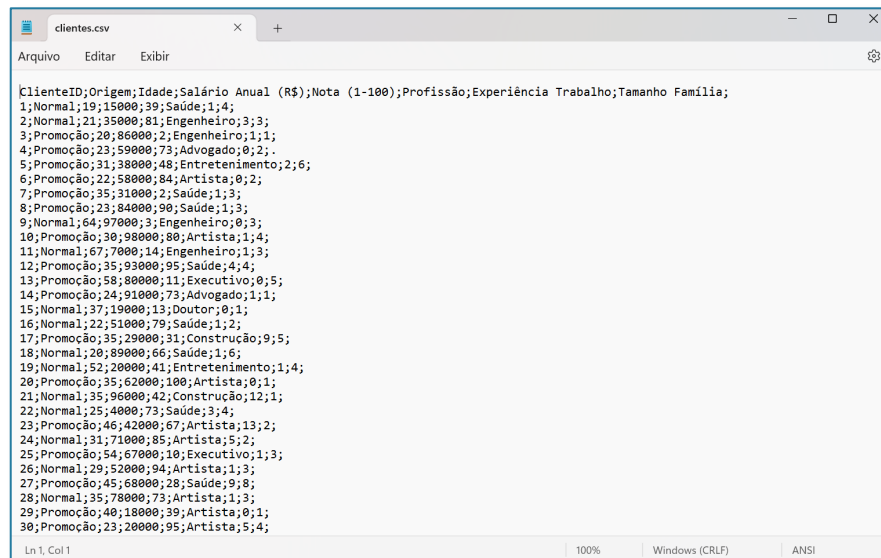
Importação de  
bibliotecas

Criação de gráficos  
usando o plotly

Após todos esses conhecimentos, seremos capazes de transformar uma tabela cheia de informações, nem um pouco fáceis de serem interpretadas ...

... em uma análise super aprofundada que servirão de base para tomada de decisão da gerência. Tudo graças a você! 😊

Prepare-se para **ver além do óbvio**.



# Entendendo a base de dados

Os dados que vamos utilizar são referentes a clientes onde vamos fazer uma análise de perfil de cliente de acordo com essas informações.

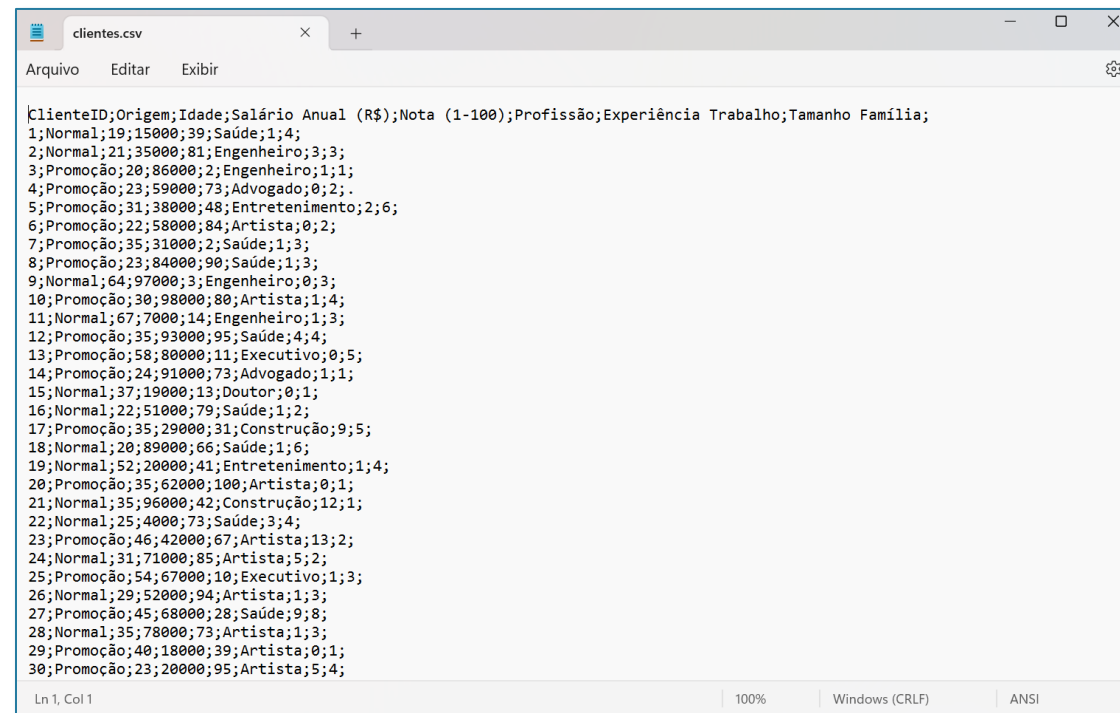
Na imagem ao lado você pode visualizar essas informações que foram extraídas em um arquivo .csv. É possível notar que o arquivo nesse formato não tem formatação das informações e nem é muito bem dividido para facilitar a visualização.

### A situação:

Nós vamos ter que verificar qual das informações mais impacta na nota desses clientes fazendo as devidas análises.

Somente olhando a tabela é difícil verificar se a profissão desse cliente impacta diretamente na sua nota, ou se a sua experiência tem um impacto relevante na nota.

A ideia é fazermos as análises para tirar as conclusões e não supor que algo está acontecendo.



```
clientes.csv
Arquivo  Editar  Exibir

ClienteID;Origem;Idade;Salário Anual (R$);Nota (1-100);Profissão;Experiência Trabalho;Tamanho Família;
1;Normal;19;15000;39;Saúde;1;4;
2;Normal;21;35000;81;Engenheiro;3;3;
3;Promoção;20;86000;2;Engenheiro;1;1;
4;Promoção;23;59000;73;Advogado;0;2;
5;Promoção;31;38000;48;Entretenimento;2;6;
6;Promoção;22;58000;84;Artista;0;2;
7;Promoção;35;31000;2;Saúde;1;3;
8;Promoção;23;84000;90;Saúde;1;3;
9;Normal;64;97000;3;Engenheiro;0;3;
10;Promoção;30;98000;80;Artista;1;4;
11;Normal;67;7000;14;Engenheiro;1;3;
12;Promoção;35;93000;95;Saúde;4;4;
13;Promoção;58;80000;11;Executivo;0;5;
14;Promoção;24;91000;73;Advogado;1;1;
15;Normal;37;19000;13;Doutor;0;1;
16;Normal;22;51000;79;Saúde;1;2;
17;Promoção;35;29000;31;Construção;9;5;
18;Normal;20;89000;66;Saúde;1;6;
19;Normal;52;20000;41;Entretenimento;1;4;
20;Promoção;35;62000;100;Artista;0;1;
21;Normal;35;96000;42;Construção;12;1;
22;Normal;25;4000;73;Saúde;3;4;
23;Promoção;46;42000;67;Artista;13;2;
24;Normal;31;71000;85;Artista;5;2;
25;Promoção;54;67000;10;Executivo;1;3;
26;Normal;29;52000;94;Artista;1;3;
27;Promoção;45;68000;28;Saúde;9;8;
28;Normal;35;78000;73;Artista;1;3;
29;Promoção;40;18000;39;Artista;0;1;
30;Promoção;23;20000;95;Artista;5;4;
```

# Entendendo a solução final

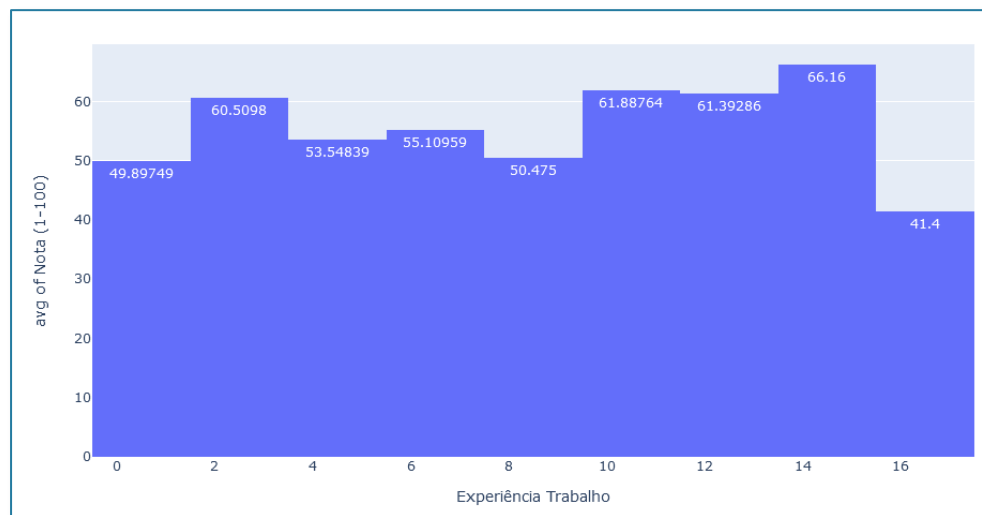
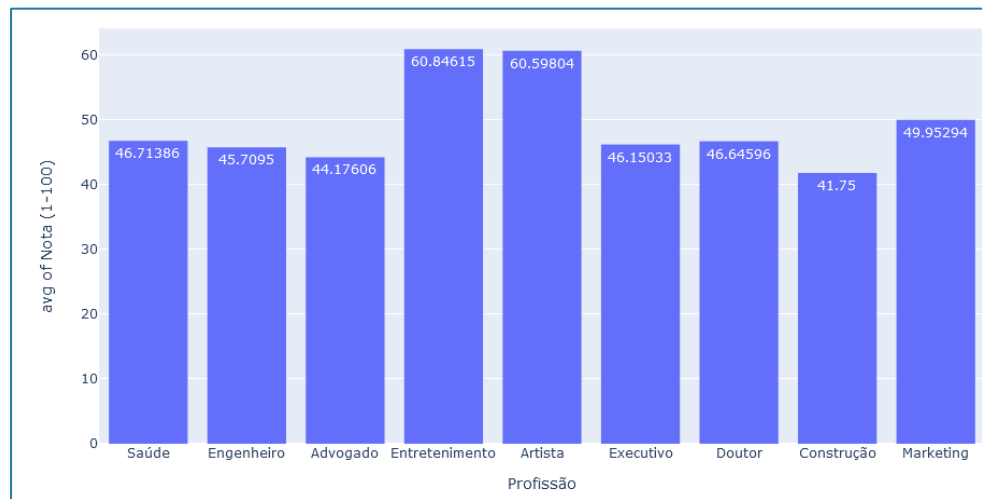
Nesse caso, a solução final vai ser um conjunto de gráficos onde vamos poder tirar nossas próprias conclusões baseado nessas análises!

Estamos aqui tratando de análise de dados. Boa parte da solução aqui é não conhecida nesse estágio.

Quando encontramos esses casos é muito importante, mais até do que a solução em um primeiro momento, definir qual o problema.

Na aula e na apostila vamos fazer uma análise exploratória de dados e buscar direcionadores de causas raiz que podem ser atacadas visando a maior nota desses clientes.

Para isso utilizaremos o Python para nos ajudar em análises gráficas dos dados como este aqui do lado ☺



Parte 2

# Importando e Visualizando os Dados



# Importando base de dados (1/2)

Como vimos na aula 1 do Intensivão, vamos usar bibliotecas que nos facilitem importar dados de planilhas Excel, arquivos .csv, etc.

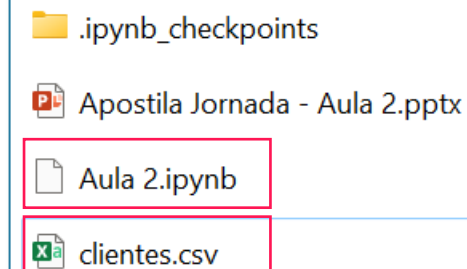
**Novamente usaremos o PANDAS.** Caso você não saiba do que estamos falando aqui, dá uma olhadinha na apostila da Aula 1 do Intensivão!! Lá a gente explica o que são bibliotecas e para que servem 😊.

Vamos começar importando o PANDAS como pd.

Feito isso, precisamos agora buscar o arquivo no nosso pc.

**IMPORTANTE:** Usaremos nessa apostila o arquivo .csv na **MESMA PASTA** do nosso notebook. (Isso evita com que você tenha que escrever o caminho completo do arquivo).

```
import pandas as pd
```



# Importando base de dados (2/2)

Na aula 1 do Intensivão de Python nós já fizemos um exemplo de como importar uma base de dados em csv. Então vamos apenas repetir o procedimento, só que aqui temos alguns detalhes.

Aqui estamos utilizando o `encoding="latin1"`, pois como temos uma base de dados em português, queremos identificar os caracteres especiais que temos (os caracteres latinos).

Outro ponto que você deve ter notado é que agora estamos utilizando o método `tabela.drop`, que serve para excluir uma informação da nossa tabela.

```
import pandas as pd

tabela = pd.read_csv("clientes.csv", encoding="latin1", sep=";")
# tabela = tabela.drop("Unnamed: 8", axis=1)
display(tabela)
```

	CienteID	Origem	Idade	Salário Anual (R\$)	Nota (1-100)	Profissão	Experiência Trabalho	Tamanho Família	Unnamed: 8
0	1	Normal	19	15000	39	Saúde	1	4	NaN
1	2	Normal	21	35000	81	Engenheiro	3	3	NaN
2	3	Promoção	20	86000	2	Engenheiro	1	1	NaN
3	4	Promoção	23	59000	73	Advogado	0	2	.
4	5	Promoção	31	38000	48	Entretenimento	2	6	NaN
...	...	...	...	...	...	...	...	...	...
1995	1996	Promoção	71	184387	48	Artista	8	7	NaN
1996	1997	Promoção	91	73158	28	Doutor	7	7	NaN
1997	1998	Normal	87	90961	14	Saúde	9	2	NaN
1998	1999	Normal	77	182109	4	Executivo	7	2	NaN
1999	2000	Normal	90	110610	64	Entretenimento	5	2	NaN

2000 rows × 9 columns

Isso é importante, pois quando estamos tratando com dados, vamos querer usar apenas as informações corretas e tratadas, assim não vamos levar erros as análises e a nossa conclusão.

Nesse caso se você apenas rodar a primeira linha de código (importando a base de dados) e logo utilizar o `display` para visualizá-la, vai notar que existe uma coluna extra chamada `Unnamed: 8`.

Por esse motivo é que estamos excluindo essa coluna, pois ela não tem informações, então vamos excluir para não ter que carregar essa coluna sem nada para a nossa análise.



# Importando e visualizando os dados

## Priorizando os dados importados

Cada vez mais, saber que dados são úteis ou não é algo fundamental no dia a dia do trabalho.

É muito comum, termos bases de dados ENORMES extraídas de sistemas.

Saber **separar o que é útil do que não é**, é fundamental para uma boa análise de dados.

Ao usarmos **display(tabela)** conseguimos avaliar um pouco melhor nossa base de dados. No entanto, uma das colunas parece não nos ajudar muito na nossa análise...

Consegue identifica-la? A coluna **Unnamed: 8**, além de não ter um rótulo que nos ajude a entender de que se trata, possui números que não seguem nenhum tipo de padrão. Portanto, pelo menos por enquanto, não nos é interessante pois é uma informação irrelevante para nosso estudo.

Sendo assim, podemos retirá-la da nossa tabela. Vamos ver como, a seguir.

Para retirarmos a coluna **Unnamed: 8**, vamos usar o método abaixo:

**.drop()**

Este método será aplicado na variável **tabela** criada na primeira linha do código para receber os dados do arquivo .csv. Este método vai precisar de alguns argumentos:

- Nome da coluna ou código da linha a ser removida: ('**Unnamed: 8**')
- Qual dos eixos deve ser excluído:
  - **0** ou '**index**' será apagada a linha indicada;
  - **1** ou '**columns**' será apagada a coluna indicada.

	ClienteID	Origem	Idade	Salário Anual (R\$)	Nota (1-100)	Profissão	Experiência Trabalho	Tamanho Família
0	1	Normal	19	15000	39	Saúde	1	4
1	2	Normal	21	35000	81	Engenheiro	3	3
2	3	Promoção	20	86000	2	Engenheiro	1	1
3	4	Promoção	23	59000	73	Advogado	0	2
4	5	Promoção	31	38000	48	Entretenimento	2	6
...	...	...	...	...	...	...	...	...
1995	1996	Promoção	71	184387	48	Artista	8	7
1996	1997	Promoção	91	73158	28	Doutor	7	7
1997	1998	Normal	87	90961	14	Saúde	9	2
1998	1999	Normal	77	182109	4	Executivo	7	2
1999	2000	Normal	90	110610	64	Entretenimento	5	2

2000 rows x 8 columns

# Visualizar a base de dados importada

Bem, já importamos nossa base de dados... Agora vamos visualizar as informações. Vamos utilizar a função Display, como fizemos na primeira aula. O Display deixa um visual mais interessante para a sua base de dados ao invés de utilizar apenas o Print.

Você deve ter notado que a base de dados em csv era muito difícil de entender e até mesmo saber qual informação está em cada uma das colunas. Com o pandas essa visualização fica bem mais amigável e prática.

**IMPORTANTE:** A base original, NÃO foi afetada. **Aqui estamos apenas importando a base de dados original, atribuindo ela a uma variável e fazendo os tratamentos que estão armazenados na variável.**

	CientelD	Origem	Idade	Salário Anual (R\$)	Nota (1-100)	Profissão	Experiência Trabalho	Tamanho Família
0	1	Normal	19	15000	39	Saúde	1	4
1	2	Normal	21	35000	81	Engenheiro	3	3
2	3	Promoção	20	86000	2	Engenheiro	1	1
3	4	Promoção	23	59000	73	Advogado	0	2
4	5	Promoção	31	38000	48	Entretenimento	2	6
...	...	...	...	...	...	...	...	...
1995	1996	Promoção	71	184387	48	Artista	8	7
1996	1997	Promoção	91	73158	28	Doutor	7	7
1997	1998	Normal	87	90961	14	Saúde	9	2
1998	1999	Normal	77	182109	4	Executivo	7	2
1999	2000	Normal	90	110610	64	Entretenimento	5	2

2000 rows × 8 columns

Parte 3

# Tratamento e Visão Geral dos Dados

# Limpando a base de dados (1/3)

É muito comum que base de dados extraídas de sistemas possuam dados faltantes e/ou dados que não são corretos. Todos esses dados influenciam diretamente nos resultados obtidos na nossa análise.

Imagine um **caso genérico** em que preciso calcular a média do salário anual dos clientes. Agora imagine que algumas dessas informações estejam vazias, elas vão impactar diretamente no nosso cálculo.

Assim, é sempre importante antes de qualquer análise, avaliar se precisamos tratar esta base de dados ou não.

```
print(tabela.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ClienteID             2000 non-null  int64
1   Origem                 2000 non-null  object
2   Idade                  2000 non-null  int64
3   Salário Anual (R$)     2000 non-null  object
4   Nota (1-100)           2000 non-null  int64
5   Profissão              1965 non-null  object
6   Experiência Trabalho   2000 non-null  int64
7   Tamanho Família       2000 non-null  int64
dtypes: int64(5), object(3)
memory usage: 125.1+ KB
None
```

Analizando nossa base conseguimos perceber que temos 2 problemas:

- **Salário Anual** está formatado como objeto, ou seja, não está com o formato de número. Isso é muito importante, pois só vamos conseguir fazer cálculos se essas informações estiverem como número, caso contrário não vamos conseguir fazer as devidas análises.
- Coluna **Profissão** possui algumas informações vazias, pois temos 1965 informações não vazias. Isso quer dizer que 35 informações estão faltando.

# Limpando a base de dados (2/3)

Para corrigir a base de dados, vamos precisar novamente usar o pandas e seus métodos.

O primeiro método que usaremos será o :

`pd.to_numeric` ([documentação](#))

Como podemos ver na imagem ao lado, esse método se utiliza de alguns argumentos (valores que ficam entre os parênteses do método):

- Coluna que deverá ser transformada;
- Definição do que será feito em caso de erro.

Dica: Sempre consulte a documentação de uma biblioteca ou método antes de utilizá-la em um código 😊

\* **Coerce** → Indica que em caso de erro, o valor a ser considerado na transformação será NaN (Not a Number, não é um número).

```
tabela["Salário Anual (R$)"] = pd.to_numeric(tabela["Salário Anual (R$)"], errors="coerce")
print(tabela.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   ClienteID             2000 non-null  int64  
1   Origem                 2000 non-null  object  
2   Idade                  2000 non-null  int64  
3   Salário Anual (R$)    1999 non-null  float64 
4   Nota (1-100)          2000 non-null  int64  
5   Profissão              1965 non-null  object  
6   Experiência Trabalho  2000 non-null  int64  
7   Tamanho Família       2000 non-null  int64  
dtypes: float64(1), int64(5), object(2)
memory usage: 125.1+ KB
None
```

Aqui você já nota que o tipo da coluna de Salário Anual agora é **float** (decimal).

Dessa forma nós vamos conseguir trabalhar com esses dados de forma correta e fazer os devidos cálculos e análises sem problema.

# Tratamento e visão geral dos dados

## Limpando a base de dados (3/3)

Corrigido o primeiro problema, nós vamos partir para as informações vazias que temos na coluna de **Profissão**.

Para isso, vamos utilizar outro método do pandas muito próximo do **.drop** que usamos anteriormente.

Esse método será o:

**.dropna()** ([documentação](#))

Ele nos permite excluir, todas as linhas que possuam a linha toda completa de dados do tipo NA (None ou Not a Number).

```
display(tabela[tabela["Profissão"].isna()])
tabela = tabela.dropna()
print(tabela.info())
```

Esse código vai mostrar inicialmente somente as informações vazias ou sem informação da coluna Profissão. Então vai visualizar somente esse “pedaço” da base de dados.

Em seguida vamos utilizar o método **.dropna()** para excluir todas essas informações, ou seja, vamos excluir todas essas linhas da nossa base de dados.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1965 entries, 0 to 1999
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ClienteID             1965 non-null   int64
1   Origem                 1965 non-null   object
2   Idade                  1965 non-null   int64
3   Salário Anual (R$)     1965 non-null   float64
4   Nota (1-100)           1965 non-null   int64
5   Profissão              1965 non-null   object
6   Experiência Trabalho   1965 non-null   int64
7   Tamanho Família       1965 non-null   int64
dtypes: float64(1), int64(5), object(2)
memory usage: 138.2+ KB
None
```

Você vai notar que agora a quantidade total de linhas que temos na tabela foi atualizada para 1965, pois essa é a quantidade de linhas que temos com informações, então foram removidas as linhas em que as profissões não haviam sido preenchidas.



Parte 4

# Analizando os Dados

# Analisando os dados

## Como seguir?

O que vamos ver daqui para a frente, vai além do Python em si.

Poderíamos explicar milhões de formas de analisar os dados do Python, mas uma coisa é essencial:

### O QUE EU QUERO RESPONDER?

Entender seu problema é fundamental. Assim, será possível orientar sua análise para resolver o problema.

Então vamos lá!

### Nosso problema é:

“O que faz com que um cliente tenha uma nota alta?”

### O que eu quero:

“Descobrir por meios das análises quais as informações eu posso analisar para saber se um cliente terá uma nota mais alta”



# Visualizando o resumo da base de dados (1/2)

O primeiro passo para a nossa análise é utilizar o método **Describe**, pois ele já vai dar um resumo das informações para que possamos entender melhor como funciona a nossa base de dados.

```
display(tabela.describe())
```

	ClienteID	Idade	Salário Anual (R\$)	Nota (1-100)	Experiência Trabalho	Tamanho Família
<b>count</b>	1965.000000	1965.000000	1965.000000	1965.000000	1965.000000	1965.000000
<b>mean</b>	1000.309924	48.894656	110616.009669	52.385242	3.675318	3.757252
<b>std</b>	578.443714	28.414889	45833.860195	28.593269	3.909676	1.968335
<b>min</b>	1.000000	0.000000	0.000000	1.000000	0.000000	1.000000
<b>25%</b>	498.000000	25.000000	74350.000000	29.000000	0.000000	2.000000
<b>50%</b>	1000.000000	48.000000	109759.000000	52.000000	1.000000	4.000000
<b>75%</b>	1502.000000	73.000000	149095.000000	77.000000	7.000000	5.000000
<b>max</b>	2000.000000	99.000000	189974.000000	100.000000	17.000000	9.000000

Não precisa se desesperar, isso é apenas um resumo das informações da nossa base de dados e vou te explicar o que significa cada um deles.

**Count** – Nada mais é do que quantidade de informações não vazias que temos, ou seja, é a quantidade de linhas que temos preenchidas em cada coluna.

**Mean** – É a média dos valores que temos em cada uma das colunas (nesse caso a coluna ClienteID é um tanto quanto irrelevante, pois não temos informações úteis para tirar dessa coluna).

**Std** – É o desvio padrão, ou seja, é quanto cada informação varia em relação a média. Quanto maior o desvio padrão, significa que o conjunto de dados está mais distante da média.

**Min** – É do que o valor mínimo em cada uma das colunas.

**Max** – É o valor máximo em cada uma das colunas.

## Visualizando o resumo da base de dados (2/2)

```
display(tabela.describe())
```

	ClientelID	Idade	Salário Anual (R\$)	Nota (1-100)	Experiência Trabalho	Tamanho Família
count	1965.000000	1965.000000	1965.000000	1965.000000	1965.000000	1965.000000
mean	1000.309924	48.894656	110616.009669	52.385242	3.675318	3.757252
std	578.443714	28.414889	45833.860195	28.593269	3.909676	1.968335
min	1.000000	0.000000	0.000000	1.000000	0.000000	1.000000
25%	498.000000	25.000000	74350.000000	29.000000	0.000000	2.000000
50%	1000.000000	48.000000	109759.000000	52.000000	1.000000	4.000000
75%	1502.000000	73.000000	149095.000000	77.000000	7.000000	5.000000
max	2000.000000	99.000000	189974.000000	100.000000	17.000000	9.000000

Você deve ter notado que entre os valores mínimo e máximo nós temos 25%, 50% e 75%. O que são esses valores? Eles são chamados de **percentis**.

Eles vão indicar que 25%, 50% e 75% das informações são inferiores ao que temos nas colunas.

Como assim? Vou exemplificar para facilitar com a coluna de **Idade**.

	ClientelID	Idade
count	1965.000000	1965.000000
mean	1000.309924	48.894656
std	578.443714	28.414889
min	1.000000	0.000000
25%	498.000000	25.000000
50%	1000.000000	48.000000
75%	1502.000000	73.000000
max	2000.000000	99.000000

Vamos analisar o percentil de 25%. Essa dado quer dizer que 25% das informações da nossa base de dados são de pessoas abaixo dos 25 anos de idade.

Então se eu tivesse uma base com 100 pessoas, 25% delas, ou seja, 25 delas seriam menores de 25 anos.

Com o 50% e o 75% é o mesmo procedimento. Então temos que 50% das pessoas da base de dados, possuem idade inferior a 48 anos.

E vamos ter que 75% das pessoas possuem idade inferior a 73 anos.

# Análise gráfica – importando o Plotly

Um dos caminhos mais comuns e usuais para analisarmos os dados é através de uma análise gráfica.

O objetivo então, é comparar a nota do cliente com as outras 7 informações que temos para verificar se alguma dessas informações tem uma relação com o cliente ter uma nota alta.

Se você parar pra pensar podemos começar com um palpite inicial de que quem tem um salário mais alto tende a ser um cliente melhor.

Mas isso é apenas um palpite. Com os gráficos é que nós vamos poder comprovar e verificar de fato quais informações são melhores para classificar os clientes dessa base de dados.

Para fazermos isso um pouco mais rápido vamos criar um código permita gerar gráficos automaticamente via Python.

Vamos importar mais uma biblioteca para nos ajudar neste processo: **plotly.express**

Caso você tenha interesse para mais informações é só acessar:

Para todos tipos de gráficos:

<https://plotly.com/python/>

Para nosso caso específico de histograma:

<https://plotly.com/python/histograms/>

Esse código vai gerar 8 gráficos, ou seja, vamos ter 8 comparações da nota do cliente. Vamos ter as comparações da nota com o ID do cliente, com a idade, salário, nota, experiência e tamanho da família.

Dessa forma vamos conseguir verificar quais informações tem maior peso na nota dos clientes.

```
import plotly.express as px

for coluna in tabela.columns:
    grafico = px.histogram(tabela, x=coluna, y="Nota (1-100)", text_auto=True, histfunc='avg', nbins=10)
    grafico.show()
```

# Análise gráfica – Criando gráficos automaticamente

Para a criação dos gráficos nós vamos utilizar uma estrutura de repetição chamada For, dessa forma ela vai conseguir repetir o bloco de código várias vezes sem que você precise repetir a escrita desse código.

```
for coluna in tabela.columns:
```

Você pode ler essa estrutura da seguinte forma: Para cada coluna dentro das colunas da tabela (tabela.columns) faça isso.

Isso quer dizer que vamos repetir o código abaixo para cada uma das colunas da nossa base de dados independente da quantidade de colunas.

Esses dados (gráfico) serão calculados a partir do uso do método **.histogram()**.

Podemos ver que os argumentos necessários são :

- **Tabela:** argumento da função, deverá ser fornecido no momento de ativação da função pelo usuário (que é a nossa base de dados);
- **X=coluna:** Serão dados do eixo X. Nesse caso está como “coluna” que vai ser cada uma das colunas de acordo com a repetição dela (começa ClientID, Idade...);
- **y=“Nota (1-100)”:** É a informação que vamos analisar no eixo Y. Aqui ela é fixa, pois vamos analisar a nota com todos os outros dados;
- **Text\_auto=True:** Habilita as informações nas barras para facilitar a visualização;
- **Histfunc='avg':** Fazendo o cálculo da média das notas para cada “fatia de informação”;
- **Nbins=10:** Número de “fatias de informação”. Um exemplo na idade é que vamos ver as colunas sendo analisadas de 0-9, depois de 10-19 e assim por diante. O mesmo vai acontecer com os outros dados para facilitar a visualização.

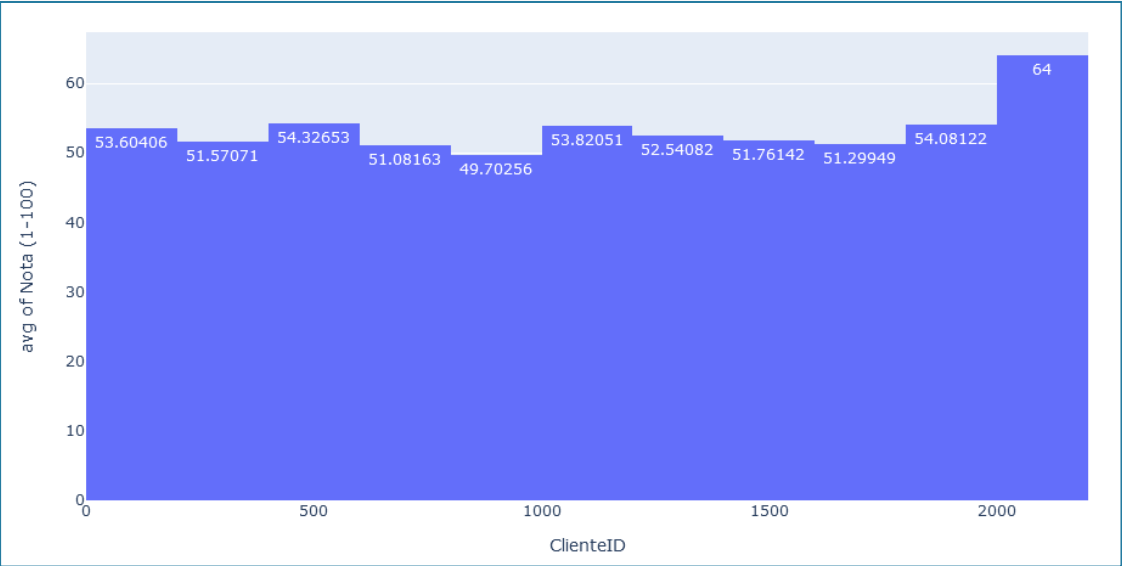
```
import plotly.express as px

for coluna in tabela.columns:
    grafico = px.histogram(tabela, x=coluna, y="Nota (1-100)", text_auto=True, histfunc='avg', nbins=10)
    grafico.show()
```



# Análise gráfica – ClienteID x Nota

Agora que já temos os gráficos criados e você já sabe as informações que foram usadas para gerar cada uma deles, nós podemos começar com a análise desses gráficos para começar a tirar algumas conclusões.



Aqui nós temos a comparação da média de notas pelo ID do Cliente.

Você deve ter notado que esse gráfico não traz muita informação relevante, até porque o ID do Cliente não é algo que traz um peso em relação a nota de cada cliente.

Nada mais é do que um número para identificar o cliente, e pelo próprio gráfico.

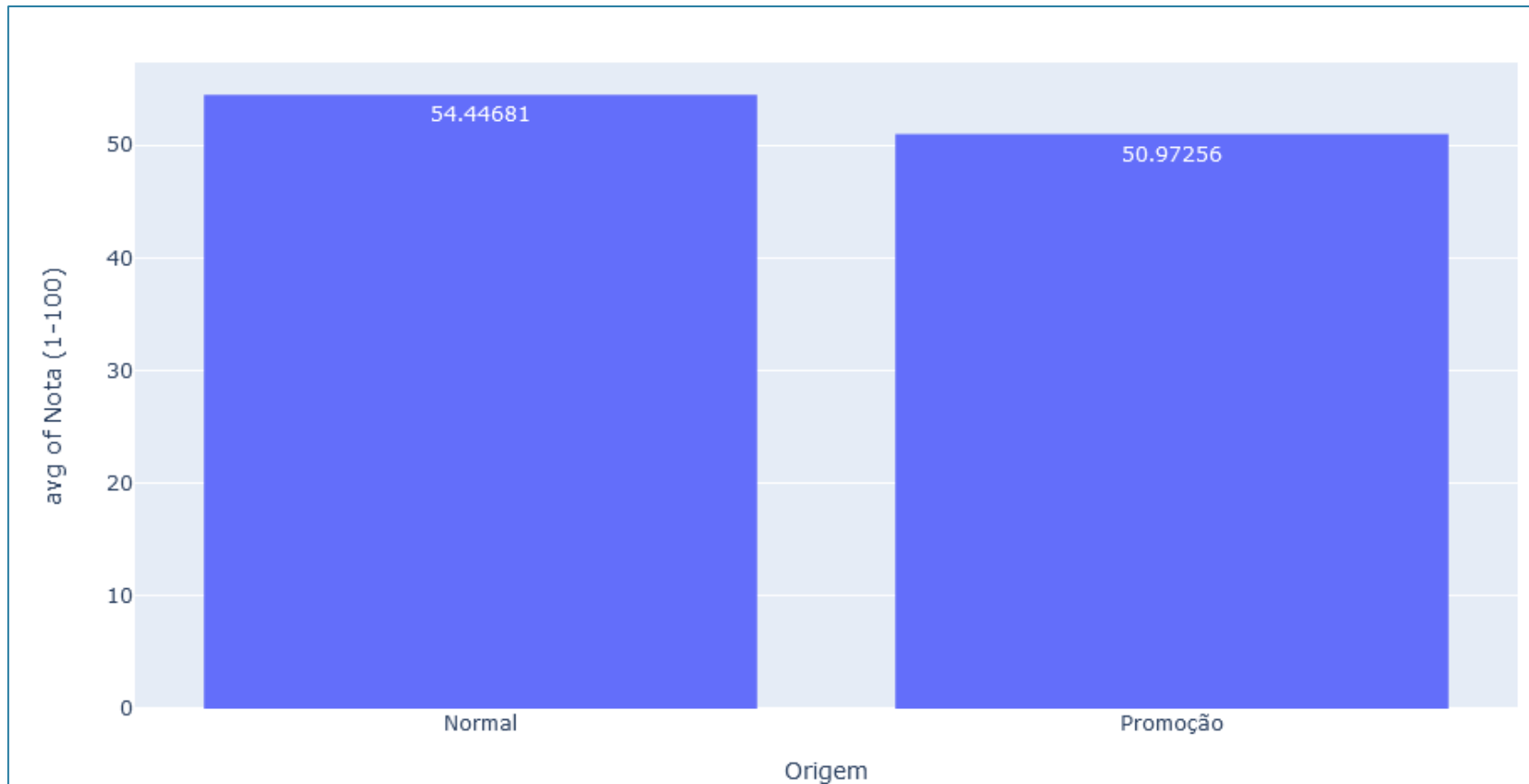
Você pode pensar, mas temos uma nota bem mais alta nos clientes com ID acima de 2000. Isso não é algo notável?

Se você parar para observar, esse bloco foi separado com os clientes com ID de 2000 a 2199. Só que na nossa base de dados o maior valor de ID do cliente é 2000, então nesse cálculo estamos tendo apenas um único valor, que é desse cliente.

Então essa informação continua não tendo muita relevância para a nossa análise.

	ClientelID	Origem	Idade	Salário Anual (R\$)	Nota (1-100)	Profissão	Experiência Trabalho	Tamanho Família
1995	1996	Promoção	71	184387.0	48	Artista	8	7
1996	1997	Promoção	91	73158.0	28	Doutor	7	7
1997	1998	Normal	87	90961.0	14	Saúde	9	2
1998	1999	Normal	77	182109.0	4	Executivo	7	2
1999	2000	Normal	90	110610.0	64	Entretenimento	5	2

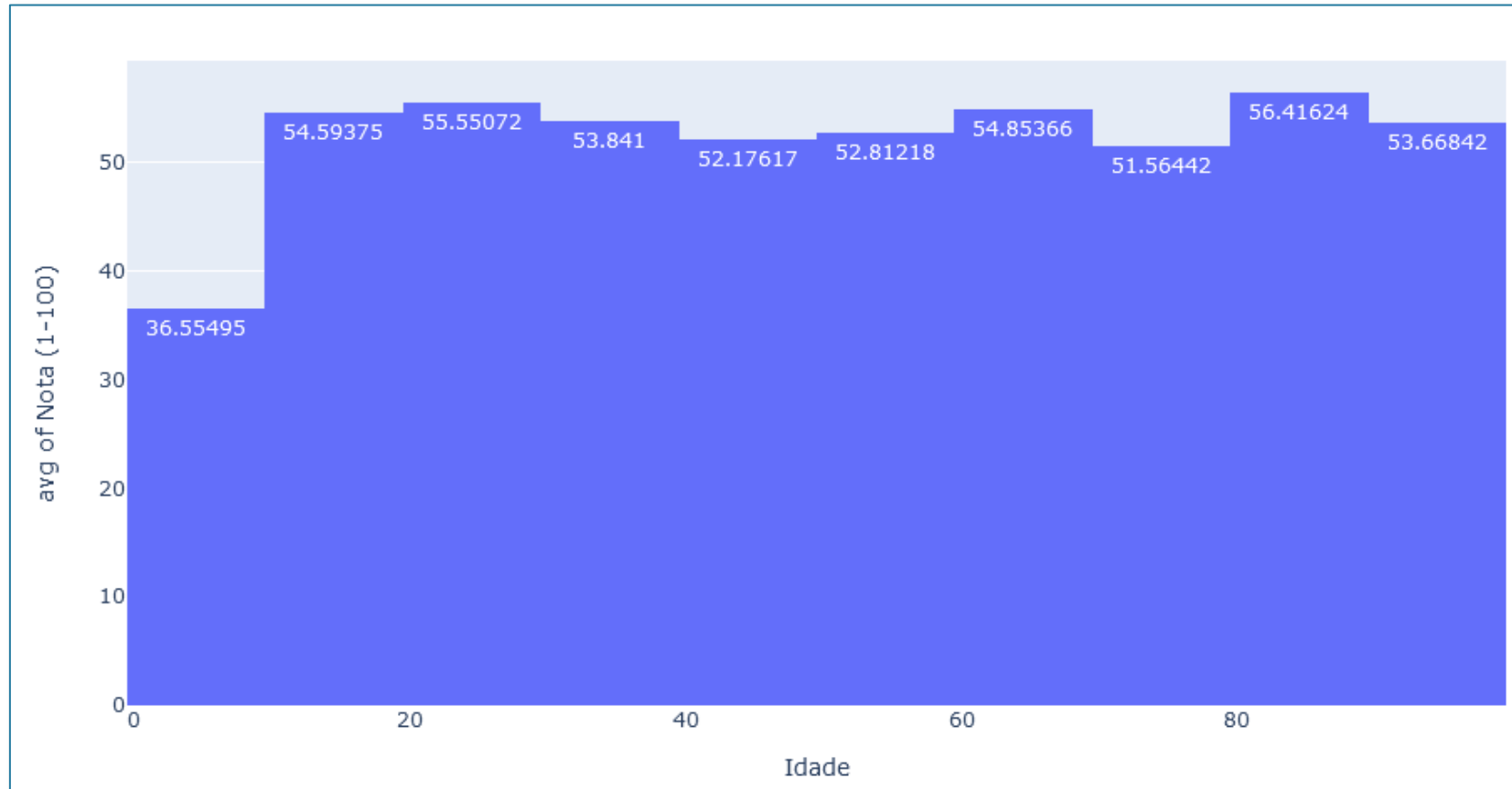
# Análise gráfica – Origem x Nota



Nesse gráfico nós temos a comparação entre a **Origem** do cliente e a **Nota**.

Temos pouca diferença entre as duas origens dos clientes, mas você já consegue notar que os clientes classificados como “Normal” possuem uma nota pouco mais elevada.

# Análise gráfica – Idade x Nota



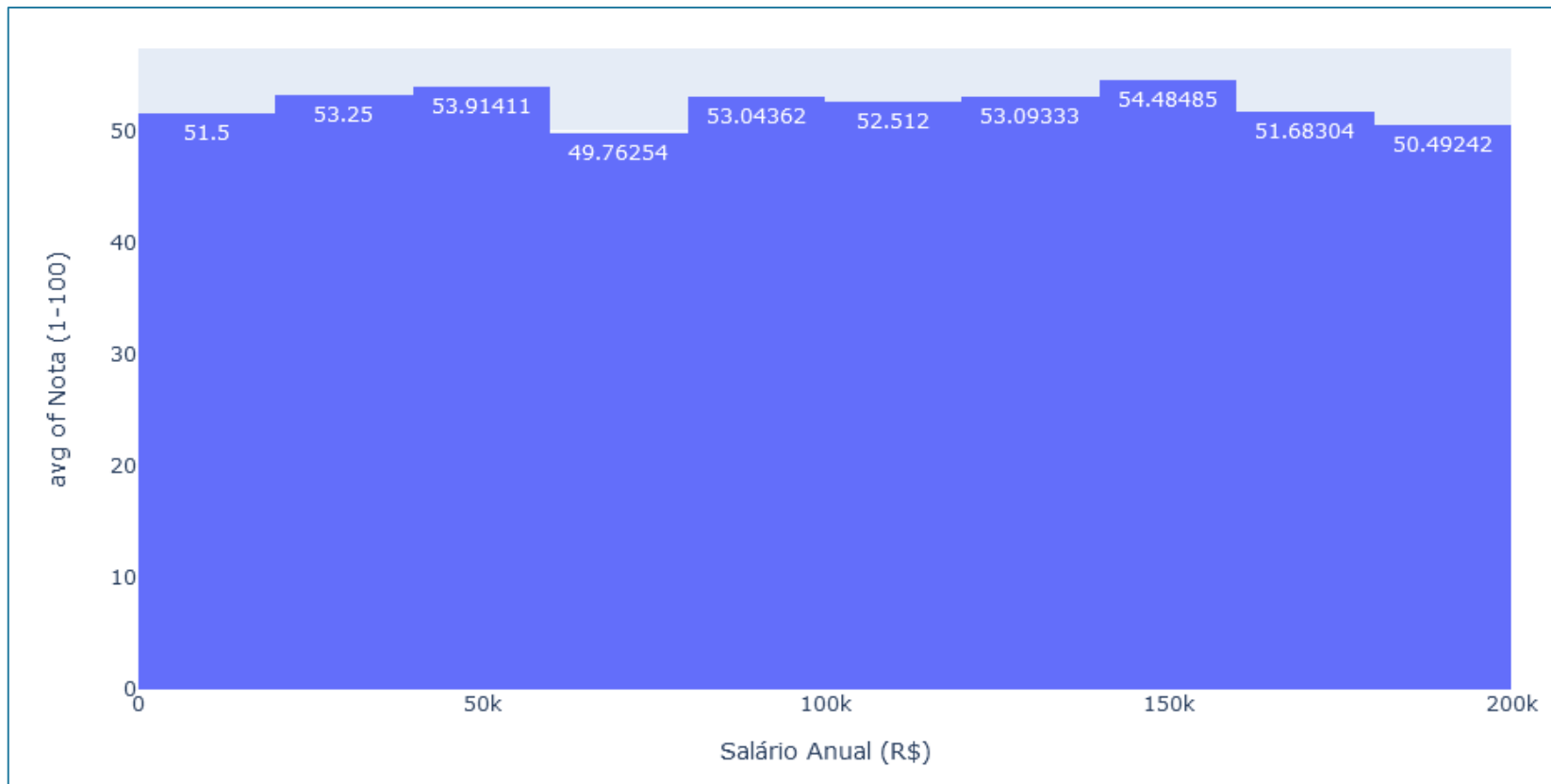
Nesse gráfico nós temos a comparação entre a **Idade** do cliente e a **Nota**.

Aqui você já nota que no primeiro bloco, que é onde temos as idades de 0 a 9 temos uma média de nota muito mais baixa, mas se você reparar, não faz muito sentido ter esse tipo de dado na nossa base não é mesmo?

Até porque pessoas dessa idade não estariam comprando, muito menos teriam profissões, então isso pode ter sido algum erro na base de dados que pode ser tratado também.

Uma sugestão poderia ser a remoção das linhas onde temos idade inferior a 18 por exemplo.

# Análise gráfica – Salário x Nota



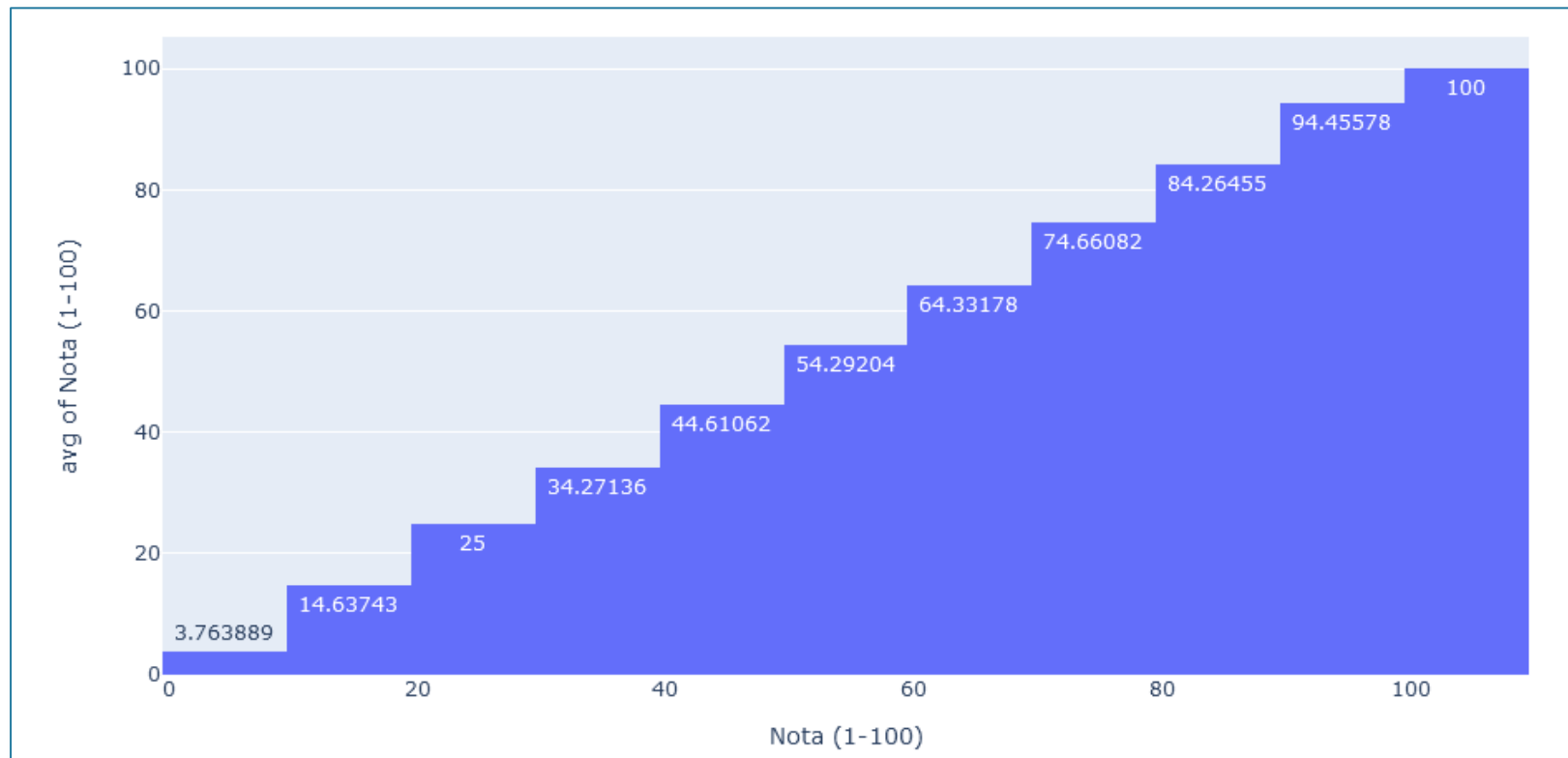
Nesse gráfico nós temos a comparação entre o **Salário** do cliente e a **Nota**.

Veja que o nosso palpite inicial era de que o salário seria um fato para classificar um cliente com uma nota mais alta.

Mas pela análise gráfica esse não é um fator de muita relevância na hora de fazer essa classificação de nota entre os clientes.

Claro que temos uma nota um pouco menor entre os clientes que recebem de R\$60.000 a R\$79.999.

# Análise gráfica – Nota x Nota



Nesse gráfico nós temos a comparação entre a **Nota** do cliente e a **Nota**.

Essa não é uma análise que vamos fazer, até porque não vai fazer sentido analisar a nota em relação a ela mesma para verificar se um cliente tem uma nota alta ou não.

Mas porque o gráfico foi criado? Lembra que fizemos uma automação para não criar os gráficos de forma manual?

Pois é, como a Nota é uma das colunas da tabela, essa análise também foi criada de forma automática.

# Análise gráfica – Profissão x Nota

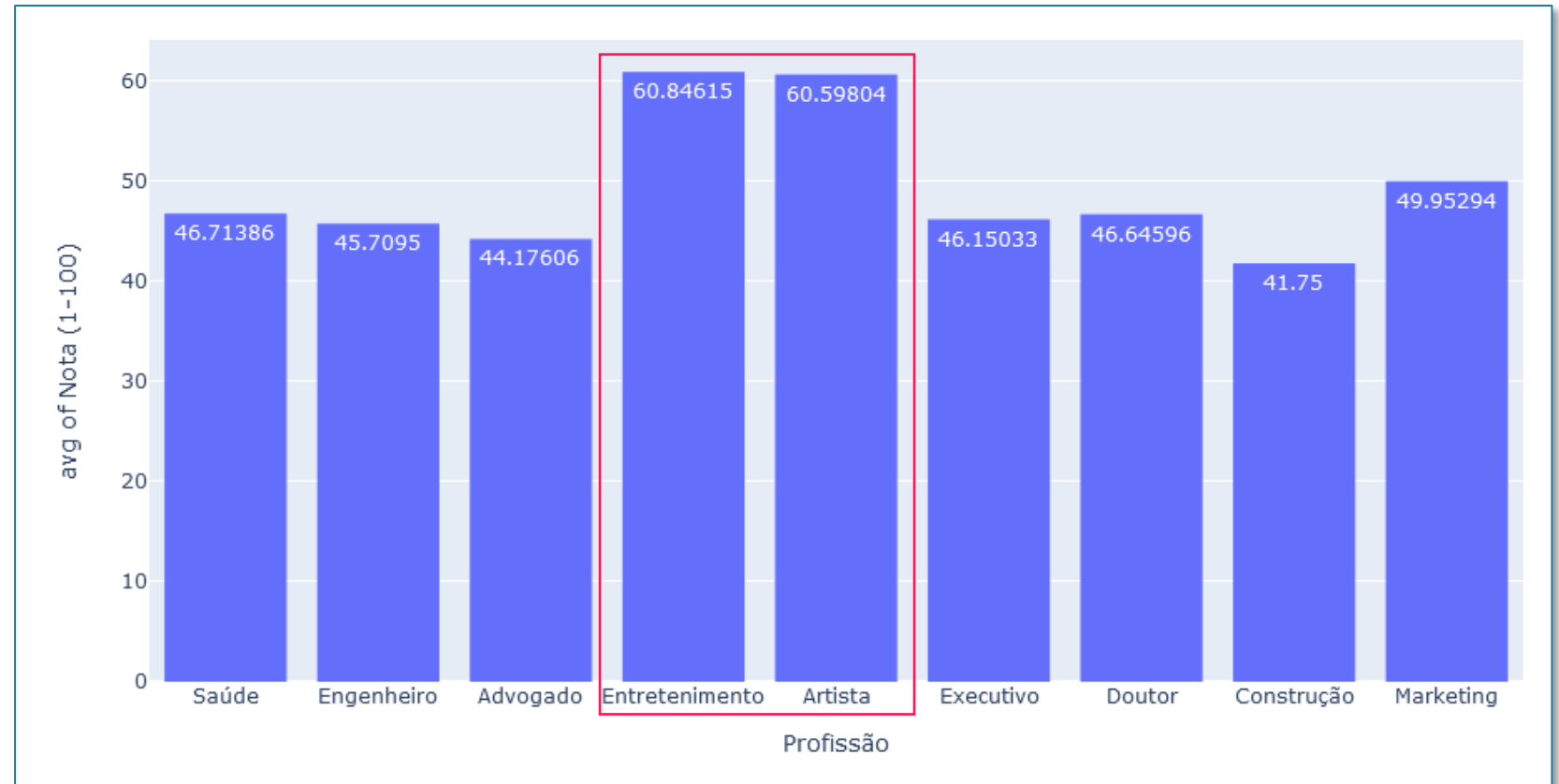
Nesse gráfico nós temos a comparação entre a **Profissão** do cliente e a **Nota**.

Veja como esse gráfico já se diferencia dos outros. Aqui claramente temos uma diferença entre notas em relação a profissão.

Nesse caso podemos verificar que as profissões **Entretenimento** e **Artista** possuem média de nota mais elevada do que as outras profissões.

Isso quer dizer que já temos um elemento que pode ser muito interessante na hora de escolher um cliente.

Isso quer dizer que os clientes das profissões **Entretenimento** e **Artista** tendem a terem uma nota maior do que as demais profissões.



Então se você tivesse que escolher um cliente por profissão, certamente seria uma dessas duas para que ele tenha uma Nota mais elevada.



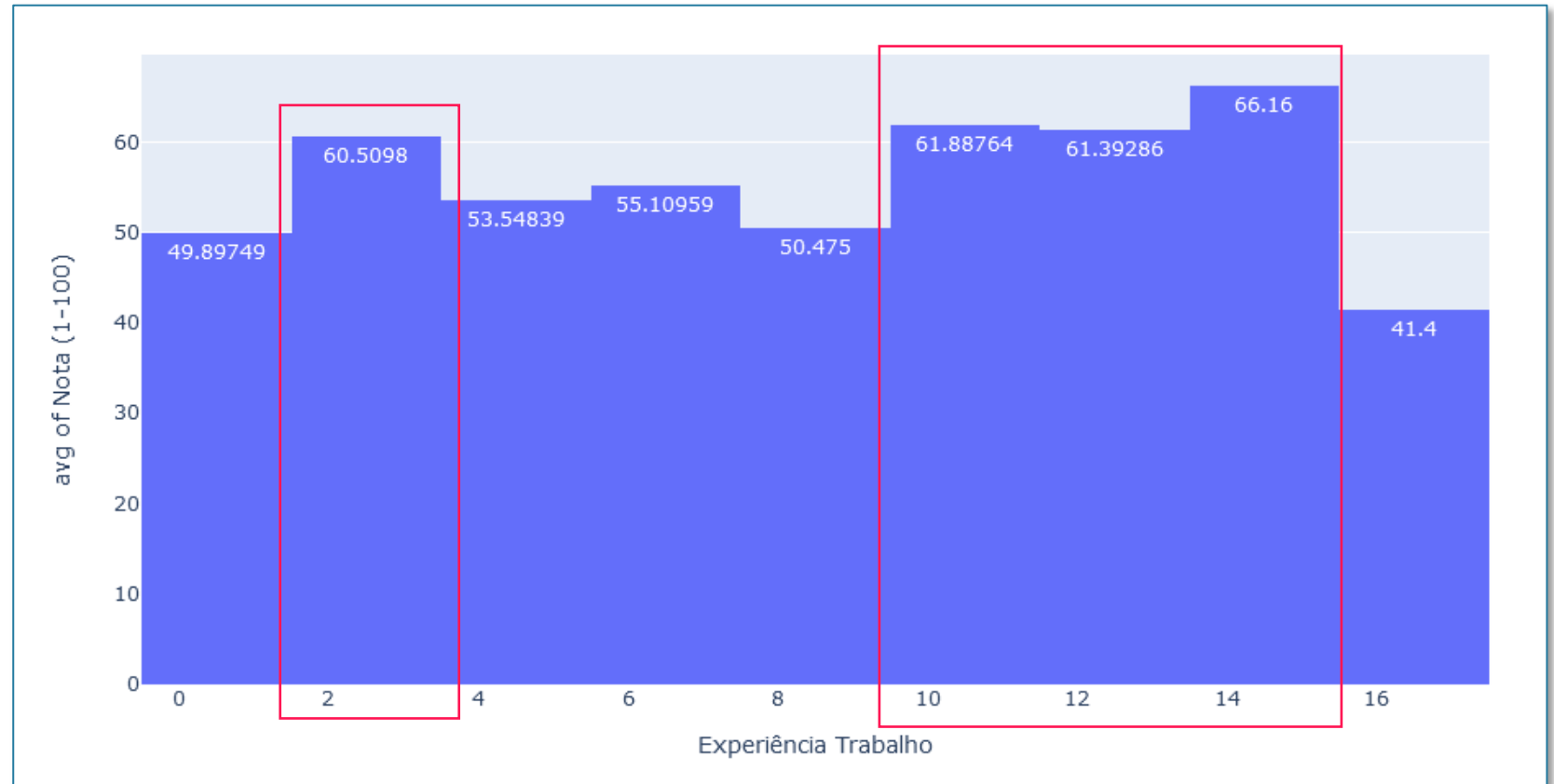
# Análise gráfica – Experiência x Nota

Nesse gráfico nós temos a comparação entre a **Experiência** do cliente e a **Nota**.

Aqui podemos verificar que também temos informações bem relevantes para a classificação do nosso cliente.

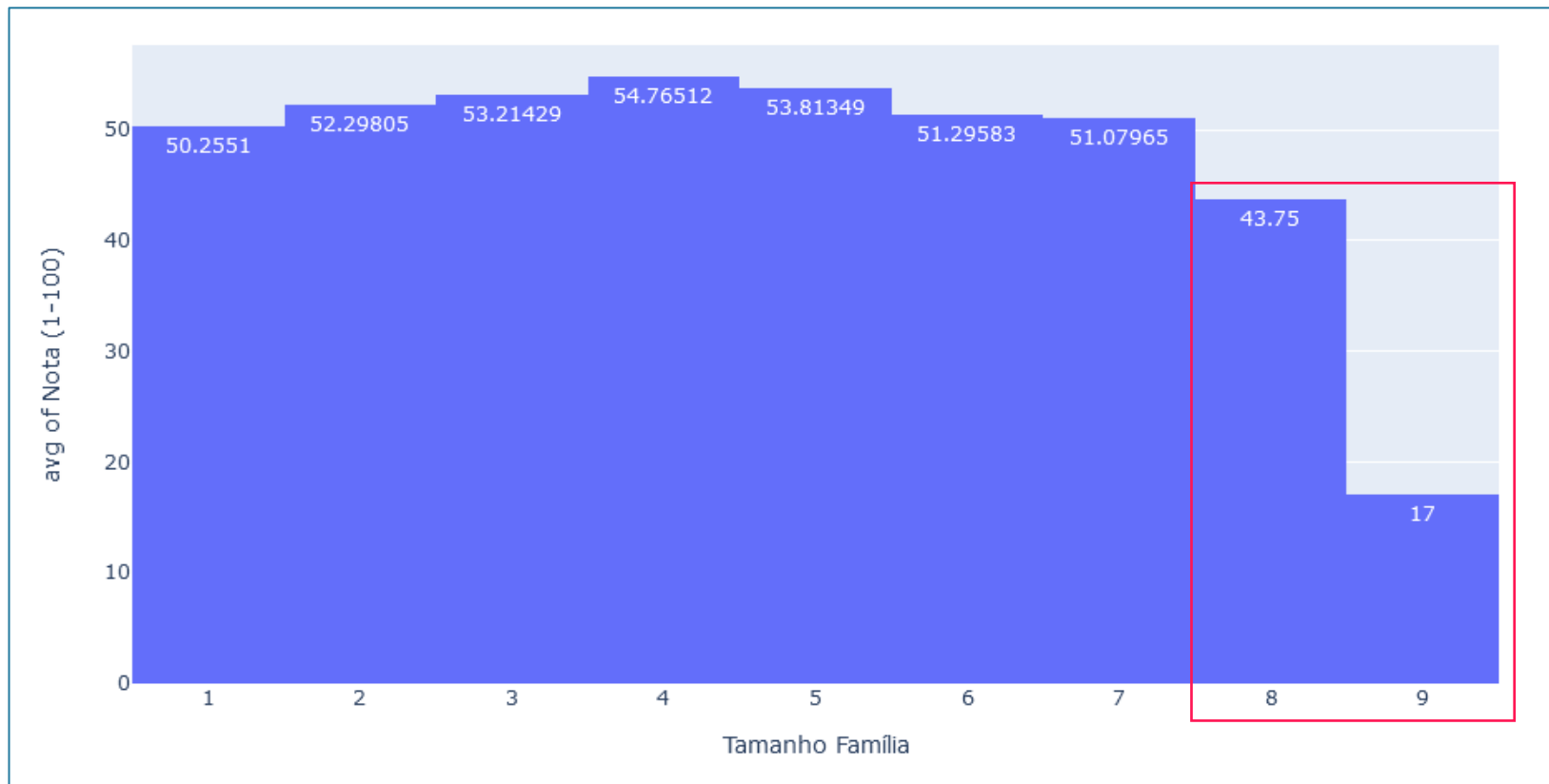
Veja que os clientes de 10 a 15 anos de experiência já possuem uma nota superior aos outros. Os clientes de 2 a 3 anos também possuem uma média de nota muito boa.

Mas por outro lado os clientes de 0 a 1 ano e acima de 15 anos de experiência já não possuem notas muito interessantes.



Neste caso você poderia considerar os clientes nessas duas faixas de experiência. Está começando a entender como é importante esse tipo de análise? Você começa a notar alguns padrões que podem ajudar muito na sua tomada de decisões.

# Análise gráfica – Tamanho Família x Nota



Nesse gráfico nós temos a comparação entre o **Tamanho Família** do cliente e a **Nota**.

Para essa última análise podemos notar que clientes acima de 7 membros na família já possuem notas muito inferiores.

Aqui não temos um elemento para classificar clientes com boas notas, mas já temos informações para saber que esses clientes tendem a ter notas muito mais baixas.

# Análise gráfica – Conclusão

Você conseguiu entender a importância de fazer uma análise de dados? Principalmente com uma base de dados tratada?

Esses são fatores que muitas das pessoas nas empresas acabam deixando passar e isso pode levar a tomada de decisões muito erradas dentro das empresas.

Então sempre que for trabalhar com análise de dados é importante que você saiba o seu problema, aonde quer chegar, o que precisa resolver.

Depois disso você já sabe qual o caminho trilhar e quais análises precisa fazer. Só não pode se esquecer de fazer o tratamento de dados correto da sua base de dados para que considere apenas informações relevantes.

De nada vale fazer diversas análises de dados complexas, com inúmeros cálculos se você estiver utilizando informações inúteis ou até mesmo inválidas.

# ***INTENSIVÃO DE PYTHON*** {#}

*100% ONLINE & GRATUITO*

Ainda não segue a gente no Instagram e nem é inscrito no nosso canal do Youtube? Então corre lá!



@hashtagprogramacao



[youtube.com/hashtag-programacao](https://youtube.com/hashtag-programacao)

