# Marketing-Analytics-Projeto

January 1, 2022

# 1 Business Analytics

# 2 Marketing Analytics

# 3 Projeto

## 3.1 Análise de Indicadores de Performance em Redes de Varejo

```
In [1]: # Versão da Linguagem Python
        from platform import python_version
        print('Versão da Linguagem Python Usada Neste Jupyter Notebook:', python_version())
```

```
Versão da Linguagem Python Usada Neste Jupyter Notebook: 3.7.3
```

## 3.2 Marketing Analytics

Marketing Analytics compreende os processos e tecnologias que permitem aos profissionais de Marketing avaliar o sucesso de suas iniciativas.

Isso é feito medindo o desempenho das campanhas de Marketing, coletando os dados e analisando os resultados. Marketing Analytics utiliza métricas importantes de negócios, como ROI (Retorno Sobre o Investimento), Atribuição de Marketing e Eficácia Geral do Marketing. Em outras palavras, o Marketing Analytics mostra se os programas de Marketing estão sendo efetivos ou não.

Marketing Analytics reúne dados de todos os canais de marketing e os consolida em uma visão de marketing comum. A partir dessa visão comum, você pode extrair resultados analíticos que podem fornecer assistência inestimável para impulsionar os esforços de marketing.

## 3.3 Instalando e Carregando os Pacotes

```
In [2]: # Para atualizar um pacote, execute o comando abaixo no terminal ou prompt de comando:
        # pip install -U nome_pacote

        # Para instalar a versão exata de um pacote, execute o comando abaixo no terminal ou p
        # !pip install nome_pacote==versão_desejada
```


title

```
        # Depois de instalar ou atualizar o pacote, reinicie o jupyter notebook.

        # Instala o pacote watermark.
        # Esse pacote é usado para gravar as versões de outros pacotes usados neste jupyter no
        #!pip install -q -U watermark
```

In [3]:
```
        # Instala o Plotly
        #!pip install -q plotly
```

In [4]:
```
        # Imports
        import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib
        import plotly
        import matplotlib.pyplot as plt
        import plotly.offline as pyoff
        import plotly.graph_objs as go
        from datetime import datetime, timedelta
        %matplotlib inline
        pyoff.init_notebook_mode()
```

In [5]:
```
        # Versões dos pacotes usados neste jupyter notebook
        %reload_ext watermark
        %watermark -a "João Souza" --iversions
```

```
Author: João Souza

numpy      : 1.16.2
pandas     : 0.24.2
```

```
seaborn   : 0.9.0
plotly    : 5.5.0
matplotlib: 3.0.3
```

### 3.4 Carregando os Dados

```
In [6]: # Carrega os dados
        dados = pd.read_csv("dados/dataset.csv", header = 0, encoding = 'unicode_escape')
```

```
In [7]: # Visualiza os dados
        dados.head()
```

```
Out[7]:   NumeroFatura CodigoProduto                          NomeProduto  Quantidade  \
        0       536365         21730   GLASS STAR FROSTED T-LIGHT HOLDER           6
        1       536365        85123A  WHITE HANGING HEART T-LIGHT HOLDER           6
        2       536365         71053                 WHITE METAL LANTERN           6
        3       536365        84406B      CREAM CUPID HEARTS COAT HANGER           8
        4       536365        84029G KNITTED UNION FLAG HOT WATER BOTTLE           6

                DataVenda  ValorUnitario  IdCliente    Pais
        0  12/1/2010 8:26           4.25    17850.0  Brasil
        1  12/1/2010 8:26           2.55    17850.0  Brasil
        2  12/1/2010 8:26           3.39    17850.0  Brasil
        3  12/1/2010 8:26           2.75    17850.0  Brasil
        4  12/1/2010 8:26           3.39    17850.0  Brasil
```

Nota: Cada linha (registro) representa um item de um pedido. Observe que a coluna Numero-Fatura se repete indicando que é um mesmo pedido com itens diferentes. Para cada item temos o produto, a quantidade, o valor unitário, o cliente e o país.

```
In [8]: # Shape
        dados.shape
```

```
Out[8]: (541800, 8)
```

```
In [9]: # Tipos de dados
        dados.dtypes
```

```
Out[9]: NumeroFatura      object
        CodigoProduto     object
        NomeProduto       object
        Quantidade         int64
        DataVenda         object
        ValorUnitario    float64
        IdCliente        float64
        Pais              object
        dtype: object
```

```
In [10]: # Describe
         dados.describe()
```

```
Out[10]:           Quantidade  ValorUnitario      IdCliente
         count  541800.000000  541800.000000  406725.000000
         mean        9.551739       4.611581   15287.754038
         std       218.103033      96.769576    1713.475925
         min    -80995.000000  -11062.060000   12346.000000
         25%         1.000000       1.250000   13954.000000
         50%         3.000000       2.080000   15152.000000
         75%        10.000000       4.130000   16791.000000
         max     80995.000000   38970.000000   18287.000000
```

```
In [11]: # Verificando valores nulos
         dados.isna().sum()
```

```
Out[11]: NumeroFatura          0
         CodigoProduto         0
         NomeProduto        1454
         Quantidade            0
         DataVenda             0
         ValorUnitario         0
         IdCliente        135075
         Pais                  0
         dtype: int64
```

```
In [12]: # Range de datas do período que ocorreram as vendas
         # (observe que devido ao tipo de dado da coluna o resultado será incorreto)
         print('Data Mínima:', dados['DataVenda'].min())
         print('Data Máxima:', dados['DataVenda'].max())
```

```
Data Mínima: 1/10/2011 10:04
Data Máxima: 9/9/2011 9:52
```

```
In [13]: # Converte a coluna de data para o tipo data
         dados.DataVenda = pd.to_datetime(dados.DataVenda)
```

```
In [14]: # Tipos de dados
         dados.dtypes
```

```
Out[14]: NumeroFatura             object
         CodigoProduto            object
         NomeProduto              object
         Quantidade                int64
         DataVenda        datetime64[ns]
         ValorUnitario           float64
         IdCliente               float64
         Pais                     object
         dtype: object
```

```
In [15]: # Range de datas do período que ocorreram as vendas, agora sim com tipo de dado corre
         print('Data Mínima:', dados['DataVenda'].min())
         print('Data Máxima:', dados['DataVenda'].max())
```

```
Data Mínima: 2010-12-01 08:26:00
Data Máxima: 2011-12-09 12:50:00
```

```
In [16]: # Países para os quais ocorreram vendas
         dados['Pais'].unique()
```

```
Out[16]: array(['Brasil', 'Uruguai', 'Australia', 'Holanda', 'Alemanha', 'Noruega',
                'Irlanda', 'Espanha', 'Poland', 'Portugal', 'Italy', 'Belgium',
                'Lithuania', 'Japan', 'Iceland', 'Channel Islands', 'Dinamarca',
                'Cyprus', 'Sweden', 'Austria', 'Israel', 'Finland', 'Bahrain',
                'Greece', 'Hong Kong', 'Cingapura', 'Iraque', 'Equador',
                'Saudi Arabia', 'Czech Republic', 'Canada', 'China', 'Inglaterra',
                'USA', 'Chile', 'Malta', 'Paraguai'], dtype=object)
```

### 3.4.1 Indicadores

### 3.4.2 Faturamento Mensal

Faturamento = Quantidade * Valor_Unitario

```
In [17]: # Extrai o mês da fatura
         dados['AnoMes'] = dados['DataVenda'].map(lambda date: 100 * date.year + date.month)
```

```
In [18]: # Visualiza os dados
         dados.head()
```

```
Out[18]:   NumeroFatura CodigoProduto                         NomeProduto  Quantidade  \
         0       536365         21730   GLASS STAR FROSTED T-LIGHT HOLDER           6
         1       536365        85123A  WHITE HANGING HEART T-LIGHT HOLDER           6
         2       536365         71053                WHITE METAL LANTERN           6
         3       536365        84406B     CREAM CUPID HEARTS COAT HANGER           8
         4       536365        84029G KNITTED UNION FLAG HOT WATER BOTTLE           6

                DataVenda  ValorUnitario  IdCliente    Pais  AnoMes
         0 2010-12-01 08:26:00           4.25    17850.0  Brasil  201012
         1 2010-12-01 08:26:00           2.55    17850.0  Brasil  201012
         2 2010-12-01 08:26:00           3.39    17850.0  Brasil  201012
         3 2010-12-01 08:26:00           2.75    17850.0  Brasil  201012
         4 2010-12-01 08:26:00           3.39    17850.0  Brasil  201012
```

```
In [19]: # Calcula o faturamento
         dados["Faturamento"] = dados["Quantidade"] * dados["ValorUnitario"]
```

```
In [20]: # Visualiza os dados
         dados.head()
```

```
Out[20]:   NumeroFatura CodigoProduto                         NomeProduto  Quantidade  \
0       536365         21730   GLASS STAR FROSTED T-LIGHT HOLDER           6
1       536365        85123A  WHITE HANGING HEART T-LIGHT HOLDER           6
2       536365         71053                WHITE METAL LANTERN           6
3       536365        84406B     CREAM CUPID HEARTS COAT HANGER           8
4       536365        84029G KNITTED UNION FLAG HOT WATER BOTTLE           6

             DataVenda  ValorUnitario  IdCliente    Pais  AnoMes  Faturamento
0 2010-12-01 08:26:00           4.25    17850.0  Brasil  201012        25.50
1 2010-12-01 08:26:00           2.55    17850.0  Brasil  201012        15.30
2 2010-12-01 08:26:00           3.39    17850.0  Brasil  201012        20.34
3 2010-12-01 08:26:00           2.75    17850.0  Brasil  201012        22.00
4 2010-12-01 08:26:00           3.39    17850.0  Brasil  201012        20.34
```

```
In [21]: # Agrupa o faturamento por mês/ano
         df_faturamento = dados.groupby(['AnoMes']).agg({'Faturamento': sum}).reset_index()
```

```
In [22]: # Tabela de dados
         df_faturamento
```

```
Out[22]:    AnoMes  Faturamento
0   201012   748957.020
1   201101   560000.260
2   201102   498062.650
3   201103   683267.080
4   201104   493207.121
5   201105   723333.510
6   201106   691123.120
7   201107   681300.111
8   201108   682680.510
9   201109  1019687.622
10  201110  1070704.670
11  201111  1461756.250
12  201112   431245.000
```

### 3.4.3 Visualização do Indicador

```
In [23]: # Plot

         # Definição dos dados no plot
         plot_data = [go.Scatter(x = df_faturamento['AnoMes'],
                                 y = df_faturamento['Faturamento'],)]

         # Layout
         plot_layout = go.Layout(xaxis = {"type": "category"},
                                 title = 'Faturamento Mensal')

         # Plot da figura
```

```
fig = go.Figure(data = plot_data, layout = plot_layout)
pyoff.iplot(fig)
```

### 3.4.4 Taxa Percentual de Crescimento Mensal

Taxa Percentual de Crescimento Mensal = Faturamento Mensal / Faturamento Mensal Anterior * 100

```
In [24]: # Usamos a função pct_change() para calcular a variação percentual mensal
         df_faturamento['CrescimentoMensal'] = df_faturamento['Faturamento'].pct_change()
```

```
In [25]: # Tabela de dados
         df_faturamento
```

Out[25]:

| | AnoMes | Faturamento | CrescimentoMensal |
|---|---|---|---|
| 0 | 201012 | 748957.020 | NaN |
| 1 | 201101 | 560000.260 | -0.252293 |
| 2 | 201102 | 498062.650 | -0.110603 |
| 3 | 201103 | 683267.080 | 0.371850 |
| 4 | 201104 | 493207.121 | -0.278163 |
| 5 | 201105 | 723333.510 | 0.466592 |
| 6 | 201106 | 691123.120 | -0.044530 |
| 7 | 201107 | 681300.111 | -0.014213 |
| 8 | 201108 | 682680.510 | 0.002026 |
| 9 | 201109 | 1019687.622 | 0.493653 |
| 10 | 201110 | 1070704.670 | 0.050032 |
| 11 | 201111 | 1461756.250 | 0.365228 |
| 12 | 201112 | 431245.000 | -0.704982 |

### 3.4.5 Visualização do Indicador 2

```
In [26]: # Plot

         # Definição dos dados no plot (filtramos o mês 12 de 2011 pois não temos dados sufici
         plot_data = [go.Scatter(x = df_faturamento.query("AnoMes < 201112")['AnoMes'],
                                 y = df_faturamento.query("AnoMes < 201112")['CrescimentoMensa

         # Layout
         plot_layout = go.Layout(xaxis = {"type": "category"},
                                 title = 'Taxa Percentual de Crescimento Mensal')

         # Plot da figura
         fig = go.Figure(data = plot_data, layout = plot_layout)
         pyoff.iplot(fig)
```

### 3.4.6 Clientes Ativos Por Mês em um País (Brasil)

Clientes ativos são aqueles que fizeram pelo menos uma compra em cada mês.

```
In [27]: # Cria um dataframe somente com dados do Brasil -> filtro()
         dados_brasil = dados.query("Pais=='Brasil'").reset_index(drop = True)

In [28]: # Usuários ativos são aqueles que fizeram pelo menos uma compra
         df_ativos_mes = dados_brasil.groupby('AnoMes')['IdCliente'].nunique().reset_index()

In [29]: # Dados
         df_ativos_mes
```

Out[29]:

| | AnoMes | IdCliente |
|---|---|---|
| 0 | 201012 | 871 |
| 1 | 201101 | 684 |
| 2 | 201102 | 714 |
| 3 | 201103 | 923 |
| 4 | 201104 | 817 |
| 5 | 201105 | 985 |
| 6 | 201106 | 943 |
| 7 | 201107 | 899 |
| 8 | 201108 | 867 |
| 9 | 201109 | 1177 |
| 10 | 201110 | 1285 |
| 11 | 201111 | 1548 |
| 12 | 201112 | 614 |

### 3.4.7 Visualização do Indicador

```
In [30]: # Plot

         # Definição dos dados no plot
         plot_data = [go.Bar(x = df_ativos_mes['AnoMes'],
                             y = df_ativos_mes['IdCliente'],)]

         # Layout
         plot_layout = go.Layout(xaxis = {"type": "category"},
                                 title = 'Clientes Ativos Por Mês em um País (Brasil)')

         # Plot da figura
         fig = go.Figure(data = plot_data, layout = plot_layout)
         pyoff.iplot(fig)
```

### 3.4.8 Total de Itens Comprados Por Mês em um País (Brasil)

Total de itens comprados por mês.

```
In [31]: # Agrupa os dados para calcular o total de itens comprados por mês no Brasil
         df_itens_mes = dados_brasil.groupby('AnoMes')['Quantidade'].sum().reset_index()

In [32]: # Dados
         df_itens_mes
```

```
Out[32]:     AnoMes  Quantidade
          0   201012      298101
          1   201101      237381
          2   201102      225641
          3   201103      279843
          4   201104      257666
          5   201105      306452
          6   201106      258522
          7   201107      324129
          8   201108      319804
          9   201109      458490
          10  201110      455612
          11  201111      642281
          12  201112      198750
```

### 3.4.9 Visualização do Indicador

```
In [33]: # Plot

         # Definição dos dados no plot
         plot_data = [go.Bar(x = df_itens_mes['AnoMes'],
                             y = df_itens_mes['Quantidade'],)]

         # Layout
         plot_layout = go.Layout(xaxis = {"type": "category"},
                                 title = 'Total de Itens Comprados Por Mês em um País (Brasil)

         # Plot da figura
         fig = go.Figure(data = plot_data, layout = plot_layout)
         pyoff.iplot(fig)
```

### 3.4.10 Faturamento Médio Mensal em um País (Brasil)

Faturamento médio por mês em um país.

```
In [34]: # Calcula o faturamento médio
         df_faturamento_medio = dados_brasil.groupby('AnoMes')['Faturamento'].mean().reset_inde

In [35]: # Dados
         df_faturamento_medio
```

```
Out[35]:     AnoMes  Faturamento
          0   201012    16.865860
          1   201101    13.614680
          2   201102    16.093027
          3   201103    16.716166
          4   201104    15.773380
          5   201105    17.713823
          6   201106    16.714748
```

```
7   201107    15.723497
8   201108    17.315899
9   201109    18.931723
10  201110    16.093582
11  201111    16.312383
12  201112    16.223363
```

#### 3.4.11 Visualização do Indicador 5

```
In [36]: # Plot

         # Definição dos dados no plot
         plot_data = [go.Bar(x = df_faturamento_medio['AnoMes'],
                             y = df_faturamento_medio['Faturamento'],)]

         # Layout
         plot_layout = go.Layout(xaxis = {"type": "category"},
                                 title = 'Faturamento Médio Mensal em um País (Brasil)')

         # Plot da figura
         fig = go.Figure(data = plot_data, layout = plot_layout)
         pyoff.iplot(fig)

In [37]: # Calcula o faturamento total por mês
         df_faturamento_total = dados_brasil.groupby('AnoMes')['Faturamento'].sum().reset_inde

In [38]: # Dados
         df_faturamento_total

Out[38]:     AnoMes  Faturamento
         0   201012   676742.620
         1   201101   434308.300
         2   201102   408247.910
         3   201103   559707.390
         4   201104   442254.041
         5   201105   596459.860
         6   201106   554478.350
         7   201107   565479.841
         8   201108   539130.500
         9   201109   862018.152
         10  201110   877438.190
         11  201111  1282805.780
         12  201112   386651.410

In [39]: # Plot

         # Definição dos dados no plot
         plot_data = [go.Bar(x = df_faturamento_total['AnoMes'],
                             y = df_faturamento_total['Faturamento'],)]
```

```
# Layout
plot_layout = go.Layout(xaxis = {"type": "category"},
                        title = 'Faturamento Total Mensal em um País (Brasil)')

# Plot da figura
fig = go.Figure(data = plot_data, layout = plot_layout)
pyoff.iplot(fig)
```

### 3.4.12 Diferença de Faturamento ao Longo do Tempo Entre Clientes Novos e Antigos

Para calcular esse indicador precisaremos de um pouco mais de criatividade. O que é um cliente novo ou antigo?

Vamos considerar cliente novo aquele com baixo volume de compras e cliente antigo aquele com alto volume de compras.

```
In [40]: # Vamos encontrar a data de menor volume de compras de cada cliente
         df_compra_minima = dados.groupby('IdCliente')["DataVenda"].min().reset_index()

In [41]: # Ajustamos os nomes das colunas
         df_compra_minima.columns = ['IdCliente', 'Data_Menor_Compra']

In [42]: # Vamos extrair o mês em que ocorreu o menor volume de compras de cada cliente
         df_compra_minima['Mes_Menor_Compra_Mensal'] = df_compra_minima['Data_Menor_Compra'].ma

In [43]: # Dados
         df_compra_minima.head()
```

```
Out[43]:    IdCliente   Data_Menor_Compra  Mes_Menor_Compra_Mensal
         0    12346.0 2011-01-18 10:01:00                   201101
         1    12347.0 2010-12-07 14:57:00                   201012
         2    12348.0 2010-12-16 19:09:00                   201012
         3    12349.0 2011-11-21 09:51:00                   201111
         4    12350.0 2011-02-02 16:01:00                   201102
```

```
In [44]: # Vamos fazer um merge entre o dataset original e o dataset de volume de compras
         dados_compras = pd.merge(dados, df_compra_minima, on = "IdCliente")
         dados_compras.head()
```

```
Out[44]:   NumeroFatura CodigoProduto                         NomeProduto  Quantidade  \
         0       536365         21730   GLASS STAR FROSTED T-LIGHT HOLDER           6
         1       536365        85123A  WHITE HANGING HEART T-LIGHT HOLDER           6
         2       536365         71053                 WHITE METAL LANTERN           6
         3       536365        84406B      CREAM CUPID HEARTS COAT HANGER           8
         4       536365        84029G KNITTED UNION FLAG HOT WATER BOTTLE           6

                    DataVenda  ValorUnitario  IdCliente    Pais  AnoMes  Faturamento  \
         0 2010-12-01 08:26:00           4.25    17850.0  Brasil  201012        25.50
         1 2010-12-01 08:26:00           2.55    17850.0  Brasil  201012        15.30
```

```
2 2010-12-01 08:26:00           3.39    17850.0  Brasil  201012        20.34
3 2010-12-01 08:26:00           2.75    17850.0  Brasil  201012        22.00
4 2010-12-01 08:26:00           3.39    17850.0  Brasil  201012        20.34

    Data_Menor_Compra  Mes_Menor_Compra_Mensal
0 2010-12-01 08:26:00                  201012
1 2010-12-01 08:26:00                  201012
2 2010-12-01 08:26:00                  201012
3 2010-12-01 08:26:00                  201012
4 2010-12-01 08:26:00                  201012
```

In [45]:
```
# Vamos criar uma nova coluna de tipo de usuário e preencher como Novo
dados_compras['TipoUsuario'] = 'Novo'
```

In [46]:
```
# Dados
dados_compras['TipoUsuario'].value_counts()
```

Out[46]:
```
Novo    406725
Name: TipoUsuario, dtype: int64
```

In [47]:
```
# Dados
dados_compras.head()
```

Out[47]:
```
  NumeroFatura CodigoProduto                         NomeProduto  Quantidade  \
0       536365         21730   GLASS STAR FROSTED T-LIGHT HOLDER           6
1       536365        85123A  WHITE HANGING HEART T-LIGHT HOLDER           6
2       536365         71053                 WHITE METAL LANTERN           6
3       536365        84406B      CREAM CUPID HEARTS COAT HANGER           8
4       536365        84029G KNITTED UNION FLAG HOT WATER BOTTLE           6

            DataVenda  ValorUnitario  IdCliente    Pais  AnoMes  Faturamento  \
0 2010-12-01 08:26:00           4.25    17850.0  Brasil  201012        25.50
1 2010-12-01 08:26:00           2.55    17850.0  Brasil  201012        15.30
2 2010-12-01 08:26:00           3.39    17850.0  Brasil  201012        20.34
3 2010-12-01 08:26:00           2.75    17850.0  Brasil  201012        22.00
4 2010-12-01 08:26:00           3.39    17850.0  Brasil  201012        20.34

    Data_Menor_Compra  Mes_Menor_Compra_Mensal TipoUsuario
0 2010-12-01 08:26:00                  201012        Novo
1 2010-12-01 08:26:00                  201012        Novo
2 2010-12-01 08:26:00                  201012        Novo
3 2010-12-01 08:26:00                  201012        Novo
4 2010-12-01 08:26:00                  201012        Novo
```

In [48]:
```
# Um cliente antigo é aquele cujo volume de compras no mês é maior que o volume mínim
# Se for verdadeiro, mudamos a coluna TipoUsuario para "Antigo" e se não, mantemos co
dados_compras.loc[dados_compras['AnoMes'] > dados_compras['Mes_Menor_Compra_Mensal'],
```

In [49]:
```
# Dados
dados_compras['TipoUsuario'].value_counts()
```

```
Out[49]: Antigo    287549
         Novo      119176
         Name: TipoUsuario, dtype: int64

In [50]: # Agora calculamos o faturamento por tipo de usuário por mês
         df_faturamento_user_mes = dados_compras.groupby(['AnoMes', 'TipoUsuario'])['Faturament

In [51]: # Removemos o mês 12 de 2011 pois não temos dados suficientes
         df_faturamento_user_mes = df_faturamento_user_mes.query("AnoMes != 201012 and AnoMes

In [52]: # Dados
         df_faturamento_user_mes
```

Out[52]:

| | AnoMes | TipoUsuario | Faturamento |
|---|---|---|---|
| 1 | 201101 | Antigo | 271616.520 |
| 2 | 201101 | Novo | 203457.860 |
| 3 | 201102 | Antigo | 287024.770 |
| 4 | 201102 | Novo | 149521.380 |
| 5 | 201103 | Antigo | 390034.530 |
| 6 | 201103 | Novo | 189930.080 |
| 7 | 201104 | Antigo | 306283.600 |
| 8 | 201104 | Novo | 119764.251 |
| 9 | 201105 | Antigo | 532392.340 |
| 10 | 201105 | Novo | 115858.740 |
| 11 | 201106 | Antigo | 515486.650 |
| 12 | 201106 | Novo | 92526.510 |
| 13 | 201107 | Antigo | 508355.610 |
| 14 | 201107 | Novo | 65882.871 |
| 15 | 201108 | Antigo | 538709.770 |
| 16 | 201108 | Novo | 77658.230 |
| 17 | 201109 | Antigo | 778161.781 |
| 18 | 201109 | Novo | 153278.591 |
| 19 | 201110 | Antigo | 819672.900 |
| 20 | 201110 | Novo | 154930.690 |
| 21 | 201111 | Antigo | 998176.360 |
| 22 | 201111 | Novo | 134231.380 |

### 3.4.13 Visualização do Indicador 6

```
In [53]: # Plot

         # Definição dos dados no plot
         plot_data = [go.Scatter(x = df_faturamento_user_mes.query("TipoUsuario == 'Antigo'")[
                                 y = df_faturamento_user_mes.query("TipoUsuario == 'Antigo'")[
                                 name = 'Cliente Antigo'),
                      go.Scatter(x = df_faturamento_user_mes.query("TipoUsuario == 'Novo'")['An
                                 y = df_faturamento_user_mes.query("TipoUsuario == 'Novo'")['Fa
                                 name = 'Cliente Novo')]
```

```
        # Layout
        plot_layout = go.Layout(xaxis = {"type": "category"},
                                title = 'Diferença de Faturamento ao Longo do Tempo Entre Cli

        # Plot da figura
        fig = go.Figure(data = plot_data, layout = plot_layout)
        pyoff.iplot(fig)
```

### 3.4.14 Taxa de Novos Clientes

Como definimos clientes novos e antigos no indicador 6, agora podemos usar os dados e calcular a proporção de novos clientes ao longo do tempo.

```
In [54]: # Calcula a taxa de novos clientes
         df_taxa_novos_clientes = dados_compras.query("TipoUsuario == 'Novo'").groupby(['AnoMes

In [55]: # Ajustamos índice e removemos valores ausentes
         df_taxa_novos_clientes = df_taxa_novos_clientes.reset_index()
         df_taxa_novos_clientes = df_taxa_novos_clientes.dropna()

In [56]: # Dados
         df_taxa_novos_clientes
```

Out[56]:

| | AnoMes | IdCliente |
|---|---|---|
| 1 | 201101 | 1.162983 |
| 2 | 201102 | 0.909091 |
| 3 | 201103 | 0.758621 |
| 4 | 201104 | 0.498333 |
| 5 | 201105 | 0.348750 |
| 6 | 201106 | 0.287990 |
| 7 | 201107 | 0.238155 |
| 8 | 201108 | 0.205412 |
| 9 | 201109 | 0.296813 |
| 10 | 201110 | 0.328052 |
| 11 | 201111 | 0.230935 |
| 12 | 201112 | 0.063863 |

### 3.4.15 Visualização do Indicador 7

```
In [57]: # Plot

         # Definição dos dados no plot
         plot_data = [go.Bar(x = df_taxa_novos_clientes.query("AnoMes > 201101 and AnoMes < 20
                             y = df_taxa_novos_clientes.query("AnoMes > 201101 and AnoMes < 20

         # Layout
         plot_layout = go.Layout(xaxis = {"type": "category"},
                                 title = 'Taxa de Novos Clientes')
```

```
# Plot da figura
fig = go.Figure(data = plot_data, layout = plot_layout)
pyoff.iplot(fig)
```

### 3.4.16 Taxa Mensal de Retenção de Clientes

Taxa Mensal de Retenção de Clientes = Clientes do Mês Anterior / Total de Clientes Ativos

```
In [58]: # Agrupamos os dados por cliente e mês e somamos o faturamento
         dados_compras_clientes = dados_compras.groupby(['IdCliente', 'AnoMes'])['Faturamento']

In [59]: # Dados
         dados_compras_clientes.head()
```

Out[59]:

| | IdCliente | AnoMes | Faturamento |
|---|---|---|---|
| 0 | 12346.0 | 201101 | 0.00 |
| 1 | 12347.0 | 201012 | 711.79 |
| 2 | 12347.0 | 201101 | 475.39 |
| 3 | 12347.0 | 201104 | 636.25 |
| 4 | 12347.0 | 201106 | 382.52 |

```
In [60]: # Agora definimos a retenção com uma tabela cruzada
         df_ret = pd.crosstab(dados_compras_clientes['IdCliente'], dados_compras_clientes['Anol

In [61]: # Dados
         df_ret.head()
```

Out[61]:

| AnoMes | IdCliente | 201012 | 201101 | 201102 | 201103 | 201104 | 201105 | 201106 | \ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 12346.0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 12347.0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | |
| 2 | 12348.0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | |
| 3 | 12349.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 4 | 12350.0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | |

| AnoMes | 201107 | 201108 | 201109 | 201110 | 201111 | 201112 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 |

```
In [62]: # Extraímos os meses
         meses = df_ret.columns[2:]
         meses

Out[62]: Index([201101, 201102, 201103, 201104, 201105, 201106, 201107, 201108, 201109,
                201110, 201111, 201112],
               dtype='object', name='AnoMes')
```

```
In [63]: # O loop abaixo vai calcular a retenção ao longo dos meses

         # Lista para gravar o resulatdo
         lista_ret = []

         # Loop
         for i in range(len(meses)-1):
             dados_retencao = {}
             mes_corrente = meses[i+1]
             mes_anterior = meses[i]
             dados_retencao['AnoMes'] = int(mes_corrente)
             dados_retencao['TotalUser'] = df_ret[mes_corrente].sum()
             dados_retencao['TotalRetido'] = df_ret[(df_ret[mes_corrente] > 0) & (df_ret[mes_a
             lista_ret.append(dados_retencao)

         lista_ret

Out[63]: [{'AnoMes': 201102, 'TotalUser': 798, 'TotalRetido': 299},
          {'AnoMes': 201103, 'TotalUser': 1020, 'TotalRetido': 345},
          {'AnoMes': 201104, 'TotalUser': 899, 'TotalRetido': 346},
          {'AnoMes': 201105, 'TotalUser': 1079, 'TotalRetido': 399},
          {'AnoMes': 201106, 'TotalUser': 1051, 'TotalRetido': 464},
          {'AnoMes': 201107, 'TotalUser': 993, 'TotalRetido': 415},
          {'AnoMes': 201108, 'TotalUser': 980, 'TotalRetido': 433},
          {'AnoMes': 201109, 'TotalUser': 1302, 'TotalRetido': 465},
          {'AnoMes': 201110, 'TotalUser': 1425, 'TotalRetido': 552},
          {'AnoMes': 201111, 'TotalUser': 1711, 'TotalRetido': 690},
          {'AnoMes': 201112, 'TotalUser': 683, 'TotalRetido': 440}]

In [64]: # Dados
         df_ret_final = pd.DataFrame(lista_ret)
         df_ret_final.head()

Out[64]:    AnoMes  TotalRetido  TotalUser
         0  201102          299        798
         1  201103          345       1020
         2  201104          346        899
         3  201105          399       1079
         4  201106          464       1051
```

Agora calculamos a proporção para encontrar o indicador.

```
In [65]: # Calculamos o indicador
         df_ret_final['TaxaRetencao'] = df_ret_final['TotalRetido'] / df_ret_final['TotalUser']
         df_ret_final

Out[65]:    AnoMes  TotalRetido  TotalUser  TaxaRetencao
         0  201102          299        798      0.374687
         1  201103          345       1020      0.338235
```

```
2   201104          346         899      0.384872
3   201105          399        1079      0.369787
4   201106          464        1051      0.441484
5   201107          415         993      0.417925
6   201108          433         980      0.441837
7   201109          465        1302      0.357143
8   201110          552        1425      0.387368
9   201111          690        1711      0.403273
10  201112          440         683      0.644217
```

### 3.4.17 Visualização do Indicador 8

In [66]:
```
# Plot

# Definição dos dados no plot
plot_data = [go.Scatter(x = df_ret_final.query("AnoMes < 201112")['AnoMes'],
                        y = df_ret_final.query("AnoMes < 201112")['TaxaRetencao'],
                        name = "taxa")]

# Layout
plot_layout = go.Layout(xaxis = {"type": "category"},
                        title = 'Taxa Mensal de Retenção de Clientes')

# Plot da figura
fig = go.Figure(data = plot_data, layout = plot_layout)
pyoff.iplot(fig)
```