AnaliseShopping

November 10, 2024

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[6]:
```
sales_data = pd.read_excel('sales_data.xlsx')
customers = pd.read_excel('customer_data.xlsx')
mall_data = pd.read_excel('shopping_mall_data.xlsx')
```

[7]:
```
sales_data.info()
sales_data.isna().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99457 entries, 0 to 99456
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   invoice_no     99457 non-null  object 
 1   customer_id    99457 non-null  object 
 2   category       99457 non-null  object 
 3   quantity       99457 non-null  int64  
 4   invoice date   99457 non-null  object 
 5   price          99457 non-null  float64
 6   shopping_mall  99457 non-null  object 
dtypes: float64(1), int64(1), object(5)
memory usage: 5.3+ MB
```

[7]:
```
invoice_no       0
customer_id      0
category         0
quantity         0
invoice date     0
price            0
shopping_mall    0
dtype: int64
```

0.1 Análise de Vendas e Desempenho de produtos

0.1.1 Análise por Categoria e Shopping Mall: Avaliar o desempenho de vendas de cada categoria de produto (como roupas ou sapatos) em diferentes shoppings. Isso pode revelar quais produtos têm mais aceitação em determinadas localidades ou tipos de shoppings.

[8]:
```
# Agrupando e somando as vendas por shopping e categoria
sales_pivot = sales_data.pivot_table(index='shopping_mall', columns='category',␣
 ↪values='quantity', aggfunc=sum)
sales_pivot
```

```
C:\Users\angel\AppData\Local\Temp\ipykernel_13904\839806654.py:2: FutureWarning:
The provided callable <built-in function sum> is currently using
DataFrameGroupBy.sum. In a future version of pandas, the provided callable will
be used directly. To keep current behavior pass the string "sum" instead.
  sales_pivot = sales_data.pivot_table(index='shopping_mall',
columns='category', values='quantity', aggfunc=sum)
```

[8]:

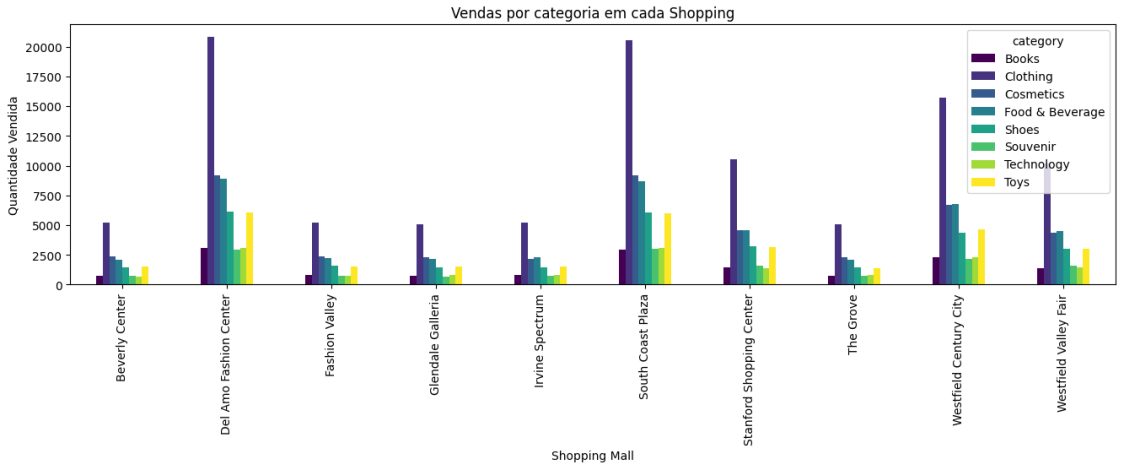
category	Books	Clothing	Cosmetics	Food & Beverage	Shoes	\
shopping_mall						
Beverly Center	756	5239	2342	2072	1459	
Del Amo Fashion Center	3099	20813	9193	8878	6112	
Fashion Valley	809	5228	2385	2216	1589	
Glendale Galleria	720	5101	2279	2186	1471	
Irvine Spectrum	792	5180	2174	2293	1473	
South Coast Plaza	2969	20513	9155	8695	6065	
Stanford Shopping Center	1468	10552	4569	4586	3237	
The Grove	730	5038	2272	2109	1452	
Westfield Century City	2271	15729	6700	6764	4349	
Westfield Valley Fair	1368	10165	4396	4478	3010	

category	Souvenir	Technology	Toys
shopping_mall			
Beverly Center	775	673	1536
Del Amo Fashion Center	2921	3067	6031
Fashion Valley	716	765	1526
Glendale Galleria	651	784	1524
Irvine Spectrum	708	780	1549
South Coast Plaza	3025	3050	5985
Stanford Shopping Center	1586	1396	3136
The Grove	726	795	1379
Westfield Century City	2197	2273	4611
Westfield Valley Fair	1566	1438	3044

[9]:
```
# Plotando um gráfico para visualizar os dados referentes a vendas por shopping␣
 ↪e categoria
```

```
sales_pivot.plot(kind='bar', figsize=(16,4), colormap='viridis', title='Vendas␣
↪por categoria em cada Shopping', xlabel='Shopping Mall', ylabel='Quantidade␣
↪Vendida' )
```

[9]:
```
<Axes: title={'center': 'Vendas por categoria em cada Shopping'},
xlabel='Shopping Mall', ylabel='Quantidade Vendida'>
```



Todos os Shoppings parecem manter um padrão em relação as categorias, Roupas lideram em todos, Cosméticos e Alimentos se mantém quase sempre no mesmo nível

0.1.2 Análise de tendências temporais: Examinar como as vendas variam ao longo do tempo

[22]:
```
sales_data.head()
```

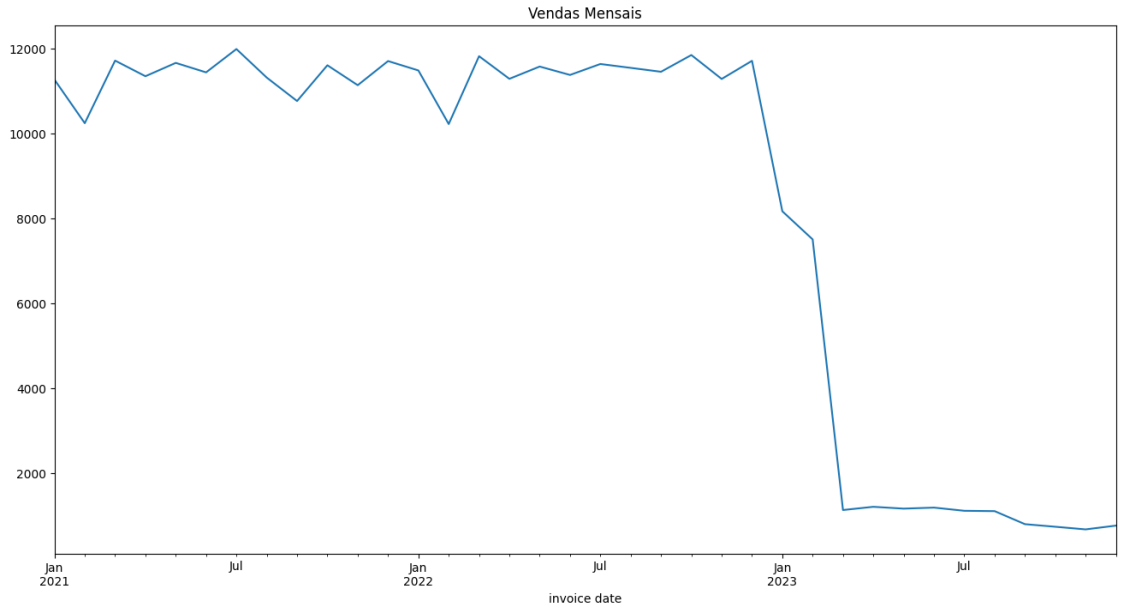
[22]:
```
             invoice_no customer_id  category  quantity    price  \
invoice date
2022-05-08      I138884     C241288  Clothing         5  1500.40
2021-12-12      I317333     C111565     Shoes         3  1800.51
2021-09-11      I127801     C266599  Clothing         1   300.08
2021-05-16      I173702     C988172     Shoes         5  3000.85
2021-10-24      I337046     C189076     Books         4    60.60

                        shopping_mall
invoice date
2022-05-08           South Coast Plaza
2021-12-12              Beverly Center
2021-09-11      Westfield Century City
2021-05-16    Stanford Shopping Center
2021-10-24           South Coast Plaza
```

[24]:
```
#sales_data['invoice date'] = pd.to_datetime(sales_data['invoice date'])
#sales_data.set_index('invoice date', inplace=True)


monthly_sales = sales_data.resample('ME').quantity.sum()

monthly_sales.plot(title='Vendas Mensais', figsize=(16,8))
plt.show()
```



0.2 Segmentação de clientes e Análise de comportamento

0.2.1 Perfil dos clientes : Criar segmentos dos clientes baseado em idade, genêro e metodos de pagamento

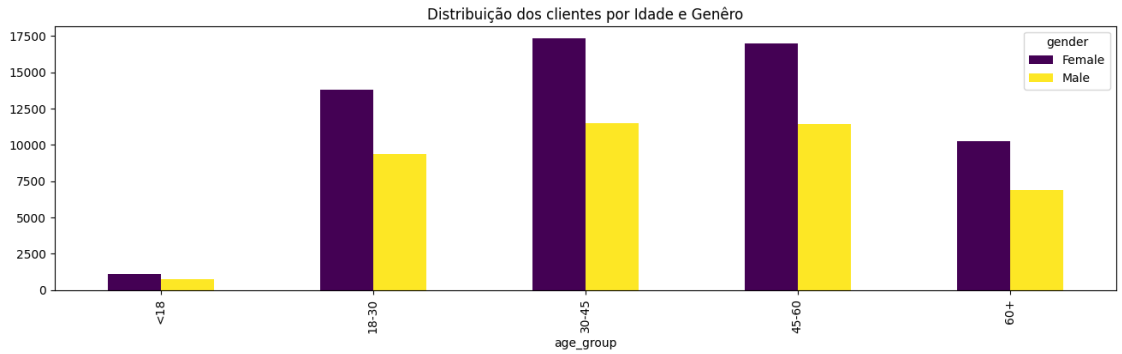
[28]:
```
customers['age_group'] = pd.cut(customers['age'], bins=[0,18,30,45,60,100],␣
 ↪labels=["<18", '18-30', '30-45', '45-60','60+'])
profile = customers.groupby(['age_group', 'gender']).size().unstack()

print(profile)
profile.plot(kind='bar', title='Distribuição dos clientes por Idade e Genêro',␣
 ↪figsize=(16,4),colormap='viridis')
plt.show()
```

```
gender      Female   Male
age_group
<18           1078    765
18-30        13814   9338
30-45        17320  11476
```

```
45-60        16959   11441
60+          10241    6906
```

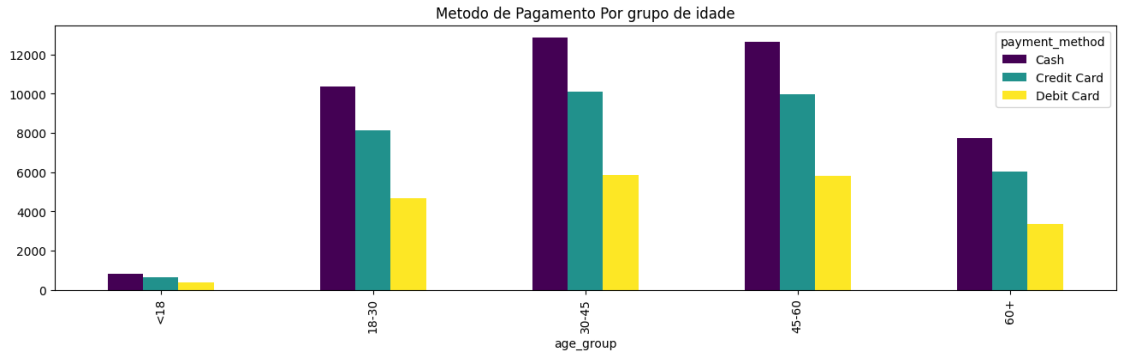
```
C:\Users\angel\AppData\Local\Temp\ipykernel_13904\1067937034.py:2:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  profile = customers.groupby(['age_group', 'gender']).size().unstack()
```



0.2.2 Preferência do método de Pagamento

[30]:
```
payment_method_distribution = customers.groupby(['age_group',␣
  ↪'payment_method']).size().unstack()
payment_method_distribution.plot(kind='bar',title='Metodo de Pagamento Por␣
  ↪grupo de idade',colormap='viridis', figsize=(16,4) )
plt.show()
```

```
C:\Users\angel\AppData\Local\Temp\ipykernel_13904\923264017.py:1: FutureWarning:
The default of observed=False is deprecated and will be changed to True in a
future version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  payment_method_distribution = customers.groupby(['age_group',
'payment_method']).size().unstack()
```



0.3 Análise avançada de vendas por cliente: Calcularemos o número de compras realizadas por cada cliente para identificar quem são os clientes recorrentes

[41]:
```
#Contar o número de transações por cliente
customer_purchase_count = sales_data.groupby('customer_id')['invoice_no'].
 ↪nunique()
print(customer_purchase_count)

#Definir Cliente como recorrente se fez mais de uma compra
recurring_customers = customer_purchase_count[customer_purchase_count > 1]

recurring_customers_percentage = len(recurring_customers) /␣
 ↪len(customer_purchase_count) * 100

print(f'Porcentagem de clientes recorrentes:␣
 ↪{round(recurring_customers_percentage,1)}%')
```

```
customer_id
C100004    1
C100005    1
C100006    1
C100012    1
C100019    1
          ..
C999886    1
C999910    1
C999974    1
C999976    1
C999995    1
Name: invoice_no, Length: 99457, dtype: int64
Porcentagem de clientes recorrentes: 0.0%
```

0.3.1 Cálculo do LTV(LifeTimeValue) dos clientes

[43]:
```
sales_data['total_price'] = sales_data['quantity'] * sales_data['price']
customer_ltv = sales_data.groupby('customer_id').total_price.sum()
print(customer_ltv.head())
```

```
customer_id
C100004    7502.00
C100005    2400.68
C100006     322.56
C100012     130.75
C100019      35.84
Name: total_price, dtype: float64
```

0.4 Análise integrada entre customers e sales

[45]:
```
# Merge entre costumers e sales
merged_data = pd.merge(sales_data,customers, on='customer_id', how='inner')

print(merged_data.head())
```

```
  invoice_no customer_id  category  quantity    price  \
0    I138884     C241288  Clothing         5  1500.40
1    I317333     C111565     Shoes         3  1800.51
2    I127801     C266599  Clothing         1   300.08
3    I173702     C988172     Shoes         5  3000.85
4    I337046     C189076     Books         4    60.60

              shopping_mall  total_price  gender   age payment_method  \
0         South Coast Plaza      7502.00  Female  28.0    Credit Card
1            Beverly Center      5401.53    Male  21.0     Debit Card
2    Westfield Century City       300.08    Male  20.0           Cash
3  Stanford Shopping Center     15004.25  Female  66.0    Credit Card
4         South Coast Plaza       242.40  Female  53.0           Cash

  age_group
0     18-30
1     18-30
2     18-30
3       60+
4     45-60
```

0.4.1 Análisando o gasto medio por faixa etária

[49]:
```
#Calculando o total gasto por cliente
merged_data['total_spent'] = merged_data['price'] * merged_data['quantity']

#Criando faixa etárias
merged_data['age_group'] = pd.cut(merged_data['age'], bins=[0,18,30,45,60,100],␣
 ↪labels=['0-18', '19-30', '31-45', '46-60', '60+'])

#Gasto médio por faixa etária
avg_spent_age_group = merged_data.groupby('age_group',␣
 ↪observed=False)['total_spent'].mean()
print('Gasto médio por faixa etária: ')
print(round(avg_spent_age_group,1))
```

```
Gasto médio por faixa etária:
age_group
0-18     2384.7
19-30    2530.0
31-45    2537.2
```

```
46-60    2525.9
60+      2534.7
Name: total_spent, dtype: float64
```

0.4.2 Análise de Preferência por categoria

[53]:
```
# Agrupando por categoria de produto e genero, somando as quantidades compradas
category_preference = merged_data.groupby(['gender', 'category'],␣
 ↪observed=False).quantity.sum().unstack()

print(f'Preferencia de categoria por genero{category_preference}')
```

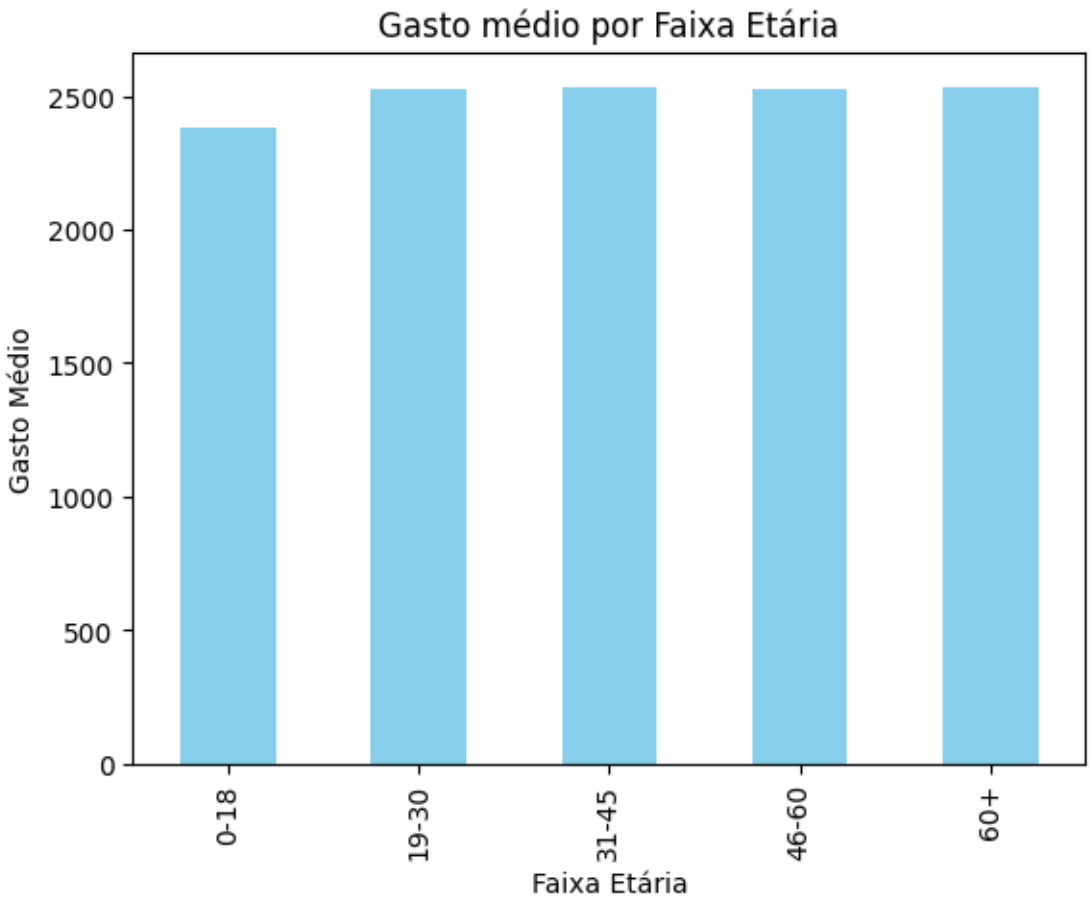
```
Preferencia de categoria por generocategory  Books  Clothing  Cosmetics  Food &
Beverage  Shoes  Souvenir  \
gender
Female     8776     62039      27261            26362  17906      8976
Male       6206     41519      18204            17915  12311      5895

category  Technology   Toys
gender
Female          8977  18362
Male            6044  11959
```

Plotando os resultados

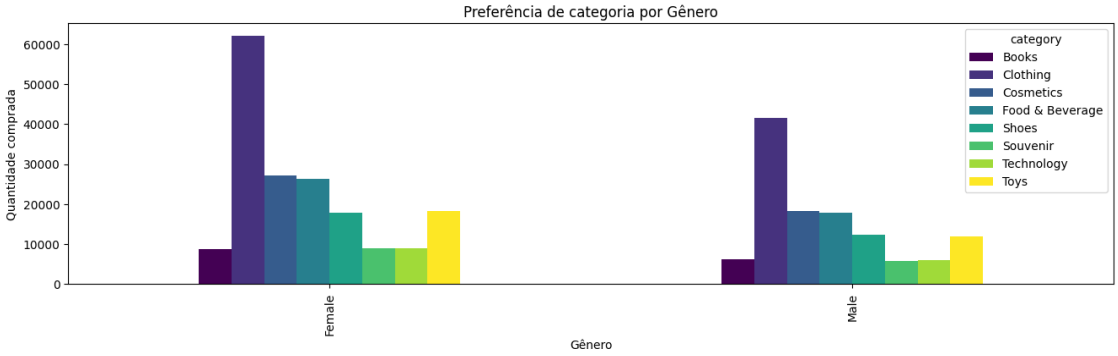
0.4.3 Visualizando o gasto médio por faixa etária

[54]:
```
avg_spent_age_group.plot(kind='bar', title='Gasto médio por Faixa Etária',␣
 ↪color='skyblue')
plt.xlabel('Faixa Etária')
plt.ylabel('Gasto Médio')
plt.show()
```



0.4.4 Visualizando as preferências de categoria por gênero

[57]:
```
category_preference.plot(kind='bar', title='Preferência de categoria por␣
↪Gênero', figsize=(16,4),colormap='viridis')
plt.xlabel("Gênero")
plt.ylabel("Quantidade comprada")
plt.show()
```



0.5 Modelagem e Previsão

0.5.1 Modelo de Previsão de Vendas

[87]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error,␣
 ↪root_mean_squared_error, r2_score, mean_absolute_percentage_error
```

[80]:
```
# Preparação dos dados
sales_data['month'] = sales_data.index.month
x = sales_data[['month']]
y = sales_data[['quantity']]

#Dividirem conjunto de treino e teste
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2,␣
 ↪random_state=0)



#treinar o modelo
model = LinearRegression()
model.fit(x_train, y_train)

#Previsoes

y_pred = model.predict(x_test)
print(y_pred)
```

```
[[2.99809114]
 [2.99809114]
 [3.00247418]
 …
 [2.99370811]
 [3.00247418]
 [2.99370811]]
```

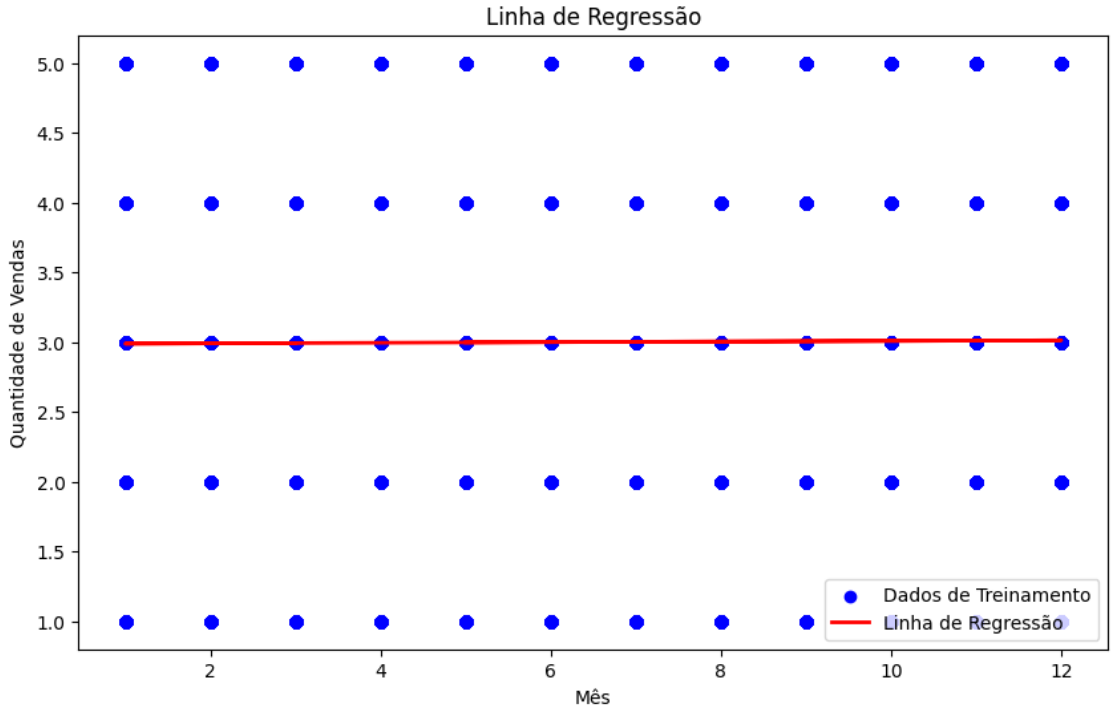
0.5.2 Avaliando o Modelo

[84]:
```
# Plotando a linha de regressão
plt.figure(figsize=(10, 6))

# Plotando os dados reais
plt.scatter(x_train, y_train, color='blue', label='Dados de Treinamento')

# Plotando a linha de regressão
plt.plot(x_test, y_pred, color='red', linewidth=2, label='Linha de Regressão')
```

```
plt.title('Linha de Regressão')
plt.xlabel('Mês')
plt.ylabel('Quantidade de Vendas')
plt.legend()
plt.show()
```



[88]:
```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = root_mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test,y_pred)


print(f"Erro Médio Absoluto (MAE): {mae:.2f}")
print(f"Coeficiente de Determinação (R²): {r2:.2f}")
print(f"Raiz do Erro Quadrático Médio (RMSE): {rmse:.2f}")
print(f"Erro Quadrático Médio (MSE): {mse:.2f}")
print(f"Erro Absoluto Médio Percentual (MAPE): {mape:.2f}%")
```

```
Erro Médio Absoluto (MAE): 1.20
Coeficiente de Determinação (R²): -0.00
Raiz do Erro Quadrático Médio (RMSE): 1.42
Erro Quadrático Médio (MSE): 2.00
```

```
Erro Absoluto Médio Percentual (MAPE): 0.63%
```

0.5.3 Conclusao

R2 : O modelo não parece estar se ajustando bem aos dados, a escolha do modelo poderia passar por uma revisão MAE, RMSE, MSE: as métricas são razoáveis, mas o R score negativo sugere que o modelo nao capta a dinâmica dos dados completamente MAPE: 0.63% é excelente, o que significa que, apesar de não explicar bem os dados, o modelo é preciso na maioria das previsões individuais. Podemos entao tentar ajusar o modelo de alguma maneira melhor.