In [1]:
```python
import pandas as pd
import numpy as np
import plotly.express as px
from matplotlib import pyplot as plt
import seaborn as sns
pd.options.display.float_format = '{:,.2f}'.format
pd.options.display.max_rows = None
pd.options.display.max_columns = None
```

In [2]:
```python
df = pd.read_csv('supermarket_sales.csv')
```

In [3]:
```python
df.head()
```

Out[3]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rating
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.14	548.97	1/5/2019	13:08	Ewallet	522.83	4.76	26.14	9.10
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.82	80.22	3/8/2019	10:29	Cash	76.40	4.76	3.82	9.60
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.22	340.53	3/3/2019	13:23	Credit card	324.31	4.76	16.22	7.40
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.29	489.05	1/27/2019	20:33	Ewallet	465.76	4.76	23.29	8.40
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.21	634.38	2/8/2019	10:37	Ewallet	604.17	4.76	30.21	5.30

In [4]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Invoice ID               1000 non-null   object 
 1   Branch                   1000 non-null   object 
 2   City                     1000 non-null   object 
 3   Customer type            1000 non-null   object 
 4   Gender                   1000 non-null   object 
 5   Product line             1000 non-null   object 
 6   Unit price               1000 non-null   float64
 7   Quantity                 1000 non-null   int64  
 8   Tax 5%                   1000 non-null   float64
 9   Total                    1000 non-null   float64
 10  Date                     1000 non-null   object 
 11  Time                     1000 non-null   object 
 12  Payment                  1000 non-null   object 
 13  cogs                     1000 non-null   float64
 14  gross margin percentage  1000 non-null   float64
 15  gross income             1000 non-null   float64
 16  Rating                   1000 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB
```

In [5]:
```python
df.columns
```

Out[5]:
```
Index(['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender',
       'Product line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Date',
       'Time', 'Payment', 'cogs', 'gross margin percentage', 'gross income',
       'Rating'],
      dtype='object')
```

Cleaning Data

In [6]:
```python
cols = ['Branch', 'City', 'Customer type', 'Gender', 'Product line', 'Payment']
```

In [7]:
```python
for col in cols:
    print(f'{col} unique Values = {df[col].unique()}')
    print('-' * 50 )
```

```
Branch unique Values = ['A' 'C' 'B']
--------------------------------------------------
City unique Values = ['Yangon' 'Naypyitaw' 'Mandalay']
--------------------------------------------------
Customer type unique Values = ['Member' 'Normal']
--------------------------------------------------
Gender unique Values = ['Female' 'Male']
--------------------------------------------------
Product line unique Values = ['Health and beauty' 'Electronic accessories' 'Home and lifestyle'
 'Sports and travel' 'Food and beverages' 'Fashion accessories']
--------------------------------------------------
Payment unique Values = ['Ewallet' 'Cash' 'Credit card']
--------------------------------------------------
```

In [8]:
```python
df.Date = pd.to_datetime(df.Date)
df.Time = pd.to_datetime(df.Time)
```

```
C:\Users\Mo_2022\AppData\Local\Temp\ipykernel_772\1287169114.py:2: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  df.Time = pd.to_datetime(df.Time)
```

In [9]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype         
---  ------                   --------------  -----         
 0   Invoice ID               1000 non-null   object        
 1   Branch                   1000 non-null   object        
 2   City                     1000 non-null   object        
 3   Customer type            1000 non-null   object        
 4   Gender                   1000 non-null   object        
 5   Product line             1000 non-null   object        
 6   Unit price               1000 non-null   float64       
 7   Quantity                 1000 non-null   int64         
 8   Tax 5%                   1000 non-null   float64       
 9   Total                    1000 non-null   float64       
 10  Date                     1000 non-null   datetime64[ns]
 11  Time                     1000 non-null   datetime64[ns]
 12  Payment                  1000 non-null   object        
 13  cogs                     1000 non-null   float64       
 14  gross margin percentage  1000 non-null   float64       
 15  gross income             1000 non-null   float64       
 16  Rating                   1000 non-null   float64       
dtypes: datetime64[ns](2), float64(7), int64(1), object(7)
memory usage: 132.9+ KB
```

In [10]:
```python
df.columns = df.columns.str.lower().str.strip()
```

In [11]:
```python
df.columns
```

Out[11]:
```
Index(['invoice id', 'branch', 'city', 'customer type', 'gender',
       'product line', 'unit price', 'quantity', 'tax 5%', 'total', 'date',
       'time', 'payment', 'cogs', 'gross margin percentage', 'gross income',
       'rating'],
      dtype='object')
```

In [12]:
```python
df.sort_values(by= 'invoice id').head(10)
```

Out[12]:

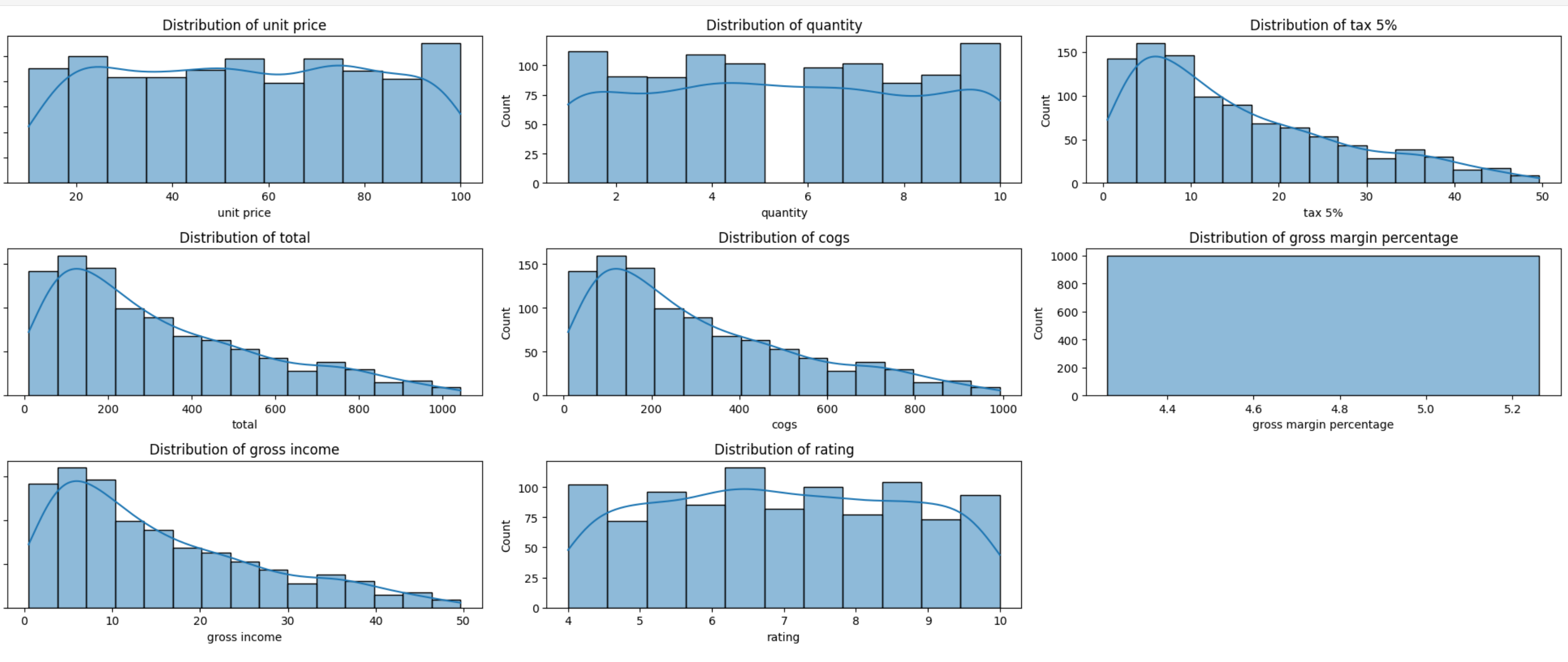
	invoice id	branch	city	customer type	gender	product line	unit price	quantity	tax 5%	total	date	time	payment	cogs	gross margin percentage	gross income	rating
162	101-17-6199	A	Yangon	Normal	Male	Food and beverages	45.79	7	16.03	336.56	2019-03-13	2024-02-23 19:44:00	Credit card	320.53	4.76	16.03	7.00
867	101-81-4070	C	Naypyitaw	Member	Female	Health and beauty	62.82	2	6.28	131.92	2019-01-17	2024-02-23 12:36:00	Ewallet	125.64	4.76	6.28	4.90
778	102-06-2002	C	Naypyitaw	Member	Male	Sports and travel	25.25	5	6.31	132.56	2019-03-20	2024-02-23 17:52:00	Cash	126.25	4.76	6.31	6.10
776	102-77-2261	C	Naypyitaw	Member	Male	Health and beauty	65.31	7	22.86	480.03	2019-03-05	2024-02-23 18:02:00	Credit card	457.17	4.76	22.86	4.20
683	105-10-6182	A	Yangon	Member	Male	Fashion accessories	21.48	2	2.15	45.11	2019-02-27	2024-02-23 12:22:00	Ewallet	42.96	4.76	2.15	6.60
169	105-31-1824	A	Yangon	Member	Male	Sports and travel	69.52	7	24.33	510.97	2019-02-01	2024-02-23 15:10:00	Credit card	486.64	4.76	24.33	8.50
56	106-35-6779	A	Yangon	Member	Male	Home and lifestyle	44.34	2	4.43	93.11	2019-03-27	2024-02-23 11:26:00	Cash	88.68	4.76	4.43	5.80
67	109-28-2512	B	Mandalay	Member	Female	Fashion accessories	97.61	6	29.28	614.94	2019-01-07	2024-02-23 15:01:00	Ewallet	585.66	4.76	29.28	9.90
824	109-86-4363	B	Mandalay	Member	Female	Sports and travel	60.08	7	21.03	441.59	2019-02-14	2024-02-23 11:36:00	Credit card	420.56	4.76	21.03	4.50
400	110-05-6330	C	Naypyitaw	Normal	Female	Food and beverages	39.43	6	11.83	248.41	2019-03-25	2024-02-23 20:18:00	Credit card	236.58	4.76	11.83	9.40

Univariate Analysis

In [13]:
```python
for col in df.iloc[:, 1:].select_dtypes('object'):
    print(f"{col}:  {df[col].unique()}")
```

```
branch:  ['A' 'C' 'B']
city:  ['Yangon' 'Naypyitaw' 'Mandalay']
customer type:  ['Member' 'Normal']
gender:  ['Female' 'Male']
product line:  ['Health and beauty' 'Electronic accessories' 'Home and lifestyle'
 'Sports and travel' 'Food and beverages' 'Fashion accessories']
payment:  ['Ewallet' 'Cash' 'Credit card']
```

In [14]:
```python
# Numerical Columns
n_cols = df.select_dtypes('number')
plt.figure(figsize=(20, 8))
for e, i in enumerate(n_cols.columns):
    plt.subplot(3, 3, e+1)
    sns.histplot(n_cols[i], kde= True)
    plt.title(f'Distribution of {i}');
    plt.tight_layout();
```

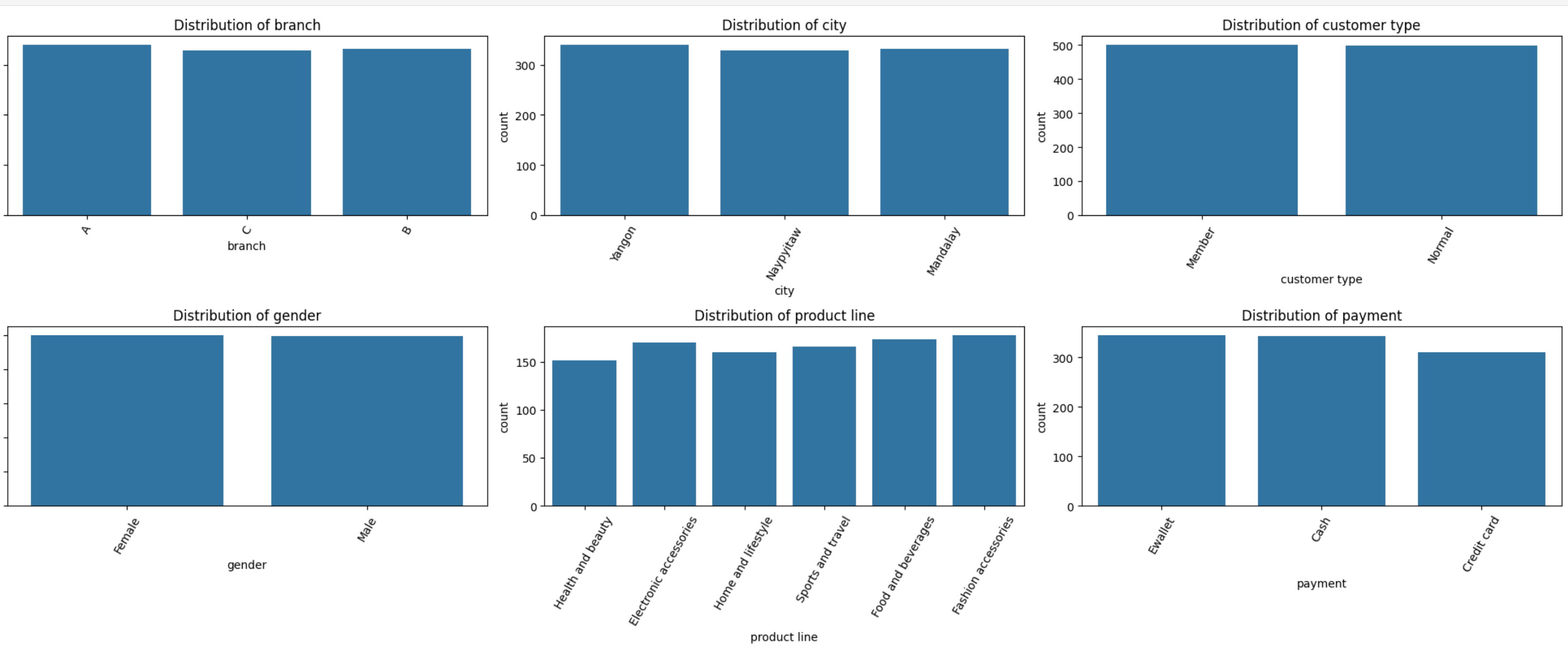


In [15]:
```python
df.select_dtypes('number').describe().T
```

Out[15]:

	count	mean	std	min	25%	50%	75%	max
unit price	1,000.00	55.67	26.49	10.08	32.88	55.23	77.94	99.96
quantity	1,000.00	5.51	2.92	1.00	3.00	5.00	8.00	10.00
tax 5%	1,000.00	15.38	11.71	0.51	5.92	12.09	22.45	49.65
total	1,000.00	322.97	245.89	10.68	124.42	253.85	471.35	1,042.65
cogs	1,000.00	307.59	234.18	10.17	118.50	241.76	448.91	993.00
gross margin percentage	1,000.00	4.76	0.00	4.76	4.76	4.76	4.76	4.76
gross income	1,000.00	15.38	11.71	0.51	5.92	12.09	22.45	49.65
rating	1,000.00	6.97	1.72	4.00	5.50	7.00	8.50	10.00

In [16]:
```python
# Object Columns
s_cols = df.select_dtypes('object')
s_cols = s_cols.iloc[:, 1:]
plt.figure(figsize= (20, 8))
for e, i in enumerate(s_cols.columns):
    plt.subplot(2, 3, e+1)
    sns.countplot(data= s_cols, x= i)
    plt.title(f'Distribution of {i}')
    plt.xticks(rotation= 60)
    plt.tight_layout();
```



In [17]:
```python
# How many M/F in each branch
grouping = df.groupby(['branch', 'gender'], as_index= False)['total'].count()
grouping
```

Out[17]:

	branch	gender	total
0	A	Female	161
1	A	Male	179
2	B	Female	162
3	B	Male	170
4	C	Female	178
5	C	Male	150

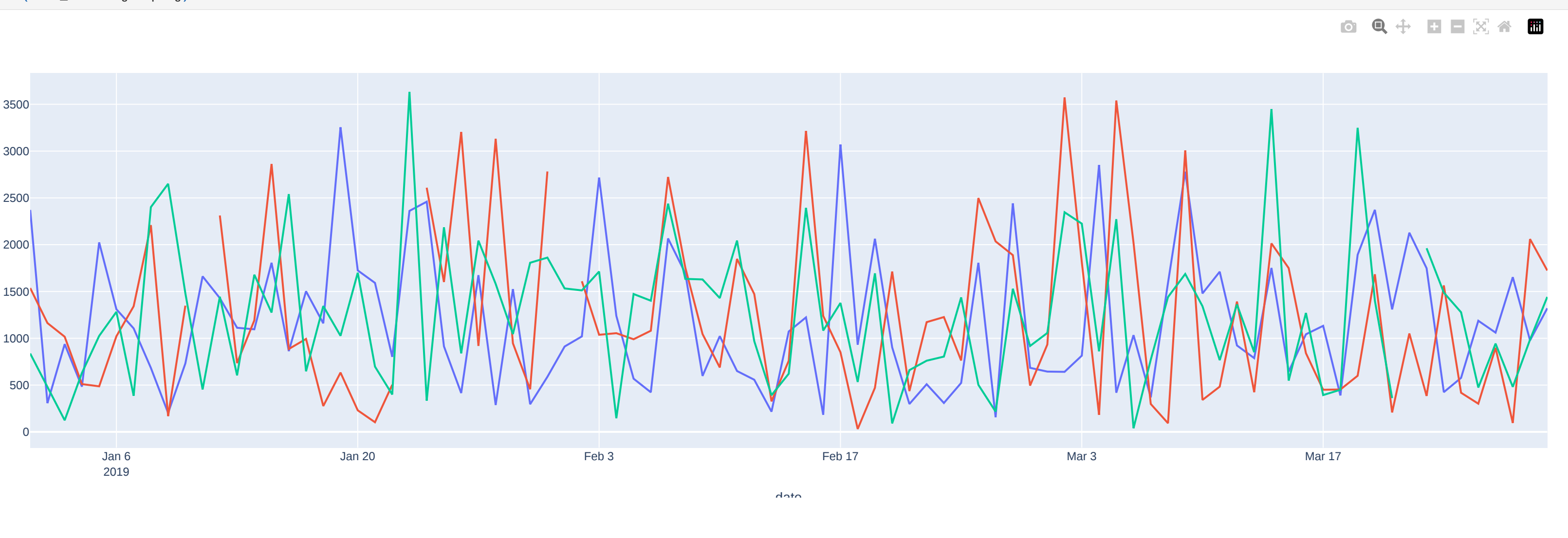
In [18]:
```python
px.bar(data_frame= grouping, x= 'gender', y= 'total', color= 'branch', title= 'Number of M/F in each branch')
```

We can notice that the number of men is greater than woman except branch 'C'. so it is interesting if we can study further why

In [19]:
```python
grouping = df[['branch', 'total', 'date']]
#grouping = grouping.set_index('date')
grouping = grouping.groupby(['date', 'branch'], as_index= False)['total'].sum()

grouping = grouping.pivot(index= 'date', columns= 'branch', values= 'total')

# grouping.plot(figsize= (15, 8))
px.line(data_frame= grouping)
```



We need to observe the count of invoices, therefore we need to normalize the totle to start with 1, to do this we will divide the totle of each column by one day >> Norm = df/ df.loc[0, :]

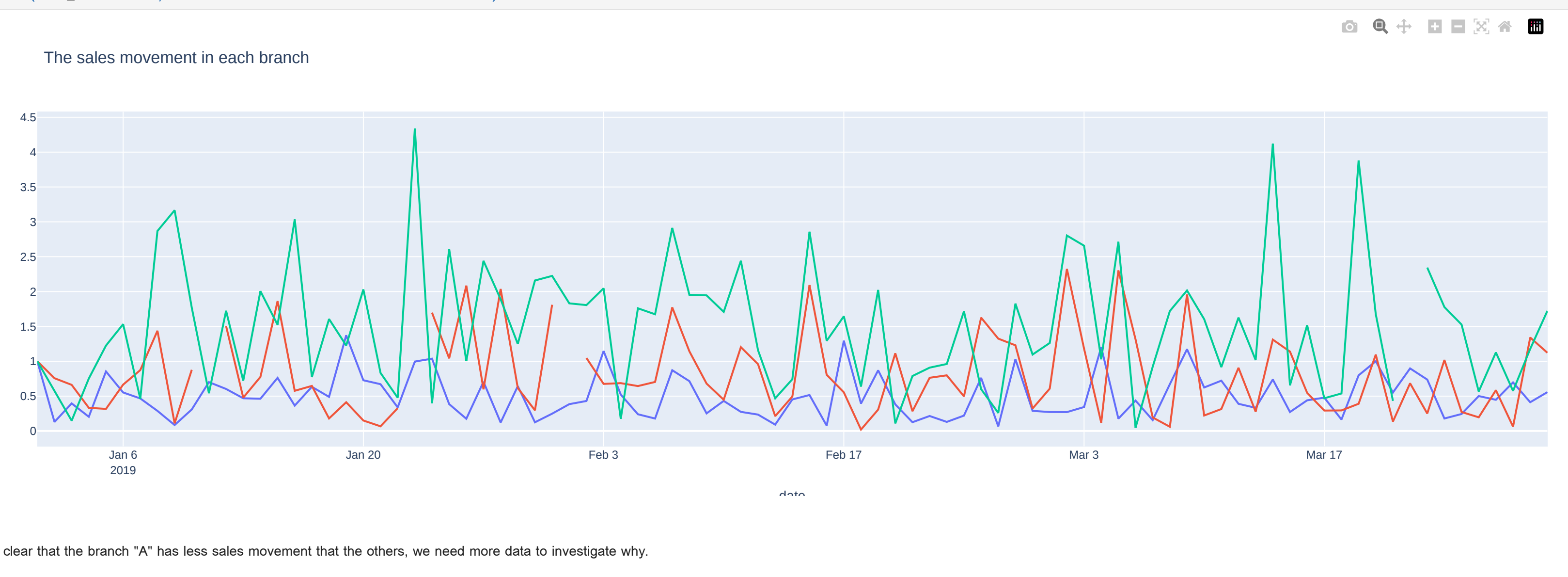
In [20]:
```python
norm = grouping / grouping.iloc[0, :]
```

In [21]:
```python
norm.head()
```

Out[21]:

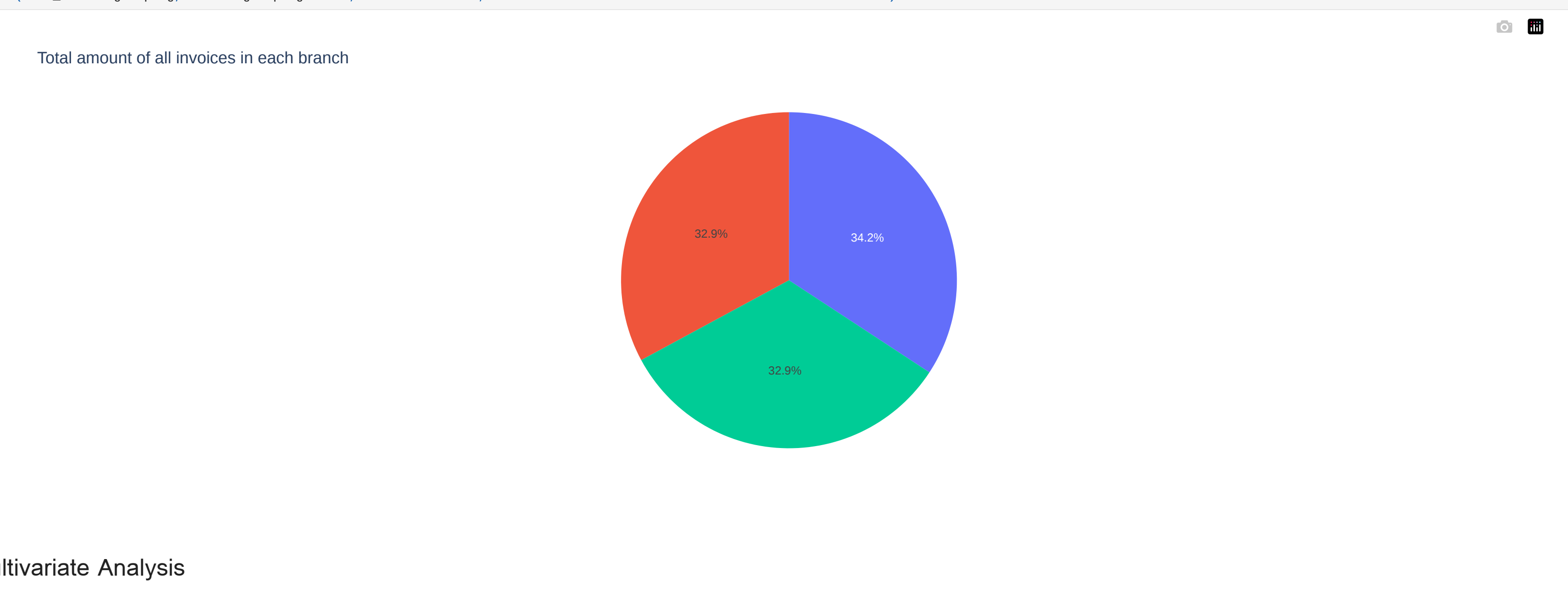
branch	A	B	C
date			
2019-01-01	1.00	1.00	1.00
2019-01-02	0.13	0.76	0.57
2019-01-03	0.40	0.66	0.15
2019-01-04	0.20	0.33	0.75
2019-01-05	0.85	0.32	1.22

In [22]:
```python
px.line(data_frame= norm, title= 'The sales movement in each branch')
```



It is clear that the branch "A" has less sales movement that the others, we need more data to investigate why.

In [23]:
```python
grouping = df.groupby('branch')['total'].sum()
px.pie(data_frame= grouping, names= grouping.index, values= 'total', title= 'Total amount of all invoices in each branch')
```



Multivariate Analysis

mean Rating of M/F in each branch

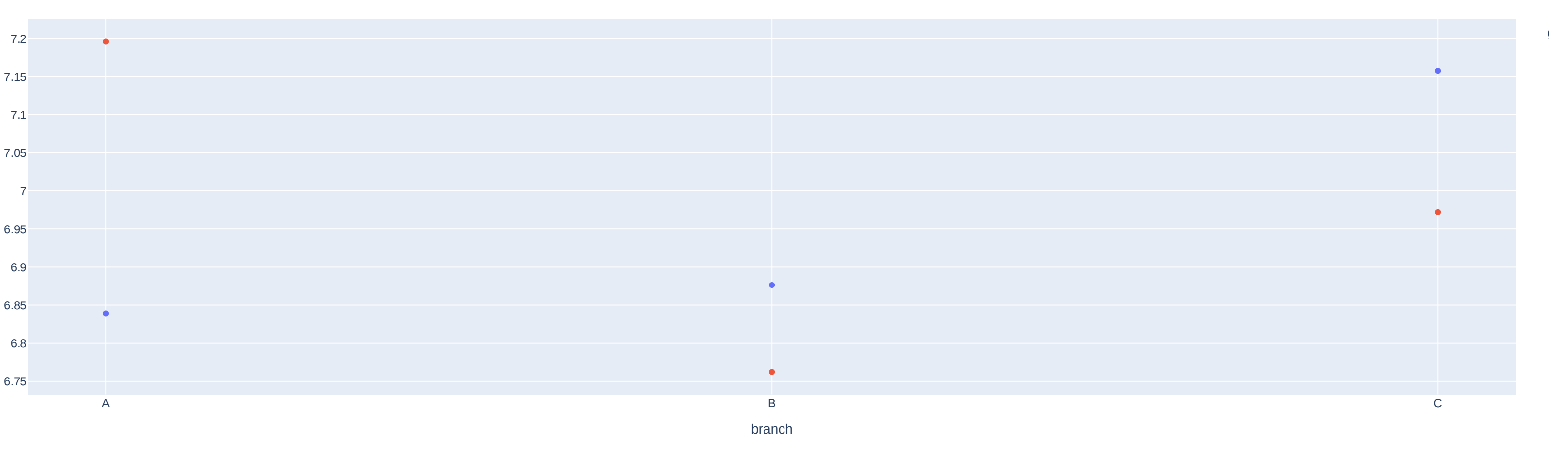
In [24]:
```python
grouping = df[['branch', 'gender', 'rating']]
```

In [25]:
```python
grouping = grouping.groupby(['branch', 'gender'], as_index= False)['rating'].mean()
grouping
```

Out[25]:

	branch	gender	rating
0	A	Female	6.84
1	A	Male	7.20
2	B	Female	6.88
3	B	Male	6.76
4	C	Female	7.16
5	C	Male	6.97

In [26]:
```python
px.scatter(data_frame= grouping, x= 'branch', y= 'rating', color= 'gender')
```



Mean Rating of M/F in each branch in each month

In [27]:
```python
grouping = df[['branch', 'gender', 'rating', 'date']]
```

In [28]:
```python
grouping.pivot_table(index = df['date'].dt.to_period('m'), columns= ['gender', 'branch'], values= 'rating', aggfunc= 'mean')
```

Out[28]:

gender	Female			Male		
branch	A	B	C	A	B	C
date						
2019-01	7.06	6.92	7.22	7.09	6.67	7.07
2019-02	6.64	6.97	7.15	7.39	7.05	7.28
2019-03	6.81	6.73	7.09	7.16	6.59	6.66