


## ✓ Загрузка данных

```
from google.colab import drive
drive.mount('/content/drive')
```

 Mounted at /content/drive

Данные взяты с сайта kaggle [данные о продажах маркетплейса Amazon](#)

```
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/Lesson14.python/amazon_market_data.csv')
```

df



	Order Date	Row ID	Order ID	Ship Mode	Customer ID	Segment	Country	City	State	Postal Code	Region	Product ID	Category	Sub-Category	Product Name	Sales	Quantity
0	01-01-20	849	CA-2017-107503	Standard Class	GA-14725	Consumer	United States	Lorain	Ohio	44052	East	FUR-FU-10003878	Furniture	Furnishings	Linden 10" Round Wall Clock, Black	48.896	
1	01-01-20	4010	CA-2017-144463	Standard Class	SC-20725	Consumer	United States	Los Angeles	California	90036	West	FUR-FU-10001215	Furniture	Furnishings	Howard Miller 11-1/2" Diameter Brentwood Wall ...	474.430	
2	01-01-20	6683	CA-2017-154466	First Class	DP-13390	Home Office	United States	Franklin	Wisconsin	53132	Central	OFF-BI-10002012	Office Supplies	Binders	Wilson Jones Easy Flow II Sheet Lifters	3.600	
3	01-01-20	8070	CA-2017-151750	Standard Class	JM-15250	Consumer	United States	Huntsville	Texas	77340	Central	OFF-ST-10002743	Office Supplies	Storage	SAFCO Boltless Steel Shelving	454.560	
4	01-01-20	8071	CA-2017-151750	Standard Class	JM-15250	Consumer	United States	Huntsville	Texas	77340	Central	FUR-FU-10002116	Furniture	Furnishings	Tenex Carpeted, Granite-Look or Clear Contempo...	141.420	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3307	30-12-20	908	CA-2017-143259	Standard Class	PO-18865	Consumer	United States	New York City	New York	10009	East	TEC-PH-10004774	Technology	Phones	Gear Head AU3700S Headset	90.930	
3308	30-12-20	909	CA-2017-143259	Standard Class	PO-18865	Consumer	United States	New York City	New York	10009	East	OFF-BI-10003684	Office Supplies	Binders	Wilson Jones Legal Size Ring Binders	52.776	
3309	30-12-20	1297	CA-2017-115427	Standard Class	EB-13975	Corporate	United States	Fairfield	California	94533	West	OFF-BI-10002103	Office Supplies	Binders	Cardinal Slant-D Ring Binder, Heavy Gauge Vinyl	13.904	
3310	30-12-20	1298	CA-2017-115427	Standard Class	EB-13975	Corporate	United States	Fairfield	California	94533	West	OFF-BI-10004632	Office Supplies	Binders	GBC Binding covers	20.720	

3311	30-12-20	5092	CA-2017-156720	Standard Class	JM-15580	Consumer	United States	Loveland	Colorado	80538	West	OFF-FA-10003472	Office Supplies	Fasteners	Bagged Rubber Bands	3.024
------	----------	------	----------------	----------------	----------	----------	---------------	----------	----------	-------	------	-----------------	-----------------	-----------	---------------------	-------

3312 rows × 19 columns

df.columns

```
Index(['Order Date', 'Row ID', 'Order ID', 'Ship Mode', 'Customer ID',
      'Segment', 'Country', 'City', 'State', 'Postal Code', 'Region',
      'Product ID', 'Category', 'Sub-Category', 'Product Name', 'Sales',
      'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

## ▼ Приведение столбцов к стилю camel\_case

```
def to_camel_case(column_name):
    components = column_name.lower().split()
    return components[0] + ''.join(word.capitalize() for word in components[1:])
```

```
# Преобразовать имена столбцов
df.columns = [to_camel_case(col) for col in df.columns]
```

df.columns

```
Index(['orderId', 'rowId', 'orderDate', 'shipMode', 'customerId', 'segment',
      'country', 'city', 'state', 'postalCode', 'region', 'productId',
      'category', 'sub-category', 'productName', 'sales', 'quantity',
      'discount', 'profit'],
      dtype='object')
```

## ▼ Приведение типов данных

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3312 entries, 0 to 3311
Data columns (total 19 columns):
 #   Column              Non-Null Count  Dtype
---  ---
 0   orderId              3312 non-null    object
 1   rowId                3312 non-null    object
 2   orderDate            3312 non-null    object
 3   shipMode             3312 non-null    object
 4   customerId           3312 non-null    object
 5   segment              3312 non-null    object
 6   country              3312 non-null    object
 7   city                 3312 non-null    object
 8   state                3312 non-null    object
 9   postalCode           3312 non-null    object
10   region               3312 non-null    object
11   productId            3312 non-null    object
12   category             3312 non-null    object
13   sub-category         3312 non-null    object
14   productName          3312 non-null    object
15   sales                3312 non-null    object
16   quantity             3312 non-null    object
17   discount             3312 non-null    object
18   profit               3312 non-null    object
```

```
0  orderDate      3312 non-null  object
1  rowId          3312 non-null  int64
2  orderId        3312 non-null  object
3  shipMode       3312 non-null  object
4  customerId     3312 non-null  object
5  segment        3312 non-null  object
6  country        3312 non-null  object
7  city           3312 non-null  object
8  state          3312 non-null  object
9  postalCode     3312 non-null  int64
10 region         3312 non-null  object
11 productId      3312 non-null  object
12 category       3312 non-null  object
13 sub-category   3312 non-null  object
14 productName    3312 non-null  object
15 sales          3312 non-null  float64
16 quantity       3312 non-null  int64
17 discount       3312 non-null  float64
18 profit         3312 non-null  float64
dtypes: float64(3), int64(3), object(13)
memory usage: 491.8+ KB
```

### Привести дату к типу datetime

```
df['orderDate'] = pd.to_datetime(df['orderDate'], format='%d-%m-%y')
```

```
df
```



	orderDate	rowId	orderId	shipMode	customerId	segment	country	city	state	postalCode	region	productId	category	sub-category	productNa
0	2020-01-01	849	CA-2017-107503	Standard Class	GA-14725	Consumer	United States	Lorain	Ohio	44052	East	FUR-FU-10003878	Furniture	Furnishings	Linden Round V Clock, Blk
1	2020-01-01	4010	CA-2017-144463	Standard Class	SC-20725	Consumer	United States	Los Angeles	California	90036	West	FUR-FU-10001215	Furniture	Furnishings	Howard Mi 11-1 Diame Brentwc Wa
2	2020-01-01	6683	CA-2017-154466	First Class	DP-13390	Home Office	United States	Franklin	Wisconsin	53132	Central	OFF-BI-10002012	Office Supplies	Binders	Wilson Jor Easy Flov Sheet Lift
3	2020-01-01	8070	CA-2017-151750	Standard Class	JM-15250	Consumer	United States	Huntsville	Texas	77340	Central	OFF-ST-10002743	Office Supplies	Storage	SAF Boltless St Shelv
4	2020-01-01	8071	CA-2017-151750	Standard Class	JM-15250	Consumer	United States	Huntsville	Texas	77340	Central	FUR-FU-10002116	Furniture	Furnishings	Ter Carpet Granite-Lc or Cl Contemp
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3307	2020-12-30	908	CA-2017-143259	Standard Class	PO-18865	Consumer	United States	New York City	New York	10009	East	TEC-PH-10004774	Technology	Phones	Gear He AU37C Head
3308	2020-12-30	909	CA-2017-143259	Standard Class	PO-18865	Consumer	United States	New York City	New York	10009	East	OFF-BI-10003684	Office Supplies	Binders	Wilson Jor Legal S Ring Bind
3309	2020-12-30	1297	CA-2017-115427	Standard Class	EB-13975	Corporate	United States	Fairfield	California	94533	West	OFF-BI-10002103	Office Supplies	Binders	Cardinal Sle D Ring Bin Heavy Gat Vi
3310	2020-12-30	1298	CA-2017-115427	Standard Class	EB-13975	Corporate	United States	Fairfield	California	94533	West	OFF-BI-10004632	Office Supplies	Binders	GBC Bind cov
3311	2020-12-30	5092	CA-2017-156720	Standard Class	JM-15580	Consumer	United States	Loveland	Colorado	80538	West	OFF-FA-10003472	Office Supplies	Fasteners	Bagg Rubber Bar

3312 rows × 19 columns

## Поиск дублей и пропусков в данных

# Поиск дублей

```
duplicates = df.duplicated().sum()  
print(f"Кол-во дублей: {duplicates}")
```

# Поиск пропусков

```
missing_values = df.isnull().sum()  
print(f"Пропуски в данных:\n{missing_values}")
```

```
→ Кол-во дублей: 0  
Пропуски в данных:  
orderId      0  
productId    0  
category     0  
sub-category 0  
productName  0  
sales        0  
quantity     0  
discount     0  
profit       0  
dtype: int64
```

## ✎ Исследовательский анализ данных

df



	orderDate	rowId	orderId	shipMode	customerId	segment	country	city	state	postalCode	region	productId	category	sub-category	productNa
0	2020-01-01	849	CA-2017-107503	Standard Class	GA-14725	Consumer	United States	Lorain	Ohio	44052	East	FUR-FU-10003878	Furniture	Furnishings	Linden Round V Clock, Blk
1	2020-01-01	4010	CA-2017-144463	Standard Class	SC-20725	Consumer	United States	Los Angeles	California	90036	West	FUR-FU-10001215	Furniture	Furnishings	Howard Mi 11-1 Diame Brentwc Wa
2	2020-01-01	6683	CA-2017-154466	First Class	DP-13390	Home Office	United States	Franklin	Wisconsin	53132	Central	OFF-BI-10002012	Office Supplies	Binders	Wilson Jor Easy Flov Sheet Lift
3	2020-01-01	8070	CA-2017-151750	Standard Class	JM-15250	Consumer	United States	Huntsville	Texas	77340	Central	OFF-ST-10002743	Office Supplies	Storage	SAF Boltless St Shelv
4	2020-01-01	8071	CA-2017-151750	Standard Class	JM-15250	Consumer	United States	Huntsville	Texas	77340	Central	FUR-FU-10002116	Furniture	Furnishings	Ter Carpet Granite-Lc or Cl Contemp
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3307	2020-12-30	908	CA-2017-143259	Standard Class	PO-18865	Consumer	United States	New York City	New York	10009	East	TEC-PH-10004774	Technology	Phones	Gear He AU37C Head
3308	2020-12-30	909	CA-2017-143259	Standard Class	PO-18865	Consumer	United States	New York City	New York	10009	East	OFF-BI-10003684	Office Supplies	Binders	Wilson Jor Legal S Ring Bind
3309	2020-12-30	1297	CA-2017-115427	Standard Class	EB-13975	Corporate	United States	Fairfield	California	94533	West	OFF-BI-10002103	Office Supplies	Binders	Cardinal Sla D Ring Bin Heavy Gat Vi
3310	2020-12-30	1298	CA-2017-115427	Standard Class	EB-13975	Corporate	United States	Fairfield	California	94533	West	OFF-BI-10004632	Office Supplies	Binders	GBC Bind cov
3311	2020-12-30	5092	CA-2017-156720	Standard Class	JM-15580	Consumer	United States	Loveland	Colorado	80538	West	OFF-FA-10003472	Office Supplies	Fasteners	Bagg Rubber Bar

3312 rows × 19 columns

## ✓ Количество уникальных заказов

```
unique_orders = df['orderId'].nunique()  
print(f"Количество уникальных заказов: {unique_orders}")
```

↗ Количество уникальных заказов: 1687

## ✓ Количество уникальных клиентов

```
unique_customers = df['customerId'].nunique()  
print(f"Количество уникальных клиентов: {unique_customers}")
```

↗ Количество уникальных клиентов: 693

### Среднее количество заказов на клиента

```
orders_per_customer = df.groupby('customerId')['orderId'].nunique().mean()  
print(f"Среднее количество заказов на клиента: {orders_per_customer:.2f}")
```

↗ Среднее количество заказов на клиента: 2.43

### Топ клиентов по количеству заказов

```
top_customers = df.groupby('customerId')['orderId'].nunique().sort_values(ascending=False).head(10)  
print("Топ-10 клиентов по числу заказов:\n", top_customers)
```

↗ Топ-10 клиентов по числу заказов:

customerId	
EP-13915	8
MH-18115	8
CV-12805	7
SJ-20125	7
Dp-13240	6
SH-20395	6
JL-15835	6
FH-14275	6
DJ-13510	6
DO-13645	6

Name: orderId, dtype: int64



### Проверка на аномалии (клиенты с 1 заказом vs постоянные)

```
customer_order_stats = df.groupby('customerId')['orderId'].nunique().value_counts()
print("Распределение клиентов по количеству заказов:\n", customer_order_stats)
```

```
⇒ Распределение клиентов по количеству заказов:
orderId
1      200
2      200
3      156
4       85
5       39
6        9
7         2
8         2
Name: count, dtype: int64
```

## ✓ Локация пользователей по регионам (визуализировать)

### Распределение заказов по регионам (столбчатая и круговая диаграммы)

```
plt.figure(figsize=(12, 6))

# Создаем цветовую палитру
colors = sns.color_palette("Oranges", len(df['region'].value_counts()))

# Устанавливаем порядок категорий
df['region'] = pd.Categorical(df['region'], categories=['West', 'East', 'Central', 'South'])

# Создаем фигуру с двумя графиками
fig, ax = plt.subplots(1, 2, figsize=(12, 6))

# Столбчатый график
bar_chart = sns.countplot(
    x="region",
    hue="region", # Используем x как hue
    data=df,
    ax=ax[0],
    palette=colors,
    legend=False # Отключаем легенду
)

# Добавляем числа (подписи) сверху столбцов
```


```
for container in bar_chart.containers:
    bar_chart.bar_label(container, fmt='%d', fontsize=10, padding=3)

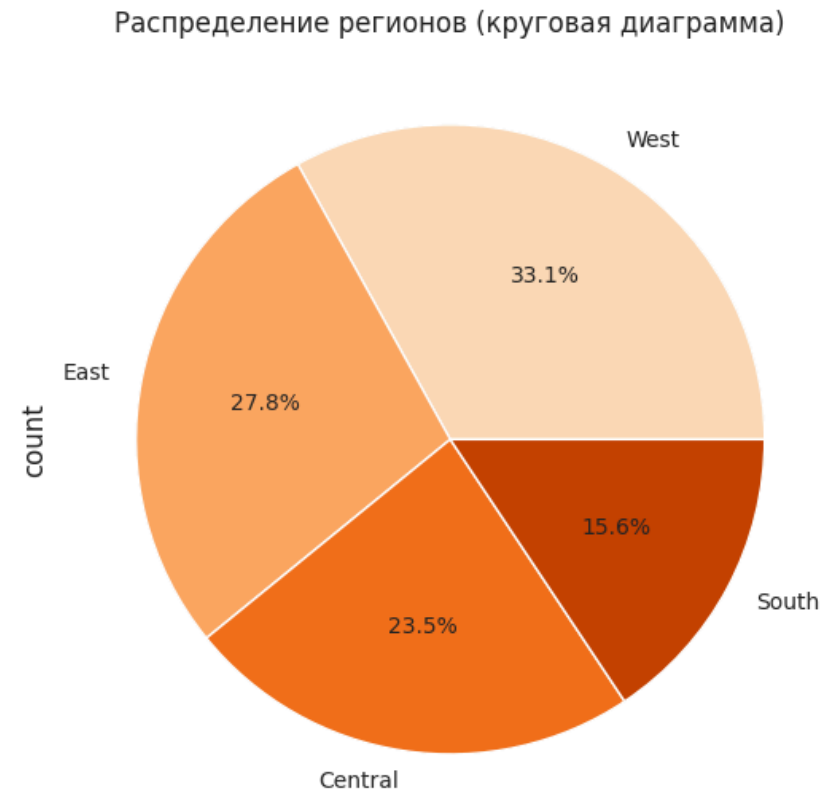
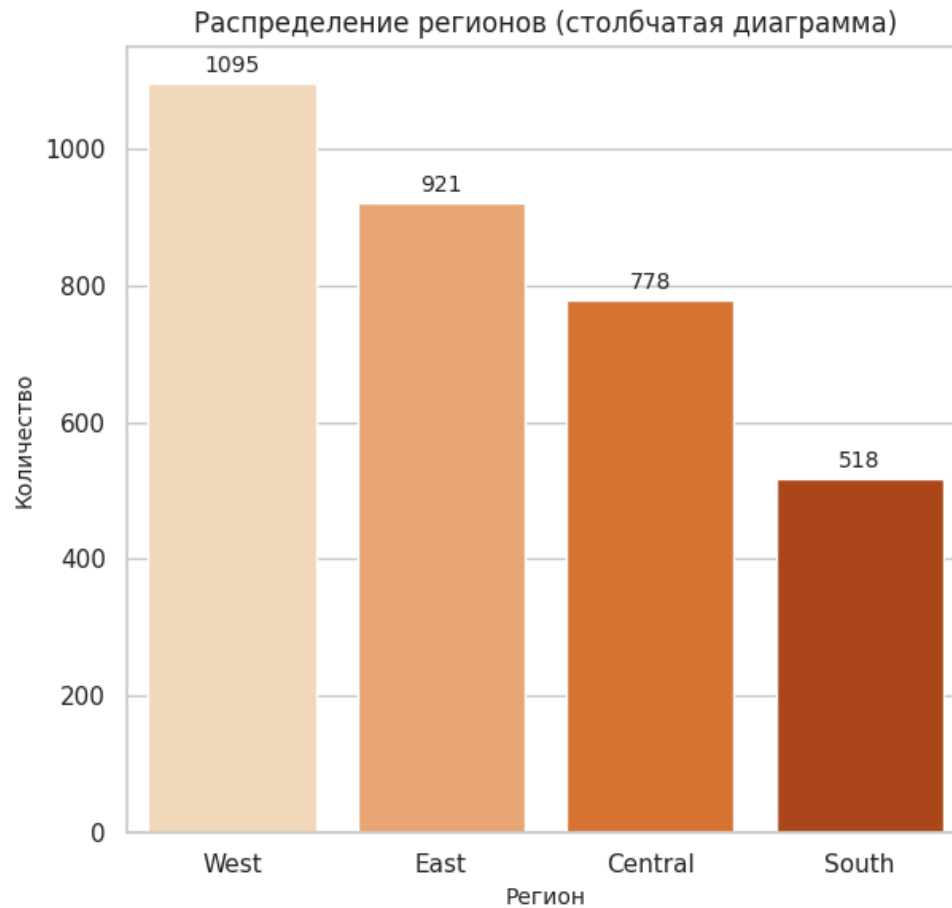
# Добавляем заголовок для столбчатого графика
ax[0].set_title('Распределение регионов (столбчатая диаграмма)', fontsize=12)
ax[0].set_xlabel('Регион', fontsize=10)
ax[0].set_ylabel('Количество', fontsize=10)

# Круговая диаграмма
pie_chart = df['region'].value_counts().plot.pie(
    ax=ax[1],
    autopct='%1.1f%%',
    shadow=False,
    textprops={'fontsize': 10},
    colors=colors
)

# Добавляем заголовок для круговой диаграммы
ax[1].set_title('Распределение регионов (круговая диаграмма)', fontsize=12)

# Настройка расположения графиков
plt.tight_layout()
plt.show()
```

 <Figure size 1200x600 with 0 Axes>



## ▼ Динамика кол-ва заказов по дате заказов

### Динамика заказов по дням/месяцам

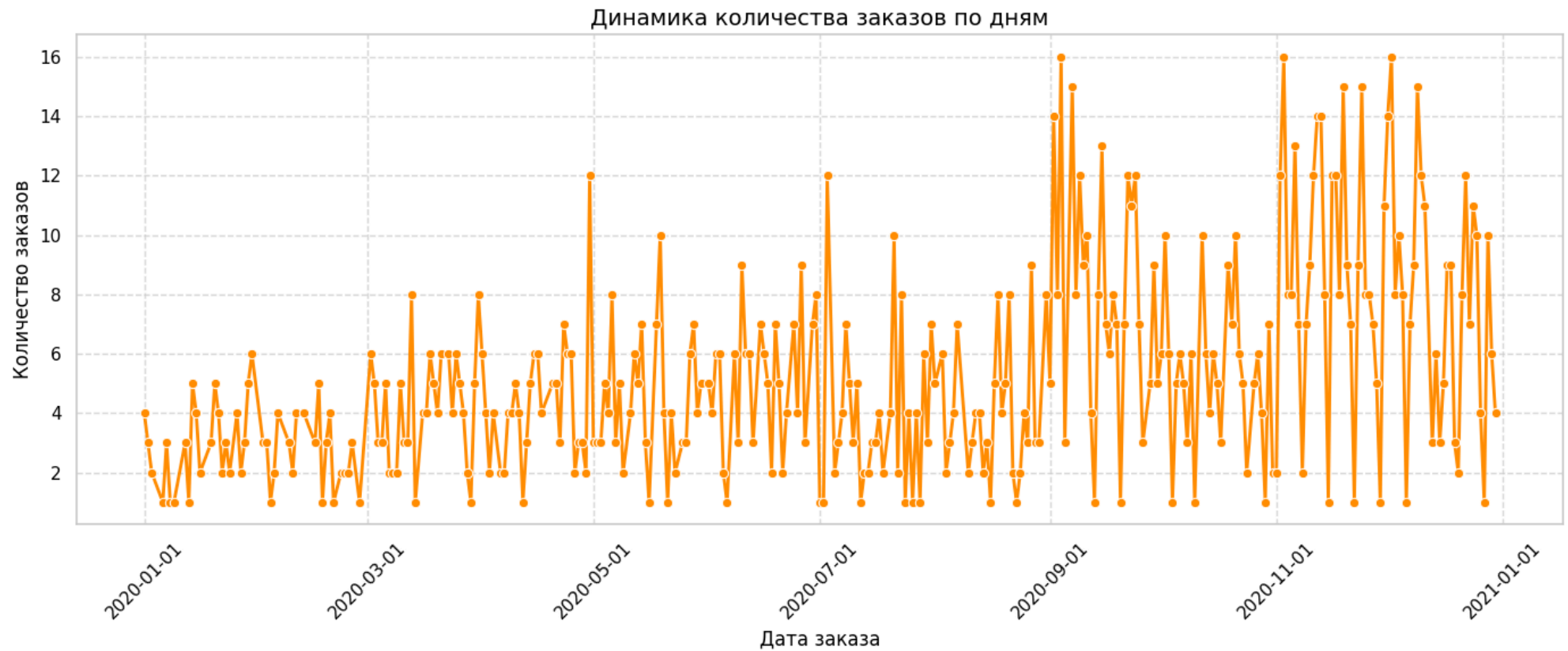
```
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.dates import DateFormatter
import pandas as pd # Не забудьте импортировать pandas, если он не был импортирован

# Группировка по дате (если orderDate в datetime)
```

```
df['orderDate'] = pd.to_datetime(df['orderDate']) # На случай, если не приведено
daily_orders = df.groupby(df['orderDate'].dt.date)['orderId'].nunique()

# Установка оранжевой цветовой палитры
sns.set_theme(style="whitegrid") # Установим стиль
sns.set_palette("Oranges") # Оранжевая палитра

# Визуализация
plt.figure(figsize=(14, 6))
sns.lineplot(data=daily_orders, marker='o', linewidth=2, color='#FF8C00') # Оранжевый цвет линии
plt.title('Динамика количества заказов по дням', fontsize=14, color='black') # Заголовок оранжевым
plt.xlabel('Дата заказа', fontsize=12, color='black') # Подпись оси X
plt.ylabel('Количество заказов', fontsize=12, color='black') # Подпись оси Y
plt.gca().xaxis.set_major_formatter(DateFormatter('%Y-%m-%d')) # Формат даты
plt.xticks(rotation=45, color='black') # Цвет подписей X
plt.yticks(color='black') # Цвет подписей Y
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



### Динамика продаж по месяцам заказов

```
# Группировка по месяцам с использованием 'ME'
monthly_orders = df.resample('ME', on='orderDate')['orderId'].nunique()

# Создание графика с улучшенным оформлением
plt.figure(figsize=(14, 6))
bars = plt.bar(
    monthly_orders.index.strftime('%Y-%m'), # Форматирование дат
    monthly_orders.values,
    color='orange', # Более приятный синий цвет
    edgecolor='darkorange', # Темно-синяя граница
    linewidth=1,
    alpha=0.8
)
```

```
plt.title('Динамика количества заказов по месяцам', fontsize=16, pad=20)
plt.xlabel('Месяц', fontsize=12, labelpad=10)
plt.ylabel('Количество заказов', fontsize=12, labelpad=10)
plt.xticks(rotation=45, ha='right') # Лучшее выравнивание подписей

# Добавление значений на столбцы
for bar in bars:
    height = bar.get_height()
    plt.text(
        bar.get_x() + bar.get_width()/2., height,
        f'{int(height)}',
        ha='center', va='bottom',
        fontsize=10
    )

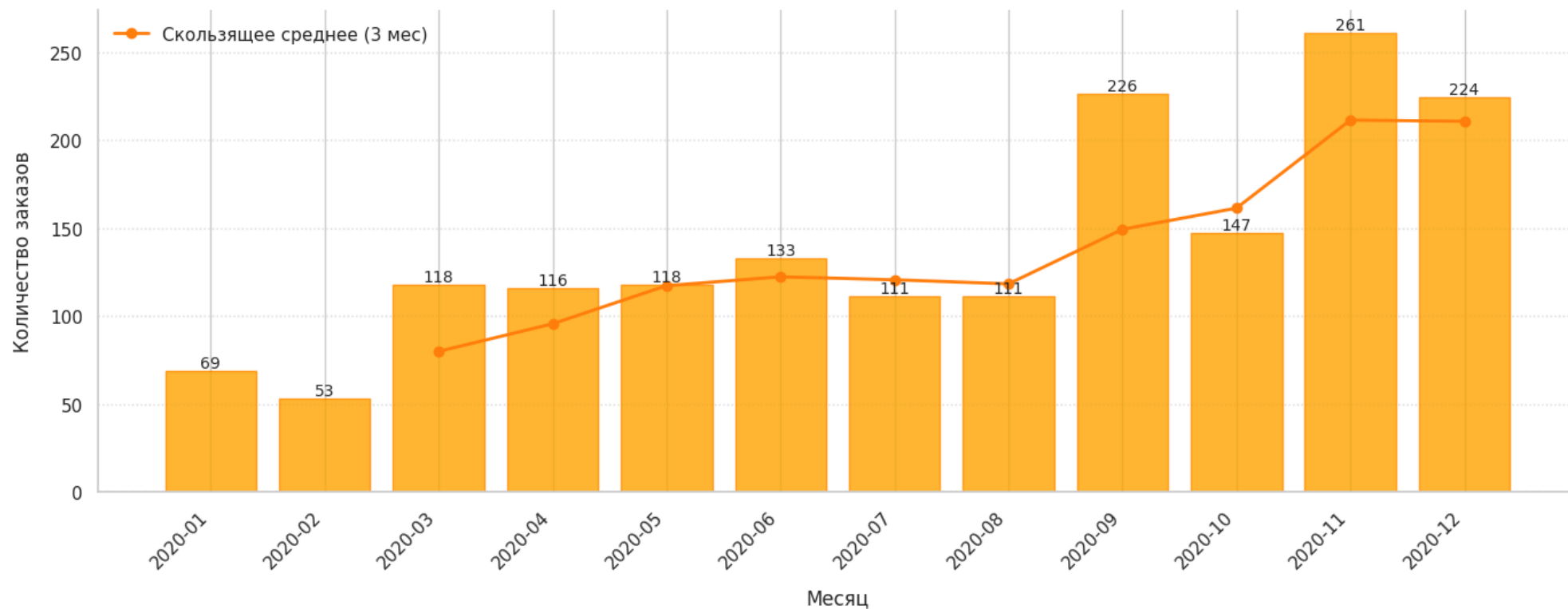
# Улучшенная сетка и оформление
plt.grid(axis='y', linestyle=':', alpha=0.6)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.tight_layout()

# Добавление линии тренда
plt.plot(
    monthly_orders.index.strftime('%Y-%m'),
    monthly_orders.rolling(window=3).mean(),
    color='#ff7f0e',
    linewidth=2,
    marker='o',
    label='Скользящее среднее (3 мес)'
)
plt.legend(frameon=False)

plt.show()
```



## Динамика количества заказов по месяцам



### Анализ недельной активности

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# День недели и час (если есть время)
df['day_of_week'] = df['orderDate'].dt.day_name()
df['hour'] = df['orderDate'].dt.hour # Если есть время

# Установка оранжевой палитры
sns.set_theme(style="whitegrid") # Устанавливаем стиль

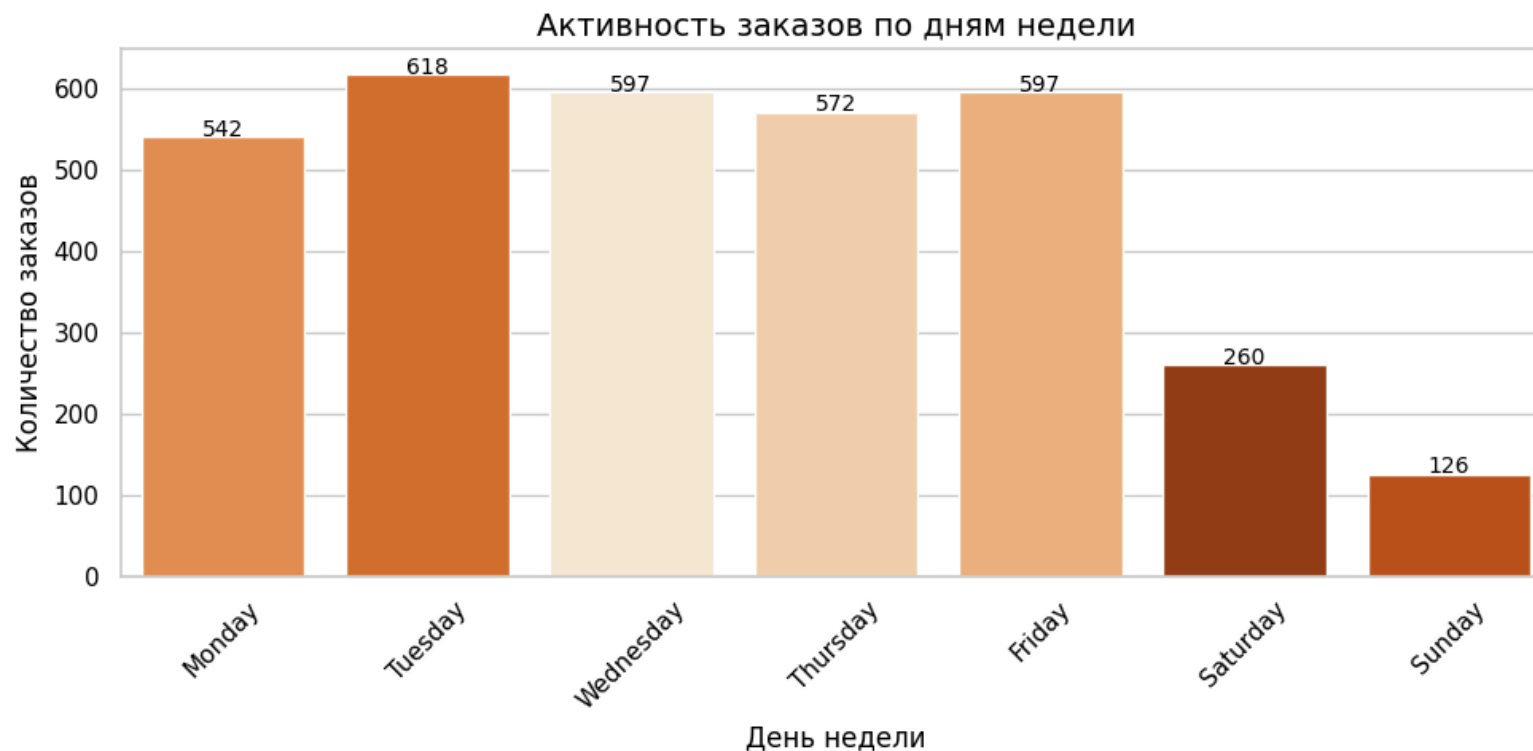
# Визуализация
plt.figure(figsize=(10, 5))
```

```
ax = sns.countplot(
    data=df,
    x='day_of_week',
    hue='day_of_week', # Добавляем hue с той же переменной
    order=['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'],
    palette="Oranges", # Используем оранжевую палитру
    legend=False # Отключаем легенду
)

# Добавляем цифры над столбцами
for p in ax.patches:
    height = p.get_height() # Высота столбца
    ax.text(
        x=p.get_x() + p.get_width() / 2, # Центр столбца
        y=height + 0.5, # Немного выше вершины столбца
        s=f'{int(height)}', # Значение (целое число)
        ha='center', # Выравнивание по центру
        fontsize=10, # Размер шрифта
        color='black' # Цвет текста
    )

# Настройки графика
plt.title('Активность заказов по дням недели', fontsize=14, color='black') # Заголовок
plt.xlabel('День недели', fontsize=12, color='black') # Подпись оси X
plt.ylabel('Количество заказов', fontsize=12, color='black') # Подпись оси Y
plt.xticks(rotation=45, color='black') # Цвет подписей оси X
plt.yticks(color='black') # Цвет подписей оси Y
plt.tight_layout()
plt.show()
```





## ✓ Распределение продаж по категориям заказов

### Распределение продаж по категориям (столбчатая диаграмма)

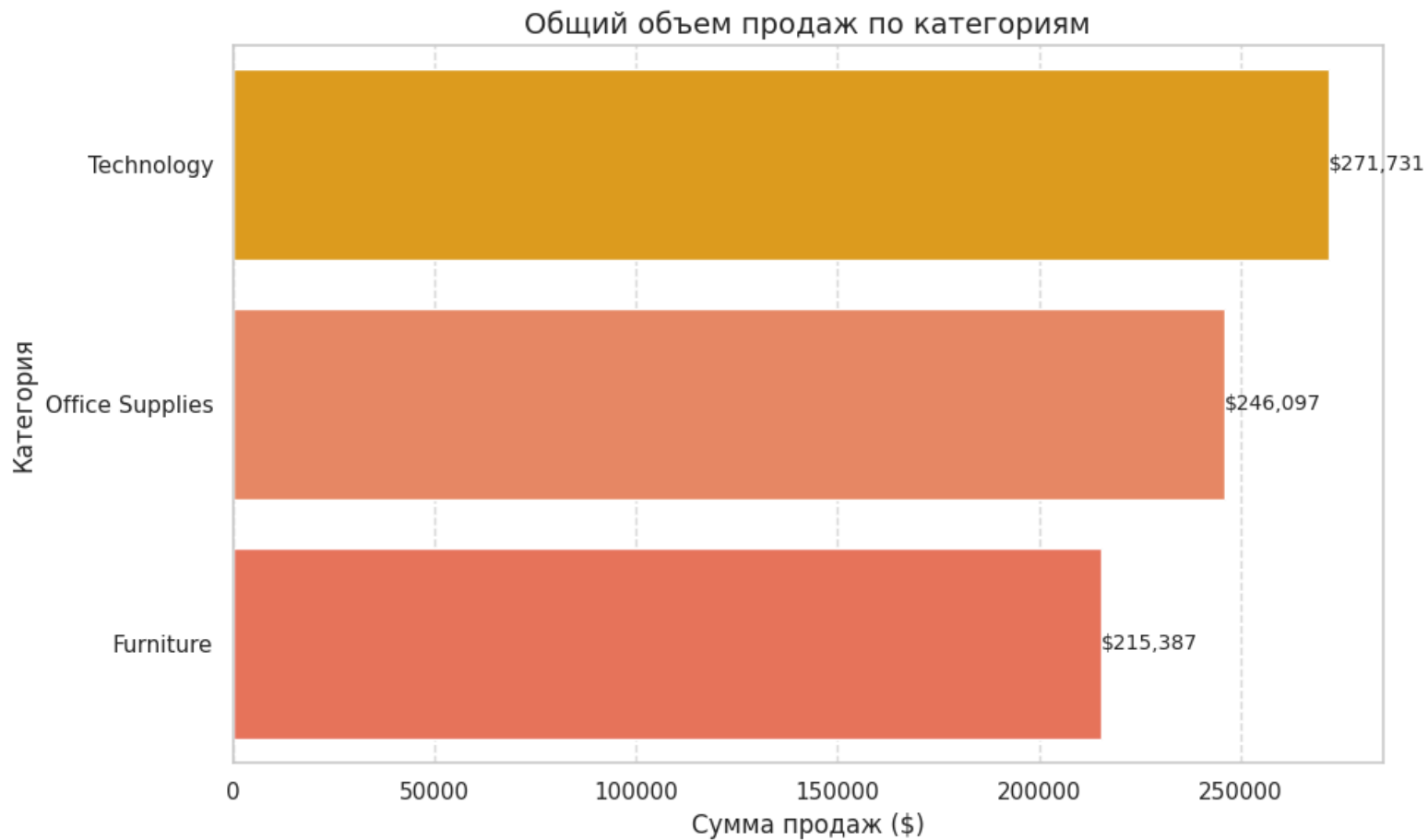
```
import matplotlib.pyplot as plt
import seaborn as sns

# Группировка по категориям и суммирование продаж
category_sales = df.groupby('category')['sales'].sum().sort_values(ascending=False)

# Настройка цветовой схемы (должно быть столько цветов, сколько категорий)
orange_palette = ['#FFA500', '#FF7F50', '#FF6347']

# Визуализация
plt.figure(figsize=(10, 6))
sns.barplot(x=category_sales.values, y=category_sales.index,
            hue=category_sales.index, # Добавлено для устранения предупреждения
```

```
palette=orange_palette,  
dodge=False, legend=False) # Отключение легенды  
  
plt.title('Общий объем продаж по категориям', fontsize=14)  
plt.xlabel('Сумма продаж ($)', fontsize=12)  
plt.ylabel('Категория', fontsize=12)  
plt.grid(axis='x', linestyle='--', alpha=0.7)  
  
# Добавление подписей с суммами  
for i, value in enumerate(category_sales.values):  
    plt.text(value, i, f'${value:,.0f}', va='center', ha='left', fontsize=10)  
  
plt.tight_layout()  
plt.show()
```



**Доля категорий в общем объеме продаж (круговая диаграмма)**

```
import matplotlib.pyplot as plt

# Группировка по категориям и суммирование продаж
category_sales = df.groupby('category')['sales'].sum().sort_values(ascending=False)

# Настройка цветовой схемы (должно быть столько цветов, сколько категорий)
orange_palette = ['#FFA500', '#FF7F50', '#FF6347']

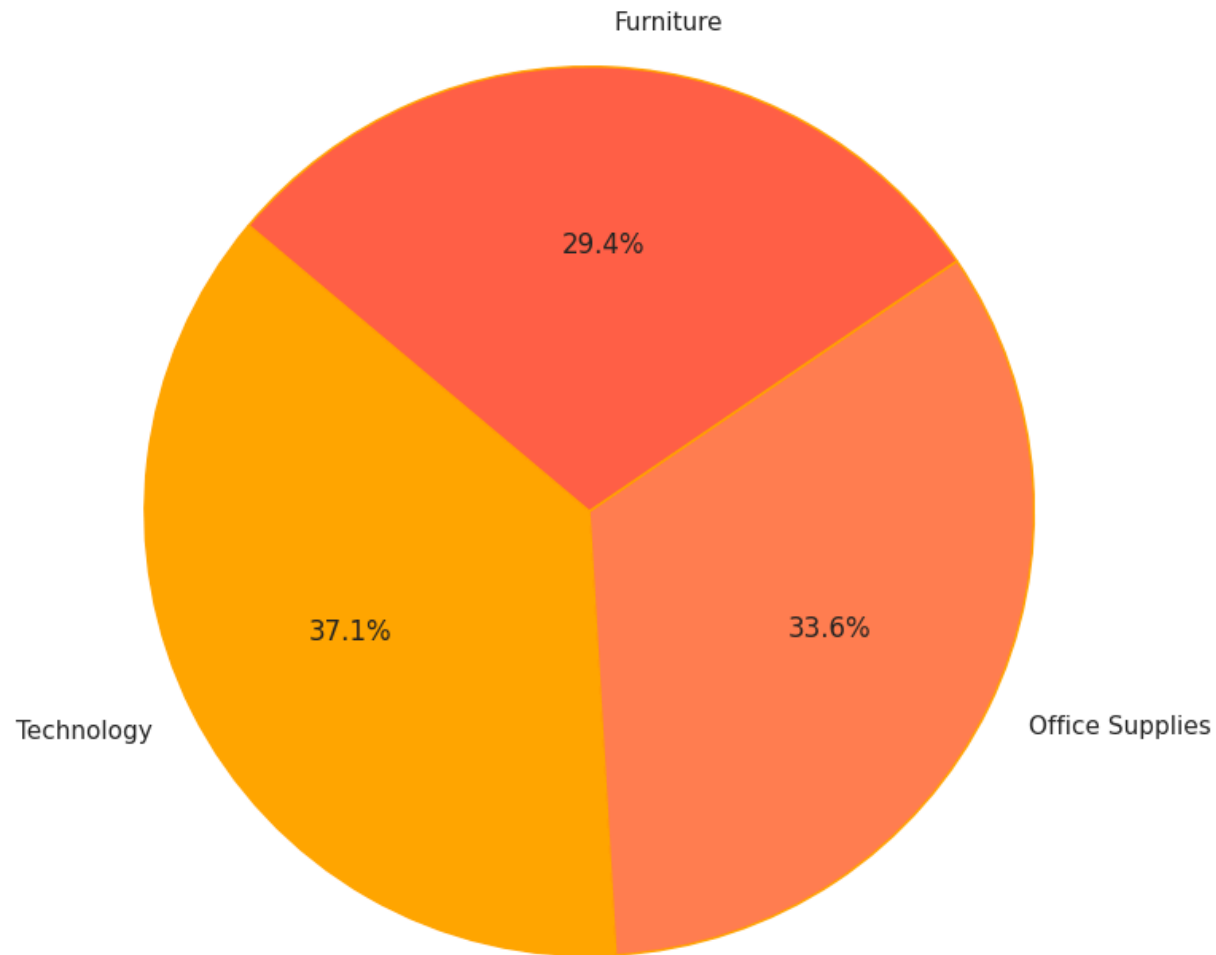
# Визуализация круговой диаграммы
plt.figure(figsize=(8, 8))
plt.pie(
    category_sales.values,
    labels=category_sales.index,
    autopct='%1.1f%%', # Процентное отображение
    startangle=140, # Начальный угол
    colors=orange_palette, # Цветовая палитра
    wedgeprops={'edgecolor': 'orange'} # Обводка сегментов
)

plt.title('Распределение продаж по категориям (%)', fontsize=14)

plt.tight_layout()
plt.show()
```



## Распределение продаж по категориям (%)

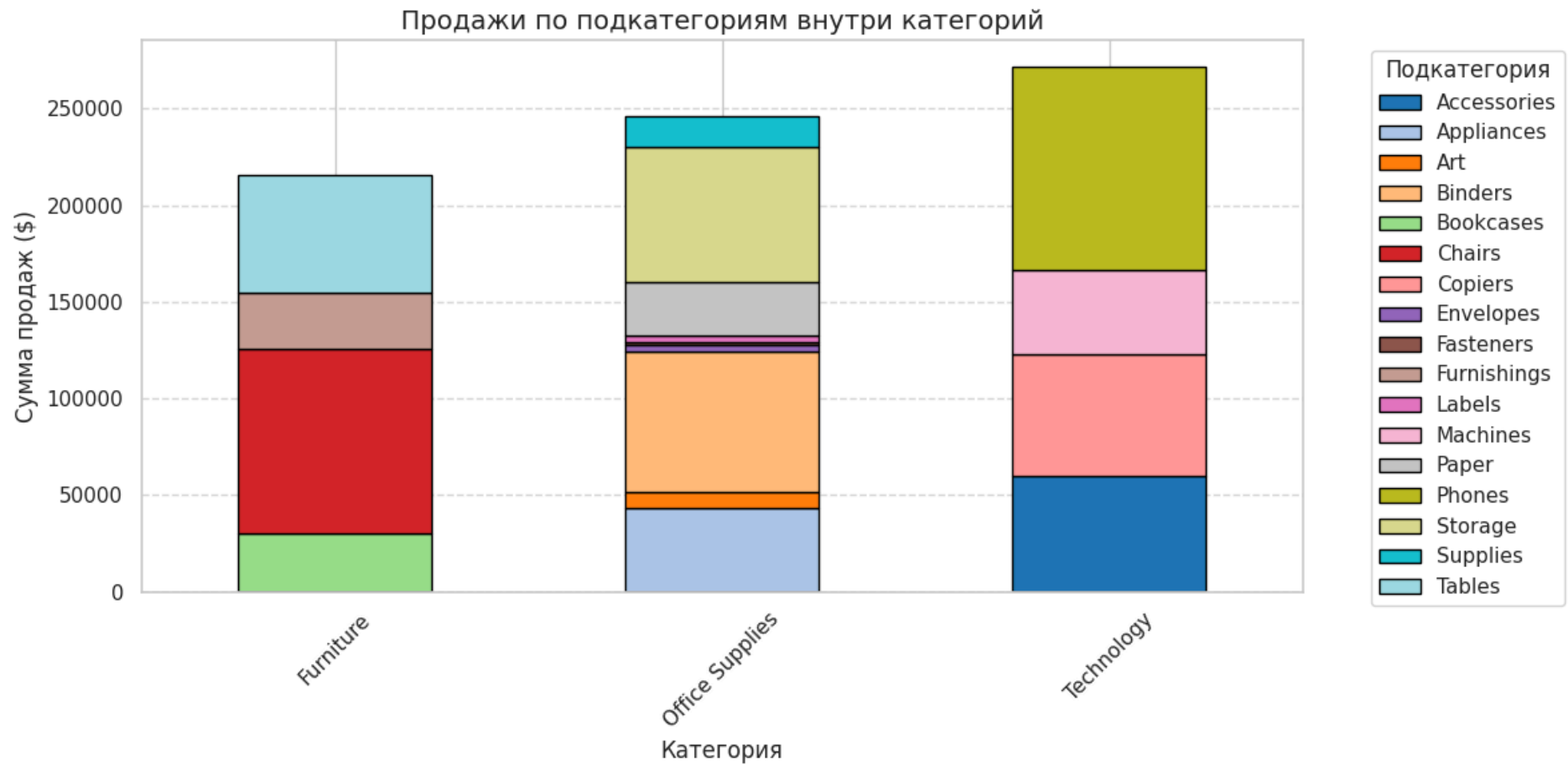


```
# Распределение подкатегорий (sub-category) внутри каждой категории
```

```
# Группировка по категориям и подкатегориям
```

```
subcat_sales = df.groupby(['category', 'sub-category'])['sales'].sum().unstack()
```

```
# Визуализация (stacked bar plot)
subcat_sales.plot(kind='bar', stacked=True, figsize=(12, 6),
                  colormap='tab20', edgecolor='black')
plt.title('Продажи по подкатегориям внутри категорий', fontsize=14)
plt.xlabel('Категория', fontsize=12)
plt.ylabel('Сумма продаж ($)', fontsize=12)
plt.xticks(rotation=45)
plt.legend(title='Подкатегория', bbox_to_anchor=(1.05, 1))
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



**Сравнение продаж, прибыли и скидок по категориям**

```
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap

# Создание кастомной оранжевой палитры
orange_palette = LinearSegmentedColormap.from_list("custom_orange", ['#FF6347', '#FF7F50', '#FFA500'])

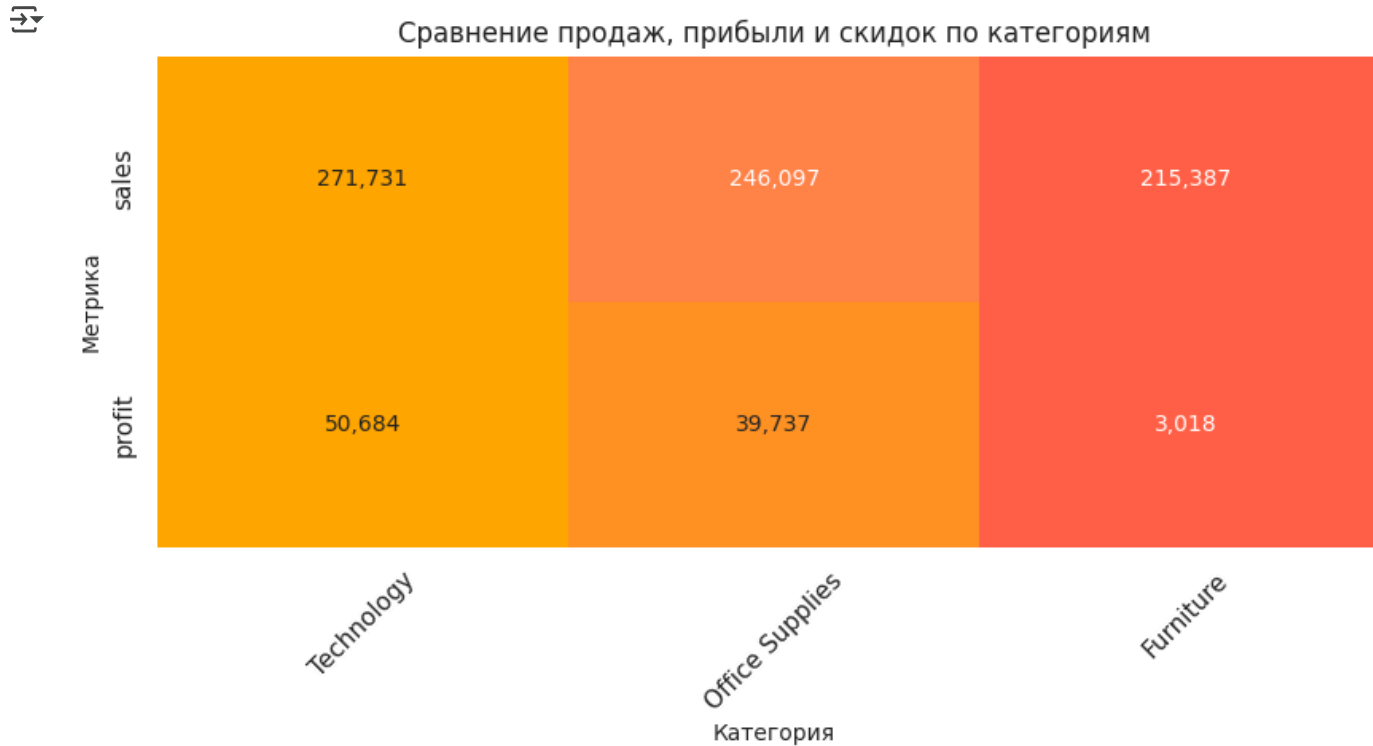
# Агрегация данных
category_stats = df.groupby('category').agg({
    'sales': 'sum',
    'profit': 'sum',
    # 'discount': 'mean' # Средний % скидки
}).sort_values('sales', ascending=False)

# Нормализация для визуализации
category_stats_norm = (category_stats - category_stats.min()) / (category_stats.max() - category_stats.min())

# Heatmap с кастомной палитрой
plt.figure(figsize=(10, 4))
sns.heatmap(category_stats_norm.T,
            annot=category_stats.T,
            fmt=',.0f',
            cmap=orange_palette, # Использование кастомной палитры

            annot_kws={'size': 10},
            cbar=False)

plt.title('Сравнение продаж, прибыли и скидок по категориям', fontsize=12)
plt.xlabel('Категория', fontsize=10)
plt.ylabel('Метрика', fontsize=10)
plt.xticks(rotation=45)
plt.show()
```



# Ящик с усами (распределение чеков по категориям)

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Определяем список цветов для каждой категории
palette = ['#FF6347', '#FF7F50', '#FFA500'] # Три оранжевых оттенка
```

```
plt.figure(figsize=(12, 8))
# Указываем палитру с количеством цветов, равным числу категорий
ax = sns.boxplot(
    data=df, x='category', y='sales', showfliers=False,
    width=0.6, palette=palette # Указываем палитру
)
```

```
# Добавляем аннотации с основными статистиками
for i, category in enumerate(df['category'].unique()):
    # Получаем данные для категории
    cat_data = df[df['category'] == category]['sales']
```

```
# Рассчитываем статистики
stats = cat_data.describe()
q1, med, q3 = stats['25%'], stats['50%'], stats['75%']

# Добавляем подписи к медиане
ax.text(i, med, f'Медиана: ${med:.2f}',
        ha='center', va='center', fontweight='bold',
        bbox=dict(facecolor='white', alpha=0.8, edgecolor='none'))

# Добавляем подписи к квартилям
ax.text(i, q1, f'Q1: ${q1:.2f}',
        ha='center', va='top', fontsize=10,
        bbox=dict(facecolor='white', alpha=0.6, edgecolor='none'))
ax.text(i, q3, f'Q3: ${q3:.2f}',
        ha='center', va='bottom', fontsize=10,
        bbox=dict(facecolor='white', alpha=0.6, edgecolor='none'))

# Добавляем линию для среднего значения
mean_val = cat_data.mean()
ax.hlines(mean_val, i-0.3, i+0.3, colors='red', linestyle='dashed', linewidth=1.5)
ax.text(i+0.35, mean_val, f'Среднее: ${mean_val:.2f}',
        ha='left', va='center', color='red')

# Настройка оформления
plt.title('Распределение суммы заказов по категориям с квартилями и медианой', fontsize=16, pad=20)
plt.xlabel('Категория товаров', fontsize=14, labelpad=12)
plt.ylabel('Сумма продаж ($)', fontsize=14, labelpad=12)
plt.xticks(rotation=0, fontsize=12)
plt.yticks(fontsize=11)
plt.grid(axis='y', linestyle=':', alpha=0.6)

# Добавляем легенду
plt.legend(handles=[
    plt.Line2D([], [], color='red', linestyle='--', label='Среднее значение'),
    plt.Rectangle((0,0), 1, 1, fc='white', ec='none', alpha=0.8, label='Медиана и квартили')
], loc='upper right')

plt.tight_layout()
plt.show()
```

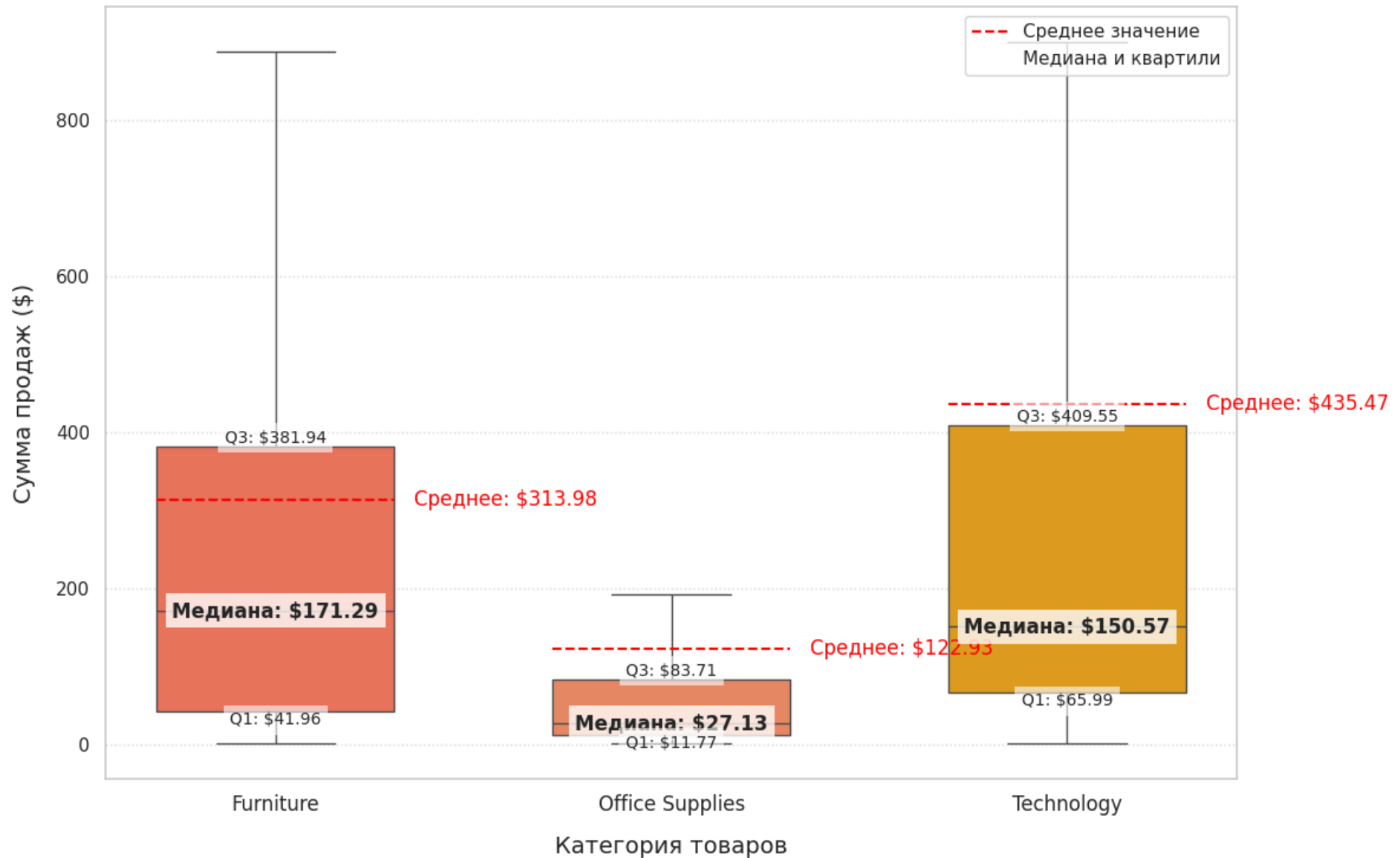


```
<ipython-input-163-17ab681e53f9>:9: FutureWarning:
```

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for

```
ax = sns.boxplot(
```

### Распределение суммы заказов по категориям с квартилями и медианой



```
# Средний чек по категориям
avg_order = df.groupby('category')['sales'].mean().sort_values(ascending=False)
print("Средний чек по категориям:\n", avg_order)
```

```
# Топ-5 товаров по продажам
top_products = df.groupby('productName')['sales'].sum().nlargest(5)
print("\nТоп-5 товаров:\n", top_products)
```

```
⇒ Средний чек по категориям:
   category
Technology    435.466043
Furniture     313.975611
Office Supplies 122.925662
Name: sales, dtype: float64
```

```
Топ-5 товаров:
   productName
Canon imageCLASS 2200 Advanced Copier    35699.898
Martin Yale Chadless Opener Electric Letter Opener 11825.902
GBC DocuBind TL300 Electric Binding System 10943.278
Hewlett Packard LaserJet 3310 Copier      9239.846
Samsung Galaxy Mega 6.3                  9239.780
Name: sales, dtype: float64
```

## ✓ Распределение продаж по городам (штатам)

### Топ-10 городов по объёму продаж

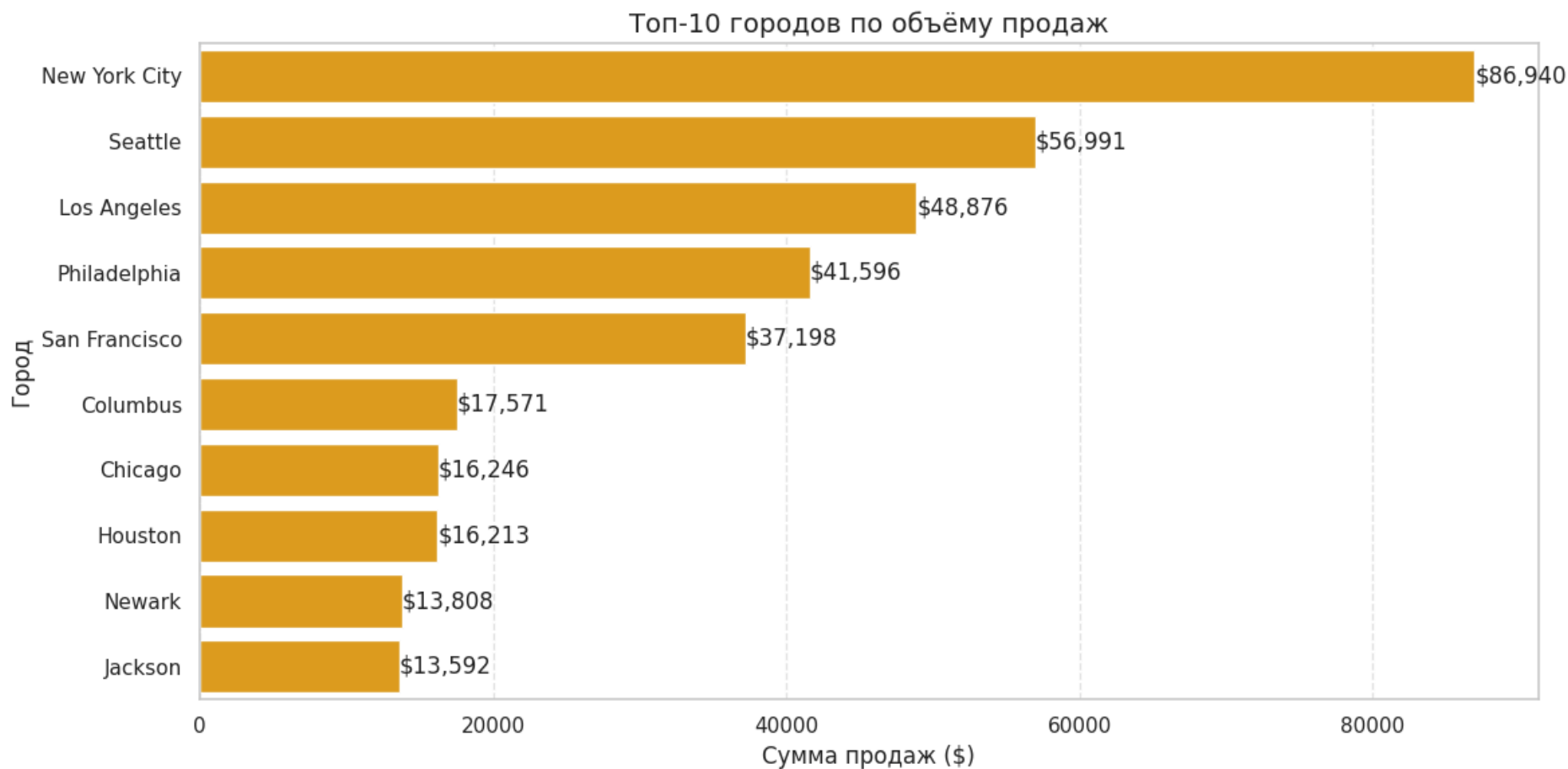
```
city_sales = df.groupby('city')['sales'].sum().nlargest(10)

plt.figure(figsize=(12,6))
sns.barplot(x=city_sales.values, y=city_sales.index, color='orange') # Указываем цвет
plt.title('Топ-10 городов по объёму продаж', fontsize=14)
plt.xlabel('Сумма продаж ($)', fontsize=12)
plt.ylabel('Город', fontsize=12)
plt.grid(axis='x', linestyle='--', alpha=0.5)

# Добавляем подписи значений
for i, value in enumerate(city_sales):
    plt.text(value, i, f'${value:,.0f}', va='center', ha='left')

plt.tight_layout()
```

```
plt.show()
```



#### Соотношение количества заказов и суммы продаж по городам

```
city_stats = df.groupby('city').agg(  
    total_sales=('sales', 'sum'),  
    order_count=('orderId', 'nunique')  
) .nlargest(10, 'total_sales')
```

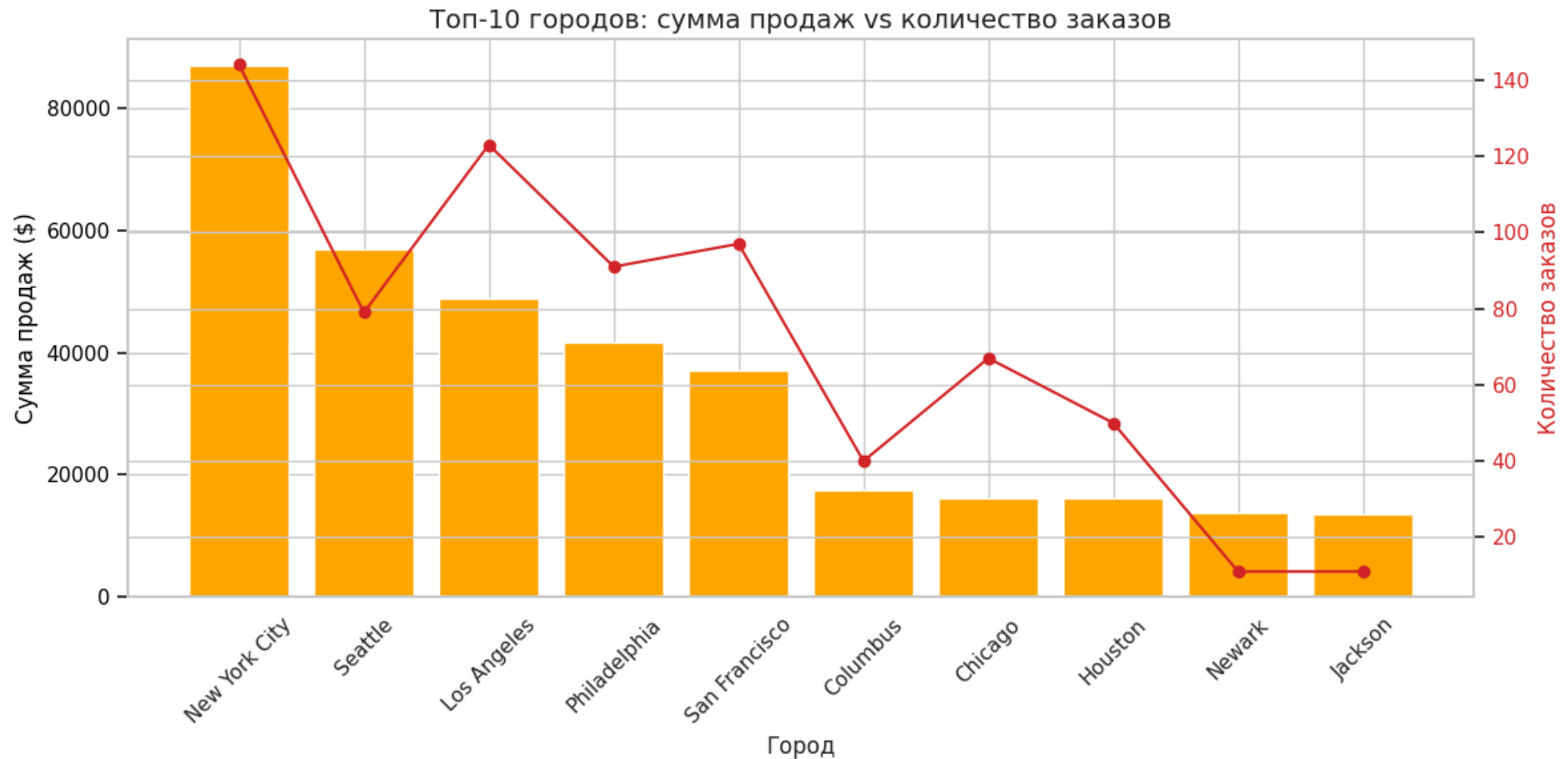
```
fig, ax1 = plt.subplots(figsize=(12,6))
```

```
# Изменяем цвет столбцов на оранжевый  
color = 'orange'
```

```
ax1.set_title('Топ-10 городов: сумма продаж vs количество заказов', fontsize=14)
ax1.bar(city_stats.index, city_stats['total_sales'], color=color)
ax1.set_xlabel('Город', fontsize=12)
ax1.set_ylabel('Сумма продаж ($)', color='black', fontsize=12)
ax1.tick_params(axis='y', labelcolor='black')
plt.xticks(rotation=45)

# Линия для количества заказов остаётся красной
ax2 = ax1.twinx()
color = 'tab:red'
ax2.plot(city_stats.index, city_stats['order_count'], color=color, marker='o')
ax2.set_ylabel('Количество заказов', color=color, fontsize=12)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()
plt.show()
```



## ✓ Самые популярные товары (Product Name)

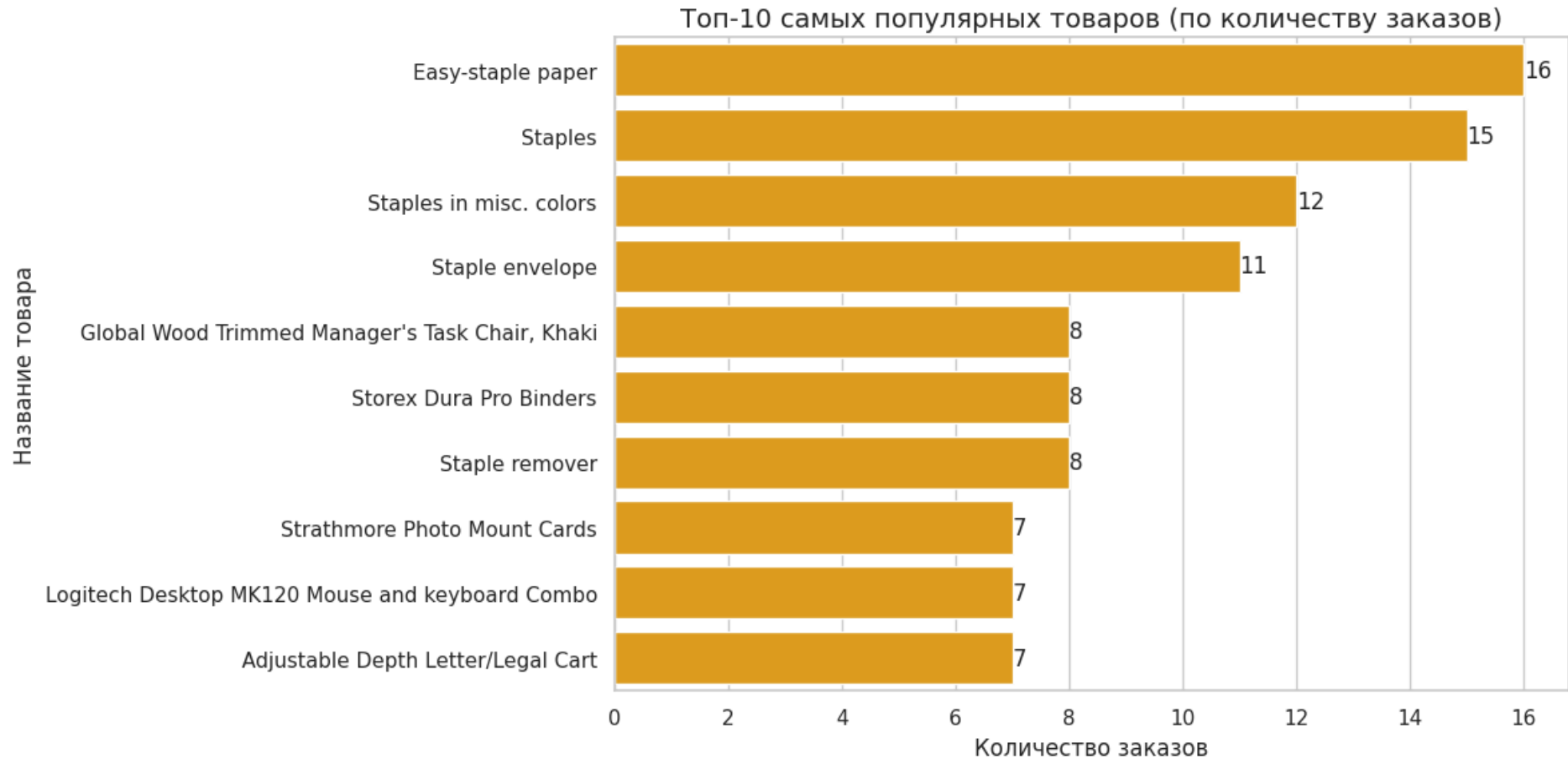
### Топ-10 товаров по количеству заказов

```
top_products = df['productName'].value_counts().head(10)

plt.figure(figsize=(12,6))
sns.barplot(x=top_products.values, y=top_products.index, color='orange') # Указываем цвет столбцов
plt.title('Топ-10 самых популярных товаров (по количеству заказов)', fontsize=14)
plt.xlabel('Количество заказов', fontsize=12)
plt.ylabel('Название товара', fontsize=12)
```

```
# Добавляем подписи значений
for i, value in enumerate(top_products):
    plt.text(value, i, f'{value}', va='center', ha='left')

plt.tight_layout()
plt.show()
```



#### Топ-10 товаров по объёму продаж

```
product_sales = df.groupby('productName')['sales'].sum().nlargest(10)

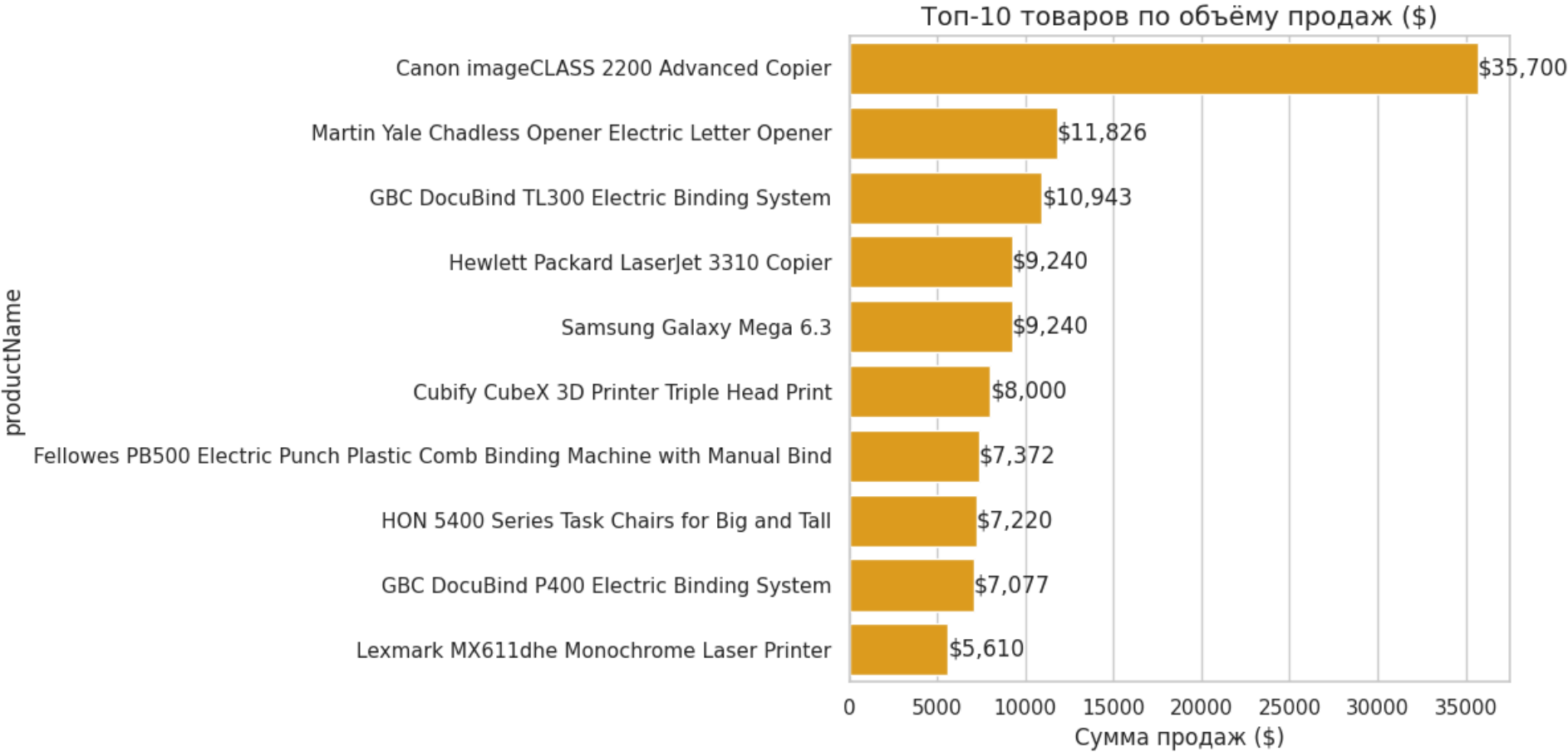
plt.figure(figsize=(12,6))
```

```
sns.barplot(x=product_sales.values, y=product_sales.index, color='orange')

plt.title('Топ-10 товаров по объёму продаж ($)', fontsize=14)
plt.xlabel('Сумма продаж ($)', fontsize=12)

for i, value in enumerate(product_sales):
    plt.text(value, i, f'${value:,.0f}', va='center', ha='left')

plt.tight_layout()
plt.show()
```



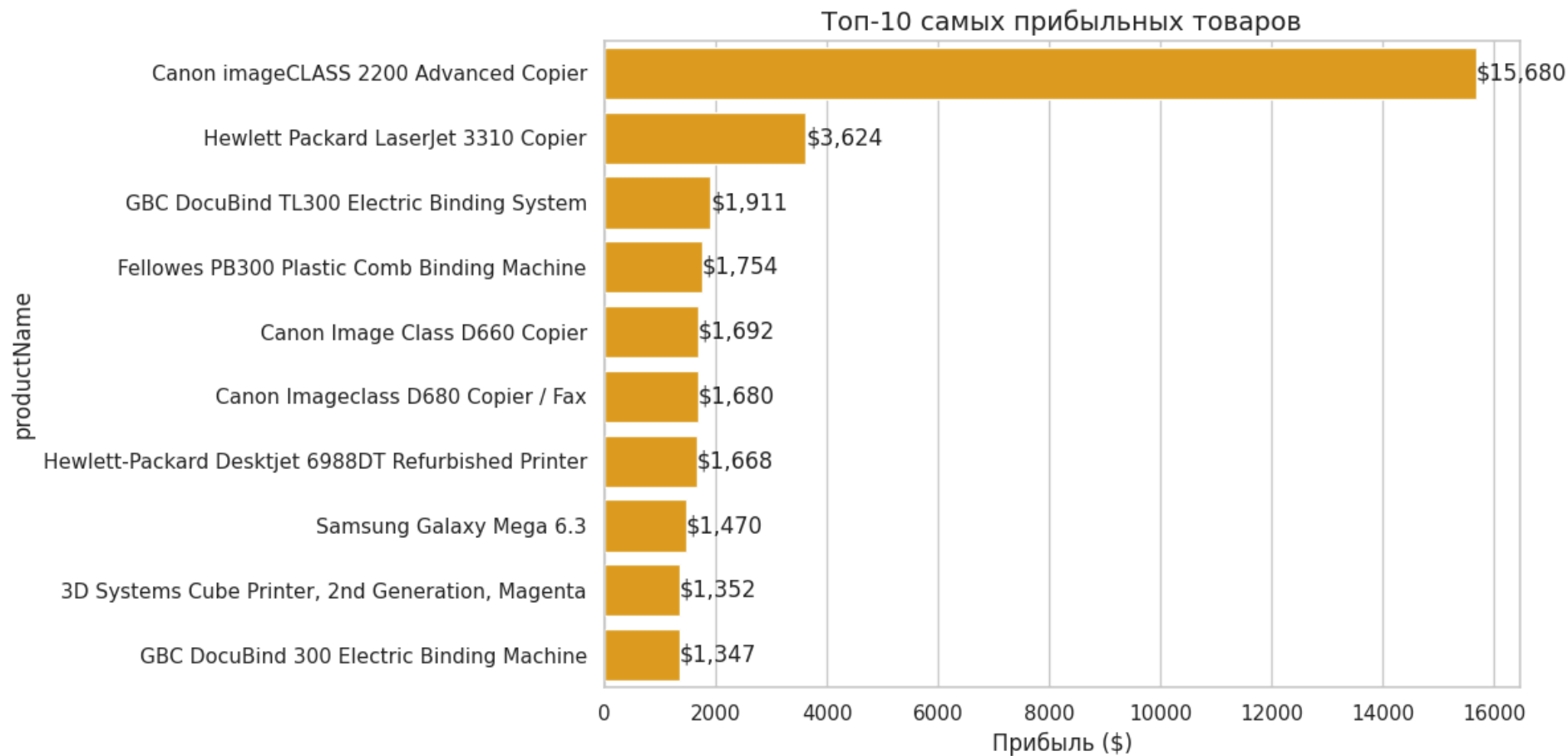
Топ-10 самых прибыльных товаров

```
product_profit = df.groupby('productName')['profit'].sum().nlargest(10)

plt.figure(figsize=(12,6))
sns.barplot(x=product_profit.values, y=product_profit.index, color='orange')
plt.title('Топ-10 самых прибыльных товаров', fontsize=14)
plt.xlabel('Прибыль ($)', fontsize=12)

for i, value in enumerate(product_profit):
    plt.text(value, i, f'${value:,.0f}', va='center', ha='left')

plt.tight_layout()
plt.show()
```



### Сравнение популярности и прибыльности



```
top_combined = df.groupby('productName').agg(
    orders=('orderId', 'count'),
    profit=('profit', 'sum')
).nlargest(10, 'orders')

fig, ax1 = plt.subplots(figsize=(12,6))

# Основной график (столбцы)
color = 'orange' # Изменяем цвет столбцов на оранжевый
bars = ax1.bar(top_combined.index, top_combined['orders'], color=color)
ax1.set_title('Топ товаров: количество заказов vs прибыль', fontsize=14)
ax1.set_ylabel('Количество заказов', color='black', fontsize=12)
ax1.tick_params(axis='y', labelcolor='black')

# Убираем сетку
ax1.grid(False)

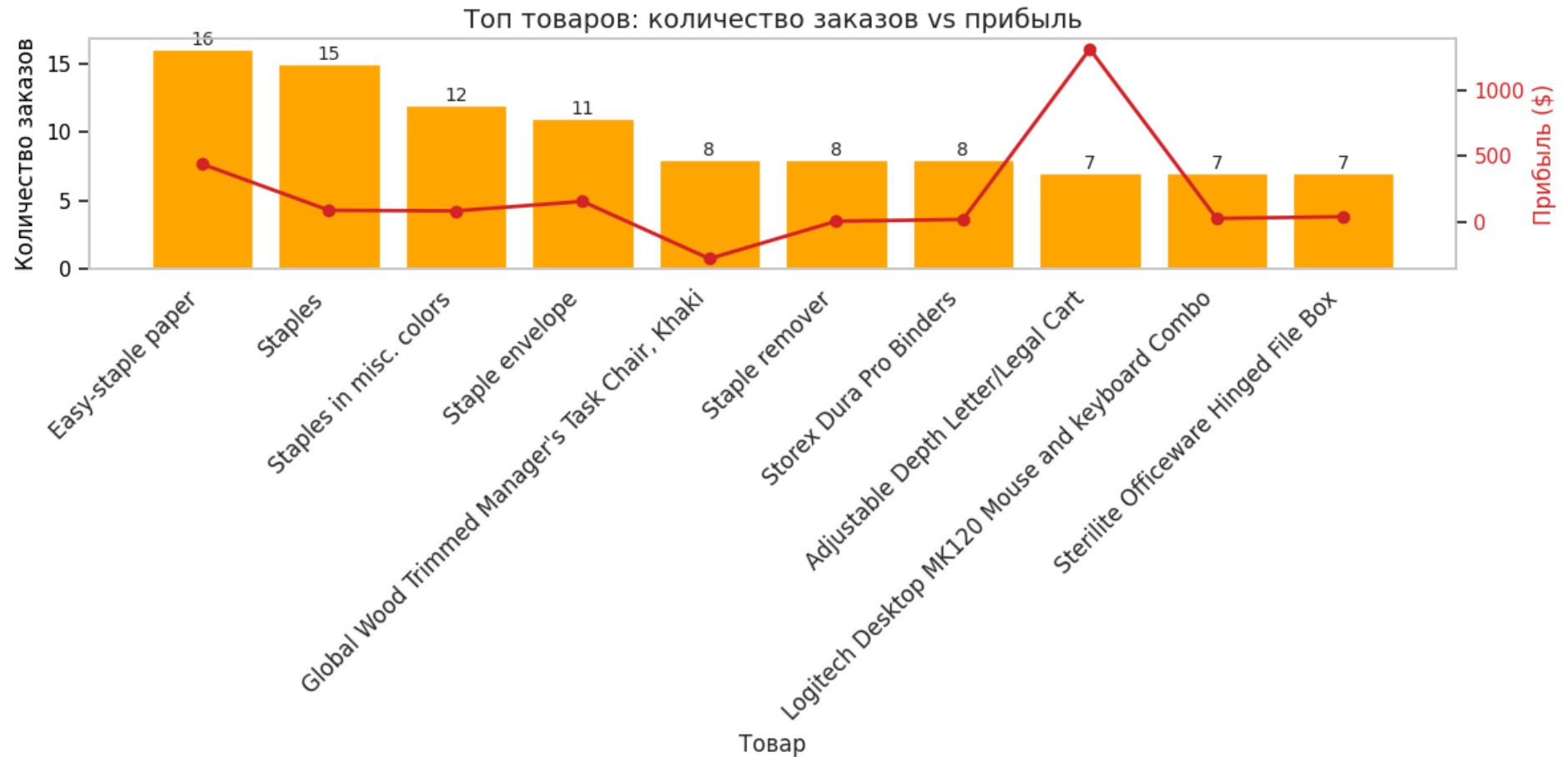
# Явное задание позиций для подписей
ax1.set_xticks(range(len(top_combined.index)))
ax1.set_xticklabels(top_combined.index, rotation=45, ha='right', fontsize=12)
ax1.set_xlabel('Товар', fontsize=12) # Общая подпись оси X

# Вторичная ось (линия прибыли)
ax2 = ax1.twinx()
color = 'tab:red'
ax2.plot(top_combined.index, top_combined['profit'], color=color, marker='o', linewidth=2)
ax2.set_ylabel('Прибыль ($)', color=color, fontsize=12)
ax2.tick_params(axis='y', labelcolor=color)

# Убираем сетку для второй оси
ax2.grid(False)

# Добавление значений на столбцы
for bar in bars:
    height = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width()/2., height,
             f'{int(height)}',
             ha='center', va='bottom', fontsize=10)

fig.tight_layout()
plt.show()
```



### Анализ товаров по категориям

```
category_products = df.groupby(['category', 'productName']).agg(
    orders=('orderId', 'count'),
    sales=('sales', 'sum')
).reset_index()

for category in df['category'].unique():
    top_category = category_products[category_products['category'] == category]\
        .nlargest(3, 'orders')

    print(f"\nТоп-3 товара в категории '{category}':")
    display(top_category[['productName', 'orders', 'sales']])
```



Топ-3 товара в категории 'Furniture':

	productName	orders	sales
171	Global Wood Trimmed Manager's Task Chair, Khaki	8	1865.090
215	KI Adjustable-Height Table	6	1401.474
219	Lesro Sheffield Collection Coffee Table, End T...	6	1784.250

Топ-3 товара в категории 'Office Supplies':

	productName	orders	sales
618	Easy-staple paper	16	1026.076
968	Staples	15	234.456
969	Staples in misc. colors	12	326.188

Топ-3 товара в категории 'Technology':

	productName	orders	sales
1338	Logitech Desktop MK120 Mouse and keyboard Combo	7	458.080
1276	GE 30524EE4	5	2939.850
1346	Logitech G700s Rechargeable Gaming Mouse	5	879.912

▼ **Распределение категорий внутри городов (штатов)**

**3 категории для каждого города (агрегированный анализ)**

```
# Группировка по городам и категориям
city_category = df.groupby(['city', 'category'])['sales'].sum().unstack().fillna(0)

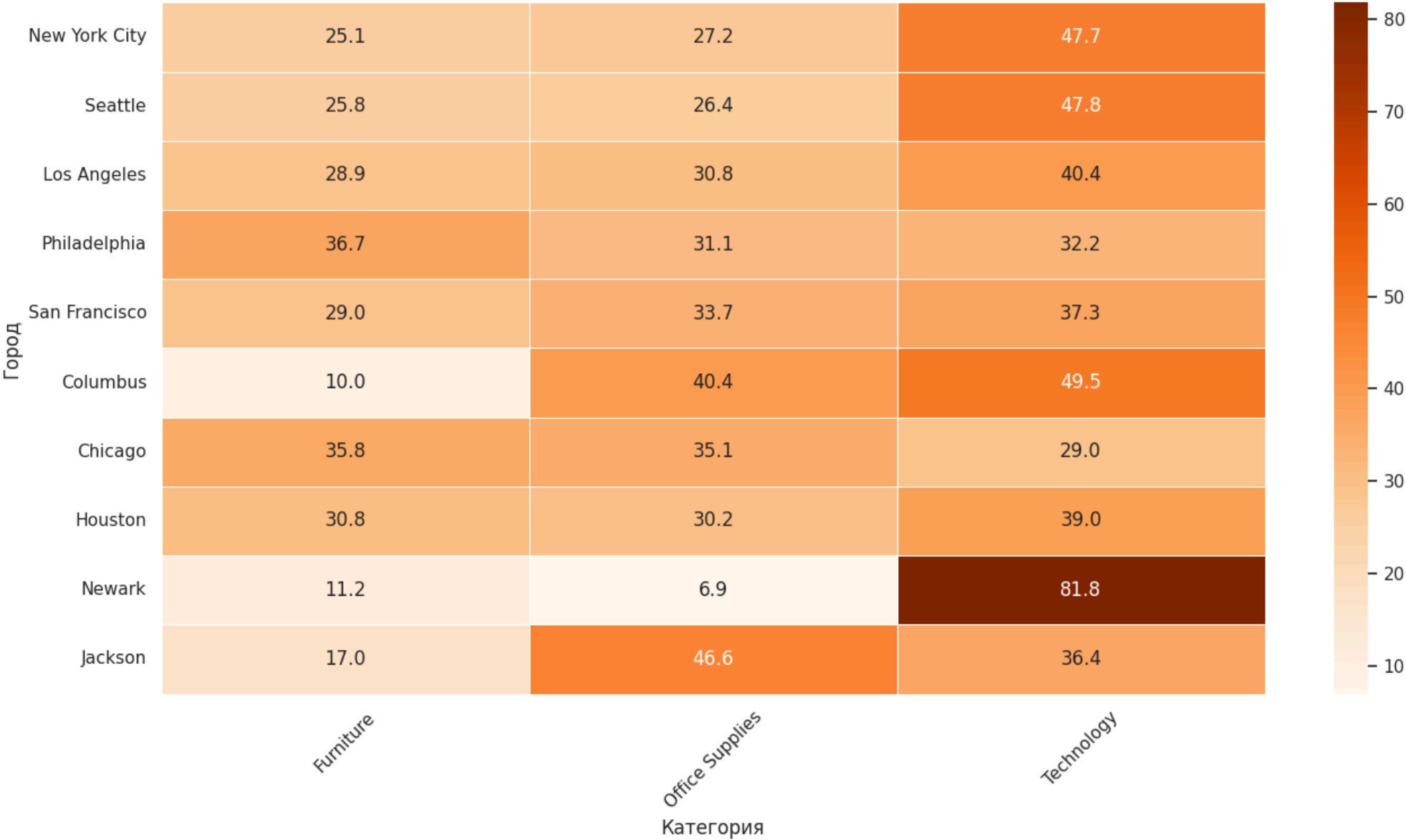
# Нормализация для процентного представления
city_category_pct = city_category.div(city_category.sum(axis=1), axis=0) * 100

# Визуализация для топ-10 городов
top_cities = df.groupby('city')['sales'].sum().nlargest(10).index
plt.figure(figsize=(14, 8))
sns.heatmap(
    city_category_pct.loc[top_cities],
    annot=True, fmt='.1f', cmap='Oranges', # Изменено на 'Oranges'
    linewidths=0.5, linecolor='white'
```

```
)  
plt.title('Доля категорий в продажах по городам (%)', fontsize=14)  
plt.xlabel('Категория', fontsize=12)  
plt.ylabel('Город', fontsize=12)  
plt.xticks(rotation=45)  
plt.tight_layout()  
plt.show()
```



Доля категорий в продажах по городам (%)



Распределение категорий по штатам (интерактивная карта)

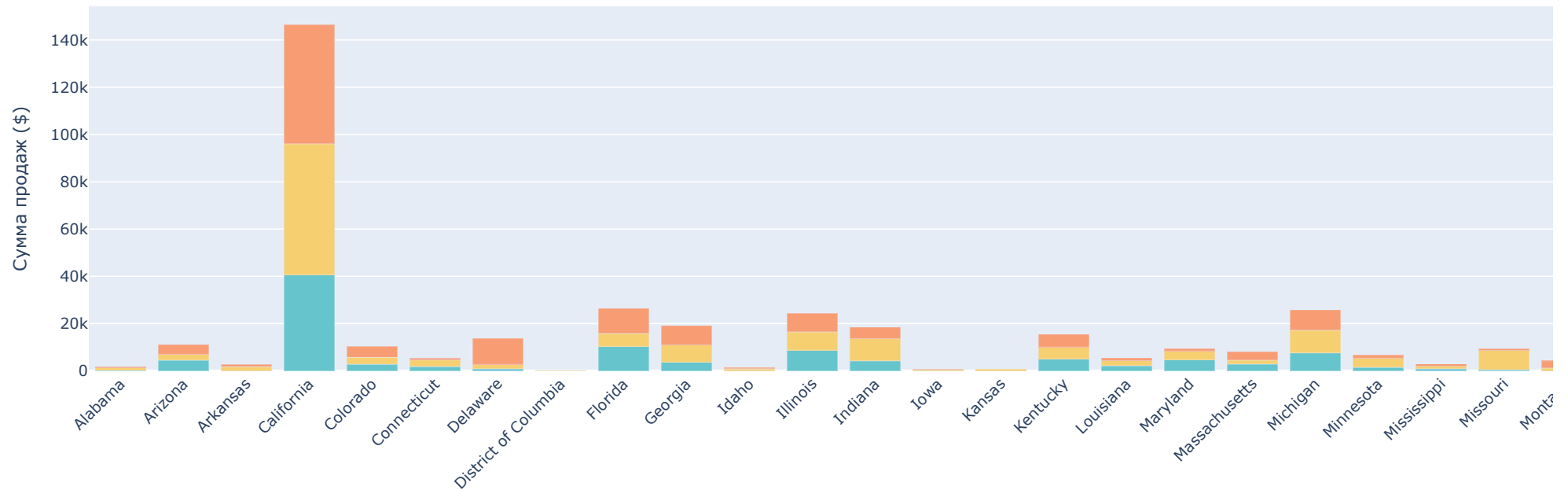
```
import plotly.express as px

# Подготовка данных
state_category = df.groupby(['state', 'category'])['sales'].sum().unstack().reset_index()

# Создание stacked bar chart для каждого штата
fig = px.bar(
    state_category.melt(id_vars='state'),
    x='state',
    y='value',
    color='category',
    title='Распределение продаж по категориям в разрезе штатов',
    labels={'value': 'Сумма продаж ($)', 'state': 'Штат', 'category': 'Категория'},
    color_discrete_sequence=px.colors.qualitative.Pastel
)
fig.update_layout(barmode='stack', xaxis_tickangle=-45)
fig.show()
```



Распределение продаж по категориям в разрезе штатов



## Сравнение структуры продаж между городами

```
# Выбираем 4 крупнейших города для сравнения
top_4_cities = df.groupby('city')['sales'].sum().nlargest(4).index

# Палитра оранжевых оттенков
custom_colors = [ '#FF6347', '#FF7F50', '#FFA500' ]

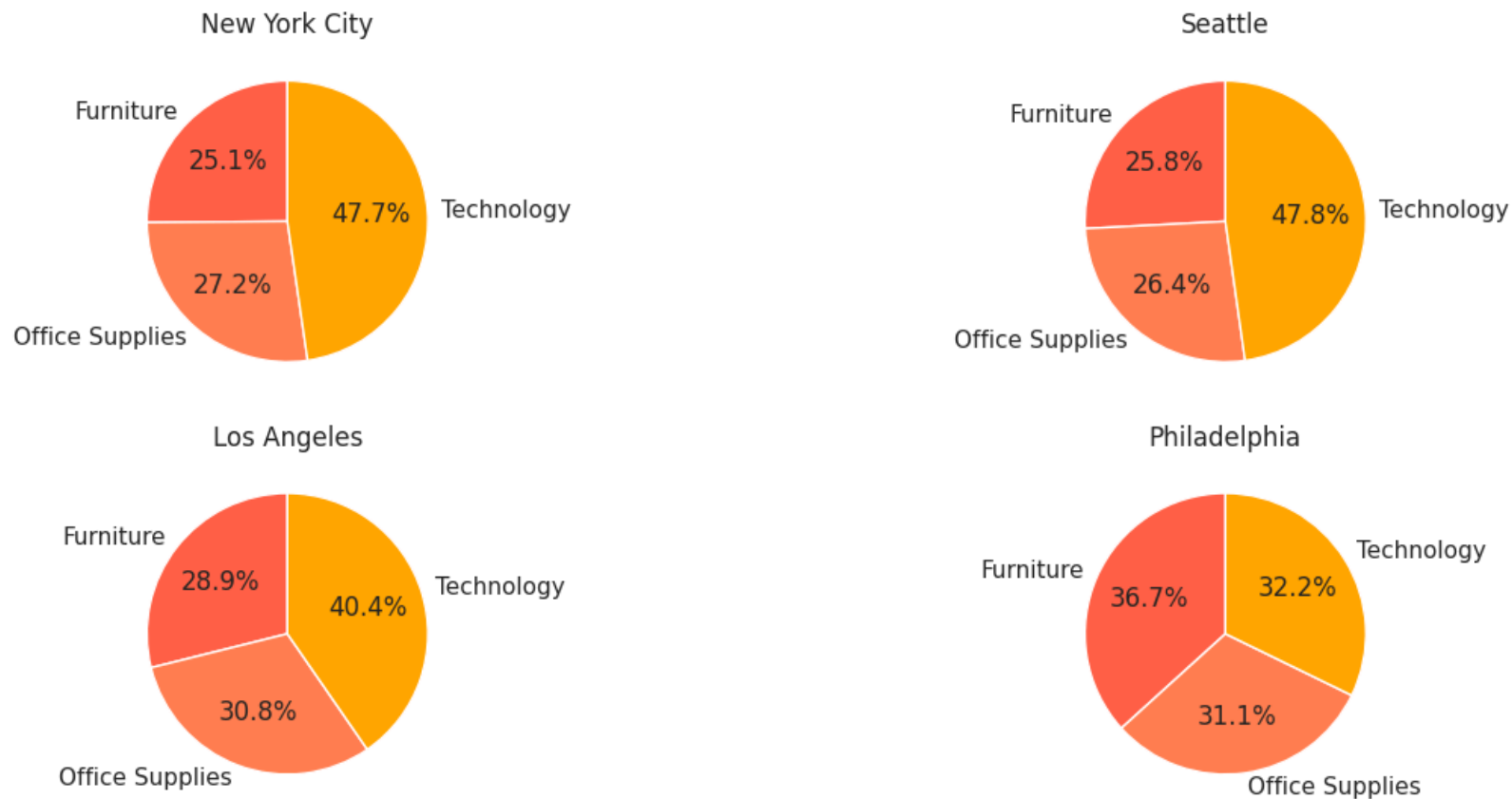
plt.figure(figsize=(14, 6))
for i, city in enumerate(top_4_cities, 1):
    city_data = df[df['city'] == city]
    category_dist = city_data.groupby('category')['sales'].sum()

    plt.subplot(2, 2, i)
    plt.pie(
        category_dist,
        labels=category_dist.index,
        autopct='%1.1f%%',
        startangle=90,
        colors=custom_colors[:len(category_dist)] # Берем столько цветов, сколько нужно
    )
    plt.title(f'{city}', fontsize=12)

plt.suptitle('Сравнение структуры продаж по категориям в крупнейших городах', fontsize=14)
plt.tight_layout()
plt.show()
```



## Сравнение структуры продаж по категориям в крупнейших городах



### Анализ специфичных категорий по регионам

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Предположим, что df уже определен и содержит данные
# Находим категории, которые доминируют в конкретных городах
city_specific = df.groupby(['city', 'category'])['sales'].sum().unstack()
city_specific['dominant'] = city_specific.idxmax(axis=1)
dominant_by_city = city_specific['dominant'].value_counts()
```

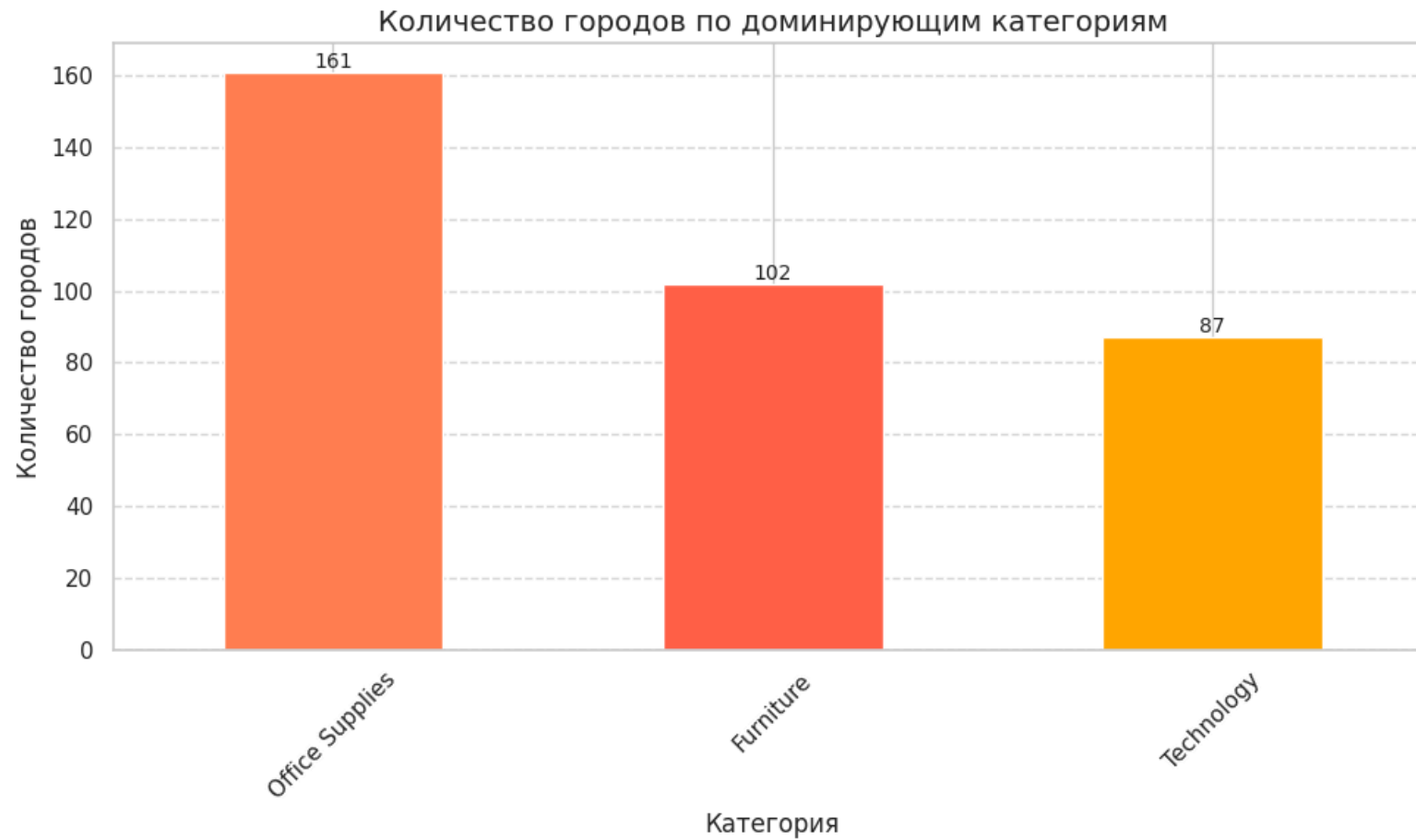
```
# Визуализация
plt.figure(figsize=(10, 6))
```



```
# Задаем цвета для категорий
colors = ['#FF7F50', '#FF6347', '#FFA500'] # Оранжевые оттенки
ax = dominant_by_city.plot(kind='bar', color=colors[:len(dominant_by_city)])

# Добавляем цифры над столбцами
for p in ax.containers:
    ax.bar_label(p, fmt='%.0f', label_type='edge', fontsize=10)

# Настройки графика
plt.title('Количество городов по доминирующим категориям', fontsize=14)
plt.xlabel('Категория', fontsize=12)
plt.ylabel('Количество городов', fontsize=12)
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



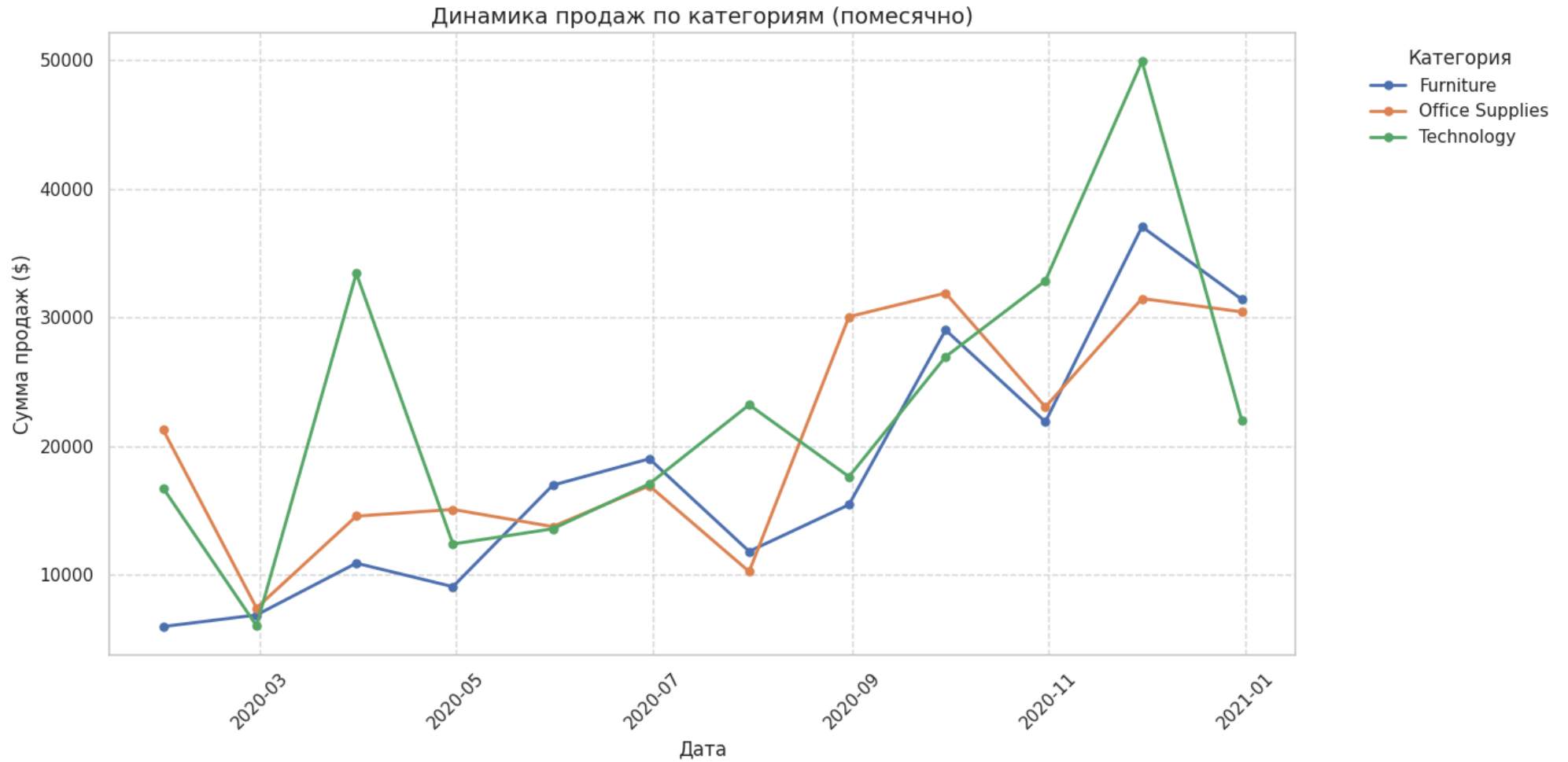
### Общая динамика продаж по категориям (месячный анализ)

```
# Группировка по месяцам и категориям (с актуальным параметром 'ME')
monthly_category = df.groupby([pd.Grouper(key='orderDate', freq='ME'), 'category'])['sales'].sum().unstack()

# Визуализация с улучшенным дизайном
plt.figure(figsize=(14, 7))
for column in monthly_category.columns:
    plt.plot(monthly_category.index, monthly_category[column],
             marker='o', markersize=5, label=column, linewidth=2)

plt.title('Динамика продаж по категориям (помесячно)', fontsize=14)
plt.xlabel('Дата', fontsize=12)
```

```
plt.ylabel('Сумма продаж ($)', fontsize=12)
plt.legend(title='Категория', bbox_to_anchor=(1.05, 1), frameon=False)
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



### Распределение продаж по дням недели

```
import matplotlib.pyplot as plt

# Добавляем день недели
df['day_of_week'] = df['orderDate'].dt.day_name()

# Группировка
weekday_category = df.groupby(['day_of_week', 'category'])['sales'].sum().unstack()

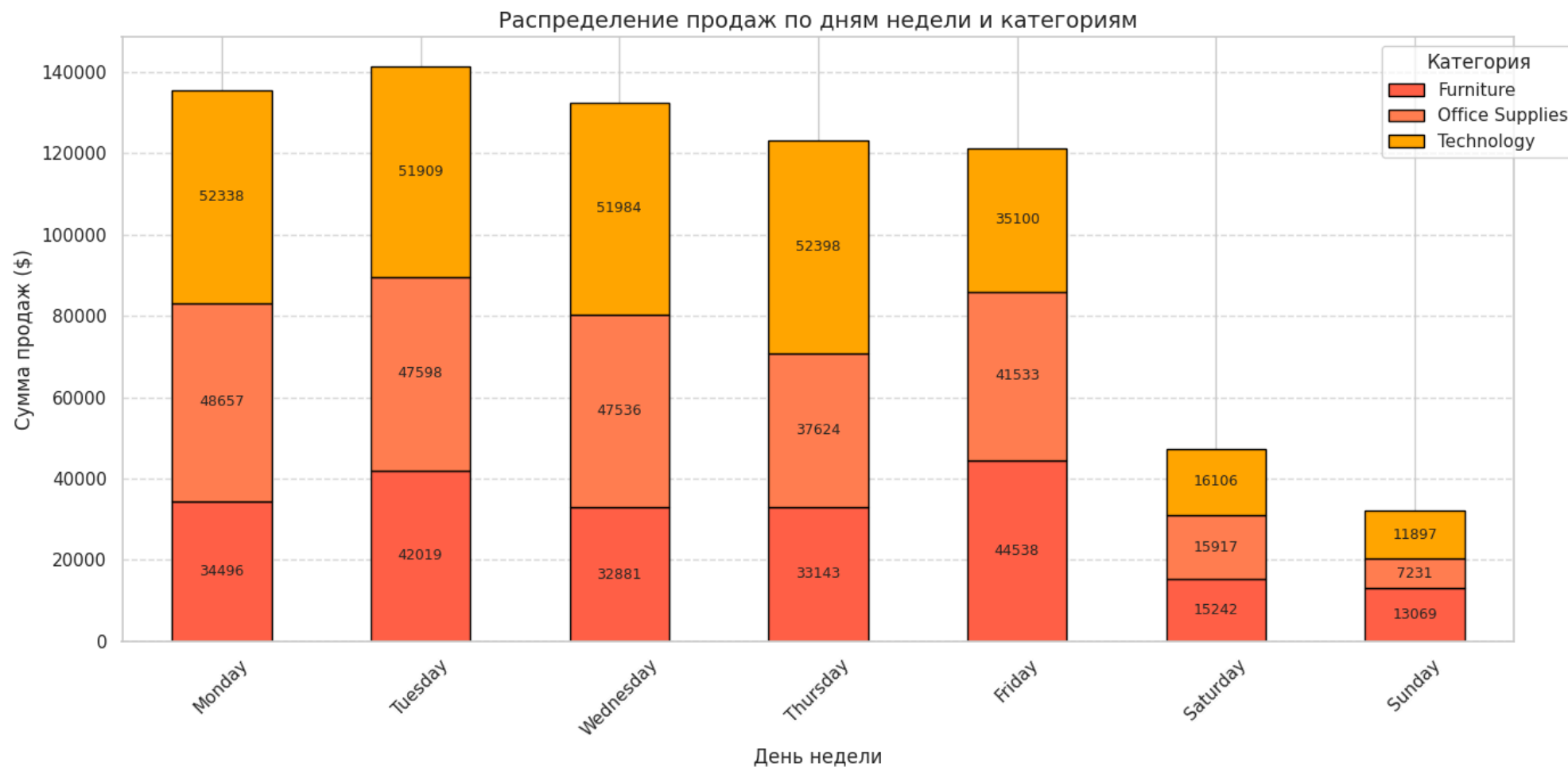
# Упорядочивание дней недели
weekday_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
weekday_category = weekday_category.loc[weekday_order]

# Определяем цвета для категорий
colors = ['#FF6347', '#FF7F50', '#FFA500'] # Цвета для категорий

# Визуализация
fig, ax = plt.subplots(figsize=(14, 7))
bars = weekday_category.plot(
    kind='bar',
    stacked=True,
    ax=ax,
    color=colors, # Используем заданные цвета
    edgecolor='black'
)

# Добавляем цифры на столбцы
for container in bars.containers:
    # Добавляем значения только для ненулевых сегментов
    ax.bar_label(container, fmt='%.0f', label_type='center', fontsize=9)

# Настройки графика
plt.title('Распределение продаж по дням недели и категориям', fontsize=14)
plt.xlabel('День недели', fontsize=12)
plt.ylabel('Сумма продаж ($)', fontsize=12)
plt.xticks(rotation=45)
plt.legend(title='Категория', bbox_to_anchor=(1.05, 1))
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



### Сезонность продаж по категориям

```
# Агрегация данных с правильным частотным параметром
seasonal_data = df.groupby([pd.Grouper(key='orderDate', freq='ME'), 'category'])['sales'].sum().unstack()

# Добавление колонки месяца для анализа сезонности
seasonal_data['month'] = seasonal_data.index.month

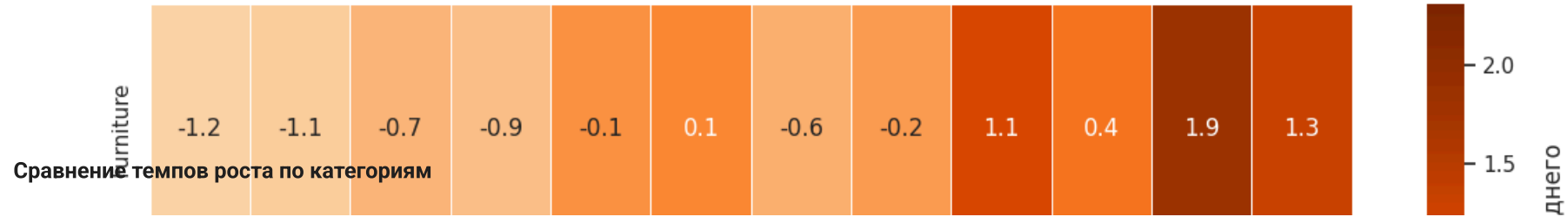
# Группировка по месяцам и расчет средних значений
monthly_avg = seasonal_data.groupby('month').mean()
```

```
# Нормализация данных
seasonal_norm = (monthly_avg - monthly_avg.mean()) / monthly_avg.std()

# Улучшенная визуализация с оранжевой гаммой
plt.figure(figsize=(14, 7))
sns.heatmap(
    seasonal_norm.T,
    cmap='Oranges', # Используем оранжевую цветовую гамму
    center=0,
    annot=True,
    fmt='.1f',
    linewidths=0.5,
    cbar_kws={'label': 'Стандартные отклонения от среднего'}
)
plt.title('Сезонность продаж по категориям', fontsize=14)
plt.xlabel('Месяц', fontsize=12)
plt.ylabel('Категория', fontsize=12)
plt.show()
```



### Сезонность продаж по категориям



```
# Расчет месячного роста
monthly_growth = monthly_category.pct_change() * 100

# Сглаживание (скользящее среднее за 3 месяца)
smooth_growth = monthly_growth.rolling(window=3).mean()

# Визуализация
plt.figure(figsize=(14, 7))
for column in smooth_growth.columns:
    plt.plot(smooth_growth.index, smooth_growth[column],
             label=column, alpha=0.7, linewidth=2)

plt.title('Темпы роста продаж по категориям (% изменения, скользящее среднее 3 мес.)', fontsize=14)
plt.xlabel('Дата', fontsize=12)
plt.ylabel('Изменение продаж (%)', fontsize=12)
plt.axhline(y=0, color='black', linestyle='--', linewidth=1)
plt.legend(title='Категория', bbox_to_anchor=(1.05, 1))
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

