Import libraries

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
```

In [2]:
```
Budget1=pd.read_csv('C:/Users/payal/Desktop/Payal/Internkaksha Project/Budget
```

In [3]:
```
Budget=pd.read_excel('C:/Users/payal/Desktop/Payal/Internkaksha Project/Budget
```

In [4]:
```
Budget.head(3)
```

Out[4]:

	Date	DateKey	Year	Quarter	MonthNum	Month	FiscalYear	FiscalQuarter	FiscalMon
0	2016-04-03	20160403	2016	Q2	4	Apr	FY2016	FQ4	
1	2016-04-04	20160404	2016	Q2	4	Apr	FY2016	FQ4	
2	2016-04-05	20160405	2016	Q2	4	Apr	FY2016	FQ4	

In [5]:
```
Calender=pd.read_excel("AdventureWorks_Database.xlsx", sheet_name="Calender")
Calender.head(3)
```

Out[5]:

	Date	DateKey	Year	Quarter	MonthNum	Month	FiscalYear	FiscalQuarter	FiscalMon
0	2016-04-03	20160403	2016	Q2	4	Apr	FY2016	FQ4	
1	2016-04-04	20160404	2016	Q2	4	Apr	FY2016	FQ4	
2	2016-04-05	20160405	2016	Q2	4	Apr	FY2016	FQ4	

In [6]:
```
Customers=pd.read_excel("AdventureWorks_Database.xlsx", sheet_name="Customers"
Customers.head(3)
```

Out[6]:

	CustomerKey	FirstName	LastName	FullName	BirthDate	MaritalStatus	Gender	YearlyIncon
0	11000	Jon	Yang	Yang, Jon	1966-04-08	M	M	900
1	11001	Eugene	Huang	Huang, Eugene	1965-05-14	S	M	600
2	11002	Ruben	Torres	Torres, Ruben	1965-08-12	M	M	600

In [7]:
```
Product=pd.read_excel("AdventureWorks_Database.xlsx", sheet_name="Product")
Product.head(3)
```

Out[7]:

	ProductKey	ProductName	SubCategory	Category	StandardCost	Color	ListPrice	DaysToMa
0	1	Adjustable Race	NaN	NaN	NaN	NaN	NaN	
1	2	Bearing Ball	NaN	NaN	NaN	NaN	NaN	
2	3	BB Ball Bearing	NaN	NaN	NaN	NaN	NaN	

In [8]:
```
Territory=pd.read_excel("AdventureWorks_Database.xlsx", sheet_name="Territory"
Territory.head(3)
```

Out[8]:

	SalesTerritoryKey	Region	Country	Group	RegionImage
0	1	Northwest	United States	North America	http://www.avising.com/me/LearnPBI /DataSources...
1	2	Northeast	United States	North America	http://www.avising.com/me/LearnPBI /DataSources...
2	3	Central	United States	North America	http://www.avising.com/me/LearnPBI /DataSources...

In [9]:
```
Sales=pd.read_excel("AdventureWorks_Database.xlsx", sheet_name="Sales")
Sales.head(3)
```

Out[9]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrd
0	310	2014-01-01	2014-01-08	21768	1	6	
1	346	2014-01-01	2014-01-08	28389	1	7	
2	346	2014-01-01	2014-01-08	25863	1	1	

3 rows × 25 columns

In [10]:
```
Sales.shape #to check dimensions
```

Out[10]: (58189, 25)

In [11]:
```
Sales.head(5) #to see first 20 rows
```

Out[11]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrd
0	310	2014-01-01	2014-01-08	21768	1	6	
1	346	2014-01-01	2014-01-08	28389	1	7	
2	346	2014-01-01	2014-01-08	25863	1	1	
3	336	2014-01-01	2014-01-08	14501	1	4	
4	346	2014-01-01	2014-01-08	11003	1	9	

5 rows × 25 columns

In [12]:
```
Sales.tail(5) #to see last 5 rows
```

Out[12]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	Sale
58184	561	2016-12-30	2017-01-07	13650	1	9	
58185	584	2016-12-30	2017-01-07	26916	1	9	
58186	605	2016-12-30	2017-01-07	27473	1	9	
58187	538	2016-12-30	2017-01-07	27473	1	9	
58188	490	2016-12-30	2017-01-07	27473	1	9	

5 rows × 25 columns

To see info of sales sheet

In [13]:
```
Sales.info() #to see info of sales sheet
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58189 entries, 0 to 58188
Data columns (total 25 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ProductKey            58189 non-null  int64
 1   OrderDate             58189 non-null  datetime64[ns]
 2   ShipDate              58189 non-null  datetime64[ns]
 3   CustomerKey           58189 non-null  int64
 4   PromotionKey          58189 non-null  int64
 5   SalesTerritoryKey     58189 non-null  int64
 6   SalesOrderNumber      58189 non-null  object
 7   SalesOrderLineNumber  58189 non-null  int64
 8   OrderQuantity         58189 non-null  int64
 9   UnitPrice             58189 non-null  float64
 10  TotalProductCost      58189 non-null  float64
 11  SalesAmount           58189 non-null  float64
 12  TaxAmt                58189 non-null  float64
 13  Unnamed: 13           0 non-null      float64
 14  Unnamed: 14           0 non-null      float64
 15  Unnamed: 15           58189 non-null  float64
 16  Unnamed: 16           58189 non-null  float64
 17  Unnamed: 17           0 non-null      float64
 18  Unnamed: 18           58189 non-null  float64
 19  Unnamed: 19           0 non-null      float64
 20  StandardCost          58189 non-null  float64
 21  List Price            58189 non-null  float64
 22  Unnamed: 22           0 non-null      float64
 23  diif std cost         58189 non-null  int64
 24  diff list price       58189 non-null  int64
dtypes: datetime64[ns](2), float64(14), int64(8), object(1)
memory usage: 11.1+ MB
```

Drop column

In [14]:
```
Sales.drop(['Unnamed: 13','Unnamed: 14','Unnamed: 15','Unnamed: 16','Unnamed:
```

In [15]:
```
Sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58189 entries, 0 to 58188
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ProductKey            58189 non-null  int64
 1   OrderDate             58189 non-null  datetime64[ns]
 2   ShipDate              58189 non-null  datetime64[ns]
 3   CustomerKey           58189 non-null  int64
 4   PromotionKey          58189 non-null  int64
 5   SalesTerritoryKey     58189 non-null  int64
 6   SalesOrderNumber      58189 non-null  object
 7   SalesOrderLineNumber  58189 non-null  int64
 8   OrderQuantity         58189 non-null  int64
 9   UnitPrice             58189 non-null  float64
 10  TotalProductCost      58189 non-null  float64
 11  SalesAmount           58189 non-null  float64
 12  TaxAmt                58189 non-null  float64
 13  Unnamed: 18           58189 non-null  float64
dtypes: datetime64[ns](2), float64(5), int64(6), object(1)
memory usage: 6.2+ MB
```

To check total null values in respective columns

In [16]:
```
pd.isnull(Sales).sum() #to check total null values in respective columns
```

Out[16]:
```
ProductKey              0
OrderDate               0
ShipDate                0
CustomerKey             0
PromotionKey            0
SalesTerritoryKey       0
SalesOrderNumber        0
SalesOrderLineNumber    0
OrderQuantity           0
UnitPrice               0
TotalProductCost        0
SalesAmount             0
TaxAmt                  0
Unnamed: 18             0
dtype: int64
```

In [17]:
```
#Sales.dropna(inplace= True) (to delete a complete row where null values are p
```

Rename column and saved new file

In [18]:
```
Sales_new= Sales.rename(columns={'Unnamed: 18':'Profit'})
Sales_new.head(3)
```

Out[18]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrd
0	310	2014-01-01	2014-01-08	21768	1	6	
1	346	2014-01-01	2014-01-08	28389	1	7	
2	346	2014-01-01	2014-01-08	25863	1	1	

merged 2 sheets 'Sales_new' and 'Customers'

In [19]:
```
merged1= Sales_new.merge(Customers,left_on='CustomerKey', right_on='CustomerKe
```

In [20]:
```
merged1.head(3)
```

Out[20]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrd
0	310	2014-01-01	2014-01-08	21768	1	6	
1	600	2016-04-16	2016-04-23	21768	1	6	
2	346	2014-01-01	2014-01-08	28389	1	7	

3 rows × 30 columns

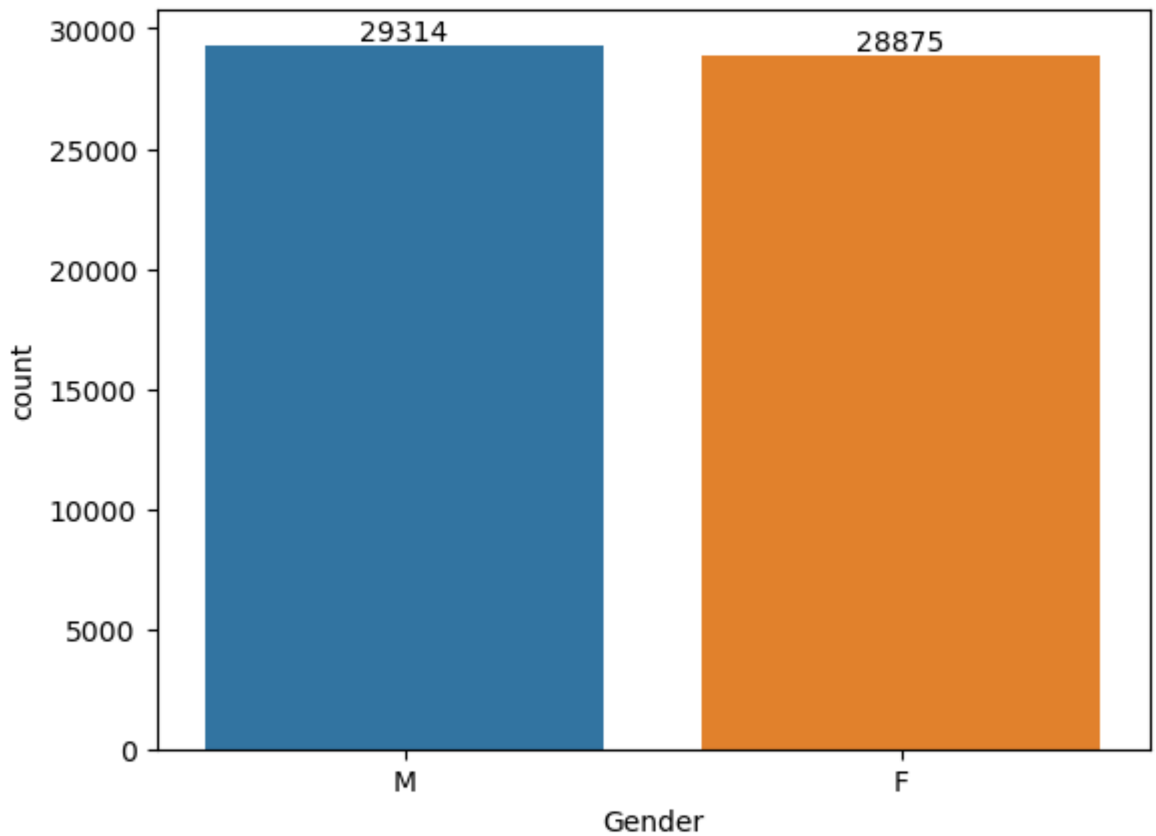
In [21]:
```
merged1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 58189 entries, 0 to 58188
Data columns (total 30 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ProductKey            58189 non-null  int64
 1   OrderDate             58189 non-null  datetime64[ns]
 2   ShipDate              58189 non-null  datetime64[ns]
 3   CustomerKey           58189 non-null  int64
 4   PromotionKey          58189 non-null  int64
 5   SalesTerritoryKey     58189 non-null  int64
 6   SalesOrderNumber      58189 non-null  object
 7   SalesOrderLineNumber  58189 non-null  int64
 8   OrderQuantity         58189 non-null  int64
 9   UnitPrice             58189 non-null  float64
 10  TotalProductCost      58189 non-null  float64
 11  SalesAmount           58189 non-null  float64
 12  TaxAmt                58189 non-null  float64
 13  Profit                58189 non-null  float64
 14  FirstName             58189 non-null  object
 15  LastName              58189 non-null  object
 16  FullName              58189 non-null  object
 17  BirthDate             58189 non-null  datetime64[ns]
 18  MaritalStatus         58189 non-null  object
 19  Gender                58189 non-null  object
 20  YearlyIncome          58189 non-null  int64
 21  TotalChildren         58189 non-null  int64
 22  NumberChildrenAtHome  58189 non-null  int64
 23  Education             58189 non-null  object
 24  Occupation            58189 non-null  object
 25  HouseOwnerFlag        58189 non-null  int64
 26  NumberCarsOwned       58189 non-null  int64
 27  AddressLine1          58189 non-null  object
 28  DateFirstPurchase     58189 non-null  datetime64[ns]
 29  CommuteDistance       58189 non-null  object
dtypes: datetime64[ns](4), float64(5), int64(11), object(10)
memory usage: 13.8+ MB
```

Exploratory Data Analysis

Genderwise Data

In [22]:
```
ax=sns.countplot(x='Gender',data=merged1)
for a in ax.containers:
    ax.bar_label(a)
```



In [23]:
```
a= merged1.groupby(['Gender'], as_index= False)['SalesAmount'].sum().sort_valu
a
```

Out[23]:

	Gender	SalesAmount
0	F	1.478780e+07
1	M	1.452004e+07

Genderwise female do more shopping than male.

Marital Status

In [24]:
```
b= merged1.groupby(['MaritalStatus'], as_index= False)['SalesAmount'].sum().so
b
```

Out[24]:

	MaritalStatus	SalesAmount
0	M	1.515764e+07
1	S	1.415020e+07

Marital status wise married person spend more money on shopping.

In [25]:
```
merged1.head(3)
```

Out[25]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrd
0	310	2014-01-01	2014-01-08	21768	1	6	
1	600	2016-04-16	2016-04-23	21768	1	6	
2	346	2014-01-01	2014-01-08	28389	1	7	

3 rows × 30 columns

Add month column

In [26]:
```
merged1['month']=pd.DatetimeIndex(merged1.OrderDate).month
```

In [27]:
```
#merged1['Year']=pd.DatetimeIndex(merged1.OrderDate).year   (extract year from
#merged1['Day']=pd.DatetimeIndex(merged1.OrderDate).day     (extract day from
#merged1['Date']=pd.DatetimeIndex(merged1.OrderDate).date   (extract date from
```

In [28]:
```
merged1.head(3)
```

Out[28]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrd
0	310	2014-01-01	2014-01-08	21768	1	6	
1	600	2016-04-16	2016-04-23	21768	1	6	
2	346	2014-01-01	2014-01-08	28389	1	7	

3 rows × 31 columns

In [29]:
```
merged1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 58189 entries, 0 to 58188
Data columns (total 31 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ProductKey            58189 non-null  int64
 1   OrderDate             58189 non-null  datetime64[ns]
 2   ShipDate              58189 non-null  datetime64[ns]
 3   CustomerKey           58189 non-null  int64
 4   PromotionKey          58189 non-null  int64
 5   SalesTerritoryKey     58189 non-null  int64
 6   SalesOrderNumber      58189 non-null  object
 7   SalesOrderLineNumber  58189 non-null  int64
 8   OrderQuantity         58189 non-null  int64
 9   UnitPrice             58189 non-null  float64
 10  TotalProductCost      58189 non-null  float64
 11  SalesAmount           58189 non-null  float64
 12  TaxAmt                58189 non-null  float64
 13  Profit                58189 non-null  float64
 14  FirstName             58189 non-null  object
 15  LastName              58189 non-null  object
 16  FullName              58189 non-null  object
 17  BirthDate             58189 non-null  datetime64[ns]
 18  MaritalStatus         58189 non-null  object
 19  Gender                58189 non-null  object
 20  YearlyIncome          58189 non-null  int64
 21  TotalChildren         58189 non-null  int64
 22  NumberChildrenAtHome  58189 non-null  int64
 23  Education             58189 non-null  object
 24  Occupation            58189 non-null  object
 25  HouseOwnerFlag        58189 non-null  int64
 26  NumberCarsOwned       58189 non-null  int64
 27  AddressLine1          58189 non-null  object
 28  DateFirstPurchase     58189 non-null  datetime64[ns]
 29  CommuteDistance       58189 non-null  object
 30  month                 58189 non-null  int64
dtypes: datetime64[ns](4), float64(5), int64(12), object(10)
memory usage: 14.2+ MB
```

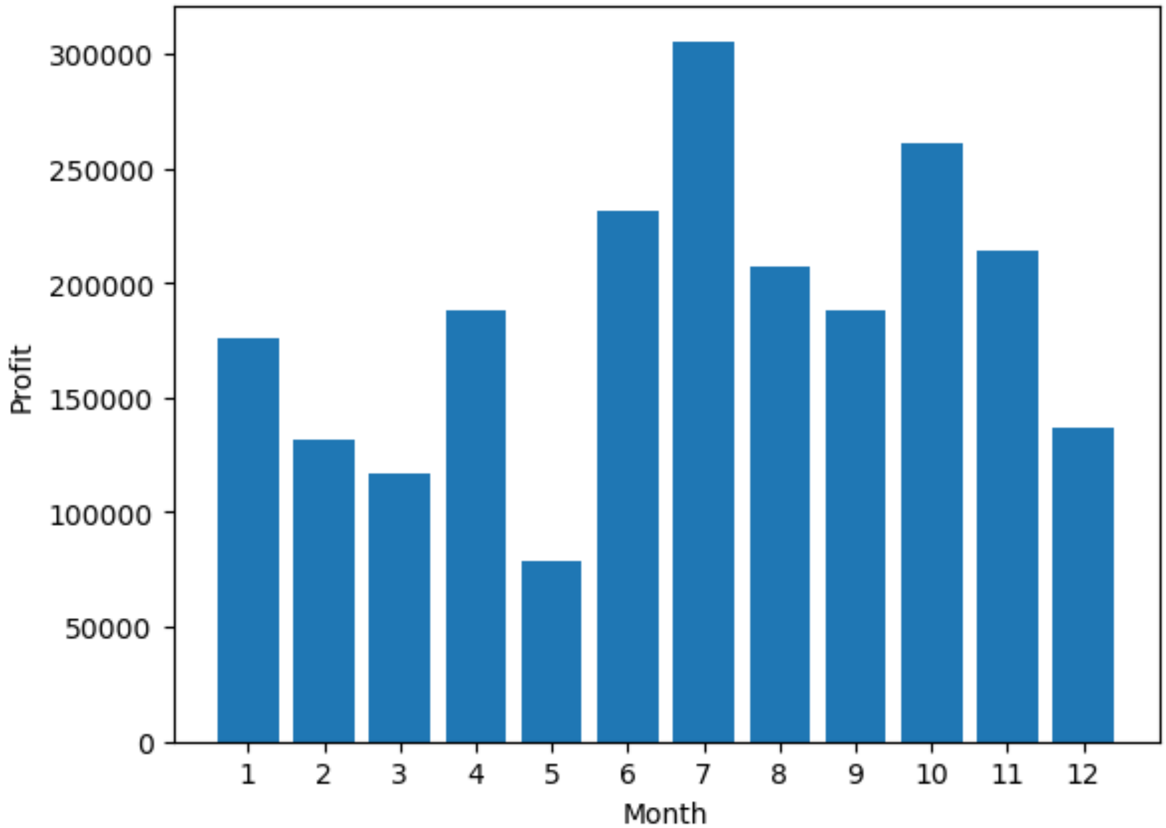
Month wise profit

In [30]:
```
profitmonth=merged1.groupby(['month'],as_index=False)['Profit'].sum()
profitmonth
```

Out[30]:

	month	Profit
0	1	175961.5956
1	2	131409.2392
2	3	116866.4056
3	4	187686.6944
4	5	79076.8800
5	6	231208.9114
6	7	305375.0136
7	8	207036.2780
8	9	188044.4050
9	10	260568.8836
10	11	214131.6922
11	12	136989.9766

In [31]:
```
months= range(1,13)
plt.bar(months, profitmonth['Profit'])
plt.xticks(months)
plt.ylabel('Profit')
plt.xlabel('Month')
plt.show()
```



There are large profit transaction in months July and October.

Region wise profit

Merged sheet 'Sales_new' and 'Territory'

In [32]:
```
merged2=Sales_new.merge(Territory,left_on='SalesTerritoryKey',right_on='SalesT
merged2.head(3)
```

Out[32]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrd
0	310	2014-01-01	2014-01-08	21768	1	6	
1	312	2014-01-19	2014-01-26	21659	1	6	
2	311	2014-01-24	2014-01-31	21710	1	6	

In [33]:
```
merged2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 58189 entries, 0 to 58188
Data columns (total 18 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ProductKey            58189 non-null  int64
 1   OrderDate             58189 non-null  datetime64[ns]
 2   ShipDate              58189 non-null  datetime64[ns]
 3   CustomerKey           58189 non-null  int64
 4   PromotionKey          58189 non-null  int64
 5   SalesTerritoryKey     58189 non-null  int64
 6   SalesOrderNumber      58189 non-null  object
 7   SalesOrderLineNumber  58189 non-null  int64
 8   OrderQuantity         58189 non-null  int64
 9   UnitPrice             58189 non-null  float64
 10  TotalProductCost      58189 non-null  float64
 11  SalesAmount           58189 non-null  float64
 12  TaxAmt                58189 non-null  float64
 13  Profit                58189 non-null  float64
 14  Region                58189 non-null  object
 15  Country               58189 non-null  object
 16  Group                 58189 non-null  object
 17  RegionImage           58189 non-null  object
dtypes: datetime64[ns](2), float64(5), int64(6), object(5)
memory usage: 8.4+ MB
```

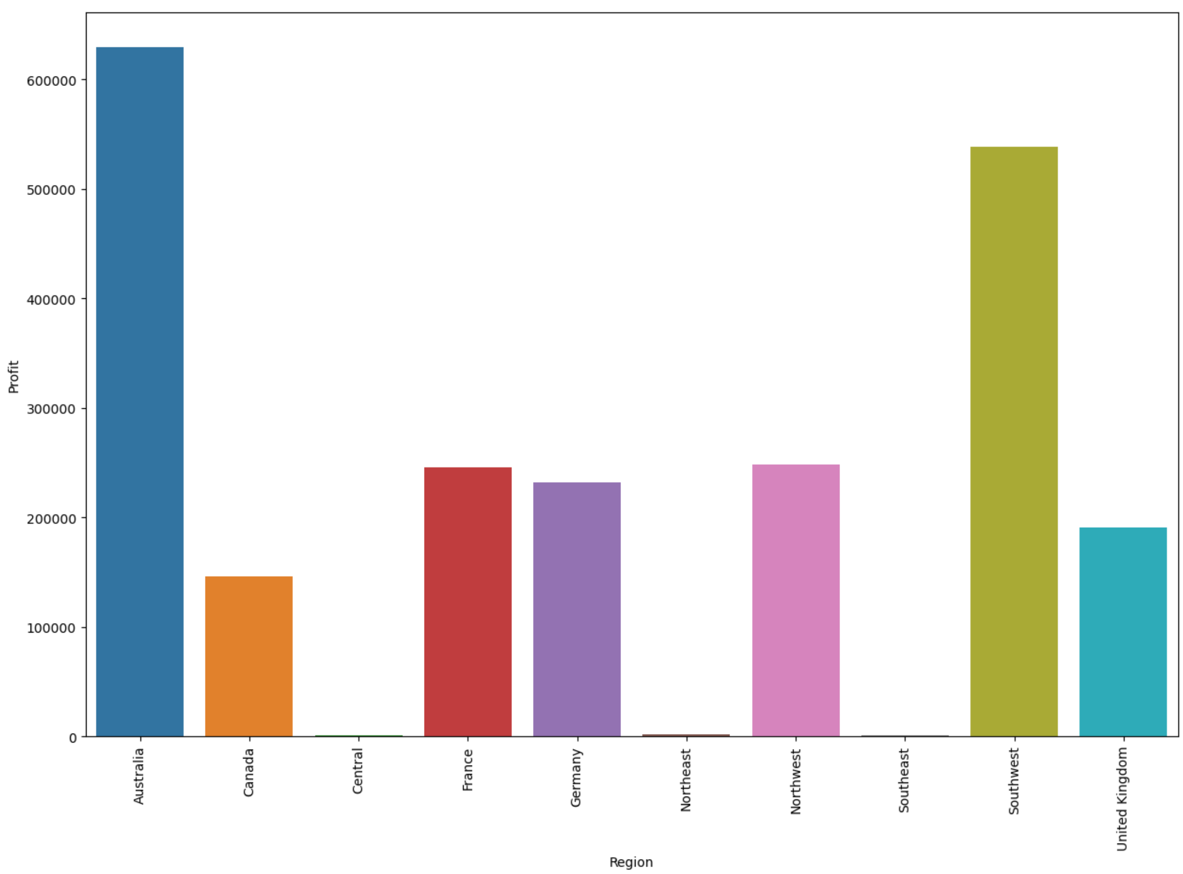
Region wise profit

In [34]:
```
RegionProfit=merged2.groupby(['Region'],as_index= False)['Profit'].sum()
RegionProfit
```

Out[34]:

	Region	Profit
0	Australia	629610.5611
1	Canada	145732.1697
2	Central	1247.4264
3	France	245792.6844
4	Germany	232018.6836
5	Northeast	1696.6601
6	Northwest	248179.7661
7	Southeast	1097.7456
8	Southwest	538090.7647
9	United Kingdom	190889.5135

In [35]:
```
plt.figure(figsize=(15,10))
sns.barplot(data=RegionProfit, x="Region", y="Profit")
plt.xticks(rotation=90)
plt.show()
```



There are more profit transactions in Australia and Southwest.

Find Age from Birthdate

In [36]:
```
from datetime import date
```

In [37]:
```
df=pd.DataFrame(data=merged1)
def age_calculate(BirthDate):
    today=date.today()
    age=today.year - BirthDate.year - ((today.month,today.day)<(BirthDate.mont
    return age
Age=df['BirthDate'].apply(age_calculate)
print(Age)
```

```
0        77
1        77
2        59
3        77
4        77
         ..
58184    53
58185    54
58186    54
58187    54
58188    54
Name: BirthDate, Length: 58189, dtype: int64
```

Import Libraries

In [38]:
```
import datetime as DT
import io
import numpy as np
import pandas as pd
```

In [39]:
```
merged1['Age']=merged1['BirthDate'].apply('{:06}'.format)
now=pd.to_datetime('now')
merged1['BirthDate']=pd.to_datetime(merged1['BirthDate'], format='%m%d%y')
merged1['BirthDate']=df['BirthDate'].where(merged1['BirthDate']<now, merged1['
merged1['age']=(now - df['BirthDate']).astype('<m8[Y]')
merged1.head(3)
```

```
C:\Users\payal\AppData\Local\Temp\ipykernel_20592\3290232451.py:2: FutureWarn
ing: The parsing of 'now' in pd.to_datetime without `utc=True` is deprecated.
In a future version, this will match Timestamp('now') and Timestamp.now()
  now=pd.to_datetime('now')
```

Out[39]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrd
0	310	2014-01-01	2014-01-08	21768	1	6	
1	600	2016-04-16	2016-04-23	21768	1	6	
2	346	2014-01-01	2014-01-08	28389	1	7	

3 rows × 33 columns

In [40]:
```
merged1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 58189 entries, 0 to 58188
Data columns (total 33 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ProductKey            58189 non-null  int64
 1   OrderDate             58189 non-null  datetime64[ns]
 2   ShipDate              58189 non-null  datetime64[ns]
 3   CustomerKey           58189 non-null  int64
 4   PromotionKey          58189 non-null  int64
 5   SalesTerritoryKey     58189 non-null  int64
 6   SalesOrderNumber      58189 non-null  object
 7   SalesOrderLineNumber  58189 non-null  int64
 8   OrderQuantity         58189 non-null  int64
 9   UnitPrice             58189 non-null  float64
 10  TotalProductCost      58189 non-null  float64
 11  SalesAmount           58189 non-null  float64
 12  TaxAmt                58189 non-null  float64
 13  Profit                58189 non-null  float64
 14  FirstName             58189 non-null  object
 15  LastName              58189 non-null  object
 16  FullName              58189 non-null  object
 17  BirthDate             58189 non-null  datetime64[ns]
 18  MaritalStatus         58189 non-null  object
 19  Gender                58189 non-null  object
 20  YearlyIncome          58189 non-null  int64
 21  TotalChildren         58189 non-null  int64
 22  NumberChildrenAtHome  58189 non-null  int64
 23  Education             58189 non-null  object
 24  Occupation            58189 non-null  object
 25  HouseOwnerFlag        58189 non-null  int64
 26  NumberCarsOwned       58189 non-null  int64
 27  AddressLine1          58189 non-null  object
 28  DateFirstPurchase     58189 non-null  datetime64[ns]
 29  CommuteDistance       58189 non-null  object
 30  month                 58189 non-null  int64
 31  Age                   58189 non-null  object
 32  age                   58189 non-null  float64
dtypes: datetime64[ns](4), float64(6), int64(12), object(11)
memory usage: 15.1+ MB
```

In [41]:
```
merged1.drop(['Age'], axis= 1, inplace= True)
```

In [42]:
```
merged1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 58189 entries, 0 to 58188
Data columns (total 32 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ProductKey            58189 non-null  int64
 1   OrderDate             58189 non-null  datetime64[ns]
 2   ShipDate              58189 non-null  datetime64[ns]
 3   CustomerKey           58189 non-null  int64
 4   PromotionKey          58189 non-null  int64
 5   SalesTerritoryKey     58189 non-null  int64
 6   SalesOrderNumber      58189 non-null  object
 7   SalesOrderLineNumber  58189 non-null  int64
 8   OrderQuantity         58189 non-null  int64
 9   UnitPrice             58189 non-null  float64
 10  TotalProductCost      58189 non-null  float64
 11  SalesAmount           58189 non-null  float64
 12  TaxAmt                58189 non-null  float64
 13  Profit                58189 non-null  float64
 14  FirstName             58189 non-null  object
 15  LastName              58189 non-null  object
 16  FullName              58189 non-null  object
 17  BirthDate             58189 non-null  datetime64[ns]
 18  MaritalStatus         58189 non-null  object
 19  Gender                58189 non-null  object
 20  YearlyIncome          58189 non-null  int64
 21  TotalChildren         58189 non-null  int64
 22  NumberChildrenAtHome  58189 non-null  int64
 23  Education             58189 non-null  object
 24  Occupation            58189 non-null  object
 25  HouseOwnerFlag        58189 non-null  int64
 26  NumberCarsOwned       58189 non-null  int64
 27  AddressLine1          58189 non-null  object
 28  DateFirstPurchase     58189 non-null  datetime64[ns]
 29  CommuteDistance       58189 non-null  object
 30  month                 58189 non-null  int64
 31  age                   58189 non-null  float64
dtypes: datetime64[ns](4), float64(6), int64(12), object(10)
memory usage: 14.7+ MB
```

In [43]:
```
merged1.head(3)
```

Out[43]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrd
0	310	2014-01-01	2014-01-08	21768	1	6	
1	600	2016-04-16	2016-04-23	21768	1	6	
2	346	2014-01-01	2014-01-08	28389	1	7	

3 rows × 32 columns

Import Libraries

In [44]:
```
!pip install openpyxl plotly -q
```

In [45]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns; sns.set_theme()
import plotly.figure_factory as ff
from itertools import combinations
from collections import Counter
import datetime as dt
import warnings
warnings.filterwarnings('ignore')
```

```
C:\Users\payal\anaconda3\Lib\site-packages\paramiko\transport.py:219: Cryptog
raphyDeprecationWarning: Blowfish has been deprecated
  "class": algorithms.Blowfish,
```

merge sheet 'Sales_new' and 'Product'

In [46]:
```
merged3=Sales_new.merge(Product,left_on='ProductKey',right_on='ProductKey')
merged3.head(3)
```

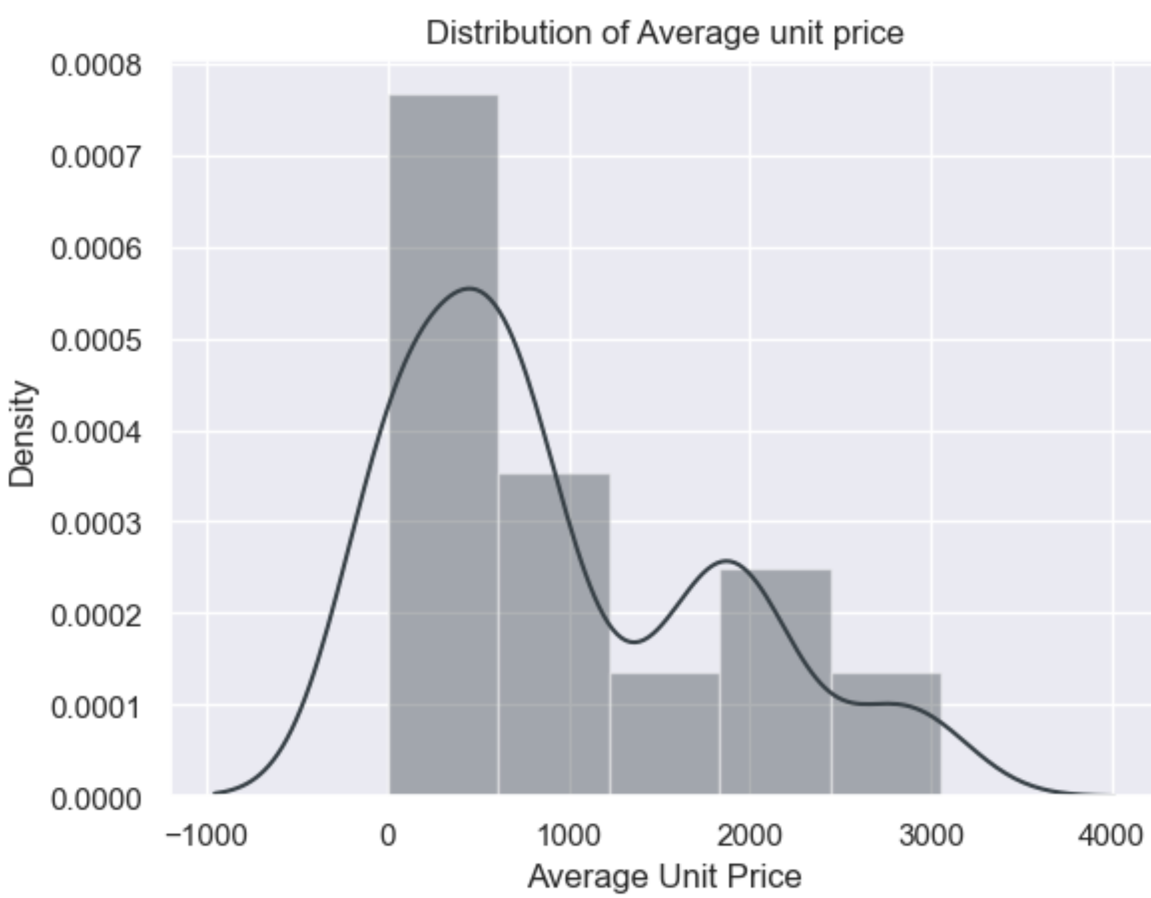
Out[46]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrd
0	310	2014-01-01	2014-01-08	21768	1	6	
1	310	2014-01-02	2014-01-09	16624	1	9	
2	310	2014-01-05	2014-01-12	27601	1	4	

3 rows × 26 columns

analysing unit price

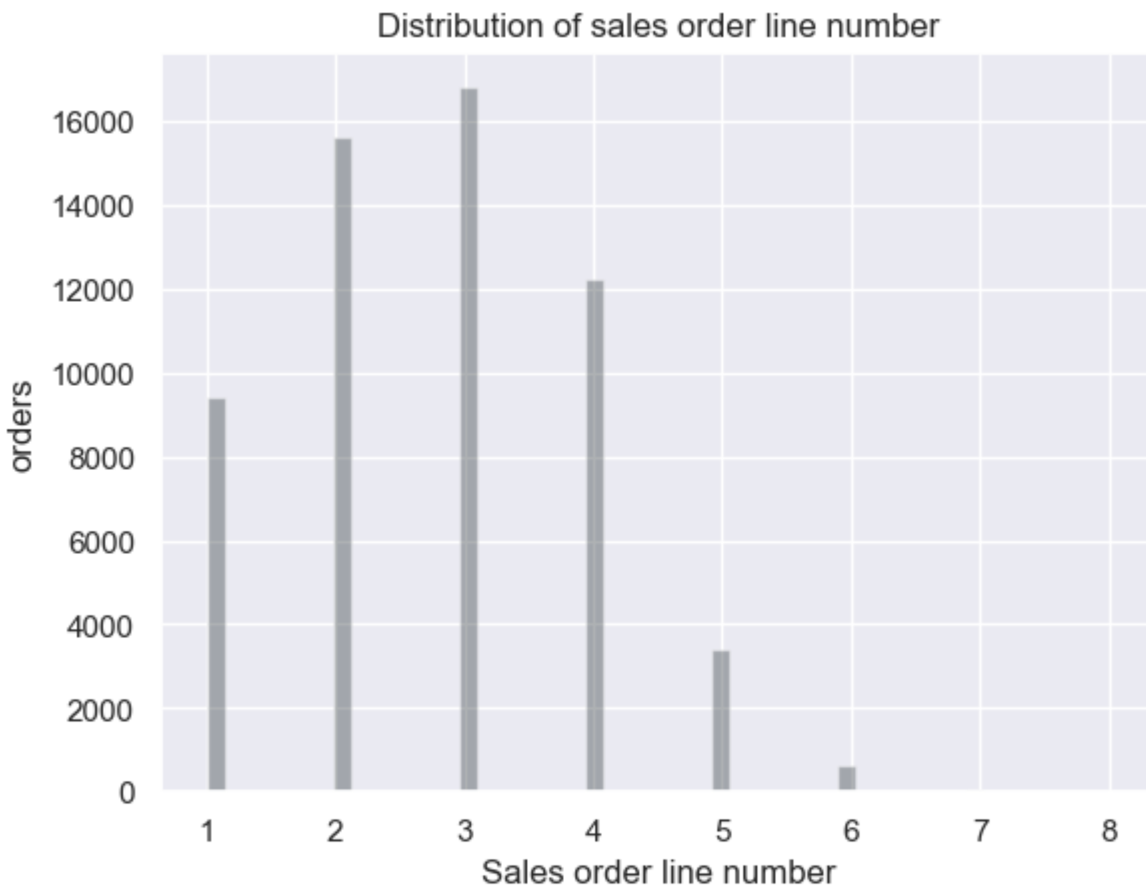
In [47]:
```
Avg_unit_price = merged3.groupby(['ProductKey'])['UnitPrice'].mean()
axis = sns.distplot(Avg_unit_price, kde=True, hist=True, color='#374046')
axis.set(title='Distribution of Average unit price',
       xlabel='Average Unit Price');
```



Maximum of the unit price is below $1000.

Sales order line number distribution

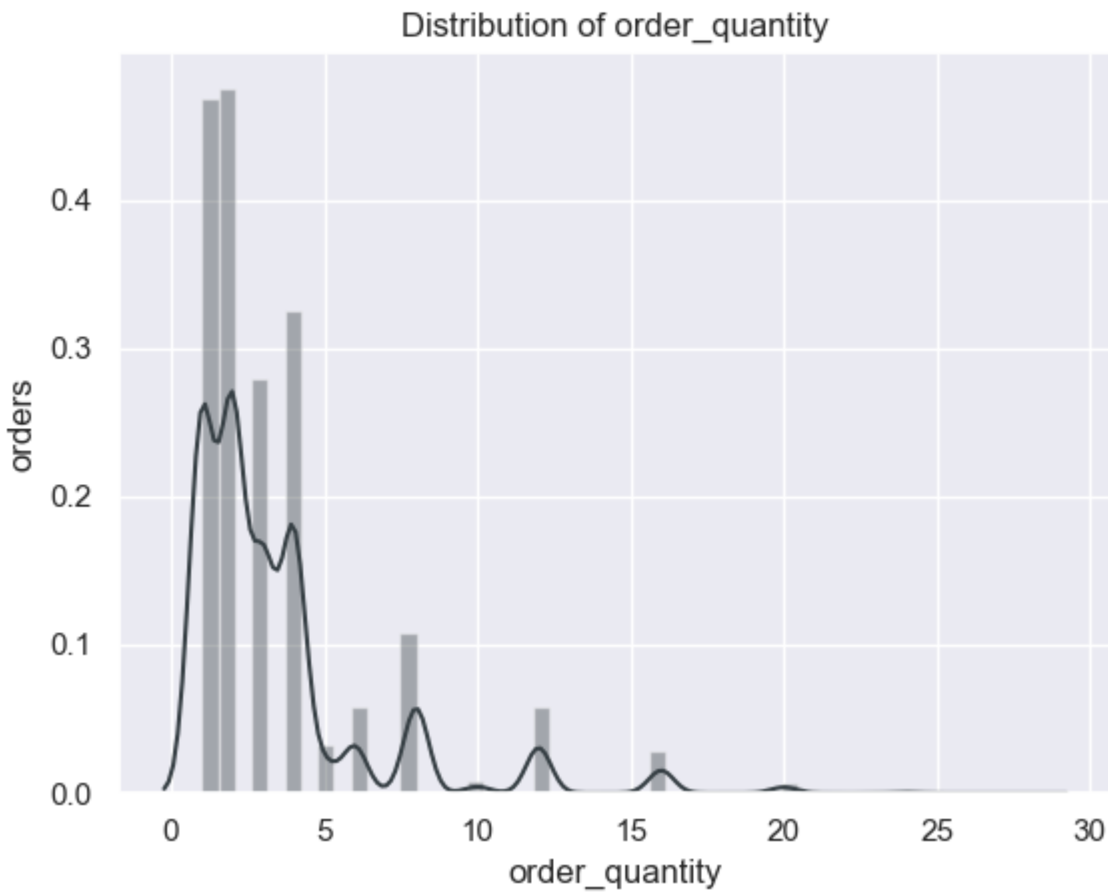
In [48]:
```
n_salesordernumber = Sales_new.groupby(['SalesOrderNumber'])['SalesOrderLineNu
axis2 = sns.distplot(n_salesordernumber, kde=False, color='#374045')
axis2.set(title='Distribution of sales order line number',
       xlabel='Sales order line number',
       ylabel='orders');
```



Most of the time 2-3 products are ordered in a single order.

Sales Order Quantity distribution

In [49]:
```
n_order_quantity = Sales_new.groupby(['SalesOrderNumber'])['OrderQuantity'].su
axis3 = sns.distplot(n_order_quantity, kde=True, hist=True,color='#374045')
axis3.set(title='Distribution of order_quantity',
       xlabel='order_quantity',
       ylabel='orders');
```



Maximum quantity ordered for a product is below 5.

Sales

Year wise sales

Add year column

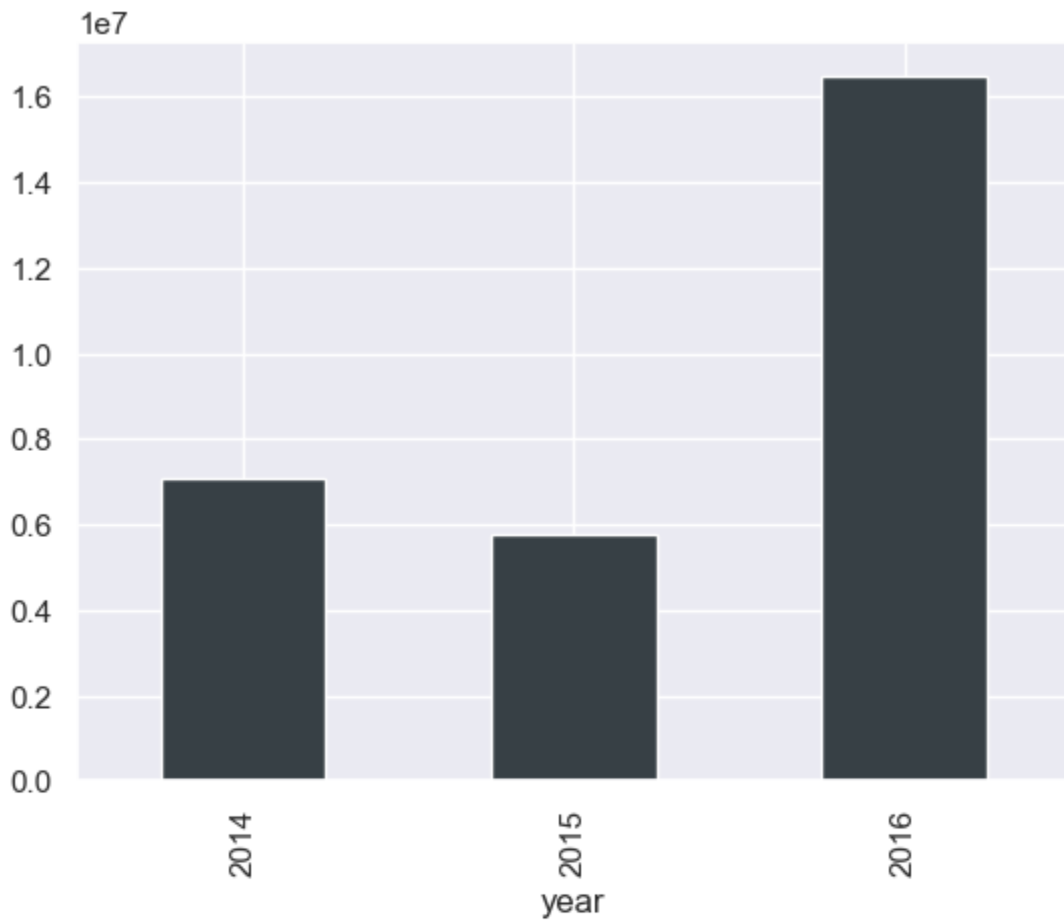
In [50]:
```
merged1['year']=pd.DatetimeIndex(merged1.OrderDate).year
merged1.head(3)
```

Out[50]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrd
0	310	2014-01-01	2014-01-08	21768	1	6	
1	600	2016-04-16	2016-04-23	21768	1	6	
2	346	2014-01-01	2014-01-08	28389	1	7	

3 rows × 33 columns

In [51]:
```
merged1.groupby('year')['SalesAmount'].sum().plot(kind='bar', color='#374045')
```



The year 2016 saw a sudden powerful movement in sales.

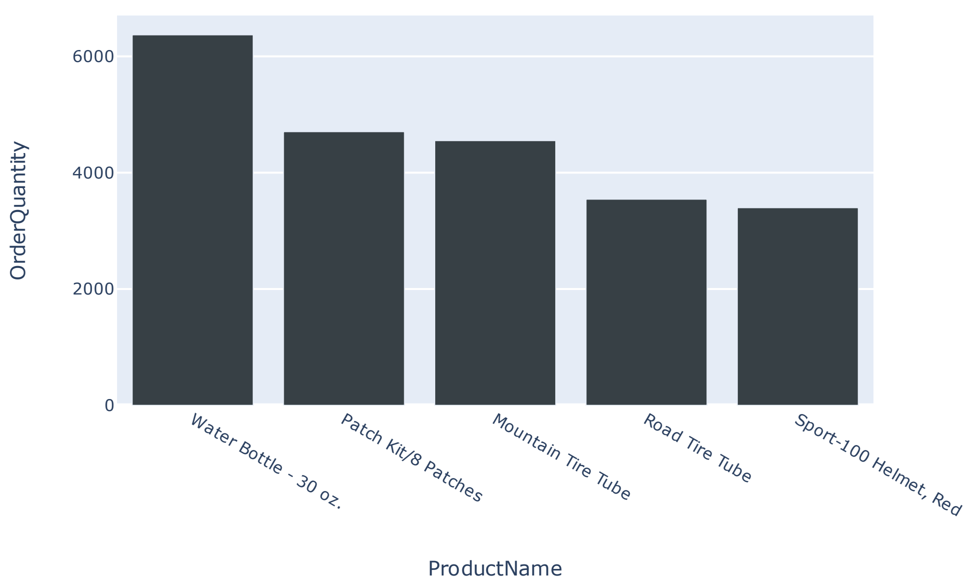
Top 5 Selling Product

In [52]:
```
top_selling_product = merged3.groupby(['Category', 'SubCategory', 'ProductName
top_selling_product
```

Out[52]:

			OrderQuantity
Category	**SubCategory**	**ProductName**	
Accessories	**Bottles and Cages**	**Water Bottle - 30 oz.**	6370
	Tires and Tubes	**Patch Kit/8 Patches**	4705
		Mountain Tire Tube	4551
		Road Tire Tube	3544
	Helmets	**Sport-100 Helmet, Red**	3398

In [53]:
```
top_selling_product.reset_index(inplace=True)
fig = px.bar(top_selling_product, x='ProductName', y='OrderQuantity',color_dis
fig.update_layout(
    autosize=True,
    width=500,
    height=300,
    margin=dict(
        l=25,
        r=25,
        b=10,
        t=10,
    ),
    font=dict(size=8))
fig.show()
```



Add year column in merged 3

In [54]:
```
merged3['year']=pd.DatetimeIndex(merged1.OrderDate).year
merged3.head(3)
```

Out[54]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrd
0	310	2014-01-01	2014-01-08	21768	1	6	
1	310	2014-01-02	2014-01-09	16624	1	9	
2	310	2014-01-05	2014-01-12	27601	1	4	

3 rows × 27 columns

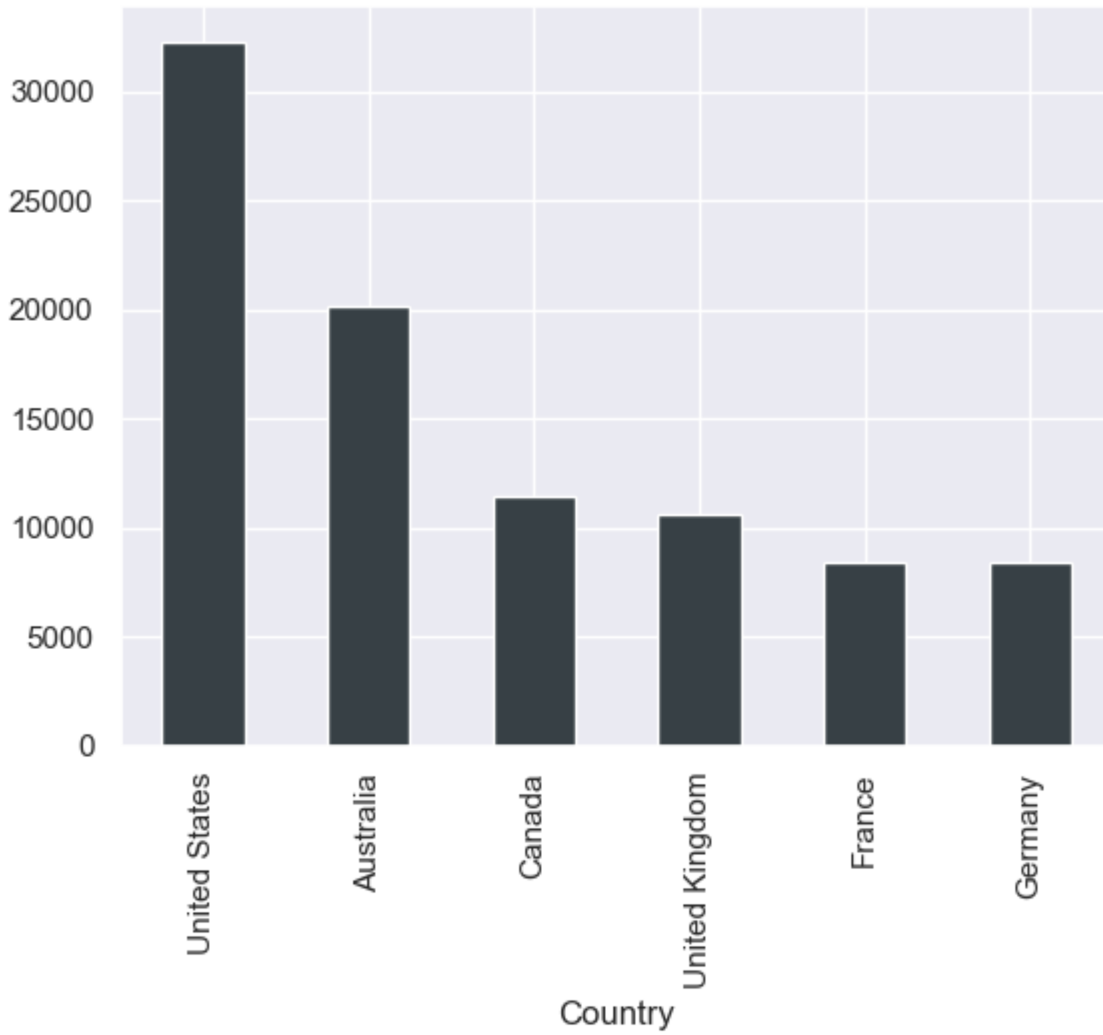
In [55]:
```
category_subcategory_qty = merged3.groupby(['year','Category', 'SubCategory'])
category_subcategory_qty = category_subcategory_qty.sort_values(['year', 'Cate
category_subcategory_qty.style.bar(subset=['OrderQuantity'], color='#D9B300')
```

Out[55]:

year	Category	SubCategory	OrderQuantity
2014	Accessories	Bottles and Cages	634
		Helmets	537
	Bikes	Mountain Bikes	600
		Road Bikes	1592
		Touring Bikes	28
2015	Accessories	Bike Stands	120
		Bottles and Cages	2242
		Helmets	1070
		Tires and Tubes	1169
	Bikes	Mountain Bikes	171
		Road Bikes	167
	Clothing	Jerseys	173
2016	Accessories	Bike Racks	493
		Bike Stands	274
		Bottles and Cages	9179
		Cleaners	1381
		Fenders	3239
		Helmets	8078
		Hydration Packs	1124
		Tires and Tubes	24349
	Bikes	Mountain Bikes	6996
		Road Bikes	10936
		Touring Bikes	3382
	Clothing	Caps	3178
		Gloves	2143
		Jerseys	4895
		Shorts	1491
		Socks	856
		Vests	824

Country wise quantity ordered

In [56]:
```
country_qty_sales = merged2.groupby('Country')['OrderQuantity'].sum().sort_val
country_qty_sales.plot(kind='bar', color='#374045');
```



High quantity of products is ordered from Australia and United States.

Profit

Overall profit based on order year, category and subcategory

In [57]:
```
cat_subcat_profit = merged3.groupby(['year','Category', 'SubCategory'])['Profi

#Sorting the results
cat_subcat_profit = cat_subcat_profit.sort_values(['year', 'Category'], ascend
cat_subcat_profit.style.bar(subset=['Profit'], color='#D9B300')
```

Out[57]:

year	Category	SubCategory	Profit
2014	Accessories	Bottles and Cages	1329.451800
		Helmets	5219.156900
	Bikes	Mountain Bikes	139185.066200
		Road Bikes	196566.973000
		Touring Bikes	3152.777600
2015	Accessories	Bike Stands	2881.080000
		Bottles and Cages	4030.055400
		Helmets	9405.969000
		Tires and Tubes	3267.578700
	Bikes	Mountain Bikes	27178.015500
		Road Bikes	10455.862300
	Clothing	Jerseys	-1310.237900
2016	Accessories	Bike Racks	14834.160000
		Bike Stands	11531.316000
		Bottles and Cages	17554.898300
		Cleaners	2762.672700
		Fenders	17641.520500
		Helmets	74557.348600
		Hydration Packs	15706.418800
		Tires and Tubes	93141.340900
	Bikes	Mountain Bikes	1268675.729200
		Road Bikes	215322.321800
		Touring Bikes	82965.115800
	Clothing	Caps	-3165.019400
		Gloves	13751.490100
		Jerseys	-36205.408500
		Shorts	28021.556700
		Socks	2003.441200
		Vests	13895.324000

Major Profit is contributed by the Bike Category.

low profit contributing product

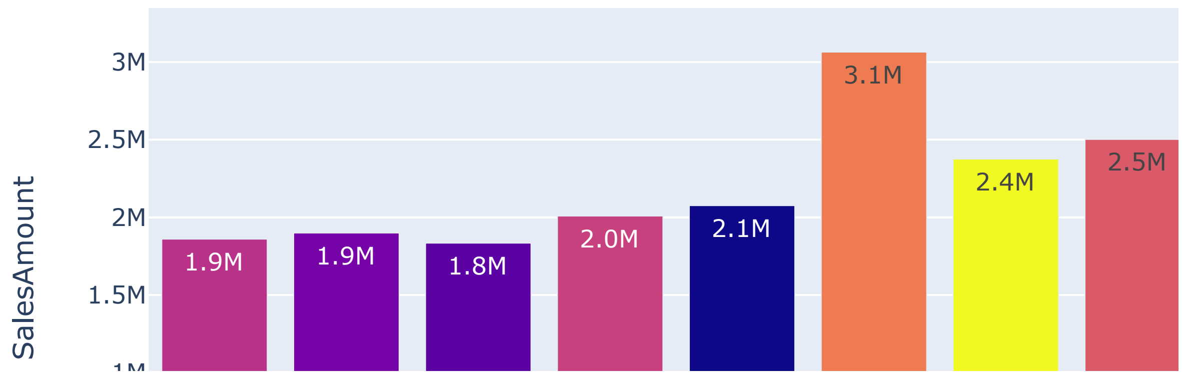
In [58]:
```
merged3.groupby(['Category', 'SubCategory','ProductName'])['Profit'].sum().ns
```

Out[58]:

Category	SubCategory	ProductName	Profit
Bikes	Touring Bikes	Touring-1000 Blue, 60	-30399.7503
	Road Bikes	Road-550-W Yellow, 38	-28835.9708
		Road-350-W Yellow, 44	-24454.7800
		Road-350-W Yellow, 40	-19973.0100
	Touring Bikes	Touring-1000 Blue, 46	-12227.4147
	Road Bikes	Road-250 Red, 52	-9167.4580
		Road-750 Black, 58	-7963.3208
	Touring Bikes	Touring-2000 Blue, 54	-6365.8200
	Road Bikes	Road-750 Black, 48	-5705.9452
Clothing	Jerseys	Short-Sleeve Classic Jersey, S	-5321.7943

Month-wise Sales Profit & Best months for sales

In [59]:
```
month_sales = merged1.groupby('month').sum()[['SalesAmount', 'Profit']]
month_sales.reset_index(inplace=True)
fig = px.bar(month_sales, x='month', y='SalesAmount',text_auto='.2s',
             hover_data=['month', 'SalesAmount'], color='Profit',
             height=400)
fig.show()
```



There are large profit transactions in the months of June, November, and December

On which day sale is maximum

Add day column in merged1

In [60]:
```
#Extracting day from OrderDate
merged1['Day']=pd.DatetimeIndex(merged1.OrderDate).day
merged1.head(5)
```

Out[60]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrd
0	310	2014-01-01	2014-01-08	21768	1	6	
1	600	2016-04-16	2016-04-23	21768	1	6	
2	346	2014-01-01	2014-01-08	28389	1	7	
3	346	2014-01-01	2014-01-08	25863	1	1	
4	578	2016-07-27	2016-08-03	25863	1	1	

5 rows × 34 columns

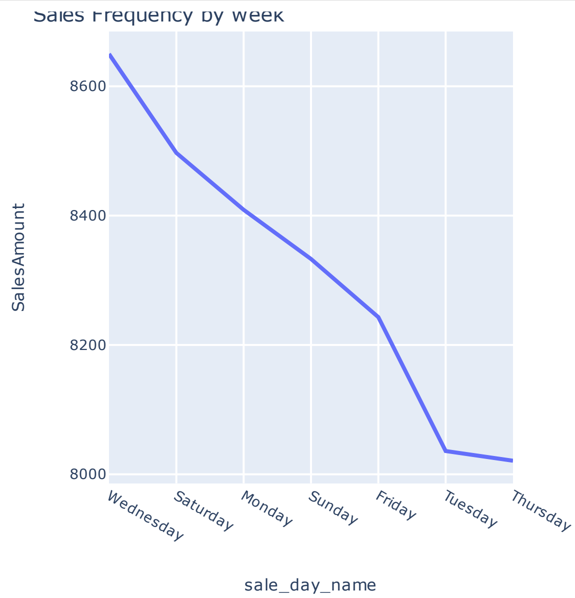
In [61]:
```
# Extracting day_name from OrderDate
merged1['sale_day_name'] = merged1['OrderDate'].dt.day_name()
merged1.head(5)
```

Out[61]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrd
0	310	2014-01-01	2014-01-08	21768	1	6	
1	600	2016-04-16	2016-04-23	21768	1	6	
2	346	2014-01-01	2014-01-08	28389	1	7	
3	346	2014-01-01	2014-01-08	25863	1	1	
4	578	2016-07-27	2016-08-03	25863	1	1	

5 rows × 35 columns

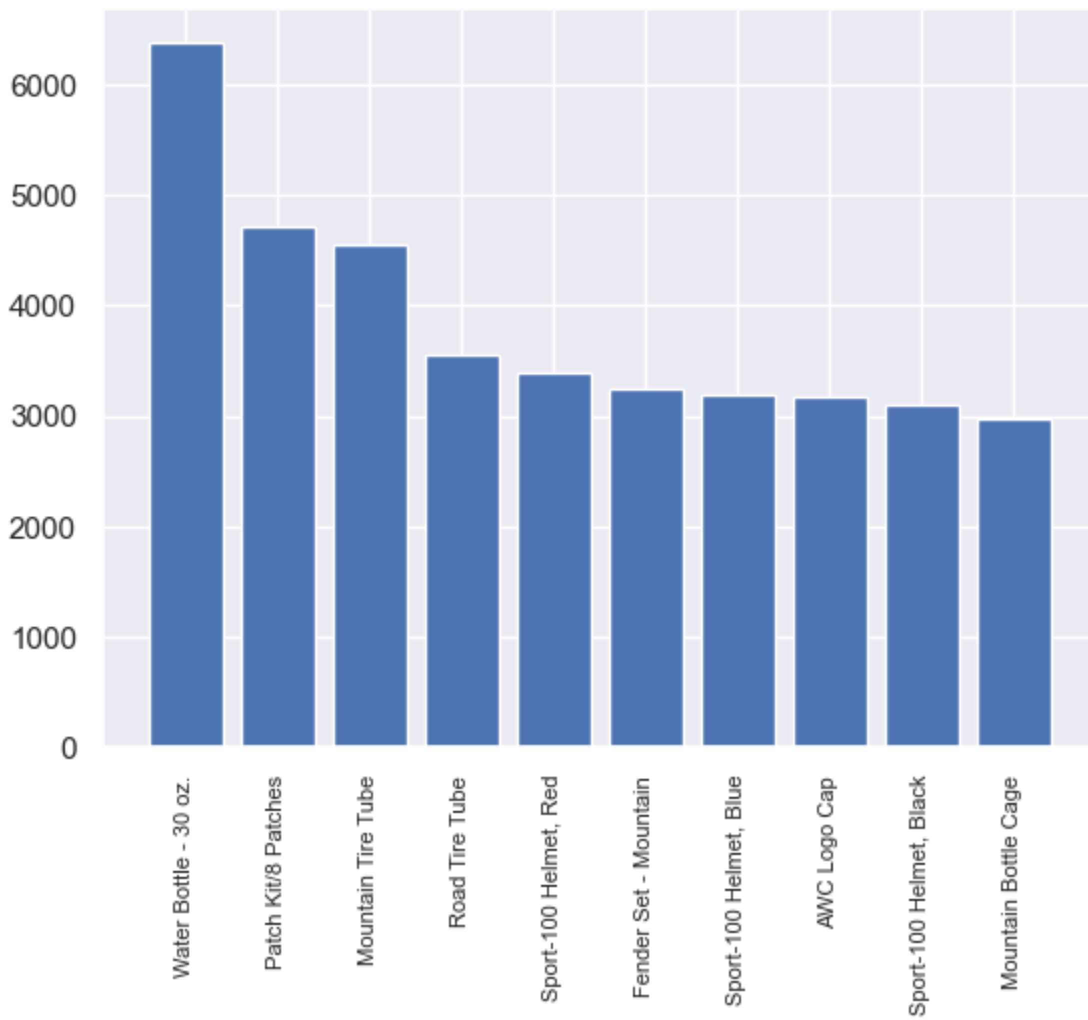
In [62]:
```
sales_by_week = merged1.groupby(['sale_day_name']).count()['SalesAmount'].rese

fig = px.line(sales_by_week, x='sale_day_name', y='SalesAmount', title='Sales
fig.update_layout(
    autosize=True,
    width=300,
    height=300,
    margin=dict(
        l=25,
        r=25,
        b=10,
        t=10,
    ),
    font=dict(size=7))
fig.show()
```



High sales orders are seen on Wednesday and Saturday, therefore we can promote our product during these workweek.

Which product sold the most?

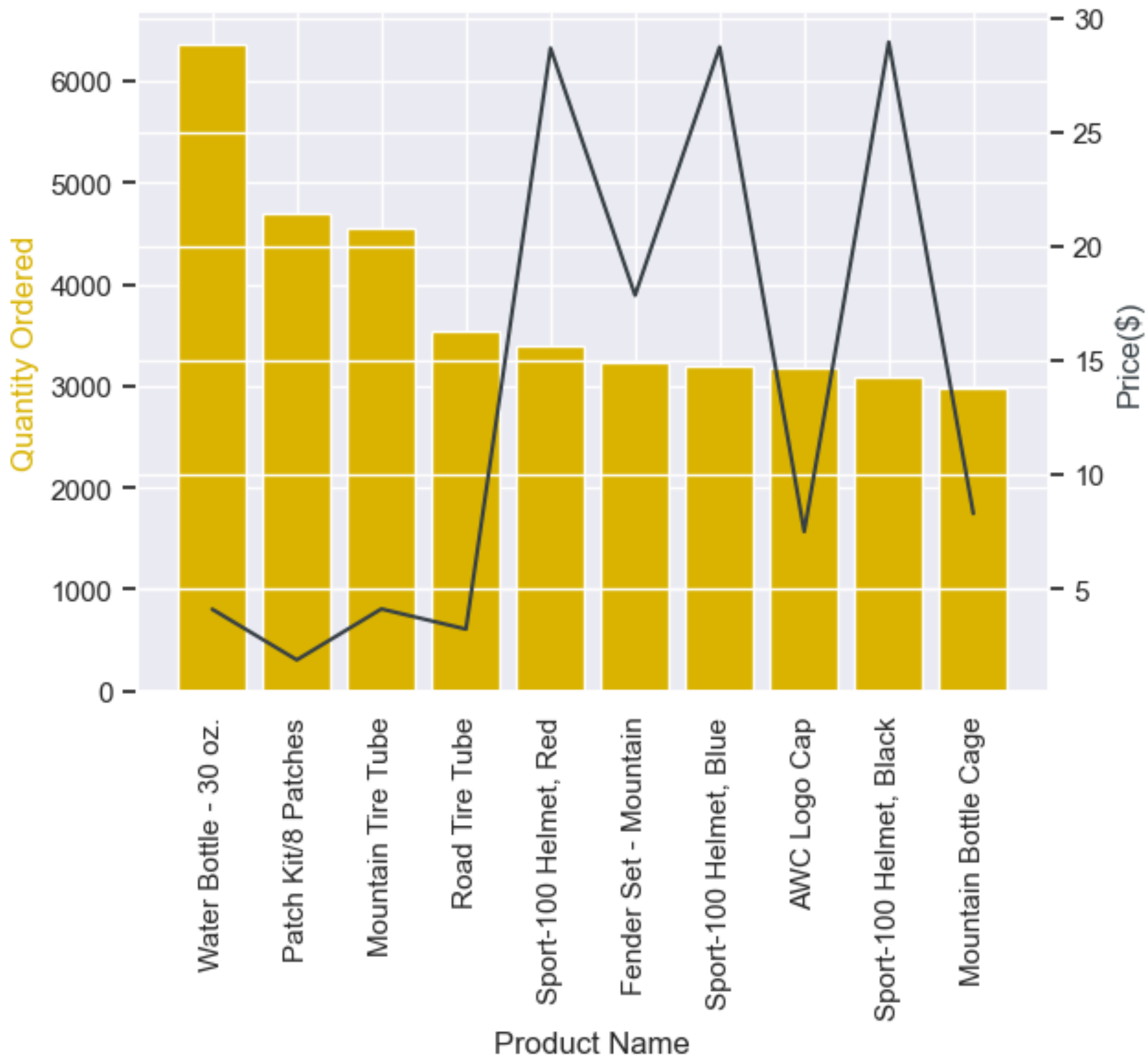
In [63]:
```
product_group = merged3.groupby('ProductName')
quantity_ordered = product_group['OrderQuantity'].sum().sort_values(ascending=
products = quantity_ordered.index.tolist()

plt.bar(products, quantity_ordered, )
plt.xticks(products, rotation='vertical', size=8)
plt.show()
```



why do you think it sold the most?

In [64]:
```
prices = merged3.groupby('ProductName').mean()['UnitPrice']
prices = prices[products]
```

In [65]:
```
fig, ax1 = plt.subplots()

ax2 = ax1.twinx()
ax1.bar(products, quantity_ordered, color='#D9B300')
ax2.plot(products, prices, '#374045')

ax1.set_xlabel('Product Name')
ax1.set_ylabel('Quantity Ordered', color='#D9B300')
ax2.set_ylabel('Price($)', color='#374045')
ax1.set_xticklabels(products, rotation='vertical')

plt.show();
```



In [66]:
```
prices.corr(quantity_ordered)
```

Out[66]:
```
-0.5333019792658484
```

There is a high negative correlation between Price and number of Quantity ordered

we can conclude that low price product has high demand.

Which products are most often sold together?

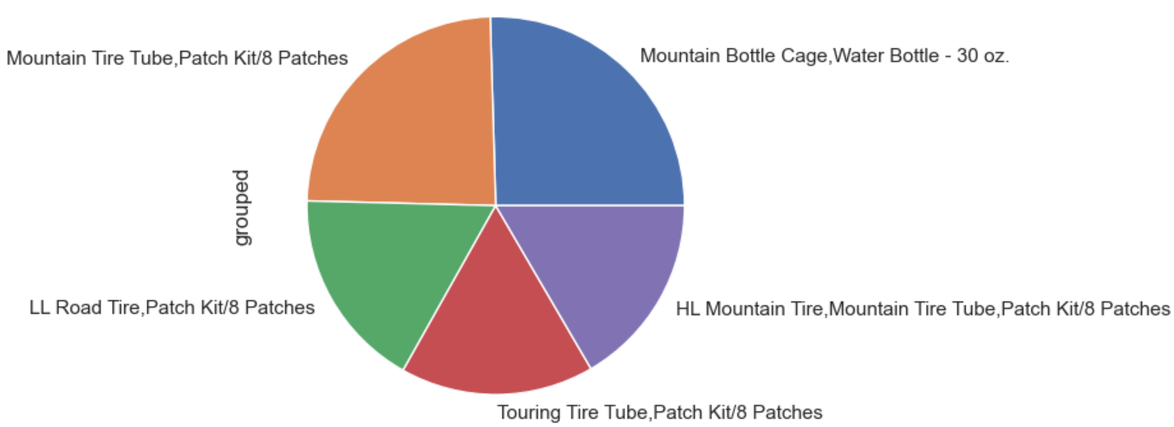
In [67]:
```
# By setting keep on False, all duplicates are True since we only want repeate
duplicate_order = merged3[merged3['SalesOrderNumber'].duplicated(keep=False)]
```

In [68]:
```
# Group the data based on sales order number and product name because the prod
# that bought together will have share same order number
duplicate_order['grouped'] = merged3.groupby('SalesOrderNumber')['ProductName'
duplicate_order = duplicate_order[['SalesOrderNumber', 'grouped']].drop_duplic
```

In [69]:
```
count = Counter()

for row in duplicate_order['grouped']:
    row_list  = row.split(',')
    count.update(Counter(combinations(row_list, 2)))

for key, value in count.most_common(10):
    print(key, value)
```

```
('Sport-100 Helmet', ' Red') 2092
('Sport-100 Helmet', ' Blue') 1981
('Sport-100 Helmet', ' Black') 1935
('Mountain Bottle Cage', 'Water Bottle - 30 oz.') 1623
('Road Bottle Cage', 'Water Bottle - 30 oz.') 1513
('HL Mountain Tire', 'Mountain Tire Tube') 915
('Sport-100 Helmet', 'Mountain Tire Tube') 831
('Touring Tire', 'Touring Tire Tube') 758
('Mountain Tire Tube', 'Patch Kit/8 Patches') 737
('Mountain Tire Tube', 'ML Mountain Tire') 727
```

In [70]:
```
count =duplicate_order['grouped'].value_counts()[0:5].plot.pie()
```



From the above pie diagram we can draw a conclusion that these products are mostly Purchased together.

Does Gender and home ownership matter in order purchasing

In [71]:
```
fig = px.imshow(merged1.groupby(["Gender", "HouseOwnerFlag"])["SalesAmount"].m
                labels=dict(color="Average Purchase"))
fig.show()
```

Compare most ordered product by gender

merged Customers and merged3(Sales & Product)

In [72]:
```
merged4=merged3.merge(Customers,left_on='CustomerKey',right_on='CustomerKey')
merged4.head(3)
```

Out[72]:

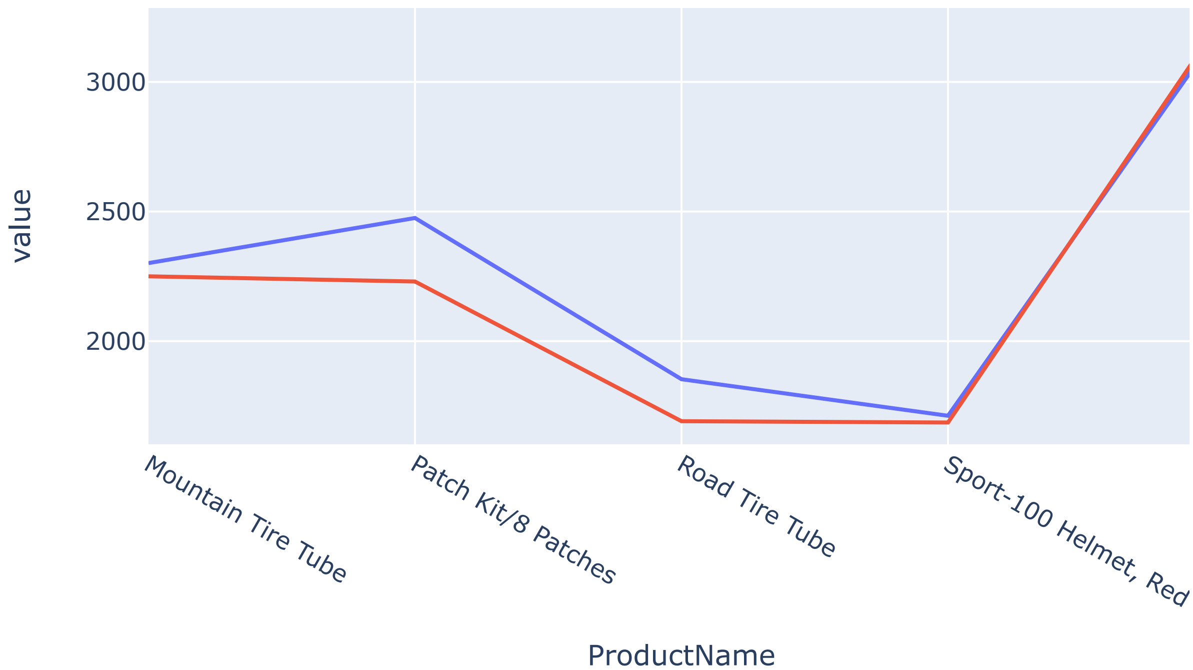
	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrd
0	310	2014-01-01	2014-01-08	21768	1	6	
1	600	2016-04-16	2016-04-23	21768	1	6	
2	310	2014-01-02	2014-01-09	16624	1	9	

3 rows × 43 columns

In [73]:
```
male = merged4[merged4["Gender"]=="M"]
female = merged4[merged4["Gender"]=="F"]
```

In [74]:
```
male_ord_qty = male.groupby(['ProductName'],as_index=False)['OrderQuantity'].s
male_ord_qty.columns=['ProductName','Order_Qty_Male']

female_ord_qty = female.groupby(['ProductName'],as_index=False)['OrderQuantity
female_ord_qty.columns=['ProductName','Order_Qty_Female']

Gender_merge = pd.merge(male_ord_qty, female_ord_qty, on='ProductName')
```

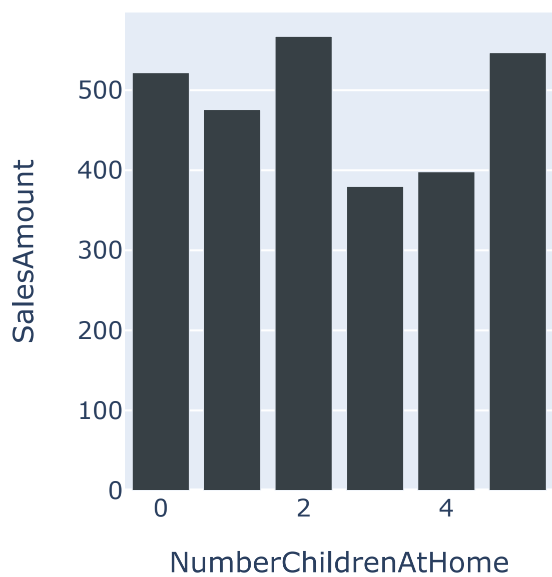
In [75]:
```
fig = px.line(Gender_merge, x="ProductName", y=["Order_Qty_Male","Order_Qty_Fe
fig.update_layout(
    autosize=True,
    width=800,
    height=400)
fig.show()
```



We see that the order quantity ordered by men and women from above diagram.

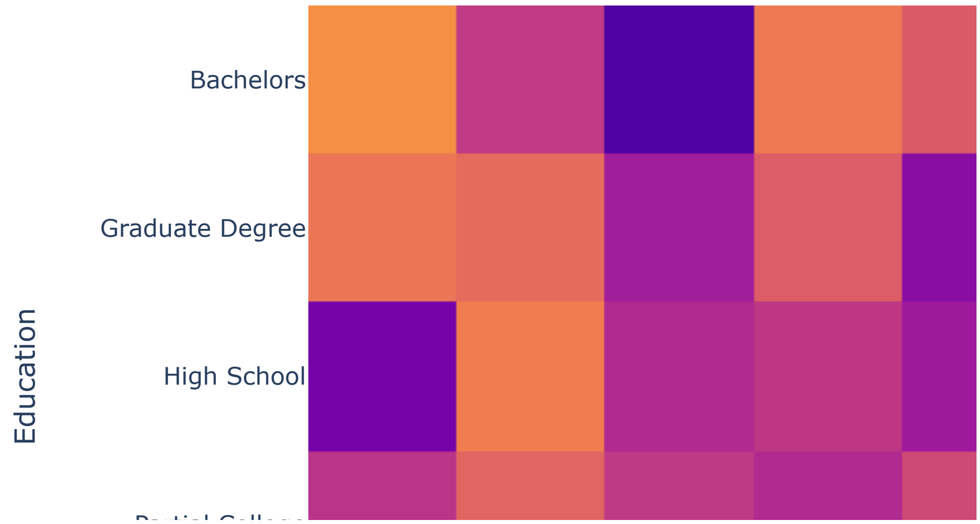
Number of children and Purchase correlation

In [76]:
```
Children= merged1.groupby(["NumberChildrenAtHome"])["SalesAmount"].mean().to_f
Children.reset_index(inplace=True)
fig = px.bar(Children, x='NumberChildrenAtHome', y='SalesAmount',color_discret
fig.update_layout(
    autosize=False,
    width=300,
    height=300,
    margin=dict(
        l=25,
        r=25,
        b=10,
        t=10,
    ))
fig.show()
```



Education, Occupation and Purchase correlation

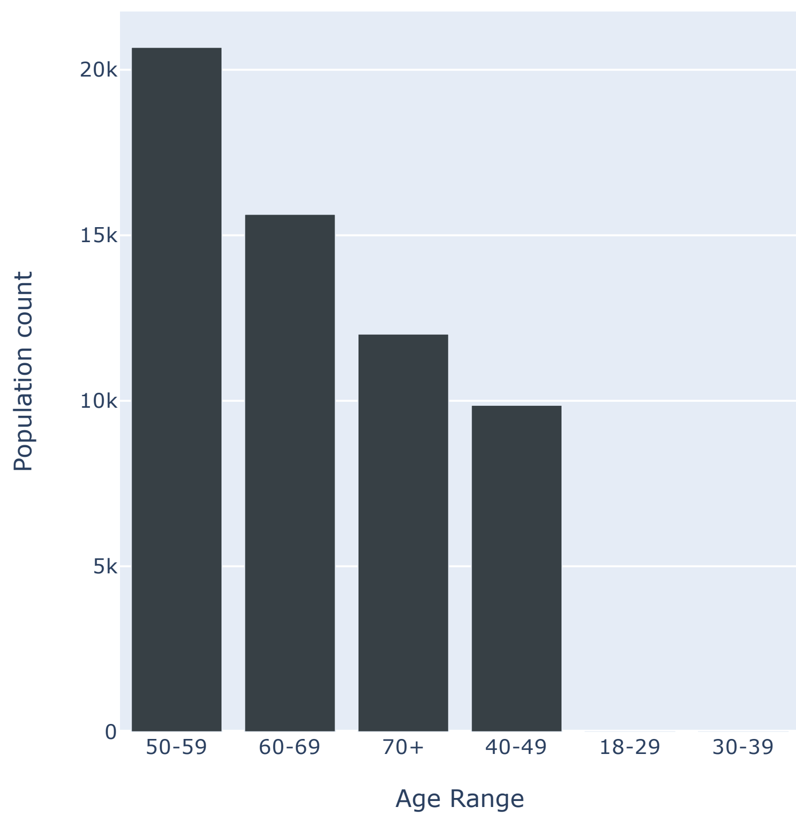
In [77]:
```
fig = px.imshow(merged1.groupby(["Education", "Occupation"])["SalesAmount"].me
                labels=dict(color="Average Purchase"))
fig.show()
```



Age-range Distribution

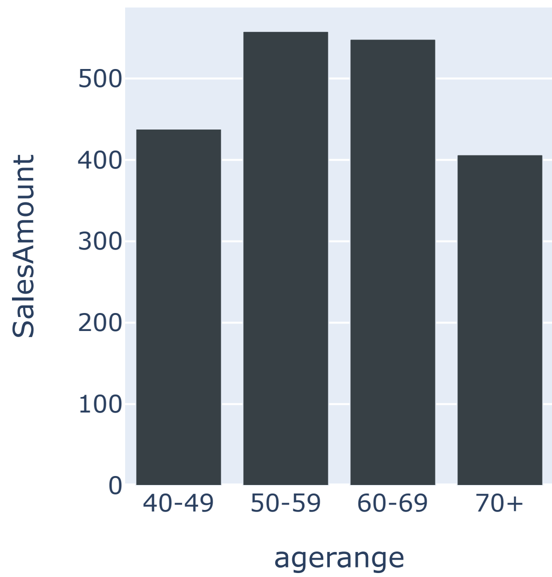
Find age-range

In [78]:
```
bins = [18, 30, 40, 50, 60, 70, 120]
labels = ['18-29', '30-39', '40-49', '50-59', '60-69', '70+']
merged1['agerange'] = pd.cut(merged1.age, bins, labels = labels,include_lowest

age_distribution = merged1['agerange'].value_counts().to_frame().reset_index()

age_distribution.columns = ['Age Range','Population count']

fig = px.bar(age_distribution, x='Age Range', y='Population count', color_disc
fig.update_layout(
    autosize=True,
    width=500,
    height=500,
    font=dict(size=10))
fig.show()
```



Which age group has produced the most revenue?

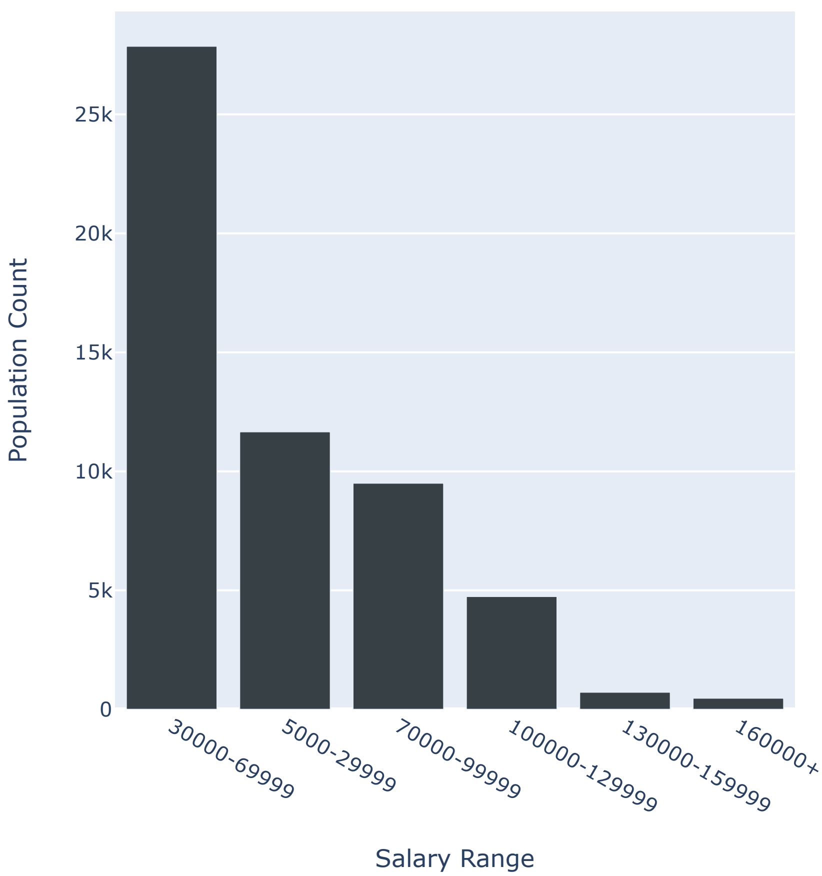
In [79]:
```
age_revenue = merged1.groupby('agerange')['SalesAmount'].mean().to_frame().dro
age_revenue.reset_index(inplace=True)
fig = px.bar(age_revenue, x='agerange', y='SalesAmount', color_discrete_sequen
fig.update_layout(
    autosize=False,
    width=300,
    height=300,
    margin=dict(
        l=25,
        r=25,
        b=10,
        t=10,
    ))
fig.show()
```



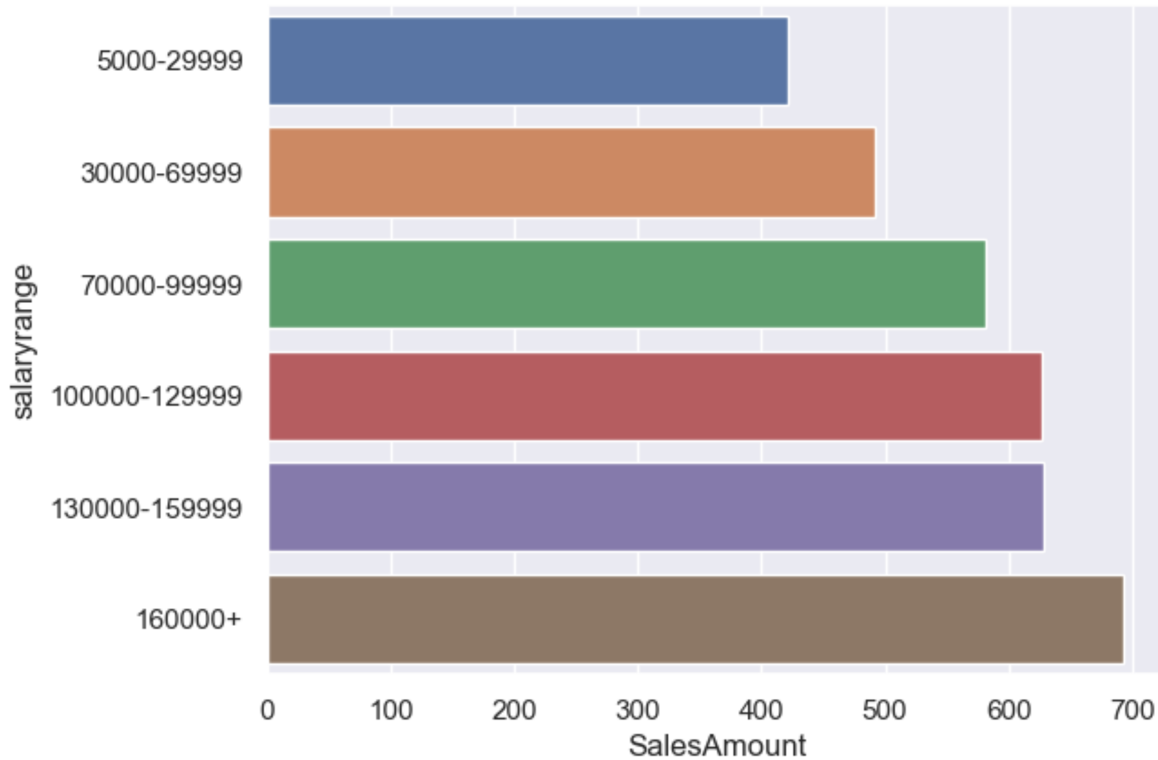
Yearly income range and purchase correlation

Find Salary Range

In [80]:
```
bins = [18000, 30000, 70000,100000, 130000, 160000, 200000]
labels = ['5000-29999', '30000-69999', '70000-99999', '100000-129999', '130000
merged1['salaryrange'] = pd.cut(merged1.YearlyIncome, bins, labels = labels,in

salary_distribution = merged1['salaryrange'].value_counts().to_frame().reset_i

salary_distribution.columns = ['Salary Range','Population Count']

fig = px.bar(salary_distribution, x='Salary Range', y='Population Count', colo
fig.update_layout(
    autosize=True,
    width=500,
    height=500,
    font=dict(size=10))
fig.show()
```



In [81]:
```
incomerange = merged1.groupby('salaryrange')['SalesAmount'].mean().to_frame()
incomerange.reset_index(inplace=True)
sns.barplot(x="SalesAmount", y="salaryrange", data=incomerange);
```



High salary range leads to increase in purchase.

In [ ]:

Insights

- Genderwise female do more shopping than male.

- Marital status wise married person spend more money on shopping.

- There are large profit transaction in months July and October.

- There are more profit transactions in Australia and Southwest.

- Maximum of the unit price is below dollar 1000.

- Most of the time 2-3 products are ordered in a single order.

- Maximum quantity ordered for a product is below 5.

- The year 2016 saw a sudden powerful movement in sales.

- High quantity of products is ordered from Australia and United States.

- Major Profit is contributed by the Bike Category.

- There are large profit transactions in the months of June, November, and December.

- High sales orders are seen on Wednesday and Saturday, therefore we can promote our product during these workweek.

- There is a high negative correlation between Price and number of Quantity ordered.

- we can conclude that low price product has high demand.

- From pie diagram we can draw a conclusion that these products are mostly Purchased together.

- We see that the order quantity ordered by men and women from line diagram.

In [ ]:

```

```