In [3]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
import plotly.express as px
import seaborn as sns
```

In [ ]:
```
```

In [4]:
```
df=pd.read_excel("Copy of Budget data1.xlsx")
```

In [5]:
```
df
```

Out[5]:

	Category	Subcategory	ProductName	ProductKey	Jan, 2016	Feb, 2016	Mar, 2016	Apr, 2016	May, 2016	Jun, 2016	Jul, 2016	Aug, 2016	Sep, 2016	Oct, 2016	
0	Accessories	Bike Racks	Hitch Rack - 4-Bike	483.0	1131	2635	4134	2179	2637	3279	2218	3287	3885	2484	
1	Accessories	Bike Stands	All-Purpose Bike Stand	486.0	666	3695	2868	4862	3439	4612	2774	3003	2401	4413	
2	Accessories	Bottles and Cages	Water Bottle - 30 oz.	477.0	1892	4727	3656	4449	4051	6257	4871	5231	5461	5529	
3	Accessories	Cleaners	Bike Wash - Dissolver	484.0	160	713	555	656	369	582	777	777	239	496	
4	Accessories	Fenders	Fender Set - Mountain	485.0	970	3014	2809	4259	3638	3721	4190	3618	3975	3892	
5	Accessories	Helmets	Sport-100 Helmet, Red	212.0	5317	16221	16752	16552	17204	25354	17584	20409	18268	20567	2
6	Accessories	Hydration Packs	Hydration Pack - 70 oz.	487.0	809	2684	2917	3425	2716	3260	3773	3523	4252	3111	
7	Accessories	Tires and Tubes	Patch Kit/8 Patches	480.0	3554	18758	20905	18046	21680	22456	23995	22922	20950	21905	2
8	SubTotal Accessories	NaN	NaN	NaN	14499	52447	54596	54428	55734	69521	60182	62770	59431	62397	7
9	Bikes	Mountain Bikes	Mountain-100 Silver, 38	344.0	370105	326786	384811	439822	458523	619456	524348	647048	557368	615032	80
10	Bikes	Road Bikes	Road-150 Red, 62	310.0	346295	289524	355097	346783	399691	546092	441037	432400	468572	483913	55
11	Bikes	Touring Bikes	Touring-2000 Blue, 60	560.0	133631	165941	178287	265901	286630	445270	299106	407069	391580	481316	50
12	SubTotal Bikes	NaN	NaN	NaN	850031	782251	918195	1052506	1144844	1610818	1264491	1486517	1417520	1580261	186
13	Clothing	Caps	AWC Logo Cap	223.0	479	1695	1462	1079	1729	2180	1588	2065	2013	2138	
14	Clothing	Gloves	Half-Finger Gloves, S	462.0	598	2474	2957	2705	2819	2966	2975	3264	2424	3181	

	Category	Subcategory	ProductName	ProductKey	Jan, 2016	Feb, 2016	Mar, 2016	Apr, 2016	May, 2016	Jun, 2016	Jul, 2016	Aug, 2016	Sep, 2016	Oct, 2016	
15	Clothing	Jerseys	Long-Sleeve Logo Jersey, S	226.0	4087	11508	12872	11809	12789	18153	16846	13497	15988	15920	1
16	Clothing	Shorts	Men's Sports Shorts, S	445.0	421	5723	7301	6335	5288	6829	4617	5384	6277	6337	
17	Clothing	Socks	Mountain Bike Socks, M	218.0	24	244	432	547	385	372	839	487	425	335	
18	Clothing	Vests	Classic Vest, S	471.0	980	2008	1980	2312	2763	2591	3379	3580	3600	4248	
19	SubTotal Clothing	NaN	NaN	NaN	6589	23652	27004	24787	25773	33091	30244	28277	30727	32159	3
20	Grand Total	NaN	NaN	NaN	871119	858350	999795	1131721	1226351	1713430	1354917	1577564	1507678	1674817	197

In [6]:
```
df.duplicated().sum()
```

Out[6]:
```
0
```

In [7]:
```
df1=pd.read_excel('Copy of AdventureWorks_Database.xlsx')
```

In [8]:
```
df1
```

Out[8]:

	Date	DateKey	Year	Quarter	MonthNum	Month	FiscalYear	FiscalQuarter	FiscalMonthNum	FiscalMonth	MonthYear	MonthYearLong	Mont
0	2016-04-03	20160403	2016	Q2	4	Apr	FY2016	FQ4	10	Apr	Apr-16	Apr-2016	
1	2016-04-04	20160404	2016	Q2	4	Apr	FY2016	FQ4	10	Apr	Apr-16	Apr-2016	
2	2016-04-05	20160405	2016	Q2	4	Apr	FY2016	FQ4	10	Apr	Apr-16	Apr-2016	
3	2016-04-06	20160406	2016	Q2	4	Apr	FY2016	FQ4	10	Apr	Apr-16	Apr-2016	
4	2016-04-07	20160407	2016	Q2	4	Apr	FY2016	FQ4	10	Apr	Apr-16	Apr-2016	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1456	2014-06-18	20140618	2014	Q2	6	Jun	FY2014	FQ4	12	Jun	Jun-14	Jun-2014	
1457	2014-06-19	20140619	2014	Q2	6	Jun	FY2014	FQ4	12	Jun	Jun-14	Jun-2014	
1458	2014-06-20	20140620	2014	Q2	6	Jun	FY2014	FQ4	12	Jun	Jun-14	Jun-2014	
1459	2014-06-21	20140621	2014	Q2	6	Jun	FY2014	FQ4	12	Jun	Jun-14	Jun-2014	
1460	2014-06-22	20140622	2014	Q2	6	Jun	FY2014	FQ4	12	Jun	Jun-14	Jun-2014	

1461 rows × 16 columns

In [9]:
```
Customer_data=pd.read_excel('Copy of AdventureWorks_Database.xlsx','Customers')
```

In [10]:
```
Customer_data
```

Out[10]:

	CustomerKey	FirstName	LastName	FullName	BirthDate	MaritalStatus	Gender	YearlyIncome	TotalChildren	NumberChildrenAtHome	Educatio
0	11000	Jon	Yang	Yang, Jon	1966-04-08	M	M	90000	2	0	Bachelor
1	11001	Eugene	Huang	Huang, Eugene	1965-05-14	S	M	60000	3	3	Bachelor
2	11002	Ruben	Torres	Torres, Ruben	1965-08-12	M	M	60000	3	3	Bachelor
3	11003	Christy	Zhu	Zhu, Christy	1968-02-15	S	F	70000	0	0	Bachelor
4	11004	Elizabeth	Johnson	Johnson, Elizabeth	1968-08-08	S	F	80000	5	5	Bachelor
...	...	...	...	...	...	...	...	...	...	...	
18479	29479	Tommy	Tang	Tang, Tommy	1958-07-04	M	M	30000	1	0	Graduat Degre
18480	29480	Nina	Raji	Raji, Nina	1960-11-10	S	F	30000	3	0	Graduat Degre
18481	29481	Ivan	Suri	Suri, Ivan	1960-01-05	S	M	30000	3	0	Graduat Degre
18482	29482	Clayton	Zhang	Zhang, Clayton	1959-03-05	M	M	30000	3	0	Bachelor
18483	29483	Jésus	Navarro	Navarro, Jésus	1959-12-08	M	M	30000	0	0	Bachelor

18484 rows × 17 columns

In [11]:
```
Product_data=pd.read_excel('Copy of AdventureWorks_Database.xlsx','Product')
```

In [12]:
```
Product_data
```

Out[12]:

	ProductKey	ProductName	SubCategory	Category	StandardCost	Color	ListPrice	DaysToManufacture	ProductLine	ModelName	
0	1	Adjustable Race	NaN	NaN	NaN	NaN	NaN	0	NaN	NaN	http://www.avis
1	2	Bearing Ball	NaN	NaN	NaN	NaN	NaN	0	NaN	NaN	http://www.avis
2	3	BB Ball Bearing	NaN	NaN	NaN	NaN	NaN	1	NaN	NaN	http://www.avis
3	4	Headset Ball Bearings	NaN	NaN	NaN	NaN	NaN	0	NaN	NaN	http://www.avis
4	5	Blade	NaN	NaN	NaN	NaN	NaN	1	NaN	NaN	http://www.avis
...	...	...	...	...	...	...	...	...	...	...	
601	602	ML Bottom Bracket	Bottom Brackets	Components	44.9506	NaN	101.24	1	NaN	ML Bottom Bracket	http://www.avis
602	603	HL Bottom Bracket	Bottom Brackets	Components	53.9416	NaN	121.49	1	NaN	HL Bottom Bracket	http://www.avis
603	604	Road-750 Black, 44	Road Bikes	Bikes	343.6496	Black	539.99	4	Road	Road-750	http://www.avis
604	605	Road-750 Black, 48	Road Bikes	Bikes	343.6496	Black	539.99	4	Road	Road-750	http://www.avis
605	606	Road-750 Black, 52	Road Bikes	Bikes	343.6496	Black	539.99	4	Road	Road-750	http://www.avis

606 rows × 13 columns

In [13]:
```
Territory_data=pd.read_excel('Copy of AdventureWorks_Database.xlsx','Territory')
```

In [14]:
```
Territory_data
```

Out[14]:

	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...
3	4	Southwest	United States	North America	http://www.avising.com/me/LearnPBI/DataSources...
4	5	Southeast	United States	North America	http://www.avising.com/me/LearnPBI/DataSources...
5	6	Canada	Canada	North America	http://www.avising.com/me/LearnPBI/DataSources...
6	7	France	France	Europe	http://www.avising.com/me/LearnPBI/DataSources...
7	8	Germany	Germany	Europe	http://www.avising.com/me/LearnPBI/DataSources...
8	9	Australia	Australia	Pacific	http://www.avising.com/me/LearnPBI/DataSources...
9	10	United Kingdom	United Kingdom	Europe	http://www.avising.com/me/LearnPBI/DataSources...
10	11	NaN	NaN	NaN	http://www.avising.com/me/LearnPBI/DataSources...

In [15]:
```
Sales_data=pd.read_excel('Copy of AdventureWorks_Database.xlsx','Sales')
```

In [16]:
```
Sales_data
```

Out[16]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrderNumber	SalesOrderLineNumber	OrderQuantity	Un
0	310	2014-01-01	2014-01-08	21768	1	6	SO43697	1	2	178
1	346	2014-01-01	2014-01-08	28389	1	7	SO43698	1	2	169
2	346	2014-01-01	2014-01-08	25863	1	1	SO43699	1	2	169
3	336	2014-01-01	2014-01-08	14501	1	4	SO43700	1	2	34
4	346	2014-01-01	2014-01-08	11003	1	9	SO43701	1	2	169
...	...	...	...	...	...	...	...	...	...	
58184	561	2016-12-30	2017-01-07	13650	1	9	SO74145	1	1	238
58185	584	2016-12-30	2017-01-07	26916	1	9	SO74146	1	1	53
58186	605	2016-12-30	2017-01-07	27473	1	9	SO74147	1	1	53
58187	538	2016-12-30	2017-01-07	27473	1	9	SO74147	2	1	2
58188	490	2016-12-30	2017-01-07	27473	1	9	SO74147	3	1	5

58189 rows × 25 columns

In [17]:
```
# Merging Data
temp_data = pd.merge(Sales_data, Product_data, on='ProductKey', how='inner')
```

In [18]:
```
temp_data
```

Out[18]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrderNumber	SalesOrderLineNumber	OrderQuantity	Un
0	310	2014-01-01	2014-01-08	21768	1	6	SO43697	1	2	178
1	310	2014-01-02	2014-01-09	16624	1	9	SO43703	1	4	89
2	310	2014-01-05	2014-01-12	27601	1	4	SO43713	1	1	357
3	310	2014-01-06	2014-01-13	13590	1	10	SO43721	1	1	357
4	310	2014-01-10	2014-01-17	16522	1	9	SO43735	1	1	357
...	...	...	...	...	...	...	...	...	...	...
58184	567	2016-12-13	2016-12-20	15662	1	10	SO72823	1	2	37
58185	567	2016-12-14	2016-12-21	26646	1	4	SO72925	1	4	18
58186	567	2016-12-20	2016-12-27	11363	1	9	SO73411	1	2	37
58187	567	2016-12-22	2016-12-29	11944	1	9	SO73577	1	1	74
58188	567	2016-12-25	2017-01-02	11261	1	4	SO73768	1	1	74

58189 rows × 37 columns

In [19]:
```
df2 = pd.merge(temp_data, Customer_data, on='CustomerKey', how='inner')
df2 = pd.merge(df2, Territory_data, on='SalesTerritoryKey', how='inner')
```

In [20]:
```
df2
```

Out[20]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrderNumber	SalesOrderLineNumber	OrderQuantity	Un
0	310	2014-01-01	2014-01-08	21768	1	6	SO43697	1	2	178
1	600	2016-04-16	2016-04-23	21768	1	6	SO56212	1	1	53
2	310	2014-01-30	2014-02-06	21727	1	6	SO43833	1	4	89
3	479	2016-11-29	2016-12-05	21727	1	6	SO71614	2	1	
4	477	2016-11-29	2016-12-05	21727	1	6	SO71614	3	1	
...	...	...	...	...	...	...	...	...	...	
58184	528	2016-11-07	2016-11-14	13145	1	2	SO70064	2	1	
58185	361	2016-11-07	2016-11-14	13145	1	2	SO70064	1	1	229
58186	480	2016-11-07	2016-11-14	13145	1	2	SO70064	4	1	
58187	530	2016-02-06	2016-02-13	27040	1	2	SO52124	1	1	
58188	480	2016-02-06	2016-02-13	27040	2	2	SO52124	2	1	

58189 rows × 57 columns

In [25]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58189 entries, 0 to 58188
Data columns (total 57 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_x       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
 25  ProductName          58189 non-null  object
 26  SubCategory          58189 non-null  object
 27  Category             58189 non-null  object
 28  StandardCost_y       58189 non-null  float64
 29  Color                30747 non-null  object
 30  ListPrice            58189 non-null  float64
 31  DaysToManufacture    58189 non-null  int64
 32  ProductLine          58189 non-null  object
 33  ModelName            58189 non-null  object
 34  Photo                58189 non-null  object
 35  ProductDescription   58189 non-null  object
 36  StartDate            58189 non-null  datetime64[ns]
 37  FirstName            58189 non-null  object
 38  LastName             58189 non-null  object
 39  FullName              58189 non-null  object
 40  BirthDate             58189 non-null  datetime64[ns]
 41  MaritalStatus         58189 non-null  object
 42  Gender                58189 non-null  object
 43  YearlyIncome          58189 non-null  int64
 44  TotalChildren         58189 non-null  int64
 45  NumberChildrenAtHome  58189 non-null  int64
 46  Education             58189 non-null  object
 47  Occupation            58189 non-null  object
 48  HouseOwnerFlag        58189 non-null  int64
 49  NumberCarsOwned       58189 non-null  int64
 50  AddressLine1          58189 non-null  object
 51  DateFirstPurchase     58189 non-null  datetime64[ns]
 52  CommuteDistance       58189 non-null  object
 53  Region                58189 non-null  object
 54  Country               58189 non-null  object
 55  Group                 58189 non-null  object
 56  RegionImage           58189 non-null  object
dtypes: datetime64[ns](5), float64(16), int64(14), object(22)
memory usage: 25.3+ MB
```

In [26]:
```
df2.duplicated().sum()
```

Out[26]:
```
0
```

In [27]:
```
df3 = pd.merge( df,df2,how='left')
```

In [28]:
```
df3
```

Out[28]:

	Category	Subcategory	ProductName	ProductKey	Jan, 2016	Feb, 2016	Mar, 2016	Apr, 2016	May, 2016	Jun, 2016	...	Occupation	HouseOwnerFlag	Nu
0	Accessories	Bike Racks	Hitch Rack - 4-Bike	483.0	1131	2635	4134	2179	2637	3279	...	Skilled Manual	1.0	
1	Accessories	Bike Racks	Hitch Rack - 4-Bike	483.0	1131	2635	4134	2179	2637	3279	...	Skilled Manual	1.0	
2	Accessories	Bike Racks	Hitch Rack - 4-Bike	483.0	1131	2635	4134	2179	2637	3279	...	Clerical	1.0	
3	Accessories	Bike Racks	Hitch Rack - 4-Bike	483.0	1131	2635	4134	2179	2637	3279	...	Professional	1.0	
4	Accessories	Bike Racks	Hitch Rack - 4-Bike	483.0	1131	2635	4134	2179	2637	3279	...	Skilled Manual	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
11854	Clothing	Vests	Classic Vest, S	471.0	980	2008	1980	2312	2763	2591	...	Manual	0.0	
11855	Clothing	Vests	Classic Vest, S	471.0	980	2008	1980	2312	2763	2591	...	Clerical	1.0	
11856	Clothing	Vests	Classic Vest, S	471.0	980	2008	1980	2312	2763	2591	...	Manual	1.0	
11857	SubTotal Clothing	NaN	NaN	NaN	6589	23652	27004	24787	25773	33091	...	NaN	NaN	
11858	Grand Total	NaN	NaN	NaN	871119	858350	999795	1131721	1226351	1713430	...	NaN	NaN	

11859 rows × 71 columns

In [29]:
```
df2
```

Out[29]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrderNumber	SalesOrderLineNumber	OrderQuantity	Un
0	310	2014-01-01	2014-01-08	21768	1	6	SO43697	1	2	178
1	600	2016-04-16	2016-04-23	21768	1	6	SO56212	1	1	53
2	310	2014-01-30	2014-02-06	21727	1	6	SO43833	1	4	89
3	479	2016-11-29	2016-12-05	21727	1	6	SO71614	2	1	
4	477	2016-11-29	2016-12-05	21727	1	6	SO71614	3	1	
...	...	...	...	...	...	...	...	...	...	
58184	528	2016-11-07	2016-11-14	13145	1	2	SO70064	2	1	
58185	361	2016-11-07	2016-11-14	13145	1	2	SO70064	1	1	229
58186	480	2016-11-07	2016-11-14	13145	1	2	SO70064	4	1	
58187	530	2016-02-06	2016-02-13	27040	1	2	SO52124	1	1	
58188	480	2016-02-06	2016-02-13	27040	2	2	SO52124	2	1	

58189 rows × 57 columns

In [30]:
```
# Assessing Data
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58189 entries, 0 to 58188
Data columns (total 57 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_x       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
 25  ProductName          58189 non-null  object
 26  SubCategory          58189 non-null  object
 27  Category             58189 non-null  object
 28  StandardCost_y       58189 non-null  float64
 29  Color                30747 non-null  object
 30  ListPrice            58189 non-null  float64
 31  DaysToManufacture    58189 non-null  int64
 32  ProductLine          58189 non-null  object
 33  ModelName            58189 non-null  object
 34  Photo                58189 non-null  object
 35  ProductDescription   58189 non-null  object
 36  StartDate            58189 non-null  datetime64[ns]
 37  FirstName            58189 non-null  object
 38  LastName             58189 non-null  object
 39  FullName             58189 non-null  object
 40  BirthDate            58189 non-null  datetime64[ns]
 41  MaritalStatus        58189 non-null  object
 42  Gender               58189 non-null  object
 43  YearlyIncome         58189 non-null  int64
 44  TotalChildren        58189 non-null  int64
 45  NumberChildrenAtHome 58189 non-null  int64
 46  Education            58189 non-null  object
 47  Occupation           58189 non-null  object
 48  HouseOwnerFlag       58189 non-null  int64
 49  NumberCarsOwned      58189 non-null  int64
 50  AddressLine1         58189 non-null  object
 51  DateFirstPurchase    58189 non-null  datetime64[ns]
 52  CommuteDistance      58189 non-null  object
 53  Region               58189 non-null  object
 54  Country              58189 non-null  object
 55  Group                58189 non-null  object
 56  RegionImage          58189 non-null  object
dtypes: datetime64[ns](5), float64(16), int64(14), object(22)
memory usage: 25.3+ MB
```

In [31]:

```python
# Check shape of the data after merging
print(f"Number of Rows: {df2.shape[0]}")
print(f"Number of Columns: {df2.shape[1]} \n")
```

```
Number of Rows: 58189
Number of Columns: 57
```

In [32]:

```python
df2.describe().transpose()
```

Out[32]:

	count	mean	min	25%	50%	75%	max	std
ProductKey	58189.0	437.208304	214.0	358.0	479.0	529.0	606.0	118.099746
OrderDate	58189	2016-06-03 03:56:09.605939200	2014-01-01 00:00:00	2016-04-01 00:00:00	2016-07-07 00:00:00	2016-10-10 00:00:00	2016-12-30 00:00:00	NaN
ShipDate	58189	2016-06-10 04:03:24.657237760	2014-01-08 00:00:00	2016-04-08 00:00:00	2016-07-14 00:00:00	2016-10-17 00:00:00	2017-01-07 00:00:00	NaN
CustomerKey	58189.0	18853.00464	11000.0	14012.0	18151.0	23450.0	29483.0	5433.374315
PromotionKey	58189.0	1.043427	1.0	1.0	1.0	1.0	14.0	0.348948
SalesTerritoryKey	58189.0	6.261716	1.0	4.0	7.0	9.0	10.0	2.960248
SalesOrderLineNumber	58189.0	1.887453	1.0	1.0	2.0	2.0	8.0	1.018829
OrderQuantity	58189.0	1.569386	1.0	1.0	1.0	2.0	4.0	1.047532
UnitPrice	58189.0	413.888218	0.5725	4.99	24.49	269.995	3578.27	833.052938
TotalProductCost	58189.0	296.539185	0.8565	3.3623	12.1924	343.6496	2171.2942	560.171436
SalesAmount	58189.0	503.66627	2.29	8.99	32.6	539.99	3578.27	941.462817
TaxAmt	58189.0	40.293303	0.1832	0.7192	2.608	43.1992	286.2616	75.317027
Unnamed: 13	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Unnamed: 14	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Unnamed: 15	58189.0	503.666269	2.29	8.99	32.6	539.99	3578.27	941.462815
Unnamed: 16	58189.0	0.000001	0.0	0.0	0.0	0.0	0.0003	0.000014
Unnamed: 17	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Unnamed: 18	58189.0	38.398254	-5106.9068	1.4335	6.2537	21.9037	1487.8356	667.349417
Unnamed: 19	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
StandardCost_x	58189.0	296.539185	0.8565	3.3623	12.1924	343.6496	2171.2942	560.171436
List Price	58189.0	503.66627	2.29	8.99	32.6	539.99	3578.27	941.462817
Unnamed: 22	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
diif std cost	58189.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	count	mean	min	25%	50%	75%	max	std
diff list price	58189.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
StandardCost_y	58189.0	296.539185	0.8565	3.3623	12.1924	343.6496	2171.2942	560.171436
ListPrice	58189.0	503.66627	2.29	8.99	32.6	539.99	3578.27	941.462817
DaysToManufacture	58189.0	1.045215	0.0	0.0	0.0	4.0	4.0	1.757395
StartDate	58189	2007-05-14 02:44:51.848974848	2005-07-01 00:00:00	2007-07-01 00:00:00	2007-07-01 00:00:00	2007-07-01 00:00:00	2007-07-01 00:00:00	NaN
BirthDate	58189	1962-03-02 12:33:19.305710720	1910-08-13 00:00:00	1954-12-20 00:00:00	1963-09-19 00:00:00	1970-07-08 00:00:00	1980-12-26 00:00:00	NaN
YearlyIncome	58189.0	59769.887779	10000.0	30000.0	60000.0	80000.0	170000.0	33128.041818
TotalChildren	58189.0	1.838921	0.0	0.0	2.0	3.0	5.0	1.614467
NumberChildrenAtHome	58189.0	1.073502	0.0	0.0	0.0	2.0	5.0	1.580055
HouseOwnerFlag	58189.0	0.69056	0.0	0.0	1.0	1.0	1.0	0.462267
NumberCarsOwned	58189.0	1.502466	0.0	1.0	2.0	2.0	4.0	1.155496
DateFirstPurchase	58189	2015-12-23 02:50:33.356820224	2014-01-01 00:00:00	2015-06-21 00:00:00	2016-03-12 00:00:00	2016-07-26 00:00:00	2016-12-30 00:00:00	NaN

In [33]:
```
# Check for duplicate data
df2.duplicated().sum()
```

Out[33]:
```
0
```

In [34]:
```
df2.columns
```

```
Out[34]: Index(['ProductKey', 'OrderDate', 'ShipDate', 'CustomerKey', 'PromotionKey',
                'SalesTerritoryKey', 'SalesOrderNumber', 'SalesOrderLineNumber',
                'OrderQuantity', 'UnitPrice', 'TotalProductCost', 'SalesAmount',
                'TaxAmt', 'Unnamed: 13', 'Unnamed: 14', 'Unnamed: 15', 'Unnamed: 16',
                'Unnamed: 17', 'Unnamed: 18', 'Unnamed: 19', 'StandardCost_x',
                'List Price', 'Unnamed: 22', 'diif std cost', 'diff list price',
                'ProductName', 'SubCategory', 'Category', 'StandardCost_y', 'Color',
                'ListPrice', 'DaysToManufacture', 'ProductLine', 'ModelName', 'Photo',
                'ProductDescription', 'StartDate', 'FirstName', 'LastName', 'FullName',
                'BirthDate', 'MaritalStatus', 'Gender', 'YearlyIncome', 'TotalChildren',
                'NumberChildrenAtHome', 'Education', 'Occupation', 'HouseOwnerFlag',
                'NumberCarsOwned', 'AddressLine1', 'DateFirstPurchase',
                'CommuteDistance', 'Region', 'Country', 'Group', 'RegionImage'],
               dtype='object')
```

```
In [35]: df2.drop('Unnamed: 13',axis=1,inplace=True)
```

```
In [36]: df2.columns
```

```
Out[36]: Index(['ProductKey', 'OrderDate', 'ShipDate', 'CustomerKey', 'PromotionKey',
                'SalesTerritoryKey', 'SalesOrderNumber', 'SalesOrderLineNumber',
                'OrderQuantity', 'UnitPrice', 'TotalProductCost', 'SalesAmount',
                'TaxAmt', 'Unnamed: 14', 'Unnamed: 15', 'Unnamed: 16', 'Unnamed: 17',
                'Unnamed: 18', 'Unnamed: 19', 'StandardCost_x', 'List Price',
                'Unnamed: 22', 'diif std cost', 'diff list price', 'ProductName',
                'SubCategory', 'Category', 'StandardCost_y', 'Color', 'ListPrice',
                'DaysToManufacture', 'ProductLine', 'ModelName', 'Photo',
                'ProductDescription', 'StartDate', 'FirstName', 'LastName', 'FullName',
                'BirthDate', 'MaritalStatus', 'Gender', 'YearlyIncome', 'TotalChildren',
                'NumberChildrenAtHome', 'Education', 'Occupation', 'HouseOwnerFlag',
                'NumberCarsOwned', 'AddressLine1', 'DateFirstPurchase',
                'CommuteDistance', 'Region', 'Country', 'Group', 'RegionImage'],
               dtype='object')
```

```
In [37]: df2.drop(df2.columns[[13,14,15,16,17,18,21]],axis = 1,inplace=True)
```

```
In [38]: df2
```

Out[38]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrderNumber	SalesOrderLineNumber	OrderQuantity	Un
0	310	2014-01-01	2014-01-08	21768	1	6	SO43697	1	2	17
1	600	2016-04-16	2016-04-23	21768	1	6	SO56212	1	1	5
2	310	2014-01-30	2014-02-06	21727	1	6	SO43833	1	4	8
3	479	2016-11-29	2016-12-05	21727	1	6	SO71614	2	1	
4	477	2016-11-29	2016-12-05	21727	1	6	SO71614	3	1	
...	...	...	...	...	...	...	...	...	...	
58184	528	2016-11-07	2016-11-14	13145	1	2	SO70064	2	1	
58185	361	2016-11-07	2016-11-14	13145	1	2	SO70064	1	1	22
58186	480	2016-11-07	2016-11-14	13145	1	2	SO70064	4	1	
58187	530	2016-02-06	2016-02-13	27040	1	2	SO52124	1	1	
58188	480	2016-02-06	2016-02-13	27040	2	2	SO52124	2	1	

58189 rows × 49 columns

In [39]:
```python
# Handling missing data
df2.isnull().sum()
```

Out[39]:
```
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
StandardCost_x             0
List Price                 0
diif std cost              0
diff list price            0
ProductName                0
SubCategory                0
Category                   0
StandardCost_y             0
Color                  27442
ListPrice                  0
DaysToManufacture          0
ProductLine                0
ModelName                  0
Photo                      0
ProductDescription         0
StartDate                  0
FirstName                  0
LastName                   0
FullName                   0
BirthDate                  0
MaritalStatus              0
Gender                     0
YearlyIncome               0
TotalChildren              0
NumberChildrenAtHome       0
Education                  0
Occupation                 0
HouseOwnerFlag             0
NumberCarsOwned            0
AddressLine1               0
DateFirstPurchase          0
```

```
CommuteDistance             0
Region                      0
Country                     0
Group                       0
RegionImage                 0
dtype: int64
```

In [40]:
```
df2.dtypes
```

Out[40]:
```
ProductKey                       int64
OrderDate               datetime64[ns]
ShipDate                datetime64[ns]
CustomerKey                      int64
PromotionKey                     int64
SalesTerritoryKey                int64
SalesOrderNumber                object
SalesOrderLineNumber             int64
OrderQuantity                    int64
UnitPrice                      float64
TotalProductCost               float64
SalesAmount                    float64
TaxAmt                         float64
StandardCost_x                 float64
List Price                     float64
diif std cost                    int64
diff list price                  int64
ProductName                     object
SubCategory                     object
Category                        object
StandardCost_y                 float64
Color                           object
ListPrice                      float64
DaysToManufacture                int64
ProductLine                     object
ModelName                       object
Photo                           object
ProductDescription              object
StartDate               datetime64[ns]
FirstName                       object
LastName                        object
FullName                        object
BirthDate               datetime64[ns]
MaritalStatus                   object
Gender                          object
YearlyIncome                     int64
TotalChildren                    int64
NumberChildrenAtHome             int64
Education                       object
Occupation                      object
HouseOwnerFlag                   int64
NumberCarsOwned                  int64
AddressLine1                    object
DateFirstPurchase       datetime64[ns]
```

```
CommuteDistance                 object
Region                          object
Country                         object
Group                           object
RegionImage                     object
dtype: object
```

In [41]:
```
def missing_pct(df2):
    # Calculate missing value and their percentage for each column
    missing_count_percent = df2.isnull().sum() * 100 / df2.shape[0]
    df2_missing_count_percent = pd.DataFrame(missing_count_percent).round(2)
    df2_missing_count_percent = df2_missing_count_percent.reset_index().rename(
                    columns={
                            'index':'Column',
                            0:'Missing_Percentage (%)'
                    }
                )
    df2_missing_value = df2.isnull().sum()
    df2_missing_value = df2_missing_value.reset_index().rename(
                    columns={
                            'index':'Column',
                            0:'Missing_value_count'
                    }
                )
    # Sort the data frame
    #df2_missing = df2_missing.sort_values('Missing_Percentage (%)', ascending=False)
    Final = df2_missing_value.merge(df2_missing_count_percent, how = 'inner', left_on = 'Column', right_on = 'Column')
    Final = Final.sort_values(by = 'Missing_Percentage (%)',ascending = False)
    return Final
```

In [42]:
```
# Applying the custom function
missing_pct(df2)
```

Out[42]:

	Column	Missing_value_count	Missing_Percentage (%)
21	Color	27442	47.16
0	ProductKey	0	0.00
36	TotalChildren	0	0.00
27	ProductDescription	0	0.00
28	StartDate	0	0.00
29	FirstName	0	0.00
30	LastName	0	0.00
31	FullName	0	0.00
32	BirthDate	0	0.00
33	MaritalStatus	0	0.00
34	Gender	0	0.00
35	YearlyIncome	0	0.00
37	NumberChildrenAtHome	0	0.00
25	ModelName	0	0.00
38	Education	0	0.00
39	Occupation	0	0.00
40	HouseOwnerFlag	0	0.00
41	NumberCarsOwned	0	0.00
42	AddressLine1	0	0.00
43	DateFirstPurchase	0	0.00
44	CommuteDistance	0	0.00
45	Region	0	0.00
46	Country	0	0.00
47	Group	0	0.00

	Column	Missing_value_count	Missing_Percentage (%)
26	Photo	0	0.00
24	ProductLine	0	0.00
1	OrderDate	0	0.00
11	SalesAmount	0	0.00
2	ShipDate	0	0.00
3	CustomerKey	0	0.00
4	PromotionKey	0	0.00
5	SalesTerritoryKey	0	0.00
6	SalesOrderNumber	0	0.00
7	SalesOrderLineNumber	0	0.00
8	OrderQuantity	0	0.00
9	UnitPrice	0	0.00
10	TotalProductCost	0	0.00
12	TaxAmt	0	0.00
23	DaysToManufacture	0	0.00
13	StandardCost_x	0	0.00
14	List Price	0	0.00
15	diif std cost	0	0.00
16	diff list price	0	0.00
17	ProductName	0	0.00
18	SubCategory	0	0.00
19	Category	0	0.00
20	StandardCost_y	0	0.00
22	ListPrice	0	0.00

	Column	Missing_value_count	Missing_Percentage (%)
48	RegionImage	0	0.00

In [43]:
```
#  Drop columns with nan values
df2= df2.dropna(axis=1)
```

In [44]:
```
df2
```

Out[44]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrderNumber	SalesOrderLineNumber	OrderQuantity	Un
0	310	2014-01-01	2014-01-08	21768	1	6	SO43697	1	2	178
1	600	2016-04-16	2016-04-23	21768	1	6	SO56212	1	1	53
2	310	2014-01-30	2014-02-06	21727	1	6	SO43833	1	4	89
3	479	2016-11-29	2016-12-05	21727	1	6	SO71614	2	1	
4	477	2016-11-29	2016-12-05	21727	1	6	SO71614	3	1	
...	...	...	...	...	...	...	...	...	...	
58184	528	2016-11-07	2016-11-14	13145	1	2	SO70064	2	1	
58185	361	2016-11-07	2016-11-14	13145	1	2	SO70064	1	1	229
58186	480	2016-11-07	2016-11-14	13145	1	2	SO70064	4	1	
58187	530	2016-02-06	2016-02-13	27040	1	2	SO52124	1	1	
58188	480	2016-02-06	2016-02-13	27040	2	2	SO52124	2	1	

58189 rows × 48 columns

In [45]:

```
df2.columns
```

Out[45]:
```
Index(['ProductKey', 'OrderDate', 'ShipDate', 'CustomerKey', 'PromotionKey',
       'SalesTerritoryKey', 'SalesOrderNumber', 'SalesOrderLineNumber',
       'OrderQuantity', 'UnitPrice', 'TotalProductCost', 'SalesAmount',
       'TaxAmt', 'StandardCost_x', 'List Price', 'diif std cost',
       'diff list price', 'ProductName', 'SubCategory', 'Category',
       'StandardCost_y', 'ListPrice', 'DaysToManufacture', 'ProductLine',
       'ModelName', 'Photo', 'ProductDescription', 'StartDate', 'FirstName',
       'LastName', 'FullName', 'BirthDate', 'MaritalStatus', 'Gender',
       'YearlyIncome', 'TotalChildren', 'NumberChildrenAtHome', 'Education',
       'Occupation', 'HouseOwnerFlag', 'NumberCarsOwned', 'AddressLine1',
       'DateFirstPurchase', 'CommuteDistance', 'Region', 'Country', 'Group',
       'RegionImage'],
      dtype='object')
```

Adding Columns

In [46]:
```
# Extracting Year from OrderDate
df2['sale_year'] = df2['OrderDate'].dt.year

# Extracting Month from OrderDate
df2['sale_month'] = df2['OrderDate'].dt.month

# Extracting day from OrderDate
df2['sale_day'] = df2['OrderDate'].dt.day

# Extracting dayofweek from OrderDate
df2['sale_week'] = df2['OrderDate'].dt.dayofweek

# Extracting day_name from OrderDate
df2['sale_day_name'] = df2['OrderDate'].dt.day_name()

# Extracting Month Year from OrderDate
df2['year_month'] = df2['OrderDate'].apply(lambda x:x.strftime('%Y-%m'))

# Calculate Total Invoice Amount
df2['total_Invoice_amount'] = df2['SalesAmount'] + df2['TaxAmt']

# Considering only salesamount and total_sales_amount to calculate profit
df2['profit'] = (df2['UnitPrice']*df2['OrderQuantity']) - df2['TotalProductCost']

# Removing extra character from the string
df2['ProductName'] = df2['ProductName'].str.replace(',','-')
```

```
# Calculate Age
df2['Age'] = df2['OrderDate'].dt.year - df2['BirthDate'].dt.year
```

In [47]:
```
df2
```

Out[47]:
```
       ProductKey  OrderDate   ShipDate  CustomerKey  PromotionKey  SalesTerritoryKey  SalesOrderNumber  SalesOrderLineNumber  OrderQuantity  Un
    0         310  2014-01-01  2014-01-08      21768             1                  6           SO43697                     1              2  178
    1         600  2016-04-16  2016-04-23      21768             1                  6           SO56212                     1              1   53
    2         310  2014-01-30  2014-02-06      21727             1                  6           SO43833                     1              4   89
    3         479  2016-11-29  2016-12-05      21727             1                  6           SO71614                     2              1
    4         477  2016-11-29  2016-12-05      21727             1                  6           SO71614                     3              1
  ...         ...         ...         ...        ...           ...                ...               ...                   ...            ...
58184         528  2016-11-07  2016-11-14      13145             1                  2           SO70064                     2              1
58185         361  2016-11-07  2016-11-14      13145             1                  2           SO70064                     1              1  229
58186         480  2016-11-07  2016-11-14      13145             1                  2           SO70064                     4              1
58187         530  2016-02-06  2016-02-13      27040             1                  2           SO52124                     1              1
58188         480  2016-02-06  2016-02-13      27040             2                  2           SO52124                     2              1

58189 rows × 57 columns
```

Exploring data

List of product's category

In [48]:
```
df2.groupby('sale_year')['SalesAmount'].sum()
```

Out[48]:
```
sale_year
2014    7.072084e+06
2015    5.762134e+06
2016    1.647362e+07
Name: SalesAmount, dtype: float64
```

List of product's subcategory

In [49]:
```
df2['SubCategory'].unique().tolist()
```

Out[49]:
```
['Road Bikes',
 'Mountain Bikes',
 'Bottles and Cages',
 'Gloves',
 'Tires and Tubes',
 'Helmets',
 'Touring Bikes',
 'Jerseys',
 'Cleaners',
 'Caps',
 'Hydration Packs',
 'Socks',
 'Fenders',
 'Vests',
 'Bike Racks',
 'Bike Stands',
 'Shorts']
```

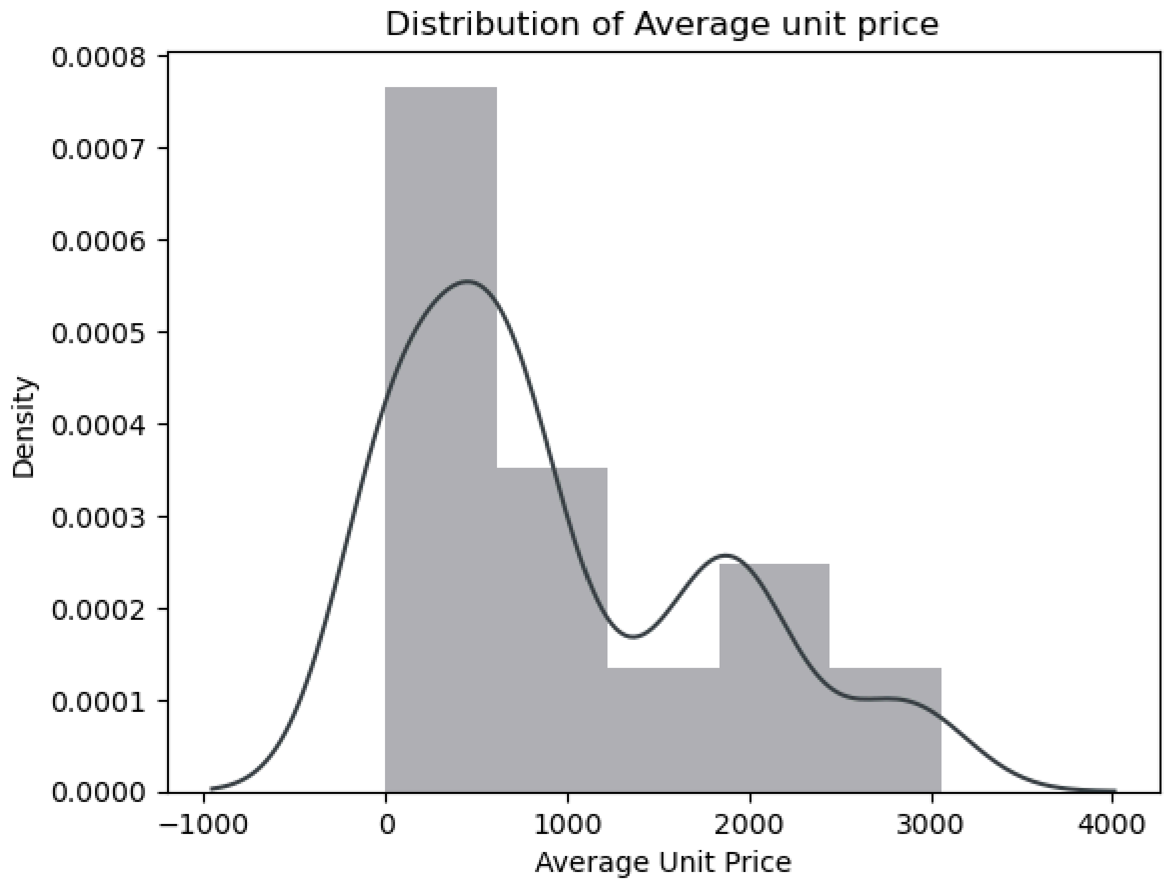
Analysing UnitPrice

In [50]:
```
Avg_unit_price = df2.groupby('ProductKey')['UnitPrice'].mean()
ax = sns.distplot(Avg_unit_price, kde=True, hist=True, color='#374045')
ax.set(title='Distribution of Average unit price',
       xlabel='Average Unit Price')
```

```
C:\Users\ersum\AppData\Local\Temp\ipykernel_15792\1028881447.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax = sns.distplot(Avg_unit_price, kde=True, hist=True, color='#374045')
```

Out[50]:

```
[Text(0.5, 1.0, 'Distribution of Average unit price'),
 Text(0.5, 0, 'Average Unit Price')]
```



In [51]:

```
#  Maximum of the product unit price is below $1000
```

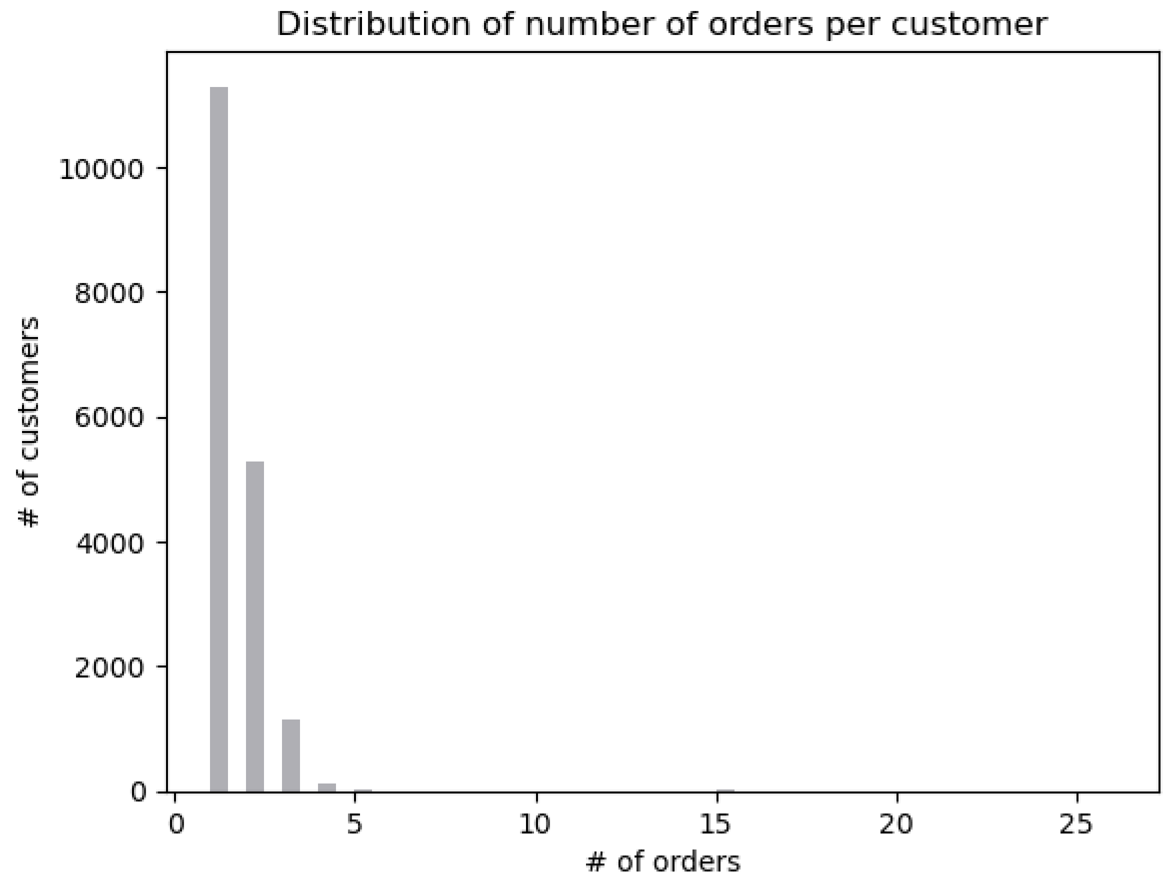
Sales order number distribution

```
In [52]: n_orders = df2.groupby(['CustomerKey'])['SalesOrderNumber'].nunique()
         multi_orders_perc = np.sum(n_orders > 1)/df2['CustomerKey'].nunique()
         print(f"{100*multi_orders_perc:.2f}% of customers ordered more than once.")
```

```
36.97% of customers ordered more than once.
```

```
In [53]: ax = sns.distplot(n_orders, kde=False, color='#374045')
         ax.set(title='Distribution of number of orders per customer',
                xlabel='# of orders',
                ylabel='# of customers');
```

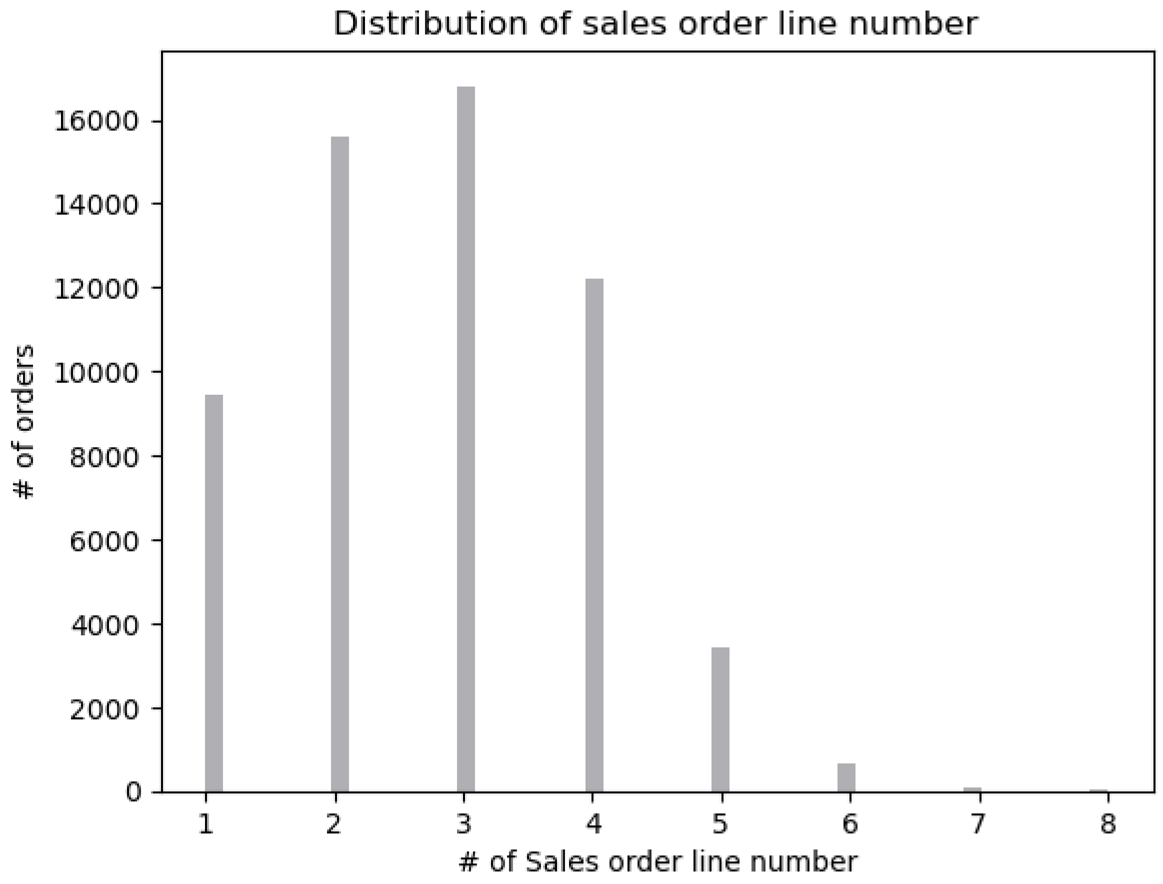
```
C:\Users\ersum\AppData\Local\Temp\ipykernel_15792\1110922211.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax = sns.distplot(n_orders, kde=False, color='#374045')
```



Sales order line number distribution

```
In [54]: n_salesordernumber = df2.groupby(['SalesOrderNumber'])['SalesOrderLineNumber'].transform('max')
         ax = sns.distplot(n_salesordernumber, kde=False, color='#374045')
         ax.set(title='Distribution of sales order line number',
                xlabel='# of Sales order line number',
                ylabel='# of orders');
```

```
C:\Users\ersum\AppData\Local\Temp\ipykernel_15792\1353084701.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax = sns.distplot(n_salesordernumber, kde=False, color='#374045')
```

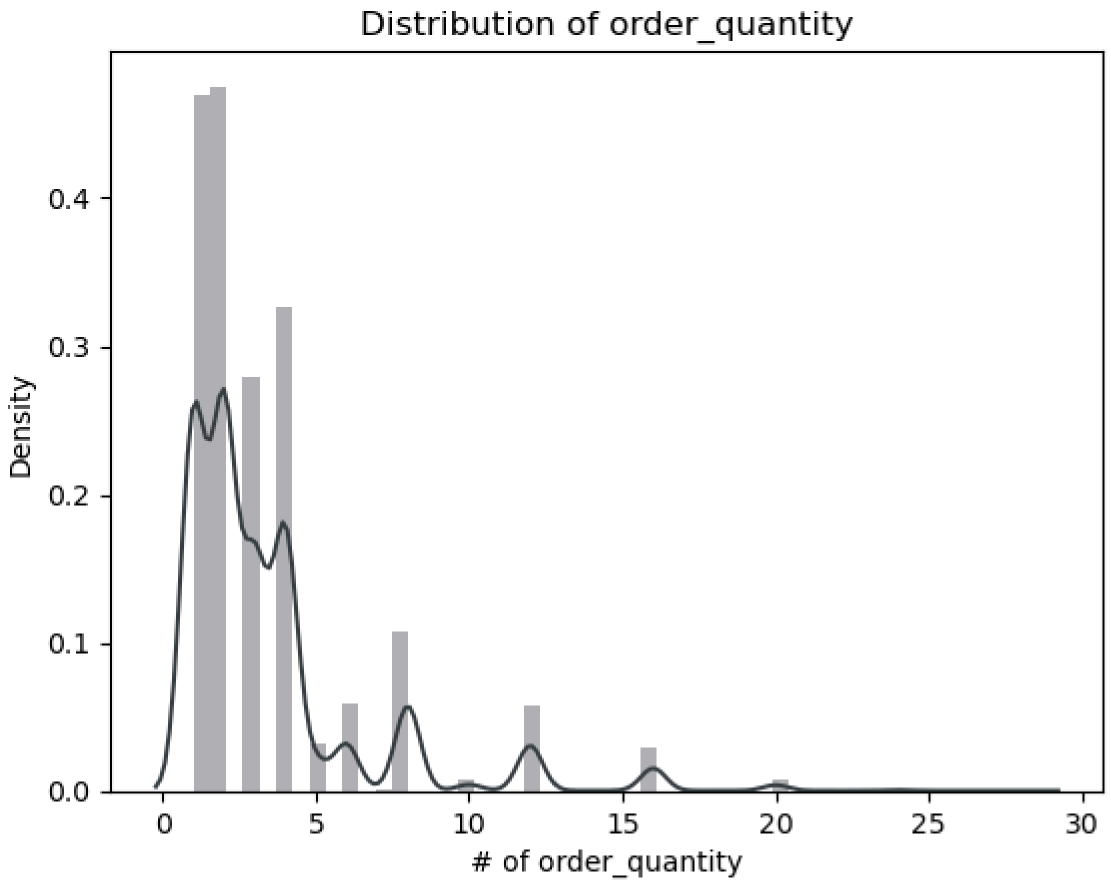


In [55]:
```
# Most of the time two to three products are ordered in a single order
```

Sales Order Quantity distribution

In [56]:
```
n_order_quantity = df2.groupby(['SalesOrderNumber'])['OrderQuantity'].sum()
ax = sns.distplot(n_order_quantity, kde=True, hist=True,color='#374045')
ax.set(title='Distribution of order_quantity',
       xlabel='# of order_quantity',
       );
```

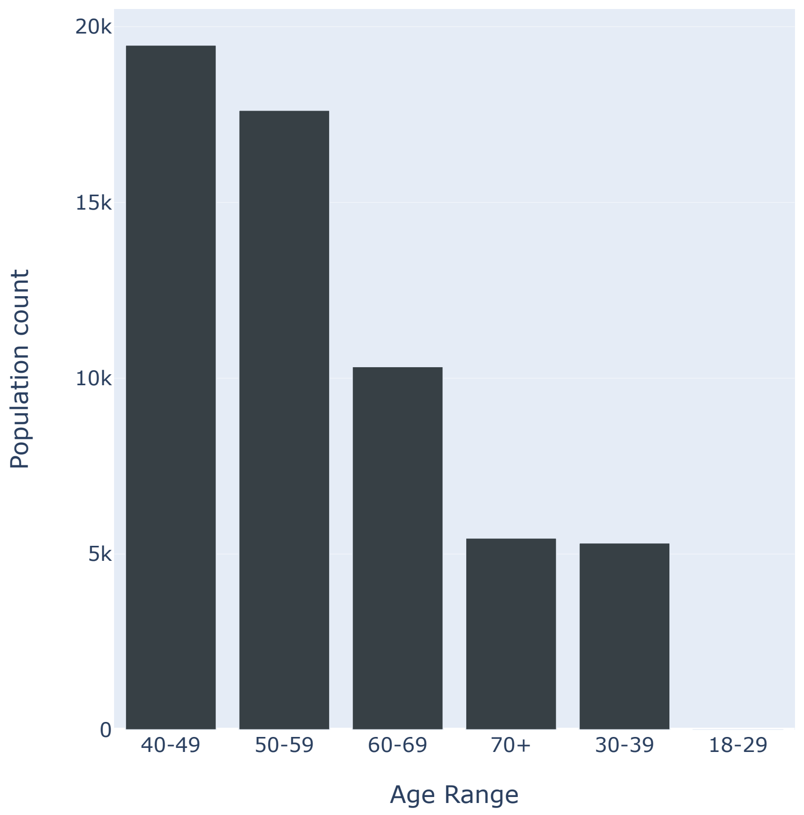
```
C:\Users\ersum\AppData\Local\Temp\ipykernel_15792\1426996600.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  ax = sns.distplot(n_order_quantity, kde=True, hist=True,color='#374045')
```



```
In [57]: # maximum quantity ordered for a product is below 2.5
```

Age Distribution

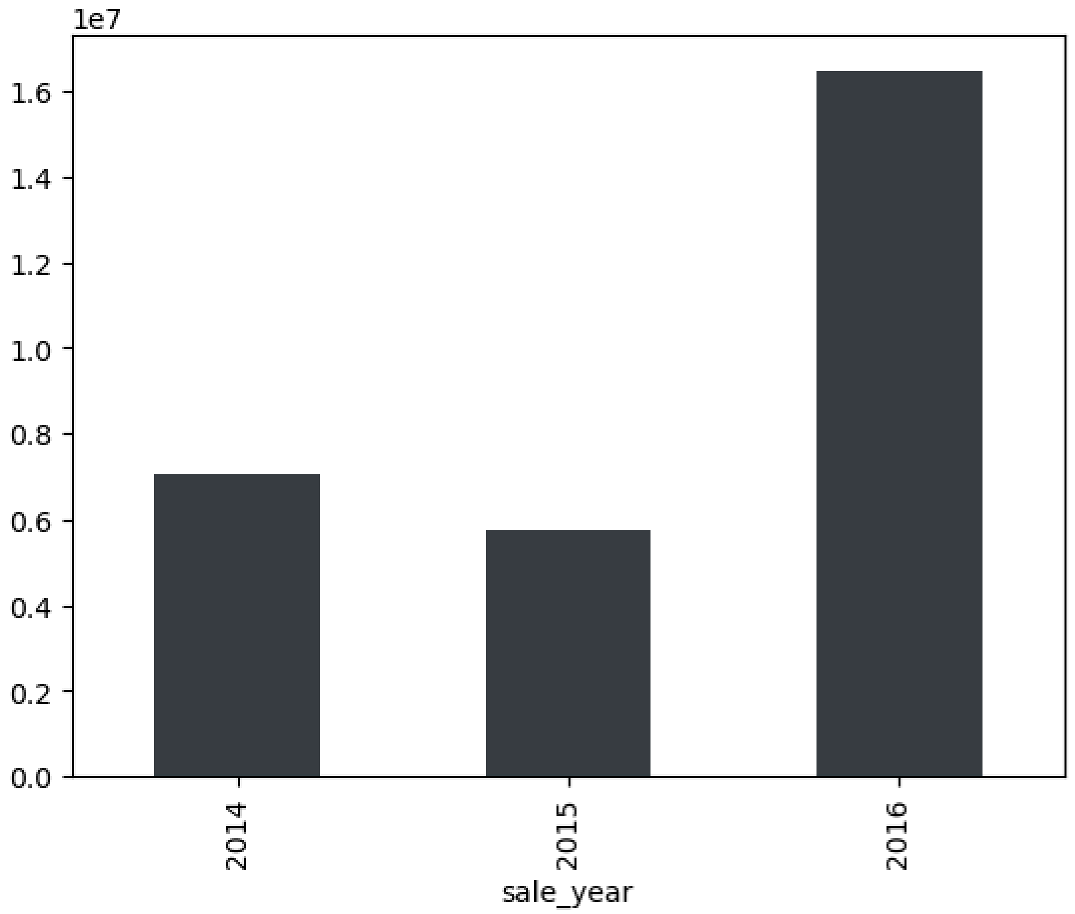
```
In [58]: bins = [18, 30, 40, 50, 60, 70, 120]
labels = ['18-29', '30-39', '40-49', '50-59', '60-69', '70+']
df2['agerange'] = pd.cut(df2.Age, bins, labels = labels,include_lowest = True)

age_distribution = df2['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_discrete_sequence=['#374045'])
fig.update_layout(
    autosize=True,
    width=500,
    height=500,
    font=dict(size=10))
fig.show()
```



In [59]:
```
# THe maximum no. of clients are between the ages of 40 and 59.
```

Sales Year wise sales

In [60]:
```
df2.groupby('sale_year')['SalesAmount'].sum().plot(kind='bar', color='#374045');
```



In [62]:
```
# sales in 2016 are maximum.sales initially start decreasing from 2014 to 2015 the increases gradually in 2016.
```

Top 5 Selling Product

In [64]:
```
top_selling_product = df2.groupby(['Category', 'SubCategory', 'ProductName'])['OrderQuantity'].sum().sort_values(ascending=False)
top_selling_product.head(5)
```

```
Out[64]: 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
         Name: OrderQuantity, dtype: int64
```

Quantity ordered based on category and subcategory from 2014 to 2016

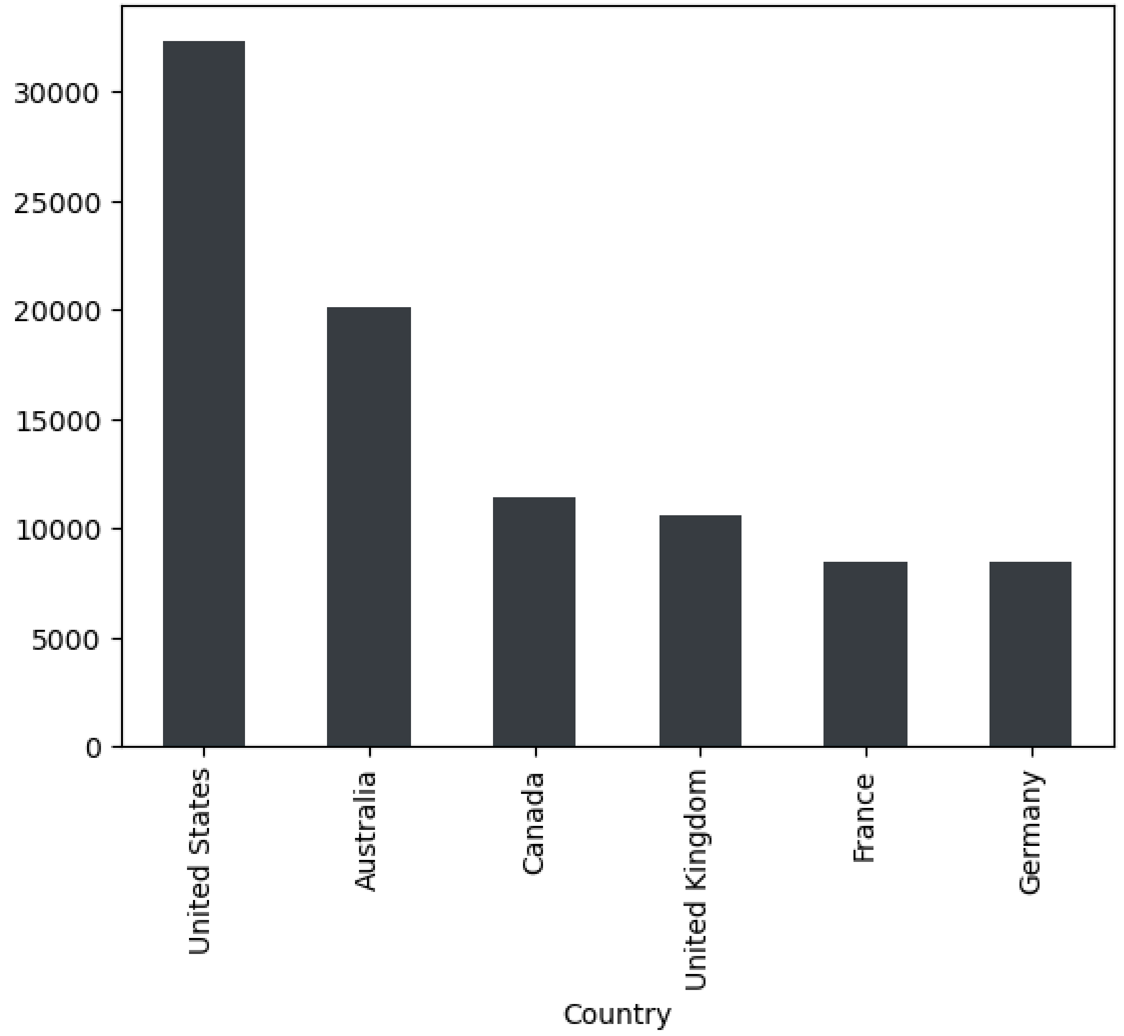
```
In [65]: cat_subcat_qty = df2.groupby(['sale_year','Category', 'SubCategory'])['OrderQuantity'].sum()
         cat_subcat_qty
```

```
Out[65]: sale_year  Category     SubCategory
         2014       Bikes        Mountain Bikes         616
                                 Road Bikes            2876
         2015       Bikes        Mountain Bikes        1661
                                 Road Bikes            3284
         2016       Accessories  Bike Racks             493
                                 Bike Stands            394
                                 Bottles and Cages    12055
                                 Cleaners              1381
                                 Fenders               3239
                                 Helmets               9685
                                 Hydration Packs       1124
                                 Tires and Tubes      25518
                    Bikes        Mountain Bikes        5490
                                 Road Bikes            6535
                                 Touring Bikes         3410
                    Clothing     Caps                  3178
                                 Gloves                2143
                                 Jerseys               5068
                                 Shorts                1491
                                 Socks                  856
                                 Vests                  824
         Name: OrderQuantity, dtype: int64
```

Country wise quantity ordered

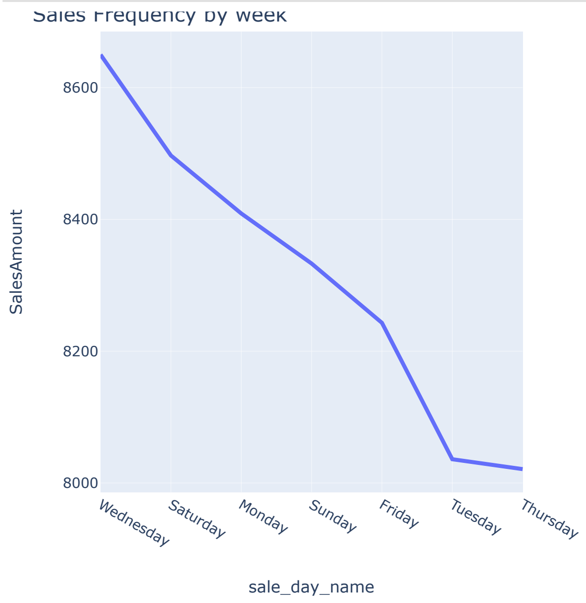
```
In [66]: country_qty_sales = df2.groupby('Country')['OrderQuantity'].sum().sort_values(ascending=False)
         country_qty_sales.plot(kind='bar', color='#374045')
```

```
Out[66]: <Axes: xlabel='Country'>
```



Highest quantity of products is ordered from United States.

```
In [67]: sales_by_week = df2.groupby(['sale_day_name']).count()['SalesAmount'].reset_index().sort_values('SalesAmount', ascending=False)

fig = px.line(sales_by_week, x='sale_day_name', y='SalesAmount', title='Sales Frequency by week')
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()
```



In [ ]:
```
#High sales orders are seen on Wednesday and Saturday, therefore we can promote our product during these workweek
```

In [68]:
```
df2
```

Out[68]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	SalesTerritoryKey	SalesOrderNumber	SalesOrderLineNumber	OrderQuantity	Un
0	310	2014-01-01	2014-01-08	21768	1	6	SO43697	1	2	17
1	600	2016-04-16	2016-04-23	21768	1	6	SO56212	1	1	5
2	310	2014-01-30	2014-02-06	21727	1	6	SO43833	1	4	8
3	479	2016-11-29	2016-12-05	21727	1	6	SO71614	2	1	
4	477	2016-11-29	2016-12-05	21727	1	6	SO71614	3	1	
...	...	...	...	...	...	...	...	...	...	
58184	528	2016-11-07	2016-11-14	13145	1	2	SO70064	2	1	
58185	361	2016-11-07	2016-11-14	13145	1	2	SO70064	1	1	22
58186	480	2016-11-07	2016-11-14	13145	1	2	SO70064	4	1	
58187	530	2016-02-06	2016-02-13	27040	1	2	SO52124	1	1	
58188	480	2016-02-06	2016-02-13	27040	2	2	SO52124	2	1	

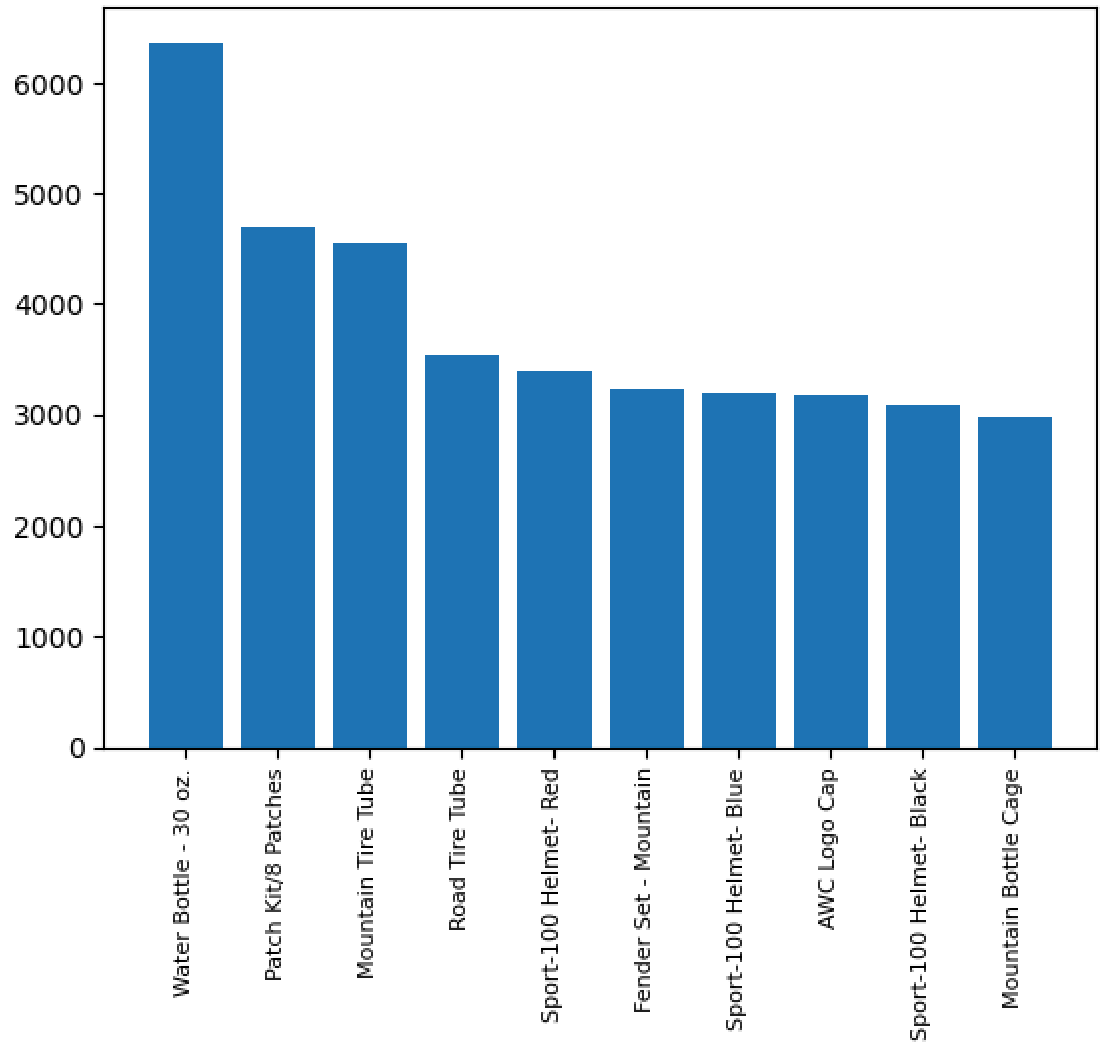
58189 rows × 58 columns

Which product sold the most? why do you think it sold the most?

In [69]:

```python
product_group = df2.groupby('ProductName')
quantity_ordered = product_group['OrderQuantity'].sum().sort_values(ascending=False)[:10]
products = quantity_ordered.index.tolist()
```

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

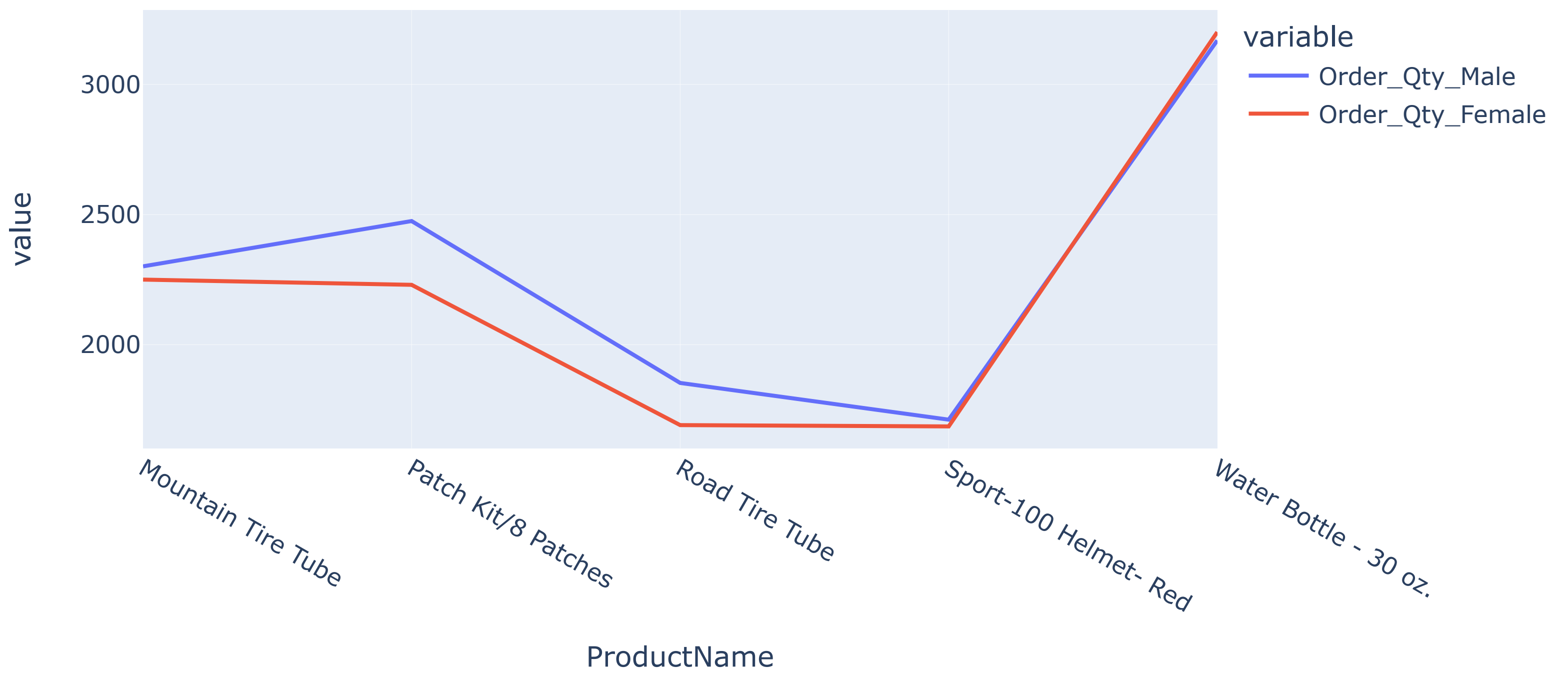


Compare most ordered product by gender

```
In [70]: male = df2[df2["Gender"]=="M"]
         female = df2[df2["Gender"]=="F"]
```

In [71]:
```
male_ord_qty = male.groupby(['ProductName'],as_index=False)['OrderQuantity'].sum().nlargest(5,'OrderQuantity').sort_values('Produ
male_ord_qty.columns=['ProductName','Order_Qty_Male']

female_ord_qty = female.groupby(['ProductName'],as_index=False)['OrderQuantity'].sum().nlargest(5,'OrderQuantity').sort_values('P
female_ord_qty.columns=['ProductName','Order_Qty_Female']

df2_merge = pd.merge(male_ord_qty, female_ord_qty, on='ProductName')
```

In [72]:
```
fig = px.line(df2_merge, x="ProductName", y=["Order_Qty_Male","Order_Qty_Female"])
fig.update_layout(
    autosize=True,
    width=800,
    height=400)
fig.show()
```



In [73]:
```
#  calculating recency for customers who had made a purchase with a company
```

```
df_recency = df2.groupby(by='FullName',
                         as_index=False)['OrderDate'].max()
df_recency.columns = ['CustomerName', 'LastPurchaseDate']
recent_date = df_recency['LastPurchaseDate'].max()
df_recency['Recency'] = df_recency['LastPurchaseDate'].apply(
    lambda x: (recent_date - x).days)
```

In [74]:
```
df_recency['Recency'].head()
```

Out[74]:
```
0    244
1     15
2    126
3    911
4     83
Name: Recency, dtype: int64
```

In [75]:
```
# Recency means How recently has the customer made a transaction with us
```

In [76]:
```
#  calculating the frequency of frequent transactions of the
#  customer in ordering/buying some product from the company.


frequency_df = df2.drop_duplicates().groupby(
    by=['FullName'], as_index=False)['OrderDate'].count()
frequency_df.columns = ['CustomerName', 'Frequency']
frequency_df.head()
```

Out[76]:

	CustomerName	Frequency
0	Adams, Aaron	4
1	Adams, Adam	2
2	Adams, Alex	2
3	Adams, Alexandra	1
4	Adams, Allison	3

In [77]:
```
# Monetary: How much does the customer spend on purchasing products from us
monetary_df = df2.groupby('FullName', as_index=False)['SalesAmount'].sum()
monetary_df.columns = ['CustomerName', 'Monetary']
monetary_df.head()
```

Out[77]:

	CustomerName	Monetary
0	Adams, Aaron	117.96
1	Adams, Adam	141.98
2	Adams, Alex	1735.98
3	Adams, Alexandra	3578.27
4	Adams, Allison	1602.47

In [78]:

```
df2.dtypes
```

Out[78]:
```
ProductKey                       int64
OrderDate               datetime64[ns]
ShipDate                datetime64[ns]
CustomerKey                      int64
PromotionKey                     int64
SalesTerritoryKey                int64
SalesOrderNumber                object
SalesOrderLineNumber             int64
OrderQuantity                    int64
UnitPrice                      float64
TotalProductCost               float64
SalesAmount                    float64
TaxAmt                         float64
StandardCost_x                 float64
List Price                     float64
diif std cost                    int64
diff list price                  int64
ProductName                     object
SubCategory                     object
Category                        object
StandardCost_y                 float64
ListPrice                      float64
DaysToManufacture                int64
ProductLine                     object
ModelName                       object
Photo                           object
ProductDescription              object
StartDate               datetime64[ns]
FirstName                       object
LastName                        object
FullName                        object
BirthDate               datetime64[ns]
MaritalStatus                   object
Gender                          object
YearlyIncome                     int64
TotalChildren                    int64
NumberChildrenAtHome             int64
Education                       object
Occupation                      object
HouseOwnerFlag                   int64
NumberCarsOwned                  int64
AddressLine1                    object
DateFirstPurchase       datetime64[ns]
CommuteDistance                 object
```

```
Region                          object
Country                         object
Group                           object
RegionImage                     object
sale_year                        int32
sale_month                       int32
sale_day                         int32
sale_week                        int32
sale_day_name                   object
year_month                      object
total_Invoice_amount           float64
profit                         float64
Age                              int32
agerange                      category
dtype: object
```

In [ ]:
```
# Count the customer Id
```

In [79]:
```
len(df2['CustomerKey'].unique())
```

Out[79]:
```
17918
```

In [ ]:
```
# Total 17918 Customer Id's are there.
```

In [ ]:
```

```

In [80]:
```
df2.groupby(['CustomerKey','FullName'])['OrderQuantity'].sum().sort_values(ascending=False)
```

Out[80]:
```
CustomerKey  FullName
11200        Griffin, Jason       115
11300        Barnes, Fernando     112
11331        Jenkins, Samantha    107
11262        Simmons, Jennifer    100
11277        Jackson, Charles      95
                                 ...
27286        Pal, Shawn             1
27289        Long, Anna             1
27290        Evans, Marcus          1
18964        Gonzalez, Thomas       1
29483        Navarro, Jésus         1
Name: OrderQuantity, Length: 17918, dtype: int64
```

In [81]:
```
# Griffin Jason is the person who ordered highest quantity .
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: