budget-sales-analysis

July 17, 2024

[1]:
```
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')
```

1 Importing excel files

[2]:
```
import numpy as np
import pandas as pd
AdventureWorks_Database=pd.read_excel(r"AdventureWorks_Database.xlsx")
AdventureWorks_Database
```

[2]:
```
            Date   DateKey  Year Quarter  MonthNum Month FiscalYear  \
0     2016-04-03  20160403  2016      Q2         4   Apr     FY2016
1     2016-04-04  20160404  2016      Q2         4   Apr     FY2016
2     2016-04-05  20160405  2016      Q2         4   Apr     FY2016
3     2016-04-06  20160406  2016      Q2         4   Apr     FY2016
4     2016-04-07  20160407  2016      Q2         4   Apr     FY2016
…          …        …    …    …     …        …   …         …
1456 2014-06-18  20140618  2014      Q2         6   Jun     FY2014
1457 2014-06-19  20140619  2014      Q2         6   Jun     FY2014
1458 2014-06-20  20140620  2014      Q2         6   Jun     FY2014
1459 2014-06-21  20140621  2014      Q2         6   Jun     FY2014
1460 2014-06-22  20140622  2014      Q2         6   Jun     FY2014

     FiscalQuarter  FiscalMonthNum FiscalMonth MonthYear MonthYearLong  \
0              FQ4              10         Apr    Apr-16      Apr-2016
1              FQ4              10         Apr    Apr-16      Apr-2016
2              FQ4              10         Apr    Apr-16      Apr-2016
```

```
3              FQ4               10          Apr    Apr-16      Apr-2016
4              FQ4               10          Apr    Apr-16      Apr-2016
…              …                 …           …        …           …
1456           FQ4               12          Jun    Jun-14      Jun-2014
1457           FQ4               12          Jun    Jun-14      Jun-2014
1458           FQ4               12          Jun    Jun-14      Jun-2014
1459           FQ4               12          Jun    Jun-14      Jun-2014
1460           FQ4               12          Jun    Jun-14      Jun-2014

      MonthYearNum  WeekdayNum Weekday WeekdayWeekend
0           201604           1     Sun        Weekend
1           201604           2     Mon        Weekday
2           201604           3     Tue        Weekday
3           201604           4     Wed        Weekday
4           201604           5     Thu        Weekday
…              …         …      …             …
1456        201406           4     Wed        Weekday
1457        201406           5     Thu        Weekday
1458        201406           6     Fri        Weekday
1459        201406           7     Sat        Weekend
1460        201406           1     Sun        Weekend

[1461 rows x 16 columns]
```

2 Importing “calender” file

[3]:
```
import numpy as np
import pandas as pd
calender=pd.read_excel(r"calender.xlsx")
calender
```

[3]:
```
            Date   DateKey  Year Quarter  MonthNum Month FiscalYear  \
0    2016-04-03  20160403  2016      Q2         4   Apr     FY2016
1    2016-04-04  20160404  2016      Q2         4   Apr     FY2016
2    2016-04-05  20160405  2016      Q2         4   Apr     FY2016
3    2016-04-06  20160406  2016      Q2         4   Apr     FY2016
4    2016-04-07  20160407  2016      Q2         4   Apr     FY2016
…         …       …    …     …        …       …   …         …
1456 2014-06-18  20140618  2014      Q2         6   Jun     FY2014
1457 2014-06-19  20140619  2014      Q2         6   Jun     FY2014
1458 2014-06-20  20140620  2014      Q2         6   Jun     FY2014
1459 2014-06-21  20140621  2014      Q2         6   Jun     FY2014
1460 2014-06-22  20140622  2014      Q2         6   Jun     FY2014

     FiscalQuarter  FiscalMonthNum FiscalMonth MonthYear MonthYearLong  \
0              FQ4              10         Apr    Apr-16      Apr-2016
```

```
1              FQ4              10          Apr    Apr-16      Apr-2016
2              FQ4              10          Apr    Apr-16      Apr-2016
3              FQ4              10          Apr    Apr-16      Apr-2016
4              FQ4              10          Apr    Apr-16      Apr-2016
…              …                …           …      …           …
1456           FQ4              12          Jun    Jun-14      Jun-2014
1457           FQ4              12          Jun    Jun-14      Jun-2014
1458           FQ4              12          Jun    Jun-14      Jun-2014
1459           FQ4              12          Jun    Jun-14      Jun-2014
1460           FQ4              12          Jun    Jun-14      Jun-2014

      MonthYearNum  WeekdayNum Weekday WeekdayWeekend
0           201604           1     Sun        Weekend
1           201604           2     Mon        Weekday
2           201604           3     Tue        Weekday
3           201604           4     Wed        Weekday
4           201604           5     Thu        Weekday
…             …          …     …             …
1456        201406           4     Wed        Weekday
1457        201406           5     Thu        Weekday
1458        201406           6     Fri        Weekday
1459        201406           7     Sat        Weekend
1460        201406           1     Sun        Weekend

[1461 rows x 16 columns]
```

[4]:
```python
import pandas as pd

column_names = calender.columns
print("Column Names:")

for col in column_names:
    print(col)
```

```
Column Names:
Date
DateKey
Year
Quarter
MonthNum
Month
FiscalYear
FiscalQuarter
FiscalMonthNum
FiscalMonth
MonthYear
MonthYearLong
```

```
MonthYearNum
WeekdayNum
Weekday
WeekdayWeekend
```

[ ]:

3 Importing “customer” file

[5]:
```
import numpy as np
import pandas as pd
customer=pd.read_excel(r"customer.xlsx")
customer
```

[5]:

	CustomerKey	FirstName	LastName	FullName	BirthDate
0	11000	Jon	Yang	Yang, Jon	1966-04-08
1	11001	Eugene	Huang	Huang, Eugene	1965-05-14
2	11002	Ruben	Torres	Torres, Ruben	1965-08-12
3	11003	Christy	Zhu	Zhu, Christy	1968-02-15
4	11004	Elizabeth	Johnson	Johnson, Elizabeth	1968-08-08
…	…	…	…	…	…
18479	29479	Tommy	Tang	Tang, Tommy	1958-07-04
18480	29480	Nina	Raji	Raji, Nina	1960-11-10
18481	29481	Ivan	Suri	Suri, Ivan	1960-01-05
18482	29482	Clayton	Zhang	Zhang, Clayton	1959-03-05
18483	29483	Jésus	Navarro	Navarro, Jésus	1959-12-08

	MaritalStatus	Gender	YearlyIncome	TotalChildren	NumberChildrenAtHome
0	M	M	90000	2	0
1	S	M	60000	3	3
2	M	M	60000	3	3
3	S	F	70000	0	0
4	S	F	80000	5	5
…	…	…	…	…	…
18479	M	M	30000	1	0
18480	S	F	30000	3	0
18481	S	M	30000	3	0
18482	M	M	30000	3	0
18483	M	M	30000	0	0

	Education	Occupation	HouseOwnerFlag	NumberCarsOwned
0	Bachelors	Professional	1	0
1	Bachelors	Professional	0	1
2	Bachelors	Professional	1	1
3	Bachelors	Professional	0	1
4	Bachelors	Professional	1	4

```
…                …            …              …               …
18479  Graduate Degree      Clerical              1               0
18480  Graduate Degree      Clerical              1               0
18481  Graduate Degree      Clerical              0               0
18482        Bachelors      Clerical              1               0
18483        Bachelors      Clerical              1               0

                    AddressLine1 DateFirstPurchase CommuteDistance
0                 3761 N. 14th St        2014-01-22       1-2 Miles
1                      2243 W St.        2014-01-18       0-1 Miles
2                5844 Linden Land        2014-01-10       2-5 Miles
3                1825 Village Pl.        2014-01-01      5-10 Miles
4             7553 Harness Circle        2014-01-26       1-2 Miles
…                             …              …               …
18479           111, rue Maillard        2015-09-08       0-1 Miles
18480           9 Katherine Drive        2016-07-18       0-1 Miles
18481                Knaackstr 4        2014-08-13       0-1 Miles
18482     1080, quai de Grenelle        2015-09-22       0-1 Miles
18483  244, rue de la Centenaire        2015-09-13       0-1 Miles

[18484 rows x 17 columns]
```

[6]:
```
column_names=customer.columns
print("column names ")
for col in column_names:
    print(col)
```

```
column names
CustomerKey
FirstName
LastName
FullName
BirthDate
MaritalStatus
Gender
YearlyIncome
TotalChildren
NumberChildrenAtHome
Education
Occupation
HouseOwnerFlag
NumberCarsOwned
AddressLine1
DateFirstPurchase
CommuteDistance
```

4 Importing “product” file

[7]:
```
import numpy as np
import pandas as pd
product=pd.read_excel(r"product.xlsx")
product
```

[7]:
```
     ProductKey            ProductName      SubCategory    Category  \
0             1        Adjustable Race              NaN         NaN
1             2           Bearing Ball              NaN         NaN
2             3        BB Ball Bearing              NaN         NaN
3             4  Headset Ball Bearings              NaN         NaN
4             5                  Blade              NaN         NaN
..          …                      …              …         …
601         602      ML Bottom Bracket  Bottom Brackets  Components
602         603      HL Bottom Bracket  Bottom Brackets  Components
603         604     Road-750 Black, 44       Road Bikes       Bikes
604         605     Road-750 Black, 48       Road Bikes       Bikes
605         606     Road-750 Black, 52       Road Bikes       Bikes

     StandardCost  Color  ListPrice  DaysToManufacture ProductLine  \
0             NaN    NaN        NaN                  0         NaN
1             NaN    NaN        NaN                  0         NaN
2             NaN    NaN        NaN                  1         NaN
3             NaN    NaN        NaN                  0         NaN
4             NaN    NaN        NaN                  1         NaN
..            …    …         …                  …          …
601       44.9506    NaN     101.24                  1         NaN
602       53.9416    NaN     121.49                  1         NaN
603     343.6496  Black     539.99                  4        Road
604     343.6496  Black     539.99                  4        Road
605     343.6496  Black     539.99                  4        Road

            ModelName                                              Photo  \
0                 NaN  http://www.avising.com/me/LearnPBI/DataSources…
1                 NaN  http://www.avising.com/me/LearnPBI/DataSources…
2                 NaN  http://www.avising.com/me/LearnPBI/DataSources…
3                 NaN  http://www.avising.com/me/LearnPBI/DataSources…
4                 NaN  http://www.avising.com/me/LearnPBI/DataSources…
..                …                                                 …
601  ML Bottom Bracket  http://www.avising.com/me/LearnPBI/DataSources…
602  HL Bottom Bracket  http://www.avising.com/me/LearnPBI/DataSources…
603          Road-750  http://www.avising.com/me/LearnPBI/DataSources…
604          Road-750  http://www.avising.com/me/LearnPBI/DataSources…
605          Road-750  http://www.avising.com/me/LearnPBI/DataSources…

                                ProductDescription  StartDate
```

```
0                                                    NaN 1998-06-01
1                                                    NaN 1998-06-01
2                                                    NaN 1998-06-01
3                                                    NaN 1998-06-01
4                                                    NaN 1998-06-01
..                                                   …          …
601       Aluminum alloy cups; large diameter spindle. 2007-07-01
602             Aluminum alloy cups and a hollow axle. 2007-07-01
603  Entry level adult bike; offers a comfortable r… 2007-07-01
604  Entry level adult bike; offers a comfortable r… 2007-07-01
605  Entry level adult bike; offers a comfortable r… 2007-07-01

[606 rows x 13 columns]
```

[8]:
```
import pandas as pd
column_names=product.columns
for col in column_names:
    print(col)
```

```
ProductKey
ProductName
SubCategory
Category
StandardCost
Color
ListPrice
DaysToManufacture
ProductLine
ModelName
Photo
ProductDescription
StartDate
```

5 Importing “sales” file

[9]:
```
import numpy as np
import pandas as pd
sales=pd.read_excel(r"sales.xlsx")
sales
```

[9]:
```
        ProductKey  OrderDate   ShipDate  CustomerKey  PromotionKey  \
0              310 2014-01-01 2014-01-08        21768             1
1              346 2014-01-01 2014-01-08        28389             1
2              346 2014-01-01 2014-01-08        25863             1
3              336 2014-01-01 2014-01-08        14501             1
4              346 2014-01-01 2014-01-08        11003             1
```

```
…           …        …          …           …            …
58184         561 2016-12-30 2017-01-07        13650            1
58185         584 2016-12-30 2017-01-07        26916            1
58186         605 2016-12-30 2017-01-07        27473            1
58187         538 2016-12-30 2017-01-07        27473            1
58188         490 2016-12-30 2017-01-07        27473            1

       SalesTerritoryKey SalesOrderNumber  SalesOrderLineNumber  \
0                      6          SO43697                     1
1                      7          SO43698                     1
2                      1          SO43699                     1
3                      4          SO43700                     1
4                      9          SO43701                     1
…                    …                …                     …
58184                  9          SO74145                     1
58185                  9          SO74146                     1
58186                  9          SO74147                     1
58187                  9          SO74147                     2
58188                  9          SO74147                     3

       OrderQuantity  UnitPrice  TotalProductCost  SalesAmount    TaxAmt  \
0                  2  1789.1350         2171.2942    3578.2700  286.2616
1                  2  1699.9950         1912.1544    3399.9900  271.9992
2                  2  1699.9950         1912.1544    3399.9900  271.9992
3                  2   349.5491          413.1463     699.0982   55.9279
4                  2  1699.9950         1912.1544    3399.9900  271.9992
…                …          …                 …           …         …
58184              1  2384.0700         1481.9379    2384.0700  190.7256
58185              1   539.9900          343.6496     539.9900   43.1992
58186              1   539.9900          343.6496     539.9900   43.1992
58187              1    21.4900            8.0373      21.4900    1.7192
58188              1    53.9900           41.5723      53.9900    4.3192

       Unnamed: 13  Unnamed: 14  Unnamed: 15  StandardCost  List Price
0        3578.2700          0.0    -764.3184     2171.2942   3578.2700
1        3399.9900          0.0    -424.3188     1912.1544   3399.9900
2        3399.9900          0.0    -424.3188     1912.1544   3399.9900
3         699.0982          0.0    -127.1944      413.1463    699.0982
4        3399.9900          0.0    -424.3188     1912.1544   3399.9900
…              …          …            …           …          …
58184    2384.0700          0.0     902.1321     1481.9379   2384.0700
58185     539.9900          0.0     196.3404      343.6496    539.9900
58186     539.9900          0.0     196.3404      343.6496    539.9900
58187      21.4900          0.0      13.4527        8.0373     21.4900
58188      53.9900          0.0      12.4177       41.5723     53.9900

[58189 rows x 18 columns]
```

[10]:
```
import pandas as pd
column_names=sales.columns
for col in column_names:
    print(col)
```

```
ProductKey
OrderDate
ShipDate
CustomerKey
PromotionKey
SalesTerritoryKey
SalesOrderNumber
SalesOrderLineNumber
OrderQuantity
UnitPrice
TotalProductCost
SalesAmount
TaxAmt
Unnamed: 13
Unnamed: 14
Unnamed: 15
StandardCost
List Price
```

6 Importing "territory" file

[11]:
```
import numpy as np
import pandas as pd
territory=pd.read_excel(r"territory.xlsx")
territory
```

[11]:

	SalesTerritoryKey	Region	Country	Group	\
0	1	Northwest	United States	North America	
1	2	Northeast	United States	North America	
2	3	Central	United States	North America	
3	4	Southwest	United States	North America	
4	5	Southeast	United States	North America	
5	6	Canada	Canada	North America	
6	7	France	France	Europe	
7	8	Germany	Germany	Europe	
8	9	Australia	Australia	Pacific	
9	10	United Kingdom	United Kingdom	Europe	
10	11	NaN	NaN	NaN	

	RegionImage
0	http://www.avising.com/me/LearnPBI/DataSources…

```
1   http://www.avising.com/me/LearnPBI/DataSources…
2   http://www.avising.com/me/LearnPBI/DataSources…
3   http://www.avising.com/me/LearnPBI/DataSources…
4   http://www.avising.com/me/LearnPBI/DataSources…
5   http://www.avising.com/me/LearnPBI/DataSources…
6   http://www.avising.com/me/LearnPBI/DataSources…
7   http://www.avising.com/me/LearnPBI/DataSources…
8   http://www.avising.com/me/LearnPBI/DataSources…
9   http://www.avising.com/me/LearnPBI/DataSources…
10  http://www.avising.com/me/LearnPBI/DataSources…
```

[12]:
```
import pandas as pd
column_names= territory.columns
for col in column_names:
    print(col)
```

```
SalesTerritoryKey
Region
Country
Group
RegionImage
```

7 Importing "budget" file

[13]:
```
import numpy as np
import pandas as pd
budget=pd.read_excel(r"budget.xlsx")
budget
```

[13]:

	Category	Subcategory	ProductName	\
0	Accessories	Bike Racks	Hitch Rack - 4-Bike	
1	Accessories	Bike Stands	All-Purpose Bike Stand	
2	Accessories	Bottles and Cages	Water Bottle - 30 oz.	
3	Accessories	Cleaners	Bike Wash - Dissolver	
4	Accessories	Fenders	Fender Set - Mountain	
5	Accessories	Helmets	Sport-100 Helmet, Red	
6	Accessories	Hydration Packs	Hydration Pack - 70 oz.	
7	Accessories	Tires and Tubes	Patch Kit/8 Patches	
8	SubTotal Accessories	NaN	NaN	
9	Bikes	Mountain Bikes	Mountain-100 Silver, 38	
10	Bikes	Road Bikes	Road-150 Red, 62	
11	Bikes	Touring Bikes	Touring-2000 Blue, 60	
12	SubTotal Bikes	NaN	NaN	
13	Clothing	Caps	AWC Logo Cap	
14	Clothing	Gloves	Half-Finger Gloves, S	
15	Clothing	Jerseys	Long-Sleeve Logo Jersey, S	

16	Clothing	Shorts	Men's Sports Shorts, S	
17	Clothing	Socks	Mountain Bike Socks, M	
18	Clothing	Vests	Classic Vest, S	
19	SubTotal Clothing	NaN	NaN	
20	Grand Total	NaN	NaN	

	ProductKey	Jan, 2016	Feb, 2016	Mar, 2016	Apr, 2016	May, 2016	\
0	483.0	1131	2635	4134	2179	2637	
1	486.0	666	3695	2868	4862	3439	
2	477.0	1892	4727	3656	4449	4051	
3	484.0	160	713	555	656	369	
4	485.0	970	3014	2809	4259	3638	
5	212.0	5317	16221	16752	16552	17204	
6	487.0	809	2684	2917	3425	2716	
7	480.0	3554	18758	20905	18046	21680	
8	NaN	14499	52447	54596	54428	55734	
9	344.0	370105	326786	384811	439822	458523	
10	310.0	346295	289524	355097	346783	399691	
11	560.0	133631	165941	178287	265901	286630	
12	NaN	850031	782251	918195	1052506	1144844	
13	223.0	479	1695	1462	1079	1729	
14	462.0	598	2474	2957	2705	2819	
15	226.0	4087	11508	12872	11809	12789	
16	445.0	421	5723	7301	6335	5288	
17	218.0	24	244	432	547	385	
18	471.0	980	2008	1980	2312	2763	
19	NaN	6589	23652	27004	24787	25773	
20	NaN	871119	858350	999795	1131721	1226351	

	Jun, 2016	Jul, 2016	Aug, 2016	Sep, 2016	Oct, 2016	Nov, 2016	\
0	3279	2218	3287	3885	2484	5441	
1	4612	2774	3003	2401	4413	3881	
2	6257	4871	5231	5461	5529	5220	
3	582	777	777	239	496	686	
4	3721	4190	3618	3975	3892	4740	
5	25354	17584	20409	18268	20567	25571	
6	3260	3773	3523	4252	3111	4985	
7	22456	23995	22922	20950	21905	24019	
8	69521	60182	62770	59431	62397	74543	
9	619456	524348	647048	557368	615032	802831	
10	546092	441037	432400	468572	483913	558232	
11	445270	299106	407069	391580	481316	506411	
12	1610818	1264491	1486517	1417520	1580261	1867474	
13	2180	1588	2065	2013	2138	1830	
14	2966	2975	3264	2424	3181	3518	
15	18153	16846	13497	15988	15920	17426	
16	6829	4617	5384	6277	6337	6300	

```
17        372        839        487        425        335        660
18       2591       3379       3580       3600       4248       3685
19      33091      30244      28277      30727      32159      33419
20    1713430    1354917    1577564    1507678    1674817    1975436
```

```
    Dec, 2016  Grand Total
0        3551        36861
1        2143        38757
2        6025        57369
3         455         6465
4        4844        43670
5       22106       221905
6        4348        39803
7       23587       242777
8       67059       687607
9      788234      6534364
10     590261      5257897
11     494823      4055965
12    1873318     15848226
13       2113        20371
14       4084        33965
15      20043       170938
16       7641        68453
17        699         5449
18       3439        34565
19      38019       333741
20    1978396     16869574
```

[14]:
```
import pandas as pd
column_names=budget.columns
for col in column_names:
    print(col)
```

```
Category
Subcategory
ProductName
ProductKey
Jan, 2016
Feb, 2016
Mar, 2016
Apr, 2016
May, 2016
Jun, 2016
Jul, 2016
Aug, 2016
Sep, 2016
Oct, 2016
```

```
Nov, 2016
Dec, 2016
Grand Total
```

8 Merging data for analysis

[15]:
```
temp_data = pd.merge(sales, product, on='ProductKey', how='inner')
df = pd.merge(temp_data, customer, on='CustomerKey', how='inner')
print(df)
df.to_excel("final_merged_data.xlsx", index=False)
```

```
       ProductKey  OrderDate   ShipDate  CustomerKey  PromotionKey  \
0             310 2014-01-01 2014-01-08        21768             1
1             600 2016-04-16 2016-04-23        21768             1
2             310 2014-01-02 2014-01-09        16624             1
3             537 2016-12-25 2017-01-02        16624             1
4             528 2016-12-25 2017-01-02        16624             1
…             …          …          …            …             …
58184         568 2016-02-26 2016-03-02        25759            13
58185         568 2016-07-25 2016-08-01        29081             2
58186         567 2016-02-04 2016-02-11        25749            13
58187         567 2016-06-10 2016-06-17        28383             1
58188         567 2016-12-04 2016-12-11        29374             1

       SalesTerritoryKey SalesOrderNumber  SalesOrderLineNumber  \
0                      6          SO43697                     1
1                      6          SO56212                     1
2                      9          SO43703                     1
3                      9          SO73763                     3
4                      9          SO73763                     2
…                    …                …                     …
58184                  7          SO53173                     1
58185                  9          SO62749                     1
58186                  7          SO52059                     1
58187                  8          SO59750                     1
58188                  8          SO72166                     1

       OrderQuantity  UnitPrice  …  YearlyIncome  TotalChildren  \
0                  2  1789.1350  …         70000              5
1                  1   539.9900  …         70000              5
2                  4   894.5675  …         90000              0
3                  1    35.0000  …         90000              0
4                  1     4.9900  …         90000              0
…                …          …  …           …              …
58184              1   742.3500  …         20000              1
58185              2   371.1750  …         20000              2
58186              4   185.5875  …         30000              1
```

```
58187              1   742.3500  …         20000              2
58188              1   742.3500  …         20000              2

       NumberChildrenAtHome            Education    Occupation  \
0                         0            Bachelors    Management
1                         0            Bachelors    Management
2                         0            Bachelors  Professional
3                         0            Bachelors  Professional
4                         0            Bachelors  Professional
…                       …                    …           …
58184                     1      Partial College        Manual
58185                     1  Partial High School      Clerical
58186                     0            Bachelors      Clerical
58187                     2          High School        Manual
58188                     2          High School        Manual

       HouseOwnerFlag  NumberCarsOwned                    AddressLine1  \
0                   1                3                601 Asilomar Dr.
1                   1                3                601 Asilomar Dr.
2                   0                3              3541 Corte Poquito
3                   0                3              3541 Corte Poquito
4                   0                3              3541 Corte Poquito
…                 …                …                               …
58184               1                1            370, rue des Rosiers
58185               1                2                  5086 Rampo Ct.
58186               0                0            1510, rue des Berges
58187               1                1  Buergermeister-ulrich-str 7500
58188               1                1                 Lieblingsweg 333

      DateFirstPurchase CommuteDistance
0            2014-01-01       10+ Miles
1            2014-01-01       10+ Miles
2            2014-01-02       10+ Miles
3            2014-01-02       10+ Miles
4            2014-01-02       10+ Miles
…                   …               …
58184        2016-02-26        2-5 Miles
58185        2016-07-25       5-10 Miles
58186        2016-02-04        0-1 Miles
58187        2016-06-10        0-1 Miles
58188        2016-12-04        0-1 Miles

[58189 rows x 46 columns]
```

[16]:
```
print(f"Number of Rows: {df.shape[0]}")
print(f"Number of Columns: {df.shape[1]} \n")
```

```
Number of Rows: 58189
```

```
Number of Columns: 46
```

9 missing data handling

[17]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 58189 entries, 0 to 58188
Data columns (total 46 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           58189 non-null  float64
 14  Unnamed: 14           58189 non-null  float64
 15  Unnamed: 15           58189 non-null  float64
 16  StandardCost_x        58189 non-null  float64
 17  List Price            58189 non-null  float64
 18  ProductName           58189 non-null  object
 19  SubCategory           58189 non-null  object
 20  Category              58189 non-null  object
 21  StandardCost_y        58189 non-null  float64
 22  Color                 30747 non-null  object
 23  ListPrice             58189 non-null  float64
 24  DaysToManufacture     58189 non-null  int64
 25  ProductLine           58189 non-null  object
 26  ModelName             58189 non-null  object
 27  Photo                 58189 non-null  object
 28  ProductDescription    58189 non-null  object
 29  StartDate             58189 non-null  datetime64[ns]
 30  FirstName             58189 non-null  object
 31  LastName              58189 non-null  object
 32  FullName              58189 non-null  object
 33  BirthDate             58189 non-null  datetime64[ns]
 34  MaritalStatus         58189 non-null  object
```

```
 35  Gender                 58189 non-null  object
 36  YearlyIncome           58189 non-null  int64
 37  TotalChildren          58189 non-null  int64
 38  NumberChildrenAtHome   58189 non-null  int64
 39  Education              58189 non-null  object
 40  Occupation             58189 non-null  object
 41  HouseOwnerFlag         58189 non-null  int64
 42  NumberCarsOwned        58189 non-null  int64
 43  AddressLine1           58189 non-null  object
 44  DateFirstPurchase      58189 non-null  datetime64[ns]
 45  CommuteDistance        58189 non-null  object
dtypes: datetime64[ns](5), float64(11), int64(12), object(18)
memory usage: 20.9+ MB
```

[18]:
```
df.isnull().sum()
```

[18]:
```
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: 13               0
Unnamed: 14               0
Unnamed: 15               0
StandardCost_x            0
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
dtype: int64
```

[19]:
```
df.dropna(inplace=True)
```

[20]:
```
df.isnull().sum()
```

[20]:
```
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: 13             0
Unnamed: 14             0
Unnamed: 15             0
StandardCost_x          0
List Price              0
ProductName             0
SubCategory             0
Category                0
StandardCost_y          0
Color                   0
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
dtype: int64
```

[21]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30747 entries, 0 to 58188
Data columns (total 46 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ProductKey            30747 non-null  int64
 1   OrderDate             30747 non-null  datetime64[ns]
 2   ShipDate              30747 non-null  datetime64[ns]
 3   CustomerKey           30747 non-null  int64
 4   PromotionKey          30747 non-null  int64
 5   SalesTerritoryKey     30747 non-null  int64
 6   SalesOrderNumber      30747 non-null  object
 7   SalesOrderLineNumber  30747 non-null  int64
 8   OrderQuantity         30747 non-null  int64
 9   UnitPrice             30747 non-null  float64
 10  TotalProductCost      30747 non-null  float64
 11  SalesAmount           30747 non-null  float64
 12  TaxAmt                30747 non-null  float64
 13  Unnamed: 13           30747 non-null  float64
 14  Unnamed: 14           30747 non-null  float64
 15  Unnamed: 15           30747 non-null  float64
 16  StandardCost_x        30747 non-null  float64
 17  List Price            30747 non-null  float64
```

```
 18  ProductName            30747 non-null  object
 19  SubCategory            30747 non-null  object
 20  Category               30747 non-null  object
 21  StandardCost_y         30747 non-null  float64
 22  Color                  30747 non-null  object
 23  ListPrice              30747 non-null  float64
 24  DaysToManufacture      30747 non-null  int64
 25  ProductLine            30747 non-null  object
 26  ModelName              30747 non-null  object
 27  Photo                  30747 non-null  object
 28  ProductDescription     30747 non-null  object
 29  StartDate              30747 non-null  datetime64[ns]
 30  FirstName              30747 non-null  object
 31  LastName               30747 non-null  object
 32  FullName               30747 non-null  object
 33  BirthDate              30747 non-null  datetime64[ns]
 34  MaritalStatus          30747 non-null  object
 35  Gender                 30747 non-null  object
 36  YearlyIncome           30747 non-null  int64
 37  TotalChildren          30747 non-null  int64
 38  NumberChildrenAtHome   30747 non-null  int64
 39  Education              30747 non-null  object
 40  Occupation             30747 non-null  object
 41  HouseOwnerFlag         30747 non-null  int64
 42  NumberCarsOwned        30747 non-null  int64
 43  AddressLine1           30747 non-null  object
 44  DateFirstPurchase      30747 non-null  datetime64[ns]
 45  CommuteDistance        30747 non-null  object
dtypes: datetime64[ns](5), float64(11), int64(12), object(18)
memory usage: 11.0+ MB
```

[22]:
```
temp_data = pd.merge(sales, product, on='ProductKey', how='inner')
df = pd.merge(temp_data, customer, on='CustomerKey', how='inner')
df
```

[22]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	\
0	310	2014-01-01	2014-01-08	21768	1	
1	600	2016-04-16	2016-04-23	21768	1	
2	310	2014-01-02	2014-01-09	16624	1	
3	537	2016-12-25	2017-01-02	16624	1	
4	528	2016-12-25	2017-01-02	16624	1	
…	…	…	…	…	…	
58184	568	2016-02-26	2016-03-02	25759	13	
58185	568	2016-07-25	2016-08-01	29081	2	
58186	567	2016-02-04	2016-02-11	25749	13	
58187	567	2016-06-10	2016-06-17	28383	1	
58188	567	2016-12-04	2016-12-11	29374	1	

```
       SalesTerritoryKey SalesOrderNumber  SalesOrderLineNumber  \
0                      6          SO43697                     1
1                      6          SO56212                     1
2                      9          SO43703                     1
3                      9          SO73763                     3
4                      9          SO73763                     2
…                    …                …                     …
58184                  7          SO53173                     1
58185                  9          SO62749                     1
58186                  7          SO52059                     1
58187                  8          SO59750                     1
58188                  8          SO72166                     1

       OrderQuantity  UnitPrice  …  YearlyIncome  TotalChildren  \
0                  2  1789.1350  …         70000              5
1                  1   539.9900  …         70000              5
2                  4   894.5675  …         90000              0
3                  1    35.0000  …         90000              0
4                  1     4.9900  …         90000              0
…                …          …  …            …              …
58184              1   742.3500  …         20000              1
58185              2   371.1750  …         20000              2
58186              4   185.5875  …         30000              1
58187              1   742.3500  …         20000              2
58188              1   742.3500  …         20000              2

       NumberChildrenAtHome            Education    Occupation  \
0                         0            Bachelors    Management
1                         0            Bachelors    Management
2                         0            Bachelors  Professional
3                         0            Bachelors  Professional
4                         0            Bachelors  Professional
…                       …                    …           …
58184                     1      Partial College        Manual
58185                     1  Partial High School      Clerical
58186                     0            Bachelors      Clerical
58187                     2          High School        Manual
58188                     2          High School        Manual

       HouseOwnerFlag  NumberCarsOwned                  AddressLine1  \
0                   1                3               601 Asilomar Dr.
1                   1                3               601 Asilomar Dr.
2                   0                3             3541 Corte Poquito
3                   0                3             3541 Corte Poquito
4                   0                3             3541 Corte Poquito
…                 …                …                             …
```

```
58184               1                1           370, rue des Rosiers
58185               1                2                 5086 Rampo Ct.
58186               0                0           1510, rue des Berges
58187               1                1  Buergermeister-ulrich-str 7500
58188               1                1                Lieblingsweg 333

      DateFirstPurchase CommuteDistance
0            2014-01-01       10+ Miles
1            2014-01-01       10+ Miles
2            2014-01-02       10+ Miles
3            2014-01-02       10+ Miles
4            2014-01-02       10+ Miles
…                   …               …
58184        2016-02-26        2-5 Miles
58185        2016-07-25       5-10 Miles
58186        2016-02-04        0-1 Miles
58187        2016-06-10        0-1 Miles
58188        2016-12-04        0-1 Miles

[58189 rows x 46 columns]
```

[23]:
```
import pandas as pd
columns_df = df.columns
for col in columns_df:
    print(col)
```

```
ProductKey
OrderDate
ShipDate
CustomerKey
PromotionKey
SalesTerritoryKey
SalesOrderNumber
SalesOrderLineNumber
OrderQuantity
UnitPrice
TotalProductCost
SalesAmount
TaxAmt
Unnamed: 13
Unnamed: 14
Unnamed: 15
StandardCost_x
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
```

[24]:
```
df['sale_year'] = df['OrderDate'].dt.year

df['sale_month'] = df['OrderDate'].dt.month

df['sale_day'] = df['OrderDate'].dt.day

df['sale_week'] = df['OrderDate'].dt.dayofweek

df['sale_day_name'] = df['OrderDate'].dt.day_name()

df['year_month'] = df['OrderDate'].apply(lambda x:x.strftime('%Y-%m'))

df['total_Invoice_amount'] = df['SalesAmount'] + df['TaxAmt']

df['profit'] = (df['UnitPrice']*df['OrderQuantity']) - df['TotalProductCost']

df['ProductName'] = df['ProductName'].str.replace(',','-')

df['Age'] = df['OrderDate'].dt.year - df['BirthDate'].dt.year
```

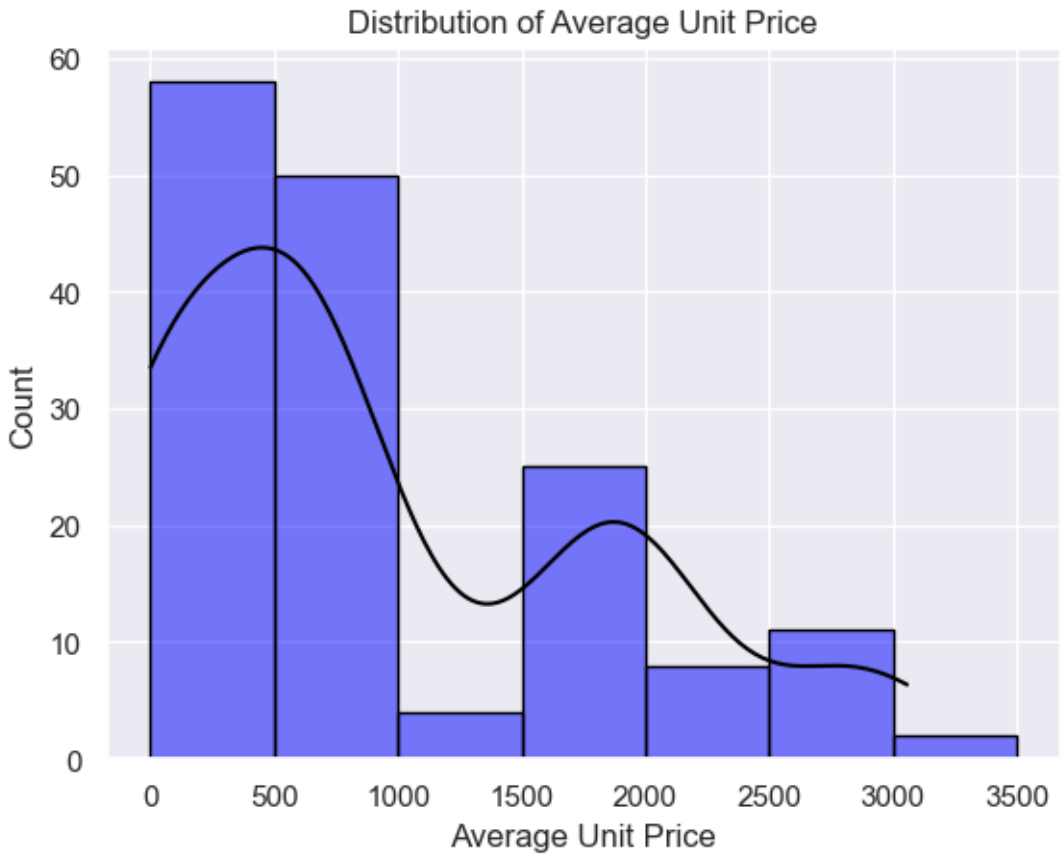
[25]:
```
categories = df['Category'].unique().tolist()


print("Unique Categories:")
print(categories)
```

```
Unique Categories:
['Bikes', 'Accessories', 'Clothing']
```

[26]:
```
subcategories = df['SubCategory'].unique().tolist()
print("\nUnique SubCategories:")
print(subcategories)
```

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

[27]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

Avg_unit_price = df.groupby(['ProductKey'])['UnitPrice'].mean()
ax = sns.histplot(Avg_unit_price, bins=np.arange(0, Avg_unit_price.max() + 500,␣
 ↪500), kde=True, color='blue', edgecolor='black')
ax.lines[0].set_color('black')  # Set KDE line color to black
ax.set(title='Distribution of Average Unit Price',
       xlabel='Average Unit Price')

plt.show()
```



- Here the products which is having average price b/w $0 to $500 is having more sales .
- Here the products which is having average price b/w $500 to $1000 is having second highest sales.

```
# here we conclude that the products which is having avg unit price of "$0 to $1000" is ha
```

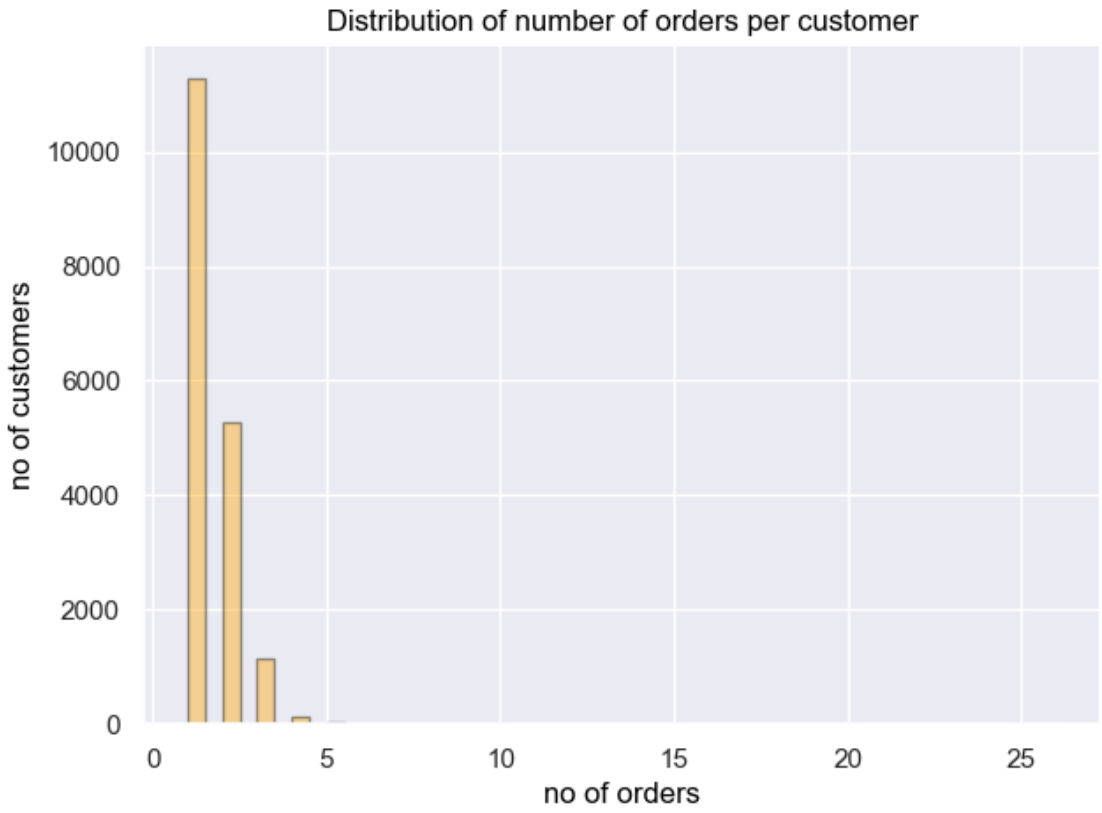
10 repeated customers rate percentage(%)

[28]:
```
import pandas as pd
import numpy as np

n_orders = df.groupby('CustomerKey')['SalesOrderNumber'].nunique()

n_repeated_customers = np.sum(n_orders > 1)

total_customers = df['CustomerKey'].nunique()

repeated_customers_perc = (n_repeated_customers / total_customers) * 100
```

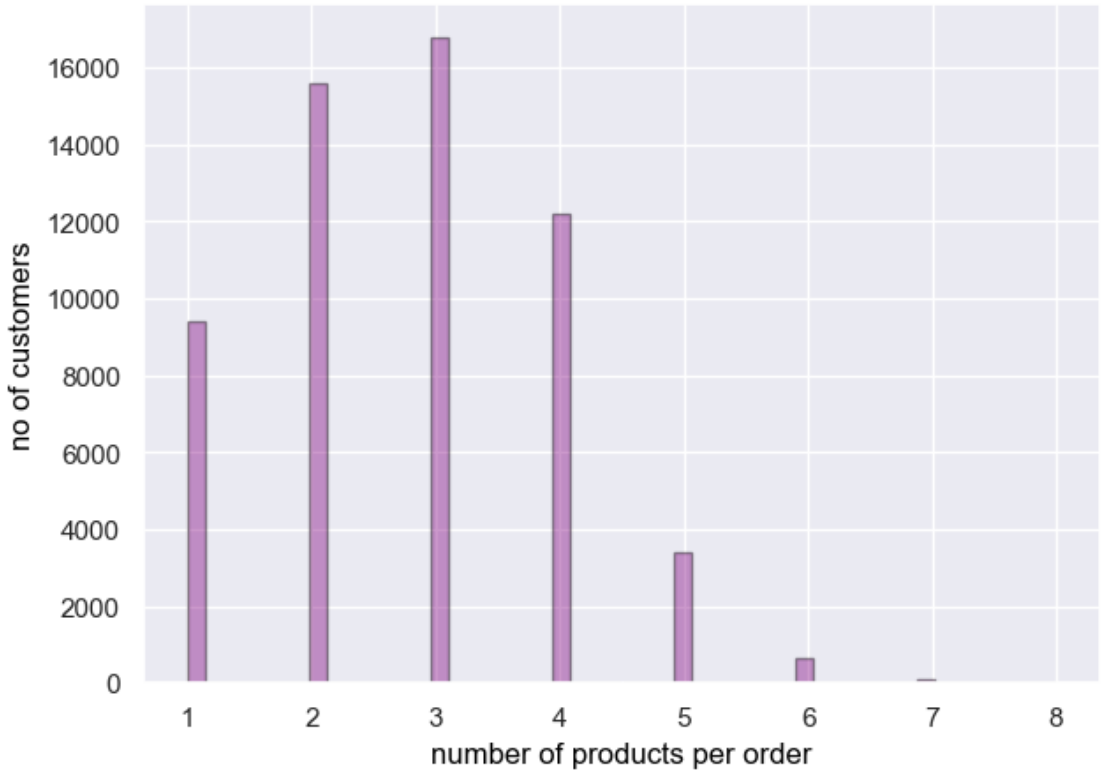
```
print(f"repeated customers rate percentage(%):{repeated_customers_perc:.2f}%")
```

```
repeated customers rate percentage(%):36.97%
```

[29]:
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(7, 5))
ax = sns.distplot(n_orders, kde=False, color='orange', hist_kws={'edgecolor':␣
 ↪'black'})

ax.set_ylabel('no of customers', color='black')

ax.set_title('Distribution of number of orders per customer', color='black')
ax.set_xlabel('no of orders', color='black')

plt.show()
```



[30]:
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(7, 5))
n_salesordernumber = df.groupby(['SalesOrderNumber'])['SalesOrderLineNumber'].
 ↪transform('max')
ax = sns.distplot(n_salesordernumber, kde=False, color='purple',␣
 ↪hist_kws={'edgecolor': 'black'})

ax.set_ylabel('no of customers', color='black')
ax.set_xlabel('number of products per order', color='black')

plt.show()
```



* Customers are ordering 2 to 3 products in each order

10.1 Sales Order Quantity distribution

[31]:
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(7, 5))
```

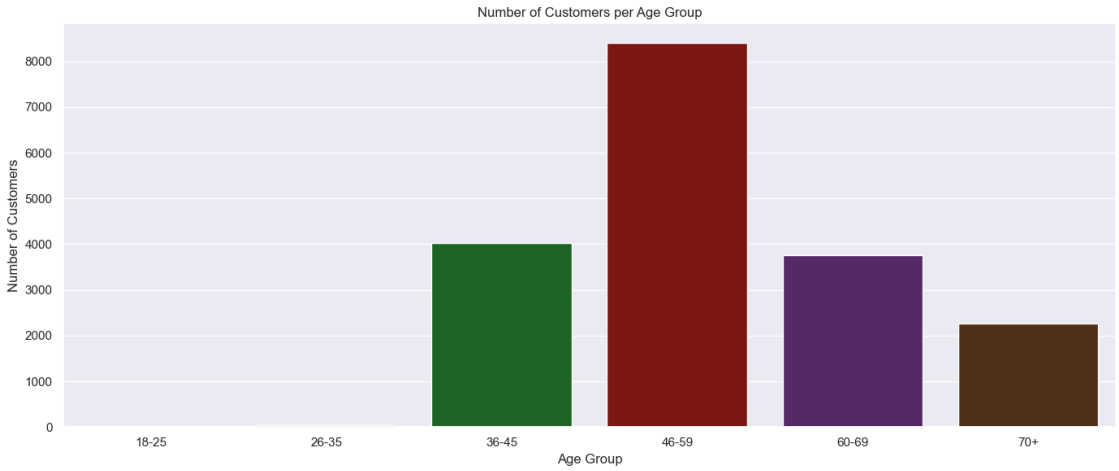
```
n_order_quantity = df.groupby(['SalesOrderNumber'])['OrderQuantity'].sum()

ax = sns.distplot(n_order_quantity, kde=True, color='orange',␣
 ↪hist_kws={'edgecolor': 'black'})

ax.set_ylabel('no of customers', color='black')

ax.set_title('Distribution of order_quantity', color='black')
ax.set_xlabel('no of order_quantity', color='black')

plt.show()
```

- maximum quantity ordered for a product is 3 or 2.

```
# we can finally infer that max quantity ordered for a product is " ur
```

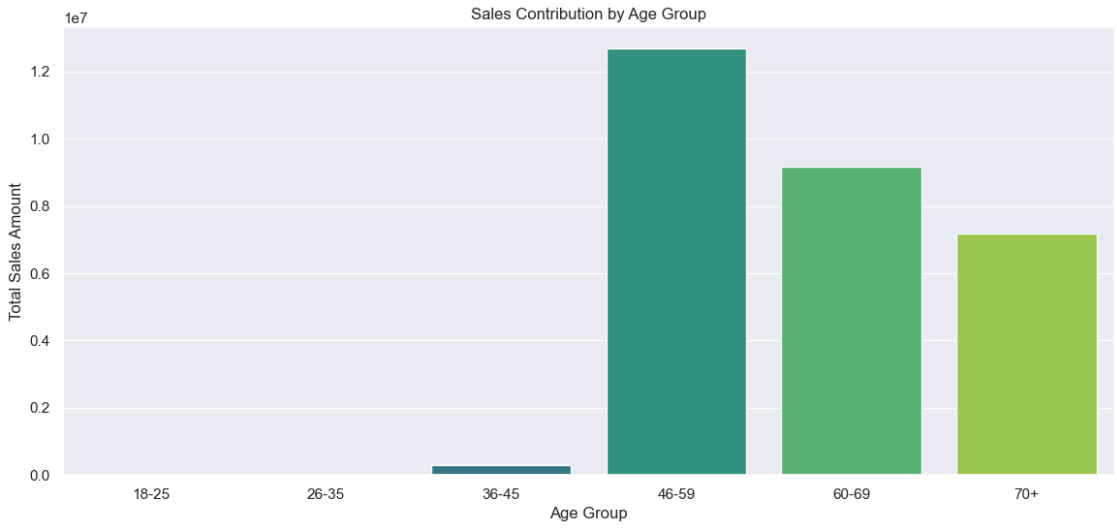
11 Customer data analysis

[32]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df['BirthDate'] = pd.to_datetime(df['BirthDate'])
df['Age'] = df['OrderDate'].dt.year - df['BirthDate'].dt.year
age_bins = [18, 25, 35, 45, 59, 69, 100]
age_labels = ['18-25', '26-35', '36-45', '46-59', '60-69', '70+']
df['AgeGroup'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels,␣
 ↪right=False)
df = df.dropna(subset=['AgeGroup'])
age_group_counts = df.groupby('AgeGroup')['CustomerKey'].nunique()
plt.figure(figsize=(14, 6))
sns.barplot(x=age_group_counts.index, y=age_group_counts.values, palette='dark')
plt.title('Number of Customers per Age Group')
plt.xlabel('Age Group')
plt.ylabel('Number of Customers')



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



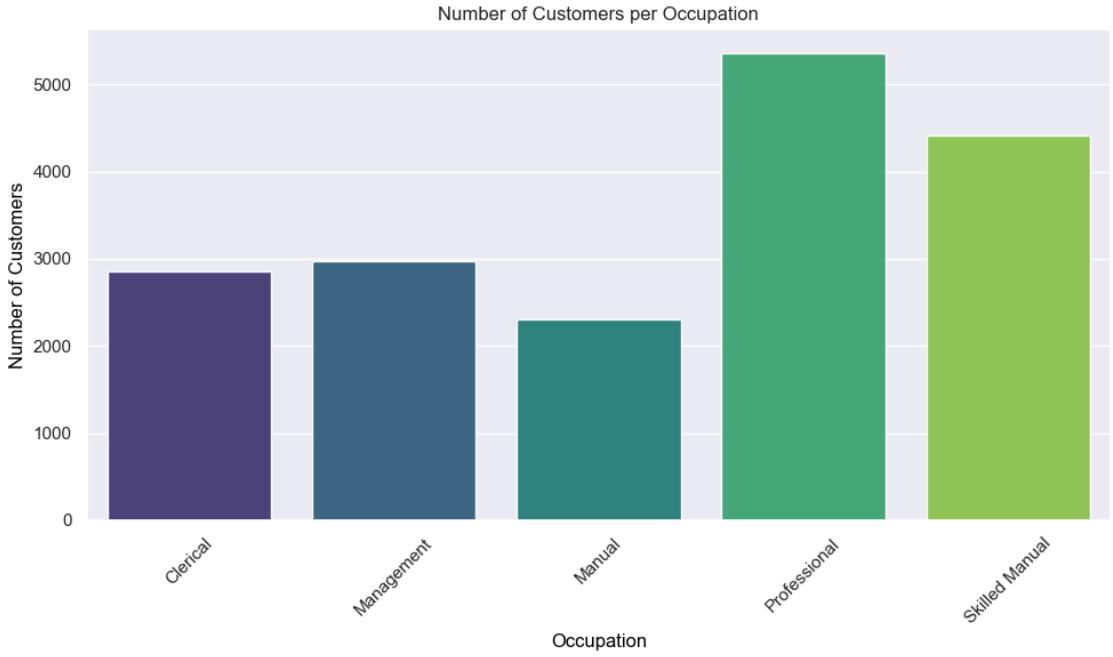
[33]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df['BirthDate'] = pd.to_datetime(df['BirthDate'])

df['Age'] = pd.datetime.now().year - df['BirthDate'].dt.year
```

```
bins = [18, 25, 35, 45, 59, 69, 100]
labels = ['18-25', '26-35', '36-45', '46-59', '60-69', '70+']
df['AgeGroup'] = pd.cut(df['Age'], bins, labels=labels, right=False)

age_group_sales = df.groupby('AgeGroup')['SalesAmount'].sum()
plt.figure(figsize=(14, 6))
sns.barplot(x=age_group_sales.index, y=age_group_sales.values,␣
 ↪palette='viridis')
plt.title('Sales Contribution by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Total Sales Amount')
plt.show()
```

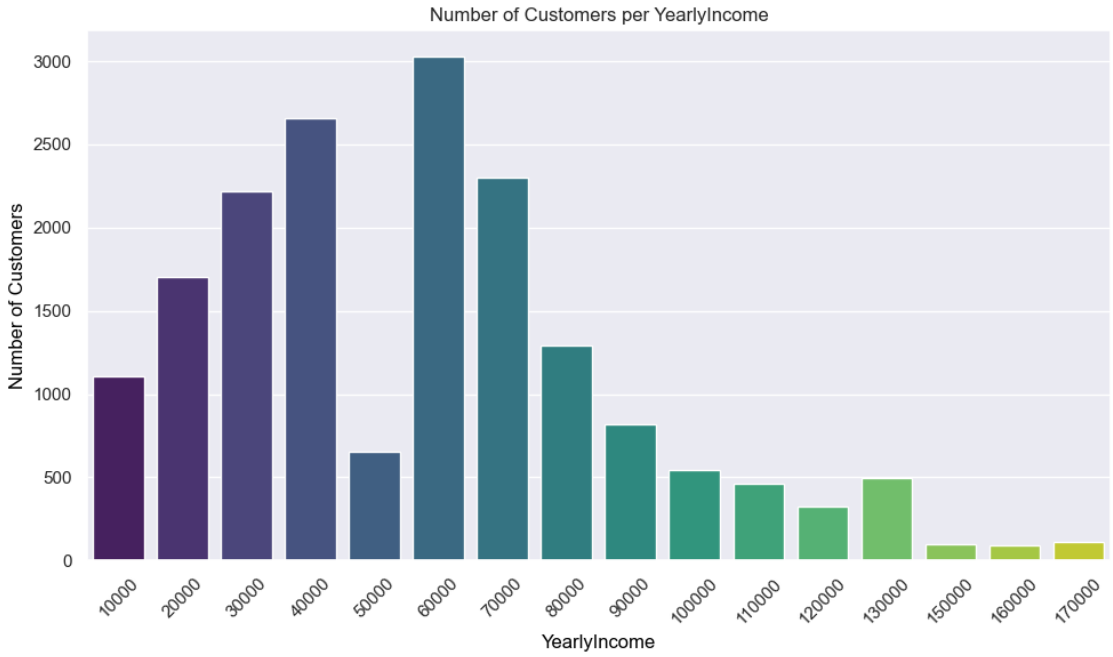


[34]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

occupation_counts = df.groupby('Occupation')['CustomerKey'].nunique().
 ↪reset_index()
occupation_counts.columns = ['Occupation', 'CustomerCount']

plt.figure(figsize=(10, 6))
ax = sns.barplot(x='Occupation', y='CustomerCount', data=occupation_counts,␣
 ↪palette='viridis')
ax.set_ylabel('Number of Customers', color='black')
ax.set_xlabel('Occupation', color='black')
ax.set_title('Number of Customers per Occupation')
```

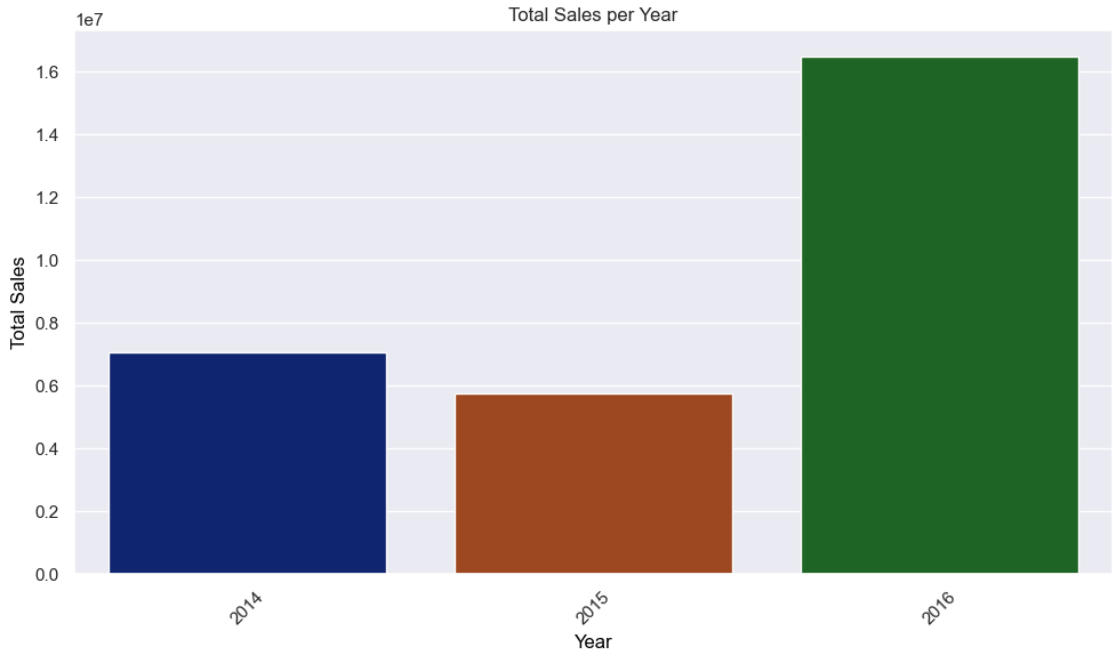
```
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[35]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

occupation_counts = df.groupby('YearlyIncome')['CustomerKey'].nunique().
 ↪reset_index()
occupation_counts.columns = ['YearlyIncome', 'CustomerCount']

plt.figure(figsize=(10, 6))
ax = sns.barplot(x='YearlyIncome', y='CustomerCount', data=occupation_counts,␣
 ↪palette='viridis')
ax.set_ylabel('Number of Customers', color='black')
ax.set_xlabel('YearlyIncome', color='black')
ax.set_title('Number of Customers per YearlyIncome')

plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

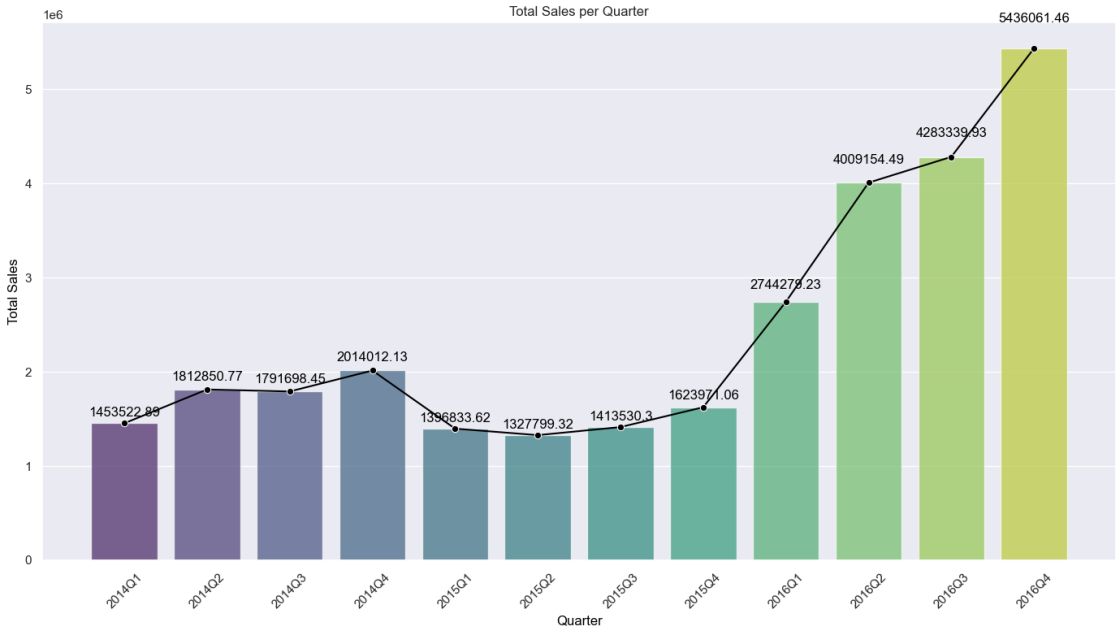


[36]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd


df['OrderYear'] = df['OrderDate'].dt.year

yearly_sales = df.groupby('OrderYear')['SalesAmount'].sum().reset_index()
yearly_sales.columns = ['OrderYear', 'TotalSales']

plt.figure(figsize=(10, 6))
ax = sns.barplot(x='OrderYear', y='TotalSales', data=yearly_sales,␣
 ↪palette='dark')
ax.set_ylabel('Total Sales', color='black')
ax.set_xlabel('Year', color='black')
ax.set_title('Total Sales per Year')

plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[37]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd


df['OrderYear'] = df['OrderDate'].dt.year
df['OrderQuarter'] = df['OrderDate'].dt.to_period('Q')

quarterly_sales = df.groupby('OrderQuarter')['SalesAmount'].sum().reset_index()
quarterly_sales.columns = ['OrderQuarter', 'TotalSales']

quarterly_sales['OrderQuarter'] = quarterly_sales['OrderQuarter'].astype(str)

plt.figure(figsize=(14, 8))
ax = sns.barplot(x='OrderQuarter', y='TotalSales', data=quarterly_sales,␣
 ↪palette='viridis', alpha=0.7)

sns.lineplot(x='OrderQuarter', y='TotalSales', data=quarterly_sales,␣
 ↪color='black', marker='o', sort=False, ax=ax)

for i, row in quarterly_sales.iterrows():
    ax.text(i, row['TotalSales'] + 0.05 * row['TotalSales'],␣
 ↪round(row['TotalSales'], 2), color='black', ha="center")

ax.set_ylabel('Total Sales', color='black')
```

```
ax.set_xlabel('Quarter', color='black')
ax.set_title('Total Sales per Quarter')
plt.xticks(rotation=45)
plt.tight_layout()

plt.show()
```



[38]:
```
top_selling_product = df.groupby(['Category', 'SubCategory',␣
↪'ProductName'])['OrderQuantity'].sum().nlargest(5).to_frame()
top_selling_product
```

[38]:

			OrderQuantity
Category	SubCategory	ProductName	
Accessories	Bottles and Cages	Water Bottle - 30 oz.	6370
	Tires and Tubes	Patch Kit/8 Patches	4697
		Mountain Tire Tube	4547
		Road Tire Tube	3536
	Helmets	Sport-100 Helmet- Red	3394

[39]:
```
cust_edu = df[(df['Education']=='Partial High␣
↪School')|(df['Education']=='Bachelors')].
↪groupby('Education')['YearlyIncome'].mean().to_frame()
cust_edu
```

[39]:

	YearlyIncome
Education	

```
Bachelors            65683.277459
Partial High School  43485.421767
```

[40]:
```
cust_edu.reset_index(inplace=True)
fig = px.bar(cust_edu, x='Education', y='YearlyIncome')
fig.update_layout(
    autosize=False,
    width=300,
    height=300,
    margin=dict(
        l=25,
        r=25,
        b=10,
        t=10,
    ))
fig.show()
```

[41]:
```
import pandas as pd

# Grouping data by sale_year, Category, and SubCategory and summing the␣
 ↪OrderQuantity
cat_subcat_qty = df.groupby(['sale_year', 'Category',␣
 ↪'SubCategory'])['OrderQuantity'].sum().to_frame()

# Sorting the resulting DataFrame by sale_year and Category in ascending order
cat_subcat_qty = cat_subcat_qty.sort_values(['sale_year', 'Category'],␣
 ↪ascending=True)

# Styling the DataFrame to display bar charts within cells for the␣
 ↪OrderQuantity column
cat_subcat_qty.style.bar(subset=['OrderQuantity'], color='red')
```

[41]:
```
<pandas.io.formats.style.Styler at 0x16b86186560>
```

[42]:
```
import pandas as pd

# Load the existing final merged data
final_merged_data = pd.read_excel("final_merged_data.xlsx")

# Load the territory data
territory_data = pd.read_excel("territory.xlsx")

# Merge the final_merged_data with territory_data on the common key␣
 ↪'SalesTerritoryKey'
final_merged_data = pd.merge(final_merged_data, territory_data,␣
 ↪on='SalesTerritoryKey', how='inner')
```

```
# Save the updated final_merged_data back to an Excel file
final_merged_data.to_excel("final_merged_data_updated.xlsx", index=False)

print("Data merged and saved to final_merged_data_updated.xlsx")
```

```
Data merged and saved to final_merged_data_updated.xlsx
```

[43]:
```
df=pd.read_excel(r"final_merged_data_updated.xlsx")
df
```

[43]:

	ProductKey	OrderDate	ShipDate	CustomerKey	PromotionKey	\
0	310	2014-01-01	2014-01-08	21768	1	
1	600	2016-04-16	2016-04-23	21768	1	
2	310	2014-01-30	2014-02-06	21727	1	
3	479	2016-11-29	2016-12-05	21727	1	
4	477	2016-11-29	2016-12-05	21727	1	
…	…	…	…	…	…	
58184	528	2016-11-07	2016-11-14	13145	1	
58185	361	2016-11-07	2016-11-14	13145	1	
58186	480	2016-11-07	2016-11-14	13145	1	
58187	530	2016-02-06	2016-02-13	27040	1	
58188	480	2016-02-06	2016-02-13	27040	2	

	SalesTerritoryKey	SalesOrderNumber	SalesOrderLineNumber	\
0	6	SO43697	1	
1	6	SO56212	1	
2	6	SO43833	1	
3	6	SO71614	2	
4	6	SO71614	3	
…	…	…	…	
58184	2	SO70064	2	
58185	2	SO70064	1	
58186	2	SO70064	4	
58187	2	SO52124	1	
58188	2	SO52124	2	

	OrderQuantity	UnitPrice	…	Occupation	HouseOwnerFlag	\
0	2	1789.1350	…	Management	1	
1	1	539.9900	…	Management	1	
2	4	894.5675	…	Skilled Manual	1	
3	1	8.9900	…	Skilled Manual	1	
4	1	4.9900	…	Skilled Manual	1	
…	…	…	…	…	…	
58184	1	4.9900	…	Skilled Manual	1	
58185	1	2294.9900	…	Skilled Manual	1	
58186	1	2.2900	…	Skilled Manual	1	

```
58187              1     4.9900  …        Clerical               1
58188              1     2.2900  …        Clerical               1

       NumberCarsOwned        AddressLine1  DateFirstPurchase  \
0                    3    601 Asilomar Dr.         2014-01-01
1                    3    601 Asilomar Dr.         2014-01-01
2                    0       4082 Shell Ct         2014-01-30
3                    0       4082 Shell Ct         2014-01-30
4                    0       4082 Shell Ct         2014-01-30
…                  …                  …                  …
58184                2    7779 Merry Drive         2016-11-07
58185                2    7779 Merry Drive         2016-11-07
58186                2    7779 Merry Drive         2016-11-07
58187                2  371 Westwood Court         2016-02-06
58188                2  371 Westwood Court         2016-02-06

       CommuteDistance     Region        Country          Group  \
0            10+ Miles     Canada         Canada  North America
1            10+ Miles     Canada         Canada  North America
2            1-2 Miles     Canada         Canada  North America
3            1-2 Miles     Canada         Canada  North America
4            1-2 Miles     Canada         Canada  North America
…                  …        …            …              …
58184       5-10 Miles  Northeast  United States  North America
58185       5-10 Miles  Northeast  United States  North America
58186       5-10 Miles  Northeast  United States  North America
58187        1-2 Miles  Northeast  United States  North America
58188        1-2 Miles  Northeast  United States  North America

                                              RegionImage
0      http://www.avising.com/me/LearnPBI/DataSources…
1      http://www.avising.com/me/LearnPBI/DataSources…
2      http://www.avising.com/me/LearnPBI/DataSources…
3      http://www.avising.com/me/LearnPBI/DataSources…
4      http://www.avising.com/me/LearnPBI/DataSources…
…                                                    …
58184  http://www.avising.com/me/LearnPBI/DataSources…
58185  http://www.avising.com/me/LearnPBI/DataSources…
58186  http://www.avising.com/me/LearnPBI/DataSources…
58187  http://www.avising.com/me/LearnPBI/DataSources…
58188  http://www.avising.com/me/LearnPBI/DataSources…

[58189 rows x 50 columns]
```

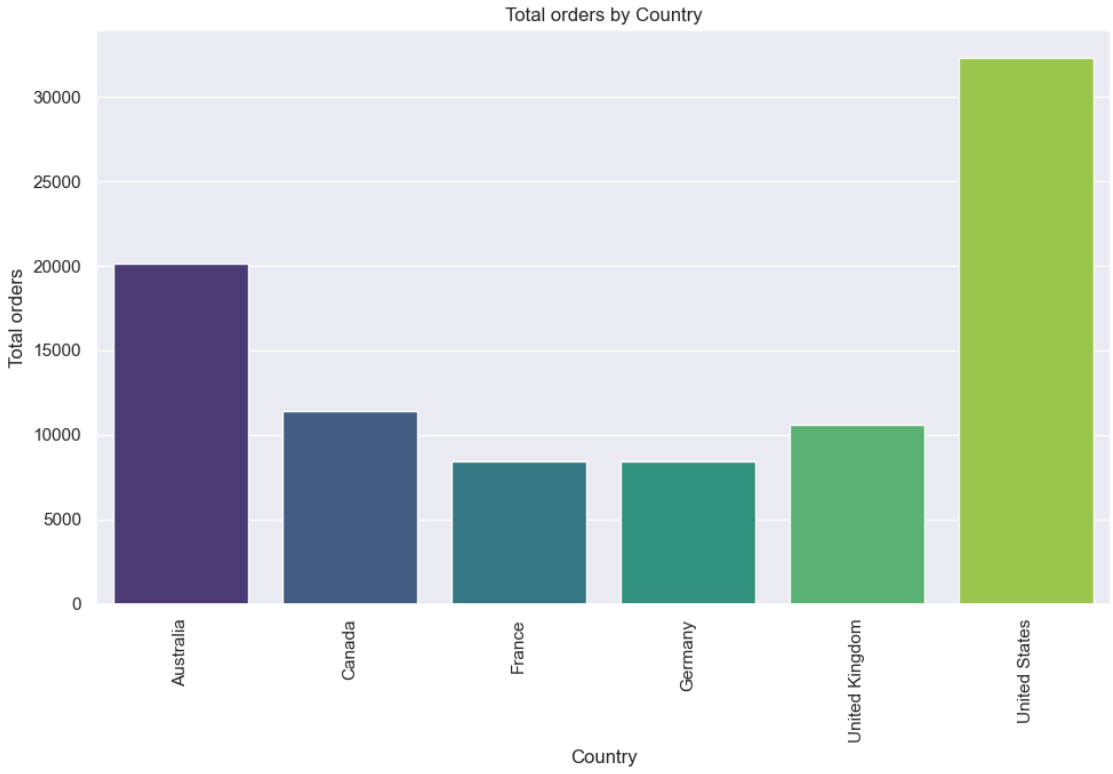
[44]:
```
import pandas as pd
col= df.columns
for i in col:
```

```
    print(i)
```

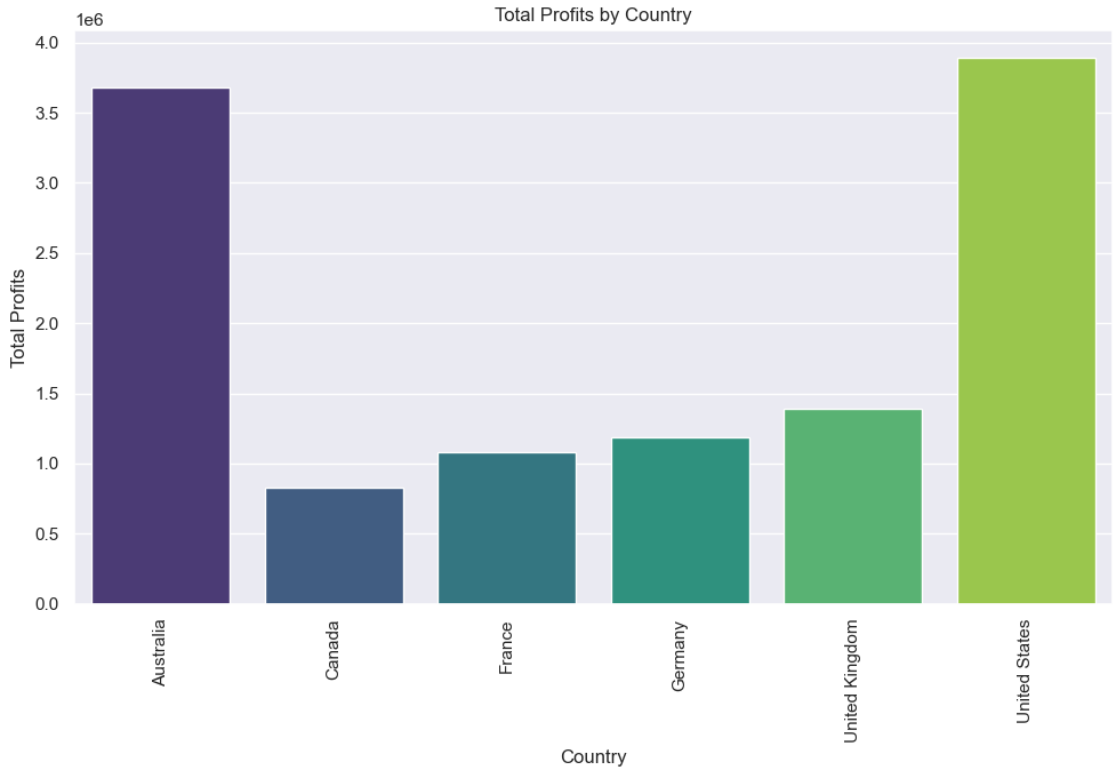
```
ProductKey
OrderDate
ShipDate
CustomerKey
PromotionKey
SalesTerritoryKey
SalesOrderNumber
SalesOrderLineNumber
OrderQuantity
UnitPrice
TotalProductCost
SalesAmount
TaxAmt
Unnamed: 13
Unnamed: 14
Unnamed: 15
StandardCost_x
List Price
ProductName
SubCategory
Category
StandardCost_y
Color
ListPrice
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
```

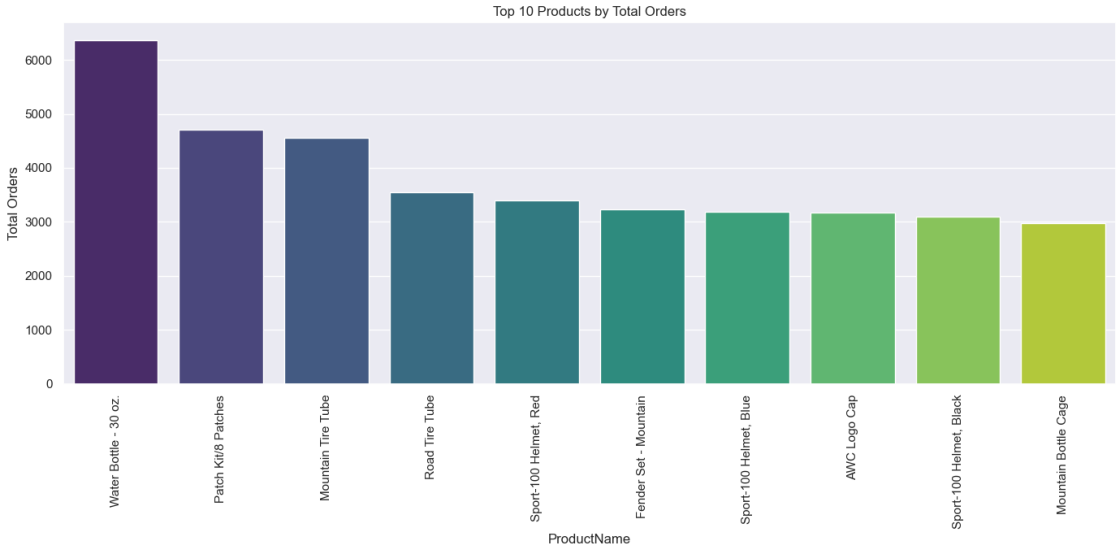
[45]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


country_orders = df.groupby('Country')['OrderQuantity'].sum().reset_index()

plt.figure(figsize=(10, 7))
ax = sns.barplot(x='Country', y='OrderQuantity', data=country_orders,␣
 ↪palette='viridis')

ax.set_title('Total orders by Country')
ax.set_xlabel('Country')
ax.set_ylabel('Total orders')
plt.xticks(rotation=90)

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



[46]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df['Profit'] = (df['SalesAmount'] - df['TotalProductCost'])
country_profits = df.groupby('Country')['Profit'].sum().reset_index()

plt.figure(figsize=(10, 7))
ax = sns.barplot(x='Country', y='Profit', data=country_profits,␣
 ↪palette='viridis')

ax.set_title('Total Profits by Country')
ax.set_xlabel('Country')
ax.set_ylabel('Total Profits')
plt.xticks(rotation=90)

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

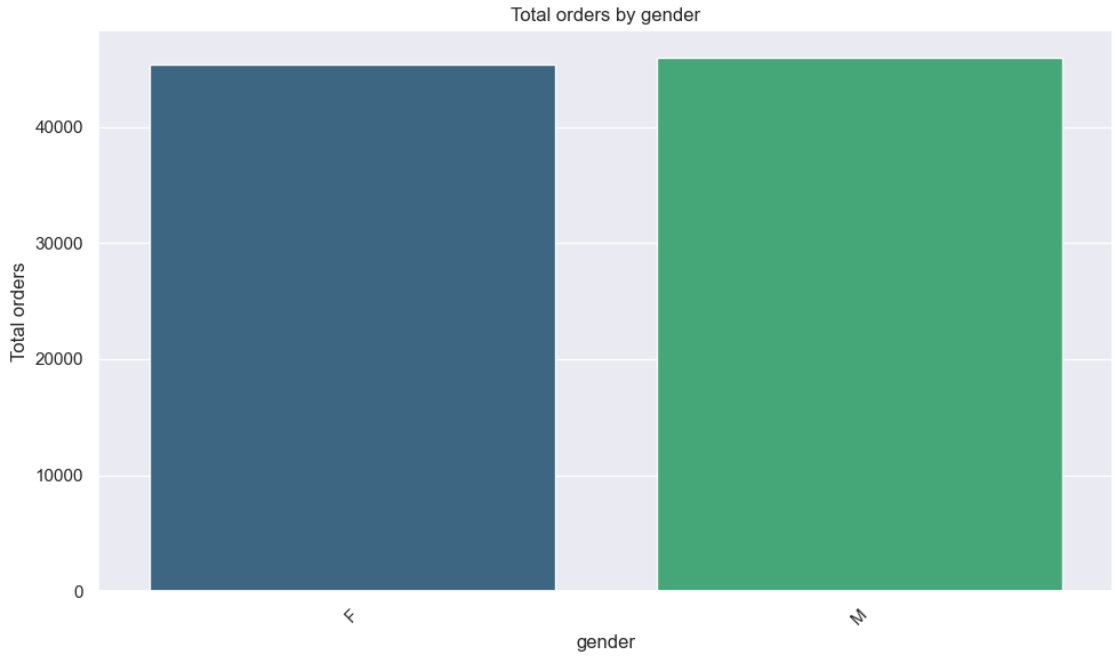


[47]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

product_orders = df.groupby('ProductName')['OrderQuantity'].sum().reset_index()

top_10_products = product_orders.sort_values(by='OrderQuantity',␣
 ↪ascending=False).head(10)

plt.figure(figsize=(14, 7))
ax = sns.barplot(x='ProductName', y='OrderQuantity', data=top_10_products,␣
 ↪palette='viridis')

ax.set_title('Top 10 Products by Total Orders')
ax.set_xlabel('ProductName')
ax.set_ylabel('Total Orders')
plt.xticks(rotation=90)

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



[48]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


gender_orders = df.groupby('Gender')['OrderQuantity'].sum().reset_index()
```

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

ax = sns.barplot(x='Gender', y='OrderQuantity', data=gender_orders,␣
 ↪palette='viridis')

ax.set_title('Total orders by gender')
ax.set_xlabel('gender')
ax.set_ylabel('Total orders')
plt.xticks(rotation=45)

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



[49]:
```
gender_orders = df.groupby('Gender')['OrderQuantity'].sum().reset_index()

gender_orders.reset_index(inplace=True)
fig = px.bar(gender_orders, x='Gender', y='OrderQuantity')
fig.update_layout(
    autosize=False,
    width=300,
    height=300,
    margin=dict(
        l=25,
        r=25,
        b=10,
```

```
        t=10,
    ))
fig.show()
```

[50]:
```
df.groupby(['Category', 'SubCategory','ProductName'])['Profit'].sum().
 ↪nsmallest(10).to_frame()
```

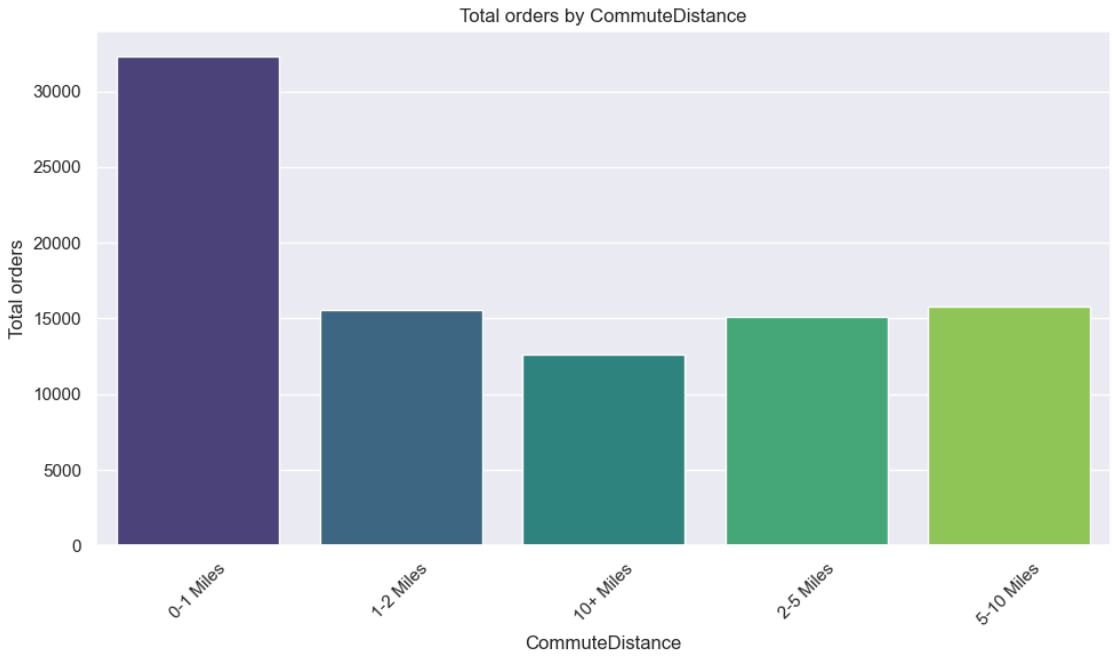
[50]:

			Profit
Category	SubCategory	ProductName	
Clothing	Socks	Racing Socks, L	1474.4574
		Racing Socks, M	1581.3837
Accessories	Cleaners	Bike Wash - Dissolver	4299.8688
	Tires and Tubes	Patch Kit/8 Patches	4314.8350
Clothing	Caps	AWC Logo Cap	4331.8315
Accessories	Tires and Tubes	Touring Tire Tube	4363.8089
Clothing	Jerseys	Long-Sleeve Logo Jersey, XL	4495.6007
		Short-Sleeve Classic Jersey, L	4544.8782
		Long-Sleeve Logo Jersey, S	4610.5777
		Short-Sleeve Classic Jersey, M	4793.2322

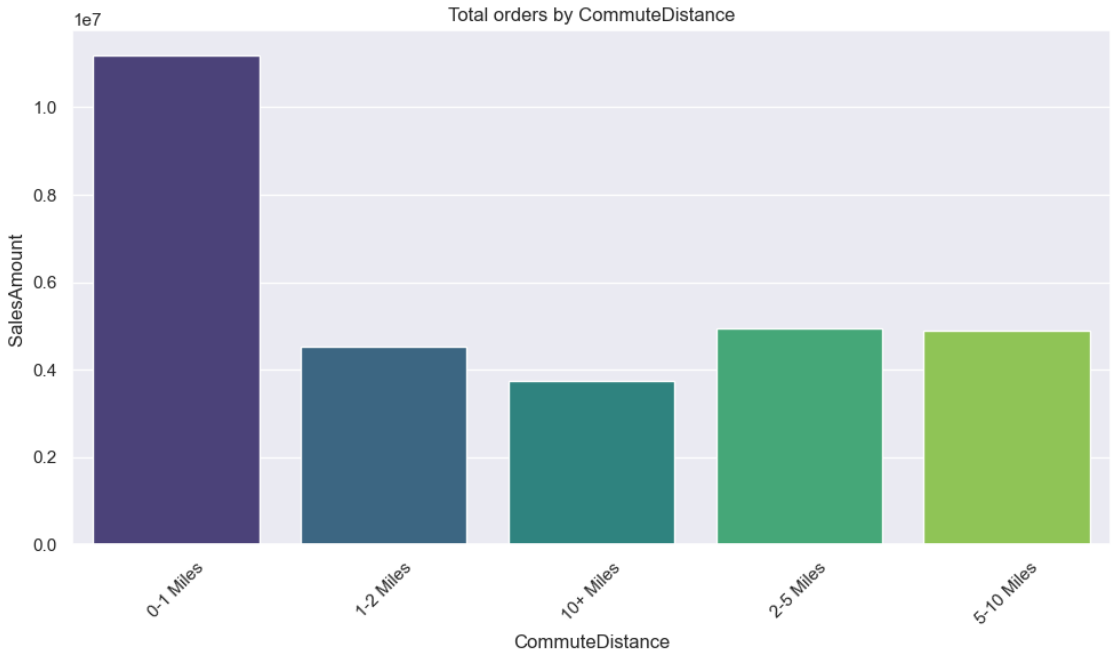
[51]:
```
country_sales = pd.DataFrame(df.groupby('Country').sum()[['SalesAmount',␣
 ↪'Profit']])
country_sales.reset_index(inplace=True)

fig = px.bar(country_sales, x='Country', y='Profit',text_auto='.2s',
             color='SalesAmount',
             height=400)
fig.show()
```

[52]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


gender_orders = df.groupby('CommuteDistance')['OrderQuantity'].sum().
 ↪reset_index()

plt.figure(figsize=(10, 6))

ax = sns.barplot(x='CommuteDistance', y='OrderQuantity', data=gender_orders,␣
 ↪palette='viridis')

ax.set_title('Total orders by CommuteDistance')
ax.set_xlabel('CommuteDistance')
ax.set_ylabel('Total orders')
plt.xticks(rotation=45)
```

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

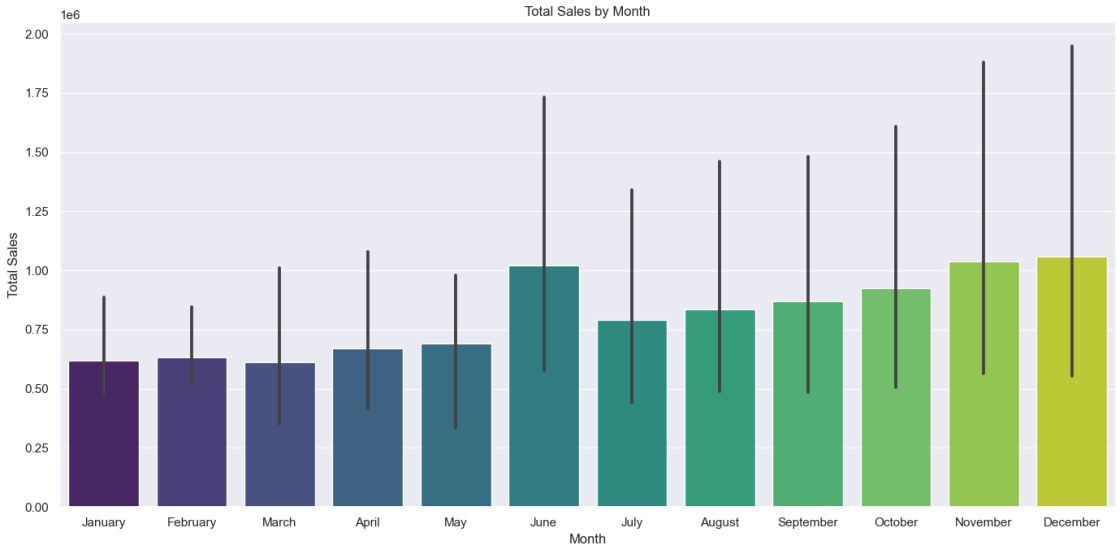


[53]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


gender_orders = df.groupby('CommuteDistance')['SalesAmount'].sum().reset_index()

plt.figure(figsize=(10, 6))

ax = sns.barplot(x='CommuteDistance', y='SalesAmount', data=gender_orders,␣
 ↪palette='viridis')

ax.set_title('Total orders by CommuteDistance')
ax.set_xlabel('CommuteDistance')
ax.set_ylabel('SalesAmount ')
plt.xticks(rotation=45)

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



[54]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the merged data
df = pd.read_excel("final_merged_data_updated.xlsx")

# Ensure OrderDate is in datetime format
df['OrderDate'] = pd.to_datetime(df['OrderDate'])

# Extract month names and year from OrderDate
df['MonthName'] = df['OrderDate'].dt.strftime('%B')
df['Year'] = df['OrderDate'].dt.year

# Combine Year and Month for sorting
df['YearMonth'] = df['OrderDate'].dt.to_period('M')

# Group by MonthName and YearMonth, and calculate total sales
monthly_sales = df.groupby(['YearMonth', 'MonthName'])['SalesAmount'].sum().
 ↪reset_index()

# Sort by YearMonth for proper ordering
monthly_sales = monthly_sales.sort_values('YearMonth')

# Plotting the bar graph
```

```
plt.figure(figsize=(14, 7))
ax = sns.barplot(x='MonthName', y='SalesAmount', data=monthly_sales,␣
 ↪palette='viridis')

# Customize the plot
ax.set_title('Total Sales by Month')
ax.set_xlabel('Month')
ax.set_ylabel('Total Sales')

# Show the plot
plt.tight_layout()
plt.show()
```



12 RFM- ANALYSIS

[55]:
```
import pandas as pd
import datetime as dt

# Load the data (assuming 'final_merged_data' is your DataFrame)
final_merged_data = pd.read_excel("final_merged_data_updated.xlsx")

# Ensure 'OrderDate' is in datetime format
final_merged_data['OrderDate'] = pd.to_datetime(final_merged_data['OrderDate'])

# Define the snapshot date (typically the day after the last order date in the␣
 ↪dataset)
snapshot_date = final_merged_data['OrderDate'].max() + dt.timedelta(days=1)
```

```python
# Calculate Recency for each customer
recency_df = final_merged_data.groupby('CustomerKey').agg({
    'OrderDate': lambda x: (snapshot_date - x.max()).days  # Recency calculation
}).reset_index()

# Rename columns to 'CustomerKey' and 'Recency'
recency_df.columns = ['CustomerKey', 'Recency']

# Display the recency result
print(recency_df.head())

# Save the recency result to an Excel file
recency_df.to_excel("recency_analysis.xlsx", index=False)
```

```
   CustomerKey  Recency
0        11000      241
1        11001       19
2        11002      308
3        11003      234
4        11004      243
```

[ ]: