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

import re

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
customers = pd.read_csv('.\\data\\QVI_purchase_behaviour.csv')
customers.head()
```

Out[2]:

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER
0	1000	YOUNG SINGLES/COUPLES	Premium
1	1002	YOUNG SINGLES/COUPLES	Mainstream
2	1003	YOUNG FAMILIES	Budget
3	1004	OLDER SINGLES/COUPLES	Mainstream
4	1005	MIDAGE SINGLES/COUPLES	Mainstream

In [3]:
```
transaction = pd.read_excel('.\\data\\QVI_transaction_data.xlsx')
transaction.head()
```

Out[3]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2
3	43329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3

I don't like the title letters in the column names and the date as a number

In [4]:
```
customers.columns = customers.columns.str.lower()
transaction.columns = transaction.columns.str.lower()
```

In [5]:
```
start_date = pd.to_datetime('1899-12-30')
transaction['date'] = transaction['date'].apply(lambda x: start_date + pd.Timedelta
```

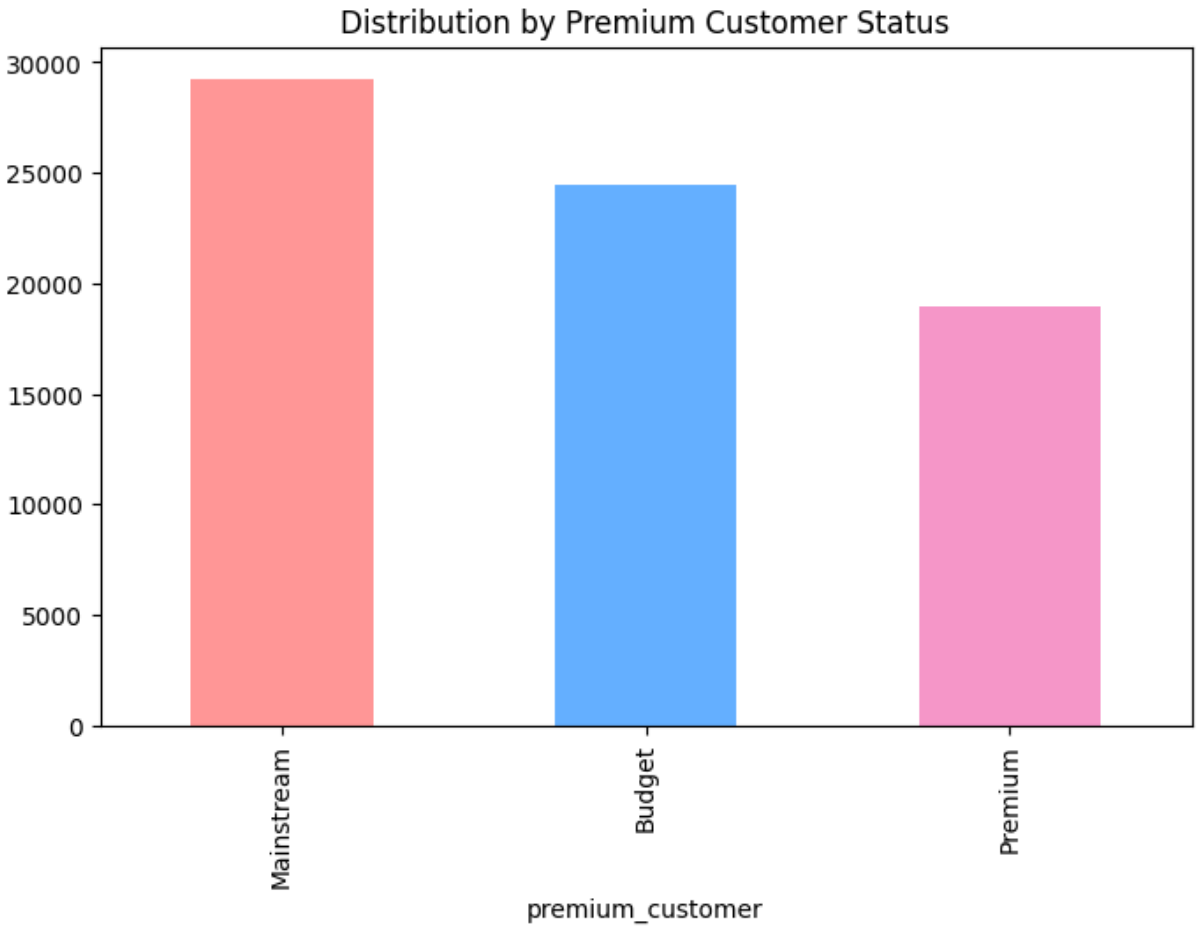
In [6]:
```
customers.info(), transaction.info()
```

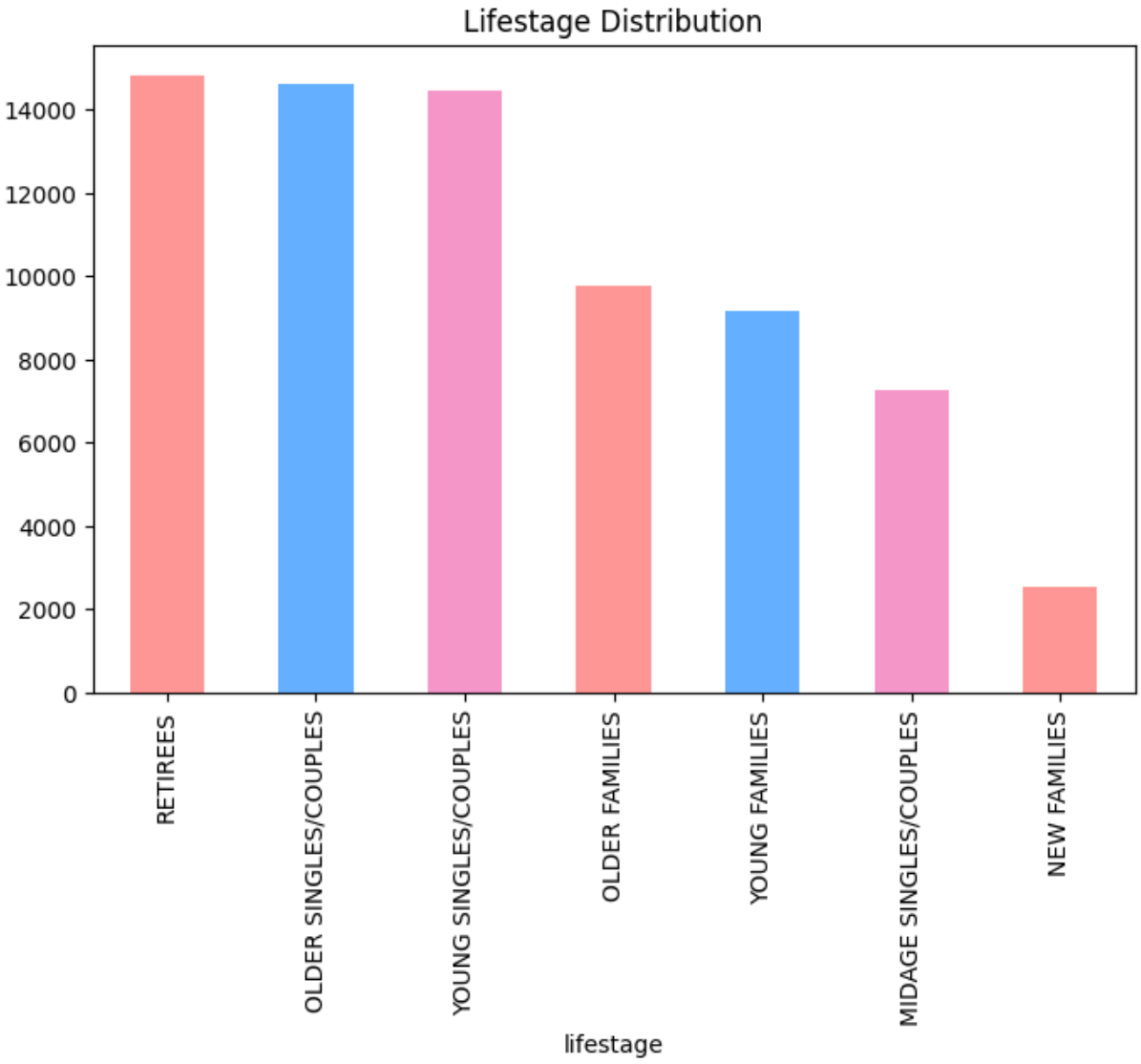
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72637 entries, 0 to 72636
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   lylty_card_nbr    72637 non-null  int64 
 1   lifestage         72637 non-null  object
 2   premium_customer  72637 non-null  object
dtypes: int64(1), object(2)
memory usage: 1.7+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264836 entries, 0 to 264835
Data columns (total 8 columns):
 #   Column          Non-Null Count   Dtype         
---  ------          --------------   -----         
 0   date            264836 non-null  datetime64[ns]
 1   store_nbr       264836 non-null  int64         
 2   lylty_card_nbr  264836 non-null  int64         
 3   txn_id          264836 non-null  int64         
 4   prod_nbr        264836 non-null  int64         
 5   prod_name       264836 non-null  object        
 6   prod_qty        264836 non-null  int64         
 7   tot_sales       264836 non-null  float64       
dtypes: datetime64[ns](1), float64(1), int64(5), object(1)
memory usage: 16.2+ MB
```

Out[6]: (None, None)

I want to see the distribution in the database with customers

In [7]:
```
# Plot distribution by Premium Customer Status
plt.figure(figsize=(8, 5))
customers.premium_customer.value_counts().plot(kind='bar', color=['#FF9999', '#66B3
plt.title('Distribution by Premium Customer Status')
plt.show()

# Plot Lifestage Distribution
plt.figure(figsize=(8, 5))
customers.lifestage.value_counts().plot(kind='bar', color=['#FF9999', '#66B3FF', '#
plt.title('Lifestage Distribution')
plt.show()
```


Distribution by Premium Customer Status
30000
25000
20000
15000
10000
5000
0
Mainstream
Budget
Premium
premium_customer



Let's look at the duplicates

In [8]:
```
transaction.duplicated().sum()
```

Out[8]: np.int64(1)

In [9]:
```
transaction.drop_duplicates(inplace=True)
transaction.duplicated().sum()
```

Out[9]: np.int64(0)

Add columns with customer types to the table with transactions

In [10]:
```
df = pd.merge(transaction, customers, on='lylty_card_nbr')
df.head()
```

Out[10]:

	date	store_nbr	lylty_card_nbr	txn_id	prod_nbr	prod_name	prod_qty	tot_sales
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15.0
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	13.8

In [11]:
```
df.shape, transaction.shape, customers.shape
```

Out[11]:
```
((264835, 10), (264835, 8), (72637, 3))
```

We only need chips

In [12]:
```
# regex to find chips and crisps and 'RRD' products in product name
chips_products = df[df['prod_name'].str.contains(r'\b(Chips?)\b|\b(Chps?)\b|\b(red)
```

Add a few necessary columns such as brand and weight

In [13]:
```
chips_products['brand'] = chips_products['prod_name'].str.split().str[0]
```

In [14]:
```
chips_products['weight'] = chips_products['prod_name'].str[-4:]
```

In [15]:
```
chips_products[['prod_qty', 'tot_sales']].describe()
```

Out[15]:

	prod_qty	tot_sales
count	101986.000000	101986.000000
mean	1.907154	6.738020
std	0.943519	3.733085
min	1.000000	1.700000
25%	2.000000	5.400000
50%	2.000000	6.600000
75%	2.000000	7.800000
max	200.000000	650.000000

I see an outlier in the data, let's take a closer look

In [16]:
```
chips_products[chips_products['prod_qty'] == 200]
```

Out[16]:

	date	store_nbr	lylty_card_nbr	txn_id	prod_nbr	prod_name	prod_qty	tot_sales
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme 380g	200	650.0
69763	2019-05-20	226	226000	226210	4	Dorito Corn Chp Supreme 380g	200	650.0

In [17]:
```
chips_products = chips_products[chips_products['lylty_card_nbr'] != 226000]
```

In [18]:
```
chips_products[['prod_qty', 'tot_sales']].describe()
```

Out[18]:

	prod_qty	tot_sales
count	101984.000000	101984.000000
mean	1.903269	6.725405
std	0.347382	2.412724
min	1.000000	1.700000
25%	2.000000	5.400000
50%	2.000000	6.600000
75%	2.000000	7.800000
max	5.000000	29.500000

In [19]:
```
chips_products.brand.unique()
```

Out[19]:
```
array(['Natural', 'Smiths', 'Doritos', 'Thins', 'Red', 'Dorito',
       'Tyrrells', 'Cobs', 'French', 'Pringles', 'RRD', 'WW', 'Snbts',
       'Sunbites'], dtype=object)
```

Many identical brands are recorded differently let's combine them

In [20]:
```
chips_products['brand'] = chips_products['brand'].replace('Dorito', 'Doritos')
```

In [21]:
```
chips_products['brand'] = chips_products['brand'].replace('Red', 'RRD')
```

In [22]:
```
chips_products['brand'] = chips_products['brand'].replace('Snbts', 'Sunbites')
```

In [23]:
```
chips_products['weight'].unique()
```

Out[23]:
```
array(['175g', '170g', '330g', '150g', '150G', '380g', '165g', '110g',
       '134g', '200g', '160g', ' 90g'], dtype=object)
```

In [24]:
```
chips_products['weight'] = chips_products['weight'].replace('150G', '150g')
```

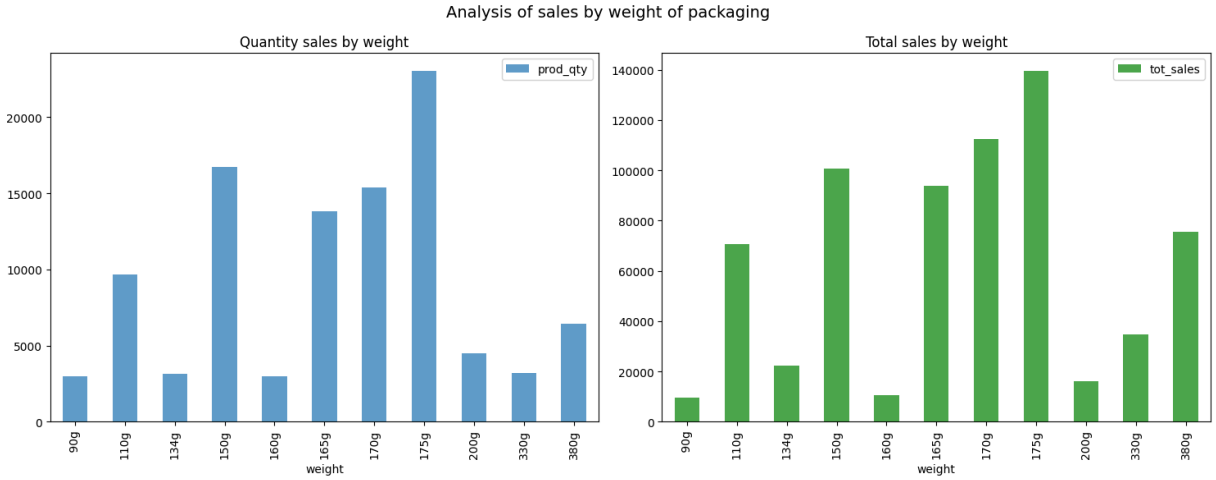
In [25]:
```
chips_products.head()
```

Out[25]:

	date	store_nbr	lylty_card_nbr	txn_id	prod_nbr	prod_name	prod_qty	tot_sales
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15.0
6	2019-05-16	4	4149	3333	16	Smiths Crinkle Chips Salt & Vinegar 330g	1	5.7
8	2018-08-20	5	5026	4525	42	Doritos Corn Chip Mexican Jalapeno 150g	1	3.9

What volume of packaging is sold more?

In [26]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))


chips_products.groupby(['weight']).agg({'prod_qty':'count'}).sort_index().plot(
    kind='bar',
    alpha=0.7,
    ax=ax1
)
ax1.set_title('Quantity sales by weight')


chips_products.groupby(['weight']).agg({'tot_sales':'sum'}).sort_index().plot(
    kind='bar',
    color='green',
    alpha=0.7,
    ax=ax2
)
ax2.set_title('Total sales by weight')


plt.suptitle('Analysis of sales by weight of packaging', fontsize=14)
plt.tight_layout()
plt.show()
```



In [27]:
```
chips_products.groupby('weight').agg({'brand': 'nunique'}).sort_values('brand', asc
```

Out[27]:

weight	brand
175g	4
150g	2
170g	2
380g	2
165g	2
90g	1
110g	1
134g	1
160g	1
200g	1
330g	1

In [28]:
```
chips_products.groupby('weight')['brand'].unique()
```

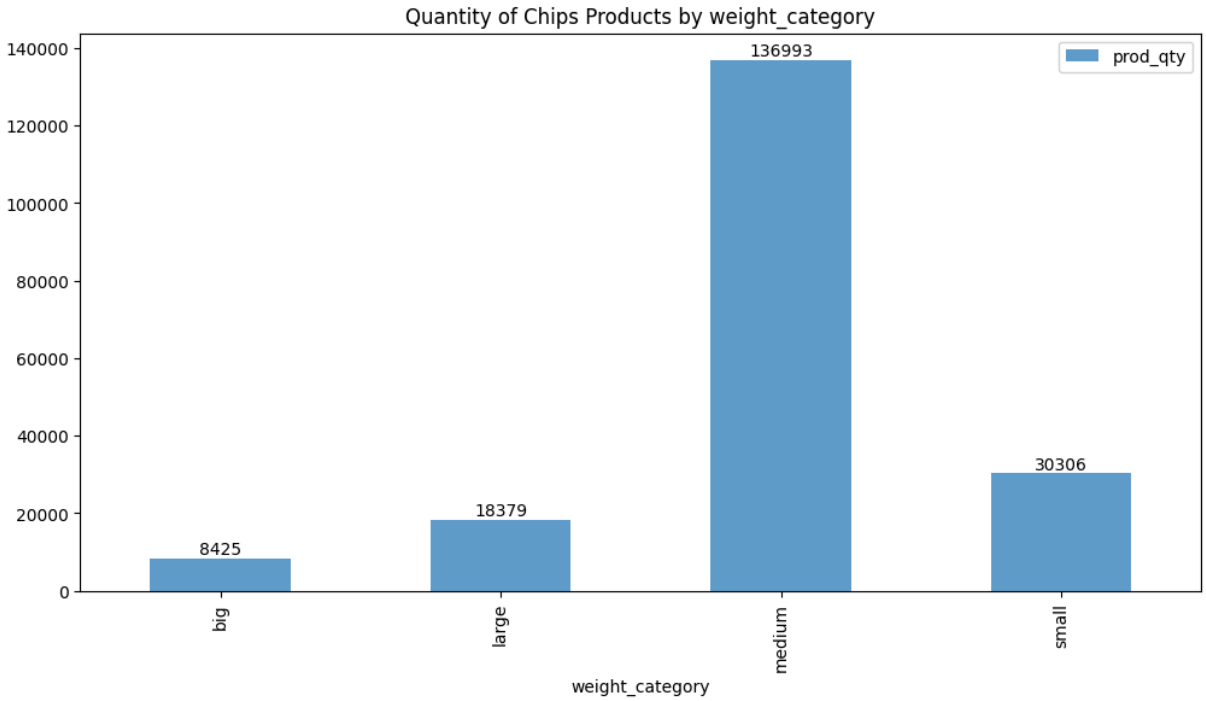
Out[28]:
```
weight
 90g                           [Sunbites]
110g                               [Cobs]
134g                           [Pringles]
150g                       [Doritos, RRD]
160g                                 [WW]
165g                      [Tyrrells, RRD]
170g                    [Smiths, Doritos]
175g    [Natural, Smiths, Thins, French]
200g                                 [WW]
330g                             [Smiths]
380g                    [Doritos, Smiths]
Name: brand, dtype: object
```

In [29]:
```
conditions = [
    chips_products['weight'].isin([' 90g', '110g', '134g']),
    chips_products['weight'].isin(['150g', '160g', '165g', '175g', '170g']),
    chips_products['weight'].isin(['200g']),
    chips_products['weight'].isin(['330g', '380g'])
]

choices = ['small', 'medium', 'big', 'large']


chips_products['weight_category'] = np.select(conditions, choices, default='unknown


chips_products.groupby('weight_category')['weight'].unique()
```

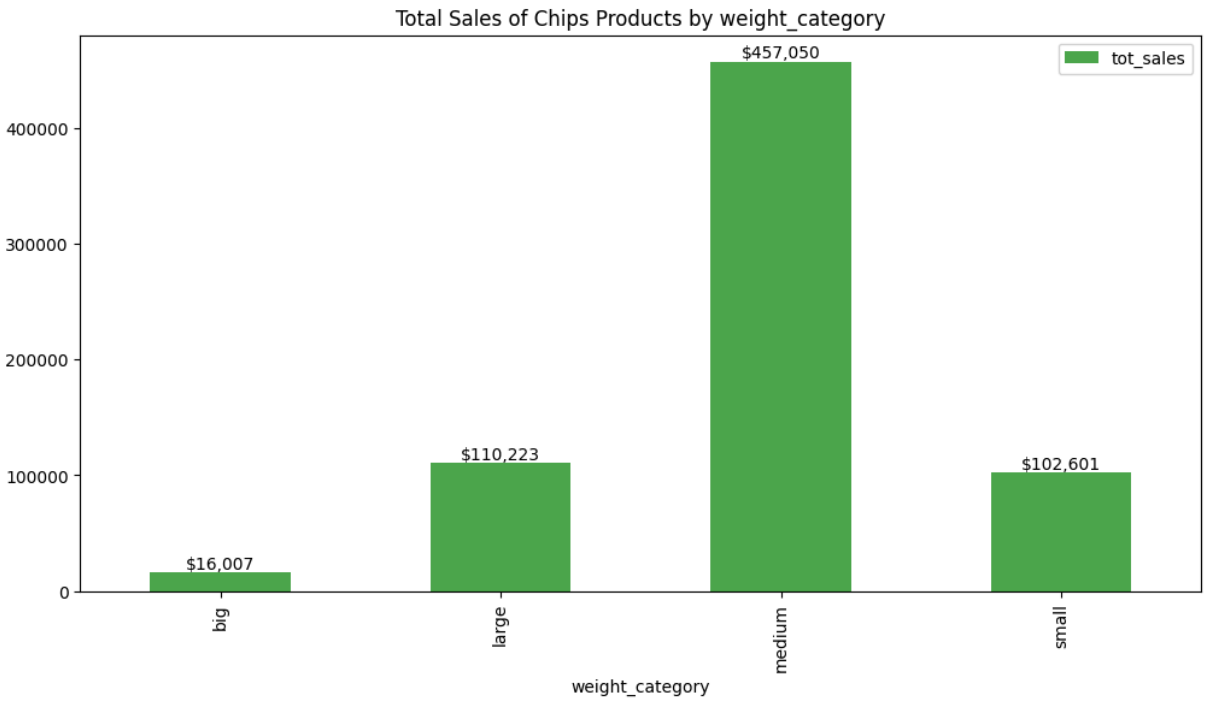
Out[29]:
```
weight_category
big                                [200g]
large                         [330g, 380g]
medium    [175g, 170g, 150g, 165g, 160g]
small                 [110g, 134g,  90g]
Name: weight, dtype: object
```

In [30]:
```
plt.figure(figsize=(12, 6))
ax = chips_products.groupby(['weight_category']).agg({'prod_qty':'sum'}).sort_index
plt.title('Quantity of Chips Products by weight_category')

# Добавляем значения над столбцами
for i, v in enumerate(chips_products.groupby(['weight_category']).agg({'prod_qty':'
    ax.text(i, v, str(int(v)), ha='center', va='bottom')

plt.show()

plt.figure(figsize=(12, 6))
ax = chips_products.groupby(['weight_category']).agg({'tot_sales':'sum'}).sort_inde
plt.title('Total Sales of Chips Products by weight_category')

# Добавляем значения над столбцами
for i, v in enumerate(chips_products.groupby(['weight_category']).agg({'tot_sales':
    ax.text(i, v, f'${int(v):,}', ha='center', va='bottom')

plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```




```
<Figure size 1200x600 with 0 Axes>
```



Let's look at the dynamics of sales over time.

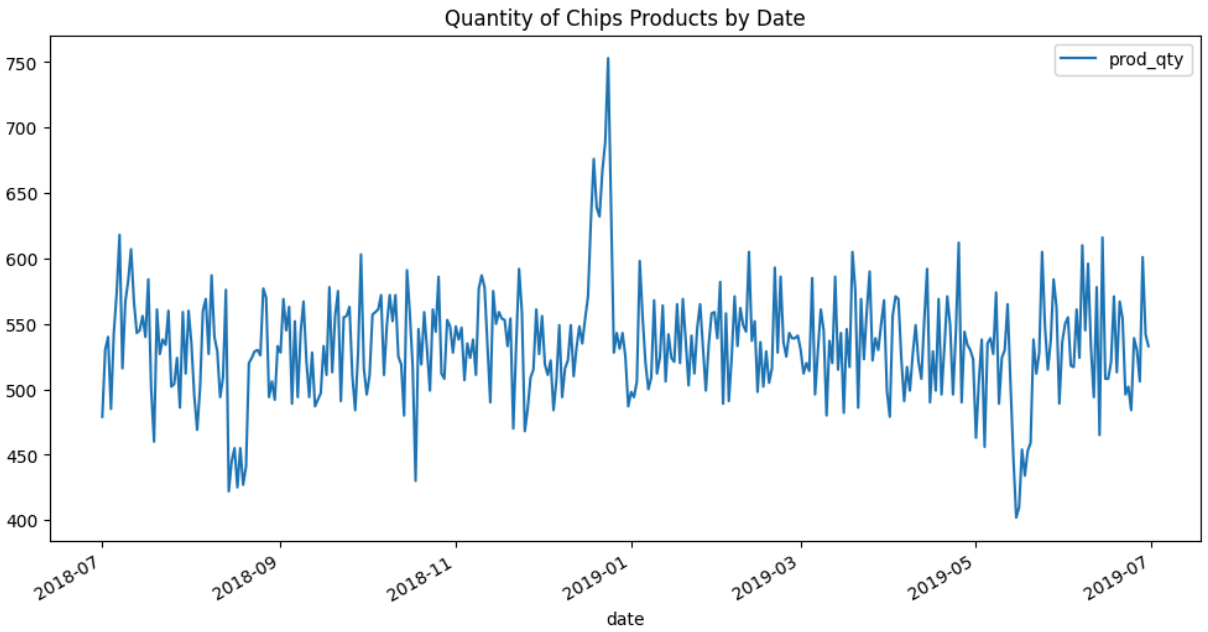
In [31]:
```
# Create a full range of dates
full_date_range = pd.date_range(start=chips_products['date'].min(),
                                end=chips_products['date'].max(),
                                freq='D')

# find missing dates
missing_dates = full_date_range[~full_date_range.isin(chips_products['date'])]
```

```
print(f"Number of missing dates: {len(missing_dates)}")
if len(missing_dates) > 0:
    print("\nMissing dates:")
    print(missing_dates)
```

```
Number of missing dates: 1

Missing dates:
DatetimeIndex(['2018-12-25'], dtype='datetime64[ns]', freq='D')
```

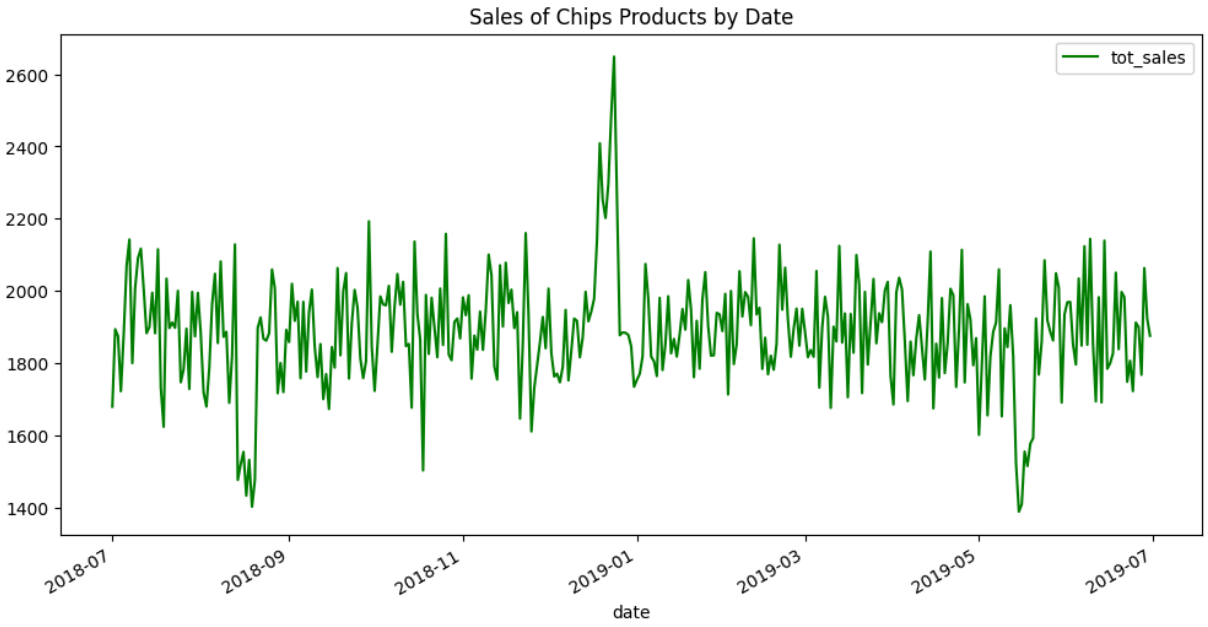
In [32]:
```
chips_products.groupby(['date']).agg({'prod_qty':'sum'}).sort_index().plot(figsize=
```

Out[32]: <Axes: title={'center': 'Quantity of Chips Products by Date'}, xlabel='date'>



In [33]:
```
chips_products.groupby(['date']).agg({'tot_sales':'sum'}).sort_index().plot(figsize
```

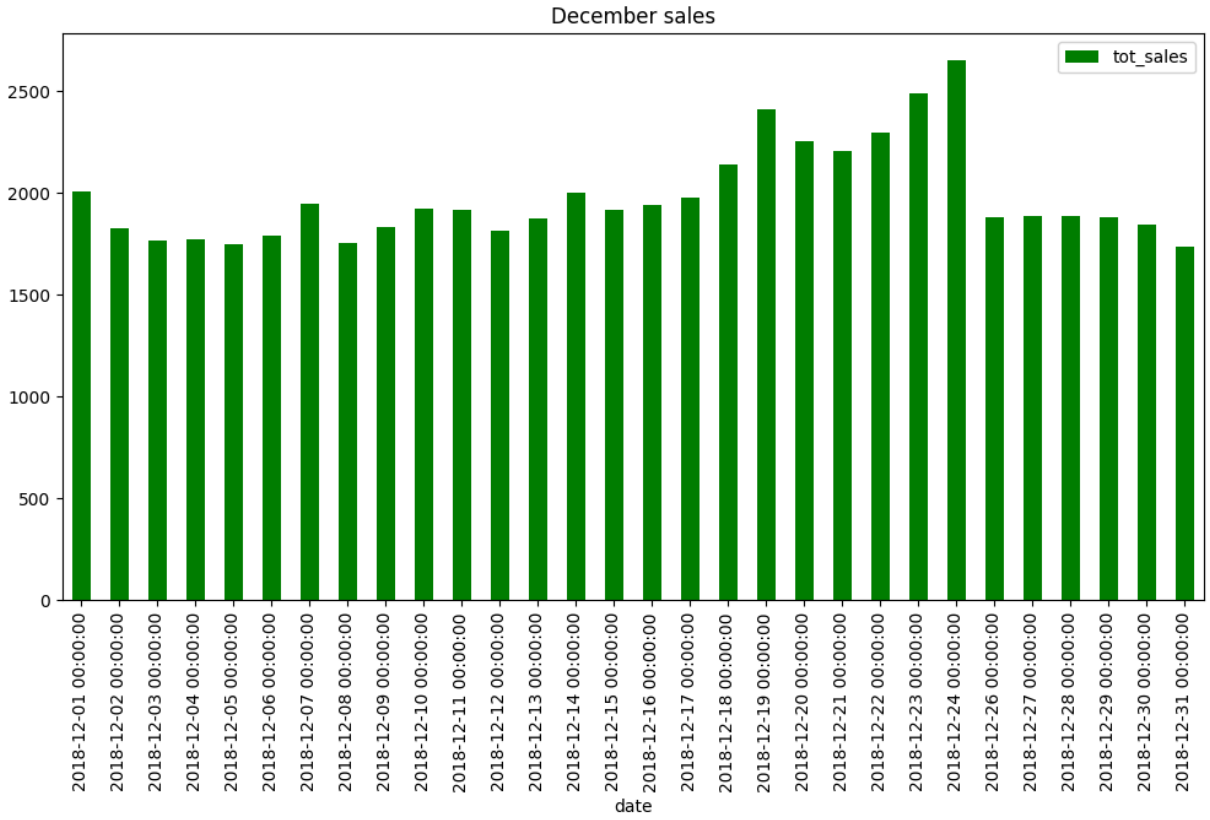
Out[33]: <Axes: title={'center': 'Sales of Chips Products by Date'}, xlabel='date'>



In [34]:
```
chips_products['month'] = chips_products['date'].dt.to_period('M')
```

In [35]:
```
december_sales = chips_products[chips_products['month'] == '2018-12'].groupby(['dat
december_sales.plot(figsize=(12, 6), kind='bar', x='date', y='tot_sales', color='gr
```

Out[35]: <Axes: title={'center': 'December sales'}, xlabel='date'>



We have a gap in the data for 2018-12-25 and then a suspicious decline in sales from the 26th.

In [36]:
```
december_sales.groupby(['date']).agg({'tot_sales':'sum'}).sort_index()
```

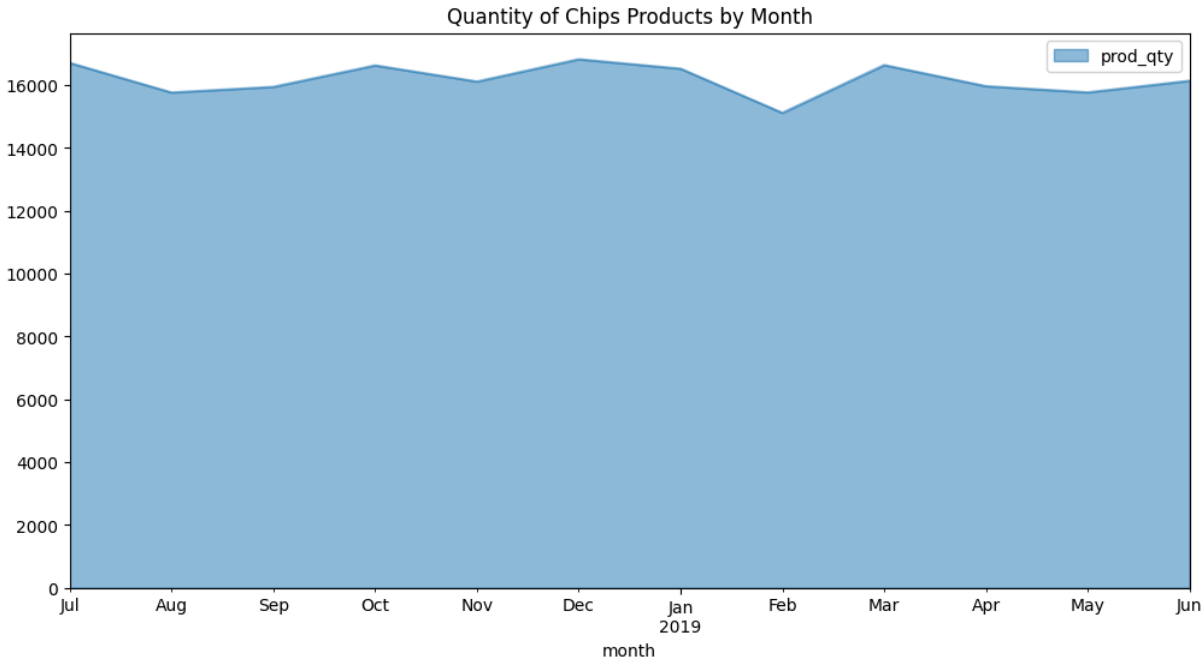
Out[36]:

date	tot_sales
2018-12-01	2006.3
2018-12-02	1827.2
2018-12-03	1762.8
2018-12-04	1770.6
2018-12-05	1746.7
2018-12-06	1789.8
2018-12-07	1947.3
2018-12-08	1751.9
2018-12-09	1833.9
2018-12-10	1923.2
2018-12-11	1916.9
2018-12-12	1816.1
2018-12-13	1871.0
2018-12-14	1997.4
2018-12-15	1915.6
2018-12-16	1942.8
2018-12-17	1977.9
2018-12-18	2140.6
2018-12-19	2409.0
2018-12-20	2252.6
2018-12-21	2202.0
2018-12-22	2298.1
2018-12-23	2487.1
2018-12-24	2649.0
2018-12-26	1877.2
2018-12-27	1884.8
2018-12-28	1884.0
2018-12-29	1877.9
2018-12-30	1846.5

	tot_sales
date	
2018-12-31	1734.4

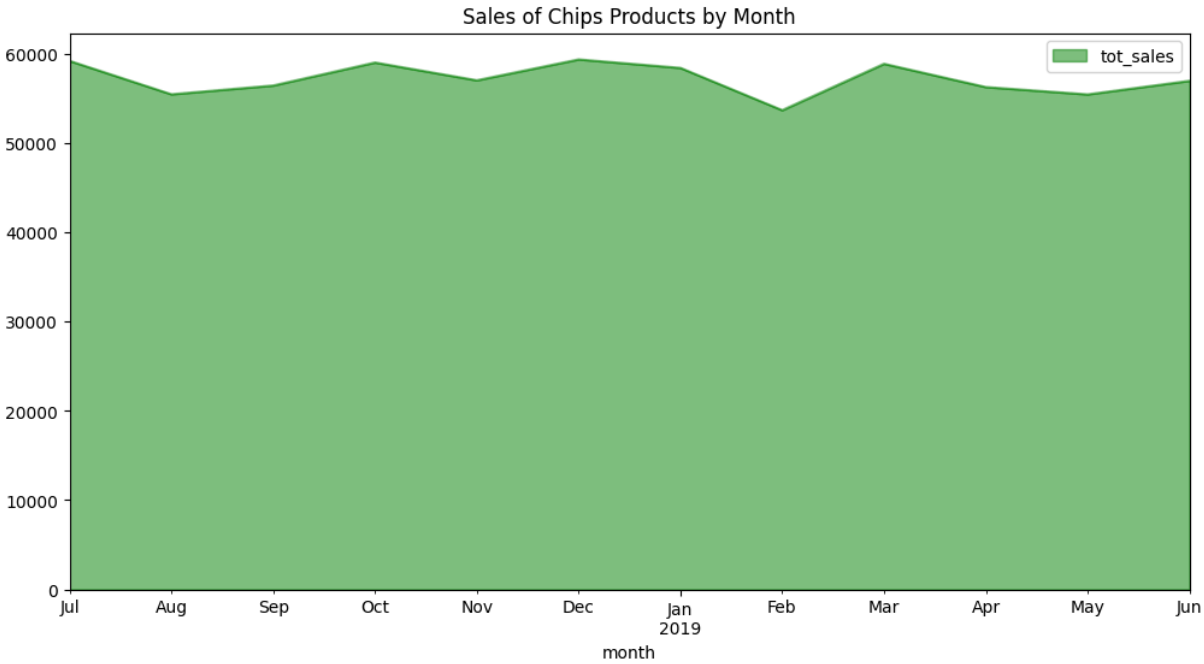
In [37]:
```
chips_products.groupby(['month']).agg({'prod_qty':'sum'}).sort_index().plot(figsize
```

Out[37]: <Axes: title={'center': 'Quantity of Chips Products by Month'}, xlabel='month'>



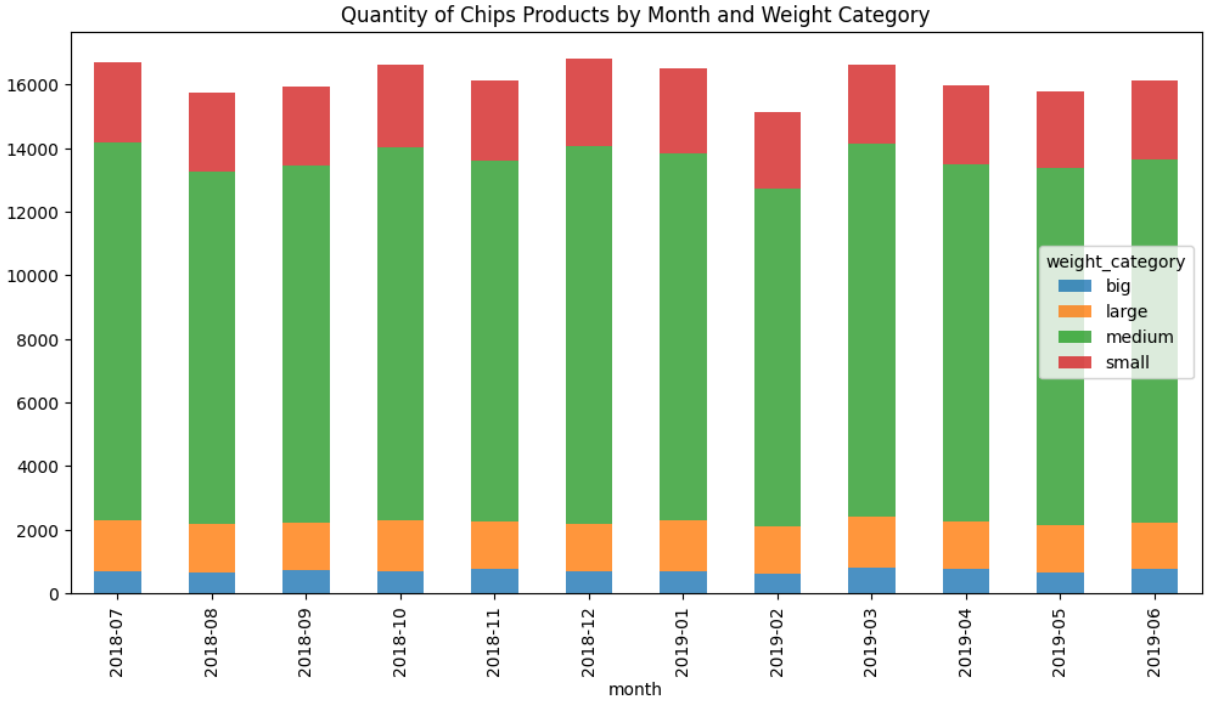
In [38]:
```
chips_products.groupby(['month']).agg({'tot_sales':'sum'}).sort_index().plot(figsiz
```

Out[38]: <Axes: title={'center': 'Sales of Chips Products by Month'}, xlabel='month'>



In [39]:
```
chips_products.pivot_table(index='month', columns='weight_category', values='prod_q
    .plot(figsize=(12, 6), kind='bar', stacked=True, alpha=0.8, title='Quantity of
```

Out[39]:
```
<Axes: title={'center': 'Quantity of Chips Products by Month and Weight Categor
y'}, xlabel='month'>
```



Let's see how each brand is sold.

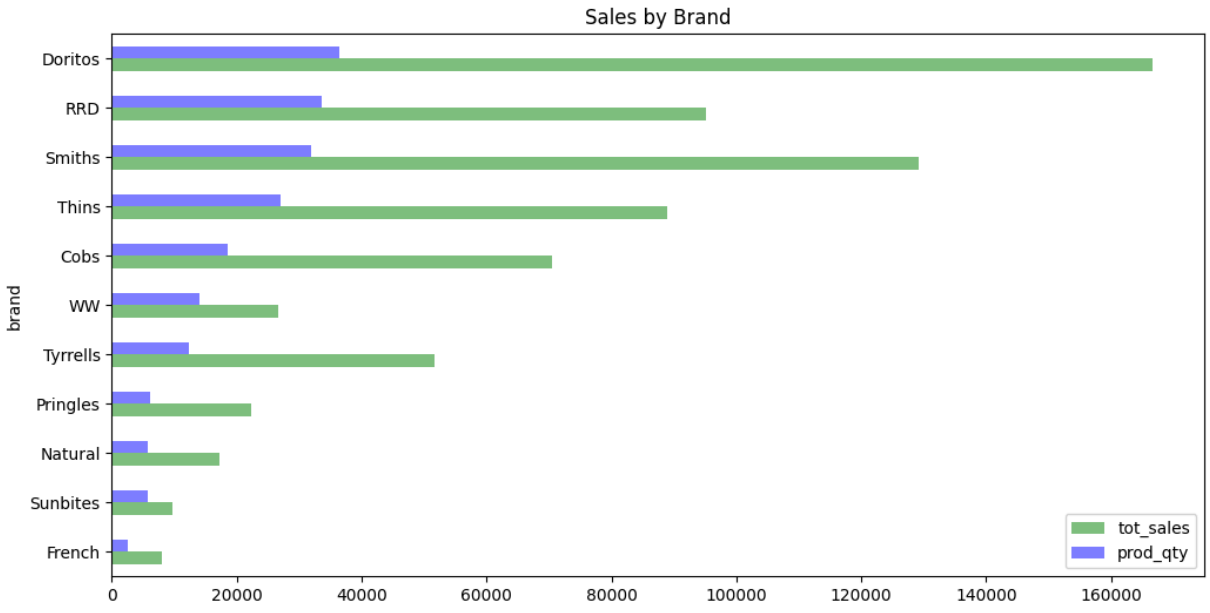
In [40]:
```
chips_products.groupby(['brand']).agg({'tot_sales':['sum','mean'], 'prod_qty':['sum
    .sort_values(by=('tot_sales', 'sum'), ascending=False)
```

Out[40]:

	tot_sales		prod_qty	
	sum	mean	sum	mean
brand				
Doritos	166649.3	8.744781	36498	1.915202
Smiths	129237.8	7.659898	31999	1.896574
RRD	95046.0	5.345970	33646	1.892457
Thins	88852.5	6.312789	26929	1.913250
Cobs	70569.8	7.280491	18571	1.915919
Tyrrells	51647.4	8.017293	12298	1.909034
WW	26655.1	3.581231	14029	1.884858
Pringles	22355.4	7.081216	6043	1.914159
Natural	17265.0	5.679276	5755	1.893092
Sunbites	9676.4	3.216888	5692	1.892287
French	7929.0	5.591678	2643	1.863893

In [41]:
```
chips_products.groupby(['brand']).agg({'tot_sales':'sum', 'prod_qty':'sum'})\
    .sort_values(by='prod_qty')\
    .plot(figsize=(12, 6),kind='barh',alpha=0.5, color=[ 'green', 'blue'], title='S
```

Out[41]: <Axes: title={'center': 'Sales by Brand'}, ylabel='brand'>



Who is buying more chips?

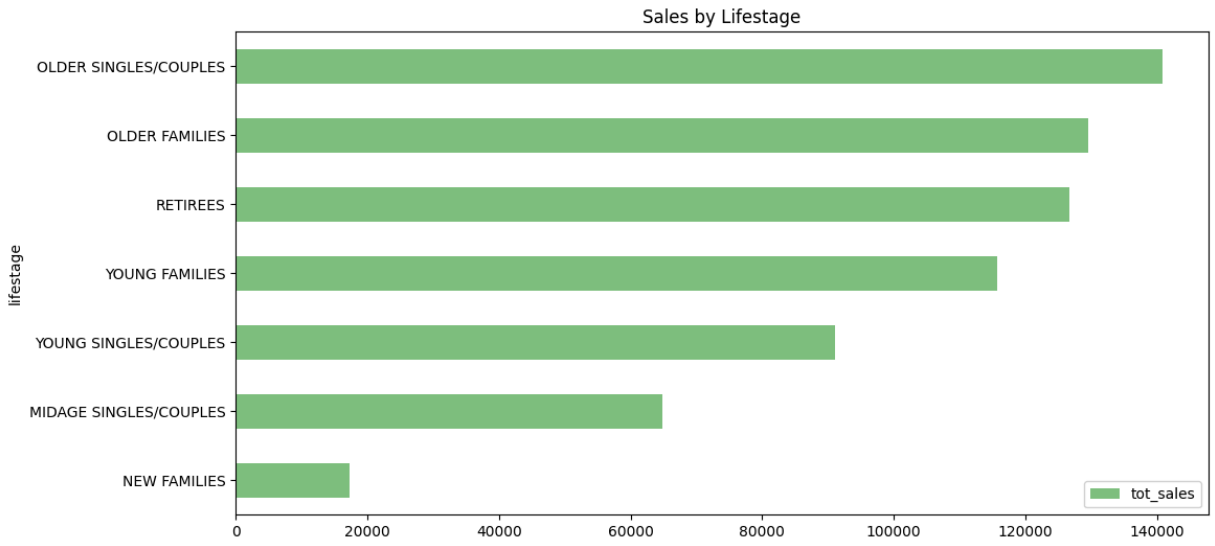
In [42]:
```
# mean sales by lifestage category
chips_products.groupby(['lifestage']).agg({'tot_sales':'mean', 'prod_qty':'mean'})
```

Out[42]:

lifestage	tot_sales	prod_qty
MIDAGE SINGLES/COUPLES	6.759145	1.894320
NEW FAMILIES	6.700713	1.847726
OLDER FAMILIES	6.694436	1.947205
OLDER SINGLES/COUPLES	6.815898	1.911543
RETIREES	6.772094	1.886525
YOUNG FAMILIES	6.695774	1.941595
YOUNG SINGLES/COUPLES	6.588514	1.820800

In [43]:
```
chips_products.groupby(['lifestage']).agg({'tot_sales':'sum'})\
    .sort_values(by='tot_sales')\
    .plot(figsize=(12, 6),kind='barh',alpha=0.5, color='green', title='Sales by Lif
```

Out[43]: <Axes: title={'center': 'Sales by Lifestage'}, ylabel='lifestage'>



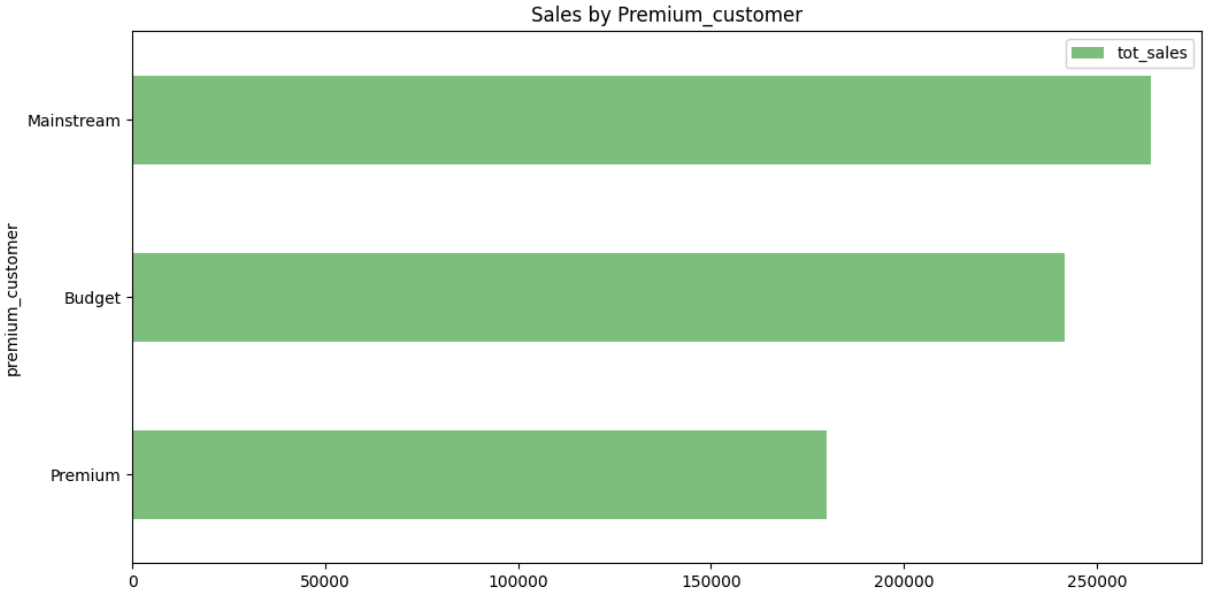
In [44]:
```
chips_products.groupby(['premium_customer']).agg({'tot_sales':'mean', 'prod_qty':'m
```

Out[44]:

premium_customer	tot_sales	prod_qty
Budget	6.671782	1.905292
Mainstream	6.806051	1.901147
Premium	6.681369	1.903606

In [45]:
```
chips_products.groupby(['premium_customer']).agg({'tot_sales':'sum'})\
    .sort_values(by='tot_sales')\
    .plot(figsize=(12, 6),kind='barh',alpha=0.5, color='green', title='Sales by Pre
```

Out[45]: <Axes: title={'center': 'Sales by Premium_customer'}, ylabel='premium_customer'>



In [46]:
```
pivot_brands = chips_products.pivot_table(
    index='brand',
    values='tot_sales',
    aggfunc={'count', 'mean', 'sum'}
)
pivot_brands.sort_values(by='sum', ascending=False)
```

Out[46]:

brand	count	mean	sum
Doritos	19057	8.744781	166649.3
Smiths	16872	7.659898	129237.8
RRD	17779	5.345970	95046.0
Thins	14075	6.312789	88852.5
Cobs	9693	7.280491	70569.8
Tyrrells	6442	8.017293	51647.4
WW	7443	3.581231	26655.1
Pringles	3157	7.081216	22355.4
Natural	3040	5.679276	17265.0
Sunbites	3008	3.216888	9676.4
French	1418	5.591678	7929.0

In [47]:
```
pivot_customer = chips_products.pivot_table(
    index='premium_customer',
    values='tot_sales',
    aggfunc={'count', 'mean', 'sum'}
```

```
)
pivot_customer.sort_values(by='sum', ascending=False)
```

Out[47]:

premium_customer	count	mean	sum
Mainstream	38805	6.806051	264108.80
Budget	36227	6.671782	241698.65
Premium	26952	6.681369	180076.25

In [48]:

```
pivot_lifestage = chips_products.pivot_table(
    index='lifestage',
    values='tot_sales',
    aggfunc={'count', 'mean', 'sum'}
)
pivot_lifestage.sort_values(by='sum', ascending=False)
```

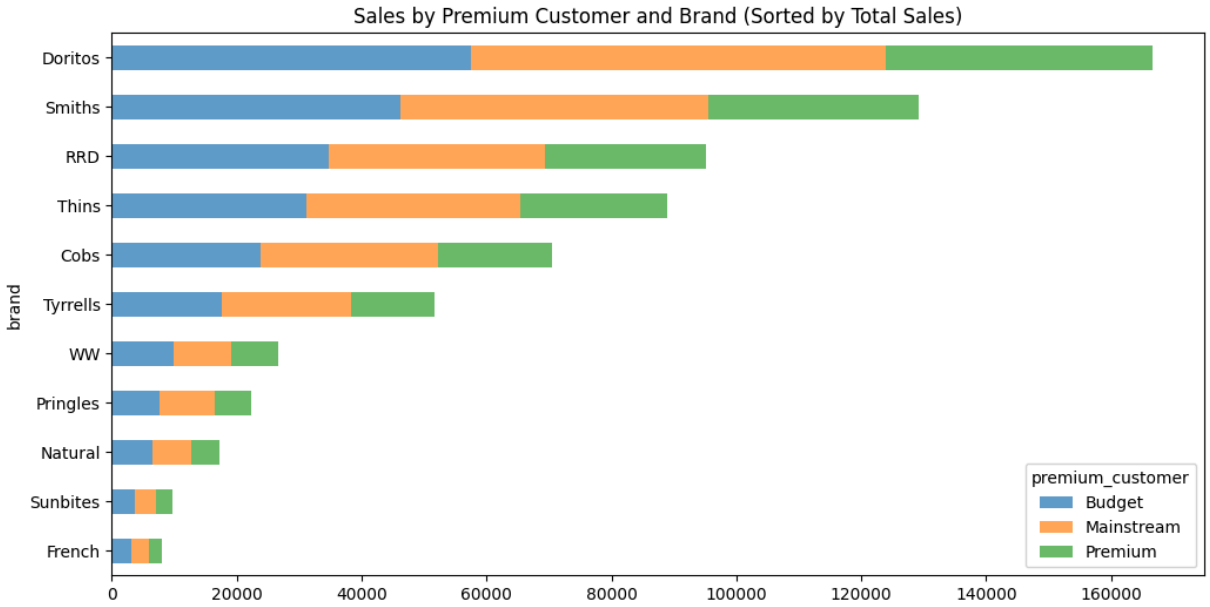
Out[48]:

lifestage	count	mean	sum
OLDER SINGLES/COUPLES	20654	6.815898	140775.55
OLDER FAMILIES	19339	6.694436	129463.70
RETIREES	18709	6.772094	126699.10
YOUNG FAMILIES	17276	6.695774	115676.20
YOUNG SINGLES/COUPLES	13817	6.588514	91033.50
MIDAGE SINGLES/COUPLES	9595	6.759145	64854.00
NEW FAMILIES	2594	6.700713	17381.65

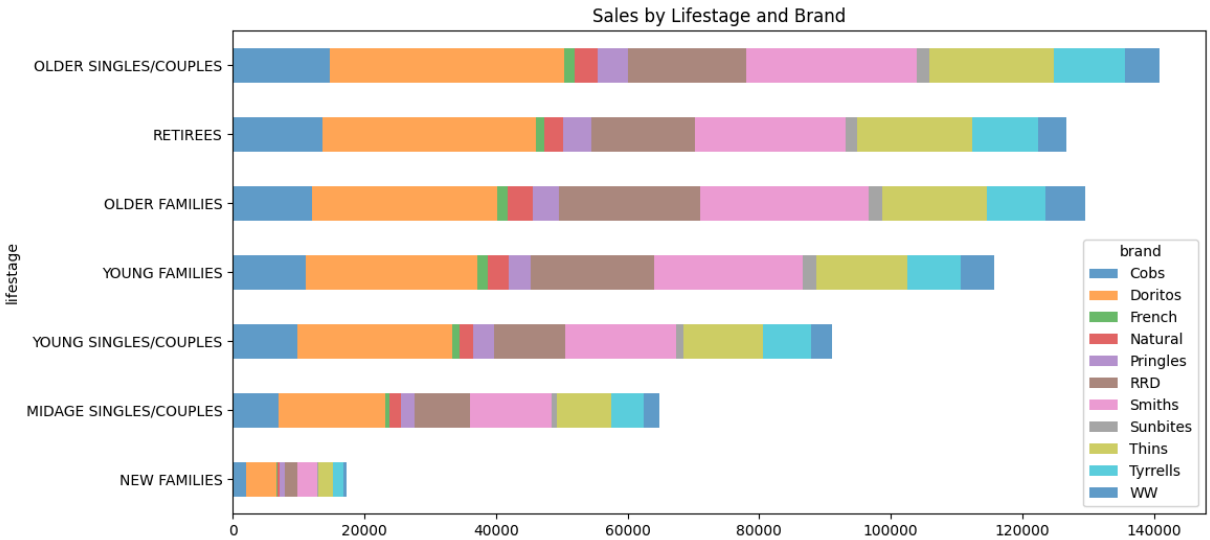
In [49]:

```
pivot = chips_products.pivot_table(
    index='brand',
    columns='premium_customer',
    values='tot_sales',
    aggfunc='sum'
)

# Sort by premium_customer
pivot_sorted = pivot.sort_values(by=pivot.columns.tolist(),
                                 ascending=True,
                                 axis=0)         # sort by index (rows)


pivot_sorted.plot(
    figsize=(12, 6),
    kind='barh',
    stacked=True,
    alpha=0.7,
```

```
    title='Sales by Premium Customer and Brand (Sorted by Total Sales)'
)
plt.show()
```



In [50]:
```
pivot_2 = chips_products.pivot_table(index='lifestage', columns='brand', values='to

pivot_2_sorted = pivot_2.sort_values(by=pivot_2.columns.tolist(),
                                     ascending=True,
                                     axis=0)

pivot_2_sorted.plot(
    figsize=(12, 6),
    kind='barh',
    stacked=True,
    alpha=0.7,
    title='Sales by Lifestage and Brand'
)
plt.show()
```



In [51]:
```
pivot = chips_products.pivot_table(
    index='brand',
    columns='premium_customer',
    values='tot_sales',
    aggfunc='sum',

)
percentage_total = pivot.div(pivot.sum().sum()) * 100


percentage_by_column = pivot.apply(lambda x: x/x.sum() * 100)


percentage_by_row = pivot.apply(lambda x: x/x.sum() * 100, axis=1)


percentage_total = percentage_total.round(2)
percentage_by_column = percentage_by_column.round(2)
percentage_by_row = percentage_by_row.round(2)
```

In [52]:
```
percentage_total.sum(axis=1).sort_values(ascending=False)
```

Out[52]:
```
brand
Doritos     24.30
Smiths      18.85
RRD         13.86
Thins       12.95
Cobs        10.30
Tyrrells     7.53
WW           3.88
Pringles     3.26
Natural      2.51
Sunbites     1.41
French       1.15
dtype: float64
```

In [53]:
```
chips_products.to_csv('.\\data\\QVI_chips_products_result.csv', index=False)
```