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

In [25]:
```
transactions= pd.read_csv('QVI_transaction_data.csv')
behaviour= pd.read_csv('QVI_purchase_behaviour.csv')
```

In [3]:
```
behaviour.head()
```

Out[3]:

	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 [4]:
```
transactions.head(3)
```

Out[4]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9

In [5]:
```
# Look for Empty cells

for i in transactions.columns:
    empty=np.mean(transactions[i].isnull())
    print( f'{i} -- {empty}% null values')
```

```
DATE -- 0.0% null values
STORE_NBR -- 0.0% null values
LYLTY_CARD_NBR -- 0.0% null values
TXN_ID -- 0.0% null values
PROD_NBR -- 0.0% null values
PROD_NAME -- 0.0% null values
PROD_QTY -- 0.0% null values
TOT_SALES -- 0.0% null values
```

In [6]:
```
for i in behaviour.columns:
    empty=np.mean(behaviour[i].isnull())
    print( f'{i} -- {empty}% null values')
```

```
LYLTY_CARD_NBR -- 0.0% null values
LIFESTAGE -- 0.0% null values
PREMIUM_CUSTOMER -- 0.0% null values
```

Data has no nulls

In [7]:
```
print(transactions.dtypes)
print(behaviour.dtypes)
```

```
DATE                int64
STORE_NBR           int64
LYLTY_CARD_NBR      int64
TXN_ID              int64
PROD_NBR            int64
PROD_NAME          object
PROD_QTY            int64
TOT_SALES         float64
dtype: object
LYLTY_CARD_NBR       int64
LIFESTAGE           object
PREMIUM_CUSTOMER    object
dtype: object
```

In [8]:
```
# Change Date column to Dates from numbers

import datetime

base_date = datetime.datetime(1900, 1, 1)

def date_change(provided_number):
    return base_date + datetime.timedelta(days=provided_number)

transactions['DATE']= transactions['DATE'].apply(date_change)

transactions.head()
```

Out[8]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
0	2018-10-19	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0
1	2019-05-16	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3
2	2019-05-22	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9
3	2018-08-19	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15.0
4	2018-08-20	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	3	13.8

In [9]:
```
transactions['PROD_NAME']=transactions.PROD_NAME.str.strip()
```

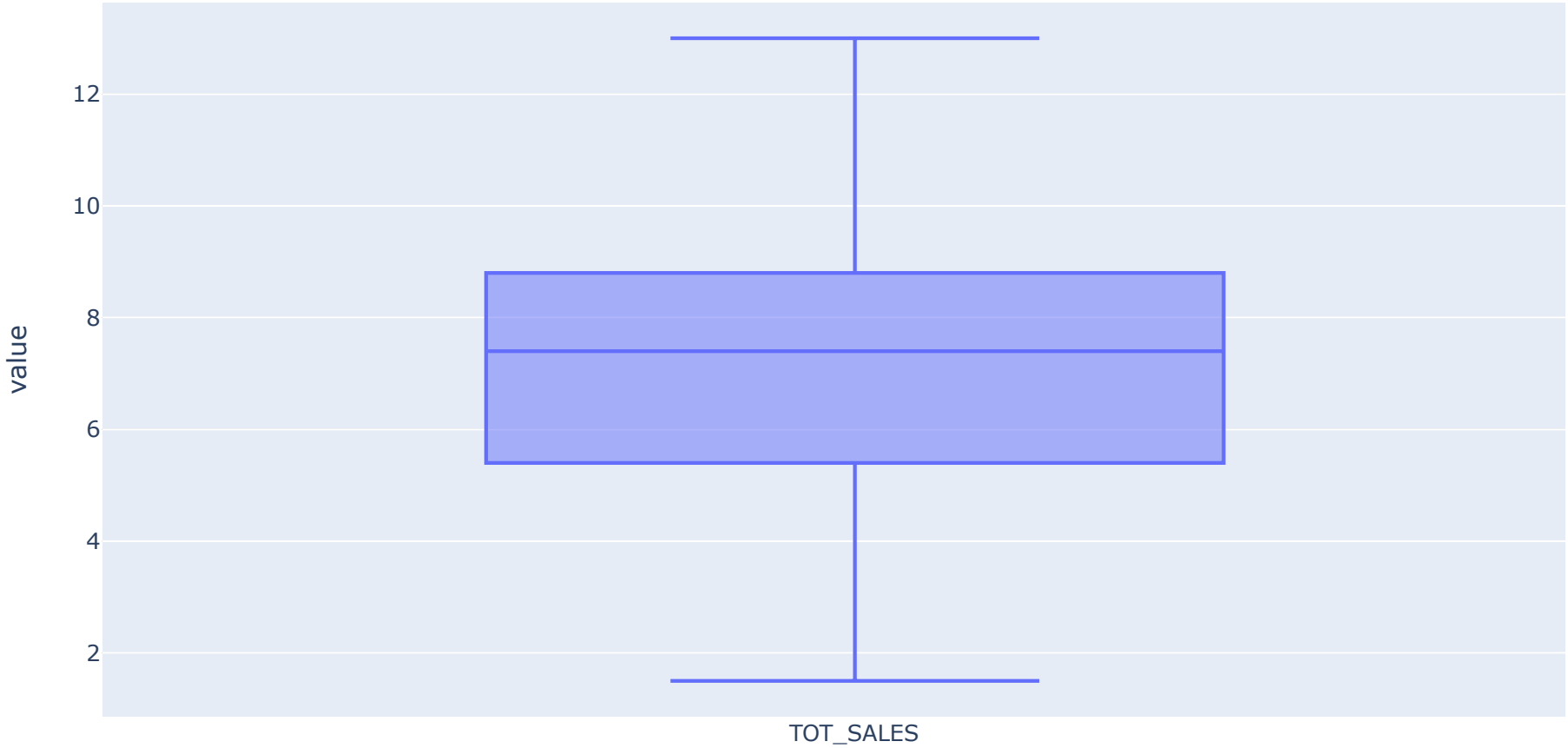
In [10]:
```
#Join behavior and transactions dataframes

df=pd.merge(transactions,behaviour)
df=pd.DataFrame(df)
df.head()
```

Out[10]:

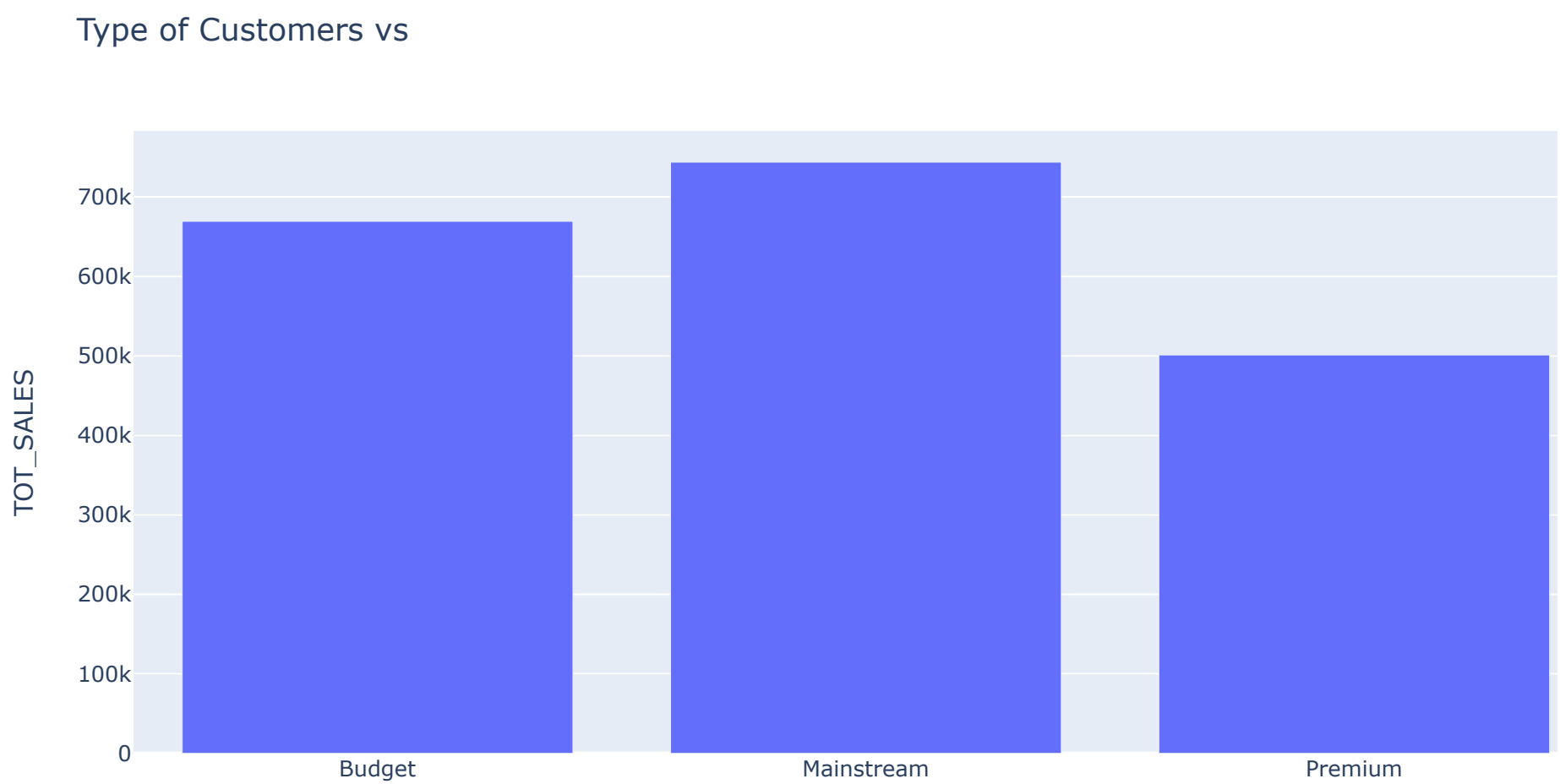
	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	LIFESTAGE	PREMIUM_CUST
0	2018-10-19	1	1000	1	5	Natural Chip Compny SeaSalt175g	2	6.0	YOUNG SINGLES/COUPLES	Pr
1	2019-05-16	1	1307	348	66	CCs Nacho Cheese 175g	3	6.3	MIDAGE SINGLES/COUPLES	E
2	2018-11-12	1	1307	346	96	WW Original Stacked Chips 160g	2	3.8	MIDAGE SINGLES/COUPLES	E
3	2019-03-11	1	1307	347	54	CCs Original 175g	1	2.1	MIDAGE SINGLES/COUPLES	E
4	2019-05-22	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	2	2.9	MIDAGE SINGLES/COUPLES	E

In [11]:
```
# Check for outliers in total sales

IQR= df['PROD_QTY'].quantile(0.75)-df['PROD_QTY'].quantile(0.25)
lower_bound = df['PROD_QTY'].quantile(0.25)- 1.5*IQR
upper_bound = df['PROD_QTY'].quantile(0.75)+ 1.5*IQR

outliers_low= df[(df['PROD_QTY']<lower_bound)]
outliers_high= df[(df['PROD_QTY']>upper_bound)]

df_no_outliers= df[~(df['PROD_QTY']>upper_bound)]
```

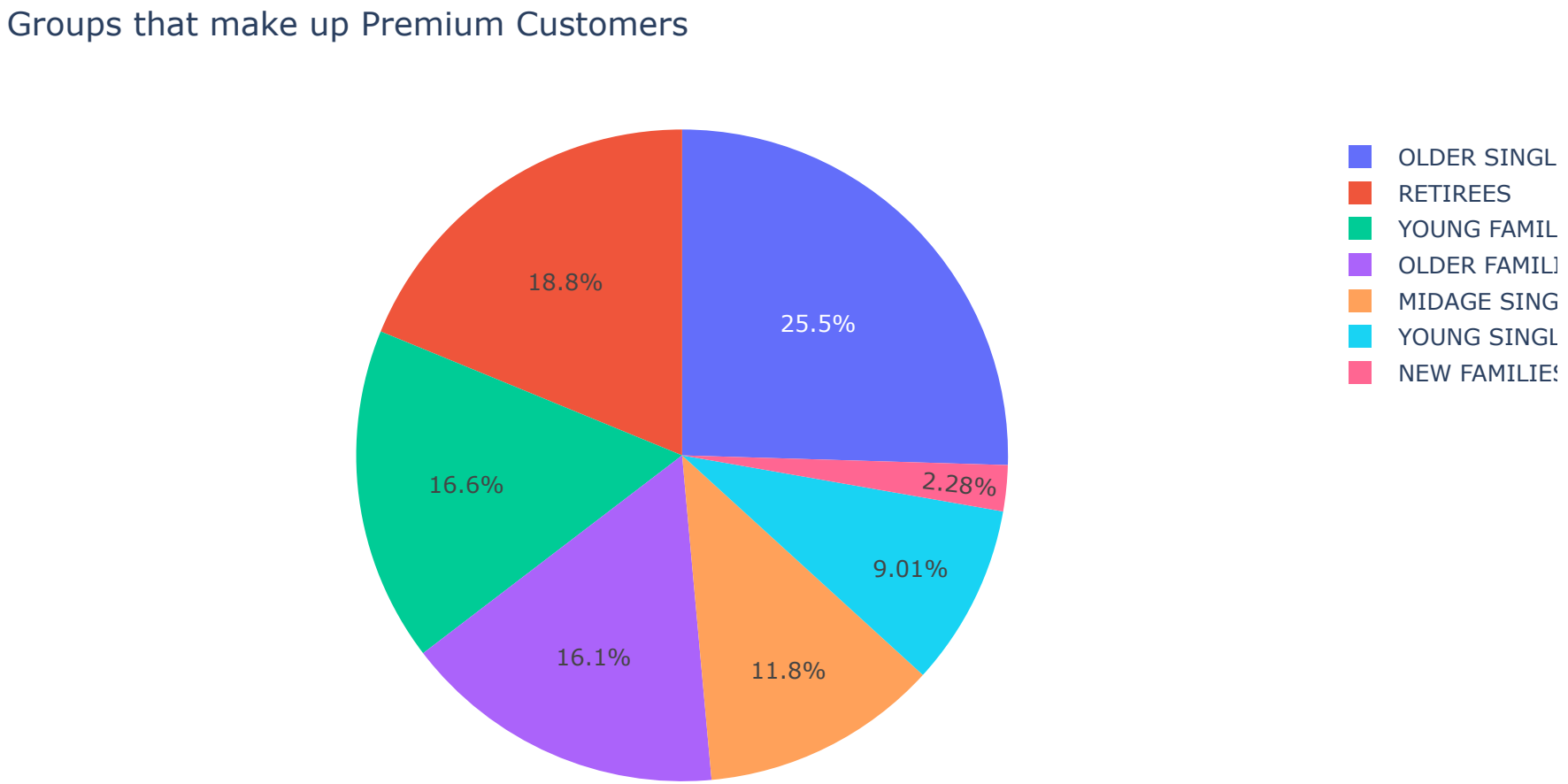
In [12]:
```
px.box(df_no_outliers['TOT_SALES'])
```

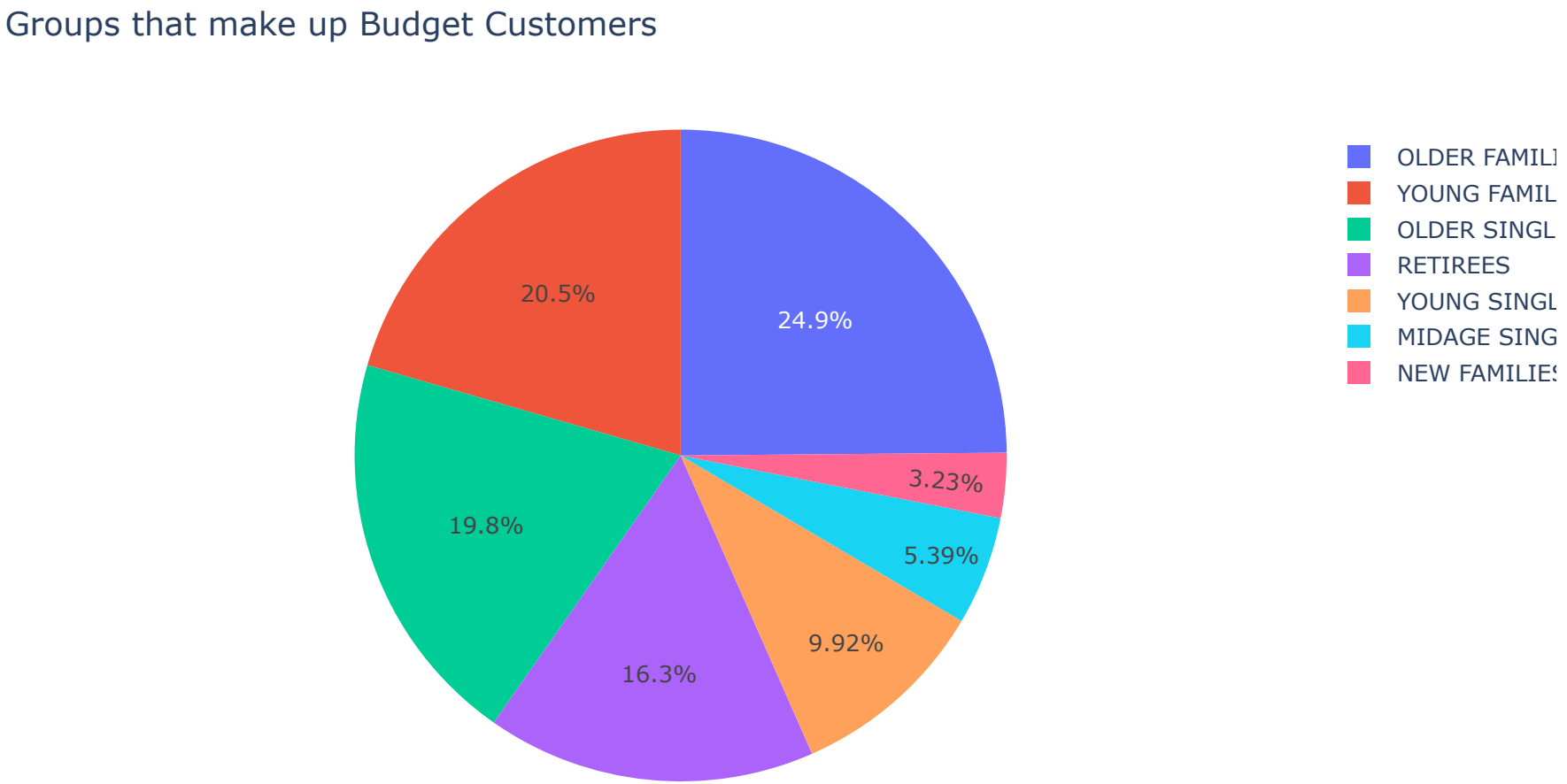


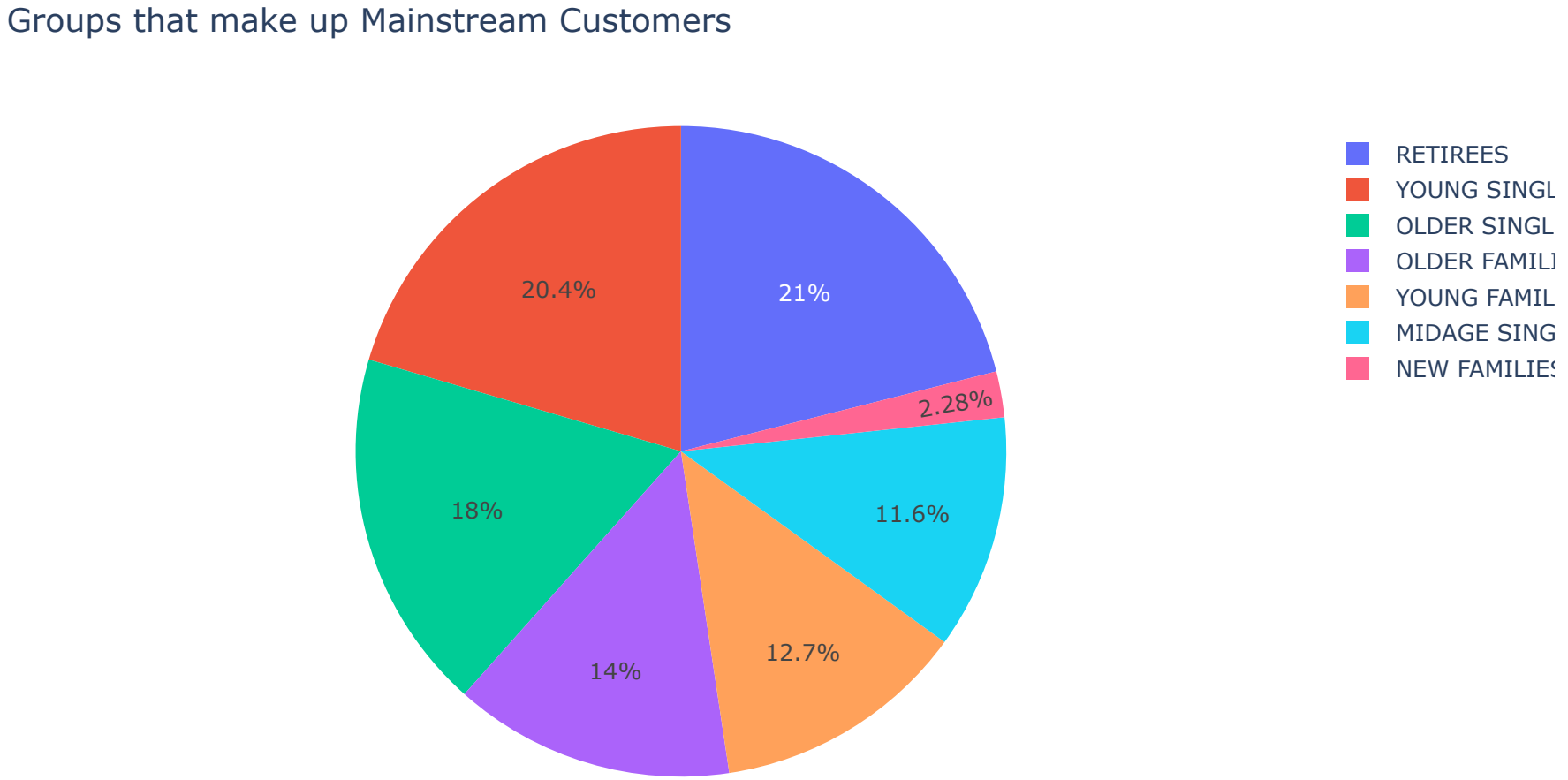
In [13]:
```
grouped=df_no_outliers.groupby('PREMIUM_CUSTOMER')['TOT_SALES'].sum().reset_index()
px.bar(grouped,x='PREMIUM_CUSTOMER',y='TOT_SALES', title='Type of Customers vs ')
```



In [14]:
```
premium_to_stage=df.groupby(['PREMIUM_CUSTOMER','LIFESTAGE']).size().reset_index(name='count')

for i in df['PREMIUM_CUSTOMER'].unique():
    data=premium_to_stage[premium_to_stage['PREMIUM_CUSTOMER']==i]
    fig=px.pie(data,names='LIFESTAGE', values='count', title=f'Groups that make up {i} Customers')
    fig.show()
```




Groups that make up Budget Customers
24.9%
20.5%
19.8%
16.3%
9.92%
5.39%
3.23%
OLDER FAMIL
YOUNG FAMIL
OLDER SINGL
RETIREES
YOUNG SINGL
MIDAGE SING
NEW FAMILIES



In [15]:
```
df['weight_g']= df['PROD_NAME'].str[-4:-1]
```

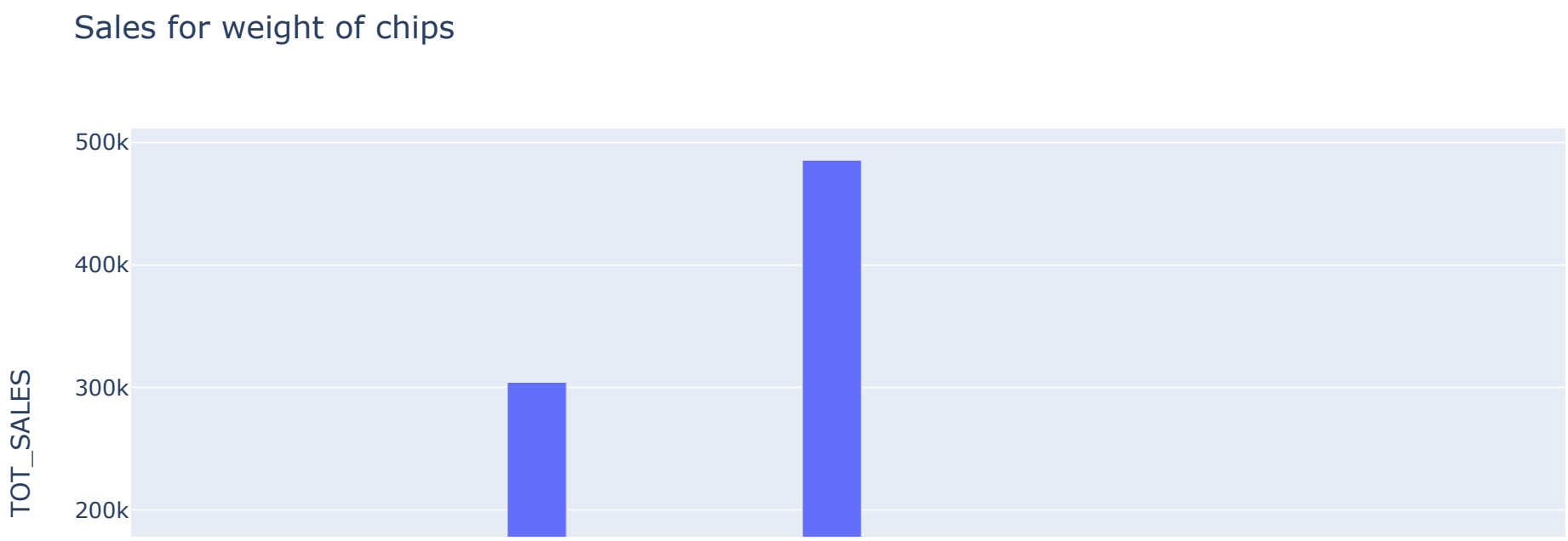
In [16]:
```
df['weight_g'].unique()
```

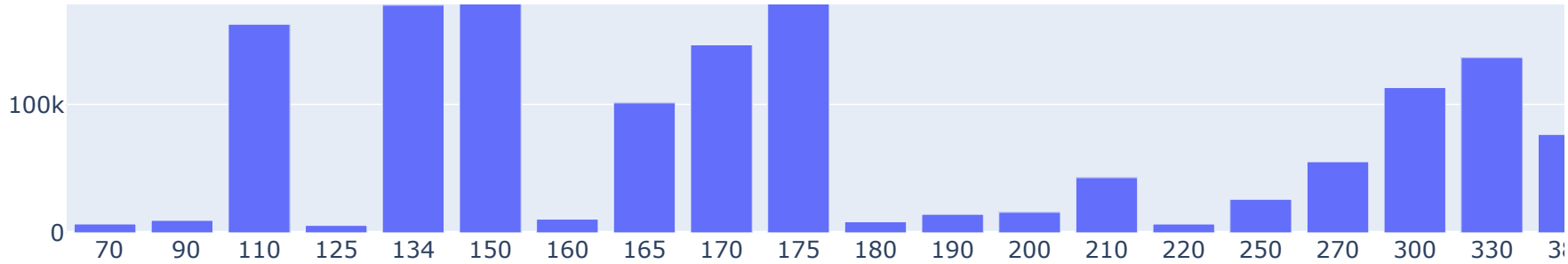
Out[16]:
```
array(['175', '160', '170', '150', '300', '165', '380', '330', '110',
       '210', '180', '200', '134', '270', '220', '125', ' 70', 'Sal',
       '250', ' 90', '190'], dtype=object)
```

In [17]:
```
df['weight_g']=df['weight_g'].replace('Sal', np.nan)
```

In [18]:
```
df_weight= df.dropna()
```

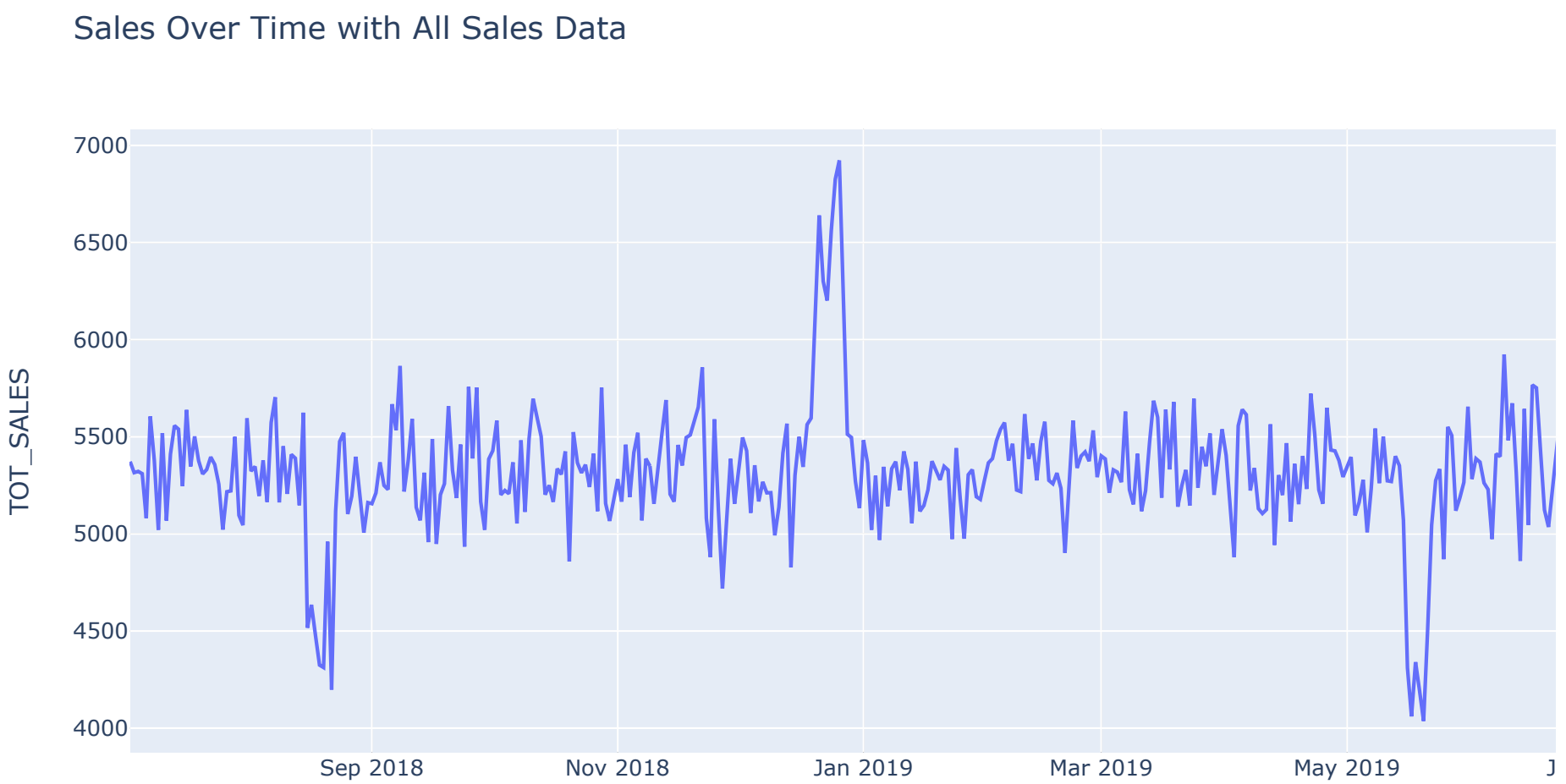
In [19]:
```
groups= df_weight.groupby('weight_g')['TOT_SALES'].sum().reset_index()


px.bar(groups,x='weight_g', y='TOT_SALES', title='Sales for weight of chips')
```



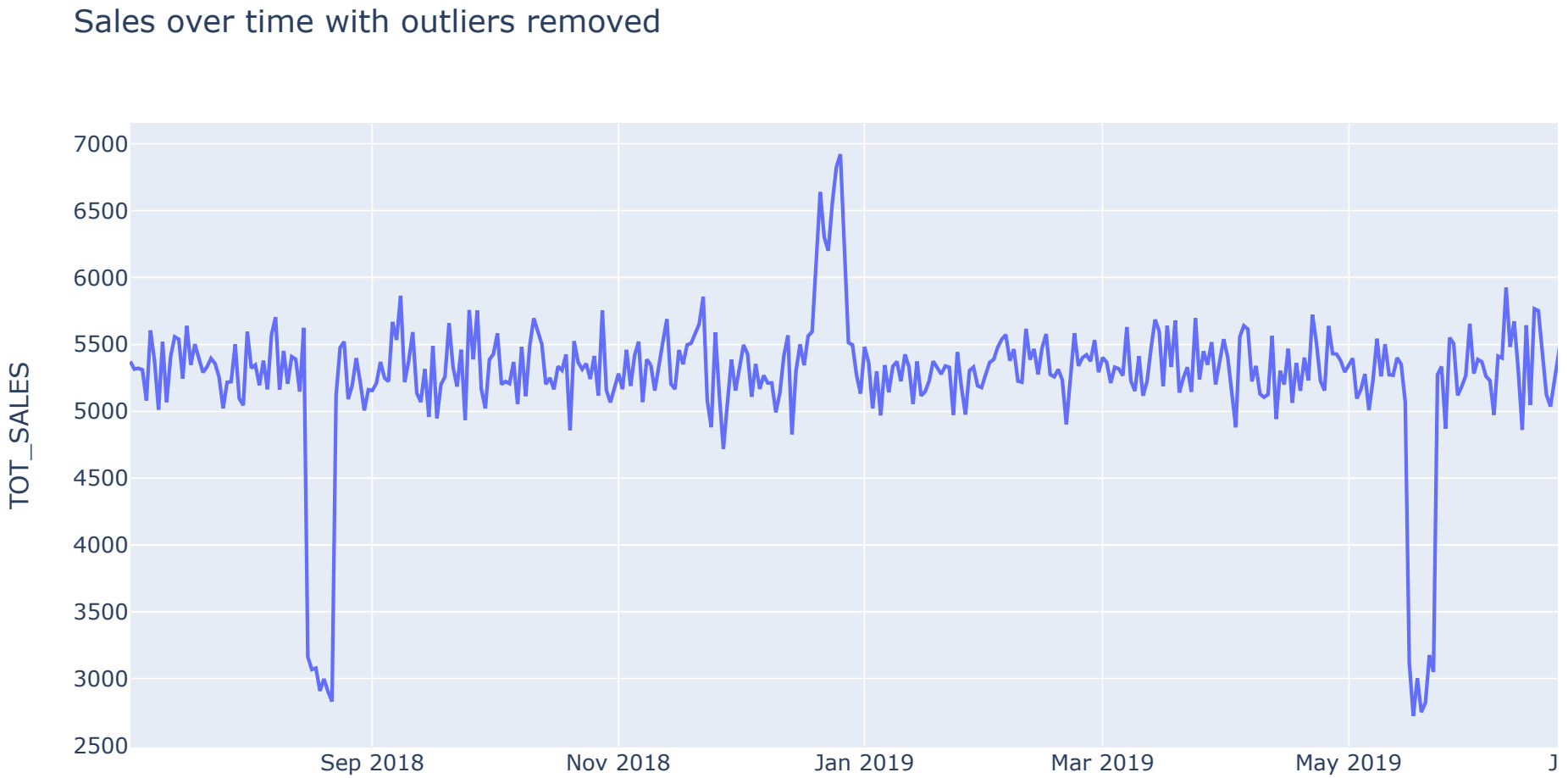


In [20]:
```
date=df.groupby('DATE')['TOT_SALES'].sum().reset_index()
```

In [21]:
```
px.line(date,x='DATE',y='TOT_SALES', title='Sales Over Time with All Sales Data')
```

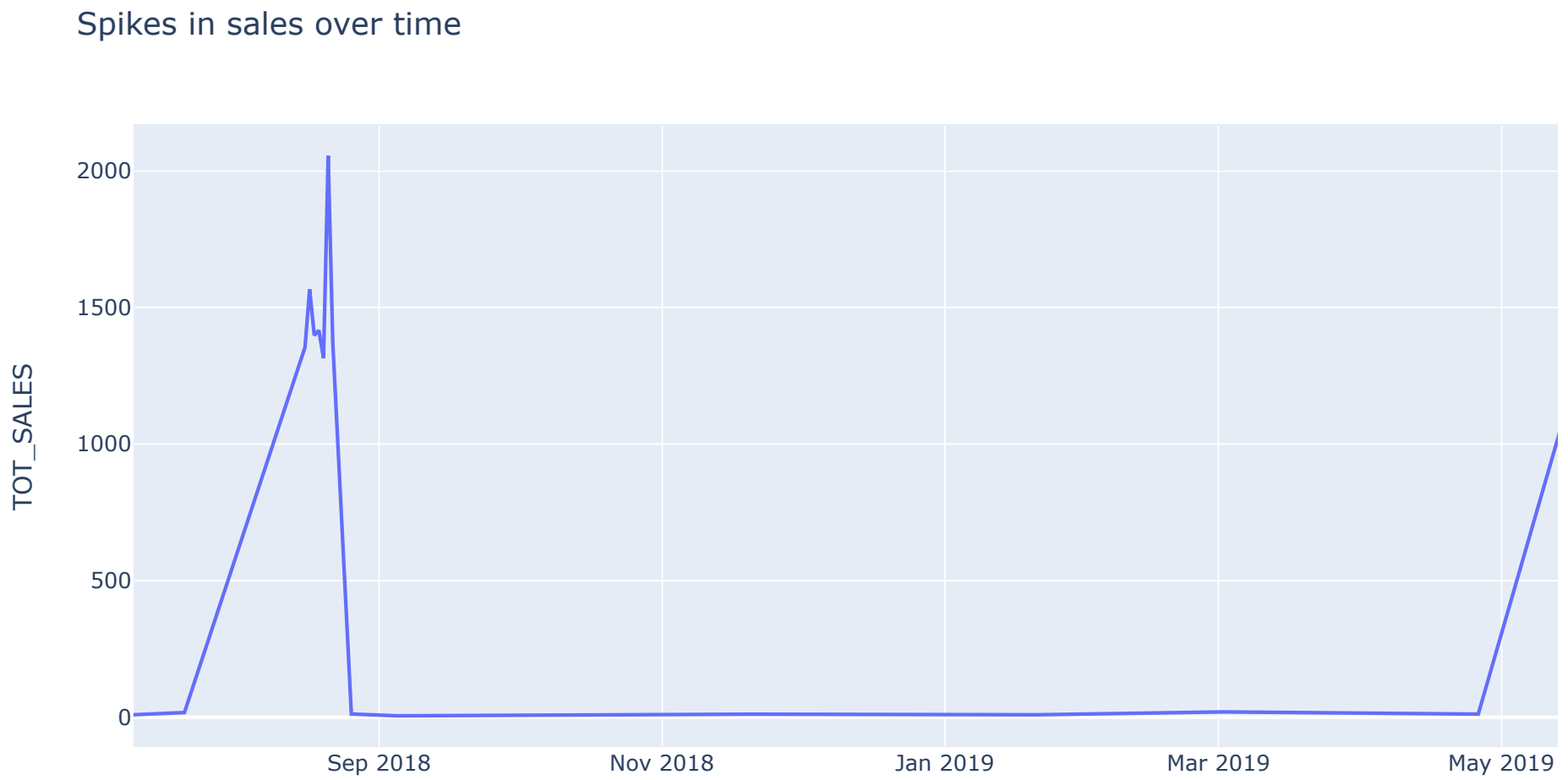


In [22]:
```
date_no_outlier=df_no_outliers.groupby('DATE')['TOT_SALES'].sum().reset_index()
px.line(date_no_outlier,x='DATE',y='TOT_SALES', title='Sales over time with outliers removed')
```



In [23]:
```
outliers = df[(df['PROD_QTY']>upper_bound)]
```

In [24]:
```
# CHeck When Outliers Happen

date_outliers=outliers.groupby('DATE')['TOT_SALES'].sum().reset_index()
px.line(date_outliers,x='DATE',y='TOT_SALES', title='Spikes in sales over time')
```



Spikes might be due to preparations for Labor Day (Sept 4) and Memorial Day (May 29)

DATE int64 STORE_NBR int64 LYLTY_CARD_NBR int64 TXN_ID int64 PROD_NBR int64 PROD_NAME object PROD_QTY int64 TOT_SALES float64 dtype: object LYLTY_CARD_NBR int64 LIFESTAGE object PREMIUM_CUSTOMER

In [ ]: