Importing Libraries & Reading in the Data

In [1]:

```
# Importing connections
import pandas as pd
pd.options.mode.chained_assignment = None  # default='warn'
import numpy as np
from matplotlib import pyplot as plt
import sys
import os
import seaborn as sns
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
import datetime as dt
```

Checking Working Directory before Importing Data

In [2]:

```
os.getcwd()
```

Out[2]:

```
'C:\\Users\\cxs1rgf\\OneDrive - The Home Depot\\Desktop\\GitHub\\Case-Study-DS2\\Code'
```

In [3]:

```
att = pd.read_excel('..\data\Core DS Case Study Data MASTER.xlsx', sheet_name = 'PRODUC
dsales = pd.read_excel('..\data\Core DS Case Study Data MASTER.xlsx', sheet_name = 'MON
calendar = pd.read_excel('..\data\Core DS Case Study Data MASTER.xlsx', sheet_name = 'F
qsales = pd.read_excel('..\data\Core DS Case Study Data MASTER.xlsx', sheet_name = 'HIS
```

Prepping Data

Formating calendar to have year month with leading 0

In [4]:

```
calendar['years'] = calendar['FISCAL_YEAR'].astype(str)
calendar['months'] = calendar['FISCAL_MONTH_NBR'].astype(str)
calendar['month']=calendar['months'].apply(lambda x: '{0:0>2}'.format(x))
calendar['yr_month'] = calendar['years'] +  calendar['month']

calendar = calendar.loc[calendar['yr_month']<'202004']
calendar.head()
```

Out[4]:

	FISCAL_YEAR	FISCAL_MONTH_NBR	FISCAL_MONTH_BEGIN_DT	FISCAL_MONTH_END_DT	years	mont
0	2017	1	2017-01-30	2017-02-26	2017	
1	2017	2	2017-02-27	2017-03-26	2017	
2	2017	3	2017-03-27	2017-04-30	2017	
3	2017	4	2017-05-01	2017-05-28	2017	
4	2017	5	2017-05-29	2017-06-25	2017	

Verify that each product has an entry for each month

Forecasting models like exponential smoothing require consistent entries, if a product is missing a month, we will want to impute a value

To do this we will merge in the fiscal Calendar using a right join (calendar data on RHS.) If the length of the merged table is larger than the sales table (LHS), then we know we have missing Fiscal Weeks that will need to be accounted for.

We also know from excel analysis that the latest date any product sold is March of 2020, we will limit the calendar to match with our existing data

We find 5 missing FW for Sales Quantity using the Historical Customer Sales table and no missing months for the Monthly Category Sales data.

We want to ultimately look at this at the Category view. We will see after we sum up to the Category level, if we are missing FW. If we are still missing values we will merge in the monthly sales estimates at the Category Level

Grouping Data

To look at the data quickly we will group by the product category name and then graph current sales, this will give us a feel for the current sales history.

We will use the quantity of items instead of the sales value. Sales value is susceptible to inflation whereas quantity is not.

To do this, we will first join in the Product Attributes 'PRODUCT_CATEGORY_NAME' into the sales quantity table, then perform a groupby product category name summing monthly sales

To verify the merge is performed correctly, we will check the length of the qsales table before and after. If we see a reduction in rows, this indicates that some products may not have a category assigned to them

In [5]:

```
qsales_sc = pd.merge(qsales, att[['PRODUCT_CATEGORY_NAME','PRODUCT_ID','CURR_RETL_PRICE
```

In [6]:

```
print(len(qsales_sc))
print(len(qsales))
```

```
6273
6273
```

In [7]:

```
qsales_sc.sort_values('PRODUCT_ID').head()
```

Out[7]:

	PRODUCT_ID	FISCAL_MONTH_END_DT	GROSS_SALES_QTY	NET_SALES_QTY	PRODUCT_CATEGORY_NA
0	113065	2019-02-03	6515	6240	PROJECT PAI

	PRODUCT_ID	FISCAL_MONTH_END_DT	GROSS_SALES_QTY	NET_SALES_QTY	PRODUCT_CATEGORY_NA
20	113065	2018-07-29	8259	7867	PROJECT PAN
21	113065	2018-01-28	8522	8198	PROJECT PAN
22	113065	2017-02-26	8803	8441	PROJECT PAN
23	113065	2017-06-25	8855	8454	PROJECT PAN

Now that the data is merged in, we will perform the group by to find sales at the category level. It is not inherently clear if Net Sales is a better metric than Gross Sales or if it will make a difference at the category level. We will graph both to see if there is a large difference between the two that may need to be accounted for

In [8]:

```
qsales_sc1 = qsales_sc.groupby(['PRODUCT_CATEGORY_NAME','FISCAL_MONTH_END_DT']).sum(['N
```

In [9]:

```
qsales_sc1.head()
```

Out[9]:

	PRODUCT_CATEGORY_NAME	FISCAL_MONTH_END_DT	GROSS_SALES_QTY	NET_SALES_QTY	CURR_R
0	1HDL KITCHEN FAUCETS	2017-02-26	25636	21794	
180	SHINGLES	2017-02-26	270709	231907	
144	PROJECT PANELS	2017-02-26	188748	181290	
252	WALL TILE	2017-02-26	92154	75551	
72	GAS SNOW BLOWERS	2017-02-26	1193	934	

Mapping in the Month/Day from Calendar1 Variable previously made

In [10]:

```
qsales_sc2 = pd.merge(qsales_sc1, calendar[['years','month','yr_month','FISCAL_MONTH_EN
len(qsales_sc2)
```

Out[10]: 291

We have the data at the Category level, we will now verify if our FW are still missing & if so which categories.

We still have 27 missing FW, we will merge in the Monthly Sales Estimate to find the missing Fiscal Weeks & divide sales by price to impute the quantity

Finding Length of QSales SC2 without the extra rows from the calendar table

In [16]:

```
qsales_sc2 = pd.merge(qsales_sc1, calendar[['years','month','yr_month','FISCAL_MONTH_EN
len(qsales_sc2)
```

Out[16]: 288

Repeat Calendar treatment for quantity sales table for dollar sales table

In [17]:
```
dsales1 = pd.merge(dsales, calendar[['years','month','yr_month','FISCAL_MONTH_END_DT']]
len(dsales1)
```

Out[17]: 315

Merging in the Category sales variable to imput sales quantity, t2 should now be 5 rows longer than qsales_sc2

In [18]:
```
a = len(qsales_sc1)
b = len(dsales1)
diff = b-a
print("There is (are) " + str(diff) + " missing FW in this dataset")
```

```
There is (are) 27 missing FW in this dataset
```

In [19]:
```
t2 = pd.merge(qsales_sc2, dsales1[['yr_month','PRODUCT_CATEGORY_NAME','GROSS_SALES_ESTI
len(t2)
```

Out[19]: 315

We can see the missing values, the associated year/month & the category

In [21]:
```
t2.tail()
```

Out[21]:

	PRODUCT_CATEGORY_NAME	FISCAL_MONTH_END_DT	GROSS_SALES_QTY	NET_SALES_QTY	CURR_R
310	GLOVES,SAFETY APPAREL	NaT	NaN	NaN	
311	PROJECT PANELS	NaT	NaN	NaN	
312	SHINGLES	NaT	NaN	NaN	
313	WALL TILE	NaT	NaN	NaN	
314	WINTER APPAREL	NaT	NaN	NaN	

Now we have a list of values where there are missing months

Due to time constraints we will impute a value of 0 for these dates, a better method would use the average of the surrounding months

In [22]:
```
slice1 = t2[['PRODUCT_CATEGORY_NAME','GROSS_SALES_ESTIMATE','yr_month','years']]
missings = slice1[slice1['GROSS_SALES_ESTIMATE'].isna()]
missings.sort_values('PRODUCT_CATEGORY_NAME').head()
```

Out[22]:

	PRODUCT_CATEGORY_NAME	GROSS_SALES_ESTIMATE	yr_month	years
288	1HDL KITCHEN FAUCETS	NaN	202001	NaN

	PRODUCT_CATEGORY_NAME	GROSS_SALES_ESTIMATE	yr_month	years
306	1HDL KITCHEN FAUCETS	NaN	202003	NaN
297	1HDL KITCHEN FAUCETS	NaN	202002	NaN
289	CABLE TIES	NaN	202001	NaN
307	CABLE TIES	NaN	202003	NaN

In [23]:

```
t2 = t2.fillna(0)
```

Cleaning up the data set after the several merges

In [24]:

```
df = t2[['PRODUCT_CATEGORY_NAME','GROSS_SALES_QTY','NET_SALES_QTY','CURR_RETL_PRICE','y
df.head(10)
```

Out[24]:

	PRODUCT_CATEGORY_NAME	GROSS_SALES_QTY	NET_SALES_QTY	CURR_RETL_PRICE	yr_month	yea
236	GAS SNOW BLOWERS	1193.0	934.0	12260.94	201701	20
239	GLOVES,SAFETY APPAREL	2201.0	2150.0	19.75	201701	20
238	CABLE TIES	73750.0	72207.0	210.15	201701	20
237	WALL TILE	92154.0	75551.0	211.54	201701	20
235	1HDL KITCHEN FAUCETS	25636.0	21794.0	1215.82	201701	20
234	SPRAY PAINT	408268.0	393733.0	96.67	201701	20
232	SHINGLES	270709.0	231907.0	3297.01	201701	20
233	PROJECT PANELS	188748.0	181290.0	280.41	201701	20
222	GAS SNOW BLOWERS	682.0	518.0	12260.94	201702	20
223	GLOVES,SAFETY APPAREL	2994.0	2935.0	19.75	201702	20

Graphing Sales by Category

Now we will graph by subclass, we find little difference between Net & Gross sales quantity at the class level. We will proceed using the net value as this will be a better representation of how much money Home Depot can expect to get from our new

We find one large outlier in Gas Snow Blowers & Gloves & Saftey Apparel have a highly seasonal shape peaking in May each year.

Since we are looking at winter apparel, we will want to use the Gloves and Saftey apparel shape & rate during the Gas Snow Blowers season time frame. We will need to smooth out the peak in Gas & Snow Blowers to get a better idea of how Gas Snow Blowers Shape looks.

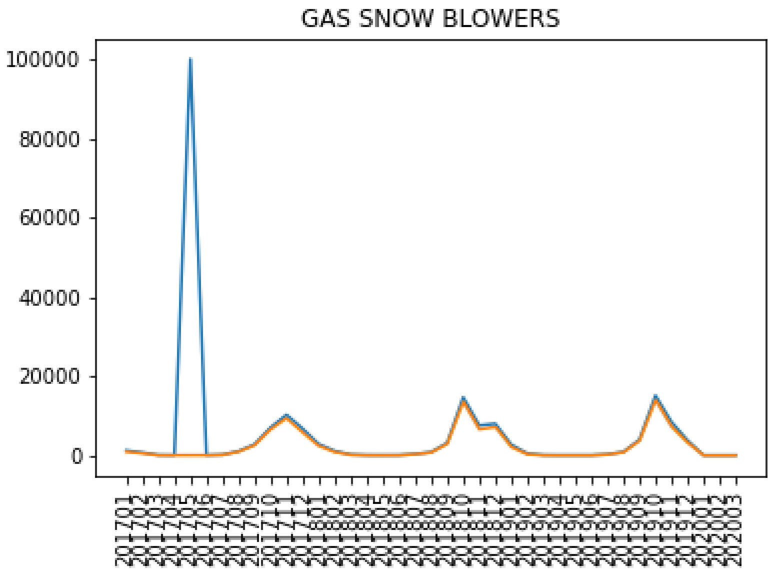
In [25]:

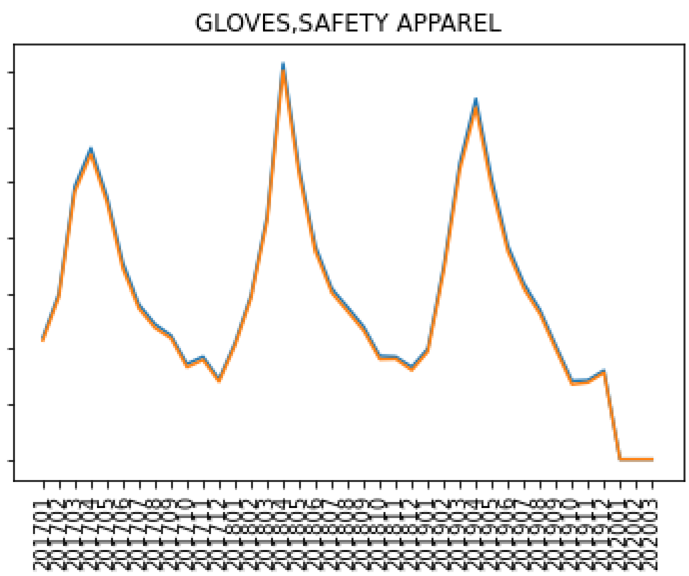
```
each_category = df['PRODUCT_CATEGORY_NAME'].unique()
```

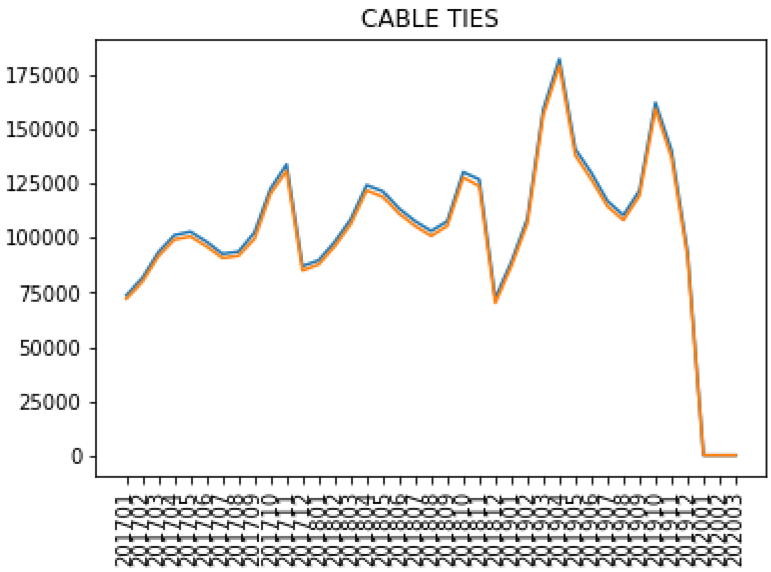
```
each_category
```

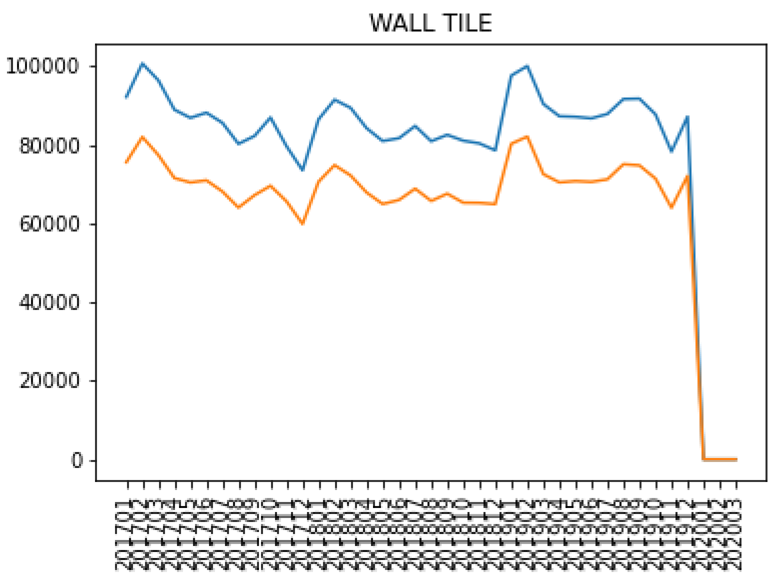
Out[25]:
```
array(['GAS SNOW BLOWERS', 'GLOVES,SAFETY APPAREL', 'CABLE TIES',
       'WALL TILE', '1HDL KITCHEN FAUCETS', 'SPRAY PAINT', 'SHINGLES',
       'PROJECT PANELS', 'WINTER APPAREL'], dtype=object)
```

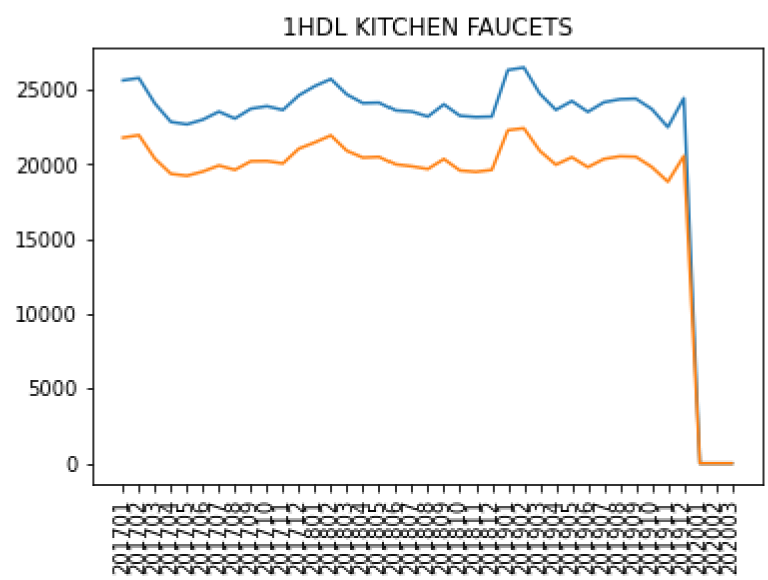
In [26]:
```python
for category in each_category:
    df3 = df.loc[ (df['PRODUCT_CATEGORY_NAME'] == category)]
    x = df3['yr_month']
    y = df3[['GROSS_SALES_QTY','NET_SALES_QTY']]

    plt.figure()
    plt.title(category)
    plt.plot(x,y)
    plt.xticks(rotation = 90)
    plt.show()
```

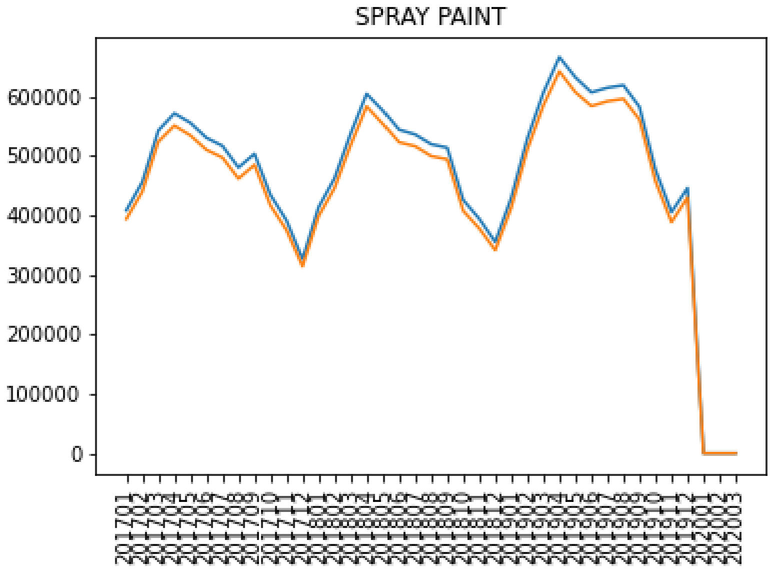


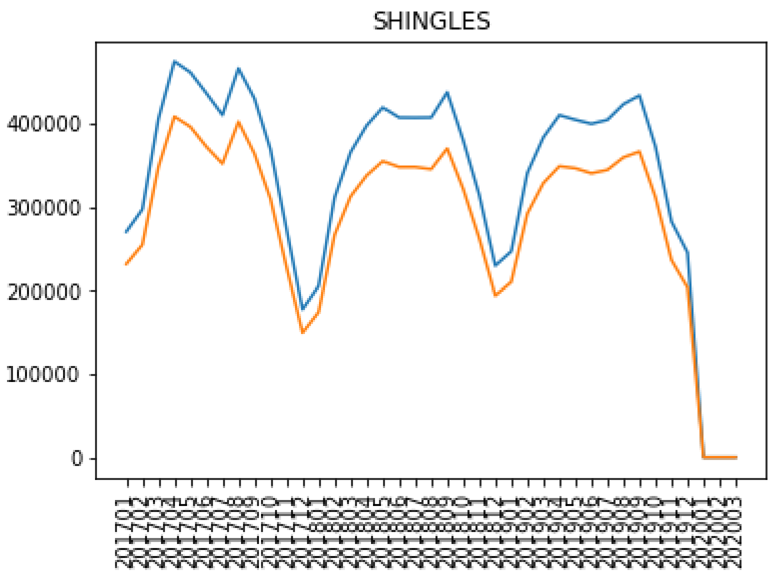


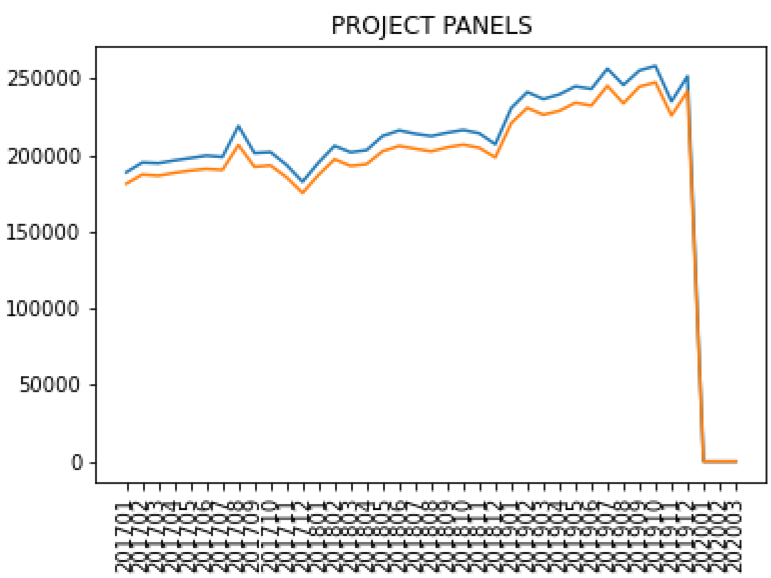

CABLE TIES
175000
150000
125000
100000
75000
50000
25000
0

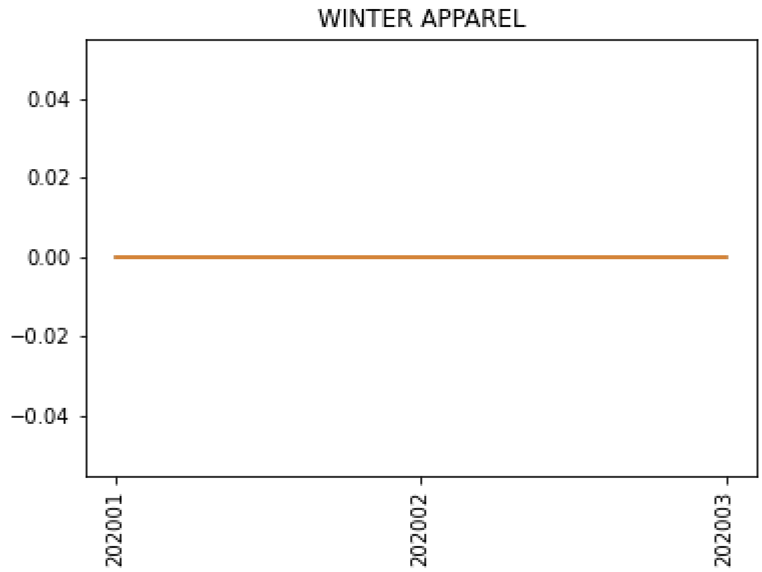

WALL TILE
100000
80000
60000
40000
20000
0


1HDL KITCHEN FAUCETS
25000
20000
15000
10000
5000
0


SPRAY PAINT
600000
500000
400000
300000
200000
100000
0


SHINGLES
400000
300000
200000
100000
0


PROJECT PANELS
250000
200000
150000
100000
50000
0



Tracking down the Outlier in *Gas Snow Blowers* Category

We find the culprit is 2017-06-25 where there was a sale of 100,034 units, we will smooth this out by using the average of the surrounding two months

In [27]:
```
df.loc[(df['PRODUCT_CATEGORY_NAME']== 'GAS SNOW BLOWERS') & (df['years'] == '2017')]
```

Out[27]:

	PRODUCT_CATEGORY_NAME	GROSS_SALES_QTY	NET_SALES_QTY	CURR_RETL_PRICE	yr_month	yea
236	GAS SNOW BLOWERS	1193.0	934.0	12260.94	201701	20
222	GAS SNOW BLOWERS	682.0	518.0	12260.94	201702	20
70	GAS SNOW BLOWERS	71.0	33.0	14692.53	201703	20
7	GAS SNOW BLOWERS	31.0	20.0	14692.53	201704	20
23	GAS SNOW BLOWERS	100034.0	30.0	14692.53	201705	20
30	GAS SNOW BLOWERS	38.0	29.0	15928.60	201706	20
54	GAS SNOW BLOWERS	138.0	125.0	15928.60	201707	20
14	GAS SNOW BLOWERS	1001.0	935.0	15928.60	201708	20
34	GAS SNOW BLOWERS	2742.0	2553.0	15928.60	201709	20
129	GAS SNOW BLOWERS	6977.0	6539.0	15928.60	201710	20
225	GAS SNOW BLOWERS	10217.0	9408.0	15928.60	201711	20
281	GAS SNOW BLOWERS	6731.0	5870.0	15928.60	201712	20

Imputing the Value

In [28]:
```
average = df['GROSS_SALES_QTY'].loc[
    (df['PRODUCT_CATEGORY_NAME'] == 'GAS SNOW BLOWERS') &
    (
        (df['yr_month'] == '201704') |
```

```
        (df['yr_month'] == '201706')
    )].mean()

df['GROSS_SALES_QTY'].loc[
    (df['yr_month'] == '201705') &
    (df['PRODUCT_CATEGORY_NAME']== 'GAS SNOW BLOWERS')] = average
```

Now that the outlier has been removed we see a sharp seasonal shape starting in September and ending May, there also appeares to be a slight upward trend of Gas Snow Blowers.

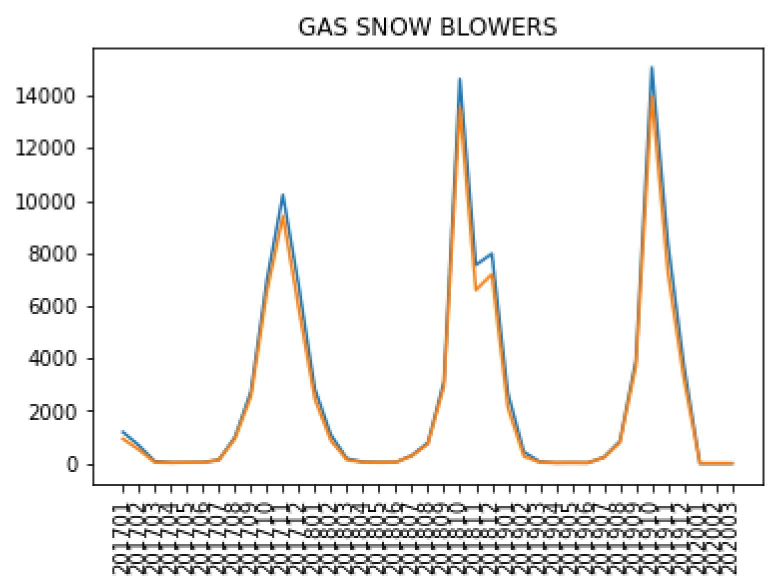
In constrast, Gloves & Safety Apparel appear to have a seasonal shape, but no trend, with peaks at about 6,500 units and an average rate of sale of 4,000 units. Snow Blowers have a both higher peaks and average. The average rate of sale for Snow Blowers is around 8,000 units with peaks of 10,000 - 14,000 units.

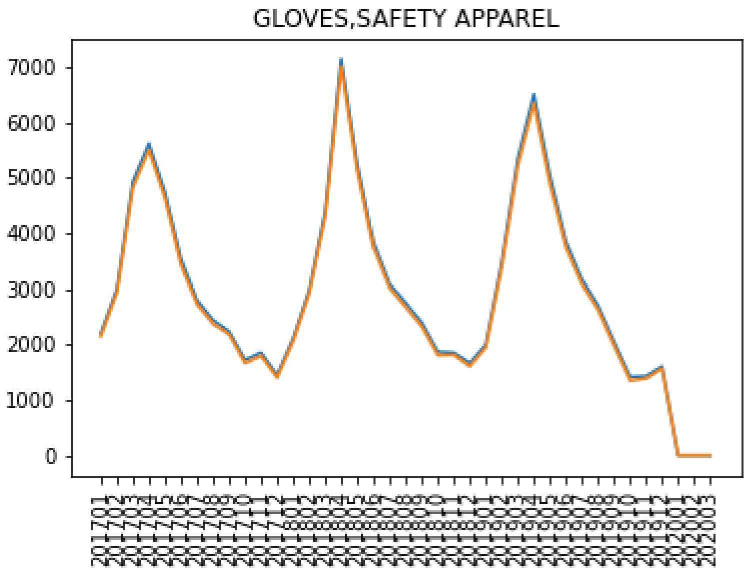
To forecast our winter apparel, we will use the seasonal shape of Snow Blowers and Scale the forecast down to the Gloves & Saftey Apparel Rates.

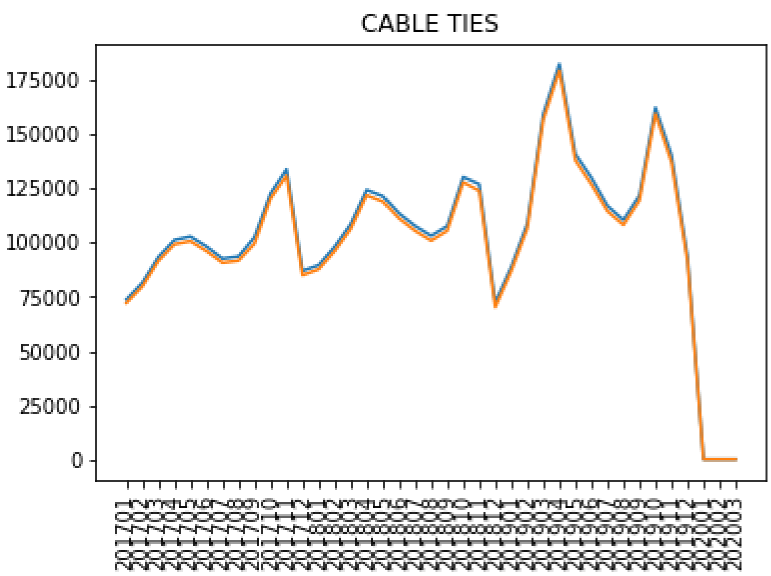
Revisualizing the Data with the Outlier Corrected

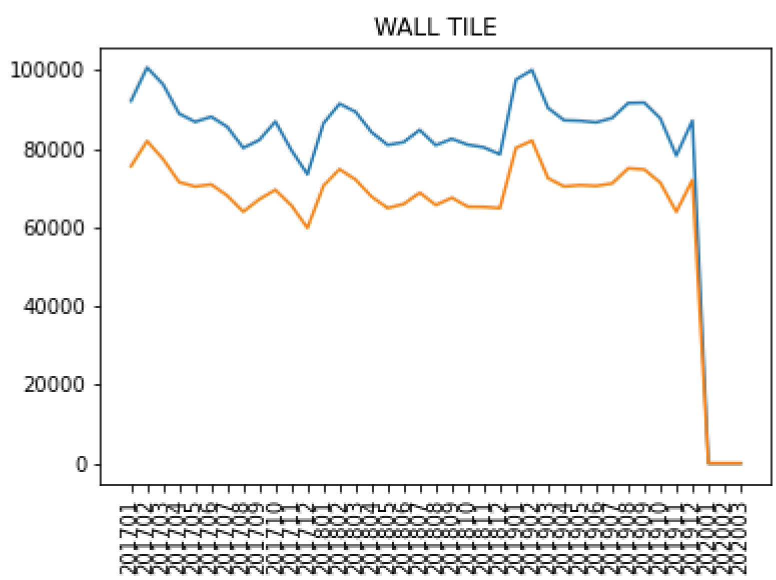
In [29]:

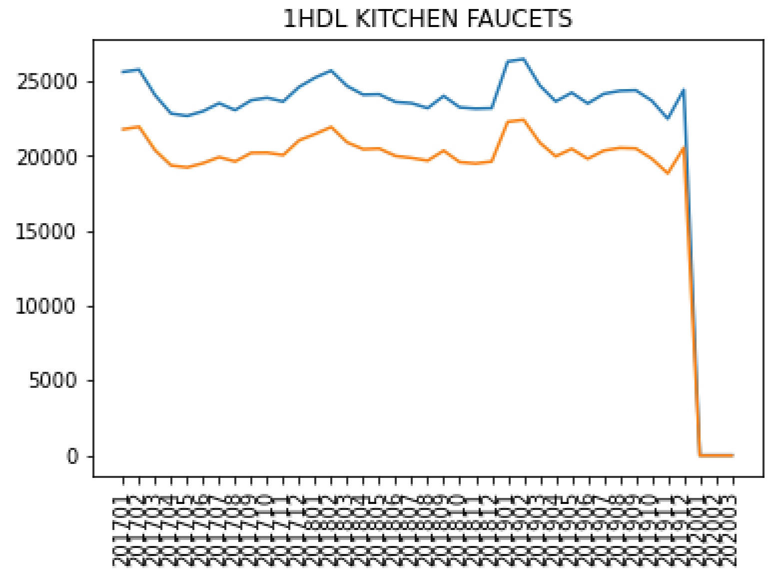
```
for category in each_category:
    df3 = df.loc[ (df['PRODUCT_CATEGORY_NAME'] == category)]
    x = df3['yr_month']
    y = df3[['GROSS_SALES_QTY','NET_SALES_QTY']]

    plt.figure()
    plt.title(category)
    plt.plot(x,y)
    plt.xticks(rotation = 90)
    plt.show()
```

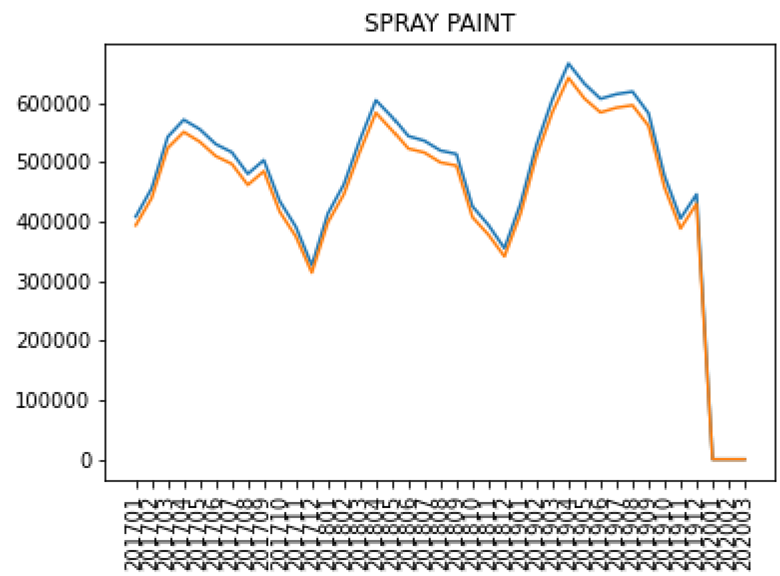


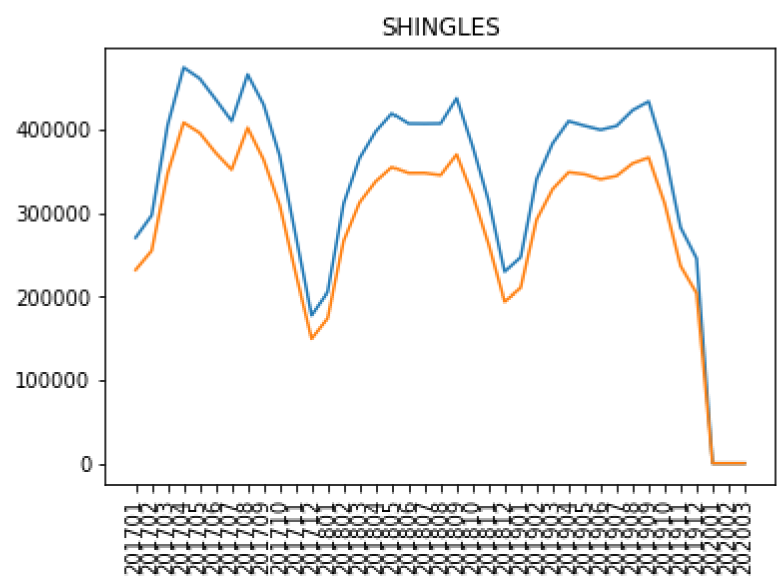

GLOVES,SAFETY APPAREL
7000
6000
5000
4000
3000
2000
1000
0

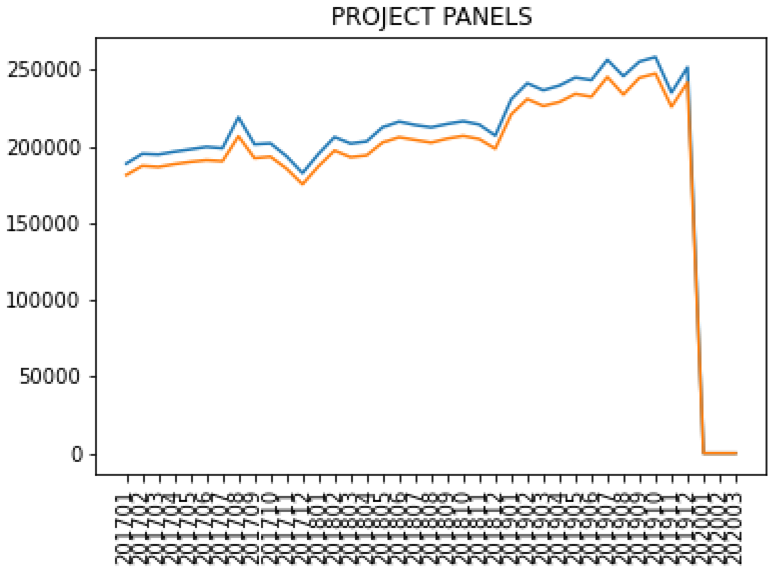

CABLE TIES
175000
150000
125000
100000
75000
50000
25000
0

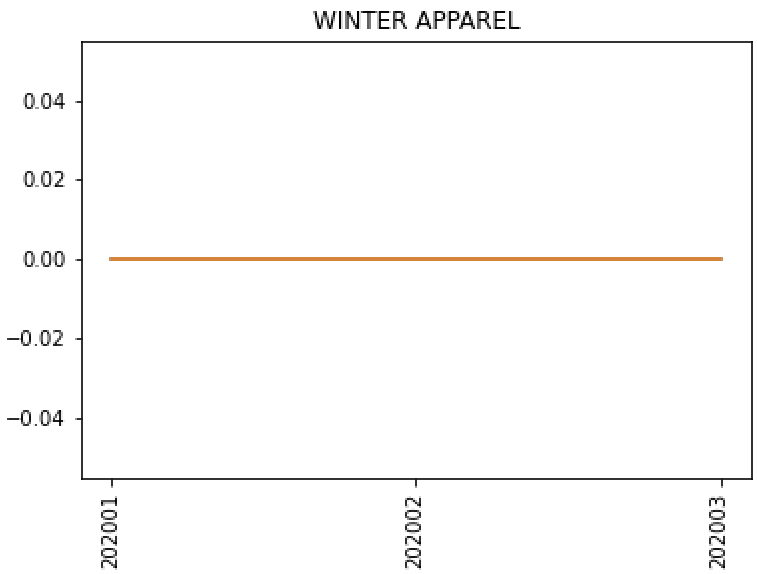

WALL TILE
100000
80000
60000
40000
20000
0


1HDL KITCHEN FAUCETS
25000
20000
15000
10000
5000
0


SPRAY PAINT
600000
500000
400000
300000
200000
100000
0


SHINGLES
400000
300000
200000
100000
0





Decomposing the Trend & Seasonality from the Reference Categories

We find little tend but do have a seasonality component, we will use a triple smoothing exponential model having a beta set at 0 as our product does have seasonality but does not have trend

In [30]:
```
df['t'] = pd.to_datetime(df['yr_month'], format = '%Y%m')
```

In [31]:
```
df.head()
```

Out[31]:

	PRODUCT_CATEGORY_NAME	GROSS_SALES_QTY	NET_SALES_QTY	CURR_RETL_PRICE	yr_month	yea
236	GAS SNOW BLOWERS	1193.0	934.0	12260.94	201701	20
239	GLOVES,SAFETY APPAREL	2201.0	2150.0	19.75	201701	20

	PRODUCT_CATEGORY_NAME	GROSS_SALES_QTY	NET_SALES_QTY	CURR_RETL_PRICE	yr_month	yea
238	CABLE TIES	73750.0	72207.0	210.15	201701	20
237	WALL TILE	92154.0	75551.0	211.54	201701	20
235	1HDL KITCHEN FAUCETS	25636.0	21794.0	1215.82	201701	20

In [63]:

Visually we see the seasonal shape & the trend, but we will verify using the deasonal_decomposition in the HoltWinters forecasting package

To use this package we will need to change the data into timeseries which we can do by setting the time variable as the index

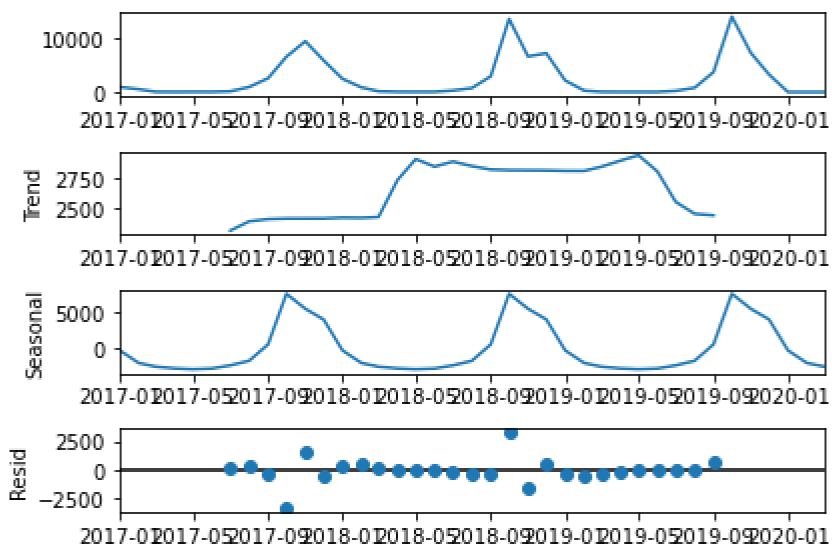
In [32]:

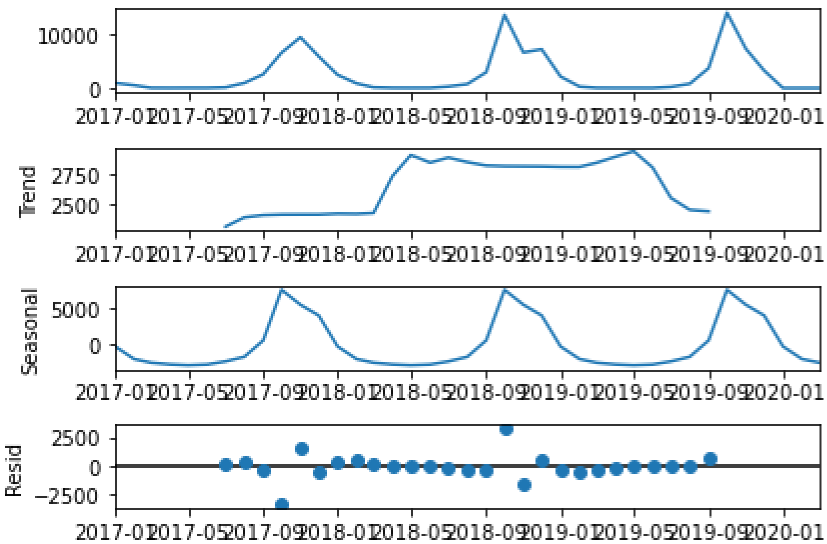
```
ts_snow = df[['NET_SALES_QTY','t']].loc[(df['PRODUCT_CATEGORY_NAME'] == 'GAS SNOW BLOWE
```

In [33]:

```
seasonal_decompose(ts_snow).plot()
```

Out[33]:





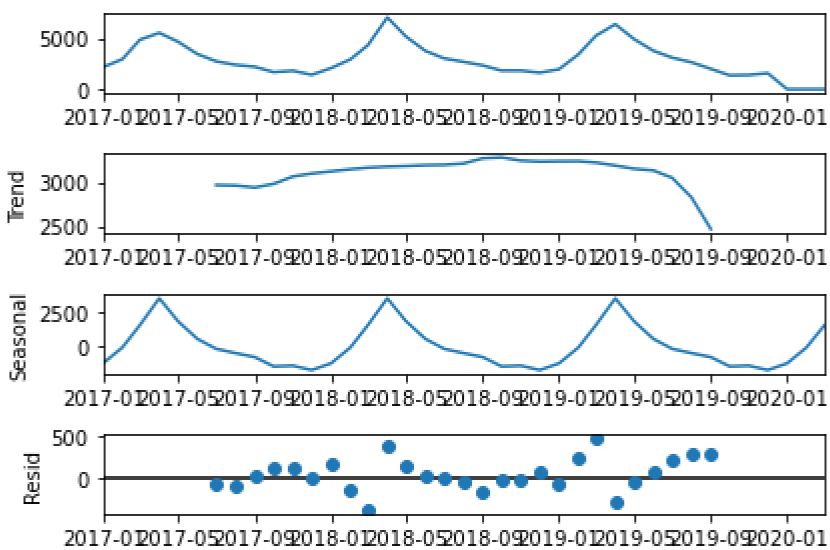
In [34]:

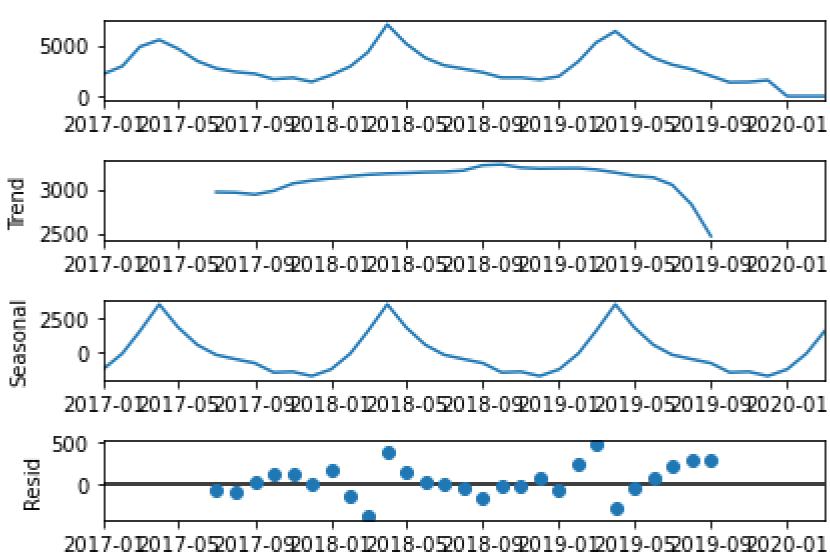
```python
ts_app = df[['NET_SALES_QTY','t']].loc[(df['PRODUCT_CATEGORY_NAME'] == 'GLOVES,SAFETY A
```

In [35]:

```python
seasonal_decompose(ts_app).plot()
```

Out[35]:





Forecasting Winter Apparel

First we will scale the winter apparel to be 1/2 that of Gas Snow Blowers, which will bring the Gas Snow Blowers to a similar rate of sale as the existing apparel in the store

We are going to scale the data down for the winter apparel to match that rate of the apparel by dividing the snowblowers in half

In [36]:

```
winterapparel = df.loc[
    (df['PRODUCT_CATEGORY_NAME']== 'GAS SNOW BLOWERS')]
winterapparel['PRODUCT_CATEGORY_NAME'] = 'WINTER APPAREL 1'
winterapparel.head()
```

Out[36]:

	PRODUCT_CATEGORY_NAME	GROSS_SALES_QTY	NET_SALES_QTY	CURR_RETL_PRICE	yr_month	yea
236	WINTER APPAREL 1	1193.0	934.0	12260.94	201701	20
222	WINTER APPAREL 1	682.0	518.0	12260.94	201702	20
70	WINTER APPAREL 1	71.0	33.0	14692.53	201703	20
7	WINTER APPAREL 1	31.0	20.0	14692.53	201704	20
23	WINTER APPAREL 1	34.5	30.0	14692.53	201705	20

In [37]:

```
winterapparel['GROSS_SALES_QTY'] = winterapparel['GROSS_SALES_QTY']*.5
winterapparel.head()
```

Out[37]:

	PRODUCT_CATEGORY_NAME	GROSS_SALES_QTY	NET_SALES_QTY	CURR_RETL_PRICE	yr_month	yea
236	WINTER APPAREL 1	596.50	934.0	12260.94	201701	20
222	WINTER APPAREL 1	341.00	518.0	12260.94	201702	20
70	WINTER APPAREL 1	35.50	33.0	14692.53	201703	20
7	WINTER APPAREL 1	15.50	20.0	14692.53	201704	20
23	WINTER APPAREL 1	17.25	30.0	14692.53	201705	20

In [62]:

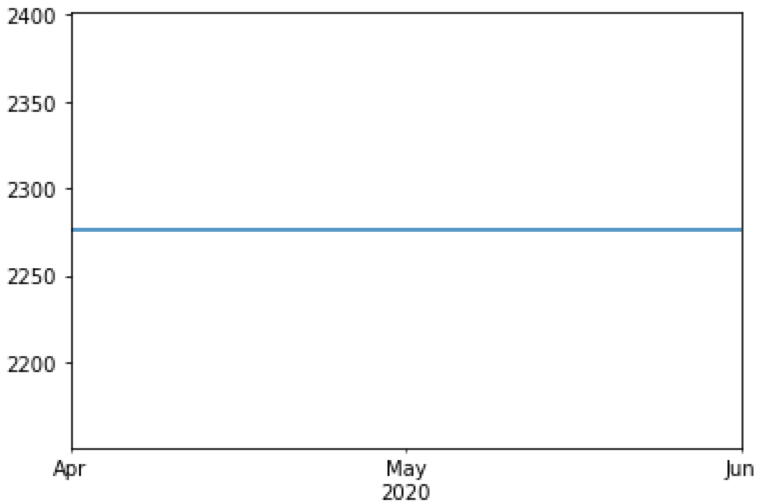
```
tsW = winterapparel[['NET_SALES_QTY','t']].loc[(winterapparel['PRODUCT_CATEGORY_NAME']
```

In [61]:

```
fit1 = SimpleExpSmoothing(tsW, initialization_method="estimated").fit(
    smoothing_level=0.2, optimized=False
)
FWinter = fit1.forecast(3).rename(r"$\alpha=0.2$")
FWinter.plot()
```

```
C:\Users\cxs1rgf\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: Valu
eWarning: No frequency information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```

Out[61]: <AxesSubplot:>

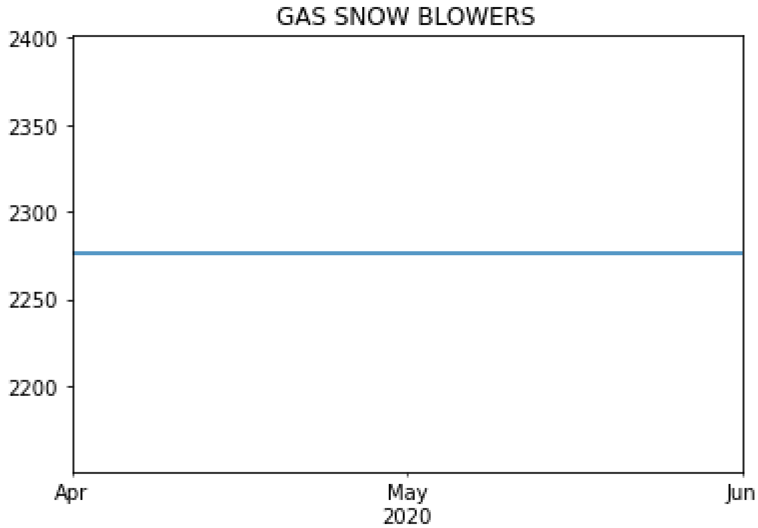


Forecasting Existing Assortment

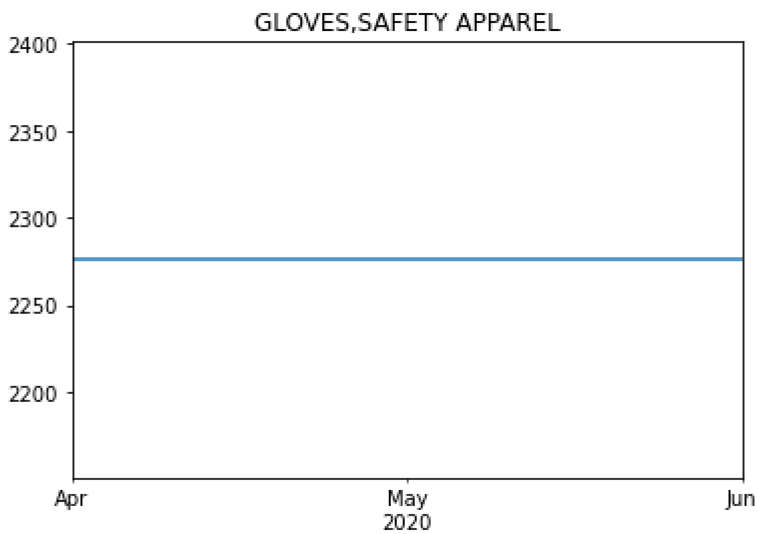
In [67]:

```
for category in each_category:
    df3  = df[['NET_SALES_QTY','t']].set_index('t')



    fit1 = SimpleExpSmoothing(tsW, initialization_method="estimated").fit(
    smoothing_level=0.2, optimized=False
    )
    plt.title(category)
    FWinter = fit1.forecast(3).rename(r"$\alpha=0.2$")
    FWinter.plot()

    plt.show()
```

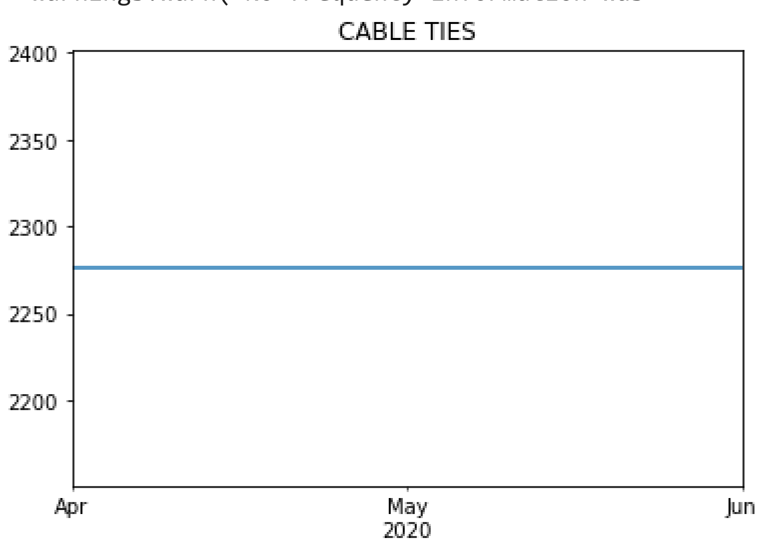
```
C:\Users\cxs1rgf\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: Valu
eWarning: No frequency information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```




```
C:\Users\cxs1rgf\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: Valu
eWarning: No frequency information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```

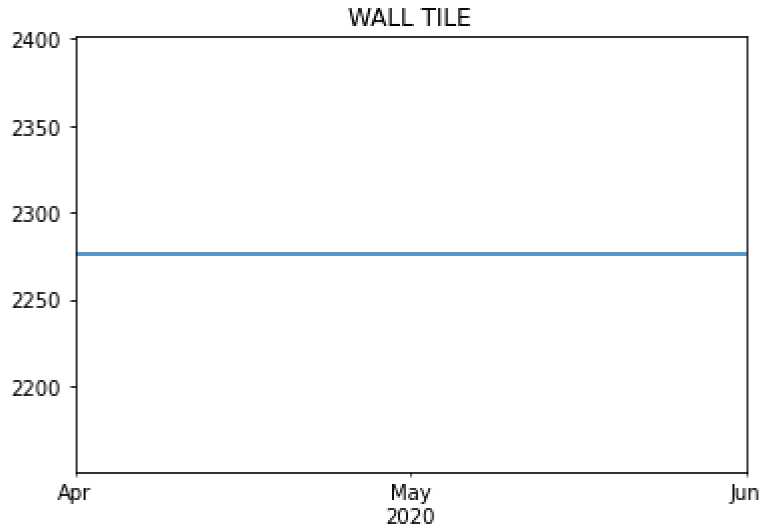



```
C:\Users\cxs1rgf\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: Valu
eWarning: No frequency information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```

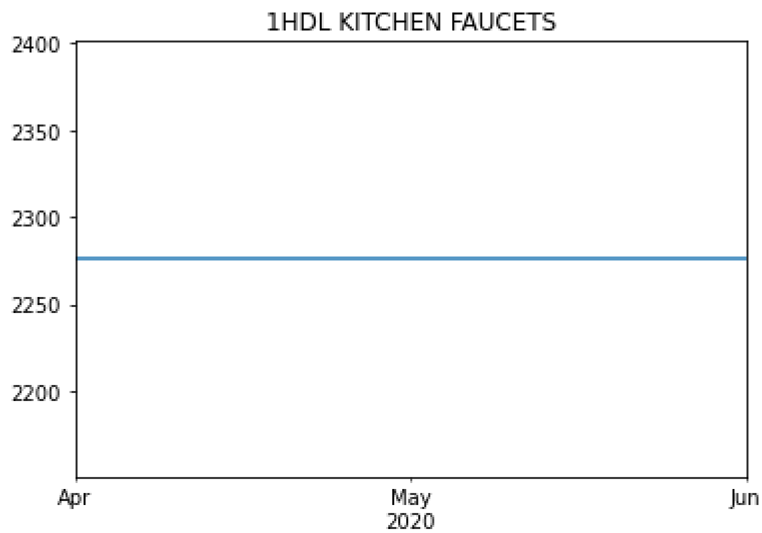



```
C:\Users\cxs1rgf\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: Valu
eWarning: No frequency information was provided, so inferred frequency MS will be used.
```

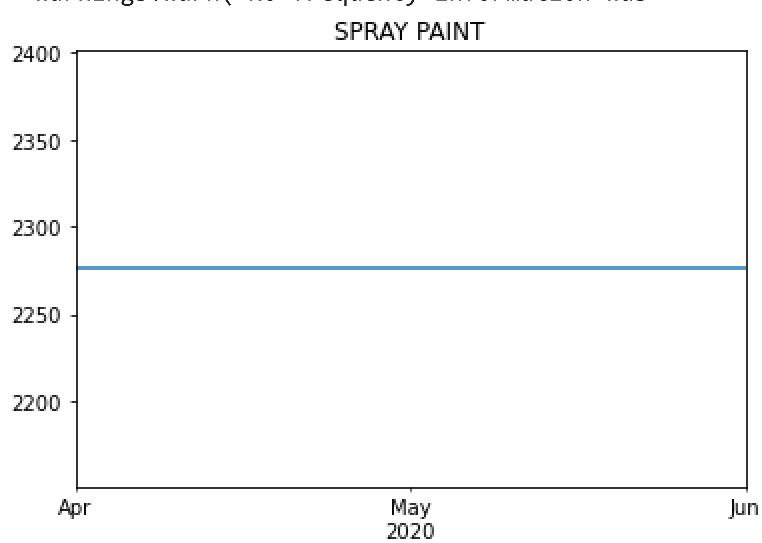
```
  warnings.warn('No frequency information was'
```




```
C:\Users\cxs1rgf\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: Valu
eWarning: No frequency information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```

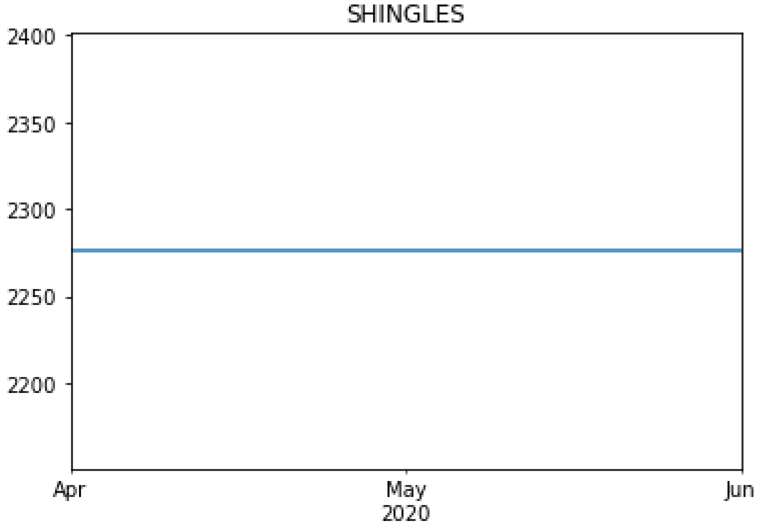



```
C:\Users\cxs1rgf\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: Valu
eWarning: No frequency information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```

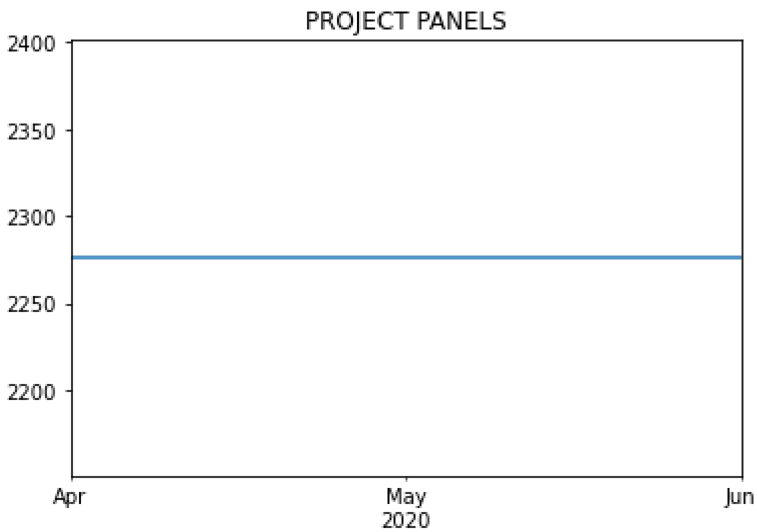



```
C:\Users\cxs1rgf\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: Valu
```

```
eWarning: No frequency information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```




```
C:\Users\cxs1rgf\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: Valu
eWarning: No frequency information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```




```
C:\Users\cxs1rgf\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:524: Valu
eWarning: No frequency information was provided, so inferred frequency MS will be used.
  warnings.warn('No frequency information was'
```