In [857]:
```
import pandas as pd
data = pd.read_csv('New_Wireless_Pipe.txt', sep="|")

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

Out[857]:

	acctno	actdt	deactdt	deactreason	goodcredit	rateplan	dealertype	AGE
0	1176913194483	06/20/1999	NaN	NaN	0	1	A1	58.0
1	1176914599423	10/04/1999	10/15/1999	NEED	1	1	A1	45.0
2	1176951913656	07/01/2000	NaN	NaN	0	1	A1	57.0
3	1176954000288	05/30/2000	NaN	NaN	1	2	A1	47.0
4	1176969186303	12/13/2000	NaN	NaN	1	1	C1	82.0
...	...	...	...	...	...	...	...	...
102250	2673974127660	12/29/2000	NaN	NaN	1	1	A2	50.0
102251	2674189951308	01/15/2001	NaN	NaN	1	2	A1	40.0
102252	2674548796918	01/15/2001	NaN	NaN	1	1	A1	16.0
102253	2675119766018	01/15/2001	NaN	NaN	1	2	B1	76.0
102254	2675135410256	01/17/2001	NaN	NaN	1	1	A1	46.0

102255 rows × 10 columns

In [859]:
```
def data_cut(da,col,list1):
    lab=[]
    for i in range(len(list1)-1):
        if i == 0:
            new = [col , ' < ' , str(list1[i+1])]
            app = ''.join(new)
            lab.append(app)
        elif i == len(list1)-2:
            new = [str(list1[i]) , ' and above']
            app = ''.join(new)
            lab.append(app)
        else:
            new = [str(list1[i]) , ' - ', str(list1[i+1])]
            app = ''.join(new)
            lab.append(app)

    category = pd.cut(da[col], list1,
                          labels=lab)
    return category
```

In [860]:
```
data.head()
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 102255 entries, 0 to 102254
Data columns (total 10 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   acctno       102255 non-null  int64  
 1   actdt        102255 non-null  object 
 2   deactdt      19635 non-null   object 
 3   deactreason  19093 non-null   object 
 4   goodcredit   102255 non-null  int64  
 5   rateplan     102255 non-null  int64  
 6   dealertype   102255 non-null  object 
 7   AGE          94547 non-null   float64
 8   Province     96348 non-null   object 
 9   sales        93650 non-null   float64
dtypes: float64(2), int64(3), object(5)
memory usage: 7.8+ MB
```

In [861]:
```
#1. Explore and describe the dataset briefly. For example, is the acctno un
#is the number of accounts activated and deactivated? When is the earliest
#latest activation/deactivation dates available? And so on….
```

In [862]:
```
print("There are" , data.acctno.count() ,"unique accounts in the dataset")
```

```
There are 102255 unique accounts in the dataset
```

In [863]:
```
print("There are", data.deactdt.isnull().sum(), "activated accounts")
# There are 82620 activated accounts.

print("There are" , data.deactdt.count(), "deactivated accounts")
# There are 19635 deactivated accounts.
```

```
There are 82620 activated accounts
There are 19635 deactivated accounts
```

In [864]:
```
dat = pd.to_datetime(data['actdt'])

print("The earliest activation date is", dat.min())
# The earliest activation date is 1999-01-20

print("The latest activation date is", dat.max())
# The latest activation date is 2001-01-20

dedat = pd.to_datetime(data['deactdt'])
print("The earliest deactivation date is", dedat.min())
print("The latest deactivation date is", dedat.max())
```

```
The earliest activation date is 1999-01-20 00:00:00
The latest activation date is 2001-01-20 00:00:00
The earliest deactivation date is 1999-01-25 00:00:00
The latest deactivation date is 2001-01-20 00:00:00
```

In [865]:
```
data.actdt = pd.to_datetime(data['actdt'])
data.deactdt = pd.to_datetime(data['deactdt'])

date = data[['actdt','deactdt']]


date_table = date.agg(['min', 'max'])
date_table.rename(index={'min': "Earliest Date", 'max': 'Latest Date'},
                  columns={'actdt': 'Activation', 'deactdt': 'Deactivation'
```

Out[865]:

	Activation	Deactivation
Earliest Date	1999-01-20	1999-01-25
Latest Date	2001-01-20	2001-01-20

In [866]:
```
data.value_counts('deactreason')
# Accouding to the deactivated accounts, most reason is NEED, following COM
```

Out[866]:
```
deactreason
NEED    6888
COMP    4722
DEBT    4020
TECH    1767
MOVE    1696
dtype: int64
```

In [867]:
```
data
```

Out[867]:

	acctno	actdt	deactdt	deactreason	goodcredit	rateplan	dealertype	AGE	Provinc
0	1176913194483	1999-06-20	NaT	NaN	0	1	A1	58.0	B
1	1176914599423	1999-10-04	1999-10-15	NEED	1	1	A1	45.0	A
2	1176951913656	2000-07-01	NaT	NaN	0	1	A1	57.0	B
3	1176954000288	2000-05-30	NaT	NaN	1	2	A1	47.0	O
4	1176969186303	2000-12-13	NaT	NaN	1	1	C1	82.0	B
...	...	...	...	...	...	...	...	...	
102250	2673974127660	2000-12-29	NaT	NaN	1	1	A2	50.0	Na
102251	2674189951308	2001-01-15	NaT	NaN	1	2	A1	40.0	B
102252	2674548796918	2001-01-15	NaT	NaN	1	1	A1	16.0	N
102253	2675119766018	2001-01-15	NaT	NaN	1	2	B1	76.0	O
102254	2675135410256	2001-01-17	NaT	NaN	1	1	A1	46.0	B

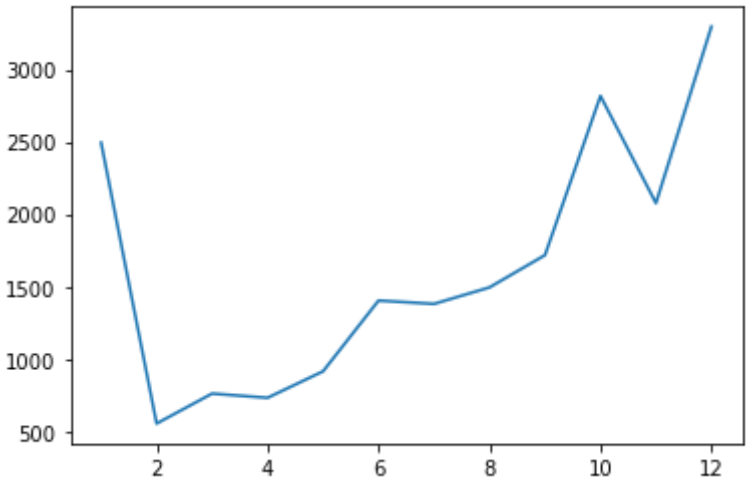
102255 rows × 10 columns

In [868]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#get month
data['deact_month'] = pd.DatetimeIndex(data["deactdt"]).month
s = data['deact_month'].value_counts().sort_index()
s

data['act_month'] = pd.DatetimeIndex(data["actdt"]).month
ss = data['act_month'].value_counts().sort_index()
ss
```

Out[868]:
```
1      8230
2      5091
3      6288
4      5431
5      6959
6      7793
7      8924
8      8092
9      7236
10     8929
11     9078
12    20204
Name: act_month, dtype: int64
```

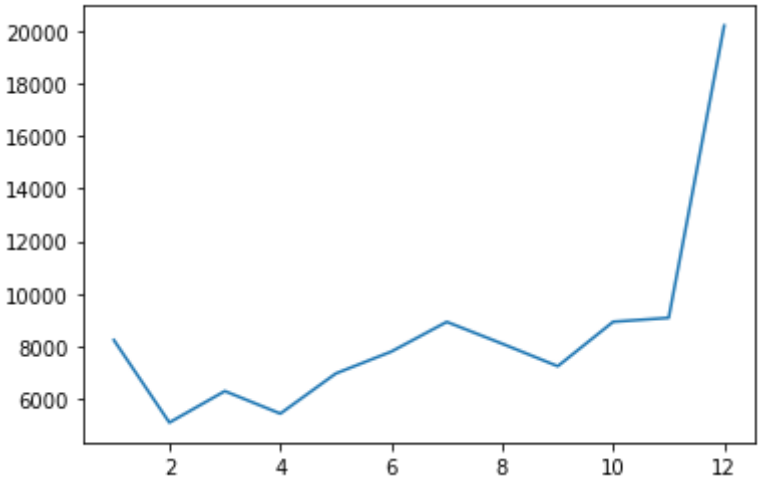
In [869]:
```
plt.plot(s)
# By the graph, more users deactivate accounts in Jan, Sep, Dec.
```

Out[869]:
```
[<matplotlib.lines.Line2D at 0x7fede1a8c5e0>]
```



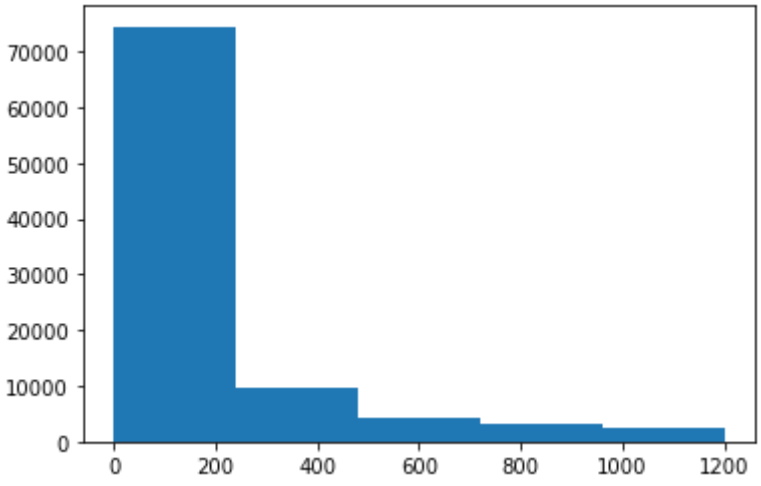
In [870]:
```
plt.plot(ss)
# By the graph, there is an outstanding increasement in Dec. Many customers
```

Out[870]:
```
[<matplotlib.lines.Line2D at 0x7fede1db6250>]
```



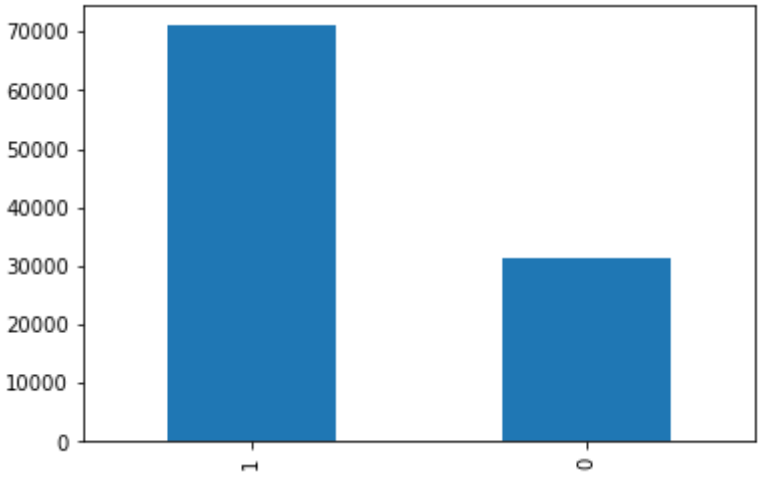
In [871]:
```
plt.hist(data.sales, bins = 5)
# By the graph, the sales betwee 0 and 200 are over 70000 accounts, and the
# This means that the many customers spend a little.
```

Out[871]:
```
(array([74446.,  9567.,  4262.,  2996.,  2379.]),
 array([   0.,  240.,  480.,  720.,  960., 1200.]),
 <BarContainer object of 5 artists>)
```



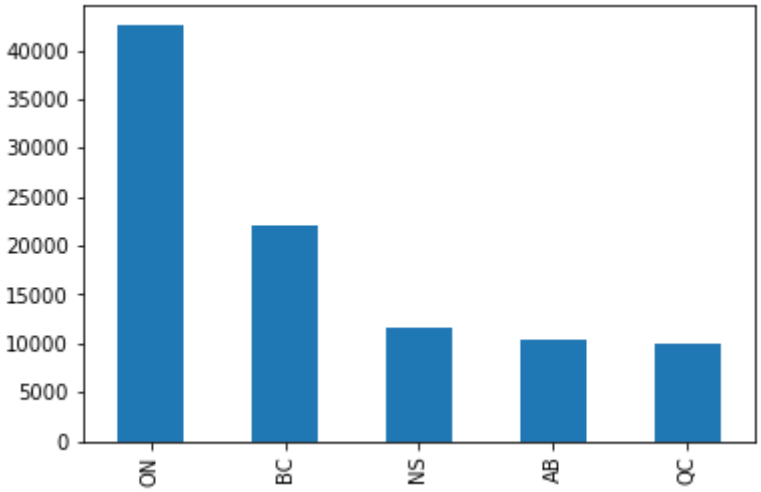
In [872]:
```
data['goodcredit'].value_counts().plot(kind='bar')
# Many customers have good credit.
```

Out[872]: <AxesSubplot:>



In [873]:
```
data['Province'].value_counts().plot(kind='bar')
# More customers from ON, and less customers from QC.
```

Out[873]: <AxesSubplot:>



In [ ]:
```

```

In [874]:
```
#2. What is the age and province distributions of active and deactivated cu
#Use dashboards to present and illustrate.
```

In [875]:
```
import pandas as pd
import numpy as np
import panel as pn

import matplotlib.pyplot as plt
```

In [876]:
```
data['deactdt']
```

Out[876]:
```
0               NaT
1        1999-10-15
2               NaT
3               NaT
4               NaT
              ...
102250          NaT
102251          NaT
102252          NaT
102253          NaT
102254          NaT
Name: deactdt, Length: 102255, dtype: datetime64[ns]
```

In [ ]:
```

```

In [877]:
```
import seaborn as sns
# sns.catplot(data=data, x="Province", y="sales", hue="goodcredit", kind="b
# sns.catplot(data=data, x="Province", y="sales", hue="dealertype", kind="b
```

In [878]:
```
#Creating the interactive dashboard
from ipywidgets import interact
@interact
def create_fare_plot(dashboard = data[['goodcredit','rateplan','dealertype'
    sns.catplot(data = data, x = dashboard, y ='sales',hue = 'Province',kin
    plt.title(f'Mean Bar Plot of the sales grouped by the {dashboard}')

# goodcredit:
```

```
interactive(children=(Dropdown(description='dashboard', options=('goodcre
dit', 'rateplan', 'dealertype'), valu…
```

In [879]:
```
data = pd.read_csv('New_Wireless_Pipe.txt', sep="|")


data['active1'] = data['deactdt']
dea = []
dea = data.active1
dea.loc[~dea.isnull()] = 0
dea.loc[dea.isnull()] = 1

data['deactive1'] = data['deactdt']
dea2 = []
dea2 = data.deactive1
dea2.loc[~dea2.isnull()] = 1
dea2.loc[dea2.isnull()] = 0

data
```

Out[879]:

	acctno	actdt	deactdt	deactreason	goodcredit	rateplan	dealertype	AGE
0	1176913194483	06/20/1999	NaN	NaN	0	1	A1	58.0
1	1176914599423	10/04/1999	10/15/1999	NEED	1	1	A1	45.0
2	1176951913656	07/01/2000	NaN	NaN	0	1	A1	57.0
3	1176954000288	05/30/2000	NaN	NaN	1	2	A1	47.0
4	1176969186303	12/13/2000	NaN	NaN	1	1	C1	82.0
...	...	...	...	...	...	...	...	...
102250	2673974127660	12/29/2000	NaN	NaN	1	1	A2	50.0
102251	2674189951308	01/15/2001	NaN	NaN	1	2	A1	40.0
102252	2674548796918	01/15/2001	NaN	NaN	1	1	A1	16.0
102253	2675119766018	01/15/2001	NaN	NaN	1	2	B1	76.0
102254	2675135410256	01/17/2001	NaN	NaN	1	1	A1	46.0

102255 rows × 12 columns

In [880]:
```
list2 = [0,20,40,60, max(data.AGE)]
age_category = data_cut(data,'AGE',list2)
data['age_category'] = age_category
data
```

Out[880]:

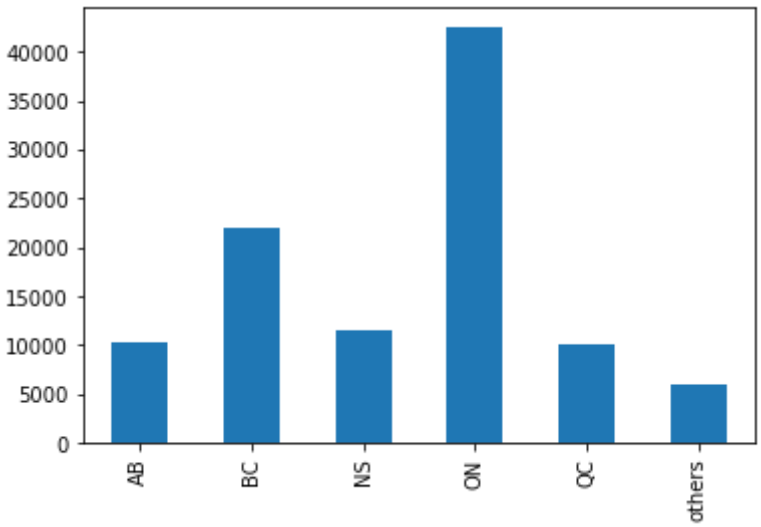
	acctno	actdt	deactdt	deactreason	goodcredit	rateplan	dealertype	AGE
0	1176913194483	06/20/1999	NaN	NaN	0	1	A1	58.0
1	1176914599423	10/04/1999	10/15/1999	NEED	1	1	A1	45.0
2	1176951913656	07/01/2000	NaN	NaN	0	1	A1	57.0
3	1176954000288	05/30/2000	NaN	NaN	1	2	A1	47.0
4	1176969186303	12/13/2000	NaN	NaN	1	1	C1	82.0
...	...	...	...	...	...	...	...	...
102250	2673974127660	12/29/2000	NaN	NaN	1	1	A2	50.0
102251	2674189951308	01/15/2001	NaN	NaN	1	2	A1	40.0
102252	2674548796918	01/15/2001	NaN	NaN	1	1	A1	16.0
102253	2675119766018	01/15/2001	NaN	NaN	1	2	B1	76.0
102254	2675135410256	01/17/2001	NaN	NaN	1	1	A1	46.0

102255 rows × 13 columns

In [881]:
```
pro=[]
pro = data.Province
pro.loc[pro.isnull()] = 'others'
```

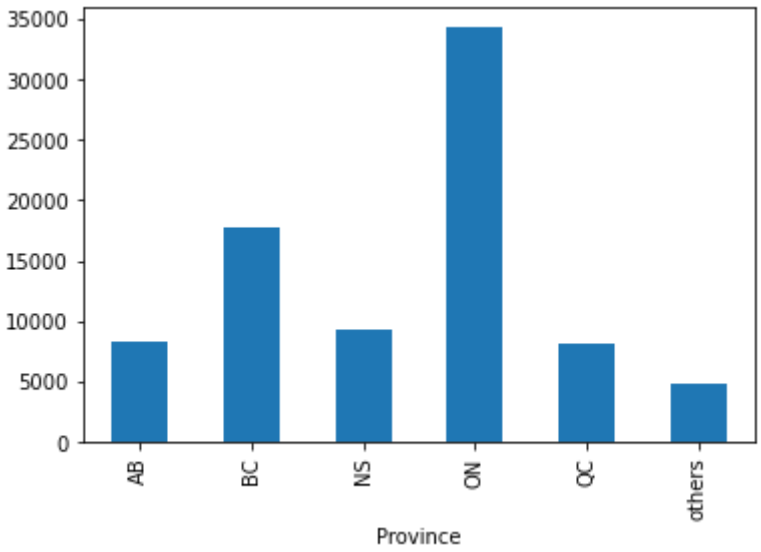
In [882]:
```
data.Province.value_counts()[data.Province.unique()].sort_index().plot(kind
```

Out[882]: <AxesSubplot:>



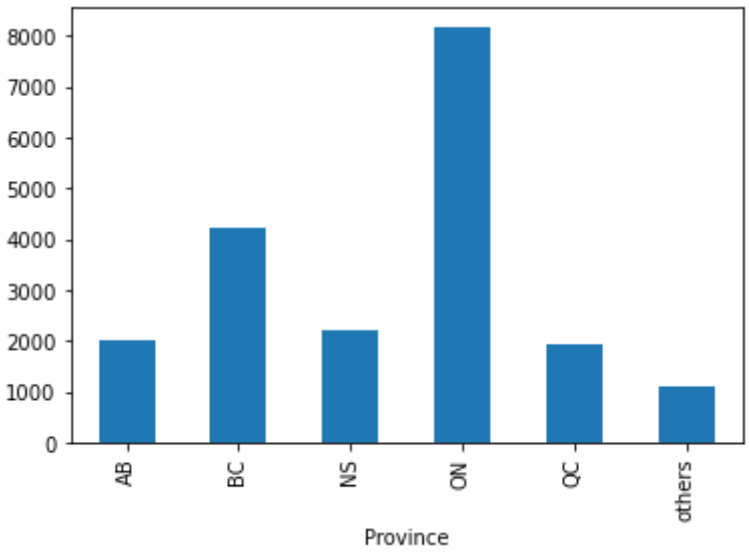
In [883]:
```
#active

#calculate sum of points for each team
data_groups = data.groupby('Province')['active1'].sum()

#create bar plot by group
data_groups.plot(kind='bar')

#By graph, ON have the most activated accounts
```

Out[883]: <AxesSubplot:xlabel='Province'>



In [884]:
```
#deactive

data_groupsde = data.groupby('Province')['deactive1'].sum()
data_groupsde.plot(kind='bar')

#By graph, ON have the most activated accounts
```

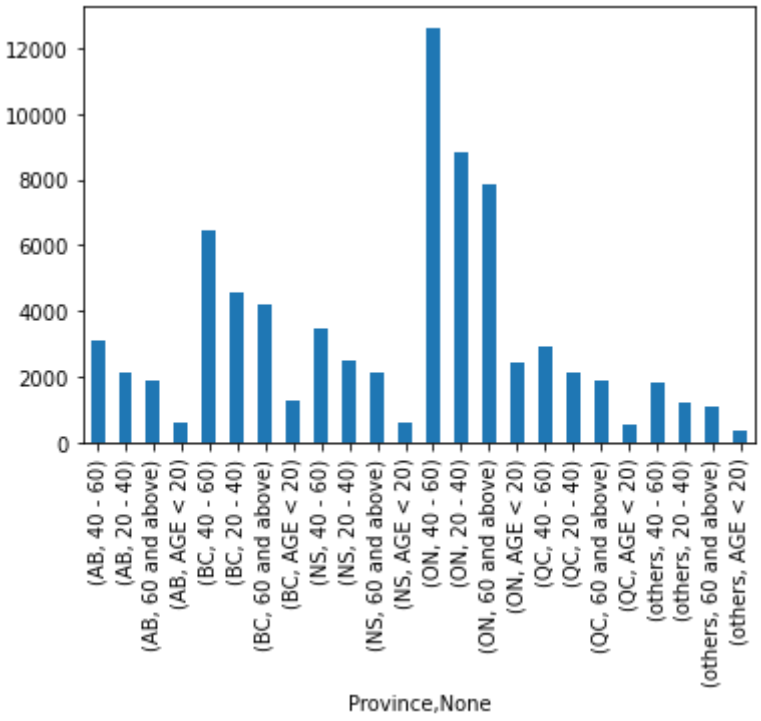
Out[884]: <AxesSubplot:xlabel='Province'>



In [885]:
```
active = data[data.deactdt.isna()]
```

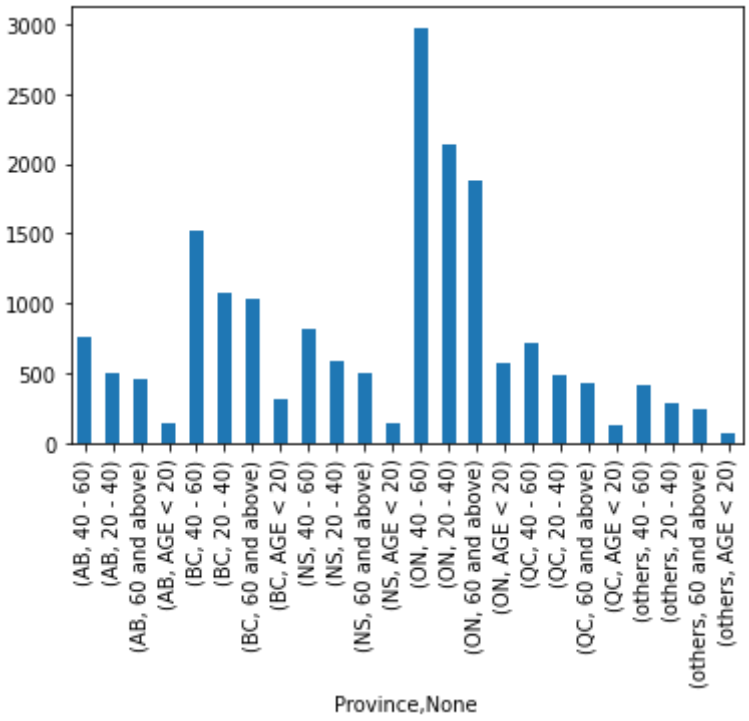
In [886]:
```
#active

data_active = data[data.deactdt.isna()]
data_groups2 = data_active.groupby('Province')['age_category'].value_counts
data_groups2.plot(kind = 'bar')
# By graph, ages of 40-60 have have the most activated accounts among each
```

Out[886]: <AxesSubplot:xlabel='Province,None'>

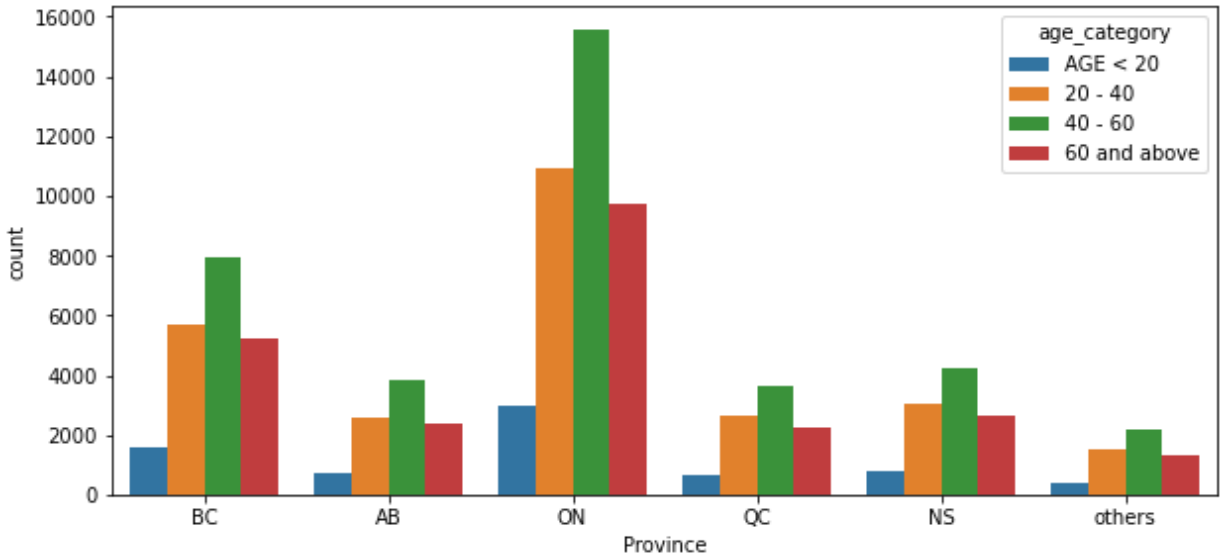


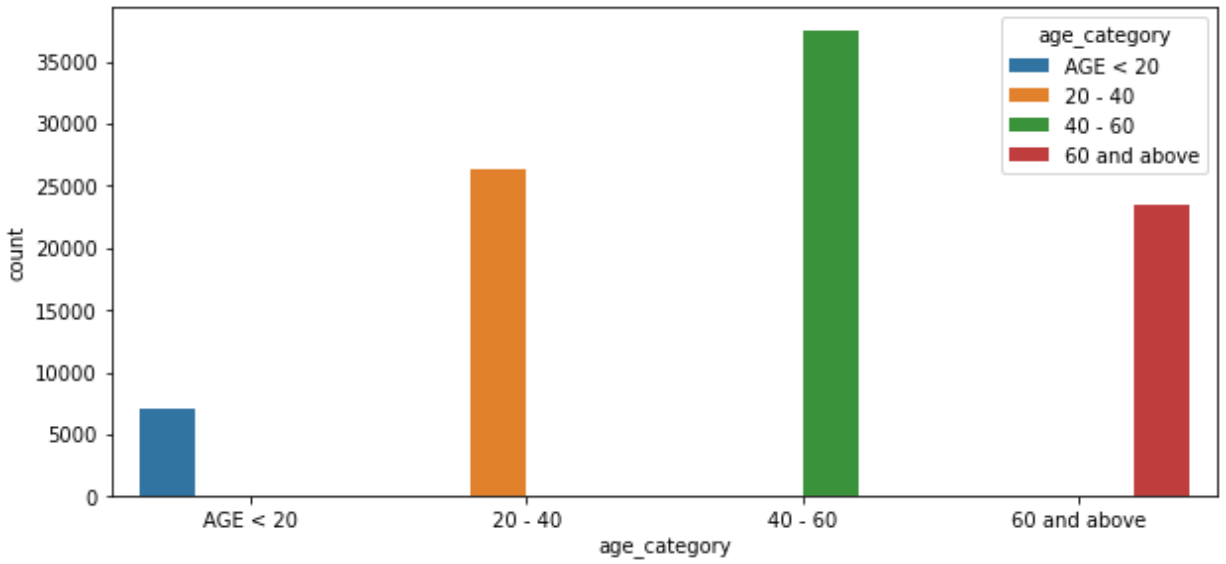
In [887]:
```
#deactive

data_deactive = data[~data.deactdt.isna()]
data_groups2 = data_deactive.groupby('Province')['age_category'].value_coun
data_groups2.plot(kind = 'bar')
# By graph, ages of 40-60 have the most deactivated accounts among each pro
```

Out[887]: <AxesSubplot:xlabel='Province,None'>



In [888]:
```
cat_features = data[[ 'Province', 'age_category']]
fig , ax = plt.subplots(2,1,figsize = (10,10))     # set up 2 x 2 frame cou
for i , subplots in zip (cat_features, ax.flatten()):
  sns.countplot(cat_features[i],hue = data[ 'age_category'],ax = subplots)
plt.show()
```





In [889]:
```
# 3. Segment the customers based on age, province and sales amount:
# Sales segment: < $100, $100---500, $500-$800, $800 and above.
# Age segments: < 20, 21-40, 41-60, 60 and above.
# Create analysis report by using the attached Excel template.
```

In [890]:
```
pa3 = data[['sales', 'AGE','Province','actdt','deactdt']]


list2 = [0,20,40,60, max(pa3.AGE)]
age_category = data_cut(data,'AGE',list2)
pa3['age_category'] = age_category
```

In [891]:
```
list1 = [0,100,500,800, max(pa3.sales)]
sale_category = data_cut(data,'sales',list1)
pa3['sale_category'] = sale_category
```

In [892]:
```
# cutnow = ['AGE < 20' , '20 - 40' , '40 - 60' , '60 and above']
# age_20 = []
# k=0
# def piv(data, list1,k):
#     for i in range(len(data)):
#         if data.age_category[i] ==  list1[k]:
#             age_20.append(1)
#         else:
#             age_20.append(0)
#     return age_20

# a = piv(pa3, cutnow,k)
# pa3['age_20'] = a
```

In [893]:
```
# age_20 = []
# k=1

# a = piv(pa3, cutnow,k)
# pa3['age_20_40'] = a
```

In [894]:
```
# age_20 = []
# k=2

# a = piv(pa3, cutnow,k)
# pa3['age_40_60'] = a
```

In [895]:
```
# age_20 = []
# k=3

# a = piv(pa3, cutnow,k)
# pa3['age_60_60+'] = a
```

In [896]:
```
West_Provinces = ['BC','SK','AB']
Ocean_Provinces = ['PE','NS','NL','NB']
Central_Provinces = ['MT','QC','ON']
pro = []

for i in range(pa3.actdt.count()):
    if pa3.Province[i] in West_Provinces:
        pro.append('West Provinces')
    elif pa3.Province[i] in Ocean_Provinces:
        pro.append('Ocean Provinces')
    elif pa3.Province[i] in Central_Provinces:
        pro.append('Central Provinces')
    else:
        pro.append('Others')


pa3['pro'] = pro
```

In [897]:
```
# def data_cut(da,col,list1):
#     lab=[]
#     for i in range(len(list1)-1):
#         if i == 0:
#             new = [col , ' < ' , str(list1[i+1])]
#             app = ''.join(new)
#             lab.append(app)
#         elif i == len(list1)-2:
#             new = [str(list1[i]) , ' and above']
#             app = ''.join(new)
#             lab.append(app)
#         else:
#             new = [str(list1[i]) , ' - ', str(list1[i+1])]
#             app = ''.join(new)
#             lab.append(app)

#     category = pd.cut(da[col], list1,
#                               labels=lab)
#     return category
```

In [ ]:
```

```

In [898]:
```
par3=pa3[['pro','Province','age_category','sale_category','AGE']]
par3
```

Out[898]:

	pro	Province	age_category	sale_category	AGE
0	West Provinces	BC	40 - 60	100 - 500	58.0
1	West Provinces	AB	40 - 60	sales < 100	45.0
2	West Provinces	BC	40 - 60	500 - 800	57.0
3	Central Provinces	ON	40 - 60	sales < 100	47.0
4	West Provinces	BC	60 and above	NaN	82.0
...	...	...	...	...	...
102250	Others	others	40 - 60	100 - 500	50.0
102251	West Provinces	BC	20 - 40	sales < 100	40.0
102252	Ocean Provinces	NS	AGE < 20	100 - 500	16.0
102253	Central Provinces	ON	60 and above	NaN	76.0
102254	West Provinces	BC	40 - 60	100 - 500	46.0

102255 rows × 5 columns

In [899]:
```
pd.pivot_table(par3,index = ['pro','Province'], columns= ['sale_category'],
               values=['AGE'],fill_value=0,aggfunc='count')
```

Out[899]:

		AGE			
	sale_category	sales < 100	100 - 500	500 - 800	800 and above
pro	Province				
Central Provinces	AB	0	0	0	0
	BC	0	0	0	0
	NS	0	0	0	0
	ON	20268	12137	1852	1642
	QC	4815	2841	408	398
	others	0	0	0	0
Ocean Provinces	AB	0	0	0	0
	BC	0	0	0	0
	NS	5519	3298	550	403
	ON	0	0	0	0
	QC	0	0	0	0
	others	0	0	0	0
Others	AB	0	0	0	0
	BC	0	0	0	0
	NS	0	0	0	0
	ON	0	0	0	0
	QC	0	0	0	0
	others	2857	1623	272	234
West Provinces	AB	4976	2923	443	395
	BC	10493	6353	1012	835
	NS	0	0	0	0
	ON	0	0	0	0
	QC	0	0	0	0
	others	0	0	0	0

In [900]:
```
df1 = pd.DataFrame(data = {'pro':['West Provinces','West Provinces','West P
                           'Province':['BC','AB','BC','ON' ,'BC'],
                           'age_category':['200 - 100', '122-300','256-66',
                           'sale_category':['100 - 500', 'sales < 100','500
                            })
df1[2:4]
```

Out[900]:

	pro	Province	age_category	sale_category
2	West Provinces	BC	256-66	500 - 800
3	Central Provinces	ON	200 - 100	sales < 100

In [901]:
```
pd.pivot_table(df1,
    index=['pro',"Province"],
    columns=["sale_category"],
    values=["age_category"],
    fill_value=0,
    aggfunc=np.sum,
)
```

Out[901]:

		age_category			
	sale_category	100 - 500	300 - 199	500 - 800	sales < 100
pro	**Province**				
Central Provinces	**ON**	0	0	0	200 - 100
West Provinces	**AB**	0	0	0	122-300
	BC	200 - 100	122-300	256-66	0

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [902]:
```
# 4. Statistical Analysis:
# 1) Calculate the tenure in days for each account and give its simple stat
```

In [903]:
```
from datetime import datetime
import numpy as np
import pandas as pd
```

In [904]:
```
train = data[['acctno','actdt','deactdt']]

train["actdt"] = pd.to_datetime(train["actdt"], format='%m/%d/%Y')
train["deactdt"] = pd.to_datetime(train["deactdt"], format='%m/%d/%Y')

max(train.actdt)
train = (train[train['deactdt'].notna()])
max(train.deactdt)
```

Out[904]:
```
Timestamp('2001-01-20 00:00:00')
```

In [905]:
```
train = data[['acctno','actdt','deactdt']]

train["actdt"] = pd.to_datetime(train["actdt"], format='%m/%d/%Y')
train["deactdt"] = pd.to_datetime(train["deactdt"], format='%m/%d/%Y')
from datetime import datetime
today = max(train.actdt)
train.deactdt.replace({np.nan:today}, inplace = True)
diff = train.deactdt - train.actdt
train['tenure'] = diff

diff1 = train.tenure - train.tenure.mean()
train['tenure_from_mean'] = diff1
```

In [906]:
```
train.tenure.max()
train.tenure.min()

train.tenure.mean()
train.tenure
```

Out[906]:
```
0        580 days
1         11 days
2        203 days
3        235 days
4         38 days
           ...
102250    22 days
102251     5 days
102252     5 days
102253     5 days
102254     3 days
Name: tenure, Length: 102255, dtype: timedelta64[ns]
```

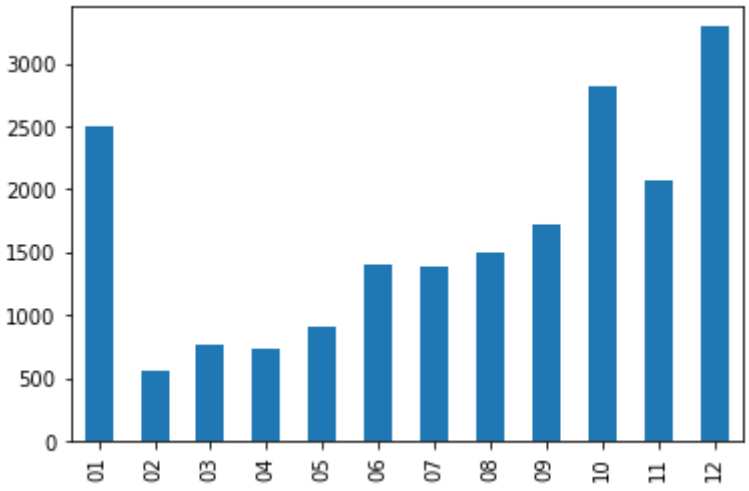
In [907]:
```
#2) Calculate the number of accounts deactivated for each month.
```

In [908]:
```
train = data[['deactdt']]
train = train.dropna()
train["deactdt"] = pd.to_datetime(train["deactdt"], format='%m/%d/%Y')
train['month'] = train['deactdt'].dt.strftime('%m')
train.pivot_table(index = ['month'], aggfunc ='size')
```

Out[908]:
```
month
01    2494
02     553
03     760
04     731
05     914
06    1403
07    1380
08    1494
09    1717
10    2817
11    2076
12    3296
dtype: int64
```

In [909]:
```
train.month.value_counts()[train.month.unique()].sort_index().plot(kind='ba

#Most customers deactivated their accounts in Jan, Sep, and Dec.
```

Out[909]: <AxesSubplot:>



In [910]:
```
# 4) Segment the account, first by account status "Active" and "Deactivated
# Tenure: < 30 days, 31---60 days, 61 days--- one year, over one year. Repo
# number of accounts of percent of all for each segment.
```

In [911]:
```
p4 = data[['actdt','deactdt']]
p4["actdt"] = pd.to_datetime(p4["actdt"], format='%m/%d/%Y')
p4["deactdt"] = pd.to_datetime(p4["deactdt"], format='%m/%d/%Y')
from datetime import datetime
today = max(p4.actdt)
p4.deactdt.replace({np.nan:today}, inplace = True)
diff = p4.deactdt - p4.actdt
p4['tenure'] = diff

p4['tenure'] = p4['tenure'].dt.days.astype('int16')
```

In [912]:
```
list_tenure = [0,30,60,365, max(p4.tenure)+1]
tenure_category = data_cut(p4,'tenure',list_tenure)
p4['tenure_category'] = tenure_category
p4
```

Out[912]:

	actdt	deactdt	tenure	tenure_category
0	1999-06-20	2001-01-20	580	365 and above
1	1999-10-04	1999-10-15	11	tenure < 30
2	2000-07-01	2001-01-20	203	60 - 365
3	2000-05-30	2001-01-20	235	60 - 365
4	2000-12-13	2001-01-20	38	30 - 60
...	...	...	...	...
102250	2000-12-29	2001-01-20	22	tenure < 30
102251	2001-01-15	2001-01-20	5	tenure < 30
102252	2001-01-15	2001-01-20	5	tenure < 30
102253	2001-01-15	2001-01-20	5	tenure < 30
102254	2001-01-17	2001-01-20	3	tenure < 30

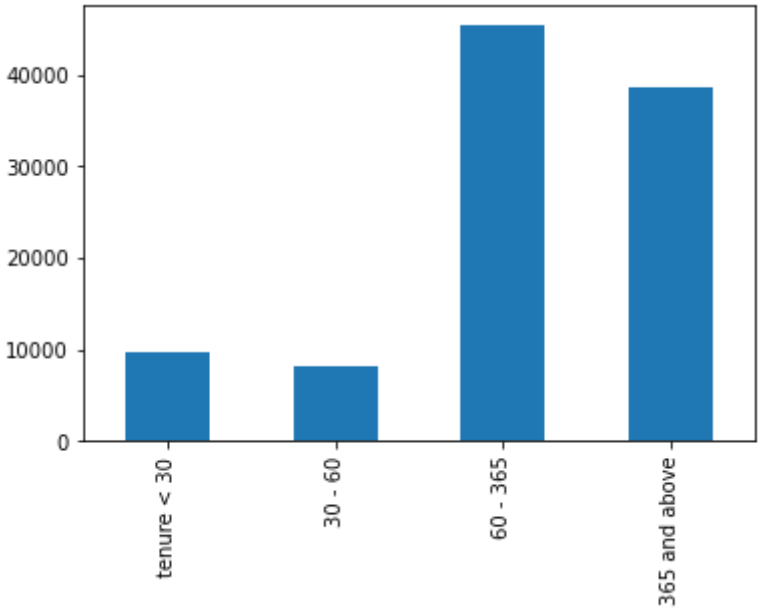
102255 rows × 4 columns

In [913]:
```
p4.tenure_category.value_counts(normalize=True)
```

Out[913]:
```
60 - 365          0.445733
365 and above     0.379417
tenure < 30       0.094933
30 - 60           0.079918
Name: tenure_category, dtype: float64
```

In [914]:
```
p4_unique_nonan = p4.tenure_category[p4.tenure_category.notna()].unique()

p4.tenure_category.value_counts()[p4_unique_nonan].sort_index().plot(kind='

#Most customers used their acc for 60-365days, followed by over a year.
```

Out[914]: <AxesSubplot:>



In [ ]:
```
```

In [915]:
```
# 5) Test the general association between the tenure segments and “Good Cre
# “RatePlan ” and “DealerType.”
```

In [916]:
```
data['tenure'] = p4['tenure']
data['tenure_category'] = p4['tenure_category']
data

import statsmodels.api as sm
from statsmodels.formula.api import ols
model = ols('tenure ~ C(goodcredit)+C(rateplan)+C(dealertype)', data=data).
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table

#p-value of goodcredit, rateplan, dealertype are way less than 0.05, reject
#tenure and C(goodcredit)+C(rateplan)+C(dealertype)
```

Out[916]:

	sum_sq	df	F	PR(>F)
C(goodcredit)	1.020136e+06	1.0	27.593917	1.499474e-07
C(rateplan)	1.595291e+08	2.0	2157.572466	0.000000e+00
C(dealertype)	4.105818e+07	3.0	370.197787	3.637456e-239
Residual	3.780067e+09	102248.0	NaN	NaN

In [ ]:
```
```

In [917]:
```
# 6) Test the general association between the account status and “Good Cred
# “RatePlan ” and “DealerType.”
```

In [918]:
```
p6_1 = data[['actdt','deactdt','goodcredit','rateplan','dealertype','sales'
p6_1
account_status = []
account_status = p6_1.deactdt
account_status.loc[~account_status.isnull()] =0
account_status = account_status.replace(np.nan, 1)
account_status
p6_1['account_status'] = (account_status)
p6_1['deactdt'] = data['deactdt']
p6_1
```

Out[918]:

	actdt	deactdt	goodcredit	rateplan	dealertype	sales	AGE	account_status
0	06/20/1999	NaN	0	1	A1	128.0	58.0	1
1	10/04/1999	10/15/1999	1	1	A1	72.0	45.0	0
2	07/01/2000	NaN	0	1	A1	593.0	57.0	1
3	05/30/2000	NaN	1	2	A1	83.0	47.0	1
4	12/13/2000	NaN	1	1	C1	NaN	82.0	1
...	...	...	...	...	...	...	...	...
102250	12/29/2000	NaN	1	1	A2	112.0	50.0	1
102251	01/15/2001	NaN	1	2	A1	87.0	40.0	1
102252	01/15/2001	NaN	1	1	A1	316.0	16.0	1
102253	01/15/2001	NaN	1	2	B1	NaN	76.0	1
102254	01/17/2001	NaN	1	1	A1	319.0	46.0	1

102255 rows × 8 columns

In [919]:
```
model = ols('account_status ~ C(goodcredit)+C(rateplan)+C(dealertype)', dat
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
#p-value of goodcredit, rateplan, dealertype are way less than 0.05, reject
#account_status and C(goodcredit)+C(rateplan)+C(dealertype)
```

Out[919]:

	sum_sq	df	F	PR(>F)
C(goodcredit)	316.307384	1.0	2095.319986	0.000000e+00
C(rateplan)	81.004500	2.0	268.299691	6.075910e-117
C(dealertype)	23.344894	3.0	51.547982	2.779235e-33
Residual	15435.254586	102248.0	NaN	NaN

In [ ]:

In [920]:
```
# 7) Is there any association between the account status and the tenure seg
# Could you find out a better tenure segmentation strategy that is more ass
# with the account status?
```

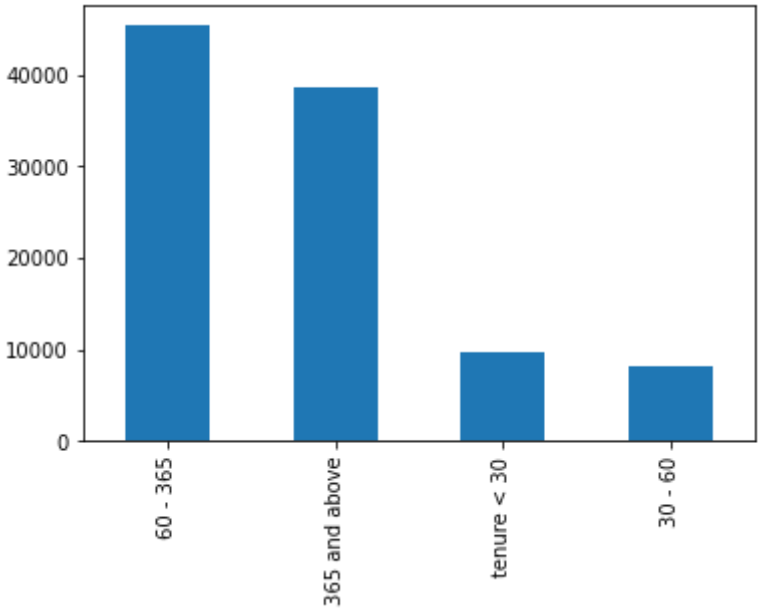
In [921]:
```
data['tenure'] = p4['tenure']
data['tenure_category'] = p4['tenure_category']
data['account_status'] = p6_1['account_status']

data

model = ols('account_status ~ C(tenure_category)', data=data).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table

#There is no association between the account status and the tenure segments
```

Out[921]:

	sum_sq	df	F	PR(>F)
C(tenure_category)	618.626423	3.0	1397.93513	0.0
Residual	15020.309304	101826.0	NaN	NaN

In [922]:
```
tenure_category.value_counts().plot(kind='bar')
```

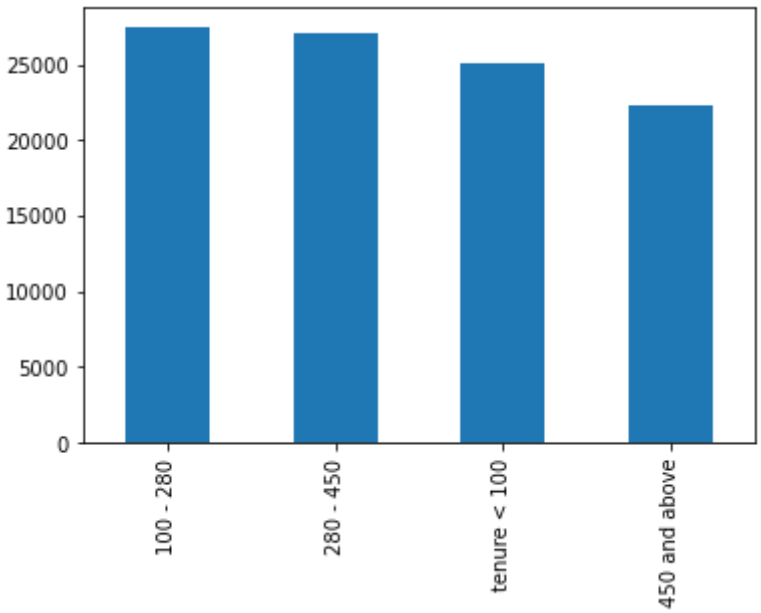
Out[922]: <AxesSubplot:>



In [923]:
```
list_tenure = [0,100,280,450, max(data.tenure)+1]
tenure_category2 = data_cut(data,'tenure',list_tenure)
```

In [924]:
```
tenure_category2.value_counts().plot(kind='bar')
```

Out[924]: <AxesSubplot:>



In [925]:
```
data['tenure_category2'] = tenure_category2
try2 = data[1:200]
try2
model = ols('account_status ~ (C(tenure_category2))', data=try2).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
```

Out[925]:

	sum_sq	df	F	PR(>F)
C(tenure_category2)	0.565953	3.0	1.30089	0.275395
Residual	28.278268	195.0	NaN	NaN

In [926]:
```
data['tenure_category2'] = tenure_category2
data['pro'] = pro

try3 = data[data['pro'] == 'West Provinces']
model = ols('account_status ~ (C(tenure_category2))', data=try3).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
```

Out[926]:

	sum_sq	df	F	PR(>F)
C(tenure_category2)	154.248948	3.0	344.637158	2.652725e-220
Residual	4799.880399	32173.0	NaN	NaN

In [927]:
```
try3 = data[data['pro'] == 'Central Provinces']
model = ols('account_status ~ (C(tenure_category2))', data=try3).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
```

Out[927]:

	sum_sq	df	F	PR(>F)
C(tenure_category2)	202.968432	3.0	451.359658	1.484424e-289
Residual	7836.013187	52277.0	NaN	NaN

In [928]:
```
# could not find a better tenure segmentation strategy that is more associa
```

In [ ]:
```

```

In [929]:
```
# 8) Does Sales amount differ among different account status, GoodCredit, a
# customer age segments?
```

In [930]:
```
p8 = data[['account_status','goodcredit','age_category','sales']]
```

In [949]:
```
p8_sample = p8.sample(n=1000, random_state=1)
p8_sample
```

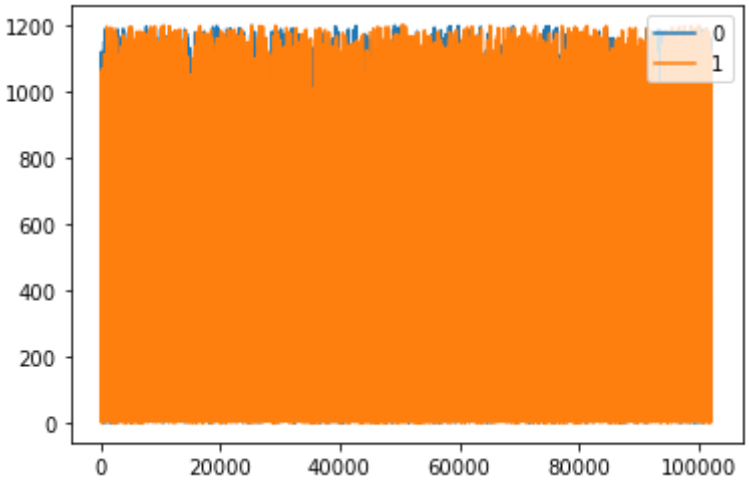
Out[949]:

	account_status	goodcredit	age_category	sales
56506	1	1	40 - 60	301.0
26004	1	1	40 - 60	48.0
19849	1	0	AGE < 20	130.0
63293	1	0	40 - 60	141.0
63744	1	0	AGE < 20	636.0
...	...	...	...	...
62642	1	0	60 and above	17.0
32759	1	1	20 - 40	NaN
51335	1	1	AGE < 20	782.0
93014	1	1	60 and above	47.0
101740	1	1	20 - 40	84.0

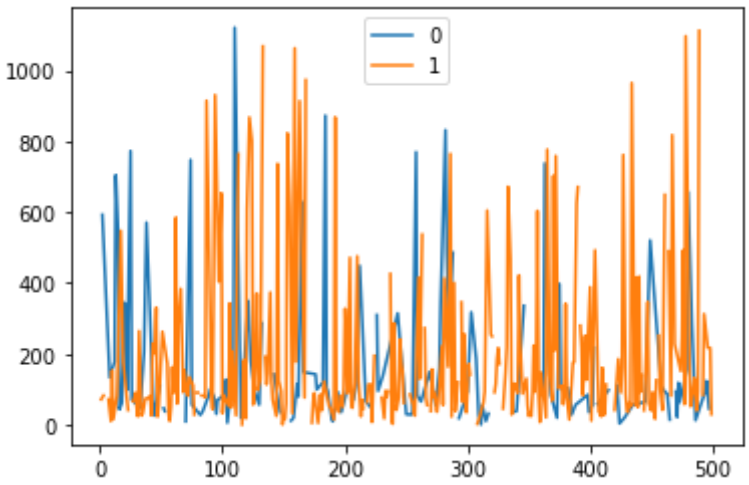
1000 rows × 4 columns

In [989]:
```
p8.groupby('goodcredit')['sales'].plot(legend=True)
```

Out[989]:
```
goodcredit
0    AxesSubplot(0.125,0.125;0.775x0.755)
1    AxesSubplot(0.125,0.125;0.775x0.755)
Name: sales, dtype: object
```

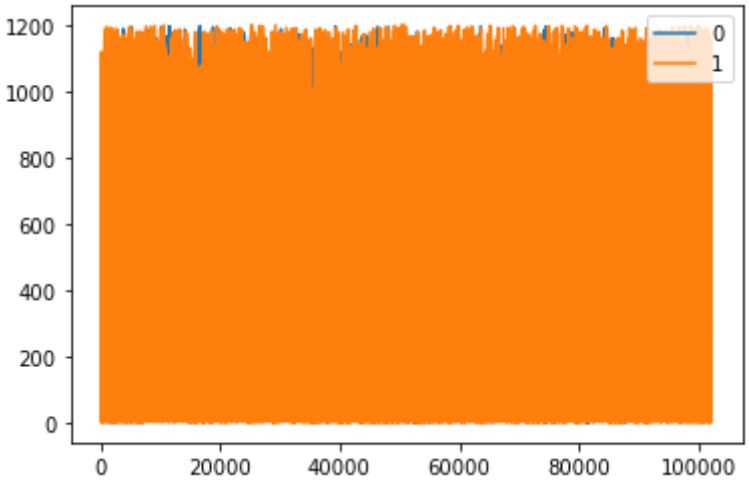


In [990]:
```
pic_num = np.arange(1, 205)
len(pic_num)
plt.clf()
a = 1
for i in range(len(pic_num)):
    b = 500*(i+1)
    p8[a:b].groupby('goodcredit')['sales'].plot(legend=True)
    plt.show()
    a = b
    plt.clf()

# Overall, sales are likely higher from customers with goodcredit than cust
```

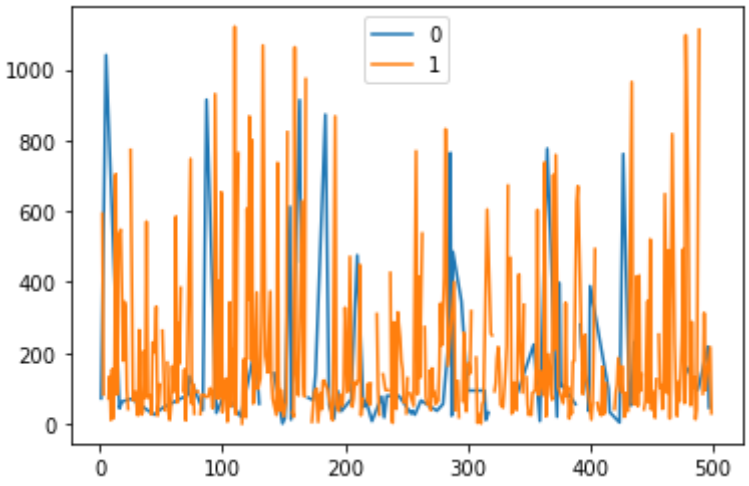


In [992]:
```
p8.groupby('account_status')['sales'].plot(legend=True)
```

Out[992]:
```
account_status
0    AxesSubplot(0.125,0.125;0.775x0.755)
1    AxesSubplot(0.125,0.125;0.775x0.755)
Name: sales, dtype: object
```

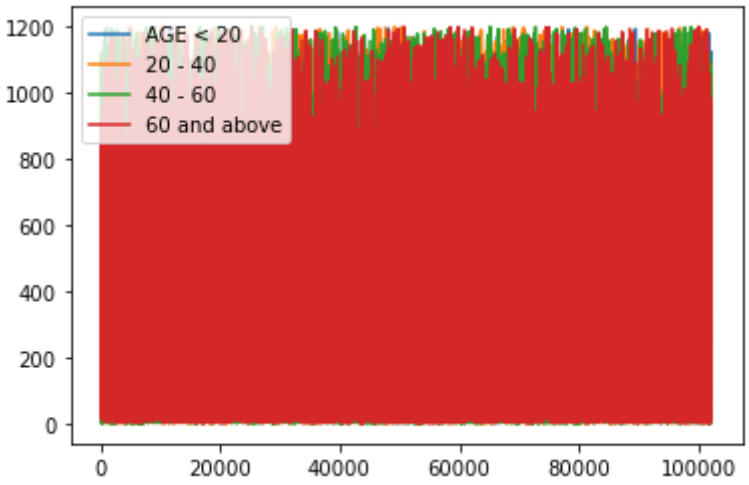


In [991]:
```
plt.clf()
a = 1
for i in range(len(pic_num)):
    b = 500*(i+1)
    p8[a:b].groupby('account_status')['sales'].plot(legend=True)
    plt.show()
    a = b
    plt.clf()


# sales from activated customers are way higher than deactivated customers
```

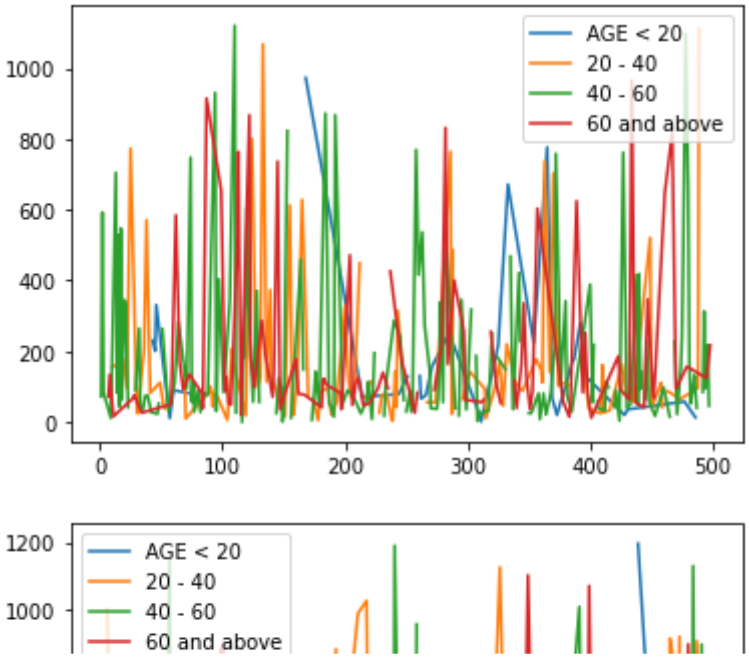


In [993]:
```
p8.groupby('age_category')['sales'].plot(legend=True)
```

Out[993]:
```
age_category
AGE < 20          AxesSubplot(0.125,0.125;0.775x0.755)
20 - 40           AxesSubplot(0.125,0.125;0.775x0.755)
40 - 60           AxesSubplot(0.125,0.125;0.775x0.755)
60 and above      AxesSubplot(0.125,0.125;0.775x0.755)
Name: sales, dtype: object
```



In [994]:
```
plt.clf()
a = 1
for i in range(len(pic_num)):
    b = 500*(i+1)
    p8[a:b].groupby('age_category')['sales'].plot(legend=True)
    plt.show()
    a = b
    plt.clf()


# sales from age<20 customers are the least
# sales from 20-40, 40-60 customers are similar
# sales from 60 and above customers are less than 20-40, 40-60 customers bu
```



In [ ]:

In [857]:
```
import pandas as pd
data = pd.read_csv('New_Wireless_Pipe.txt', sep="|")

import warnings
warnings.filterwarnings("ignore")
data.head

def data_cut(da,col,list1):
    lab=[]
    for i in range(len(list1)-1):
        if i == 0:
            new = [col , ' < ' , str(list1[i+1])]
            app = ''.join(new)
            lab.append(app)
        elif i == len(list1)-2:
            new = [str(list1[i]) , ' and above']
            app = ''.join(new)
            lab.append(app)
        else:
            new = [str(list1[i]) , ' - ', str(list1[i+1])]
            app = ''.join(new)
            lab.append(app)

    category = pd.cut(da[col], list1,
                          labels=lab)
    return category

data.head()
data.info()

#1. Explore and describe the dataset briefly. For example, is the acctno un
#is the number of accounts activated and deactivated? When is the earliest
#latest activation/deactivation dates available? And so on….

print("There are" , data.acctno.count() ,"unique accounts in the dataset")


print("There are", data.deactdt.isnull().sum(), "activated accounts")
# There are 82620 activated accounts.

print("There are" , data.deactdt.count(), "deactivated accounts")
# There are 19635 deactivated accounts.

dat = pd.to_datetime(data['actdt'])

print("The earliest activation date is", dat.min())
# The earliest activation date is 1999-01-20

print("The latest activation date is", dat.max())
# The latest activation date is 2001-01-20

dedat = pd.to_datetime(data['deactdt'])
print("The earliest deactivation date is", dedat.min())
print("The latest deactivation date is", dedat.max())


data.actdt = pd.to_datetime(data['actdt'])
```

```
data.deactdt = pd.to_datetime(data['deactdt'])

date = data[['actdt','deactdt']]


date_table = date.agg(['min', 'max'])
date_table.rename(index={'min': "Earliest Date", 'max': 'Latest Date'},
                  columns={'actdt': 'Activation', 'deactdt': 'Deactivation'
```

```
data.value_counts('deactreason')
# Accouding to the deactivated accounts, most reason is NEED, following COM

data
```

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#get month
data['deact_month'] = pd.DatetimeIndex(data["deactdt"]).month
s = data['deact_month'].value_counts().sort_index()
s

data['act_month'] = pd.DatetimeIndex(data["actdt"]).month
ss = data['act_month'].value_counts().sort_index()
ss

plt.plot(s)
# By the graph, more users deactivate accounts in Jan, Sep, Dec.

plt.plot(ss)
# By the graph, there is an outstanding increasement in Dec. Many customers

plt.hist(data.sales, bins = 5)
# By the graph, the sales betwee 0 and 200 are over 70000 accounts, and the
# This means that the many customers spend a little.

data['goodcredit'].value_counts().plot(kind='bar')
# Many customers have good credit.

data['Province'].value_counts().plot(kind='bar')
# More customers from ON, and less customers from QC.
```

```
#2. What is the age and province distributions of active and deactivated cu
#Use dashboards to present and illustrate.

import pandas as pd
import numpy as np
import panel as pn
```

```
import matplotlib.pyplot as plt

data['deactdt']



import seaborn as sns
# sns.catplot(data=data, x="Province", y="sales", hue="goodcredit", kind="b
# sns.catplot(data=data, x="Province", y="sales", hue="dealertype", kind="b

#Creating the interactive dashboard
from ipywidgets import interact
@interact
def create_fare_plot(dashboard = data[['goodcredit','rateplan','dealertype'
    sns.catplot(data = data, x = dashboard, y ='sales',hue = 'Province',kin
    plt.title(f'Mean Bar Plot of the sales grouped by the {dashboard}')

# goodcredit:

data = pd.read_csv('New_Wireless_Pipe.txt', sep="|")


data['active1'] = data['deactdt']
dea = []
dea = data.active1
dea.loc[~dea.isnull()] = 0
dea.loc[dea.isnull()] = 1

data['deactive1'] = data['deactdt']
dea2 = []
dea2 = data.deactive1
dea2.loc[~dea2.isnull()] = 1
dea2.loc[dea2.isnull()] = 0

data

list2 = [0,20,40,60, max(data.AGE)]
age_category = data_cut(data,'AGE',list2)
data['age_category'] = age_category
data

pro=[]
pro = data.Province
pro.loc[pro.isnull()] = 'others'


data.Province.value_counts()[data.Province.unique()].sort_index().plot(kind

#active

#calculate sum of points for each team
data_groups = data.groupby('Province')['active1'].sum()

#create bar plot by group
data_groups.plot(kind='bar')

#By graph, ON have the most activated accounts
```

```
#deactive

data_groupsde = data.groupby('Province')['deactive1'].sum()
data_groupsde.plot(kind='bar')

#By graph, ON have the most activated accounts

active = data[data.deactdt.isna()]

#active

data_active = data[data.deactdt.isna()]
data_groups2 = data_active.groupby('Province')['age_category'].value_counts
data_groups2.plot(kind = 'bar')
# By graph, ages of 40-60 have have the most activated accounts among each

#deactive

data_deactive = data[~data.deactdt.isna()]
data_groups2 = data_deactive.groupby('Province')['age_category'].value_coun
data_groups2.plot(kind = 'bar')
# By graph, ages of 40-60 have the most deactivated accounts among each pro

cat_features = data[[ 'Province', 'age_category']]
fig , ax = plt.subplots(2,1,figsize = (10,10))     # set up 2 x 2 frame cou
for i , subplots in zip (cat_features, ax.flatten()):
  sns.countplot(cat_features[i],hue = data[ 'age_category'],ax = subplots)
plt.show()

# 3. Segment the customers based on age, province and sales amount:
# Sales segment: < $100, $100---500, $500-$800, $800 and above.
# Age segments: < 20, 21-40, 41-60, 60 and above.
# Create analysis report by using the attached Excel template.

pa3 = data[['sales', 'AGE','Province','actdt','deactdt']]


list2 = [0,20,40,60, max(pa3.AGE)]
age_category = data_cut(data,'AGE',list2)
pa3['age_category'] = age_category

list1 = [0,100,500,800, max(pa3.sales)]
sale_category = data_cut(data,'sales',list1)
pa3['sale_category'] = sale_category

# cutnow = ['AGE < 20' , '20 - 40' , '40 - 60' , '60 and above']
# age_20 = []
# k=0
# def piv(data, list1,k):
#     for i in range(len(data)):
#         if data.age_category[i] ==  list1[k]:
#             age_20.append(1)
#         else:
#             age_20.append(0)
#     return age_20
```

```
# a = piv(pa3, cutnow,k)
# pa3['age_20'] = a

# age_20 = []
# k=1

# a = piv(pa3, cutnow,k)
# pa3['age_20_40'] = a

# age_20 = []
# k=2

# a = piv(pa3, cutnow,k)
# pa3['age_40_60'] = a

# age_20 = []
# k=3

# a = piv(pa3, cutnow,k)
# pa3['age_60_60+'] = a

West_Provinces = ['BC','SK','AB']
Ocean_Provinces = ['PE','NS','NL','NB']
Central_Provinces = ['MT','QC','ON']
pro = []

for i in range(pa3.actdt.count()):
    if pa3.Province[i] in West_Provinces:
        pro.append('West Provinces')
    elif pa3.Province[i] in Ocean_Provinces:
        pro.append('Ocean Provinces')
    elif pa3.Province[i] in Central_Provinces:
        pro.append('Central Provinces')
    else:
        pro.append('Others')


pa3['pro'] = pro

# def data_cut(da,col,list1):
#     lab=[]
#     for i in range(len(list1)-1):
#         if i == 0:
#             new = [col , ' < ' , str(list1[i+1])]
#             app = ''.join(new)
#             lab.append(app)
#         elif i == len(list1)-2:
#             new = [str(list1[i]) , ' and above']
#             app = ''.join(new)
#             lab.append(app)
#         else:
#             new = [str(list1[i]) , ' - ', str(list1[i+1])]
#             app = ''.join(new)
#             lab.append(app)

#     category = pd.cut(da[col], list1,
#                               labels=lab)
```

```
#     return category
```

```
par3=pa3[['pro','Province','age_category','sale_category','AGE']]
par3

pd.pivot_table(par3,index = ['pro','Province'], columns= ['sale_category'],
               values=['AGE'],fill_value=0,aggfunc='count')
```

```
df1 = pd.DataFrame(data = {'pro':['West Provinces','West Provinces','West P
                           'Province':['BC','AB','BC','ON' ,'BC'],
                           'age_category':['200 - 100', '122-300','256-66',
                           'sale_category':['100 - 500', 'sales < 100','500
                            })
df1[2:4]

pd.pivot_table(df1,
    index=['pro',"Province"],
    columns=["sale_category"],
    values=["age_category"],
    fill_value=0,
    aggfunc=np.sum,
)
```

```
# 4. Statistical Analysis:
# 1) Calculate the tenure in days for each account and give its simple stat

from datetime import datetime
import numpy as np
import pandas as pd

train = data[['acctno','actdt','deactdt']]

train["actdt"] = pd.to_datetime(train["actdt"], format='%m/%d/%Y')
train["deactdt"] = pd.to_datetime(train["deactdt"], format='%m/%d/%Y')

max(train.actdt)
train = (train[train['deactdt'].notna()])
max(train.deactdt)

train = data[['acctno','actdt','deactdt']]

train["actdt"] = pd.to_datetime(train["actdt"], format='%m/%d/%Y')
train["deactdt"] = pd.to_datetime(train["deactdt"], format='%m/%d/%Y')
```

```
from datetime import datetime
today = max(train.actdt)
train.deactdt.replace({np.nan:today}, inplace = True)
diff = train.deactdt - train.actdt
train['tenure'] = diff

diff1 = train.tenure - train.tenure.mean()
train['tenure_from_mean'] = diff1

train.tenure.max()
train.tenure.min()

train.tenure.mean()
train.tenure

#2) Calculate the number of accounts deactivated for each month.

train = data[['deactdt']]
train = train.dropna()
train["deactdt"] = pd.to_datetime(train["deactdt"], format='%m/%d/%Y')
train['month'] = train['deactdt'].dt.strftime('%m')
train.pivot_table(index = ['month'], aggfunc ='size')


train.month.value_counts()[train.month.unique()].sort_index().plot(kind='ba

#Most customers deactivated their accounts in Jan, Sep, and Dec.

# 4) Segment the account, first by account status "Active" and "Deactivated
# Tenure: < 30 days, 31---60 days, 61 days--- one year, over one year. Repo
# number of accounts of percent of all for each segment.

p4 = data[['actdt','deactdt']]
p4["actdt"] = pd.to_datetime(p4["actdt"], format='%m/%d/%Y')
p4["deactdt"] = pd.to_datetime(p4["deactdt"], format='%m/%d/%Y')
from datetime import datetime
today = max(p4.actdt)
p4.deactdt.replace({np.nan:today}, inplace = True)
diff = p4.deactdt - p4.actdt
p4['tenure'] = diff

p4['tenure'] = p4['tenure'].dt.days.astype('int16')


list_tenure = [0,30,60,365, max(p4.tenure)+1]
tenure_category = data_cut(p4,'tenure',list_tenure)
p4['tenure_category'] = tenure_category
p4

p4.tenure_category.value_counts(normalize=True)

p4_unique_nonan = p4.tenure_category[p4.tenure_category.notna()].unique()

p4.tenure_category.value_counts()[p4_unique_nonan].sort_index().plot(kind='


#Most customers used their acc for 60-365days, followed by over a year.
```

```
# 5) Test the general association between the tenure segments and “Good Cre
# “RatePlan ” and “DealerType.”

data['tenure'] = p4['tenure']
data['tenure_category'] = p4['tenure_category']
data

import statsmodels.api as sm
from statsmodels.formula.api import ols
model = ols('tenure ~ C(goodcredit)+C(rateplan)+C(dealertype)', data=data).
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table

#p-value of goodcredit, rateplan, dealertype are way less than 0.05, reject
#tenure and C(goodcredit)+C(rateplan)+C(dealertype)



# 6) Test the general association between the account status and “Good Cred
# “RatePlan ” and “DealerType.”

p6_1 = data[['actdt','deactdt','goodcredit','rateplan','dealertype','sales'
p6_1
account_status = []
account_status = p6_1.deactdt
account_status.loc[~account_status.isnull()] =0
account_status = account_status.replace(np.nan, 1)
account_status
p6_1['account_status'] = (account_status)
p6_1['deactdt'] = data['deactdt']
p6_1

model = ols('account_status ~ C(goodcredit)+C(rateplan)+C(dealertype)', dat
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
#p-value of goodcredit, rateplan, dealertype are way less than 0.05, reject
#account_status and C(goodcredit)+C(rateplan)+C(dealertype)



# 7) Is there any association between the account status and the tenure seg
# Could you find out a better tenure segmentation strategy that is more ass
# with the account status?

data['tenure'] = p4['tenure']
data['tenure_category'] = p4['tenure_category']
data['account_status'] = p6_1['account_status']

data

model = ols('account_status ~ C(tenure_category)', data=data).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
```

```
#There is no association between the account status and the tenure segments

tenure_category.value_counts().plot(kind='bar')

list_tenure = [0,100,280,450, max(data.tenure)+1]
tenure_category2 = data_cut(data,'tenure',list_tenure)

tenure_category2.value_counts().plot(kind='bar')

data['tenure_category2'] = tenure_category2
try2 = data[1:200]
try2
model = ols('account_status ~ (C(tenure_category2))', data=try2).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table

data['tenure_category2'] = tenure_category2
data['pro'] = pro

try3 = data[data['pro'] == 'West Provinces']
model = ols('account_status ~ (C(tenure_category2))', data=try3).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table

try3 = data[data['pro'] == 'Central Provinces']
model = ols('account_status ~ (C(tenure_category2))', data=try3).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table

# could not find a better tenure segmentation strategy that is more associa



# 8) Does Sales amount differ among different account status, GoodCredit, a
# customer age segments?

p8 = data[['account_status','goodcredit','age_category','sales']]

p8_sample = p8.sample(n=1000, random_state=1)
p8_sample

p8.groupby('goodcredit')['sales'].plot(legend=True)

pic_num = np.arange(1, 205)
len(pic_num)
plt.clf()
a = 1
for i in range(len(pic_num)):
    b = 500*(i+1)
    p8[a:b].groupby('goodcredit')['sales'].plot(legend=True)
    plt.show()
    a = b
    plt.clf()

# Overall, sales are likely higher from customers with goodcredit than cust
```

```
p8.groupby('account_status')['sales'].plot(legend=True)

plt.clf()
a = 1
for i in range(len(pic_num)):
    b = 500*(i+1)
    p8[a:b].groupby('account_status')['sales'].plot(legend=True)
    plt.show()
    a = b
    plt.clf()


# sales from activated customers are way higher than deactivated customers

p8.groupby('age_category')['sales'].plot(legend=True)

plt.clf()
a = 1
for i in range(len(pic_num)):
    b = 500*(i+1)
    p8[a:b].groupby('age_category')['sales'].plot(legend=True)
    plt.show()
    a = b
    plt.clf()


# sales from age<20 customers are the least
# sales from 20-40, 40-60 customers are similar
# sales from 60 and above customers are less than 20-40, 40-60 customers bu

import pandas as pd
data = pd.read_csv('New_Wireless_Pipe.txt', sep="|")

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

Out[857]:

	acctno	actdt	deactdt	deactreason	goodcredit	rateplan	dealertype	AGE
0	1176913194483	06/20/1999	NaN	NaN	0	1	A1	58.0
1	1176914599423	10/04/1999	10/15/1999	NEED	1	1	A1	45.0
2	1176951913656	07/01/2000	NaN	NaN	0	1	A1	57.0
3	1176954000288	05/30/2000	NaN	NaN	1	2	A1	47.0
4	1176969186303	12/13/2000	NaN	NaN	1	1	C1	82.0
...	...	...	...	...	...	...	...	...
102250	2673974127660	12/29/2000	NaN	NaN	1	1	A2	50.0
102251	2674189951308	01/15/2001	NaN	NaN	1	2	A1	40.0
102252	2674548796918	01/15/2001	NaN	NaN	1	1	A1	16.0
102253	2675119766018	01/15/2001	NaN	NaN	1	2	B1	76.0
102254	2675135410256	01/17/2001	NaN	NaN	1	1	A1	46.0

102255 rows × 10 columns

In [859]:
```
def data_cut(da,col,list1):
    lab=[]
    for i in range(len(list1)-1):
        if i == 0:
            new = [col , ' < ' , str(list1[i+1])]
            app = ''.join(new)
            lab.append(app)
        elif i == len(list1)-2:
            new = [str(list1[i]) , ' and above']
            app = ''.join(new)
            lab.append(app)
        else:
            new = [str(list1[i]) , ' - ', str(list1[i+1])]
            app = ''.join(new)
            lab.append(app)

    category = pd.cut(da[col], list1,
                          labels=lab)
    return category
```

In [860]:
```
data.head()
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 102255 entries, 0 to 102254
Data columns (total 10 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   acctno       102255 non-null  int64  
 1   actdt        102255 non-null  object 
 2   deactdt      19635 non-null   object 
 3   deactreason  19093 non-null   object 
 4   goodcredit   102255 non-null  int64  
 5   rateplan     102255 non-null  int64  
 6   dealertype   102255 non-null  object 
 7   AGE          94547 non-null   float64
 8   Province     96348 non-null   object 
 9   sales        93650 non-null   float64
dtypes: float64(2), int64(3), object(5)
memory usage: 7.8+ MB
```

In [861]:
```
#1. Explore and describe the dataset briefly. For example, is the acctno un
#is the number of accounts activated and deactivated? When is the earliest
#latest activation/deactivation dates available? And so on….
```

In [862]:
```
print("There are" , data.acctno.count() ,"unique accounts in the dataset")
```

```
There are 102255 unique accounts in the dataset
```

In [863]:
```
print("There are", data.deactdt.isnull().sum(), "activated accounts")
# There are 82620 activated accounts.

print("There are" , data.deactdt.count(), "deactivated accounts")
# There are 19635 deactivated accounts.
```

```
There are 82620 activated accounts
There are 19635 deactivated accounts
```

In [864]:
```
dat = pd.to_datetime(data['actdt'])

print("The earliest activation date is", dat.min())
# The earliest activation date is 1999-01-20

print("The latest activation date is", dat.max())
# The latest activation date is 2001-01-20

dedat = pd.to_datetime(data['deactdt'])
print("The earliest deactivation date is", dedat.min())
print("The latest deactivation date is", dedat.max())
```

```
The earliest activation date is 1999-01-20 00:00:00
The latest activation date is 2001-01-20 00:00:00
The earliest deactivation date is 1999-01-25 00:00:00
The latest deactivation date is 2001-01-20 00:00:00
```

In [865]:
```
data.actdt = pd.to_datetime(data['actdt'])
data.deactdt = pd.to_datetime(data['deactdt'])

date = data[['actdt','deactdt']]


date_table = date.agg(['min', 'max'])
date_table.rename(index={'min': "Earliest Date", 'max': 'Latest Date'},
                  columns={'actdt': 'Activation', 'deactdt': 'Deactivation'
```

Out[865]:

	Activation	Deactivation
Earliest Date	1999-01-20	1999-01-25
Latest Date	2001-01-20	2001-01-20

In [866]:
```
data.value_counts('deactreason')
# Accouding to the deactivated accounts, most reason is NEED, following COM
```

Out[866]:
```
deactreason
NEED    6888
COMP    4722
DEBT    4020
TECH    1767
MOVE    1696
dtype: int64
```

In [867]:
```
data
```

Out[867]:

	acctno	actdt	deactdt	deactreason	goodcredit	rateplan	dealertype	AGE	Provinc
0	1176913194483	1999-06-20	NaT	NaN	0	1	A1	58.0	B
1	1176914599423	1999-10-04	1999-10-15	NEED	1	1	A1	45.0	A
2	1176951913656	2000-07-01	NaT	NaN	0	1	A1	57.0	B
3	1176954000288	2000-05-30	NaT	NaN	1	2	A1	47.0	O
4	1176969186303	2000-12-13	NaT	NaN	1	1	C1	82.0	B
...	...	...	...	...	...	...	...	...	
102250	2673974127660	2000-12-29	NaT	NaN	1	1	A2	50.0	Na
102251	2674189951308	2001-01-15	NaT	NaN	1	2	A1	40.0	B
102252	2674548796918	2001-01-15	NaT	NaN	1	1	A1	16.0	N
102253	2675119766018	2001-01-15	NaT	NaN	1	2	B1	76.0	O
102254	2675135410256	2001-01-17	NaT	NaN	1	1	A1	46.0	B

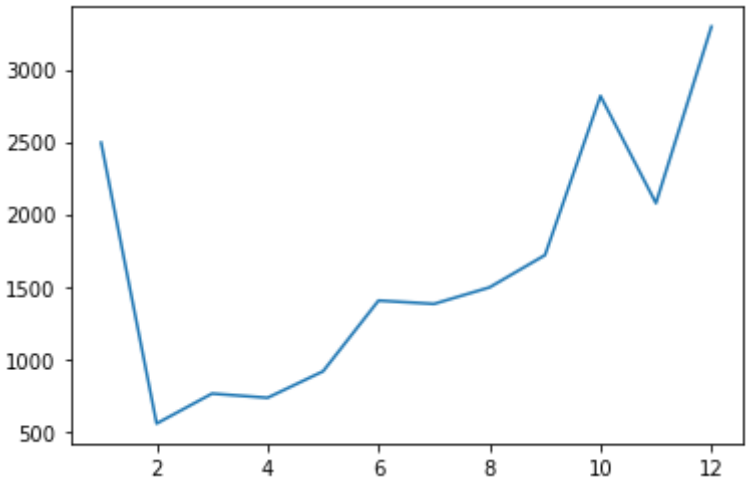
102255 rows × 10 columns

In [868]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#get month
data['deact_month'] = pd.DatetimeIndex(data["deactdt"]).month
s = data['deact_month'].value_counts().sort_index()
s

data['act_month'] = pd.DatetimeIndex(data["actdt"]).month
ss = data['act_month'].value_counts().sort_index()
ss
```

Out[868]:
```
1      8230
2      5091
3      6288
4      5431
5      6959
6      7793
7      8924
8      8092
9      7236
10     8929
11     9078
12    20204
Name: act_month, dtype: int64
```

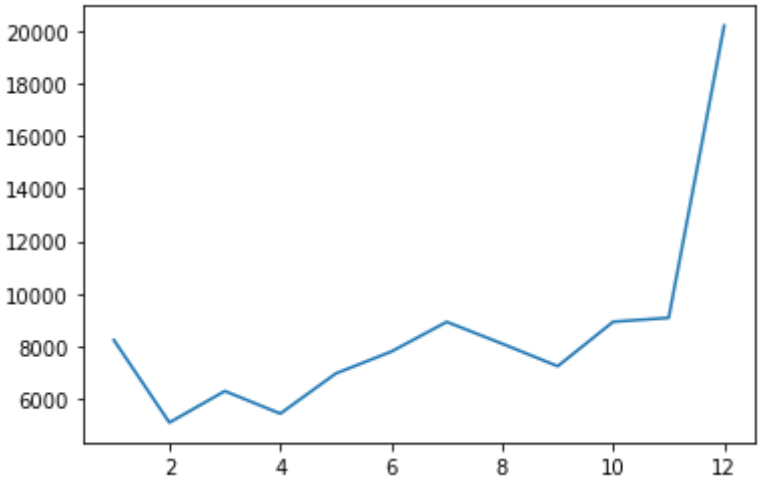
In [869]:
```
plt.plot(s)
# By the graph, more users deactivate accounts in Jan, Sep, Dec.
```

Out[869]:
```
[<matplotlib.lines.Line2D at 0x7fede1a8c5e0>]
```



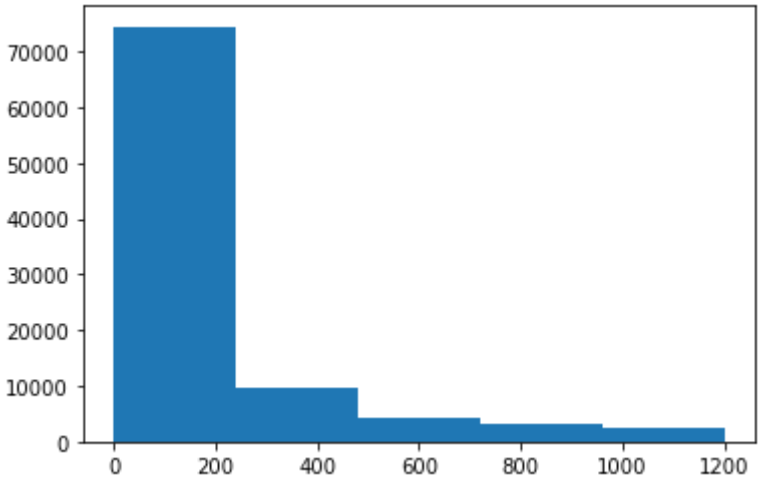
In [870]:
```
plt.plot(ss)
# By the graph, there is an outstanding increasement in Dec. Many customers
```

Out[870]:
```
[<matplotlib.lines.Line2D at 0x7fede1db6250>]
```



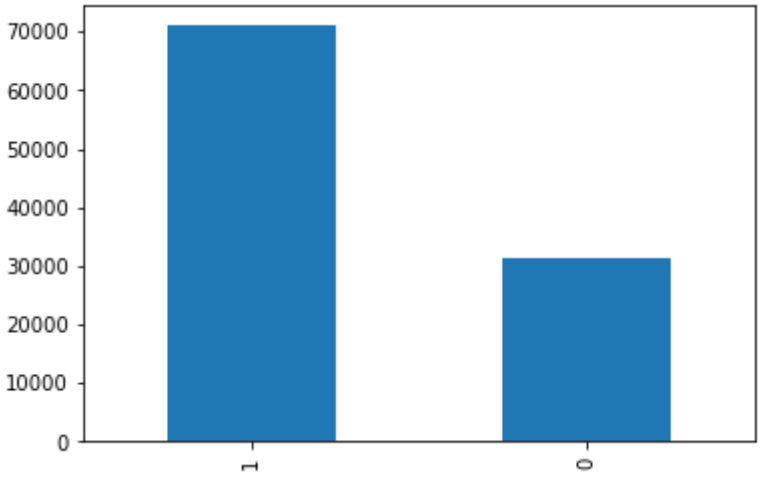
In [871]:
```
plt.hist(data.sales, bins = 5)
# By the graph, the sales betwee 0 and 200 are over 70000 accounts, and the
# This means that the many customers spend a little.
```

Out[871]:
```
(array([74446.,  9567.,  4262.,  2996.,  2379.]),
 array([   0.,  240.,  480.,  720.,  960., 1200.]),
 <BarContainer object of 5 artists>)
```



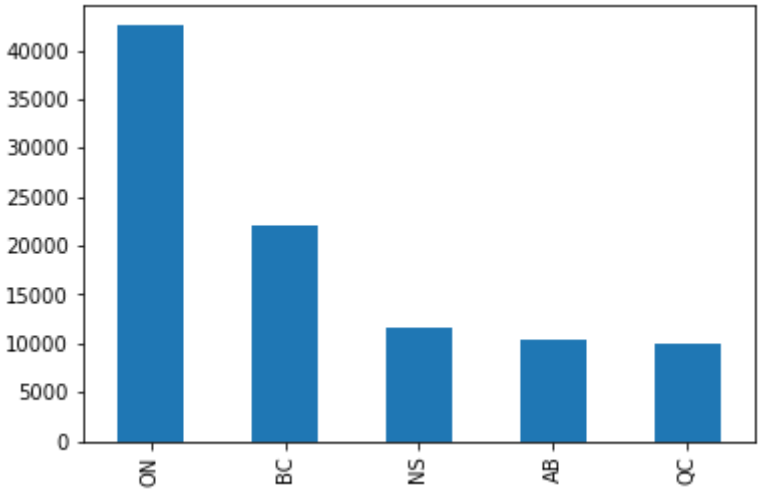
In [872]:
```
data['goodcredit'].value_counts().plot(kind='bar')
# Many customers have good credit.
```

Out[872]: <AxesSubplot:>



In [873]:
```
data['Province'].value_counts().plot(kind='bar')
# More customers from ON, and less customers from QC.
```

Out[873]: <AxesSubplot:>



In [ ]:
```

```

In [874]:
```
#2. What is the age and province distributions of active and deactivated cu
#Use dashboards to present and illustrate.
```

In [875]:
```
import pandas as pd
import numpy as np
import panel as pn

import matplotlib.pyplot as plt
```

In [876]:
```
data['deactdt']
```

Out[876]:
```
0               NaT
1        1999-10-15
2               NaT
3               NaT
4               NaT
              ...
102250          NaT
102251          NaT
102252          NaT
102253          NaT
102254          NaT
Name: deactdt, Length: 102255, dtype: datetime64[ns]
```

In [ ]:
```

```

In [877]:
```
import seaborn as sns
# sns.catplot(data=data, x="Province", y="sales", hue="goodcredit", kind="b
# sns.catplot(data=data, x="Province", y="sales", hue="dealertype", kind="b
```

In [878]:
```
#Creating the interactive dashboard
from ipywidgets import interact
@interact
def create_fare_plot(dashboard = data[['goodcredit','rateplan','dealertype'
    sns.catplot(data = data, x = dashboard, y ='sales',hue = 'Province',kin
    plt.title(f'Mean Bar Plot of the sales grouped by the {dashboard}')

# goodcredit:
```

```
interactive(children=(Dropdown(description='dashboard', options=('goodcre
dit', 'rateplan', 'dealertype'), valu…
```

In [879]:
```
data = pd.read_csv('New_Wireless_Pipe.txt', sep="|")


data['active1'] = data['deactdt']
dea = []
dea = data.active1
dea.loc[~dea.isnull()] = 0
dea.loc[dea.isnull()] = 1

data['deactive1'] = data['deactdt']
dea2 = []
dea2 = data.deactive1
dea2.loc[~dea2.isnull()] = 1
dea2.loc[dea2.isnull()] = 0

data
```

Out[879]:

	acctno	actdt	deactdt	deactreason	goodcredit	rateplan	dealertype	AGE
0	1176913194483	06/20/1999	NaN	NaN	0	1	A1	58.0
1	1176914599423	10/04/1999	10/15/1999	NEED	1	1	A1	45.0
2	1176951913656	07/01/2000	NaN	NaN	0	1	A1	57.0
3	1176954000288	05/30/2000	NaN	NaN	1	2	A1	47.0
4	1176969186303	12/13/2000	NaN	NaN	1	1	C1	82.0
...	...	...	...	...	...	...	...	...
102250	2673974127660	12/29/2000	NaN	NaN	1	1	A2	50.0
102251	2674189951308	01/15/2001	NaN	NaN	1	2	A1	40.0
102252	2674548796918	01/15/2001	NaN	NaN	1	1	A1	16.0
102253	2675119766018	01/15/2001	NaN	NaN	1	2	B1	76.0
102254	2675135410256	01/17/2001	NaN	NaN	1	1	A1	46.0

102255 rows × 12 columns

In [880]:
```
list2 = [0,20,40,60, max(data.AGE)]
age_category = data_cut(data,'AGE',list2)
data['age_category'] = age_category
data
```

Out[880]:

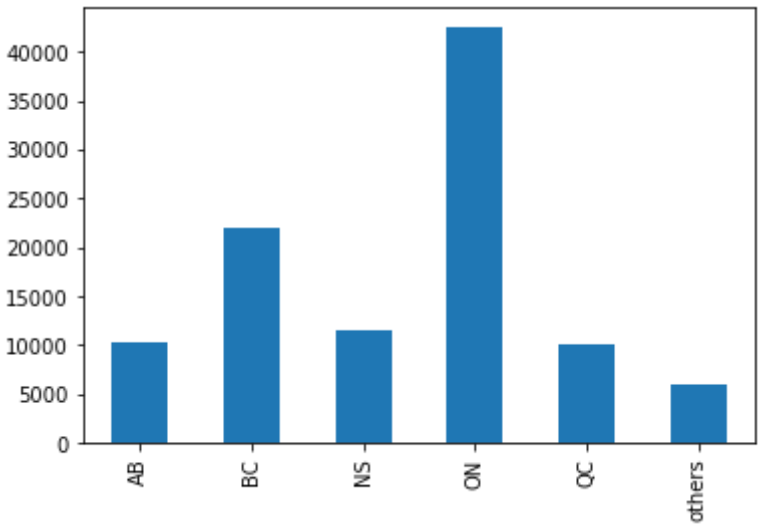
	acctno	actdt	deactdt	deactreason	goodcredit	rateplan	dealertype	AGE
0	1176913194483	06/20/1999	NaN	NaN	0	1	A1	58.0
1	1176914599423	10/04/1999	10/15/1999	NEED	1	1	A1	45.0
2	1176951913656	07/01/2000	NaN	NaN	0	1	A1	57.0
3	1176954000288	05/30/2000	NaN	NaN	1	2	A1	47.0
4	1176969186303	12/13/2000	NaN	NaN	1	1	C1	82.0
...	...	...	...	...	...	...	...	...
102250	2673974127660	12/29/2000	NaN	NaN	1	1	A2	50.0
102251	2674189951308	01/15/2001	NaN	NaN	1	2	A1	40.0
102252	2674548796918	01/15/2001	NaN	NaN	1	1	A1	16.0
102253	2675119766018	01/15/2001	NaN	NaN	1	2	B1	76.0
102254	2675135410256	01/17/2001	NaN	NaN	1	1	A1	46.0

102255 rows × 13 columns

In [881]:
```
pro=[]
pro = data.Province
pro.loc[pro.isnull()] = 'others'
```

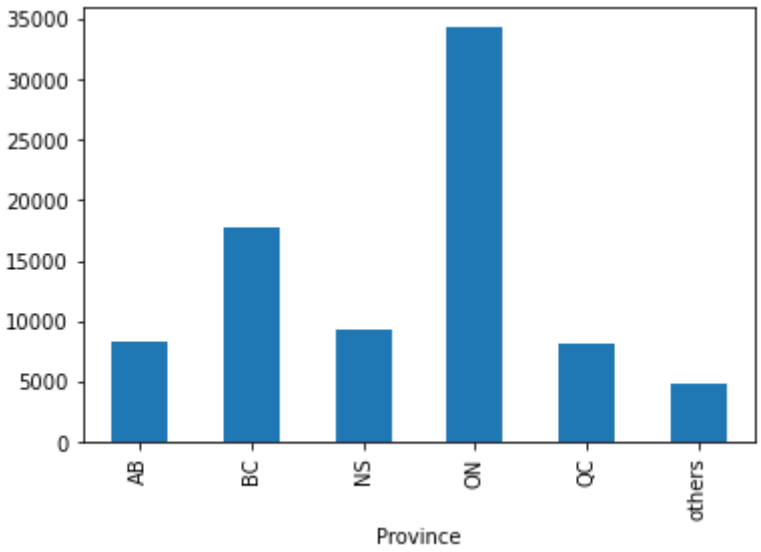
In [882]:
```
data.Province.value_counts()[data.Province.unique()].sort_index().plot(kind
```

Out[882]: <AxesSubplot:>



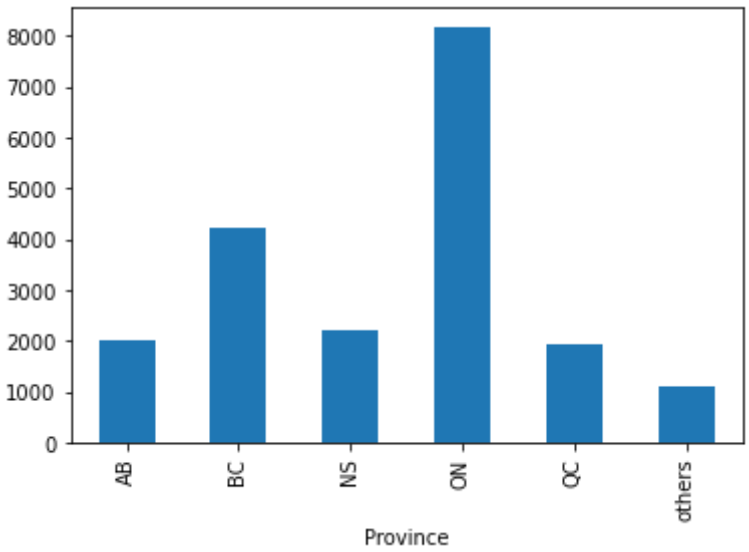
In [883]:
```
#active

#calculate sum of points for each team
data_groups = data.groupby('Province')['active1'].sum()

#create bar plot by group
data_groups.plot(kind='bar')

#By graph, ON have the most activated accounts
```

Out[883]: <AxesSubplot:xlabel='Province'>



In [884]:
```
#deactive

data_groupsde = data.groupby('Province')['deactive1'].sum()
data_groupsde.plot(kind='bar')

#By graph, ON have the most activated accounts
```

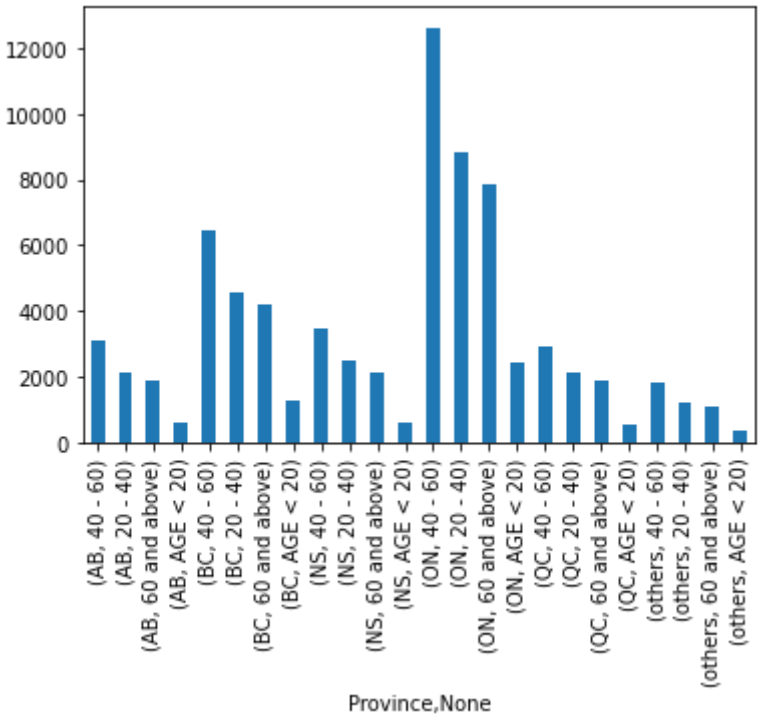
Out[884]: <AxesSubplot:xlabel='Province'>



In [885]:
```
active = data[data.deactdt.isna()]
```

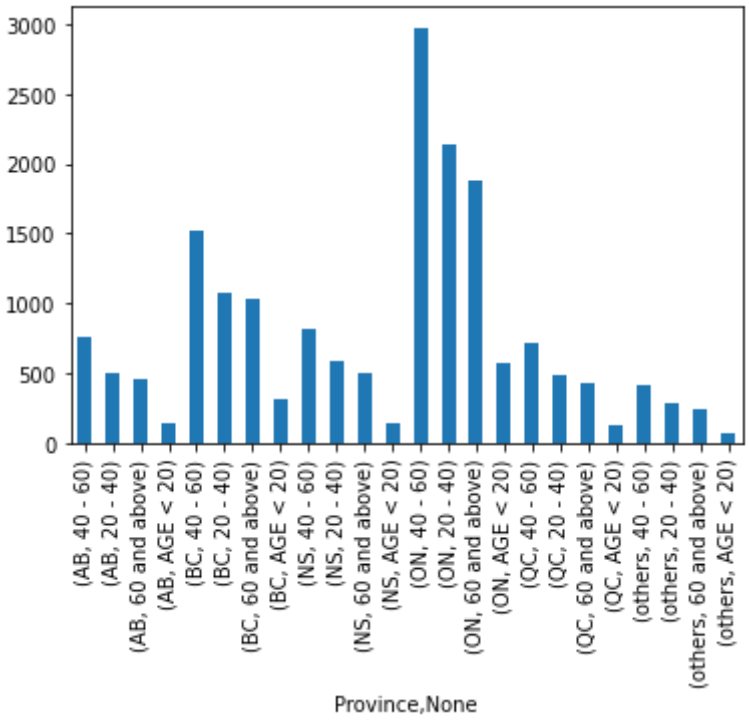
In [886]:
```
#active

data_active = data[data.deactdt.isna()]
data_groups2 = data_active.groupby('Province')['age_category'].value_counts
data_groups2.plot(kind = 'bar')
# By graph, ages of 40-60 have have the most activated accounts among each
```

Out[886]: <AxesSubplot:xlabel='Province,None'>

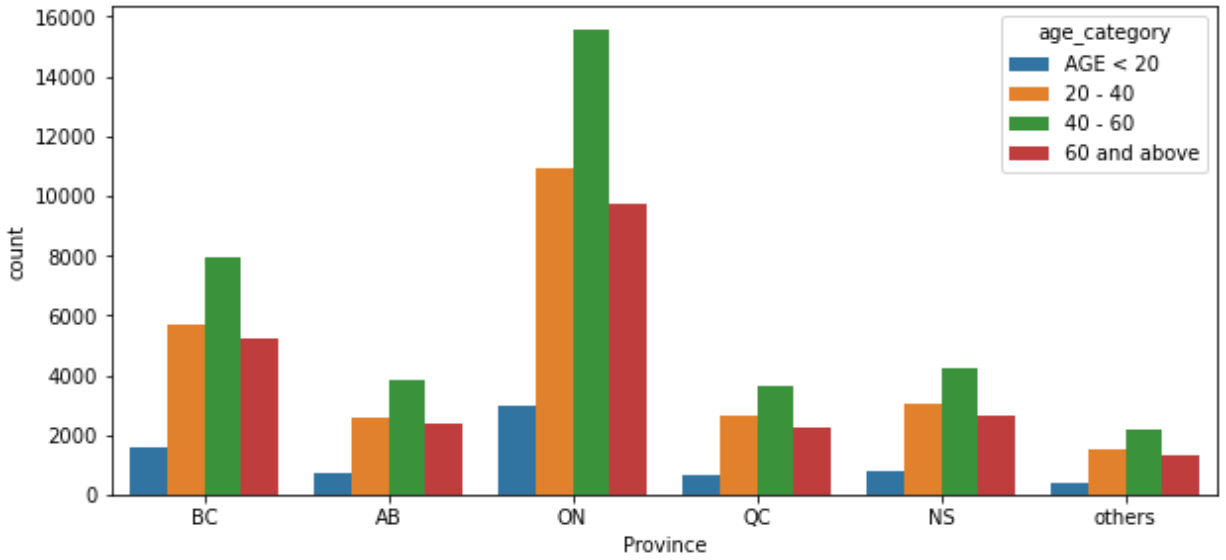


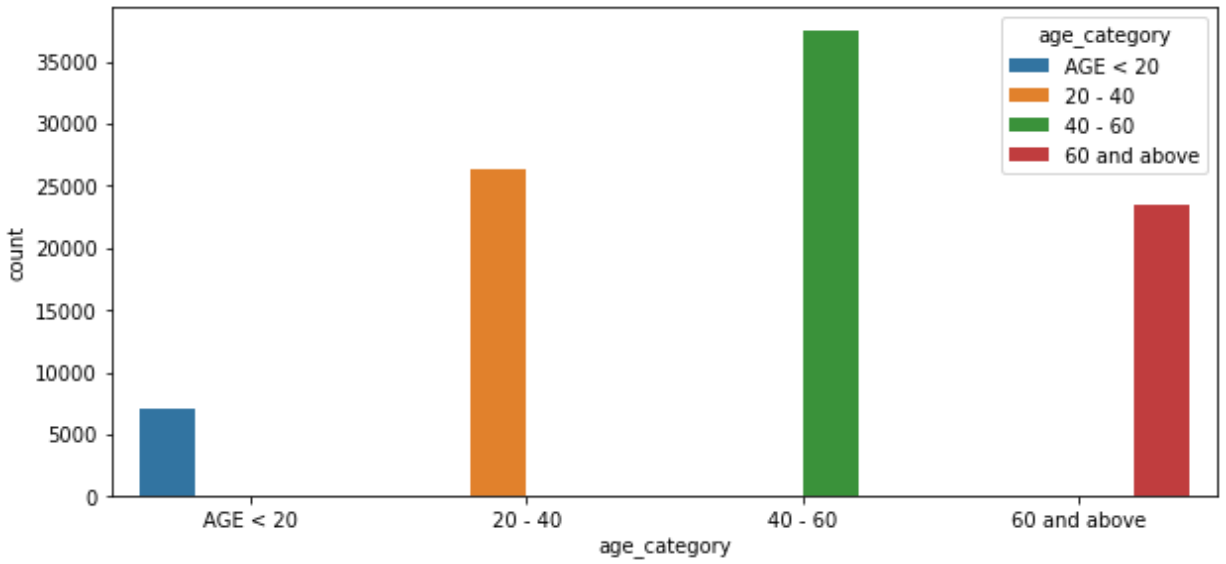
In [887]:
```
#deactive

data_deactive = data[~data.deactdt.isna()]
data_groups2 = data_deactive.groupby('Province')['age_category'].value_coun
data_groups2.plot(kind = 'bar')
# By graph, ages of 40-60 have the most deactivated accounts among each pro
```

Out[887]: <AxesSubplot:xlabel='Province,None'>



In [888]:
```
cat_features = data[[ 'Province', 'age_category']]
fig , ax = plt.subplots(2,1,figsize = (10,10))     # set up 2 x 2 frame cou
for i , subplots in zip (cat_features, ax.flatten()):
  sns.countplot(cat_features[i],hue = data[ 'age_category'],ax = subplots)
plt.show()
```





In [889]:
```
# 3. Segment the customers based on age, province and sales amount:
# Sales segment: < $100, $100---500, $500-$800, $800 and above.
# Age segments: < 20, 21-40, 41-60, 60 and above.
# Create analysis report by using the attached Excel template.
```

In [890]:
```
pa3 = data[['sales', 'AGE','Province','actdt','deactdt']]


list2 = [0,20,40,60, max(pa3.AGE)]
age_category = data_cut(data,'AGE',list2)
pa3['age_category'] = age_category
```

In [891]:
```
list1 = [0,100,500,800, max(pa3.sales)]
sale_category = data_cut(data,'sales',list1)
pa3['sale_category'] = sale_category
```

In [892]:
```
# cutnow = ['AGE < 20' , '20 - 40' , '40 - 60' , '60 and above']
# age_20 = []
# k=0
# def piv(data, list1,k):
#     for i in range(len(data)):
#         if data.age_category[i] ==  list1[k]:
#             age_20.append(1)
#         else:
#             age_20.append(0)
#     return age_20

# a = piv(pa3, cutnow,k)
# pa3['age_20'] = a
```

In [893]:
```
# age_20 = []
# k=1

# a = piv(pa3, cutnow,k)
# pa3['age_20_40'] = a
```

In [894]:
```
# age_20 = []
# k=2

# a = piv(pa3, cutnow,k)
# pa3['age_40_60'] = a
```

In [895]:
```
# age_20 = []
# k=3

# a = piv(pa3, cutnow,k)
# pa3['age_60_60+'] = a
```

In [896]:
```
West_Provinces = ['BC','SK','AB']
Ocean_Provinces = ['PE','NS','NL','NB']
Central_Provinces = ['MT','QC','ON']
pro = []

for i in range(pa3.actdt.count()):
    if pa3.Province[i] in West_Provinces:
        pro.append('West Provinces')
    elif pa3.Province[i] in Ocean_Provinces:
        pro.append('Ocean Provinces')
    elif pa3.Province[i] in Central_Provinces:
        pro.append('Central Provinces')
    else:
        pro.append('Others')


pa3['pro'] = pro
```

In [897]:
```
# def data_cut(da,col,list1):
#     lab=[]
#     for i in range(len(list1)-1):
#         if i == 0:
#             new = [col , ' < ' , str(list1[i+1])]
#             app = ''.join(new)
#             lab.append(app)
#         elif i == len(list1)-2:
#             new = [str(list1[i]) , ' and above']
#             app = ''.join(new)
#             lab.append(app)
#         else:
#             new = [str(list1[i]) , ' - ', str(list1[i+1])]
#             app = ''.join(new)
#             lab.append(app)

#     category = pd.cut(da[col], list1,
#                               labels=lab)
#     return category
```

In [ ]:
```

```

In [898]:
```
par3=pa3[['pro','Province','age_category','sale_category','AGE']]
par3
```

Out[898]:

	pro	Province	age_category	sale_category	AGE
0	West Provinces	BC	40 - 60	100 - 500	58.0
1	West Provinces	AB	40 - 60	sales < 100	45.0
2	West Provinces	BC	40 - 60	500 - 800	57.0
3	Central Provinces	ON	40 - 60	sales < 100	47.0
4	West Provinces	BC	60 and above	NaN	82.0
...	...	...	...	...	...
102250	Others	others	40 - 60	100 - 500	50.0
102251	West Provinces	BC	20 - 40	sales < 100	40.0
102252	Ocean Provinces	NS	AGE < 20	100 - 500	16.0
102253	Central Provinces	ON	60 and above	NaN	76.0
102254	West Provinces	BC	40 - 60	100 - 500	46.0

102255 rows × 5 columns

In [899]:
```
pd.pivot_table(par3,index = ['pro','Province'], columns= ['sale_category'],
               values=['AGE'],fill_value=0,aggfunc='count')
```

Out[899]:

		AGE			
	sale_category	sales < 100	100 - 500	500 - 800	800 and above
pro	Province				
Central Provinces	AB	0	0	0	0
	BC	0	0	0	0
	NS	0	0	0	0
	ON	20268	12137	1852	1642
	QC	4815	2841	408	398
	others	0	0	0	0
Ocean Provinces	AB	0	0	0	0
	BC	0	0	0	0
	NS	5519	3298	550	403
	ON	0	0	0	0
	QC	0	0	0	0
	others	0	0	0	0
Others	AB	0	0	0	0
	BC	0	0	0	0
	NS	0	0	0	0
	ON	0	0	0	0
	QC	0	0	0	0
	others	2857	1623	272	234
West Provinces	AB	4976	2923	443	395
	BC	10493	6353	1012	835
	NS	0	0	0	0
	ON	0	0	0	0
	QC	0	0	0	0
	others	0	0	0	0

In [900]:
```
df1 = pd.DataFrame(data = {'pro':['West Provinces','West Provinces','West P
                           'Province':['BC','AB','BC','ON' ,'BC'],
                           'age_category':['200 - 100', '122-300','256-66',
                           'sale_category':['100 - 500', 'sales < 100','500
                            })
df1[2:4]
```

Out[900]:

	pro	Province	age_category	sale_category
2	West Provinces	BC	256-66	500 - 800
3	Central Provinces	ON	200 - 100	sales < 100

In [901]:
```
pd.pivot_table(df1,
    index=['pro',"Province"],
    columns=["sale_category"],
    values=["age_category"],
    fill_value=0,
    aggfunc=np.sum,
)
```

Out[901]:

		age_category			
	sale_category	100 - 500	300 - 199	500 - 800	sales < 100
pro	Province				
Central Provinces	**ON**	0	0	0	200 - 100
West Provinces	**AB**	0	0	0	122-300
	BC	200 - 100	122-300	256-66	0

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [902]:
```
# 4. Statistical Analysis:
# 1) Calculate the tenure in days for each account and give its simple stat
```

In [903]:
```
from datetime import datetime
import numpy as np
import pandas as pd
```

In [904]:
```
train = data[['acctno','actdt','deactdt']]

train["actdt"] = pd.to_datetime(train["actdt"], format='%m/%d/%Y')
train["deactdt"] = pd.to_datetime(train["deactdt"], format='%m/%d/%Y')

max(train.actdt)
train = (train[train['deactdt'].notna()])
max(train.deactdt)
```

Out[904]:
```
Timestamp('2001-01-20 00:00:00')
```

In [905]:
```
train = data[['acctno','actdt','deactdt']]

train["actdt"] = pd.to_datetime(train["actdt"], format='%m/%d/%Y')
train["deactdt"] = pd.to_datetime(train["deactdt"], format='%m/%d/%Y')
from datetime import datetime
today = max(train.actdt)
train.deactdt.replace({np.nan:today}, inplace = True)
diff = train.deactdt - train.actdt
train['tenure'] = diff

diff1 = train.tenure - train.tenure.mean()
train['tenure_from_mean'] = diff1
```

In [906]:
```
train.tenure.max()
train.tenure.min()

train.tenure.mean()
train.tenure
```

Out[906]:
```
0        580 days
1         11 days
2        203 days
3        235 days
4         38 days
           ...
102250    22 days
102251     5 days
102252     5 days
102253     5 days
102254     3 days
Name: tenure, Length: 102255, dtype: timedelta64[ns]
```

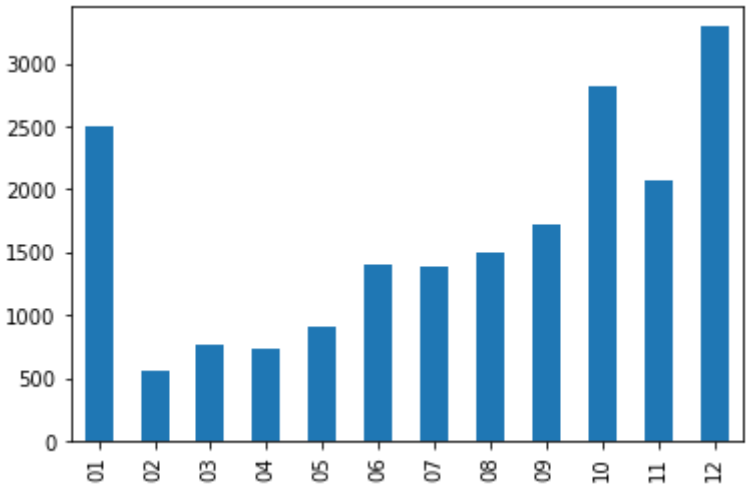
In [907]:
```
#2) Calculate the number of accounts deactivated for each month.
```

In [908]:
```
train = data[['deactdt']]
train = train.dropna()
train["deactdt"] = pd.to_datetime(train["deactdt"], format='%m/%d/%Y')
train['month'] = train['deactdt'].dt.strftime('%m')
train.pivot_table(index = ['month'], aggfunc ='size')
```

Out[908]:
```
month
01    2494
02     553
03     760
04     731
05     914
06    1403
07    1380
08    1494
09    1717
10    2817
11    2076
12    3296
dtype: int64
```

In [909]:
```
train.month.value_counts()[train.month.unique()].sort_index().plot(kind='ba

#Most customers deactivated their accounts in Jan, Sep, and Dec.
```

Out[909]: <AxesSubplot:>



In [910]:
```
# 4) Segment the account, first by account status "Active" and "Deactivated
# Tenure: < 30 days, 31---60 days, 61 days--- one year, over one year. Repo
# number of accounts of percent of all for each segment.
```

In [911]:
```
p4 = data[['actdt','deactdt']]
p4["actdt"] = pd.to_datetime(p4["actdt"], format='%m/%d/%Y')
p4["deactdt"] = pd.to_datetime(p4["deactdt"], format='%m/%d/%Y')
from datetime import datetime
today = max(p4.actdt)
p4.deactdt.replace({np.nan:today}, inplace = True)
diff = p4.deactdt - p4.actdt
p4['tenure'] = diff

p4['tenure'] = p4['tenure'].dt.days.astype('int16')
```

In [912]:
```
list_tenure = [0,30,60,365, max(p4.tenure)+1]
tenure_category = data_cut(p4,'tenure',list_tenure)
p4['tenure_category'] = tenure_category
p4
```

Out[912]:

	actdt	deactdt	tenure	tenure_category
0	1999-06-20	2001-01-20	580	365 and above
1	1999-10-04	1999-10-15	11	tenure < 30
2	2000-07-01	2001-01-20	203	60 - 365
3	2000-05-30	2001-01-20	235	60 - 365
4	2000-12-13	2001-01-20	38	30 - 60
...	...	...	...	...
102250	2000-12-29	2001-01-20	22	tenure < 30
102251	2001-01-15	2001-01-20	5	tenure < 30
102252	2001-01-15	2001-01-20	5	tenure < 30
102253	2001-01-15	2001-01-20	5	tenure < 30
102254	2001-01-17	2001-01-20	3	tenure < 30

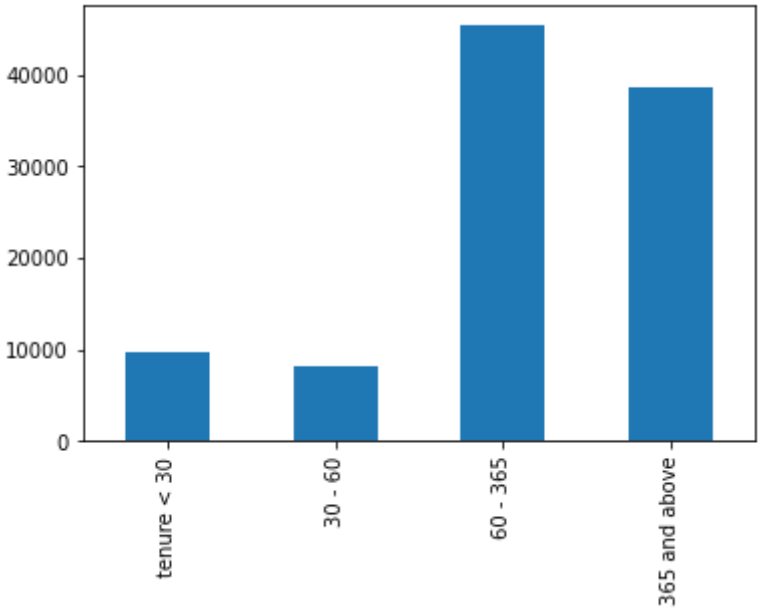
102255 rows × 4 columns

In [913]:
```
p4.tenure_category.value_counts(normalize=True)
```

Out[913]:
```
60 - 365          0.445733
365 and above     0.379417
tenure < 30       0.094933
30 - 60           0.079918
Name: tenure_category, dtype: float64
```

In [914]:
```
p4_unique_nonan = p4.tenure_category[p4.tenure_category.notna()].unique()

p4.tenure_category.value_counts()[p4_unique_nonan].sort_index().plot(kind='

#Most customers used their acc for 60-365days, followed by over a year.
```

Out[914]: <AxesSubplot:>



In [ ]:

In [915]:
```
# 5) Test the general association between the tenure segments and “Good Cre
# “RatePlan ” and “DealerType.”
```

In [916]:
```
data['tenure'] = p4['tenure']
data['tenure_category'] = p4['tenure_category']
data

import statsmodels.api as sm
from statsmodels.formula.api import ols
model = ols('tenure ~ C(goodcredit)+C(rateplan)+C(dealertype)', data=data).
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table

#p-value of goodcredit, rateplan, dealertype are way less than 0.05, reject
#tenure and C(goodcredit)+C(rateplan)+C(dealertype)
```

Out[916]:

	sum_sq	df	F	PR(>F)
C(goodcredit)	1.020136e+06	1.0	27.593917	1.499474e-07
C(rateplan)	1.595291e+08	2.0	2157.572466	0.000000e+00
C(dealertype)	4.105818e+07	3.0	370.197787	3.637456e-239
Residual	3.780067e+09	102248.0	NaN	NaN

In [ ]:
```
```

In [917]:
```
# 6) Test the general association between the account status and “Good Cred
# “RatePlan ” and “DealerType.”
```

In [918]:
```
p6_1 = data[['actdt','deactdt','goodcredit','rateplan','dealertype','sales'
p6_1
account_status = []
account_status = p6_1.deactdt
account_status.loc[~account_status.isnull()] =0
account_status = account_status.replace(np.nan, 1)
account_status
p6_1['account_status'] = (account_status)
p6_1['deactdt'] = data['deactdt']
p6_1
```

Out[918]:

	actdt	deactdt	goodcredit	rateplan	dealertype	sales	AGE	account_status
0	06/20/1999	NaN	0	1	A1	128.0	58.0	1
1	10/04/1999	10/15/1999	1	1	A1	72.0	45.0	0
2	07/01/2000	NaN	0	1	A1	593.0	57.0	1
3	05/30/2000	NaN	1	2	A1	83.0	47.0	1
4	12/13/2000	NaN	1	1	C1	NaN	82.0	1
...	...	...	...	...	...	...	...	...
102250	12/29/2000	NaN	1	1	A2	112.0	50.0	1
102251	01/15/2001	NaN	1	2	A1	87.0	40.0	1
102252	01/15/2001	NaN	1	1	A1	316.0	16.0	1
102253	01/15/2001	NaN	1	2	B1	NaN	76.0	1
102254	01/17/2001	NaN	1	1	A1	319.0	46.0	1

102255 rows × 8 columns

In [919]:
```
model = ols('account_status ~ C(goodcredit)+C(rateplan)+C(dealertype)', dat
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
#p-value of goodcredit, rateplan, dealertype are way less than 0.05, reject
#account_status and C(goodcredit)+C(rateplan)+C(dealertype)
```

Out[919]:

	sum_sq	df	F	PR(>F)
C(goodcredit)	316.307384	1.0	2095.319986	0.000000e+00
C(rateplan)	81.004500	2.0	268.299691	6.075910e-117
C(dealertype)	23.344894	3.0	51.547982	2.779235e-33
Residual	15435.254586	102248.0	NaN	NaN

In [ ]:

In [920]:
```
# 7) Is there any association between the account status and the tenure seg
# Could you find out a better tenure segmentation strategy that is more ass
# with the account status?
```

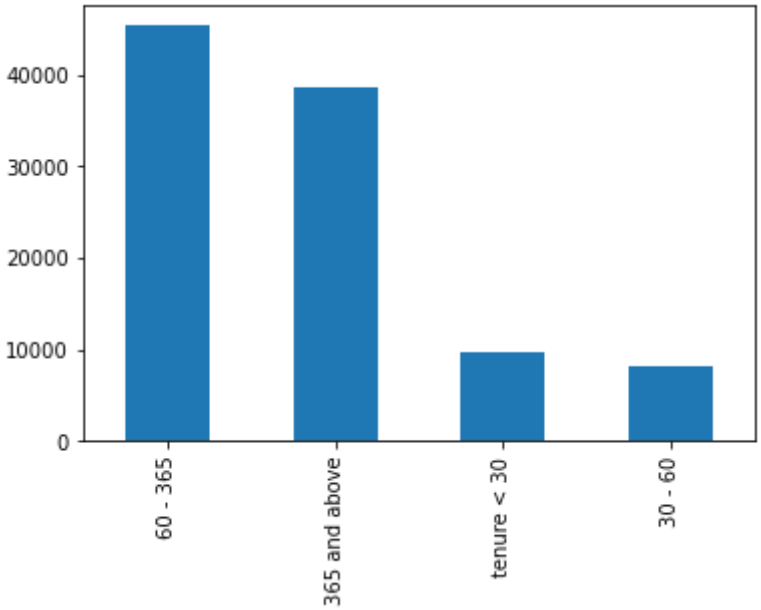
In [921]:
```
data['tenure'] = p4['tenure']
data['tenure_category'] = p4['tenure_category']
data['account_status'] = p6_1['account_status']

data

model = ols('account_status ~ C(tenure_category)', data=data).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table

#There is no association between the account status and the tenure segments
```

Out[921]:

	sum_sq	df	F	PR(>F)
C(tenure_category)	618.626423	3.0	1397.93513	0.0
Residual	15020.309304	101826.0	NaN	NaN

In [922]:
```
tenure_category.value_counts().plot(kind='bar')
```

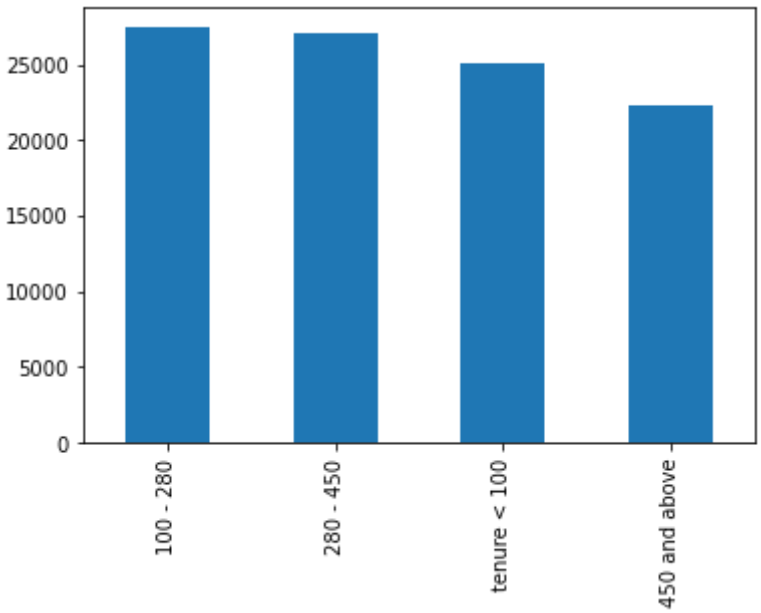
Out[922]: <AxesSubplot:>



In [923]:
```
list_tenure = [0,100,280,450, max(data.tenure)+1]
tenure_category2 = data_cut(data,'tenure',list_tenure)
```

In [924]:
```
tenure_category2.value_counts().plot(kind='bar')
```

Out[924]: <AxesSubplot:>



In [925]:
```
data['tenure_category2'] = tenure_category2
try2 = data[1:200]
try2
model = ols('account_status ~ (C(tenure_category2))', data=try2).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
```

Out[925]:

	sum_sq	df	F	PR(>F)
C(tenure_category2)	0.565953	3.0	1.30089	0.275395
Residual	28.278268	195.0	NaN	NaN

In [926]:
```
data['tenure_category2'] = tenure_category2
data['pro'] = pro

try3 = data[data['pro'] == 'West Provinces']
model = ols('account_status ~ (C(tenure_category2))', data=try3).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
```

Out[926]:

	sum_sq	df	F	PR(>F)
C(tenure_category2)	154.248948	3.0	344.637158	2.652725e-220
Residual	4799.880399	32173.0	NaN	NaN

In [927]:
```
try3 = data[data['pro'] == 'Central Provinces']
model = ols('account_status ~ (C(tenure_category2))', data=try3).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
anova_table
```

Out[927]:

	sum_sq	df	F	PR(>F)
C(tenure_category2)	202.968432	3.0	451.359658	1.484424e-289
Residual	7836.013187	52277.0	NaN	NaN

In [928]:
```
# could not find a better tenure segmentation strategy that is more associa
```

In [ ]:
```

```

In [929]:
```
# 8) Does Sales amount differ among different account status, GoodCredit, a
# customer age segments?
```

In [930]:
```
p8 = data[['account_status','goodcredit','age_category','sales']]
```

In [949]:
```
p8_sample = p8.sample(n=1000, random_state=1)
p8_sample
```

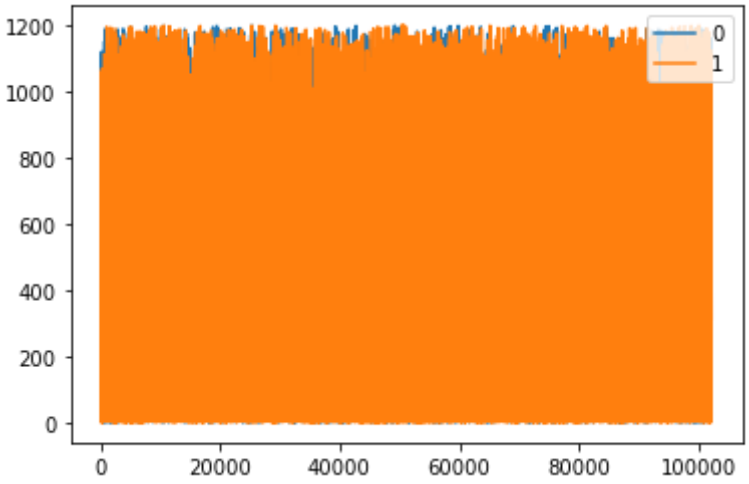
Out[949]:

	account_status	goodcredit	age_category	sales
56506	1	1	40 - 60	301.0
26004	1	1	40 - 60	48.0
19849	1	0	AGE < 20	130.0
63293	1	0	40 - 60	141.0
63744	1	0	AGE < 20	636.0
...	...	...	...	...
62642	1	0	60 and above	17.0
32759	1	1	20 - 40	NaN
51335	1	1	AGE < 20	782.0
93014	1	1	60 and above	47.0
101740	1	1	20 - 40	84.0

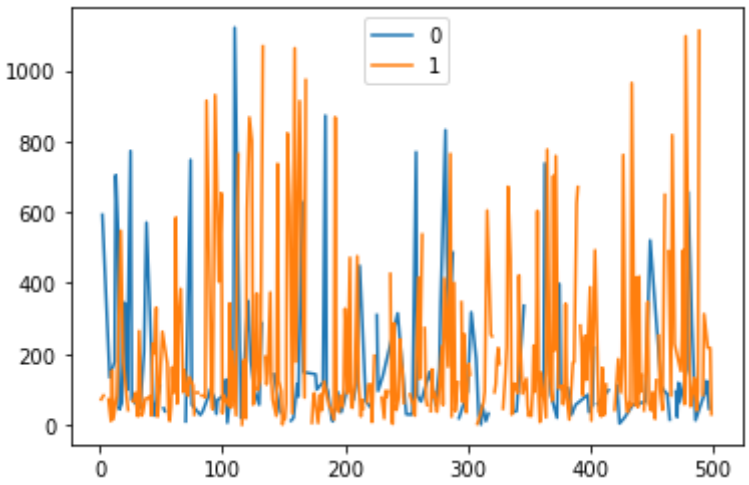
1000 rows × 4 columns

In [989]:
```
p8.groupby('goodcredit')['sales'].plot(legend=True)
```

Out[989]:
```
goodcredit
0    AxesSubplot(0.125,0.125;0.775x0.755)
1    AxesSubplot(0.125,0.125;0.775x0.755)
Name: sales, dtype: object
```

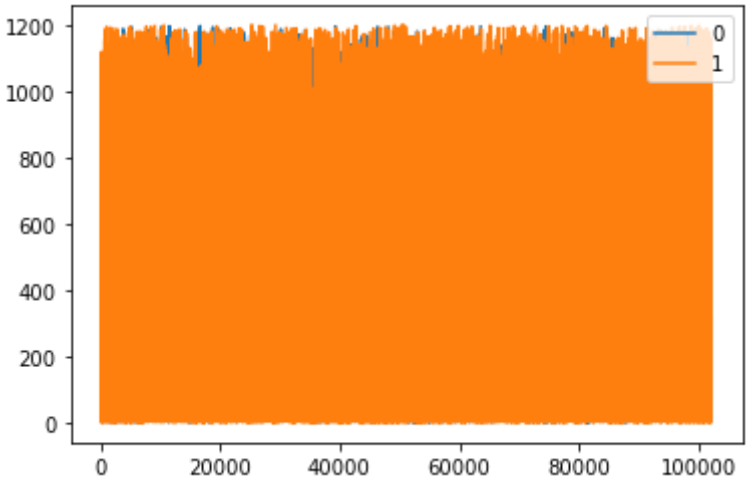


In [990]:
```
pic_num = np.arange(1, 205)
len(pic_num)
plt.clf()
a = 1
for i in range(len(pic_num)):
    b = 500*(i+1)
    p8[a:b].groupby('goodcredit')['sales'].plot(legend=True)
    plt.show()
    a = b
    plt.clf()

# Overall, sales are likely higher from customers with goodcredit than cust
```

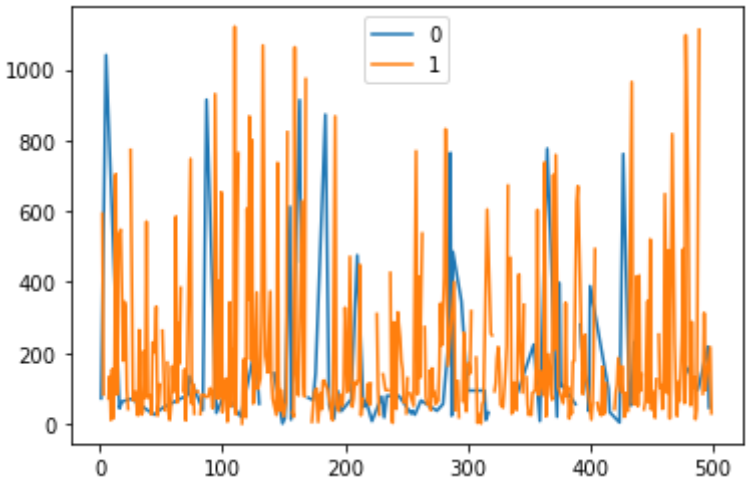


In [992]:
```
p8.groupby('account_status')['sales'].plot(legend=True)
```

Out[992]:
```
account_status
0    AxesSubplot(0.125,0.125;0.775x0.755)
1    AxesSubplot(0.125,0.125;0.775x0.755)
Name: sales, dtype: object
```

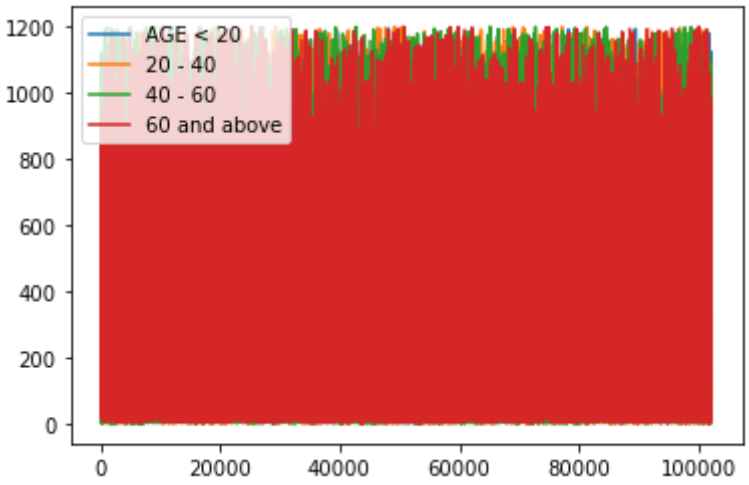


In [991]:
```
plt.clf()
a = 1
for i in range(len(pic_num)):
    b = 500*(i+1)
    p8[a:b].groupby('account_status')['sales'].plot(legend=True)
    plt.show()
    a = b
    plt.clf()


# sales from activated customers are way higher than deactivated customers
```

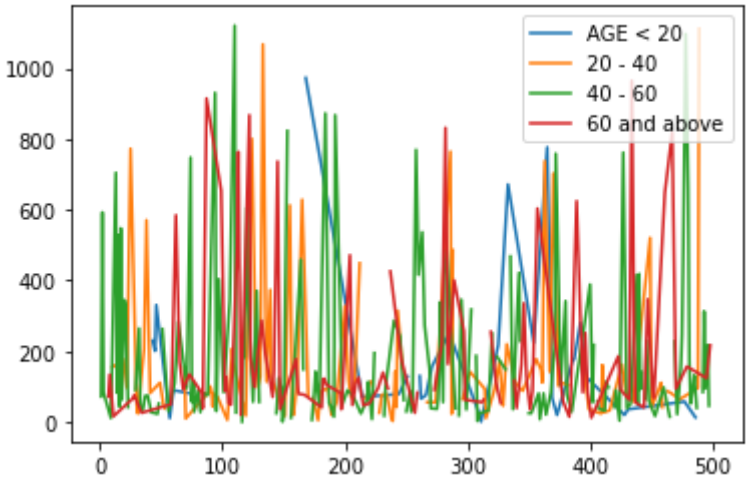


In [993]:
```
p8.groupby('age_category')['sales'].plot(legend=True)
```

Out[993]:
```
age_category
AGE < 20          AxesSubplot(0.125,0.125;0.775x0.755)
20 - 40           AxesSubplot(0.125,0.125;0.775x0.755)
40 - 60           AxesSubplot(0.125,0.125;0.775x0.755)
60 and above      AxesSubplot(0.125,0.125;0.775x0.755)
Name: sales, dtype: object
```



In [994]:
```
plt.clf()
a = 1
for i in range(len(pic_num)):
    b = 500*(i+1)
    p8[a:b].groupby('age_category')['sales'].plot(legend=True)
    plt.show()
    a = b
    plt.clf()


# sales from age<20 customers are the least
# sales from 20-40, 40-60 customers are similar
# sales from 60 and above customers are less than 20-40, 40-60 customers bu
```



In [ ]:

In [ ]: