In [4]:

```
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
import seaborn as sns
import os

data = pd.read_csv("D:\\javeed\\DS documents\\abid dataset\\electric motor
temp\\pmsm_temperature_data.csv")
```

In [5]:

```
##############stratify sampling############
from sklearn.model_selection import train_test_split
train, test = train_test_split(data, test_size = 0.2, stratify=data.profile_id)
```

In [4]:

```
#only profile_id is categorical data
data.columns
```

Out[4]:

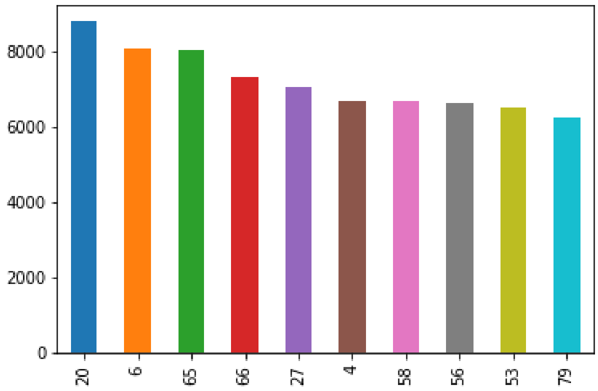
```
Index(['ambient', 'coolant', 'u_d', 'u_q', 'motor_speed', 'torque', 'i_d',
       'i_q', 'pm', 'stator_yoke', 'stator_tooth', 'stator_winding',
       'profile_id'],
      dtype='object')
```

In [5]:

```
########bar plot#########
test['profile_id'].value_counts().head(10).plot.bar()
```

Out[5]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2071c5b6630>
```

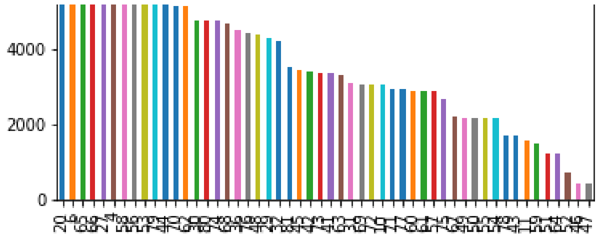


In [6]:

```
test['profile_id'].value_counts().plot.bar() # 20 profile_id is mostly occuring
```

Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2071c5b6240>
```

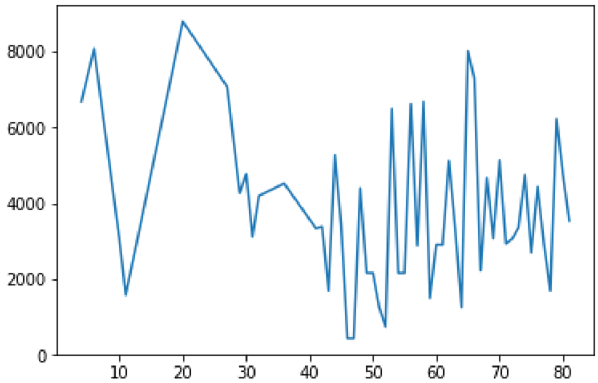


In [7]:

```
##########line plot######
test['profile_id'].value_counts().sort_index().plot.line()# as line plot also showing 20 is the mo
st occuring profile_id
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2071df10438>
```

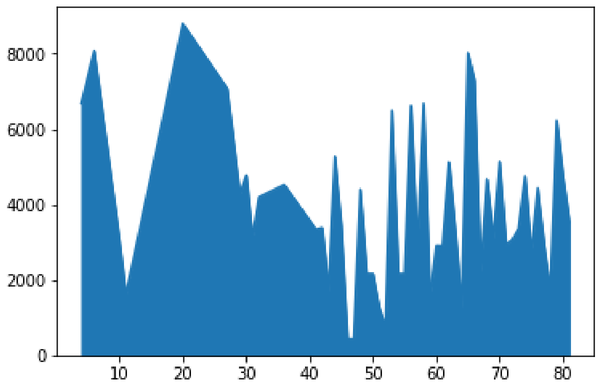


In [8]:

```
##########area plot######
test.profile_id.value_counts().sort_index().plot.area()
#as line plot also showing 20 is the most occuring profile_id
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2071df90358>
```

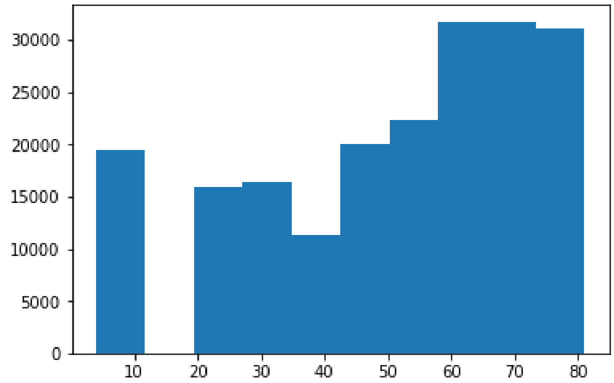


In [9]:

```
plt.hist(test.profile_id)
#data is not normal
```

Out[9]:

```
(array([19390.,     0., 15866., 16353., 11246., 19977., 22311., 31704.,
        31741., 31026.]),
 array([ 4. , 11.7, 19.4, 27.1, 34.8, 42.5, 50.2, 57.9, 65.6, 73.3, 81. ]),
 <a list of 10 Patch objects>)
```

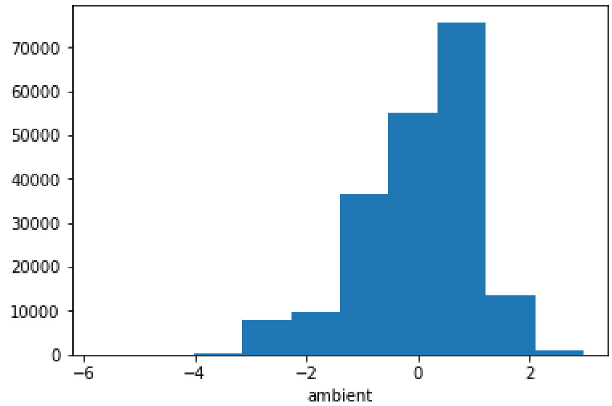


In [13]:

```
plt.hist(test.ambient);plt.xlabel('ambient')
#data is not normal
```

Out[13]:

```
Text(0.5, 0, 'ambient')
```

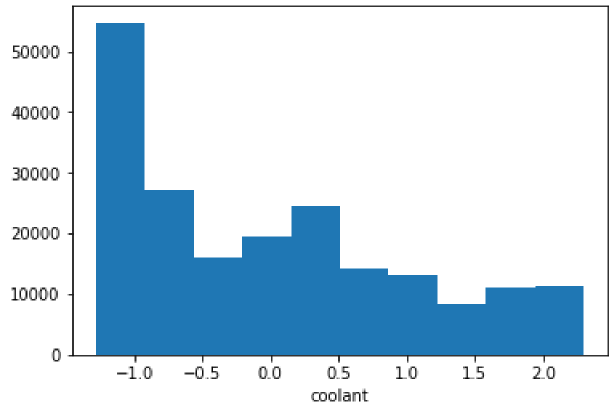


In [15]:

```
plt.hist(test.coolant);plt.xlabel('coolant')
#data is not normal
```

Out[15]:

```
Text(0.5, 0, 'coolant')
```



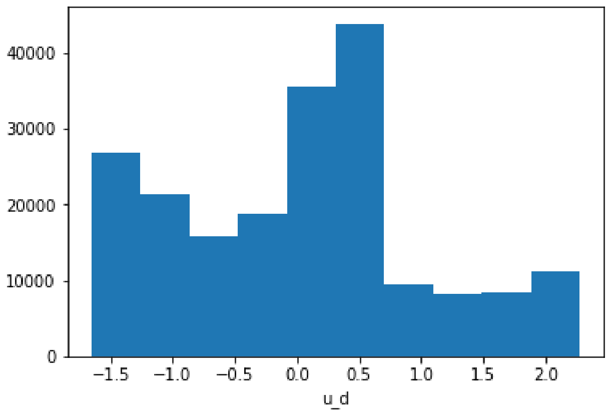
In [16]:

```
plt.hist(test.u_d);plt.xlabel('u_d')
#data is not normal
```

```
#data is not normal
```

Out[16]:

```
Text(0.5, 0, 'u_d')
```

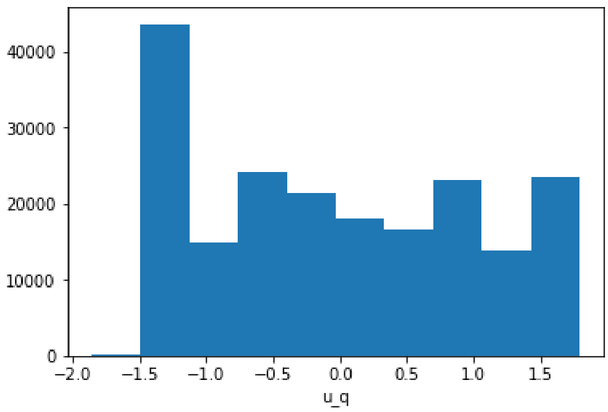


In [18]:

```python
plt.hist(test.u_q);plt.xlabel('u_q')
#data is not normal
```

Out[18]:

```
Text(0.5, 0, 'u_q')
```

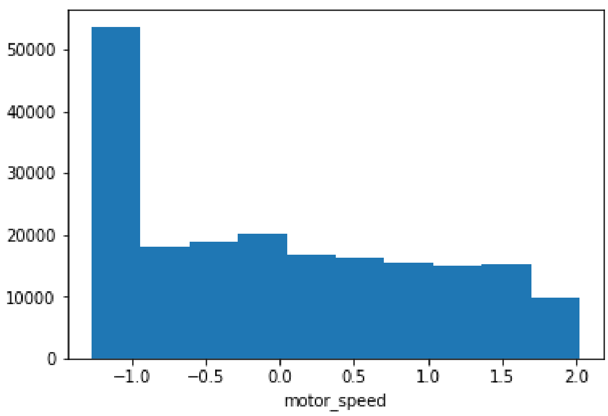


In [19]:

```python
plt.hist(test.motor_speed);plt.xlabel('motor_speed')
#data is not normal
```

Out[19]:

```
Text(0.5, 0, 'motor_speed')
```

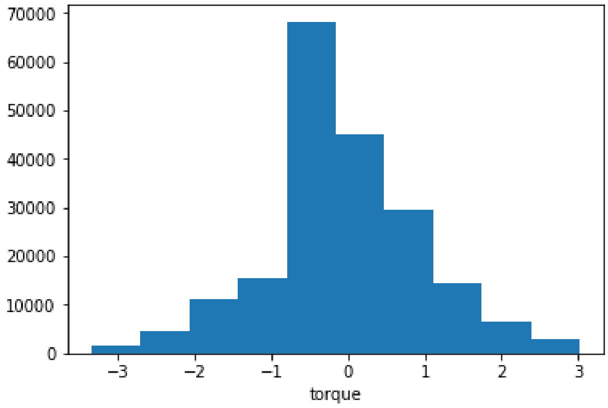


In [21]:

```
plt.hist(test.torque);plt.xlabel('torque')
#kind of normal
```

Out[21]:

```
Text(0.5, 0, 'torque')
```

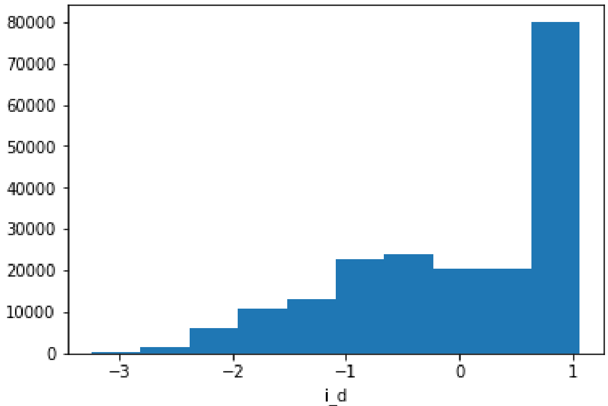


In [22]:

```
plt.hist(test.i_d);plt.xlabel('i_d')
#data is not normal
```

Out[22]:

```
Text(0.5, 0, 'i_d')
```

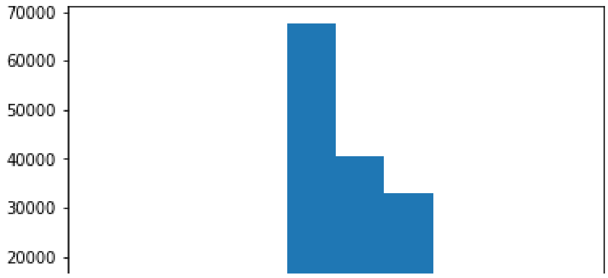


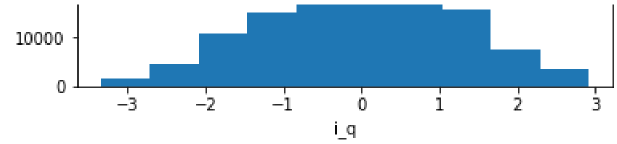
In [23]:

```
plt.hist(test.i_q);plt.xlabel('i_q')
#data is nearly normal
```

Out[23]:

```
Text(0.5, 0, 'i_q')
```



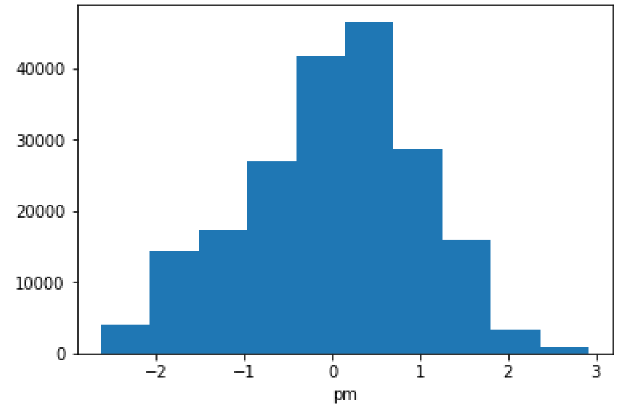


In [24]:

```
plt.hist(test.pm);plt.xlabel('pm')#kind of normal
```

Out[24]:

```
Text(0.5, 0, 'pm')
```

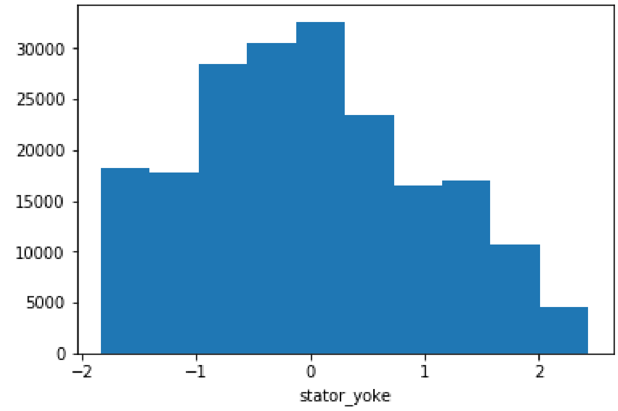


In [25]:

```
plt.hist(test.stator_yoke);plt.xlabel('stator_yoke')#data is not normal
```

Out[25]:

```
Text(0.5, 0, 'stator_yoke')
```

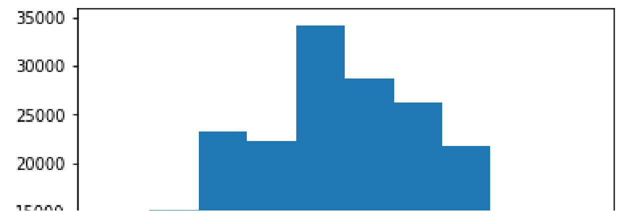


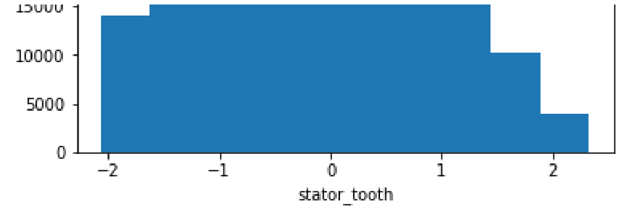
In [26]:

```
plt.hist(test.stator_tooth);plt.xlabel('stator_tooth')#data is not normal
```

Out[26]:

```
Text(0.5, 0, 'stator_tooth')
```



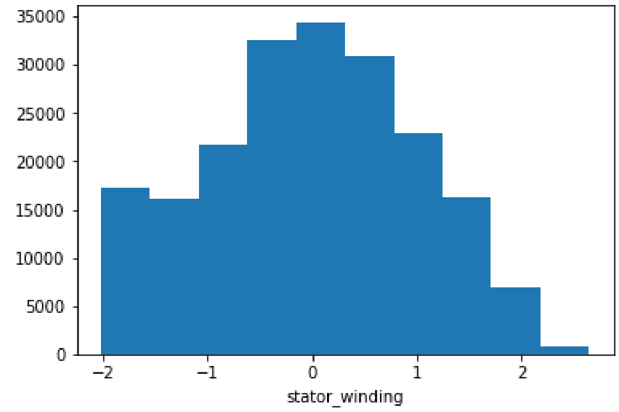


In [27]:

```
plt.hist(test.stator_winding);plt.xlabel('stator_winding')#data is not normal
```

Out[27]:

```
Text(0.5, 0, 'stator_winding')
```

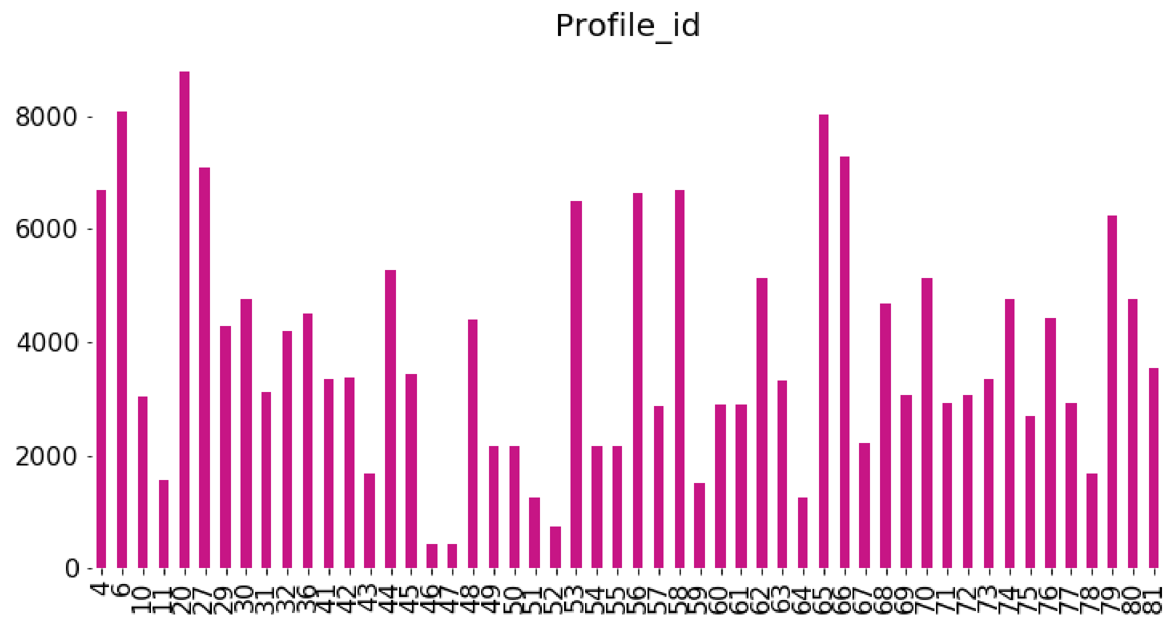


In [1]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [6]:

```
ax = test.profile_id.value_counts().sort_index().plot.bar(
    figsize=(12, 6),
    color='mediumvioletred',
    fontsize=16
)
ax.set_title("Profile_id", fontsize=20)
sns.despine(bottom=True, left=True)
```

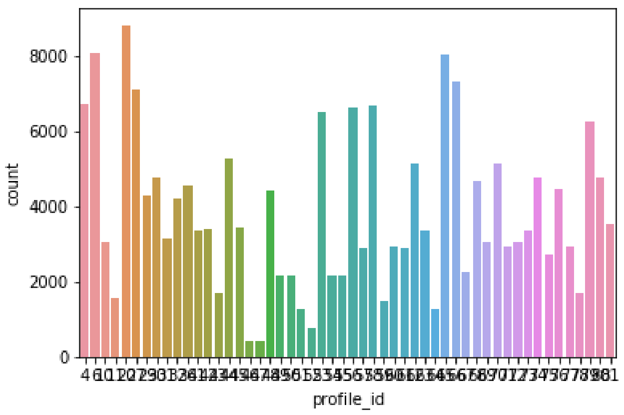


In [11]:

```
#######count plot

sns.countplot(test.profile_id)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x186893108d0>
```

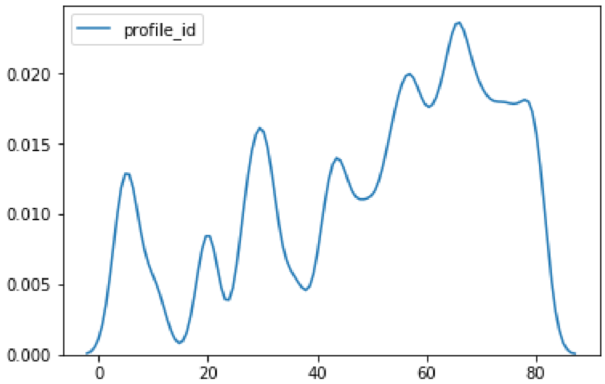


In [31]:

```
#KDE plot

sns.kdeplot(test.profile_id)

sns.kdeplot(test.profile_id, test.ambient)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-31-22b74d811b55> in <module>
      3 sns.kdeplot(test.profile_id)
      4
----> 5 sns.kdeplot(test.profile_id, test.ambient)

D:\javeed\DS documents\anaconda install\lib\site-packages\seaborn\distributions.py in
kdeplot(data, data2, shade, vertical, kernel, bw, gridsize, cut, clip, legend, cumulative,
shade_lowest, cbar, cbar_ax, cbar_kws, ax, **kwargs)
    685         ax = _bivariate_kdeplot(x, y, shade, shade_lowest,
    686                                 kernel, bw, gridsize, cut, clip, legend,
--> 687                                 cbar, cbar_ax, cbar_kws, ax, **kwargs)
    688     else:
    689         ax = _univariate_kdeplot(data, shade, vertical, kernel, bw,

D:\javeed\DS documents\anaconda install\lib\site-packages\seaborn\distributions.py in
_bivariate_kdeplot(x, y, filled, fill_lowest, kernel, bw, gridsize, cut, clip, axlabel, cbar,
cbar_ax, cbar_kws, ax, **kwargs)
    391     # Calculate the KDE
    392     if _has_statsmodels:
--> 393         xx, yy, z = _statsmodels_bivariate_kde(x, y, bw, gridsize, cut, clip)
    394     else:
    395         xx, yy, z = _scipy_bivariate_kde(x, y, bw, gridsize, cut, clip)

D:\javeed\DS documents\anaconda install\lib\site-packages\seaborn\distributions.py in
_statsmodels_bivariate_kde(x, y, bw, gridsize, cut, clip)
    465     y_support = _kde_support(y, kde.bw[1], gridsize, clip[1])
    466     xx, yy = np.meshgrid(x_support, y_support)
--> 467     z = kde.pdf([xx.ravel(), yy.ravel()]).reshape(xx.shape)
    468     return xx, yy, z
    469

D:\javeed\DS documents\anaconda install\lib\site-
packages\statsmodels\nonparametric\kernel_density.py in pdf(self, data_predict)
    194             pdf_est.append(gpke(self.bw, data=self.data,
    195                                 data_predict=data_predict[i, :],
--> 196                                 var_type=self.var_type) / self.nobs)
    197
```

```
    198         pdf_est = np.squeeze(pdf_est)

D:\javeed\DS documents\anaconda install\lib\site-packages\statsmodels\nonparametric\_kernel_base.py in gpke(bw, data, data_predict, var_type, ckertype, okertype, ukertype, tosum)
    508     for ii, vtype in enumerate(var_type):
    509         func = kernel_func[kertypes[vtype]]
--> 510         Kval[:, ii] = func(bw[ii], data[:, ii], data_predict[ii])
    511
    512     iscontinuous = np.array([c == 'c' for c in var_type])

D:\javeed\DS documents\anaconda install\lib\site-packages\statsmodels\nonparametric\kernels.py in gaussian(h, Xi, x)
    126
    127     """
--> 128     return (1. / np.sqrt(2 * np.pi)) * np.exp(-(Xi - x)**2 / (h**2 * 2.))
    129
    130

KeyboardInterrupt:
```

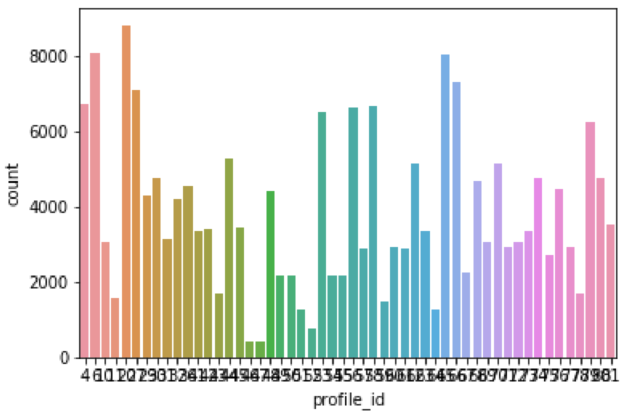


In [10]:

```
#######count plot

sns.countplot(test.profile_id)
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1868946db38>
```



In [30]:

```
import plotly.express as px
fig = px.box(test, y="profile_id")
fig.show()
```

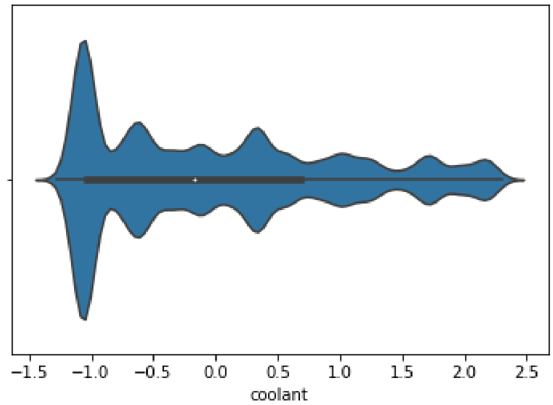
In [14]:

```
sns.violinplot(test.coolant)#As we can see we have a higher density between -0.5 and -1.5
```

```
D:\javeed\DS documents\anaconda install\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning:

Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` inst
ead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`,
which will result either in an error or a different result.
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18698c9a3c8>
```

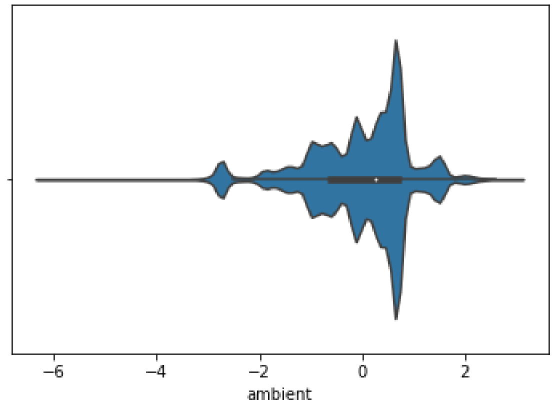


In [15]:

```
sns.violinplot(test.ambient)#As we can see we have a higher density between 0 and 1.
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18689730320>
```

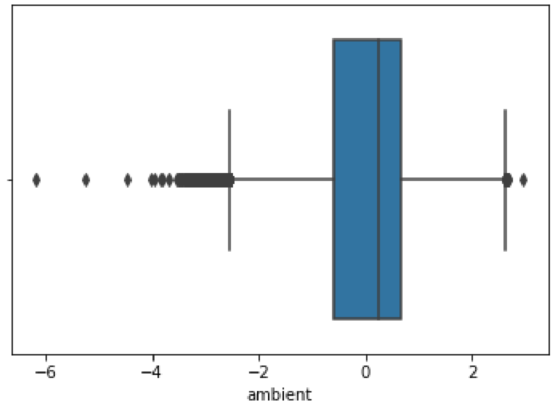


In [16]:

```
sns.boxplot(x=test.ambient)
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18698e72cf8>
```



In [17]:

```
test.columns
```

Out[17]:

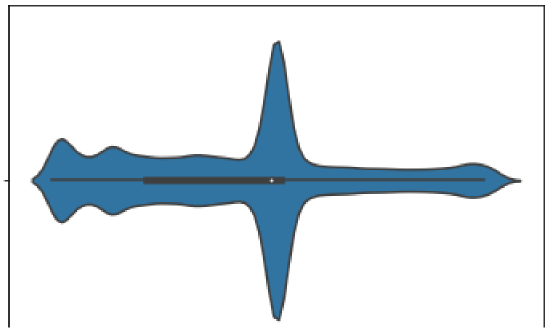
```
Index(['ambient', 'coolant', 'u_d', 'u_q', 'motor_speed', 'torque', 'i_d',
       'i_q', 'pm', 'stator_yoke', 'stator_tooth', 'stator_winding',
       'profile_id'],
      dtype='object')
```

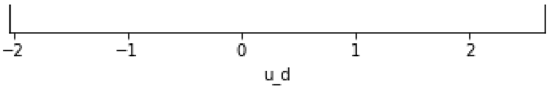
In [18]:

```
sns.violinplot(test.u_d)#As we can see we have a higher density between 0 and 1
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18698cf6eb8>
```



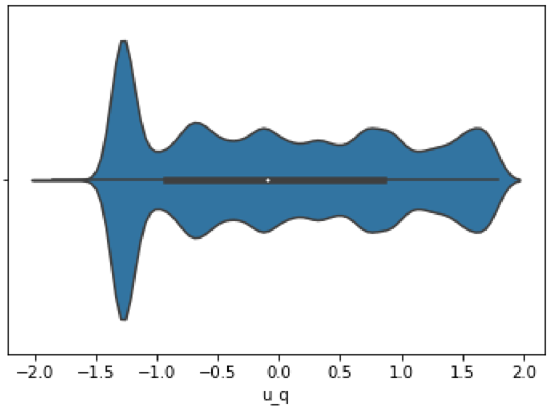


In [19]:

```
sns.violinplot(test.u_q)#As we can see we have a higher density between -1.5 and -1.0
```

Out[19]:

<matplotlib.axes._subplots.AxesSubplot at 0x18698d37748>

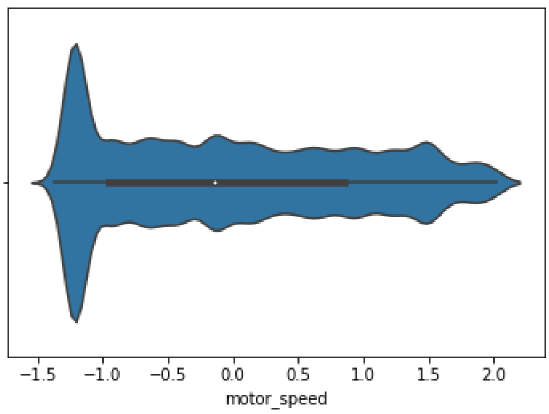


In [21]:

```
sns.violinplot(test.motor_speed)#As we can see we have a higher density between -1.5 and -1.0
```

Out[21]:

<matplotlib.axes._subplots.AxesSubplot at 0x18698ea0048>

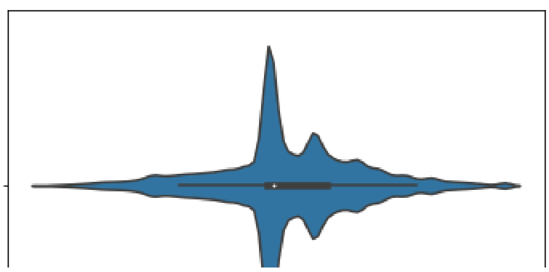


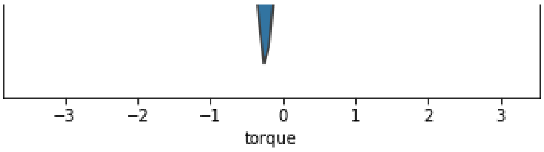
In [22]:

```
sns.violinplot(test.torque)#As we can see we have a higher density between -1.0 and 0
```

Out[22]:

<matplotlib.axes._subplots.AxesSubplot at 0x18698ee66d8>



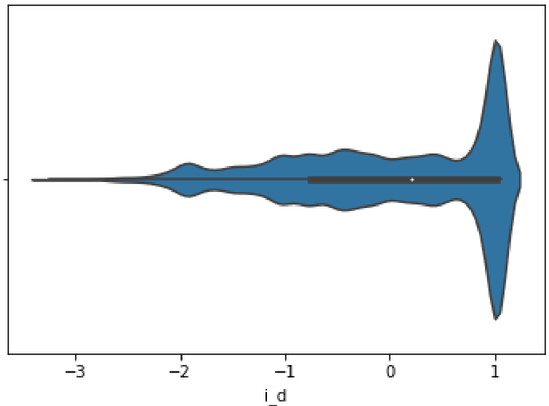


In [23]:

```
sns.violinplot(test.i_d)##As we can see we have a higher density between 0.5 and 1.0
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x186990bf828>
```

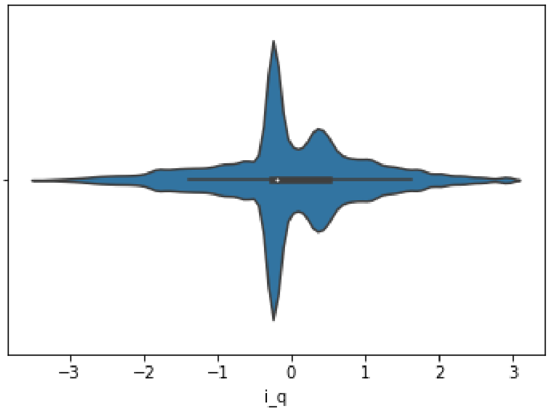


In [24]:

```
sns.violinplot(test.i_q)#As we can see we have a higher density between -1.0 and 0
```

Out[24]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18698f378d0>
```

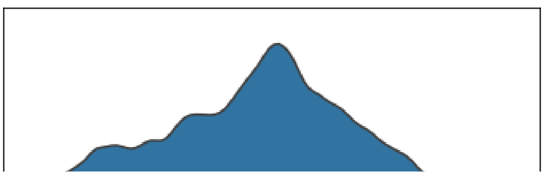


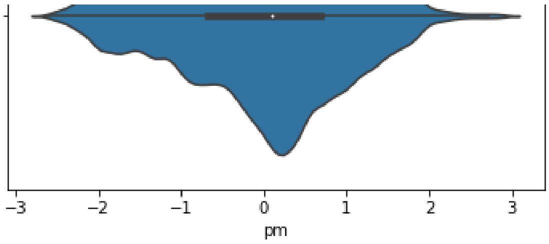
In [25]:

```
sns.violinplot(test.pm)##As we can see we have a higher density between -1.0 and 1.0 . also data i
s normally distributed
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18698fa1c88>
```



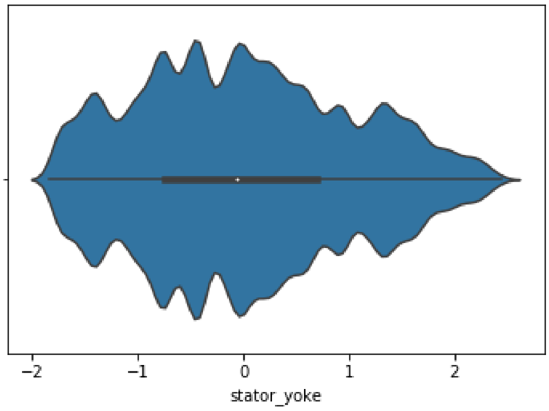


In [26]:

```
sns.violinplot(test.stator_yoke)#As we can see we have a higher density between -1.0 and 1.0 . also data is normally distributed
```

Out[26]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18698fe6da0>
```

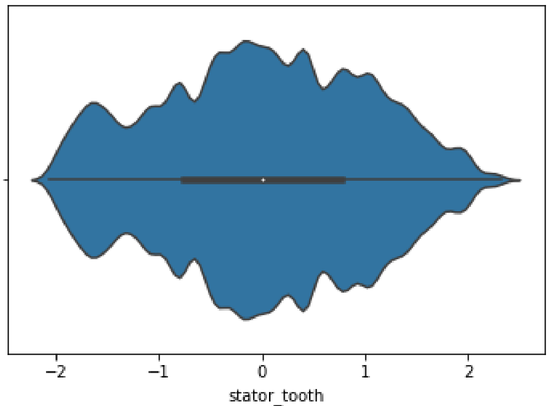


In [27]:

```
sns.violinplot(test.stator_tooth)#As we can see we have a higher density between -1.0 and 1.0 . also data is normally distributed
```

Out[27]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18699034ac8>
```

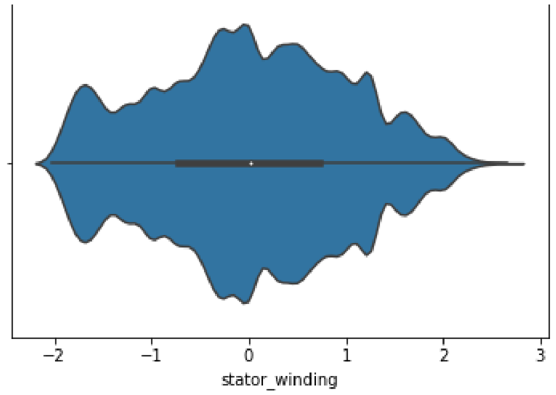


In [28]:

```
sns.violinplot(test.stator_winding)#As we can see we have a higher density between -1.0 and 1.0 . also data is normally distributed
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1869907eef0>
```

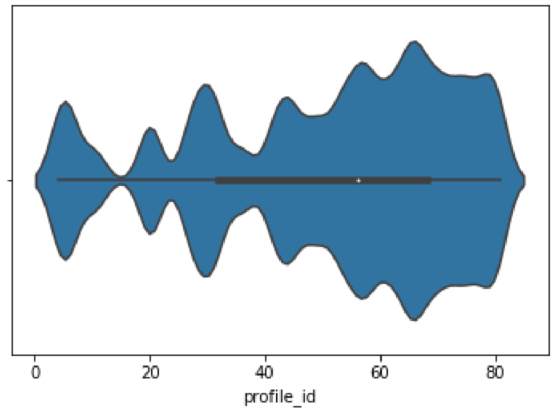


In [29]:

```
sns.violinplot(test.profile_id)#As we can see we have a higher density between 60 and 80
```

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x186991215c0>
```



In [32]:

```
#bivariate analysis
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
import seaborn as sns
import os

data = pd.read_csv("D:\\javeed\\DS documents\\abid dataset\\electric motor
temp\\pmsm_temperature_data.csv")
```

In [33]:

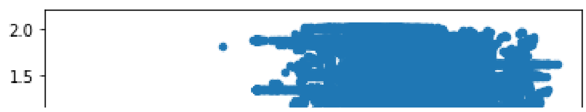
```
##############stratify sampling############
from sklearn.model_selection import train_test_split
train, test = train_test_split(data, test_size = 0.1, stratify=data.profile_id)
```

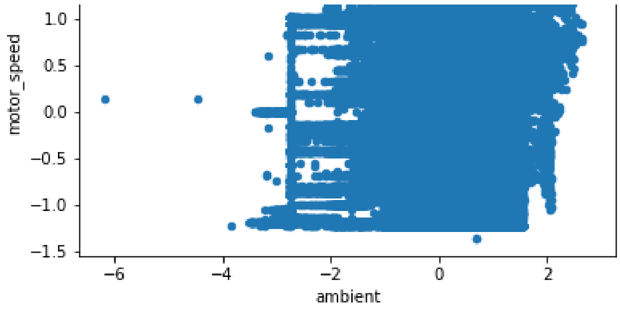
In [34]:

```
#scatter plot
test.plot.scatter(x='ambient', y='motor_speed') # Overplotting
```

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18699839a20>
```



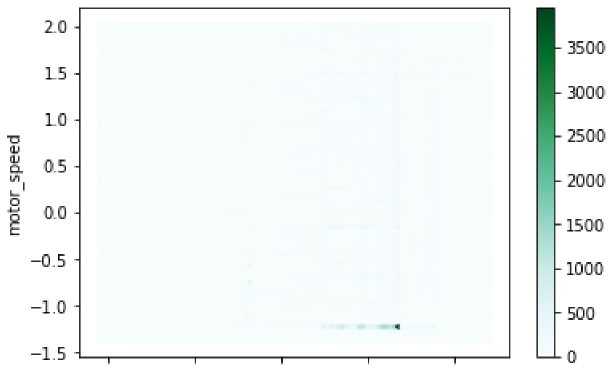


In [35]:

```
#hexplot
test.plot.hexbin(x='ambient', y = 'motor_speed')
```

Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x186880f4b70>
```



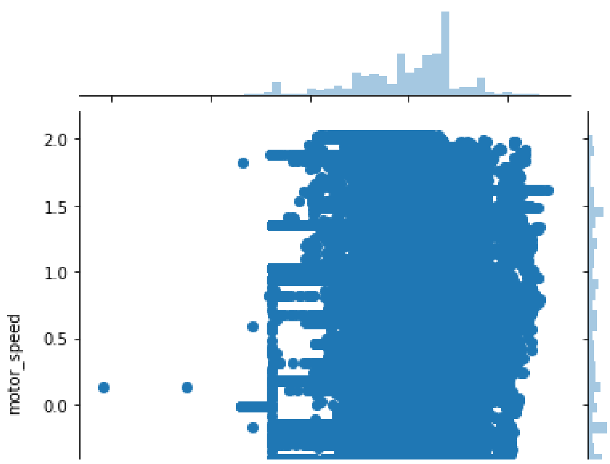
In [37]:

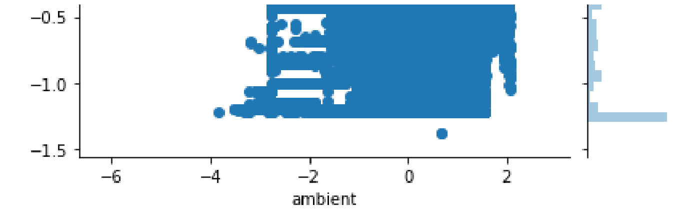
```
#jointplot - scatter plot with histogram


sns.jointplot(x='ambient', y='motor_speed', data=test)
```

```
D:\javeed\DS documents\anaconda install\lib\site-packages\scipy\stats\stats.py:1713:
FutureWarning:

Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` inst
ead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`,
which will result either in an error or a different result.
```

Out[37]:

```
<seaborn.axisgrid.JointGrid at 0x186881be400>
```



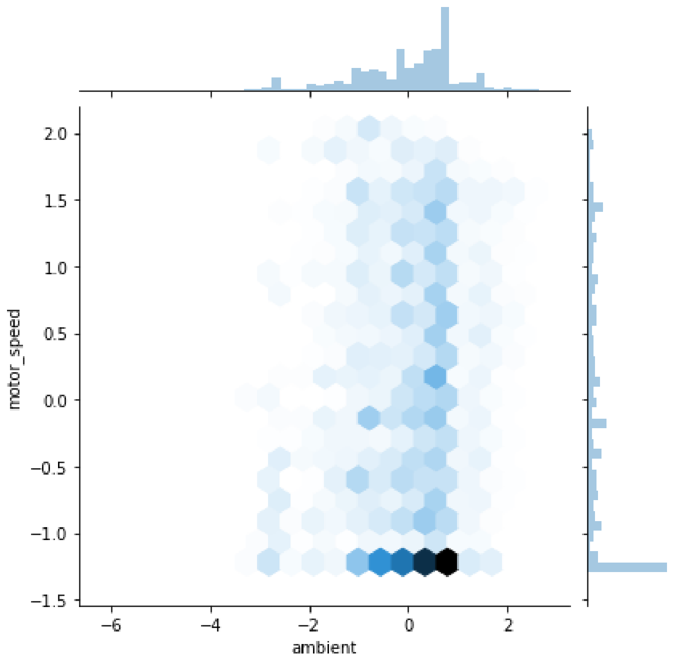


In [38]:

```
#jointplot - hex plot with histogram
sns.jointplot(x='ambient', y='motor_speed', data = test, kind='hex',gridsize=20)
#most of the time ambient 1 for top speed
```

Out[38]:

```
<seaborn.axisgrid.JointGrid at 0x18688367128>
```

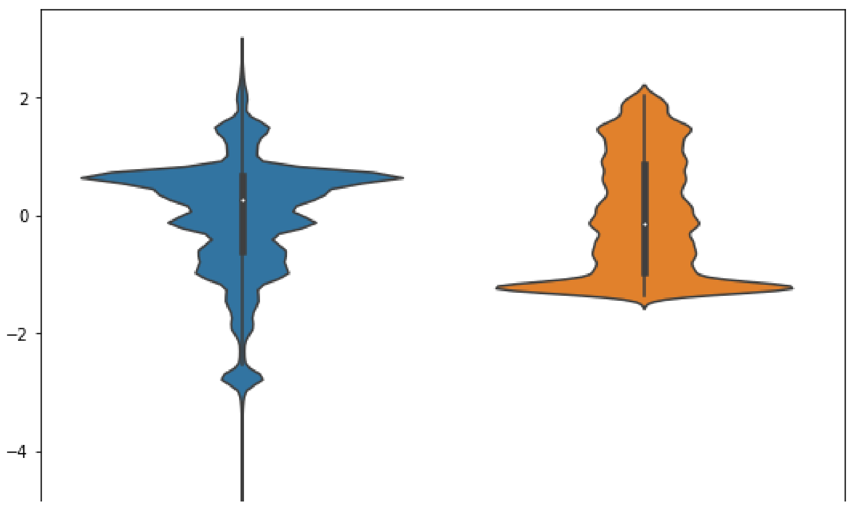


In [42]:

```
#violinplot bivariate analysis
fig, ax = plt.subplots(figsize =(9, 7))
sns.violinplot(ax = ax, data = test.iloc[:, [0,4]])
```

Out[42]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1868882ab70>
```



In [40]:

```
test.columns
```

Out[40]:

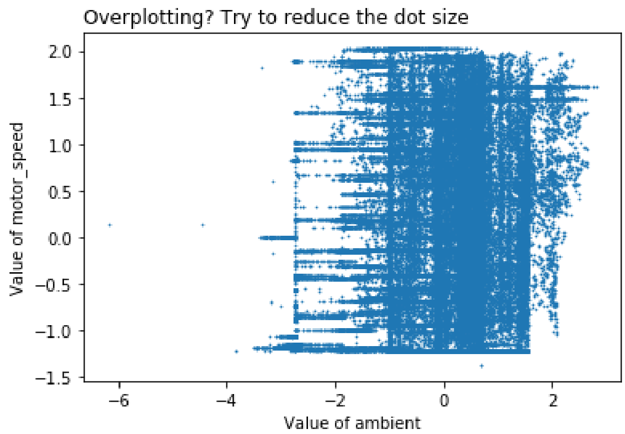
```
Index(['ambient', 'coolant', 'u_d', 'u_q', 'motor_speed', 'torque', 'i_d',
       'i_q', 'pm', 'stator_yoke', 'stator_tooth', 'stator_winding',
       'profile_id'],
      dtype='object')
```

In [43]:

```
#################Overplotting? Try to reduce the dot size#### ##to avoid Overplotting

plt.plot( 'ambient', 'motor_speed', data=test, linestyle='', marker='o', markersize=0.7)
plt.xlabel('Value of ambient')
plt.ylabel('Value of motor_speed')
plt.title('Overplotting? Try to reduce the dot size', loc='left')
```

Out[43]:

```
Text(0.0, 1.0, 'Overplotting? Try to reduce the dot size')
```

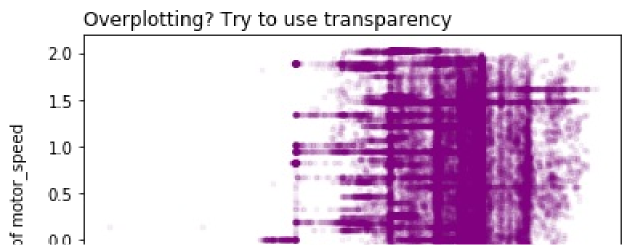


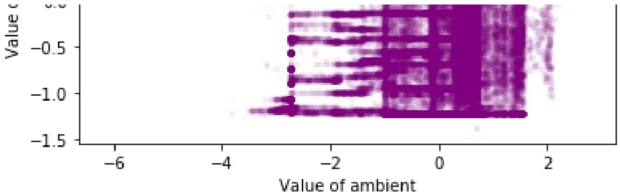
In [44]:

```
############### Plot with transparency
plt.plot( 'ambient', 'motor_speed', data=test, linestyle='', marker='o', markersize=3, alpha=0.05,
color="purple")

# Titles
plt.xlabel('Value of ambient')
plt.ylabel('Value of motor_speed')
plt.title('Overplotting? Try to use transparency', loc='left')
```

Out[44]:

```
Text(0.0, 1.0, 'Overplotting? Try to use transparency')
```



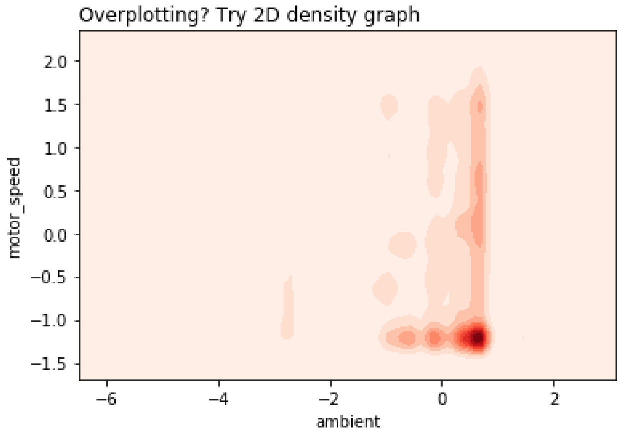


In [45]:

```
# 2D density plot:
sns.kdeplot(test.ambient, test.motor_speed, cmap="Reds", shade=True)
plt.title('Overplotting? Try 2D density graph', loc='left')
```

Out[45]:

```
Text(0.0, 1.0, 'Overplotting? Try 2D density graph')
```

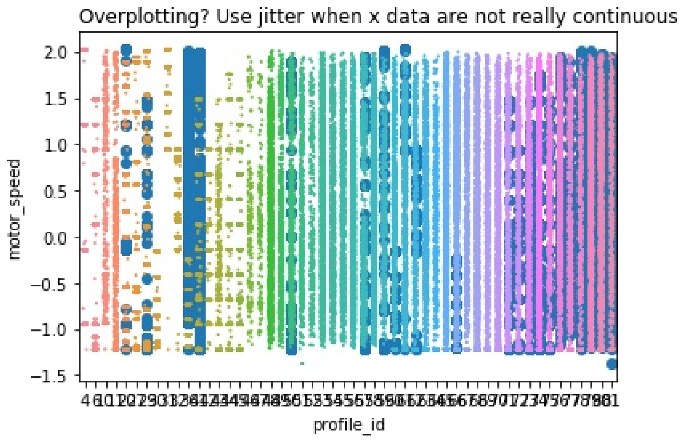


In [55]:

```
## jitter plot
plt.plot('profile_id', 'motor_speed', data=test, linestyle='', marker='o')
# A scatterplot with jitter
sns.stripplot(test.profile_id, test.motor_speed, jitter=0.2, size=2)
plt.title('Overplotting? Use jitter when x data are not really continuous', loc='left')
```

Out[55]:

```
Text(0.0, 1.0, 'Overplotting? Use jitter when x data are not really continuous')
```

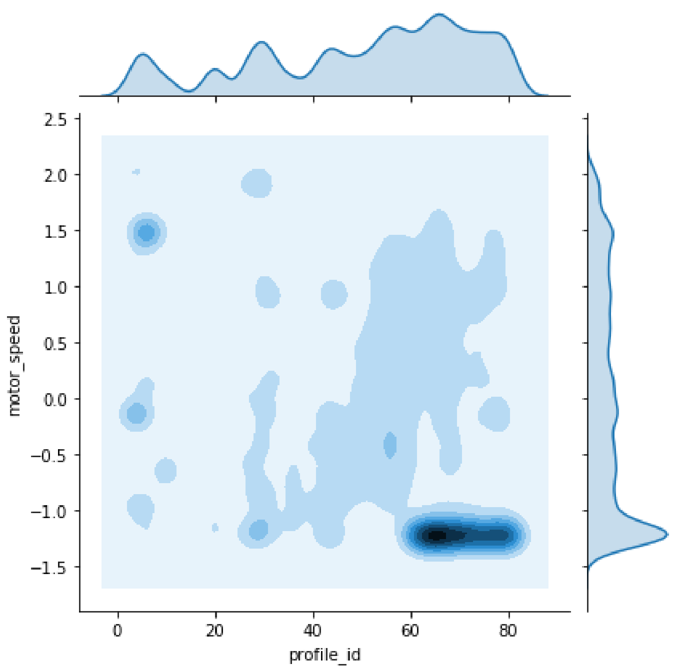


In [56]:

```
# 2D density + marginal distribution: # end Overplotting
sns.jointplot(x=test.profile_id, y=test.motor_speed, kind='kde')
#profile 60 to 80 we have maximum data points
```

Out[56]:

```
<seaborn.axisgrid.JointGrid at 0x18688a2cf28>
```



In [58]:

```
test.corr()
```

Out[58]:

	ambient	coolant	u_d	u_q	motor_speed	torque	i_d	i_q	pm	stator_yoke	stator_tooth
ambient	1.000000	0.435968	0.193876	0.089550	0.079700	-0.261640	0.004153	-0.259916	0.504129	0.454255	0.400112
coolant	0.435968	1.000000	0.178737	0.029656	-0.029132	-0.191232	0.105356	-0.187890	0.434476	0.874417	0.690920
u_d	0.193876	0.178737	1.000000	-0.033531	-0.234527	-0.821703	0.354045	-0.796955	-0.080517	0.043407	-0.062748
u_q	0.089550	0.029656	-0.033531	1.000000	0.718074	-0.032426	-0.183316	-0.021847	0.103508	0.106862	0.149155
motor_speed	0.079700	-0.029132	-0.234527	0.718074	1.000000	0.026430	-0.723185	0.008519	0.332334	0.184723	0.334359
torque	-0.261640	-0.191232	-0.821703	-0.032426	0.026430	1.000000	-0.235471	0.996534	-0.074752	-0.095074	-0.014759
i_d	0.004153	0.105356	0.354045	-0.183316	-0.723185	-0.235471	1.000000	-0.201350	-0.299160	-0.181569	-0.387853
i_q	-0.259916	-0.187890	-0.796955	-0.021847	0.008519	0.996534	-0.201350	1.000000	-0.088323	-0.101644	-0.028716
pm	0.504129	0.434476	-0.080517	0.103508	0.332334	-0.074752	-0.299160	-0.088323	1.000000	0.698294	0.770814
stator_yoke	0.454255	0.874417	0.043407	0.106862	0.184723	-0.095074	-0.181569	-0.101644	0.698294	1.000000	0.950418
stator_tooth	0.400112	0.690920	-0.062748	0.149155	0.334359	-0.014759	-0.387853	-0.028716	0.770814	0.950418	1.000000
stator_winding	0.305852	0.512431	-0.145875	0.125337	0.392796	0.076369	-0.539332	0.056697	0.731815	0.846468	0.965834
profile_id	0.383908	0.497454	0.299761	-0.126799	-0.167936	-0.257732	0.139955	-0.256700	0.156114	0.395612	0.279370

In [ ]: