

In [1]:

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

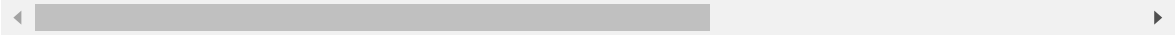
In [2]:

```
df1=pd.read_csv(r'C:\Users\user\Downloads\6_Salesworkload1.csv')
df1
```

Out[2]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	Hour
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	
...	...	...	...	...	...	...	...	...	
7653	06.2017	9.0	Sweden	29650.0	Gothenburg	12.0	Checkout	6322.323	
7654	06.2017	9.0	Sweden	29650.0	Gothenburg	16.0	Customer Services	4270.479	
7655	06.2017	9.0	Sweden	29650.0	Gothenburg	11.0	Delivery	0	
7656	06.2017	9.0	Sweden	29650.0	Gothenburg	17.0	others	2224.929	
7657	06.2017	9.0	Sweden	29650.0	Gothenburg	18.0	all	39652.2	

7658 rows × 14 columns



In [4]:

```
df=df1.head(50)  
df
```

Out[4]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursL
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	
5	10.2016	1.0	United Kingdom	88253.0	London (I)	6.0	Meat	8270.316	
6	10.2016	1.0	United Kingdom	88253.0	London (I)	13.0	Food	16468.251	
7	10.2016	1.0	United Kingdom	88253.0	London (I)	7.0	Clothing	4698.471	
8	10.2016	1.0	United Kingdom	88253.0	London (I)	8.0	Household	1183.272	
9	10.2016	1.0	United Kingdom	88253.0	London (I)	9.0	Hardware	2029.815	
10	10.2016	1.0	United Kingdom	88253.0	London (I)	14.0	Non Food	7911.558	
11	10.2016	1.0	United Kingdom	88253.0	London (I)	15.0	Admin	4308.243	
12	10.2016	1.0	United Kingdom	88253.0	London (I)	12.0	Checkout	5825.097	
13	10.2016	1.0	United Kingdom	88253.0	London (I)	16.0	Customer Services	3320.085	
14	10.2016	1.0	United Kingdom	88253.0	London (I)	11.0	Delivery	0	
15	10.2016	1.0	United Kingdom	88253.0	London (I)	17.0	others	2253.252	
16	10.2016	1.0	United Kingdom	88253.0	London (I)	18.0	all	40086.486	
17	10.2016	1.0	United Kingdom	38976.0	Manchester	1.0	Dry	2583.687	
18	10.2016	1.0	United Kingdom	38976.0	Manchester	2.0	Frozen	5145.345	
19	10.2016	1.0	United Kingdom	38976.0	Manchester	3.0	other	47.205	
20	10.2016	1.0	United Kingdom	38976.0	Manchester	4.0	Fish	3008.532	
21	10.2016	1.0	United Kingdom	38976.0	Manchester	5.0	Fruits & Vegetables	8909.157	
22	10.2016	1.0	United Kingdom	38976.0	Manchester	6.0	Meat	15779.058	

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursL
23	10.2016	1.0	United Kingdom	38976.0	Manchester	13.0	Food	35472.984	
24	10.2016	1.0	United Kingdom	38976.0	Manchester	7.0	Clothing	7889.529	
25	10.2016	1.0	United Kingdom	38976.0	Manchester	8.0	Household	1762.32	
26	10.2016	1.0	United Kingdom	38976.0	Manchester	9.0	Hardware	2797.683	
27	10.2016	1.0	United Kingdom	38976.0	Manchester	14.0	Non Food	12449.532	
28	10.2016	1.0	United Kingdom	38976.0	Manchester	15.0	Admin	6967.458	
29	10.2016	1.0	United Kingdom	38976.0	Manchester	12.0	Checkout	11719.428	
30	10.2016	1.0	United Kingdom	38976.0	Manchester	16.0	Customer Services	5491.515	
31	10.2016	1.0	United Kingdom	38976.0	Manchester	11.0	Delivery	0	
32	10.2016	1.0	United Kingdom	38976.0	Manchester	17.0	others	2300.457	
33	10.2016	1.0	United Kingdom	38976.0	Manchester	18.0	all	74401.374	
34	10.2016	1.0	United Kingdom	17647.0	Liverpool	1.0	Dry	2341.368	
35	10.2016	1.0	United Kingdom	17647.0	Liverpool	2.0	Frozen	3077.766	1
36	10.2016	1.0	United Kingdom	17647.0	Liverpool	3.0	other	47.205	
37	10.2016	1.0	United Kingdom	17647.0	Liverpool	4.0	Fish	2196.606	2
38	10.2016	1.0	United Kingdom	17647.0	Liverpool	5.0	Fruits & Vegetables	3383.025	
39	10.2016	1.0	United Kingdom	17647.0	Liverpool	6.0	Meat	16493.427	7
40	10.2016	1.0	United Kingdom	17647.0	Liverpool	13.0	Food	27539.397	11
41	10.2016	1.0	United Kingdom	17647.0	Liverpool	7.0	Clothing	8528.37	
42	10.2016	1.0	United Kingdom	17647.0	Liverpool	8.0	Household	1957.434	
43	10.2016	1.0	United Kingdom	17647.0	Liverpool	9.0	Hardware	2580.54	
44	10.2016	1.0	United Kingdom	17647.0	Liverpool	14.0	Non Food	13066.344	
45	10.2016	1.0	United Kingdom	17647.0	Liverpool	15.0	Admin	4947.084	
46	10.2016	1.0	United Kingdom	17647.0	Liverpool	12.0	Checkout	8965.803	6
47	10.2016	1.0	United Kingdom	17647.0	Liverpool	16.0	Customer Services	3584.433	

MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursL
-----------	------------	---------	---------	------	---------	------------	----------	--------

```
In [5]:
48 df.info()
Out[5]:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype
---  -
0   MonthYear             50 non-null    object
1   Time index            50 non-null    float64
2   Country               50 non-null    object
3   StoreID               50 non-null    float64
4   City                  50 non-null    object
5   Dept_ID               50 non-null    float64
6   Dept. Name            50 non-null    object
7   HoursOwn              50 non-null    object
8   HoursLease            50 non-null    float64
9   Sales units           50 non-null    float64
10  Turnover              50 non-null    float64
11  Customer               0 non-null     float64
12  Area (m2)             50 non-null    object
13  Opening hours         50 non-null    object
dtypes: float64(7), object(7)
memory usage: 5.6+ KB
```

10.2016	1.0	United Kingdom	17647.0	Liverpool	11.0	Delivery	0
10.2016	1.0	United Kingdom	17647.0	Liverpool	17.0	others	2624.598

```
In [6]:
df.describe()
Out[6]:
```

	Time index	StoreID	Dept_ID	HoursLease	Sales units	Turnover	Customer
count	50.0	50.000000	50.000000	50.000000	5.000000e+01	5.000000e+01	0.0
mean	1.0	48904.900000	9.300000	60.960000	1.231844e+06	4.066998e+06	NaN
std	0.0	29839.520941	5.304022	213.640644	2.088301e+06	6.868434e+06	NaN
min	1.0	17647.000000	1.000000	0.000000	0.000000e+00	0.000000e+00	NaN
25%	1.0	17647.000000	5.000000	0.000000	5.504125e+04	1.477058e+05	NaN
50%	1.0	38976.000000	9.000000	0.000000	3.093425e+05	7.400520e+05	NaN
75%	1.0	88253.000000	14.000000	0.000000	9.128262e+05	3.521022e+06	NaN
max	1.0	88253.000000	18.000000	1152.000000	7.476680e+06	2.571973e+07	NaN

```
dtypes: float64(7), object(7)
memory usage: 5.6+ KB
```

```
In [7]:
df.columns
Out[7]:
```

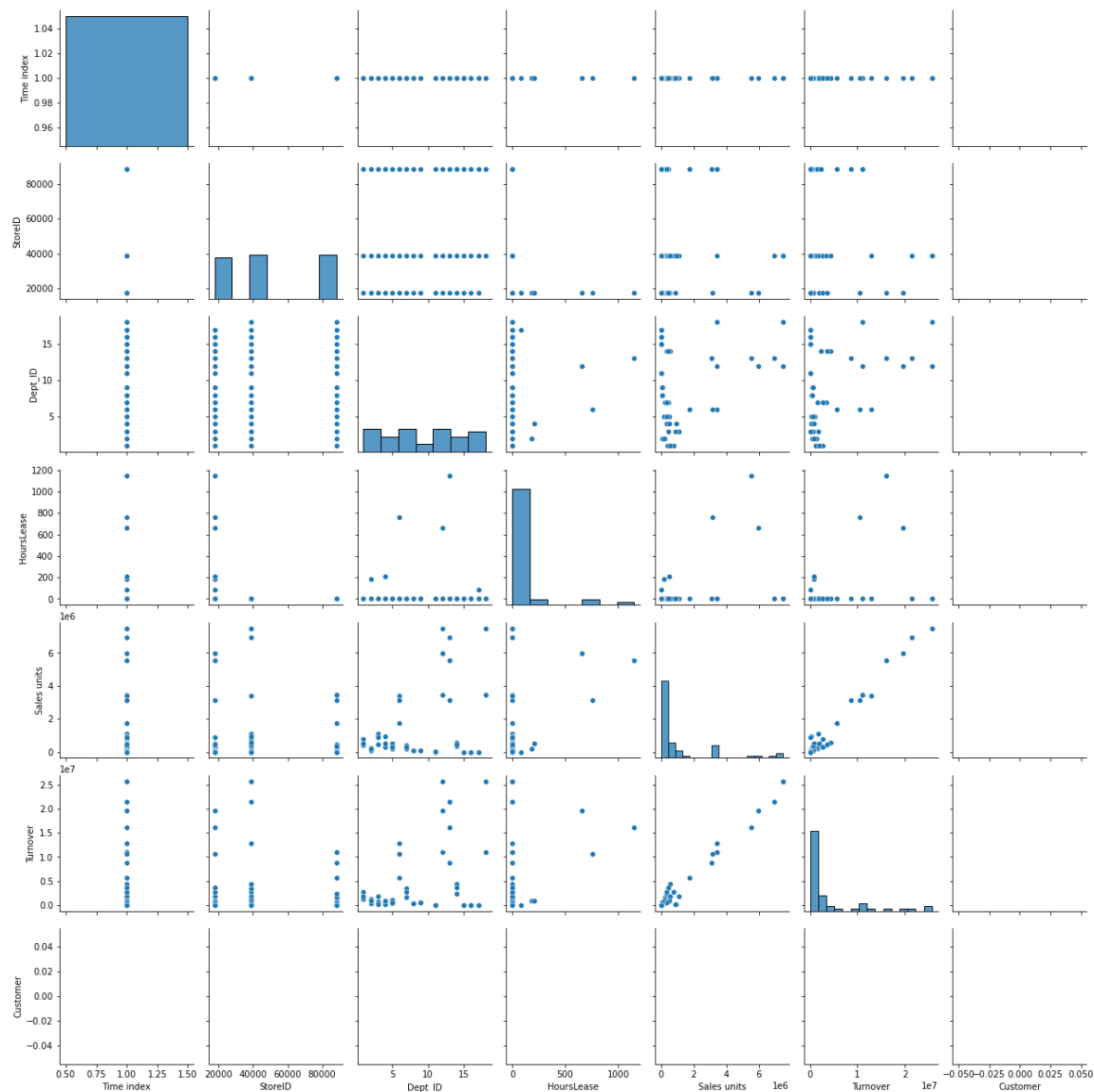
```
Index(['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Customer', 'Area (m2)', 'Opening hours'],
      dtype='object')
```

In [8]:

```
sns.pairplot(df)
```

Out[8]:

&lt;seaborn.axisgrid.PairGrid at 0x16fa2965370&gt;



In [9]:

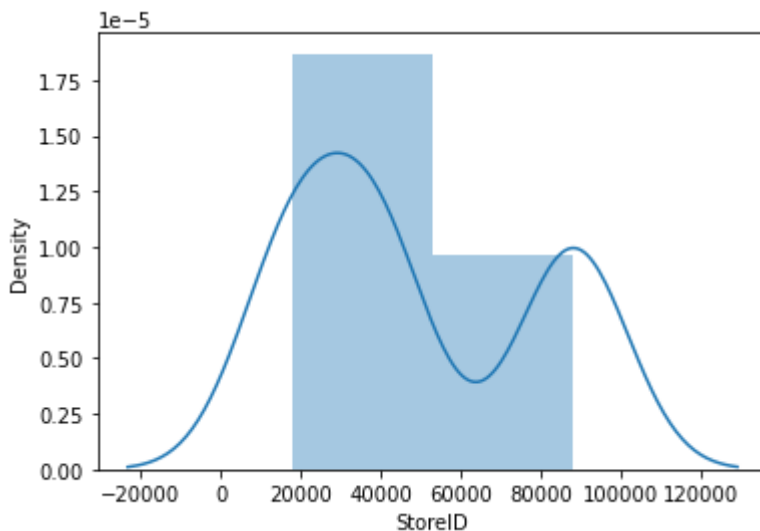
```
sns.distplot(df['StoreID'])
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:  
FutureWarning: `distplot` is a deprecated function and will be removed in  
a future version. Please adapt your code to use either `displot` (a figure  
-level function with similar flexibility) or `histplot` (an axes-level fun  
ction for histograms).

```
warnings.warn(msg, FutureWarning)
```

Out[9]:

```
<AxesSubplot:xlabel='StoreID', ylabel='Density'>
```

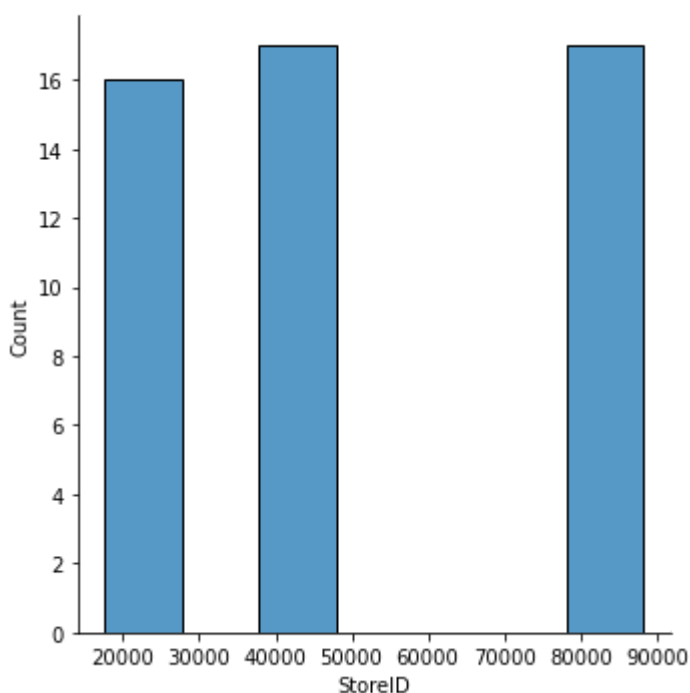


In [10]:

```
sns.displot(df["StoreID"])
```

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x16fa55f58e0>
```



In [11]:

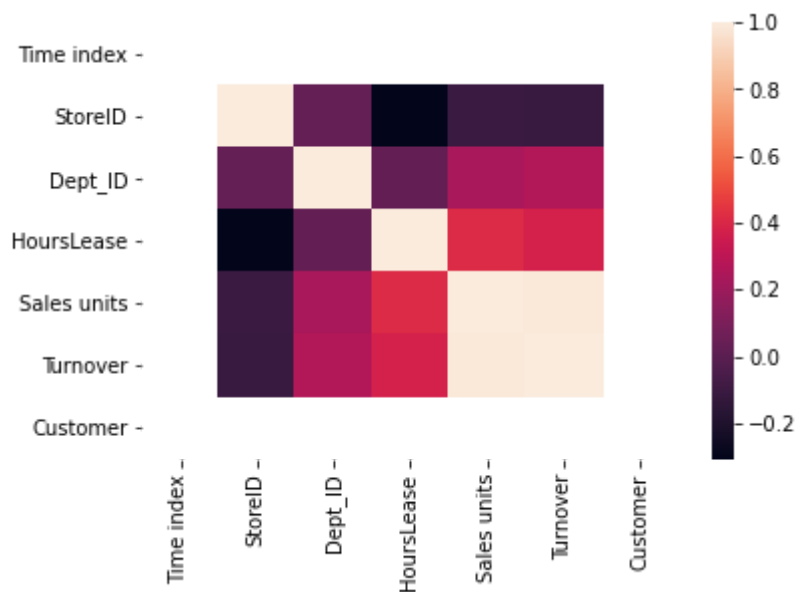
```
df1=df[['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',  
        'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',  
        'Customer', 'Area (m2)', 'Opening hours']]
```

In [12]:

```
sns.heatmap(df1.corr())
```

Out[12]:

<AxesSubplot:>





In [13]:

```
df2=df.dropna(axis=1)  
df2
```

Out[13]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursL
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	
5	10.2016	1.0	United Kingdom	88253.0	London (I)	6.0	Meat	8270.316	
6	10.2016	1.0	United Kingdom	88253.0	London (I)	13.0	Food	16468.251	
7	10.2016	1.0	United Kingdom	88253.0	London (I)	7.0	Clothing	4698.471	
8	10.2016	1.0	United Kingdom	88253.0	London (I)	8.0	Household	1183.272	
9	10.2016	1.0	United Kingdom	88253.0	London (I)	9.0	Hardware	2029.815	
10	10.2016	1.0	United Kingdom	88253.0	London (I)	14.0	Non Food	7911.558	
11	10.2016	1.0	United Kingdom	88253.0	London (I)	15.0	Admin	4308.243	
12	10.2016	1.0	United Kingdom	88253.0	London (I)	12.0	Checkout	5825.097	
13	10.2016	1.0	United Kingdom	88253.0	London (I)	16.0	Customer Services	3320.085	
14	10.2016	1.0	United Kingdom	88253.0	London (I)	11.0	Delivery	0	
15	10.2016	1.0	United Kingdom	88253.0	London (I)	17.0	others	2253.252	
16	10.2016	1.0	United Kingdom	88253.0	London (I)	18.0	all	40086.486	
17	10.2016	1.0	United Kingdom	38976.0	Manchester	1.0	Dry	2583.687	
18	10.2016	1.0	United Kingdom	38976.0	Manchester	2.0	Frozen	5145.345	
19	10.2016	1.0	United Kingdom	38976.0	Manchester	3.0	other	47.205	
20	10.2016	1.0	United Kingdom	38976.0	Manchester	4.0	Fish	3008.532	
21	10.2016	1.0	United Kingdom	38976.0	Manchester	5.0	Fruits & Vegetables	8909.157	
22	10.2016	1.0	United Kingdom	38976.0	Manchester	6.0	Meat	15779.058	

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursL
23	10.2016	1.0	United Kingdom	38976.0	Manchester	13.0	Food	35472.984	
24	10.2016	1.0	United Kingdom	38976.0	Manchester	7.0	Clothing	7889.529	
25	10.2016	1.0	United Kingdom	38976.0	Manchester	8.0	Household	1762.32	
26	10.2016	1.0	United Kingdom	38976.0	Manchester	9.0	Hardware	2797.683	
27	10.2016	1.0	United Kingdom	38976.0	Manchester	14.0	Non Food	12449.532	
28	10.2016	1.0	United Kingdom	38976.0	Manchester	15.0	Admin	6967.458	
29	10.2016	1.0	United Kingdom	38976.0	Manchester	12.0	Checkout	11719.428	
30	10.2016	1.0	United Kingdom	38976.0	Manchester	16.0	Customer Services	5491.515	
31	10.2016	1.0	United Kingdom	38976.0	Manchester	11.0	Delivery	0	
32	10.2016	1.0	United Kingdom	38976.0	Manchester	17.0	others	2300.457	
33	10.2016	1.0	United Kingdom	38976.0	Manchester	18.0	all	74401.374	
34	10.2016	1.0	United Kingdom	17647.0	Liverpool	1.0	Dry	2341.368	
35	10.2016	1.0	United Kingdom	17647.0	Liverpool	2.0	Frozen	3077.766	1
36	10.2016	1.0	United Kingdom	17647.0	Liverpool	3.0	other	47.205	
37	10.2016	1.0	United Kingdom	17647.0	Liverpool	4.0	Fish	2196.606	2
38	10.2016	1.0	United Kingdom	17647.0	Liverpool	5.0	Fruits & Vegetables	3383.025	
39	10.2016	1.0	United Kingdom	17647.0	Liverpool	6.0	Meat	16493.427	7
40	10.2016	1.0	United Kingdom	17647.0	Liverpool	13.0	Food	27539.397	11
41	10.2016	1.0	United Kingdom	17647.0	Liverpool	7.0	Clothing	8528.37	
42	10.2016	1.0	United Kingdom	17647.0	Liverpool	8.0	Household	1957.434	
43	10.2016	1.0	United Kingdom	17647.0	Liverpool	9.0	Hardware	2580.54	
44	10.2016	1.0	United Kingdom	17647.0	Liverpool	14.0	Non Food	13066.344	
45	10.2016	1.0	United Kingdom	17647.0	Liverpool	15.0	Admin	4947.084	
46	10.2016	1.0	United Kingdom	17647.0	Liverpool	12.0	Checkout	8965.803	6
47	10.2016	1.0	United Kingdom	17647.0	Liverpool	16.0	Customer Services	3584.433	

```

MonthYear  Time  Country  StoreID  City  Dept_ID  Dept.  HoursOwn  HoursL
index
In [34]:
48 10.2016  1.0  United  17647.0  Liverpool  11.0  Delivery  0
x=df1[['MonthYear']]
y=df1[['MonthYear']]
49 10.2016  1.0  United  17647.0  Liverpool  17.0  others  2624.598
   Kingdom

```

In [35]:

```
from sklearn.model_selection import train_test_split
```

In [36]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [37]:

```

from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)#ValueError: Input contains NaN, infinity or a value too large for

```

Out[37]:

```
LinearRegression()
```

In [38]:

```
print(lr.intercept_)
```

```
[0.]
```

In [39]:

```

coef= pd.DataFrame(lr.coef_)
coef

```

Out[39]:

```

0
0 1.0

```

In [40]:

```
print(lr.score(x_test,y_test))
```

```
1.0
```

In [41]:

```
prediction = lr.predict(x_test)
plt.scatter(y_test, prediction)
```

```

-----
-
TypeError                                Traceback (most recent call las
t)
<ipython-input-41-10d398fd7dc3> in <module>
      1 prediction = lr.predict(x_test)
----> 2 plt.scatter(y_test, prediction)

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\pyplot.py in scatter
(x, y, s, c, marker, cmap, norm, vmin, vmax, alpha, linewidths, verts, edg
ecolors, plotnonfinite, data, **kwargs)
    2888         verts=cbook.deprecation._deprecated_parameter,
    2889         edgcolors=None, *, plotnonfinite=False, data=None, **kwar
gs):
-> 2890         __ret = gca().scatter(
    2891             x, y, s=s, c=c, marker=marker, cmap=cmap, norm=norm,
    2892             vmin=vmin, vmax=vmax, alpha=alpha, linewidths=linewidths,

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\__init__.py in inner
(ax, data, *args, **kwargs)
    1445     def inner(ax, *args, data=None, **kwargs):
    1446         if data is None:
-> 1447             return func(ax, *map(sanitize_sequence, args), **kwarg
s)
    1448
    1449         bound = new_sig.bind(ax, *args, **kwargs)

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py
in wrapper(*inner_args, **inner_kwargs)
    409         else deprecation_addendum,
    410         **kwargs)
--> 411     return func(*inner_args, **inner_kwargs)
    412
    413     return wrapper

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py in sca
tter(self, x, y, s, c, marker, cmap, norm, vmin, vmax, alpha, linewidths,
verts, edgcolors, plotnonfinite, **kwargs)
    4430         # Process **kwargs to handle aliases, conflicts with expli
cit kwargs:
    4431
-> 4432         self._process_unit_info(xdata=x, ydata=y, kwargs=kwargs)
    4433         x = self.convert_xunits(x)
    4434         y = self.convert_yunits(y)

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axes\_base.py in _pr
ocess_unit_info(self, xdata, ydata, kwargs)
    2187         return kwargs
    2188
-> 2189         kwargs = _process_single_axis(xdata, self.xaxis, 'xunits',
kwargs)
    2190         kwargs = _process_single_axis(ydata, self.yaxis, 'yunits',
kwargs)
    2191         return kwargs

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axes\_base.py in _pr
ocess_single_axis(data, axis, unit_name, kwargs)
    2170         # We only need to update if there is nothing set y
et.
    2171         if not axis.have_units():
-> 2172             axis.update_units(data)

```

```

2173
2174         # Check for units in the kwargs, and if present update
axis

```

```

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\axis.py in update_un
its(self, data)
    1464         neednew = self.converter != converter
    1465         self.converter = converter
-> 1466         default = self.converter.default_units(data, self)
    1467         if default is not None and self.units is None:
    1468             self.set_units(default)

```

```

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\category.py in defau
lt_units(data, axis)
    105         # the conversion call stack is default_units -> axis_info
-> convert
    106         if axis.units is None:
--> 107             axis.set_units(UnitData(data))
    108         else:
    109             axis.units.update(data)

```

```

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\category.py in __ini
t__(self, data)
    174         self._counter = itertools.count()
    175         if data is not None:
--> 176             self.update(data)
    177
    178         @staticmethod

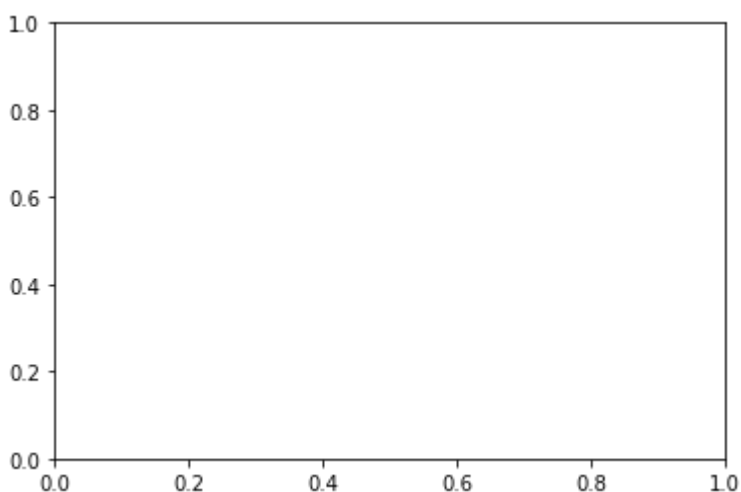
```

```

C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\category.py in updat
e(self, data)
    207         # check if convertible to number:
    208         convertible = True
--> 209         for val in OrderedDict.fromkeys(data):
    210             # OrderedDict just iterates over unique values in dat
a.
    211             cbook._check_isinstance((str, bytes), value=val)

```

**TypeError:** unhashable type: 'numpy.ndarray'



In [42]:

```
lr.score(x_test,y_test)
```

Out[42]:

1.0

In [43]:

```
lr.score(x_train,y_train)
```

Out[43]:

1.0

In [44]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [45]:

```
rr=Ridge(alpha=10)  
rr.fit(x_train,y_train)
```

Out[45]:

Ridge(alpha=10)

In [46]:

```
rr.score(x_test,y_test)
```

Out[46]:

0.75

In [47]:

```
la=Lasso(alpha=10)  
la.fit(x_train,y_train)
```

Out[47]:

Lasso(alpha=10)

In [48]:

```
la.score(x_test,y_test)
```

Out[48]:

0.75