In [24]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [3]:

```
df=pd.read_csv(r"C:\Users\hp\Downloads\bottle.csv.zip")
df
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_11820\1559421767.py:1: DtypeWarning: Columns (47,73) have mixed types. Spec
ify dtype option on import or set low_memory=False.
  df=pd.read_csv(r"C:\Users\hp\Downloads\bottle.csv.zip")
```

Out[3]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	...	R_PHAEO	R_PRES	R_SAMP	DIC1	DIC2
0	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	0	10.500	33.4400	NaN	25.64900	NaN	...	NaN	0	NaN	NaN	NaN
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	10.460	33.4400	NaN	25.65600	NaN	...	NaN	8	NaN	NaN	NaN
2	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	10	10.460	33.4370	NaN	25.65400	NaN	...	NaN	10	NaN	NaN	NaN
3	1	4	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	19	10.450	33.4200	NaN	25.64300	NaN	...	NaN	19	NaN	NaN	NaN
4	1	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0020A-7	20	10.450	33.4210	NaN	25.64300	NaN	...	NaN	20	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
864858	34404	864859	093.4 026.4	20-1611SR-MX-310-2239-09340264-0000A-7	0	18.744	33.4083	5.805	23.87055	108.74	...	0.18	0	NaN	NaN	NaN

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	...	R_PHAEO	R_PRES	R_SAMP	DIC1	DIC2
864859	34404	864860	093.4 026.4	20-1611SR-MX-310-2239-09340264-0002A-3	2	18.744	33.4083	5.805	23.87072	108.74	...	0.18	2	4.0	NaN	NaN
864860	34404	864861	093.4 026.4	20-1611SR-MX-310-2239-09340264-0005A-3	5	18.692	33.4150	5.796	23.88911	108.46	...	0.18	5	3.0	NaN	NaN
864861	34404	864862	093.4 026.4	20-1611SR-MX-310-2239-09340264-0010A-3	10	18.161	33.4062	5.816	24.01426	107.74	...	0.31	10	2.0	NaN	NaN
864862	34404	864863	093.4 026.4	20-1611SR-MX-310-2239-09340264-0015A-3	15	17.533	33.3880	5.774	24.15297	105.66	...	0.61	15	1.0	NaN	NaN

864863 rows × 74 columns

```
In [6]: df=df[['Salnty','T_degC']]
```

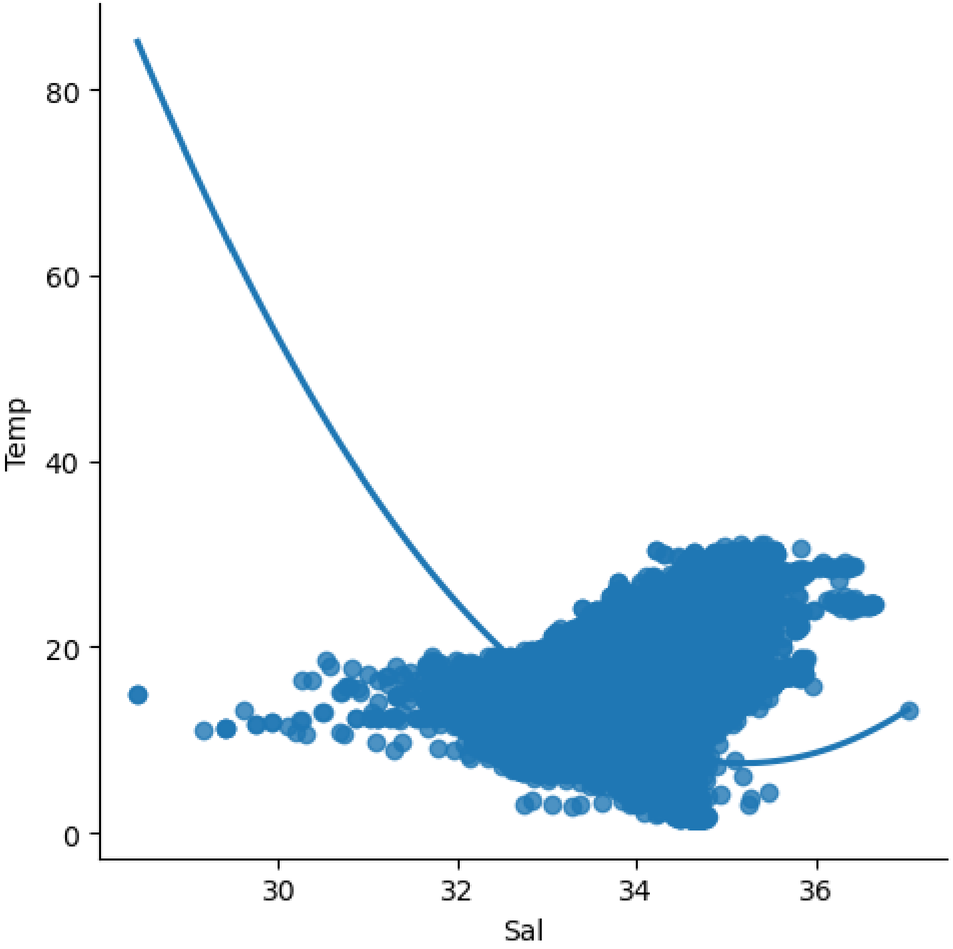
```
In [12]: df.columns=['Sal','Temp']
```

In [13]:
```
df.head(10)
```

Out[13]:

	Sal	Temp
0	33.440	10.50
1	33.440	10.46
2	33.437	10.46
3	33.420	10.45
4	33.421	10.45
5	33.431	10.45
6	33.440	10.45
7	33.424	10.24
8	33.420	10.06
9	33.494	9.86

```
In [14]: sns.lmplot(x="Sal",y="Temp",data=df,order=2,ci=None)

Out[14]: <seaborn.axisgrid.FacetGrid at 0x287c9759630>
```



In [15]:
```
df.describe()
```

Out[15]:

	Sal	Temp
count	817509.000000	853900.000000
mean	33.840350	10.799677
std	0.461843	4.243825
min	28.431000	1.440000
25%	33.488000	7.680000
50%	33.863000	10.060000
75%	34.196900	13.880000
max	37.034000	31.140000

In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 864863 entries, 0 to 864862
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Sal     817509 non-null  float64
 1   Temp    853900 non-null  float64
dtypes: float64(2)
memory usage: 13.2 MB
```

In [20]:
```
df.fillna(method='ffill',inplace=True)
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_11820\4116506308.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.fillna(method='ffill',inplace=True)
```

In [21]:

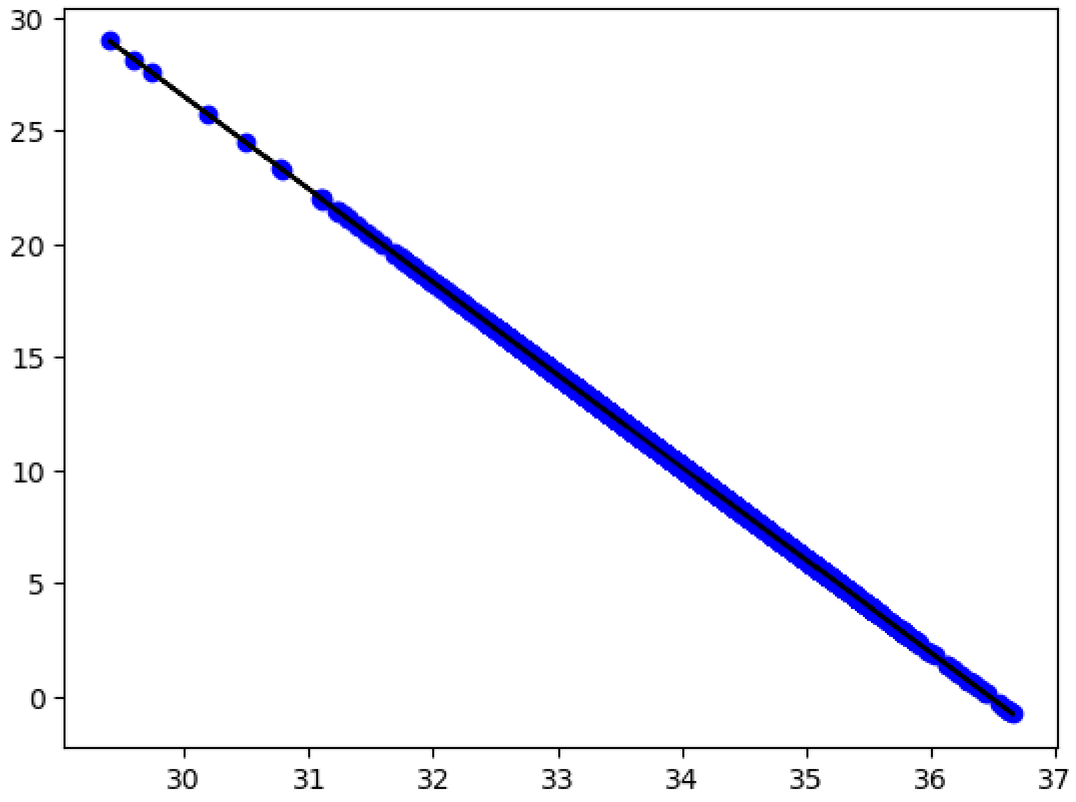
```
x=np.array(df['Sal']).reshape(-1,1)
y=np.array(df['Temp']).reshape(-1,1)
```

In [22]:

```
df.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

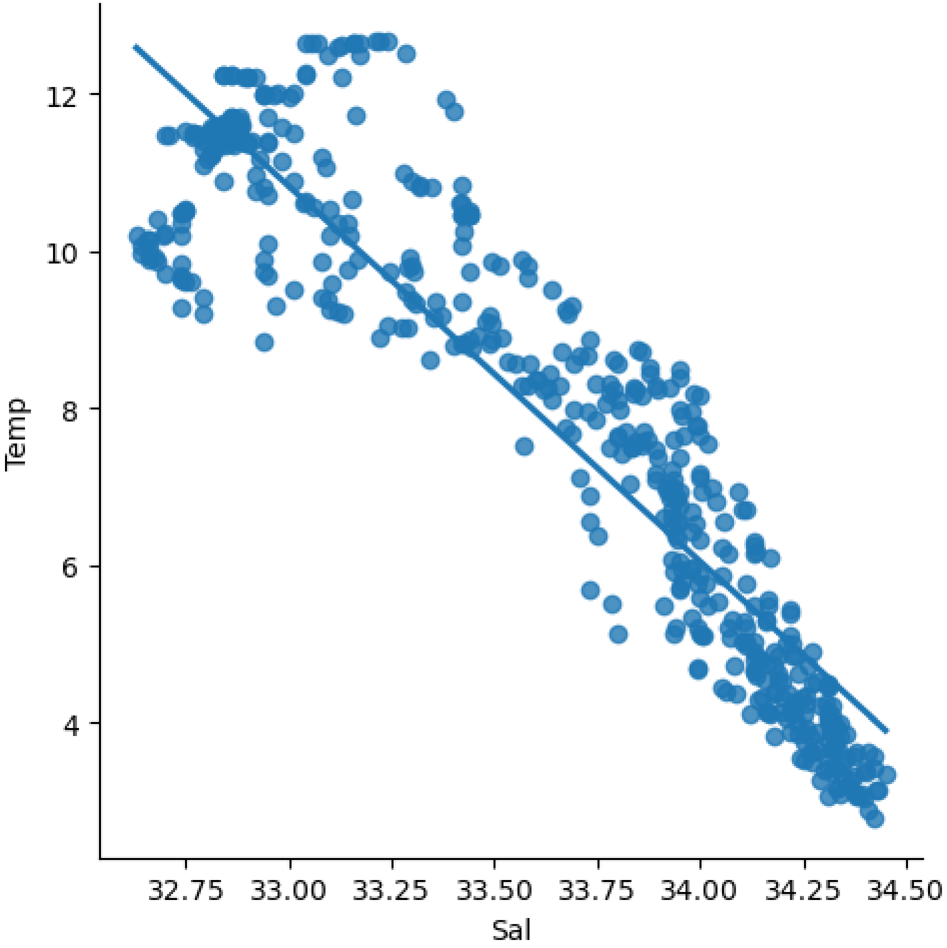
```
C:\Users\hp\AppData\Local\Temp\ipykernel_11820\693062840.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versu
s-a-copy)
  df.dropna(inplace=True)

0.20447179626481982
```

In [25]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_pred,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [27]:

```
df500=df[:][:500]
sns.lmplot(x="Sal",y="Temp",data=df500,order=1,ci=None)
x=np.array(df500['Sal']).reshape(-1,1)
y=np.array(df500['Temp']).reshape(-1,1)
df500.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```



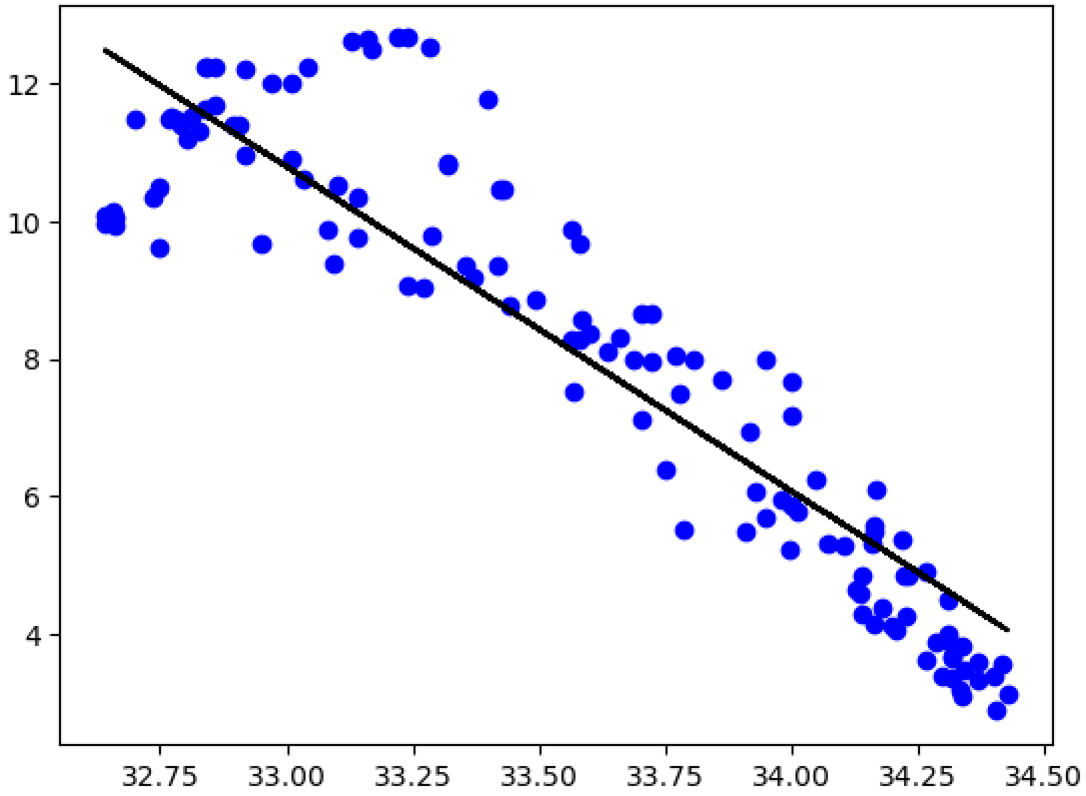
In [28]:

```
regr=LinearRegression()
regr.fit(x_train,y_train)
print("Regression:",regr.score(x_test,y_test))
```

```
Regression: 0.8528244252312662
```

In [30]:

```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [31]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 Score:",r2)
```

```
R2 Score: 0.8528244252312662
```

CONCLUSION:- Dataset we have taken is poor for Linear Model but Similar data can be used for this model

In [8]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```
In [4]: df=pd.read_csv(r"C:\Users\hp\Downloads\fiat500_VehicleSelection_Dataset.csv")
        df
```

```
Out[4]:
           ID  model  engine_power  age_in_days      km  previous_owners        lat        lon  price
      0     1  lounge           51          882   25000                1  44.907242   8.611560   8900
      1     2     pop           51         1186   32500                1  45.666359  12.241890   8800
      2     3   sport           74         4658  142228                1  45.503300  11.417840   4200
      3     4  lounge           51         2739  160000                1  40.633171  17.634609   6000
      4     5     pop           73         3074  106880                1  41.903221  12.495650   5700
    ...   ...     ...          ...          ...     ...              ...        ...        ...    ...
   1533  1534   sport           51         3712  115280                1  45.069679   7.704920   5200
   1534  1535  lounge           74         3835  112000                1  45.845692   8.666870   4600
   1535  1536     pop           51         2223   60457                1  45.481541   9.413480   7500
   1536  1537  lounge           51         2557   80750                1  45.000702   7.682270   5990
   1537  1538     pop           51         1766   54276                1  40.323410  17.568270   7900

1538 rows × 9 columns
```

```
In [5]: df=df[['age_in_days','km']]
        df.columns=['age','km']
```

In [6]:

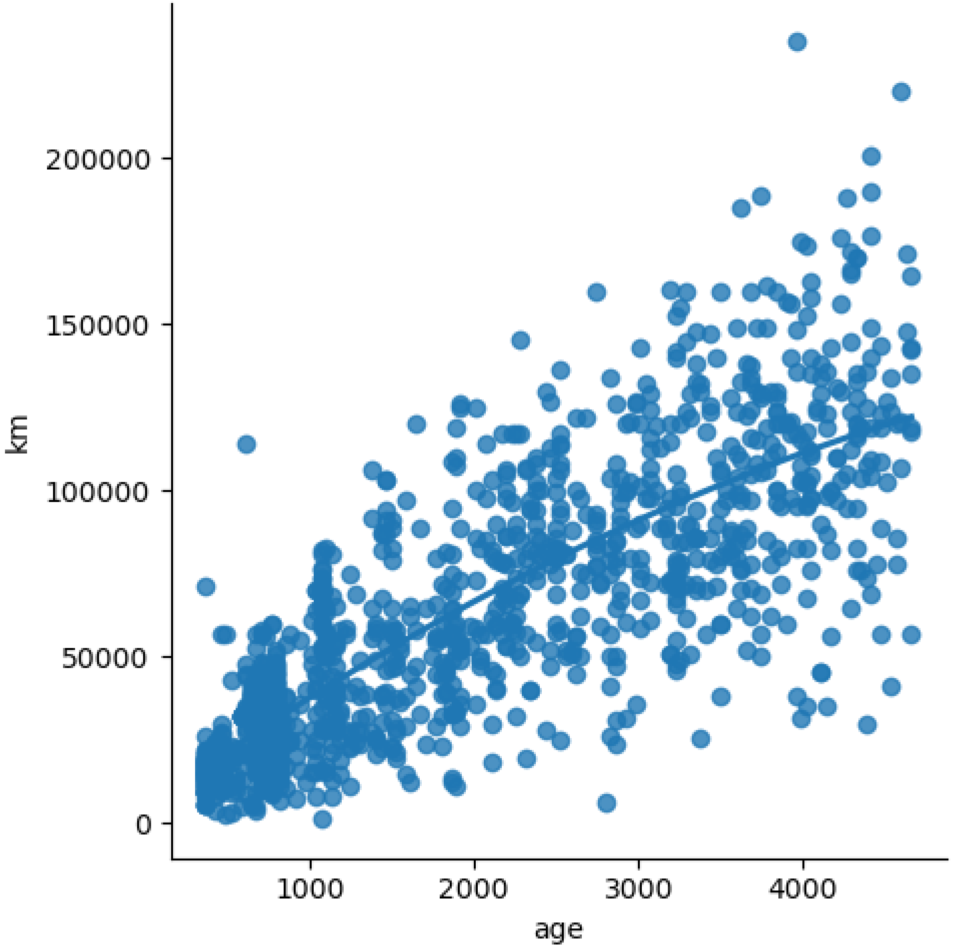
```
df.head(10)
```

Out[6]:

	age	km
0	882	25000
1	1186	32500
2	4658	142228
3	2739	160000
4	3074	106880
5	3623	70225
6	731	11600
7	1521	49076
8	4049	76000
9	3653	89000

In [9]:
```
sns.lmplot(x="age",y="km",data=df,order=2,ci=None)
```

Out[9]:
```
<seaborn.axisgrid.FacetGrid at 0x20f1c55f820>
```



In [10]:

```
df.describe()
```

Out[10]:

	age	km
count	1538.000000	1538.000000
mean	1650.980494	53396.011704
std	1289.522278	40046.830723
min	366.000000	1232.000000
25%	670.000000	20006.250000
50%	1035.000000	39031.000000
75%	2616.000000	79667.750000
max	4658.000000	235000.000000

In [11]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   age     1538 non-null   int64
 1   km      1538 non-null   int64
dtypes: int64(2)
memory usage: 24.2 KB
```

In [12]:

```
df.fillna(method='ffill',inplace=True)
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_14192\4116506308.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.fillna(method='ffill',inplace=True)
```

In [13]:

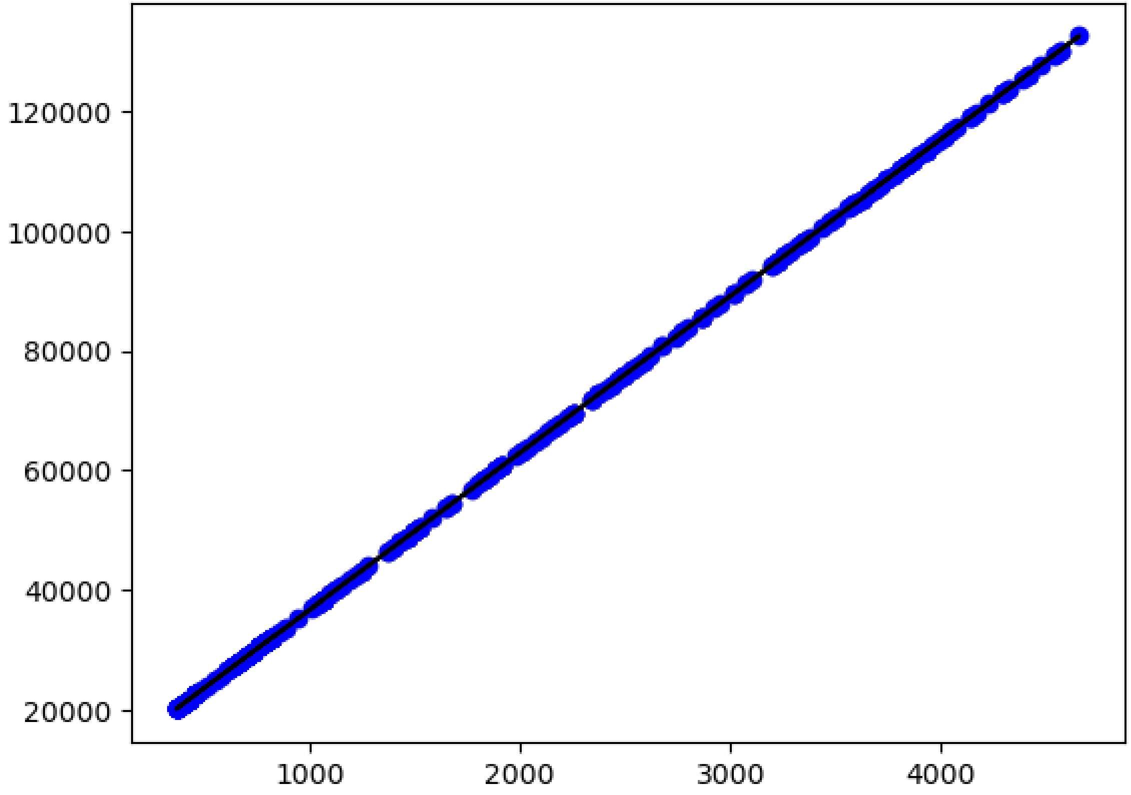
```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['km']).reshape(-1,1)
```

In [14]:

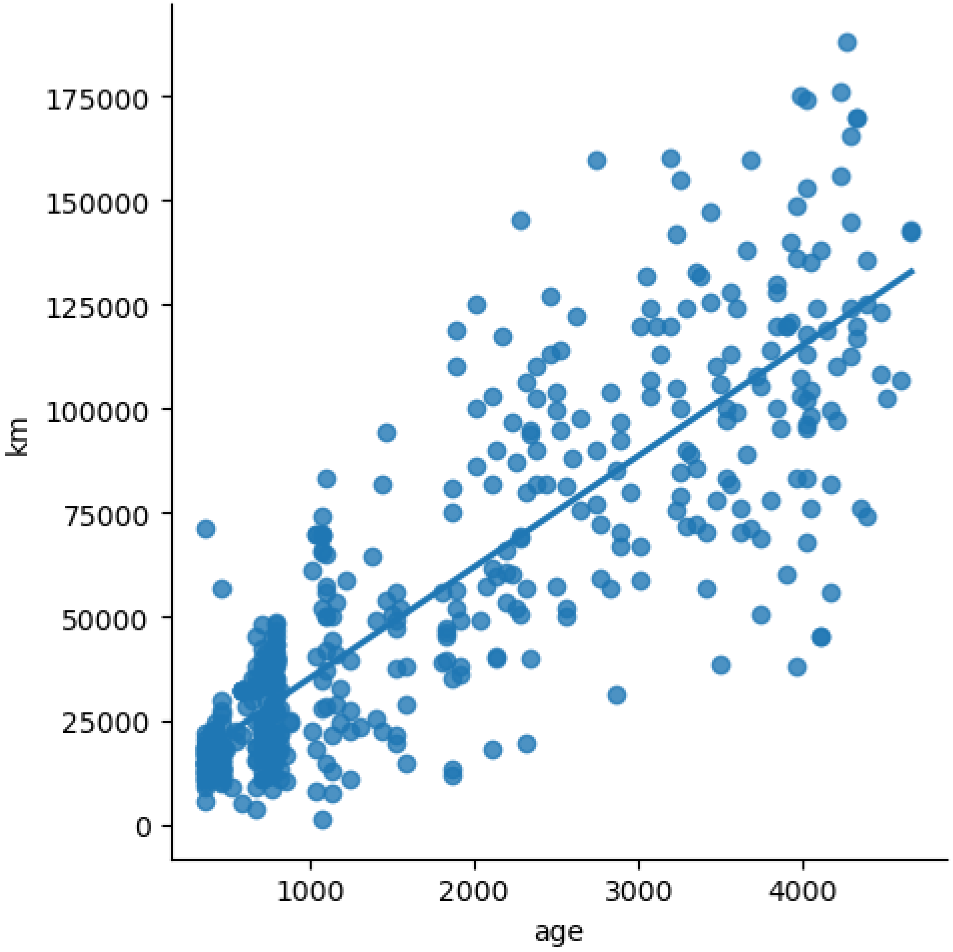
```
df.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.6676465296907084

C:\Users\hp\AppData\Local\Temp\ipykernel_14192\693062840.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versu
s-a-copy)
  df.dropna(inplace=True)
```

In [15]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_pred,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



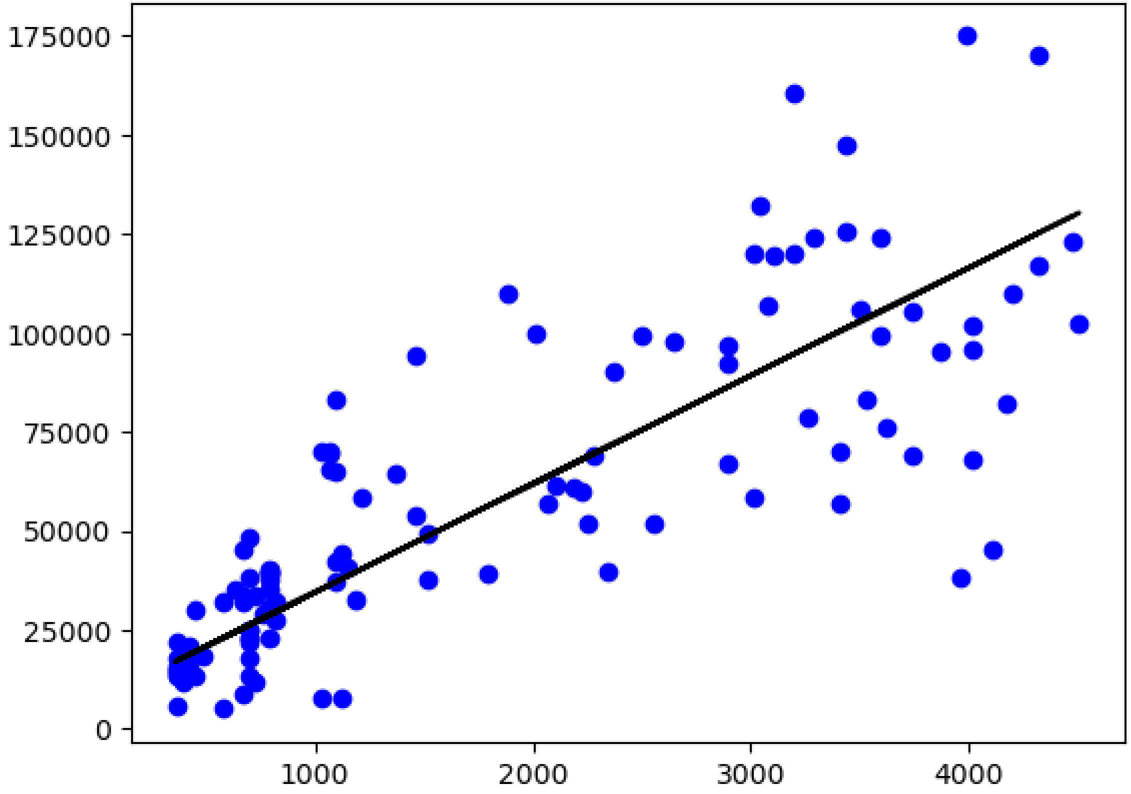
In [17]:
```
df500=df[:][:500]
sns.lmplot(x="age",y="km",data=df500,order=1,ci=None)
x=np.array(df500['age']).reshape(-1,1)
y=np.array(df500['km']).reshape(-1,1)
df500.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```



In [18]:
```
regr=LinearRegression()
regr.fit(x_train,y_train)
print("Regression:",regr.score(x_test,y_test))
```

```
Regression: 0.6673169580454327
```

In [19]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [23]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [24]:
```
df=pd.read_csv(r"C:\Users\hp\Downloads\data.csv")
df
```

Out[24]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	3	1340	0	1955
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	5	3370	280	1921
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	4	1930	0	1966
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000	1963
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	4	1140	800	1976
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	4	1510	0	1954
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	3	1460	0	1983
4597	2014-07-09 00:00:00	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0	3	3010	0	2009
4598	2014-07-10 00:00:00	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0	3	1070	1020	1974
4599	2014-07-10 00:00:00	2.206000e+05	3.0	2.50	1490	8102	2.0	0	0	4	1490	0	1990

4600 rows × 18 columns

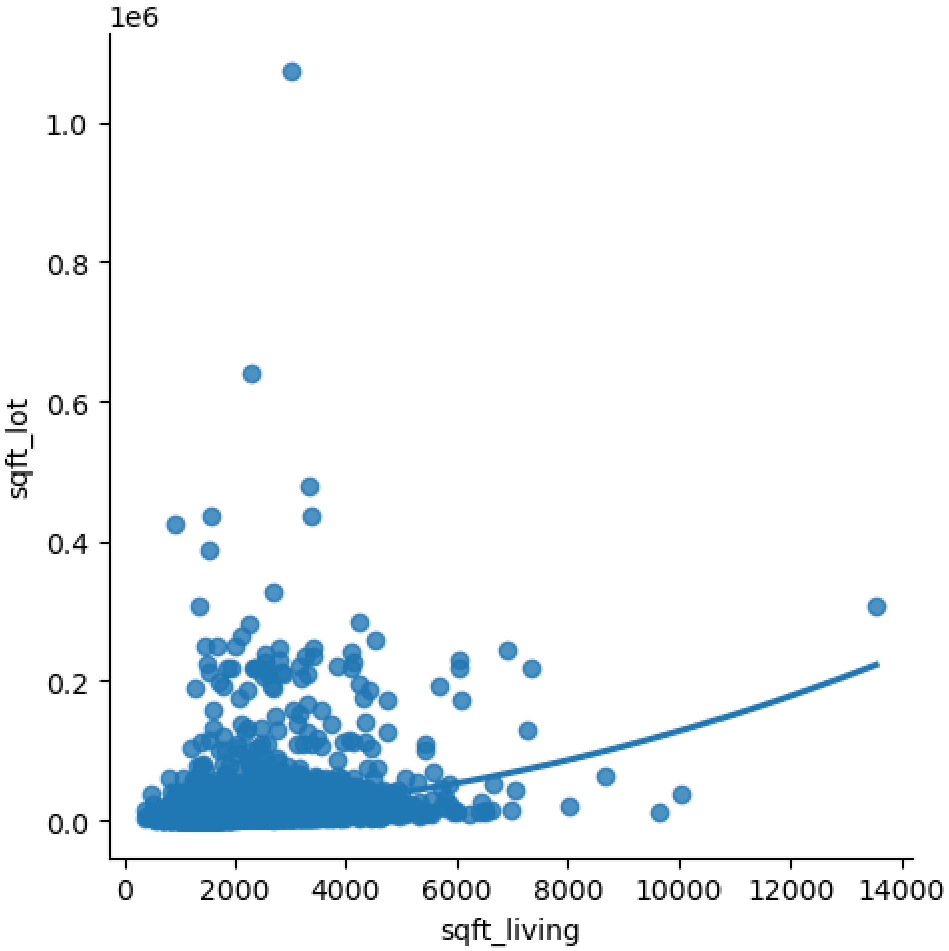
In [25]:

```
df=df[['sqft_living','sqft_lot']]
df.head(10)
```

Out[25]:

	sqft_living	sqft_lot
0	1340	7912
1	3650	9050
2	1930	11947
3	2000	8030
4	1940	10500
5	880	6380
6	1350	2560
7	2710	35868
8	2430	88426
9	1520	6200

```
In [26]: sns.lmplot(x='sqft_living',y='sqft_lot',data=df,order=2,ci=None)

Out[26]: <seaborn.axisgrid.FacetGrid at 0x1e0ec7051e0>
```



In [5]:
```
df.describe()
```

Out[5]:

	bedrooms	bathrooms
count	4600.000000	4600.000000
mean	3.400870	2.160815
std	0.908848	0.783781
min	0.000000	0.000000
25%	3.000000	1.750000
50%	3.000000	2.250000
75%	4.000000	2.500000
max	9.000000	8.000000

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   bedrooms   4600 non-null   float64
 1   bathrooms  4600 non-null   float64
dtypes: float64(2)
memory usage: 72.0 KB
```

In [7]:
```
df.fillna(method='ffill',inplace=True)
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_17392\4116506308.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.fillna(method='ffill',inplace=True)
```

In [27]:

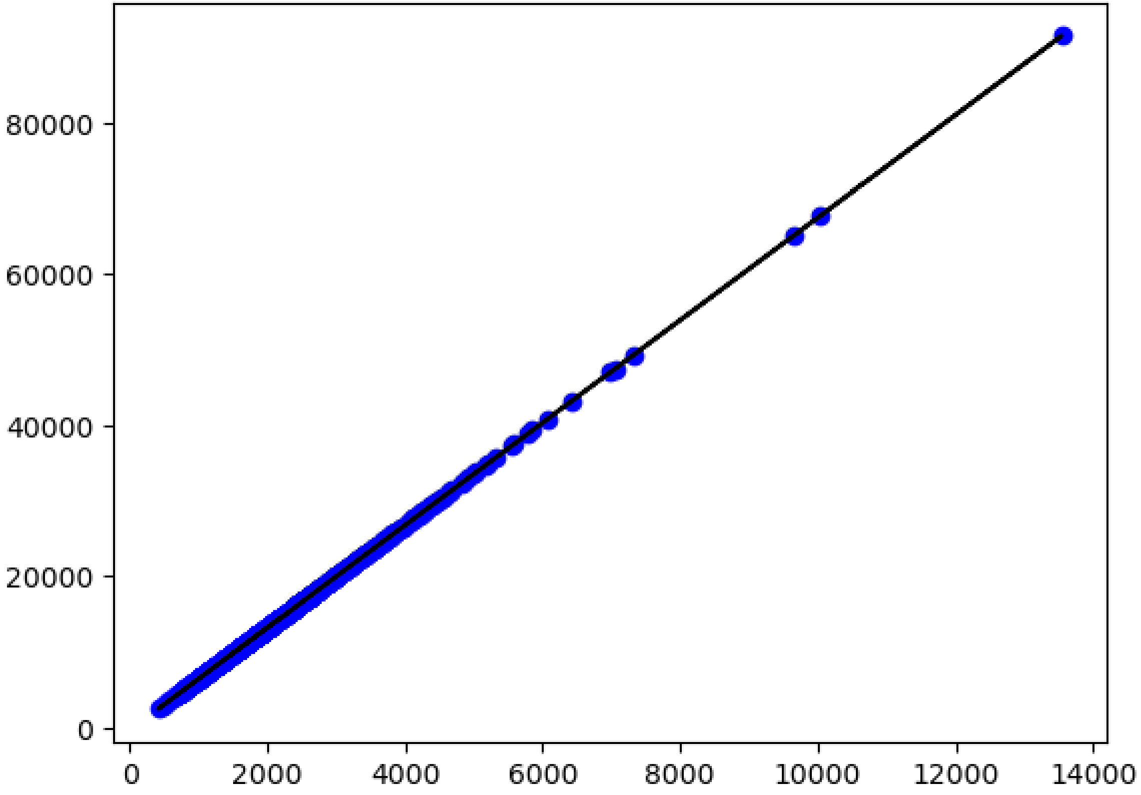
```
x=np.array(df['sqft_living']).reshape(-1,1)
y=np.array(df['sqft_lot']).reshape(-1,1)
```

In [28]:

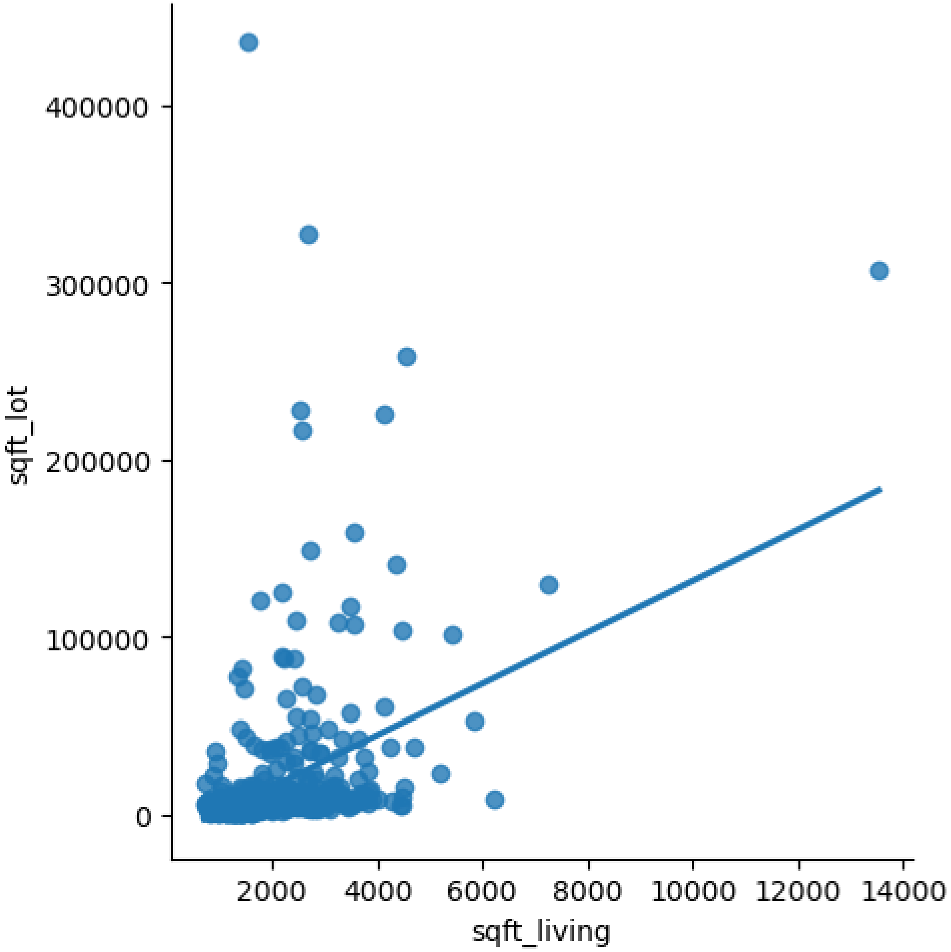
```
df.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.04892389884768433

C:\Users\hp\AppData\Local\Temp\ipykernel_17392\693062840.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versu
s-a-copy)
  df.dropna(inplace=True)
```

In [29]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_pred,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [30]:
```
df500=df[:][:500]
sns.lmplot(x="sqft_living",y="sqft_lot",data=df500,order=1,ci=None)
x=np.array(df500['sqft_living']).reshape(-1,1)
y=np.array(df500['sqft_lot']).reshape(-1,1)
df500.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```



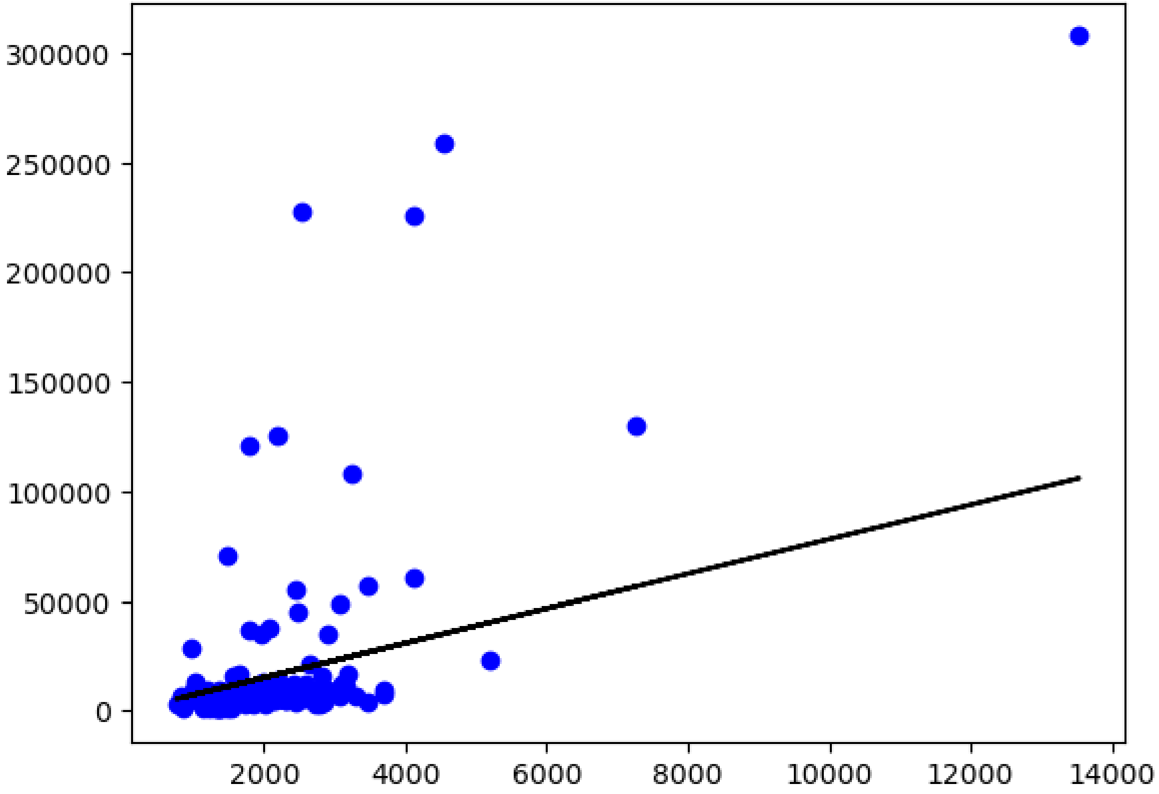
In [32]:

```
regr=LinearRegression()
regr.fit(x_train,y_train)
print("Regression:",regr.score(x_test,y_test))
```

```
Regression: 0.22602089337703057
```

In [33]:

```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```


300000
250000
200000
150000
100000
50000
0
2000
4000
6000
8000
10000
12000
14000