In [33]:
```
#Linear Regression model
#step1:problem statement-How Best Fit The Dataset?
```

In [34]:
```
#step 1:importing all the required libraries
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 [35]:
```
#step 2:reading the dataset
df=pd.read_csv(r"C:\Users\mouni\Downloads\ms.csv")
df
```

```
C:\Users\mouni\AppData\Local\Temp\ipykernel_2712\307571082.py:2: DtypeWarning: Columns (47,73) ha
ve mixed types. Specify dtype option on import or set low_memory=False.
  df=pd.read_csv(r"C:\Users\mouni\Downloads\ms.csv")
```

Out[35]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	...	R_PHAEO	R_PRE
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	
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-	8	10.460	33.4400	NaN	25.65600	NaN	...	NaN	

In [36]:
```
df=df[['Salnty', 'T_degC']]
df.columns=['Sal', 'Temp']
```

In [37]:

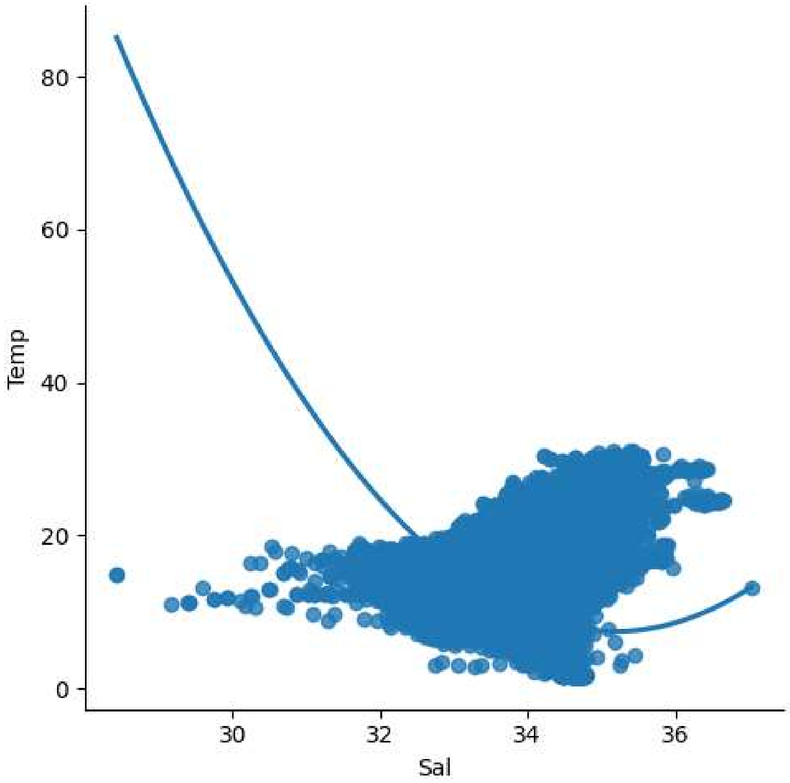
```
df.head(20)
```

Out[37]:

```
       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
10  33.510   9.83
11  33.580   9.67
12  33.640   9.50
13  33.689   9.32
14  33.847   8.76
15  33.860   8.71
16  33.876   8.53
17     NaN   8.45
18  33.926   8.26
19  33.980   7.96
```

In [38]:
```
#step-3:explaining the data scatter -plotting the data scatter
sns.lmplot(x='Sal',y='Temp',data=df,order=2,ci=None)
```

Out[38]:
```
<seaborn.axisgrid.FacetGrid at 0x1718a24e380>
```



In [39]:
```
df.describe()
```

Out[39]:

	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 [8]:
```
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 [40]:

```
#step-4:data cleaning-eliminating nan or missing input numbers
df.fillna(method='ffill')
```

Out[40]:

	Sal	Temp
0	33.4400	10.500
1	33.4400	10.460
2	33.4370	10.460
3	33.4200	10.450
4	33.4210	10.450
...	...	...
864858	33.4083	18.744
864859	33.4083	18.744
864860	33.4150	18.692
864861	33.4062	18.161
864862	33.3880	17.533

864863 rows × 2 columns

In [41]:

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

In [42]:

```
df.dropna(inplace=True)
```

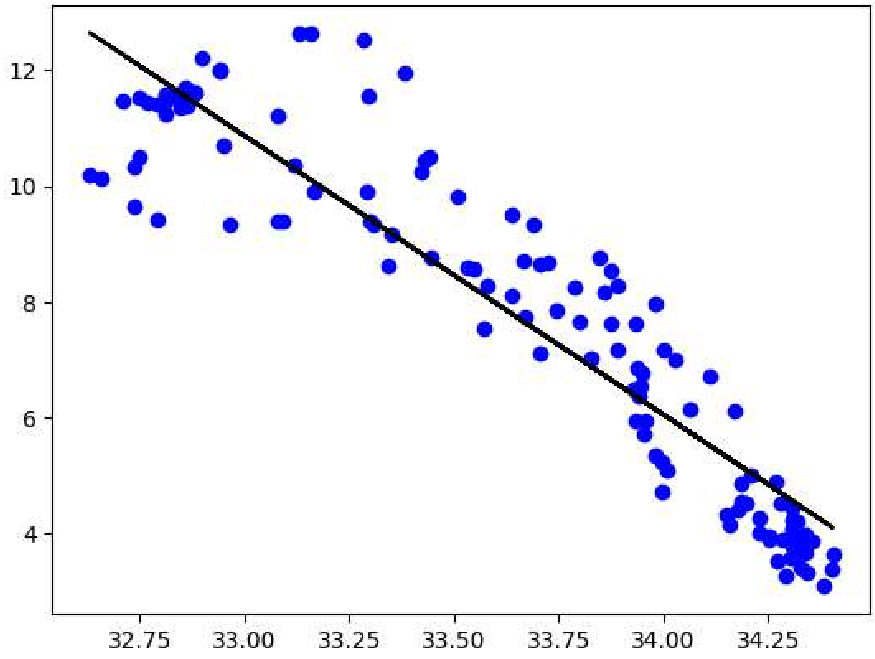
```
C:\Users\mouni\AppData\Local\Temp\ipykernel_2712\1379821321.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

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.dropna(inplace=True)
```

In [49]:

```
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))
```

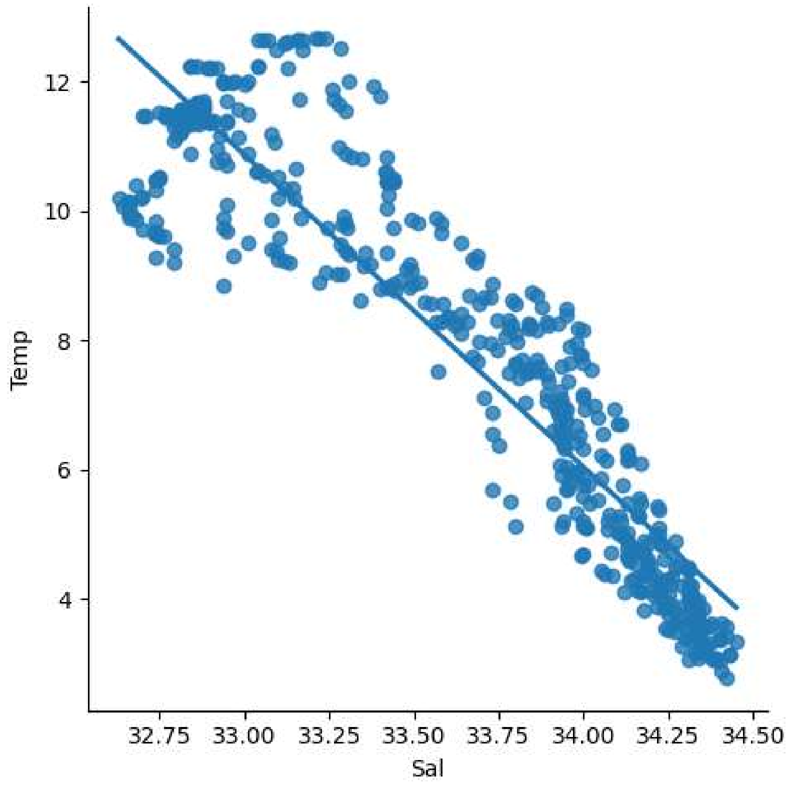
```
0.8660360853253846
```

In [50]:
```
#step-6:exploring our results
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



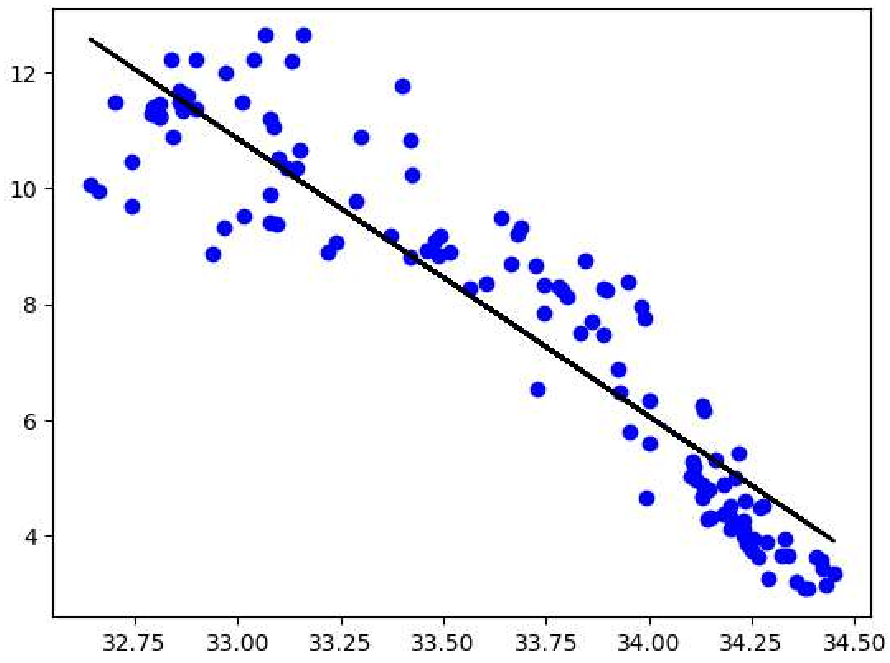
In [51]:
```
#step-7:working with a smallest dataset
df500=df[:][:500]
sns.lmplot(x="Sal",y="Temp",data=df500,order=1,ci=None)
```

Out[51]:
```
<seaborn.axisgrid.FacetGrid at 0x171891c7430>
```



```
In [52]: df500.fillna(method='ffill',inplace=True)
         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)
         regr=LinearRegression()
         regr.fit(x_train,y_train)
         print("Regression:",regr.score(x_test,y_test))
         y_pred=regr.predict(x_test)
         plt.scatter(x_test,y_test,color='b')
         plt.plot(x_test,y_pred,color='k')
         plt.show()
```

```
Regression: 0.8670081927554179
```



```
In [53]: #step-8:evaluation of model
         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: 0.8670081927554179
```

```
In [48]: #step-9:conclusion
         #dataset we have taken is poor for linear model but with the smaller data works well with linear mode
```

```
In [ ]:
```

```
In [54]: #Linear Regression model
         #step1:problem statement-How Best Fit The Dataset?
```

In [55]:

```
#step 1:importing all the required libraries
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 [79]:

```
#step 2:reading the dataset
dt=pd.read_csv(r"C:\Users\mouni\Downloads\fiat500_VehicleSelection_Dataset.csv")
dt
```

Out[79]:

	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 [80]:

```
dt=dt[['engine_power','age_in_days']]
dt.columns=['Eng','Age']
```

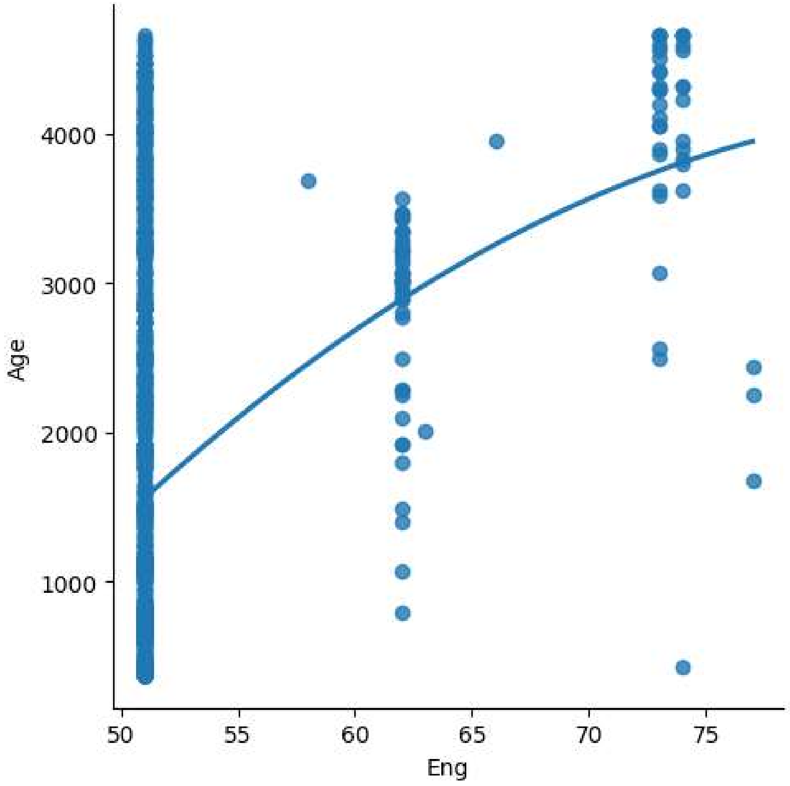
```
In [81]: dt.head(20)
```

Out[81]:

	Eng	Age
0	51	882
1	51	1186
2	74	4658
3	51	2739
4	73	3074
5	74	3623
6	51	731
7	51	1521
8	73	4049
9	51	3653
10	51	790
11	51	366
12	51	456
13	51	3835
14	51	1035
15	51	1096
16	73	4200
17	51	2223
18	51	2861
19	51	425

In [82]:
```
#step-3:explaining the data scatter -plotting the data scatter
sns.lmplot(x='Eng',y='Age',data=dt,order=2,ci=None)
```

Out[82]:
```
<seaborn.axisgrid.FacetGrid at 0x171896a4dc0>
```



In [83]:
```
dt.describe()
```

Out[83]:

	Eng	Age
count	1538.000000	1538.000000
mean	51.904421	1650.980494
std	3.988023	1289.522278
min	51.000000	366.000000
25%	51.000000	670.000000
50%	51.000000	1035.000000
75%	51.000000	2616.000000
max	77.000000	4658.000000

In [84]:
```
dt.info()
```

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

In [85]:

```
#step-4:data cleaning-eliminating nan or missing input numbers
dt.fillna(method='ffill')
```

Out[85]:

	Eng	Age
0	51	882
1	51	1186
2	74	4658
3	51	2739
4	73	3074
...	...	...
1533	51	3712
1534	74	3835
1535	51	2223
1536	51	2557
1537	51	1766

1538 rows × 2 columns

In [87]:

```
x=np.array(dt['Eng']).reshape(-1,1)
y=np.array(dt['Age']).reshape(-1,1)
```

In [88]:

```
dt.dropna(inplace=True)
```

```
C:\Users\mouni\AppData\Local\Temp\ipykernel_2712\735218168.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/indexi
ng.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/inde
xing.html#returning-a-view-versus-a-copy)
  dt.dropna(inplace=True)
```

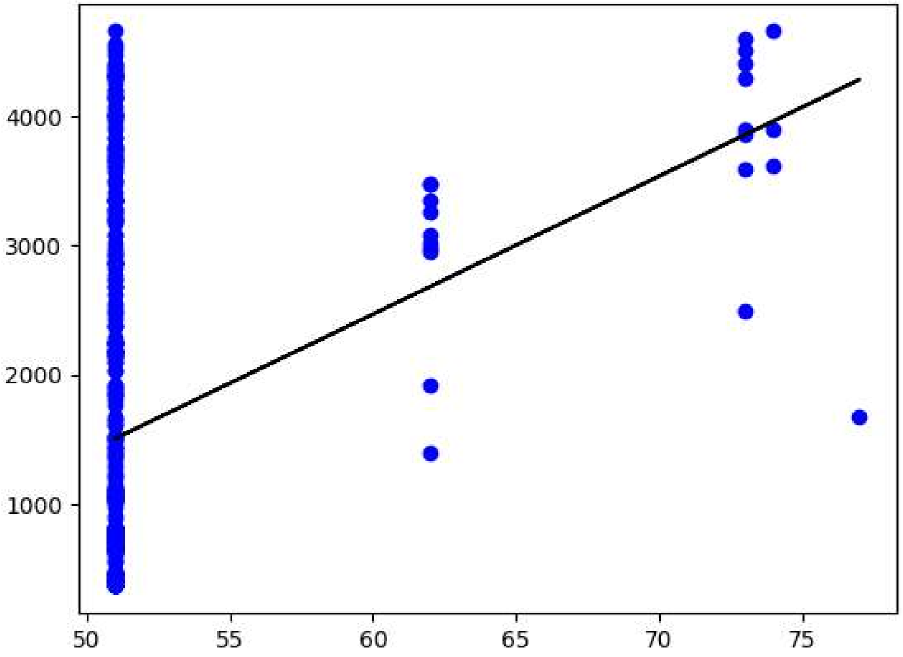
In [89]:

```
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))
```

```
0.06484025007545291
```

In [90]:

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

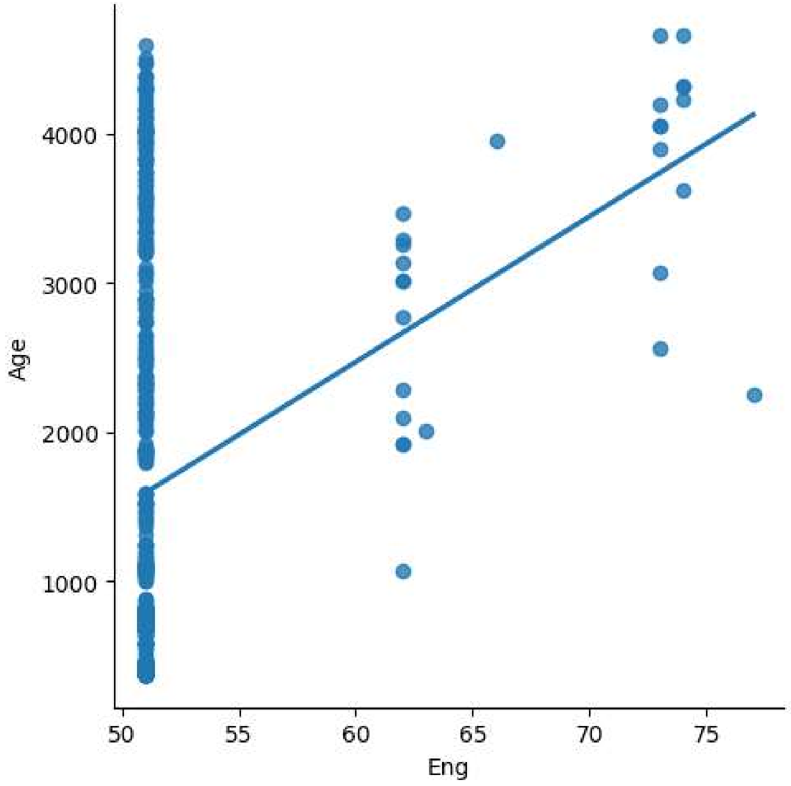


In [91]:

```
#step-7:working with a smallest dataset
dt500=dt[:][:500]
sns.lmplot(x="Eng",y="Age",data=dt500,order=1,ci=None)
```

Out[91]:

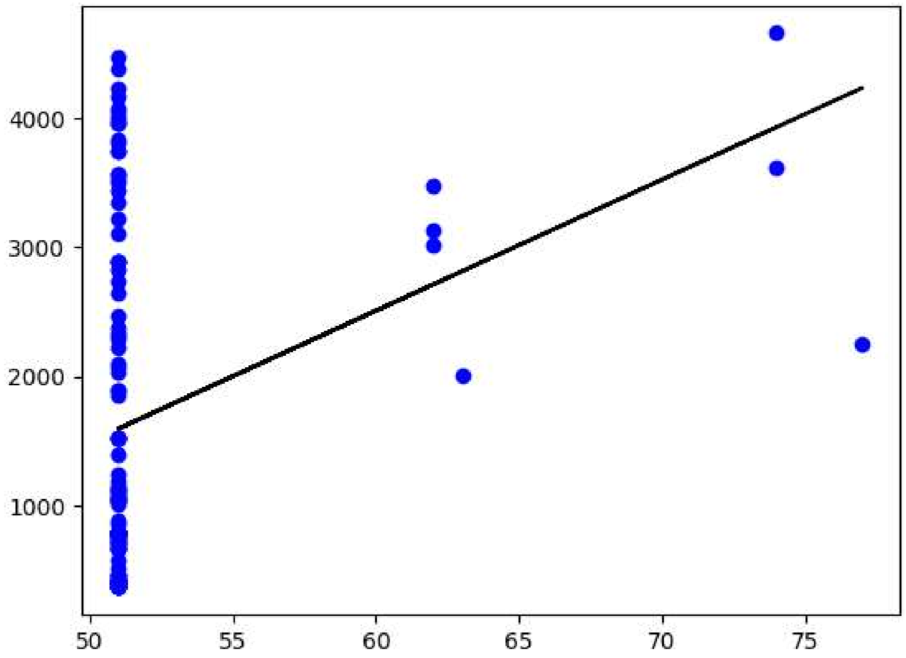
```
<seaborn.axisgrid.FacetGrid at 0x17189452470>
```



In [92]:

```
dt500.fillna(method='ffill',inplace=True)
x=np.array(dt500['Eng']).reshape(-1,1)
y=np.array(dt500['Age']).reshape(-1,1)
dt500.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("Regression:",regr.score(x_test,y_test))
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

```
Regression: 0.07405657148022737
```



In [93]:

```
#step-8:evaluation of model
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: 0.07405657148022737
```

In [94]:

```
#step-9:conclusion
#dataset we have taken is poor for linear model but with the smaller data works well with linear mode
```

In [ ]: