

In [1]:

```
#step1
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 [2]:

*#step2*

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

C:\Users\shaik\AppData\Local\Temp\ipykernel\_20820\84675588.py:2: DtypeWarning: Columns (47,73) have mixed types. Specify dtype option on import or set low\_memory=False.

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

Out[2]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta
0	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	0	10.500	33.4400	NaN	25.64900
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	10.460	33.4400	NaN	25.65600

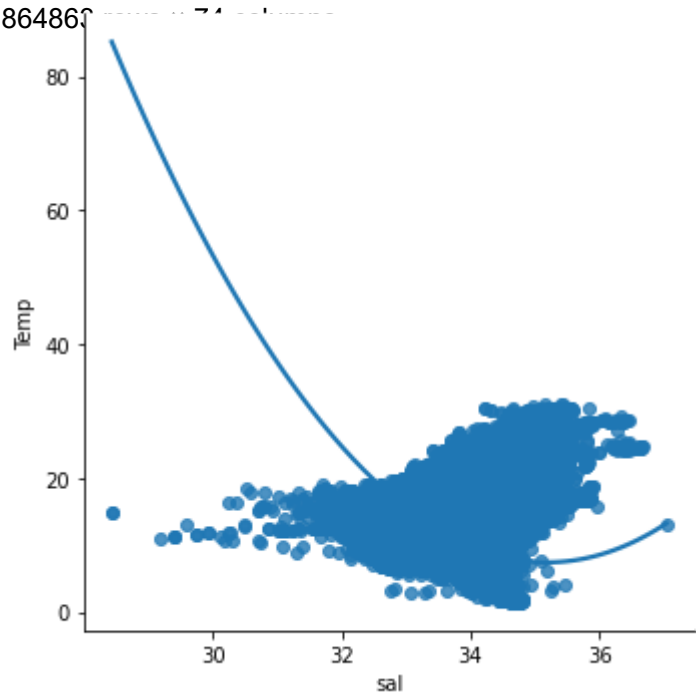
In [3]:

```
#taking two columns from dataset
k=k[['Salnty','T_degC']]
#re naming the attributes
k.columns=['sal','Temp']
k.head(10)
```

Out[3]:										
	sal	Temp								
0	33.440	10.50								
1	33.440	10.46	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	20	10.450	33.4210	NaN	25.64300
2	33.437	10.46								
3	33.420	10.45	...	...	...	...	...	...	...	...
4	33.421	10.45								
5	33.431	10.45								
864858	34404	10.45	864859	093.4 026.4	20-1611SR-MX-310-2239-09340264-0000A-7	0	18.744	33.4083	5.805	23.87055
6	33.440	10.45								
7	33.424	10.24								
8	33.420	10.06								
864859	34404	10.45	864860	093.4 026.4	20-1611SR-MX-310-2239-09340264-0002A-3	2	18.744	33.4083	5.805	23.87072
864860	34404	10.45	864861	093.4 026.4	20-1611SR-MX-310-2239-09340264-0005A-3	5	18.692	33.4150	5.796	23.88911
864861	34404	10.45	864862	093.4 026.4	20-1611SR-MX-310-2239-09340264-0010A-3	10	18.161	33.4062	5.816	24.01426

```
In [4]: Cst_Cnt  Btl_Cnt  Sta_ID  Depth_ID  Depthm  T_degC  Salnty  O2ml_L  STheta
```

```
#step3
#exploring the data scatter-plotting the data
sns.lmplot(x="sal",y="Temp",data=k,order=2,ci=None)
864862 34404 864863 093.4 MX-310- 15 17.533 33.3880 5.774 24.15297
Out[4]:
026.4 2239-
09340264-
0015A-3
<seaborn.axisgrid.FacetGrid at 0x24b656fc880>
```



```
In [5]:
```

```
k.describe()
```

```
Out[5]:
```

	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 [6]:

k.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 [7]:

```
#step4
#data cleaning-eliminating
k.fillna(method='ffill',inplace=True)
k
```

C:\Users\shaik\AppData\Local\Temp\ipykernel\_20820\463270389.py:3: 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) (https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy)

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

Out[7]:

	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 [8]:

```
#step5
#traing our model
x=np.array(k['sal']).reshape(-1,1)
y=np.array(k['Temp']).reshape(-1,1)
```

In [9]:

```
k.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```

C:\Users\shaik\AppData\Local\Temp\ipykernel\_20820\53174099.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) ([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))  
k.dropna(inplace=True)

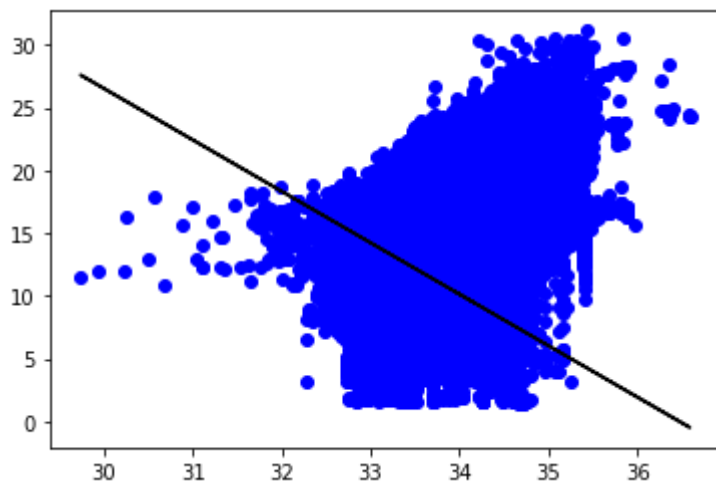
In [13]:

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

0.20341540626628107

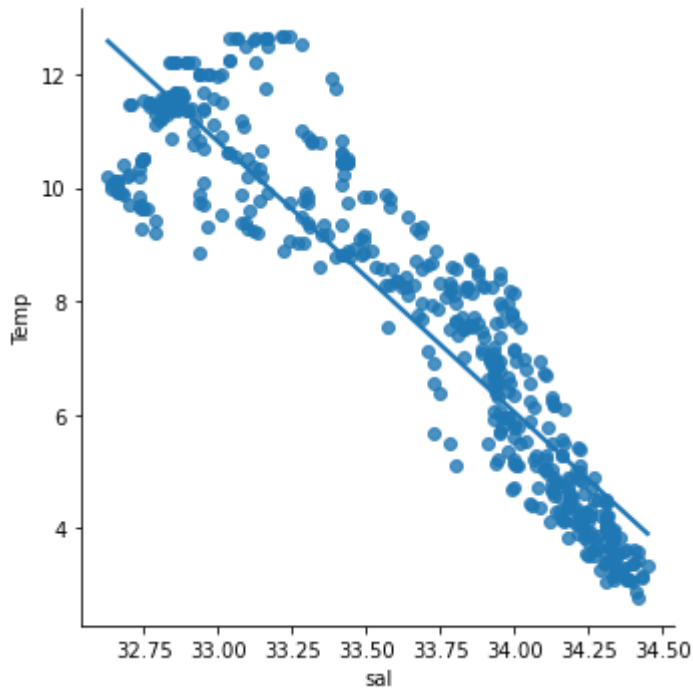
In [16]:

```
#step6
#exploring our results
#data scatter of predicted values
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 [18]:

```
#selectin first 500 rows
k500=k[:][:500]
sns.lmplot(x="sal",y="Temp",data=k500,order=1,ci=None)
k500.fillna(method='ffill',inplace=True)
x=np.array(k500['sal']).reshape(-1,1)
y=np.array(k500['Temp']).reshape(-1,1)
k500.dropna(inplace=True)
```



In [19]:

```
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))
```

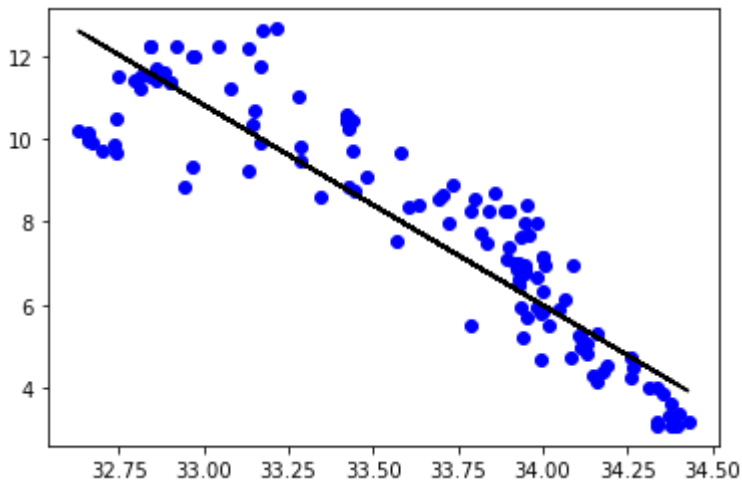
regression: 0.805140331004344

In [20]:

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

Out[20]:

<function matplotlib.pyplot.show(close=None, block=None)>



In [30]:

```
#step8
#evaluation of model
#train the model
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
#evaluate the model on test data
model =LinearRegression()
model.fit(x_train,y_train)
```

Out[30]:

LinearRegression()

In [31]:

```
#step9:
'''dataset we have taken is poor for linear model but with smaller data works well'''
```

Out[31]:

'dataset we have taken is poor for linear model but with smaller data work  
s well'

In [ ]:



