In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random
```

In [ ]:
```
bottle = pd.read_csv('bottle.csv')
```

```
/Users/adisrikanth/opt/anaconda3/lib/python3.8/site-packages/IPython/core/interactiveshell.py:3165: DtypeWarning: Columns (47,73) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [ ]:
```
bottle.head()
```

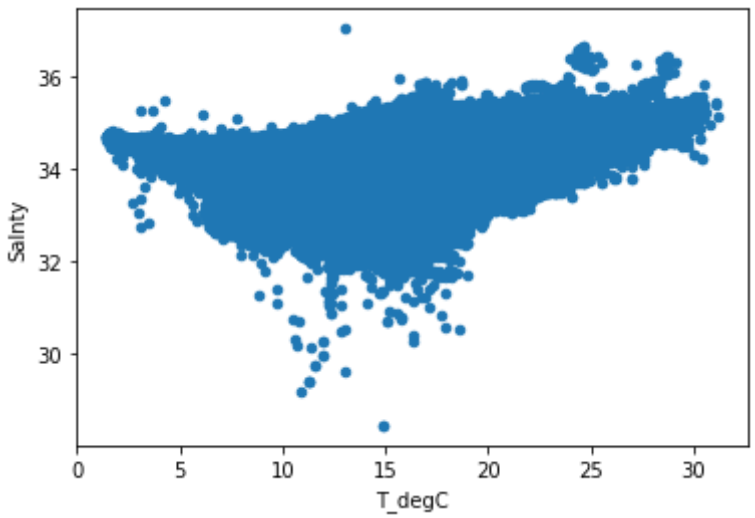
Out[ ]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2Sat	...
0	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	0	10.50	33.440	NaN	25.649	NaN	...
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	10.46	33.440	NaN	25.656	NaN	...
2	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	10	10.46	33.437	NaN	25.654	NaN	...
3	1	4	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	19	10.45	33.420	NaN	25.643	NaN	...
4	1	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0020A-7	20	10.45	33.421	NaN	25.643	NaN	...

5 rows × 74 columns

In [ ]:
```
bottle.plot.scatter(x='T_degC', y='Salnty')
```

Out[ ]: <AxesSubplot:xlabel='T_degC', ylabel='Salnty'>



In [ ]:
```
salnty = np.array(bottle['Salnty'])
temp = np.array(bottle['T_degC'])
```

Question (a)

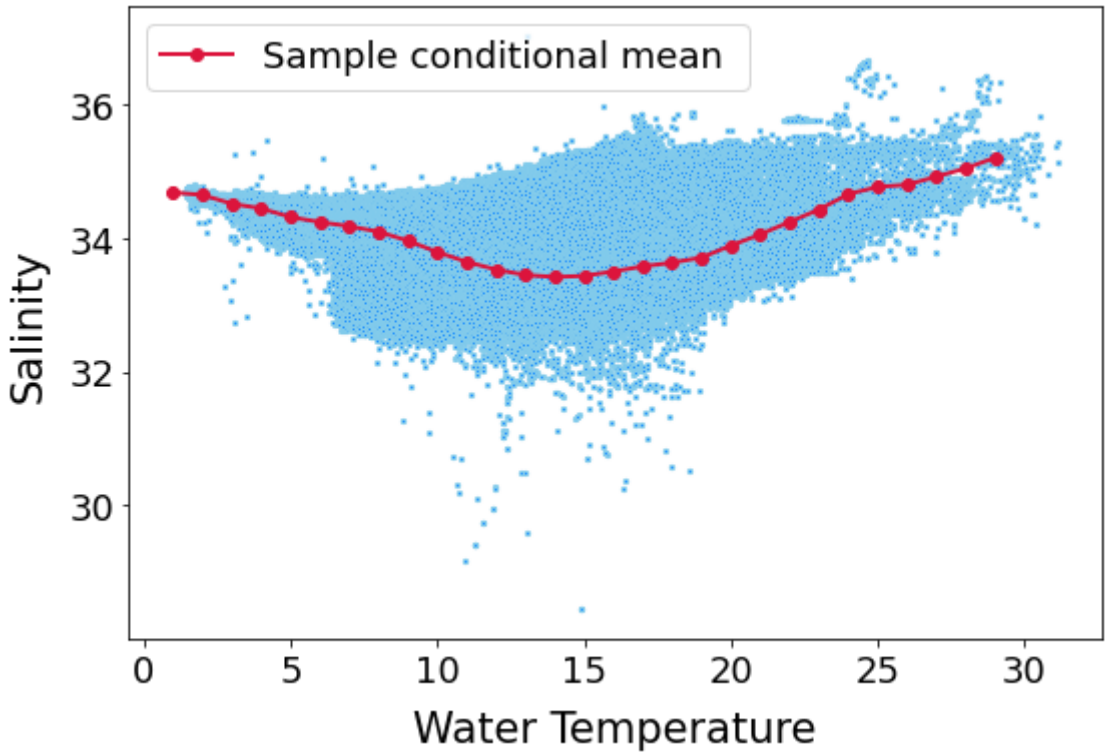
Estimate conditional mean of sanlinity given temperature, and plot the conditional mean together with the scatter plot.

Note: Both salinity and temperature can be modeled with continous random variables. To simplify the conditional expectation, we discretize tempertures by bins with equal width. We then aggregate the salinity data within each bin.

In [ ]:
```
max_val = np.max(temp)
width_bin = 2
fig = plt.figure(figsize = (9,6))
plt.scatter(temp,salnty, s=5, c="dodgerblue", marker='o', edgecolor="skyblue")

# TODO: create bins from 0 to the maximum to discretize continous temperatures
grid = np.arange(0,int(max(bottle.T_degC)))

# TODO: Compute the conditional expection of sanlinity given temperture
cond_average_salnty = np.zeros(len(grid))

bottle['discrete_temp'] = bottle['T_degC'].round()
bottle_grouped = bottle.groupby('discrete_temp')['Salnty'].agg('mean')

for i in range(1, len(grid)):
    cond_average_salnty[i] = bottle_grouped[i]


plt.plot(grid[1:-1],cond_average_salnty[1:-1],'-o',lw=2,color='crimson', label="
plt.ylabel("Salinity", fontsize=21,labelpad=10)
plt.xlabel("Water Temperature", fontsize=21,labelpad=10)

plt.legend(fontsize=18)
# plt.xlim(-5,30)
# plt.ylim(-12,23)
```

```
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.gcf().subplots_adjust(bottom=0.15)
plt.gcf().subplots_adjust(left=0.15)
plt.savefig('conditional_expectation.pdf')
```

```
<ipython-input-36-19b036eb0a32>:30: UserWarning: Creating legend with loc="best"
can be slow with large amounts of data.
  plt.savefig('conditional_expectation.pdf')
<ipython-input-36-19b036eb0a32>:30: UserWarning: Creating legend with loc="best"
can be slow with large amounts of data.
  plt.savefig('conditional_expectation.pdf')
/Users/adisrikanth/opt/anaconda3/lib/python3.8/site-packages/IPython/core/pylabt
ools.py:132: UserWarning: Creating legend with loc="best" can be slow with large
amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```



Justification

We choose this bin size as it generates a discrete value for each integer temperature value that can be taken. This system is generalizable and also is easy to interpret for someone viewing the data. Also, given that we have 800,000+ rows in our dataframe, there should be adequate coverage for 30 discrete values.

Question (b)

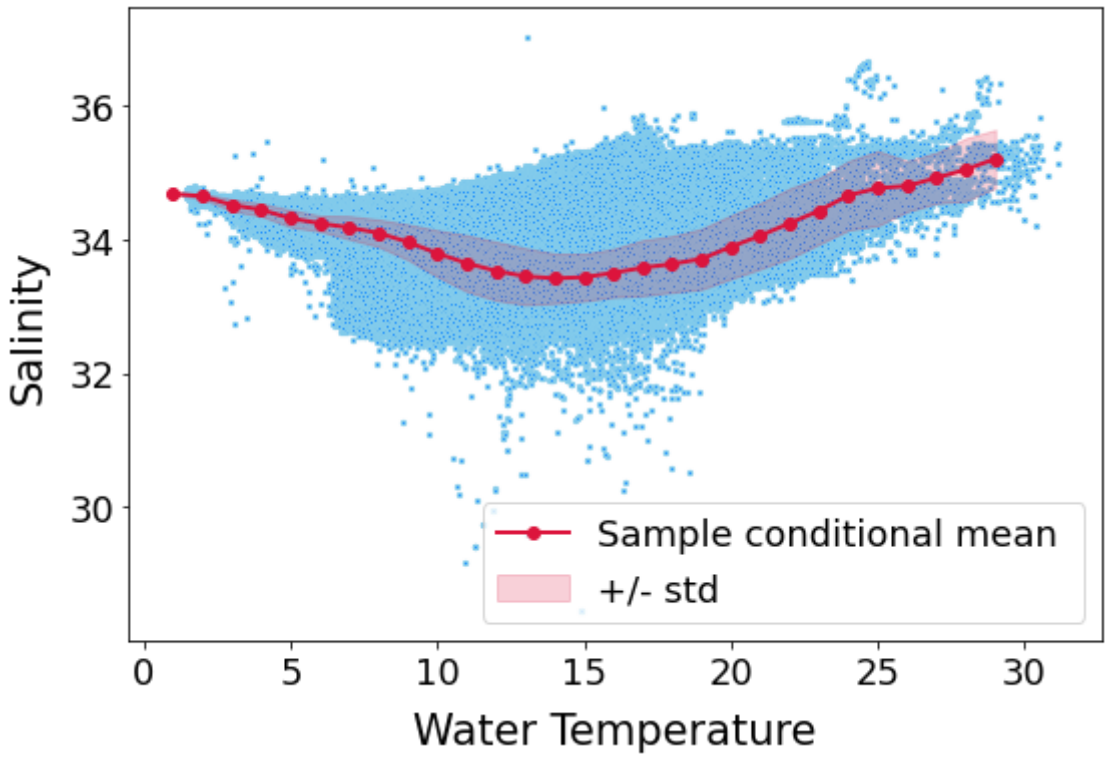
Estimate conditional standard deviation, and plot the confidence interval within +/- one conditional standard deviation.

In [ ]:

```
# TODO: Compute the conditional standard deviation of sanlinity given temperture
cond_std_salnty = np.zeros(len(grid))

bottle_grouped_std = bottle.groupby('discrete_temp')['Salnty'].agg('std')
```

```
for i in range(1, len(grid)):
    cond_std_salnty[i] = bottle_grouped_std[i]

fig = plt.figure(figsize = (9,6))
plt.scatter(temp,salnty, s=5, c="dodgerblue", marker='o', edgecolor="skyblue")


plt.plot(grid[1:-1],cond_average_salnty[1:-1],'-o',lw=2,color='crimson', label="
plt.fill_between(grid[1:-1], cond_average_salnty[1:-1]-cond_std_salnty[1:-1],
                 cond_average_salnty[1:-1]+cond_std_salnty[1:-1], color='crimson'

plt.ylabel("Salinity", fontsize=21,labelpad=10)
plt.xlabel("Water Temperature", fontsize=21,labelpad=10)

plt.legend(fontsize=18)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.gcf().subplots_adjust(bottom=0.15)
plt.gcf().subplots_adjust(left=0.15)
plt.savefig('conditional_expectation_w_std.pdf')
```

```
<ipython-input-37-fa8350877ab8>:25: UserWarning: Creating legend with loc="best"
can be slow with large amounts of data.
  plt.savefig('conditional_expectation_w_std.pdf')
<ipython-input-37-fa8350877ab8>:25: UserWarning: Creating legend with loc="best"
can be slow with large amounts of data.
  plt.savefig('conditional_expectation_w_std.pdf')
/Users/adisrikanth/opt/anaconda3/lib/python3.8/site-packages/IPython/core/pylabt
ools.py:132: UserWarning: Creating legend with loc="best" can be slow with large
amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```



Part C

We do not necessarily expect all of our estimates to equally reliable at all points. The more data we have to make an estimate, the more reliable it tends to be. As

a result, if different temperatures have varying amounts of data that we use to make a conditional estimate, the reliability of these estimates will also vary from case to case.