In [55]:

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

In [56]:

```
df=pd.read_csv("https://raw.githubusercontent.com/ektanegi25/Cement-strength-predic
```

In [57]:

```
df.head()
```

Out[57]:

	Cement (component 1)(kg in a m^3 mixture)	Blast Furnace Slag (component 2)(kg in a m^3 mixture)	Fly Ash (component 3)(kg in a m^3 mixture)	Water (component 4)(kg in a m^3 mixture)	Superplasticizer (component 5)(kg in a m^3 mixture)	Coarse Aggregate (component 6)(kg in a m^3 mixture)	Fine Aggregate (componen 7)(kg in a m^3 mixture
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5

In [58]:

```
df.tail()
```

Out[58]:

	Cement (component 1)(kg in a m^3 mixture)	Blast Furnace Slag (component 2)(kg in a m^3 mixture)	Fly Ash (component 3)(kg in a m^3 mixture)	Water (component 4)(kg in a m^3 mixture)	Superplasticizer (component 5)(kg in a m^3 mixture)	Coarse Aggregate (component 6)(kg in a m^3 mixture)	Aggre (compon 7)(kg mixt
1025	276.4	116.0	90.3	179.6	8.9	870.1	7
1026	322.2	0.0	115.6	196.0	10.4	817.9	8
1027	148.5	139.4	108.6	192.7	6.1	892.4	7
1028	159.1	186.7	0.0	175.6	11.3	989.6	7
1029	260.9	100.5	78.3	200.6	8.6	864.5	7

In [59]:

```
name_Col=df.columns.tolist()
```

In [60]:

```
name_Col
```

```
Out[60]: ['Cement (component 1)(kg in a m^3 mixture)',
          'Blast Furnace Slag (component 2)(kg in a m^3 mixture)',
          'Fly Ash (component 3)(kg in a m^3 mixture)',
          'Water  (component 4)(kg in a m^3 mixture)',
          'Superplasticizer (component 5)(kg in a m^3 mixture)',
          'Coarse Aggregate  (component 6)(kg in a m^3 mixture)',
          'Fine Aggregate (component 7)(kg in a m^3 mixture)',
          'Age (day)',
          'Concrete compressive strength(MPa, megapascals) ']
```

```
In [61]: name_Col[-1].split("(")
```

```
Out[61]: ['Concrete compressive strength', 'MPa, megapascals) ']
```

```
In [62]: name_Col=[i.split('(')[0]for i in name_Col]
```

```
In [63]: name_Col
```

```
Out[63]: ['Cement ',
          'Blast Furnace Slag ',
          'Fly Ash ',
          'Water  ',
          'Superplasticizer ',
          'Coarse Aggregate  ',
          'Fine Aggregate ',
          'Age ',
          'Concrete compressive strength']
```

```
In [64]: df.columns=name_Col
```

```
In [65]: df.columns
```

```
Out[65]: Index(['Cement ', 'Blast Furnace Slag ', 'Fly Ash ', 'Water  ',
                'Superplasticizer ', 'Coarse Aggregate  ', 'Fine Aggregate ', 'Age ',
                'Concrete compressive strength'],
               dtype='object')
```

```
In [66]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Cement                         1030 non-null   float64
 1   Blast Furnace Slag             1030 non-null   float64
 2   Fly Ash                        1030 non-null   float64
 3   Water                          1030 non-null   float64
 4   Superplasticizer               1030 non-null   float64
 5   Coarse Aggregate               1030 non-null   float64
 6   Fine Aggregate                 1030 non-null   float64
 7   Age                            1030 non-null   int64  
 8   Concrete compressive strength  1030 non-null   float64
dtypes: float64(8), int64(1)
memory usage: 72.5 KB
```

In [67]:
```
df.describe()
```

Out[67]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	F Aggreg
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.0000
mean	281.165631	73.895485	54.187136	181.566359	6.203112	972.918592	773.5788
std	104.507142	86.279104	63.996469	21.355567	5.973492	77.753818	80.1754
min	102.000000	0.000000	0.000000	121.750000	0.000000	801.000000	594.0000
25%	192.375000	0.000000	0.000000	164.900000	0.000000	932.000000	730.9500
50%	272.900000	22.000000	0.000000	185.000000	6.350000	968.000000	779.5100
75%	350.000000	142.950000	118.270000	192.000000	10.160000	1029.400000	824.0000
max	540.000000	359.400000	200.100000	247.000000	32.200000	1145.000000	992.6000

In [68]:
```
df.isna().sum()
```

Out[68]:
```
Cement                           0
Blast Furnace Slag               0
Fly Ash                          0
Water                            0
Superplasticizer                 0
Coarse Aggregate                 0
Fine Aggregate                   0
Age                              0
Concrete compressive strength    0
dtype: int64
```

In [69]:
```
df.duplicated().sum()
```

Out[69]:
```
25
```

In [70]:
```
df[df.duplicated()==True]
```

Out[70]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Concrete compressive strength
77	425.0	106.3	0.0	153.5	16.5	852.1	887.1	3	33.398217
80	425.0	106.3	0.0	153.5	16.5	852.1	887.1	3	33.398217
86	362.6	189.0	0.0	164.9	11.6	944.7	755.8	3	35.301171
88	362.6	189.0	0.0	164.9	11.6	944.7	755.8	3	35.301171
91	362.6	189.0	0.0	164.9	11.6	944.7	755.8	3	35.301171
100	425.0	106.3	0.0	153.5	16.5	852.1	887.1	7	49.201007
103	425.0	106.3	0.0	153.5	16.5	852.1	887.1	7	49.201007
109	362.6	189.0	0.0	164.9	11.6	944.7	755.8	7	55.895819
111	362.6	189.0	0.0	164.9	11.6	944.7	755.8	7	55.895819
123	425.0	106.3	0.0	153.5	16.5	852.1	887.1	28	60.294676
126	425.0	106.3	0.0	153.5	16.5	852.1	887.1	28	60.294676
132	362.6	189.0	0.0	164.9	11.6	944.7	755.8	28	71.298713
134	362.6	189.0	0.0	164.9	11.6	944.7	755.8	28	71.298713
137	362.6	189.0	0.0	164.9	11.6	944.7	755.8	28	71.298713
146	425.0	106.3	0.0	153.5	16.5	852.1	887.1	56	64.300532
149	425.0	106.3	0.0	153.5	16.5	852.1	887.1	56	64.300532
155	362.6	189.0	0.0	164.9	11.6	944.7	755.8	56	77.297154
157	362.6	189.0	0.0	164.9	11.6	944.7	755.8	56	77.297154
160	362.6	189.0	0.0	164.9	11.6	944.7	755.8	56	77.297154
169	425.0	106.3	0.0	153.5	16.5	852.1	887.1	91	65.196851
172	425.0	106.3	0.0	153.5	16.5	852.1	887.1	91	65.196851
177	362.6	189.0	0.0	164.9	11.6	944.7	755.8	91	79.296635
179	362.6	189.0	0.0	164.9	11.6	944.7	755.8	91	79.296635
182	362.6	189.0	0.0	164.9	11.6	944.7	755.8	91	79.296635
809	252.0	0.0	0.0	185.0	0.0	1111.0	784.0	28	19.691435

In [71]:

```
df.drop_duplicates(keep='first',inplace=True)
```

In [72]:

```
df.duplicated().sum()
```

Out[72]:

```
0
```

In [73]:

```
df
```

Out[73]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Concrete compressive strength
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.986111
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.887366
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.269535
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.052780
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.296075
...	...	...	...	...	...	...	...	...	...
1025	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28	44.284354
1026	322.2	0.0	115.6	196.0	10.4	817.9	813.4	28	31.178794
1027	148.5	139.4	108.6	192.7	6.1	892.4	780.0	28	23.696601
1028	159.1	186.7	0.0	175.6	11.3	989.6	788.9	28	32.768036
1029	260.9	100.5	78.3	200.6	8.6	864.5	761.5	28	32.401235

1005 rows × 9 columns

In [74]:

```
df.reset_index(drop=True,inplace=True)
```

In [75]:

```
df
```

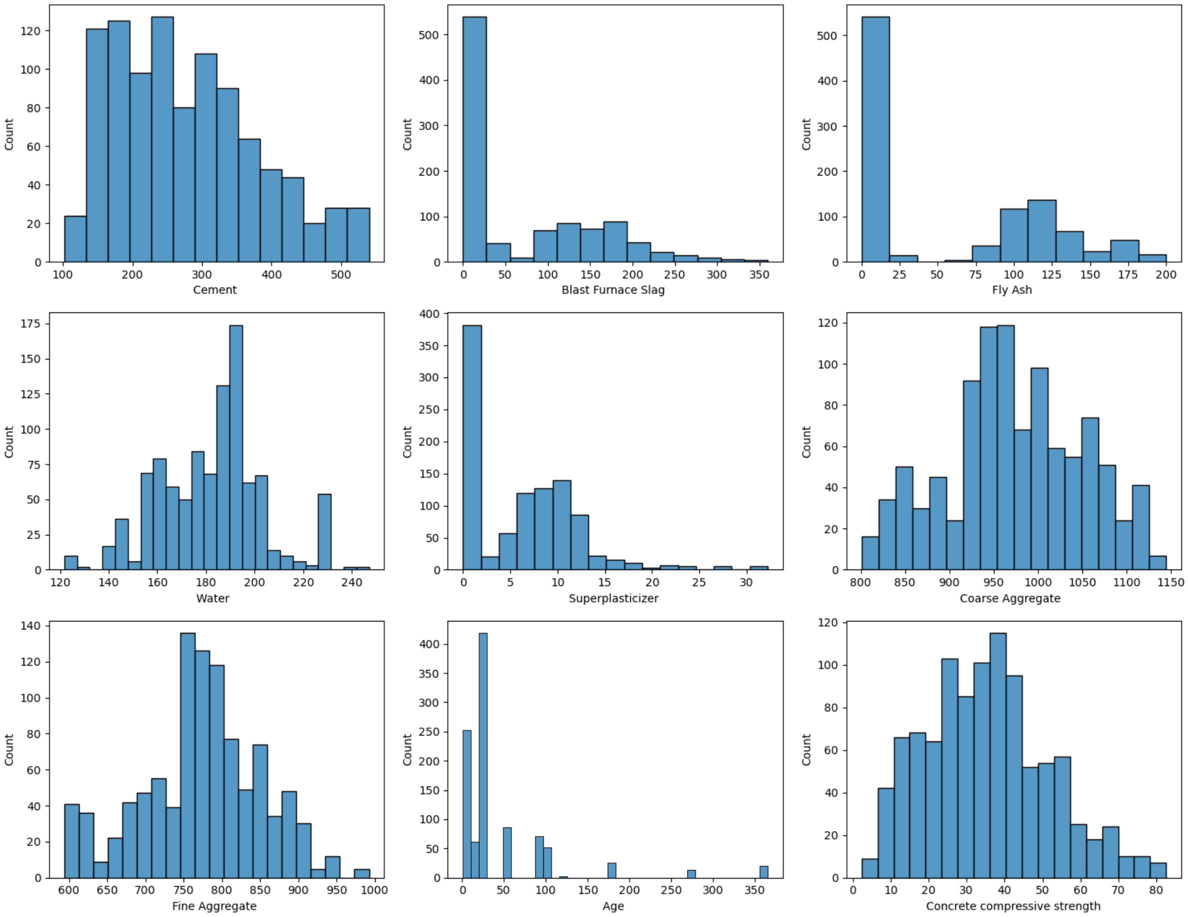
Out[75]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Concrete compressive strength
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.986111
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.887366
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.269535
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.052780
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.296075
...	...	...	...	...	...	...	...	...	...
1000	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28	44.284354
1001	322.2	0.0	115.6	196.0	10.4	817.9	813.4	28	31.178794
1002	148.5	139.4	108.6	192.7	6.1	892.4	780.0	28	23.696601
1003	159.1	186.7	0.0	175.6	11.3	989.6	788.9	28	32.768036
1004	260.9	100.5	78.3	200.6	8.6	864.5	761.5	28	32.401235

1005 rows × 9 columns

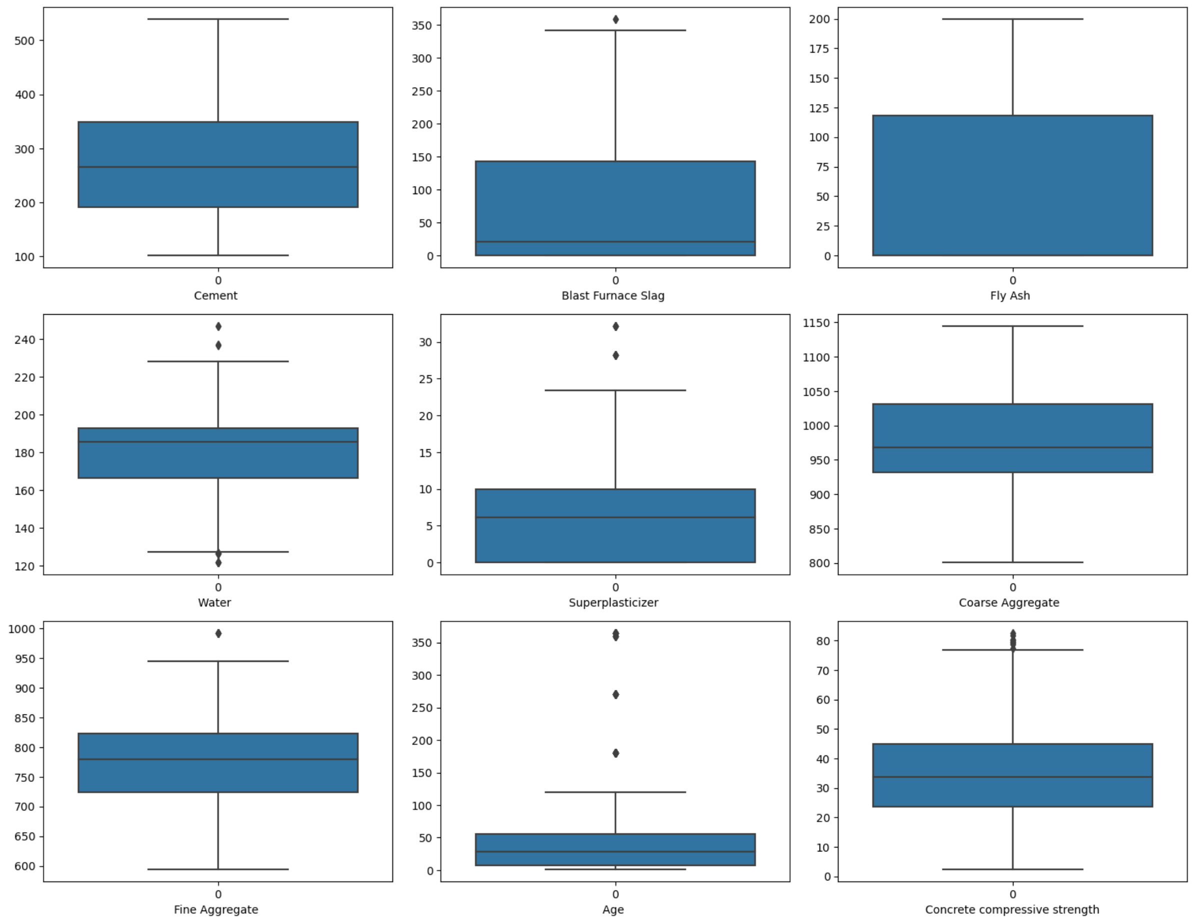
In [76]:

```
plt.figure(figsize=(15,15),facecolor='white')
plotnumber=1
for i in df.columns:
    ax=plt.subplot(4,3,plotnumber)
    sns.histplot(df[i])
    plt.xlabel(i,fontsize=10)
    plotnumber +=1
plt.tight_layout()
plt.show()
```



In [77]:

```
plt.figure(figsize=(15,15),facecolor='white')
plotnumber=1
for i in df.columns:
    ax=plt.subplot(4,3,plotnumber)
    sns.boxplot(df[i])
    plt.xlabel(i,fontsize=10)
    plotnumber +=1
plt.tight_layout()
plt.show()
```



In [78]:
```
df.columns
```

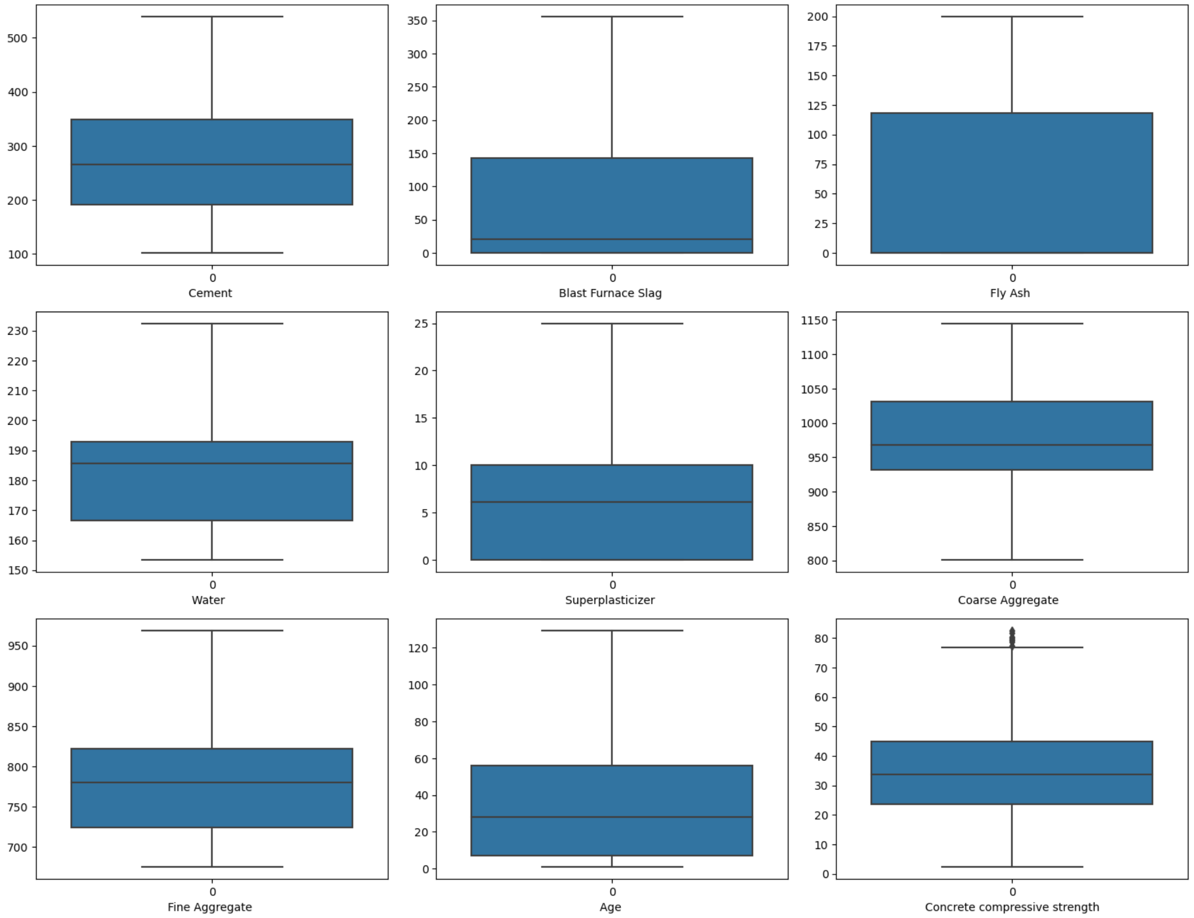
Out[78]:
```
Index(['Cement ', 'Blast Furnace Slag ', 'Fly Ash ', 'Water  ',
       'Superplasticizer ', 'Coarse Aggregate  ', 'Fine Aggregate ', 'Age ',
       'Concrete compressive strength'],
      dtype='object')
```

In [79]:
```
outliers=['Blast Furnace Slag ',"Water  ", "Superplasticizer ", 'Fine Aggregate ',
```

In [80]:
```
def outlier_capping(dataframe:pd.DataFrame,outliers:list):
    df=dataframe.copy()
    for i in outliers:
        q1=df[i].quantile(0.25)
        q3=df[i].quantile(0.75)
        iqr=q3-q1
        upper_limit=q3+1.5*iqr
        lower_limit=q3-1.5*iqr
        df.loc[df[i]>upper_limit,i]=upper_limit
        df.loc[df[i]<lower_limit,i]=lower_limit
    return df
```

In [81]:
```
 df=outlier_capping(dataframe=df,outliers=outliers)
```

In [82]:
```
plt.figure(figsize = (15,15), facecolor = 'white')
plotnumber = 1
for i in df.columns:
    ax = plt.subplot(4,3, plotnumber)
    sns.boxplot(df[i])
    plt.xlabel(i, fontsize = 10)
    plotnumber +=1
plt.tight_layout()
plt.show()
```

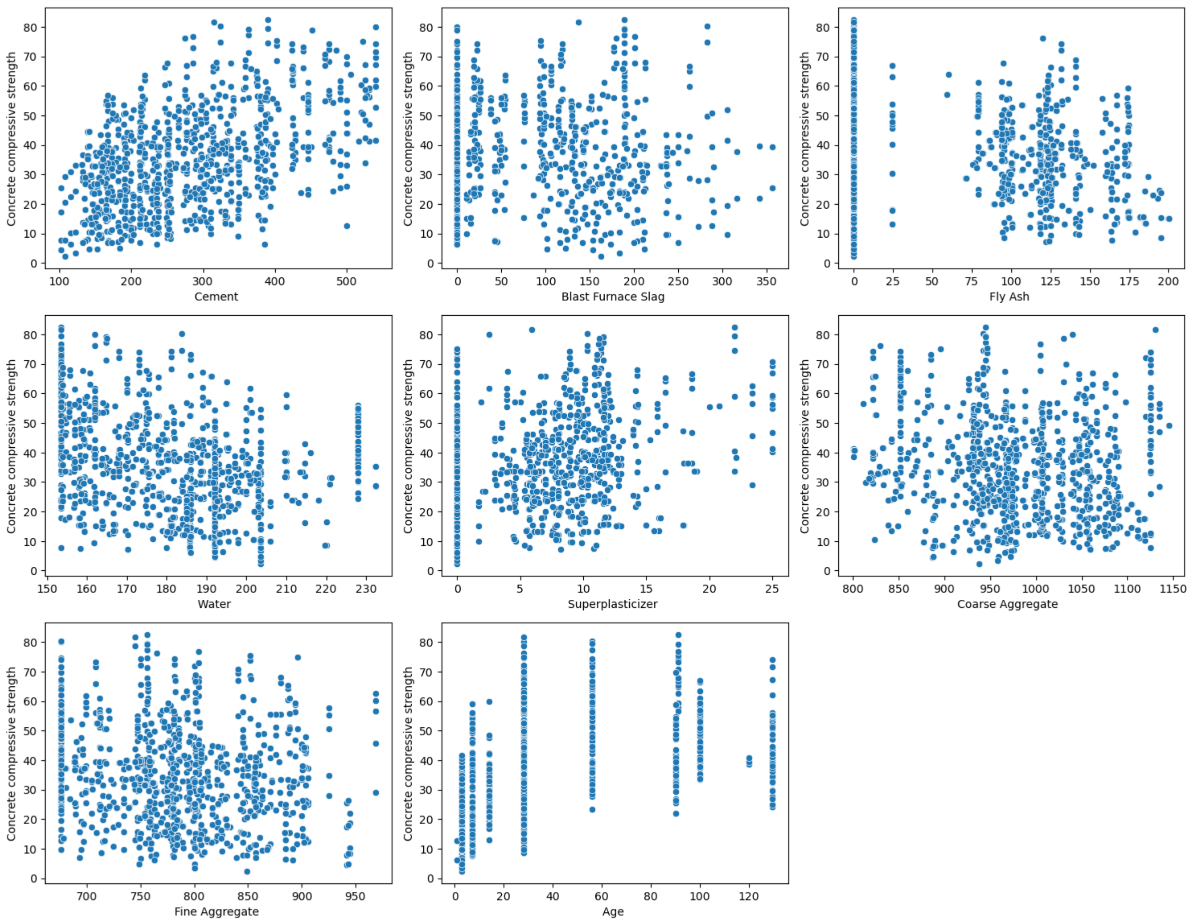


In [83]:

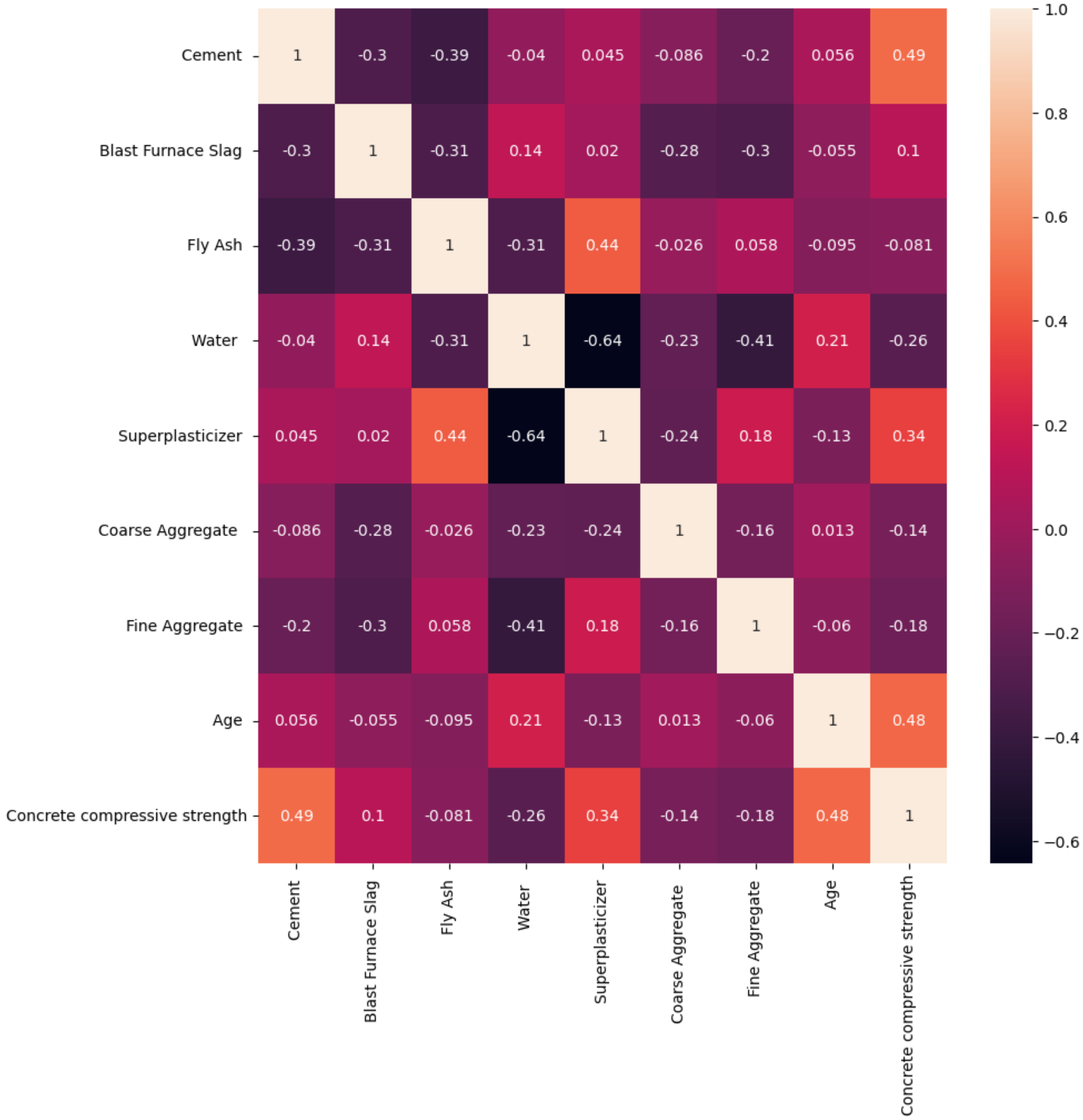
```
X=df.drop('Concrete compressive strength',axis=1)
y=df['Concrete compressive strength']
```

In [84]:

```
plt.figure(figsize = (15,15), facecolor = 'white')
plotnumber = 1
for i in X.columns:
    ax = plt.subplot(4,3, plotnumber)
    sns.scatterplot(x = df[i], y = y)
    plt.xlabel(i, fontsize = 10)
    plotnumber +=1
plt.tight_layout()
plt.show()
```



In [85]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



In [86]:

```
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(X, y, test_size = 0.3, random_state
```

In [87]:

```
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler, RobustScaler, MinMaxScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import mean_squared_error, r2_score
```

In [88]:

```
def check_model_performance(preprocessor, xtrain, ytrain, xtest, ytest):
    models = {'Linear Regression' : LinearRegression(),
              'Ridge Regression': Ridge(alpha = 1),
              'Lassor Regression': Lasso(alpha = 1),
              'Random Foreest Regression': RandomForestRegressor(max_depth= 5),
              "Gradient Boosting Regression": GradientBoostingRegressor(learning_rat
    for model_name, model in models.items():
        pipeline = make_pipeline(preprocessor, model)
        pipeline.fit(xtrain, ytrain)
        y_pred = pipeline.predict(xtest)
        mse = mean_squared_error(ytest, y_pred)
        r2 = r2_score(ytest, y_pred)
        print(f"{model_name} - Mean Squared Error = {mse:.2f} \n{model_name} - r2_s
```

In [89]:

```
preprocessor_01 = make_pipeline(KNNImputer(n_neighbors=3), StandardScaler())
preprocessor_02 = make_pipeline(KNNImputer(n_neighbors=3), MinMaxScaler())
preprocessor_03 = make_pipeline(KNNImputer(n_neighbors=3), RobustScaler())

print(f"{'=' * 10} Result for StandardScaler {'=' *10}")
check_model_performance(preprocessor_01, xtrain, ytrain, xtest, ytest)

print(f"\n{'=' * 10} Result for MinMaxScaler {'=' *10}")
check_model_performance(preprocessor_02, xtrain, ytrain, xtest, ytest)

print(f"\n{'=' * 10} Result for RobustScaler {'=' *10}")
check_model_performance(preprocessor_03, xtrain, ytrain, xtest, ytest)
```

```
========== Result for StandardScaler ==========
Linear Regression - Mean Squared Error = 88.36
Linear Regression - r2_score = 0.69
Ridge Regression - Mean Squared Error = 88.32
Ridge Regression - r2_score = 0.69
Lassor Regression - Mean Squared Error = 100.03
Lassor Regression - r2_score = 0.65
Random Foreest Regression - Mean Squared Error = 51.89
Random Foreest Regression - r2_score = 0.82
Gradient Boosting Regression - Mean Squared Error = 34.41
Gradient Boosting Regression - r2_score = 0.88

========== Result for MinMaxScaler ==========
Linear Regression - Mean Squared Error = 88.36
Linear Regression - r2_score = 0.69
Ridge Regression - Mean Squared Error = 88.32
Ridge Regression - r2_score = 0.69
Lassor Regression - Mean Squared Error = 181.19
Lassor Regression - r2_score = 0.37
Random Foreest Regression - Mean Squared Error = 52.63
Random Foreest Regression - r2_score = 0.82
Gradient Boosting Regression - Mean Squared Error = 34.35
Gradient Boosting Regression - r2_score = 0.88

========== Result for RobustScaler ==========
Linear Regression - Mean Squared Error = 88.36
Linear Regression - r2_score = 0.69
Ridge Regression - Mean Squared Error = 88.21
Ridge Regression - r2_score = 0.69
Lassor Regression - Mean Squared Error = 105.68
Lassor Regression - r2_score = 0.63
Random Foreest Regression - Mean Squared Error = 53.20
Random Foreest Regression - r2_score = 0.81
Gradient Boosting Regression - Mean Squared Error = 34.38
Gradient Boosting Regression - r2_score = 0.88
```

In [90]:
```
from sklearn.model_selection import GridSearchCV
param_grid={'n_estimators':[10,20],
            'learning_rate':[0.1,0.01],
            'max_depth':[5,3,7],
            'min_samples_split':[2,4],
            'min_samples_leaf':[1,2,3]}
```

In [108…
```
gb_rg = GradientBoostingRegressor()
grid = GridSearchCV(gb_rg, param_grid, scoring = 'neg_mean_squared_error', cv = 5,
grid.fit(xtrain, ytrain)
```

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
```

Out[108]:
```
▸              GridSearchCV
▸ estimator: GradientBoostingRegressor
      ▸ GradientBoostingRegressor
```

In [109…
```
grid.best_params_
```

Out[109]:
```
{'learning_rate': 0.1,
 'max_depth': 7,
 'min_samples_leaf': 1,
 'min_samples_split': 4,
 'n_estimators': 20}
```

In [110…
```
grid.best_score_
```

Out[110]:
```
-34.37396133399231
```

In [111…
```
grid.best_estimator_.score(xtest, ytest)
```

Out[111]:
```
0.8707908801478589
```

In [ ]: