Data Mining & Data Preprocessing

In [30]:

```
#Importing of Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [31]:

```
# Importing of Dataset (Ensure the path of your dataset)
Mining_Dataset=pd.read_csv(r'C:\Users\91965\Downloads\Mining_Dataset.csv', decimal=',',pars
```

In [32]:

```
Mining_Dataset.columns
```

Out[32]:

```
Index(['date', '% Iron Feed', '% Silica Feed', 'Starch Flow', 'Amina Flow',
       'Ore Pulp Flow', 'Ore Pulp pH', 'Ore Pulp Density',
       'Flotation Column 01 Air Flow', 'Flotation Column 02 Air Flow',
       'Flotation Column 03 Air Flow', 'Flotation Column 04 Air Flow',
       'Flotation Column 05 Air Flow', 'Flotation Column 06 Air Flow',
       'Flotation Column 07 Air Flow', 'Flotation Column 01 Level',
       'Flotation Column 02 Level', 'Flotation Column 03 Level',
       'Flotation Column 04 Level', 'Flotation Column 05 Level',
       'Flotation Column 06 Level', 'Flotation Column 07 Level',
       '% Iron Concentrate', '% Silica Concentrate'],
      dtype='object')
```

In [33]:

```
Mining_Dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 737453 entries, 0 to 737452
Data columns (total 24 columns):
 #   Column                        Non-Null Count   Dtype         
---  ------                        --------------   -----         
 0   date                          737453 non-null  datetime64[ns]
 1   % Iron Feed                   737453 non-null  float64       
 2   % Silica Feed                 737453 non-null  float64       
 3   Starch Flow                   737453 non-null  float64       
 4   Amina Flow                    737453 non-null  float64       
 5   Ore Pulp Flow                 737453 non-null  float64       
 6   Ore Pulp pH                   737453 non-null  float64       
 7   Ore Pulp Density              737453 non-null  float64       
 8   Flotation Column 01 Air Flow  737453 non-null  float64       
 9   Flotation Column 02 Air Flow  737453 non-null  float64       
 10  Flotation Column 03 Air Flow  737453 non-null  float64       
 11  Flotation Column 04 Air Flow  737453 non-null  float64       
 12  Flotation Column 05 Air Flow  737453 non-null  float64       
 13  Flotation Column 06 Air Flow  737453 non-null  float64       
 14  Flotation Column 07 Air Flow  737453 non-null  float64       
 15  Flotation Column 01 Level     737453 non-null  float64       
 16  Flotation Column 02 Level     737453 non-null  float64       
 17  Flotation Column 03 Level     737453 non-null  float64       
 18  Flotation Column 04 Level     737453 non-null  float64       
 19  Flotation Column 05 Level     737453 non-null  float64       
 20  Flotation Column 06 Level     737453 non-null  float64       
 21  Flotation Column 07 Level     737453 non-null  float64       
 22  % Iron Concentrate            737453 non-null  float64       
 23  % Silica Concentrate          737453 non-null  float64       
dtypes: datetime64[ns](1), float64(23)
memory usage: 135.0 MB
```

In [34]:

```
# Description of the Dataset
Mining_Dataset.describe()
```

Out[34]:

	% Iron Feed	% Silica Feed	Starch Flow	Amina Flow	Ore Pulp Flow	Ore Pulp
count	737453.000000	737453.000000	737453.000000	737453.000000	737453.000000	737453.000(
mean	56.294739	14.651716	2869.140569	488.144697	397.578372	9.767(
std	5.157744	6.807439	1215.203734	91.230534	9.699785	0.387(
min	42.740000	1.310000	0.002026	241.669000	376.249000	8.753(
25%	52.670000	8.940000	2076.320000	431.796000	394.264000	9.527(
50%	56.080000	13.850000	3018.430000	504.393000	399.249000	9.798(
75%	59.720000	19.600000	3727.730000	553.257000	402.968000	10.038(
max	65.780000	33.400000	6300.230000	739.538000	418.641000	10.808(

8 rows × 23 columns

In [35]:

```
#checking and removal of existing negative values in the dataset
for cols in Mining_Dataset.columns.tolist()[1:]:
    df = Mining_Dataset.loc[Mining_Dataset[cols] > 0]
df.info()
```

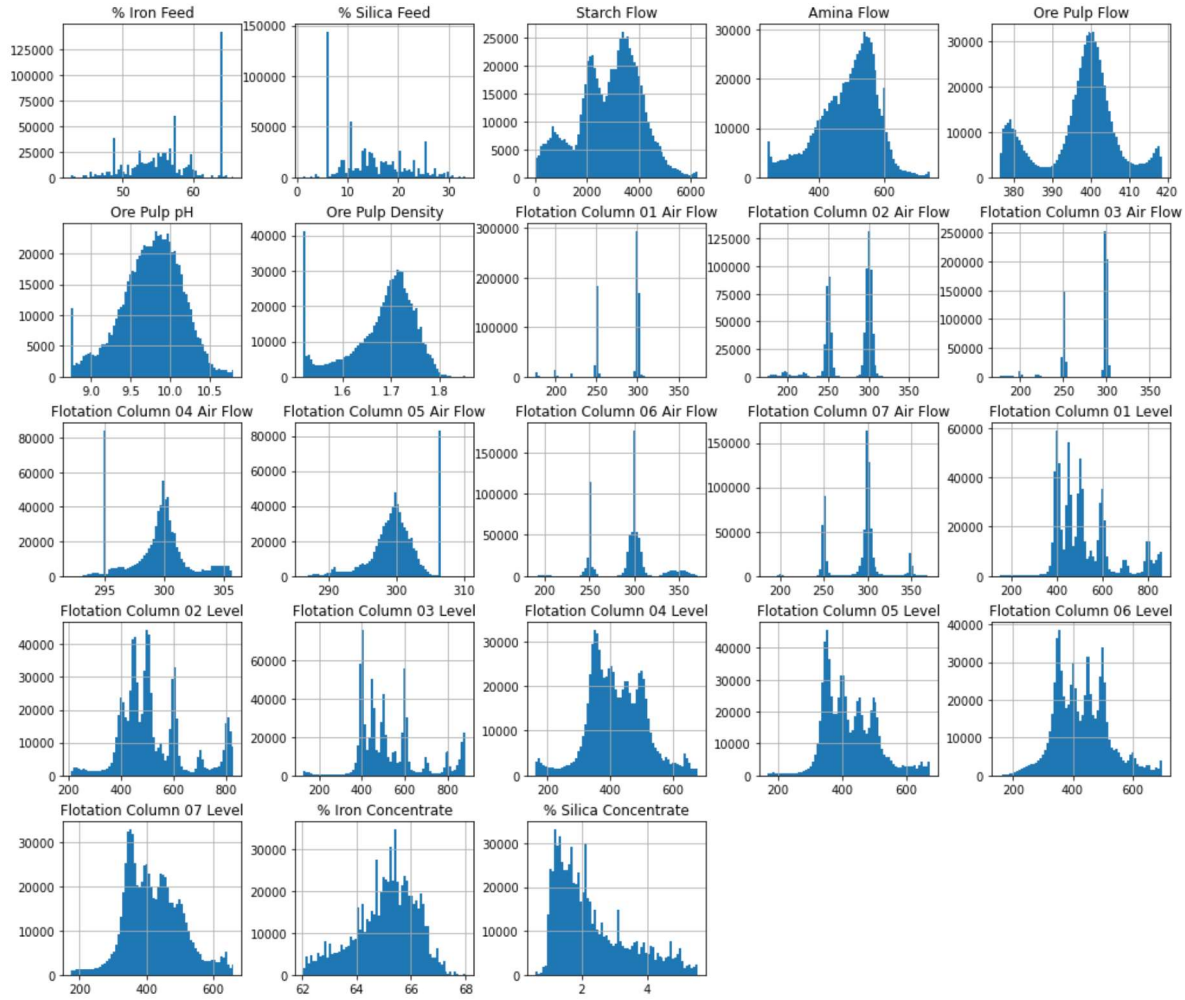
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 737453 entries, 0 to 737452
Data columns (total 24 columns):
 #   Column                        Non-Null Count   Dtype         
---  ------                        --------------   -----         
 0   date                          737453 non-null  datetime64[ns]
 1   % Iron Feed                   737453 non-null  float64       
 2   % Silica Feed                 737453 non-null  float64       
 3   Starch Flow                   737453 non-null  float64       
 4   Amina Flow                    737453 non-null  float64       
 5   Ore Pulp Flow                 737453 non-null  float64       
 6   Ore Pulp pH                   737453 non-null  float64       
 7   Ore Pulp Density              737453 non-null  float64       
 8   Flotation Column 01 Air Flow  737453 non-null  float64       
 9   Flotation Column 02 Air Flow  737453 non-null  float64       
 10  Flotation Column 03 Air Flow  737453 non-null  float64       
 11  Flotation Column 04 Air Flow  737453 non-null  float64       
 12  Flotation Column 05 Air Flow  737453 non-null  float64       
 13  Flotation Column 06 Air Flow  737453 non-null  float64       
 14  Flotation Column 07 Air Flow  737453 non-null  float64       
 15  Flotation Column 01 Level     737453 non-null  float64       
 16  Flotation Column 02 Level     737453 non-null  float64       
 17  Flotation Column 03 Level     737453 non-null  float64       
 18  Flotation Column 04 Level     737453 non-null  float64       
 19  Flotation Column 05 Level     737453 non-null  float64       
 20  Flotation Column 06 Level     737453 non-null  float64       
 21  Flotation Column 07 Level     737453 non-null  float64       
 22  % Iron Concentrate            737453 non-null  float64       
 23  % Silica Concentrate          737453 non-null  float64       
dtypes: datetime64[ns](1), float64(23)
memory usage: 140.7 MB
```

In [36]:

```
# Indexing of Date Colomn
df=df.set_index('date')
```

In [37]:

```
# Visualize data distribution to identify whether outliers exist
import matplotlib.pyplot as plt
from matplotlib import style
df.hist(bins = 70, figsize = (17,15))
plt.show()
plt.suptitle('figure title', color='w')
```



Out[37]:

```
Text(0.5, 0.98, 'figure title')
```

```
<Figure size 432x288 with 0 Axes>
```

In [38]:

```
df.head()
```

Out[38]:

date	% Iron Feed	% Silica Feed	Starch Flow	Amina Flow	Ore Pulp Flow	Ore Pulp pH	Ore Pulp Density	Flotation Column 01 Air Flow	Flotation Column 02 Air Flow	Flotatic Colun 03 A Flo
2017-03-10 01:00:00	55.2	16.98	3019.53	557.434	395.713	10.0664	1.74	249.214	253.235	250.5
2017-03-10 01:00:00	55.2	16.98	3024.41	563.965	397.383	10.0672	1.74	249.719	250.532	250.8
2017-03-10 01:00:00	55.2	16.98	3043.46	568.054	399.668	10.0680	1.74	249.741	247.874	250.3
2017-03-10 01:00:00	55.2	16.98	3047.36	568.665	397.939	10.0689	1.74	249.917	254.487	250.0
2017-03-10 01:00:00	55.2	16.98	3033.69	558.167	400.254	10.0697	1.74	250.203	252.136	249.8

5 rows × 23 columns

In [39]:

```
#Removal of Outliers using z-score method & datapoints lying outside 2 standard deviations
df_copy=df
from scipy import stats
df_copy=df[(np.abs(stats.zscore(df)) < 2).all(axis=1)]
df_copy.info()

#For each column, first it computes the Z-score of each value in the column, relative to th
#Then is takes the absolute of Z-score because the direction does not matter, only if it is
#all(axis=1) ensures that for each row, all column satisfy the constraint.
#Finally, result of this condition is used to index the dataframe.
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 346475 entries, 2017-03-10 01:00:00 to 2017-09-09 22:00:00
Data columns (total 23 columns):
 #   Column                        Non-Null Count   Dtype  
---  ------                        --------------   -----  
 0   % Iron Feed                   346475 non-null  float64
 1   % Silica Feed                 346475 non-null  float64
 2   Starch Flow                   346475 non-null  float64
 3   Amina Flow                    346475 non-null  float64
 4   Ore Pulp Flow                 346475 non-null  float64
 5   Ore Pulp pH                   346475 non-null  float64
 6   Ore Pulp Density              346475 non-null  float64
 7   Flotation Column 01 Air Flow  346475 non-null  float64
 8   Flotation Column 02 Air Flow  346475 non-null  float64
 9   Flotation Column 03 Air Flow  346475 non-null  float64
 10  Flotation Column 04 Air Flow  346475 non-null  float64
 11  Flotation Column 05 Air Flow  346475 non-null  float64
 12  Flotation Column 06 Air Flow  346475 non-null  float64
 13  Flotation Column 07 Air Flow  346475 non-null  float64
 14  Flotation Column 01 Level     346475 non-null  float64
 15  Flotation Column 02 Level     346475 non-null  float64
 16  Flotation Column 03 Level     346475 non-null  float64
 17  Flotation Column 04 Level     346475 non-null  float64
 18  Flotation Column 05 Level     346475 non-null  float64
 19  Flotation Column 06 Level     346475 non-null  float64
 20  Flotation Column 07 Level     346475 non-null  float64
 21  % Iron Concentrate            346475 non-null  float64
 22  % Silica Concentrate          346475 non-null  float64
dtypes: float64(23)
memory usage: 63.4 MB
```

In [40]:

```
df_copy.head()
```

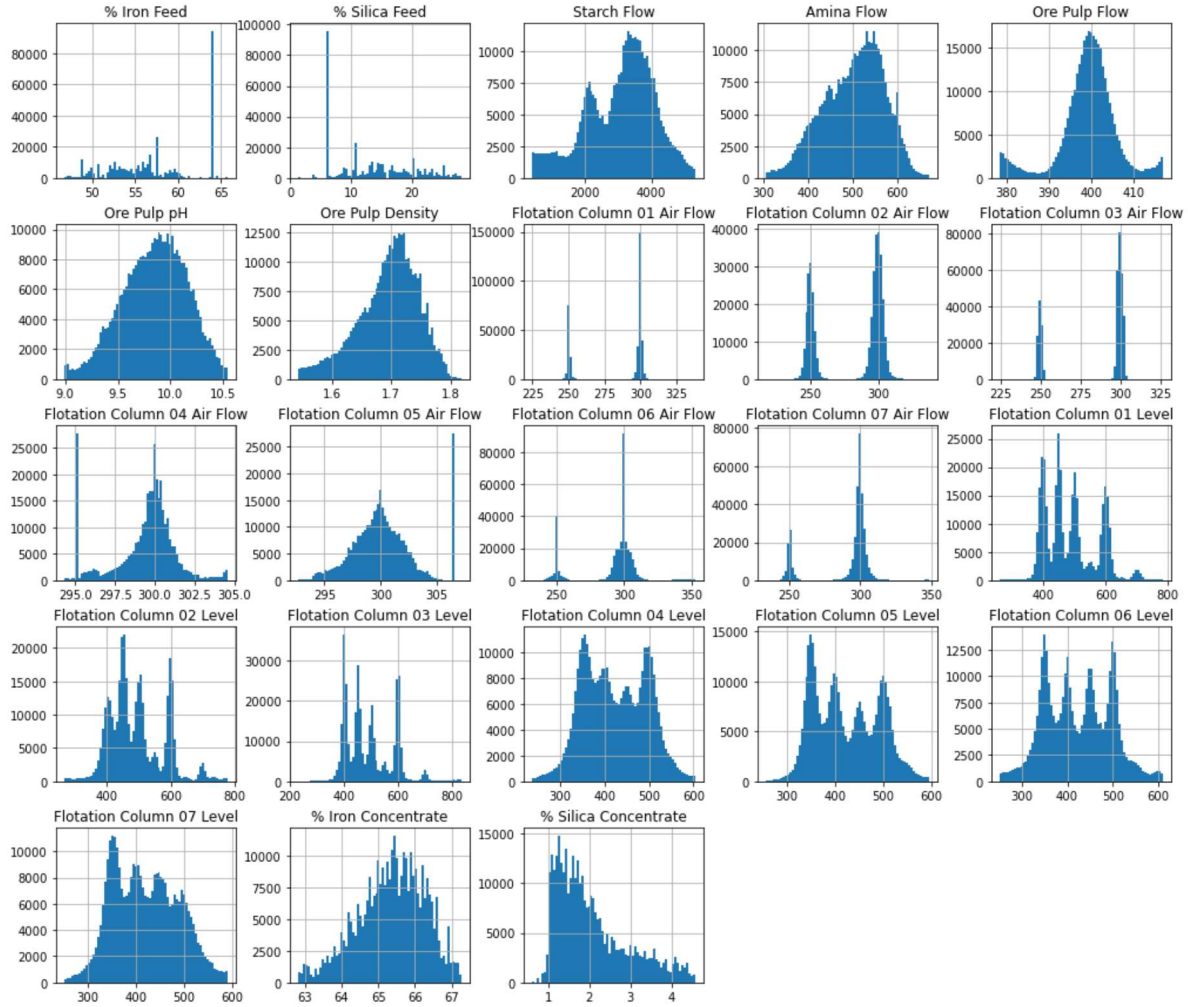
Out[40]:

date	% Iron Feed	% Silica Feed	Starch Flow	Amina Flow	Ore Pulp Flow	Ore Pulp pH	Ore Pulp Density	Flotation Column 01 Air Flow	Flotation Column 02 Air Flow	Flotatic Colun 03 A Flo
2017-03-10 01:00:00	55.2	16.98	3019.53	557.434	395.713	10.0664	1.74	249.214	253.235	250.5
2017-03-10 01:00:00	55.2	16.98	3024.41	563.965	397.383	10.0672	1.74	249.719	250.532	250.8
2017-03-10 01:00:00	55.2	16.98	3043.46	568.054	399.668	10.0680	1.74	249.741	247.874	250.3
2017-03-10 01:00:00	55.2	16.98	3047.36	568.665	397.939	10.0689	1.74	249.917	254.487	250.0
2017-03-10 01:00:00	55.2	16.98	3033.69	558.167	400.254	10.0697	1.74	250.203	252.136	249.8

5 rows × 23 columns

In [41]:

```
# Visualization of data after removal of outliers
df_copy.hist(bins = 70, figsize = (17,15))
plt.show()
plt.suptitle('figure title', color='w')
df_copy.info()
```



```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 346475 entries, 2017-03-10 01:00:00 to 2017-09-09 22:00:00
Data columns (total 23 columns):
 #   Column                        Non-Null Count   Dtype  
---  ------                        --------------   -----  
 0   % Iron Feed                   346475 non-null  float64
 1   % Silica Feed                 346475 non-null  float64
 2   Starch Flow                   346475 non-null  float64
 3   Amina Flow                    346475 non-null  float64
 4   Ore Pulp Flow                 346475 non-null  float64
 5   Ore Pulp pH                   346475 non-null  float64
 6   Ore Pulp Density              346475 non-null  float64
 7   Flotation Column 01 Air Flow  346475 non-null  float64
 8   Flotation Column 02 Air Flow  346475 non-null  float64
 9   Flotation Column 03 Air Flow  346475 non-null  float64
 10  Flotation Column 04 Air Flow  346475 non-null  float64
 11  Flotation Column 05 Air Flow  346475 non-null  float64
 12  Flotation Column 06 Air Flow  346475 non-null  float64
 13  Flotation Column 07 Air Flow  346475 non-null  float64
 14  Flotation Column 01 Level     346475 non-null  float64
 15  Flotation Column 02 Level     346475 non-null  float64
 16  Flotation Column 03 Level     346475 non-null  float64
 17  Flotation Column 04 Level     346475 non-null  float64
 18  Flotation Column 05 Level     346475 non-null  float64
 19  Flotation Column 06 Level     346475 non-null  float64
 20  Flotation Column 07 Level     346475 non-null  float64
 21  % Iron Concentrate            346475 non-null  float64
 22  % Silica Concentrate          346475 non-null  float64
dtypes: float64(23)
memory usage: 63.4 MB


<Figure size 432x288 with 0 Axes>
```

In [42]:

```
#Dropping % Iron Concentrate Given that it is an output of laboratory analysis & Irrevelant
df_copy.drop(columns=['% Iron Concentrate'],inplace=True)
df_copy.head()
```

```
E:\91965\lib\site-packages\pandas\core\frame.py:4308: SettingWithCopyWarnin
g:
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://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  return super().drop(
```

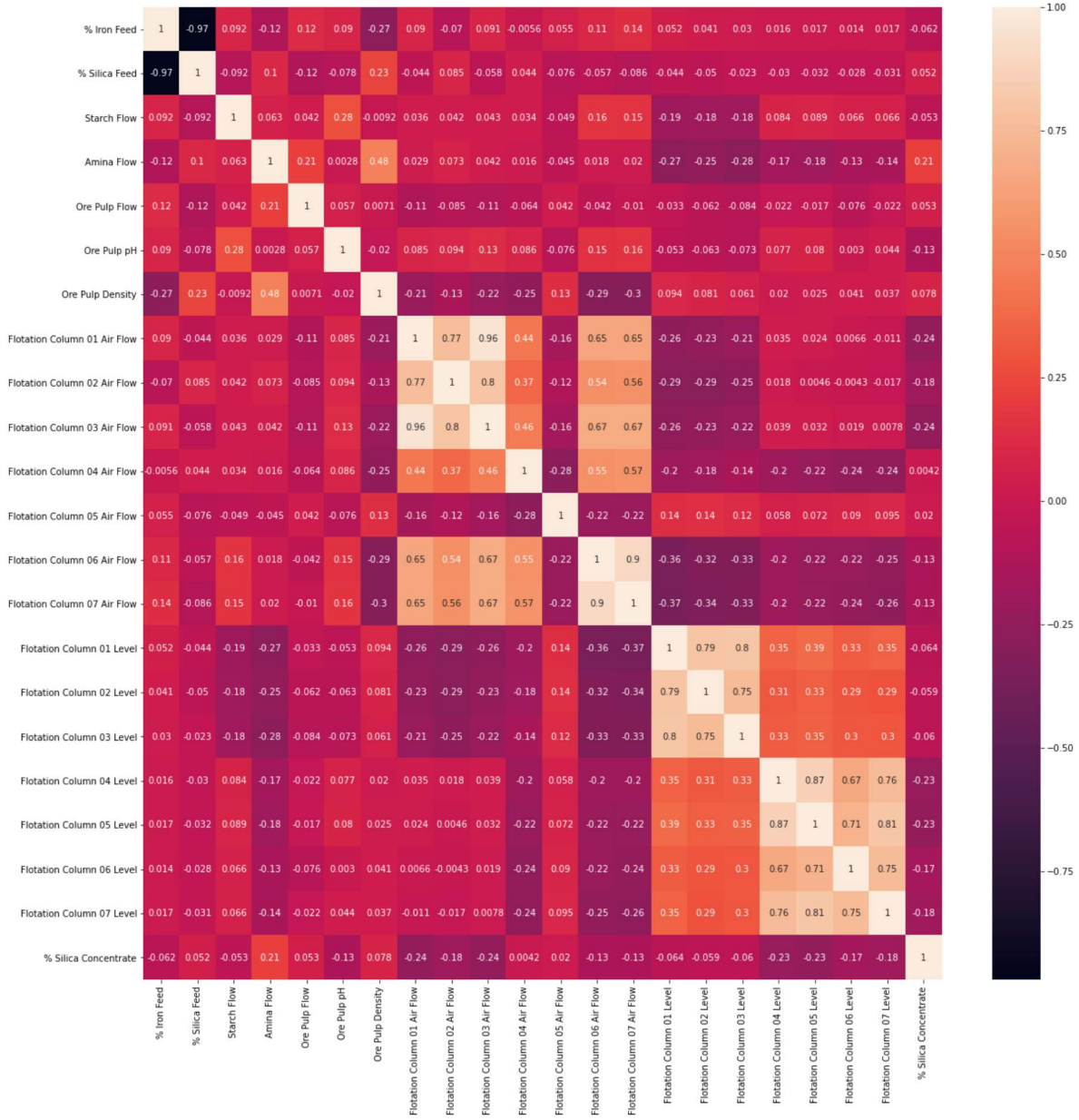
Out[42]:

date	% Iron Feed	% Silica Feed	Starch Flow	Amina Flow	Ore Pulp Flow	Ore Pulp pH	Ore Pulp Density	Flotation Column 01 Air Flow	Flotation Column 02 Air Flow	Flotatic Colun 03 A Flo
2017-03-10 01:00:00	55.2	16.98	3019.53	557.434	395.713	10.0664	1.74	249.214	253.235	250.5
2017-03-10 01:00:00	55.2	16.98	3024.41	563.965	397.383	10.0672	1.74	249.719	250.532	250.8
2017-03-10 01:00:00	55.2	16.98	3043.46	568.054	399.668	10.0680	1.74	249.741	247.874	250.3
2017-03-10 01:00:00	55.2	16.98	3047.36	568.665	397.939	10.0689	1.74	249.917	254.487	250.0
2017-03-10 01:00:00	55.2	16.98	3033.69	558.167	400.254	10.0697	1.74	250.203	252.136	249.8

5 rows × 22 columns

In [43]:

```
# Finding of Correlation among the Features
import seaborn as sns
ml_mining_data=df_copy
plt.figure(figsize=(20, 20))
p = sns.heatmap(ml_mining_data.corr(), annot=True);
```

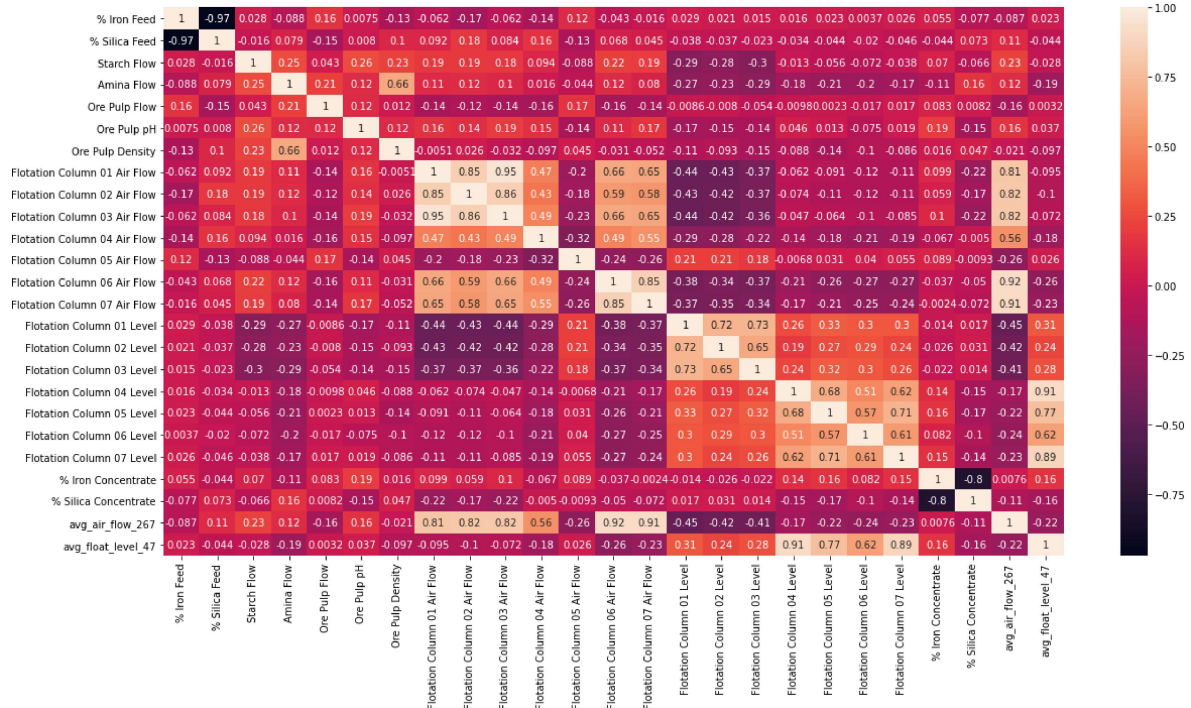


In [44]:

```
# Extraction of Features

df_copy=ml_mining_data.drop(['% Silica Feed', 'Starch Flow','Ore Pulp Flow','Flotation Colu
                         'Flotation Column 03 Air Flow', 'Flotation Column 04 Air Flow','Flo
                         'Flotation Column 01 Level','Flotation Column 02 Level', 'Flotation
                         'Flotation Column 05 Level','Flotation Column 06 Level'], axis = 1)


#Decision Criteria was based on
#1) correlation of feature to target variable (note some features with low correlation were
#2) removing collinier data points to prevent overfitting
#3) Reviewing the distribution of features and ensuring the feature has continous distribut

#commented below are the columns to keep based on manual Decisions on which columns to keep
#ml_mining_data['% Iron Feed', 'Amina Flow','Ore Pulp pH', 'Flotation Column 02 Air Flow',
                          #   'Flotation Column 06 Air Flow','Flotation Column 07 Air Flow'
                           #  'Flotation Column 04 Level', 'Flotation Column 07 Level',
                            # '% Silica Concentrate']
```

In [45]:

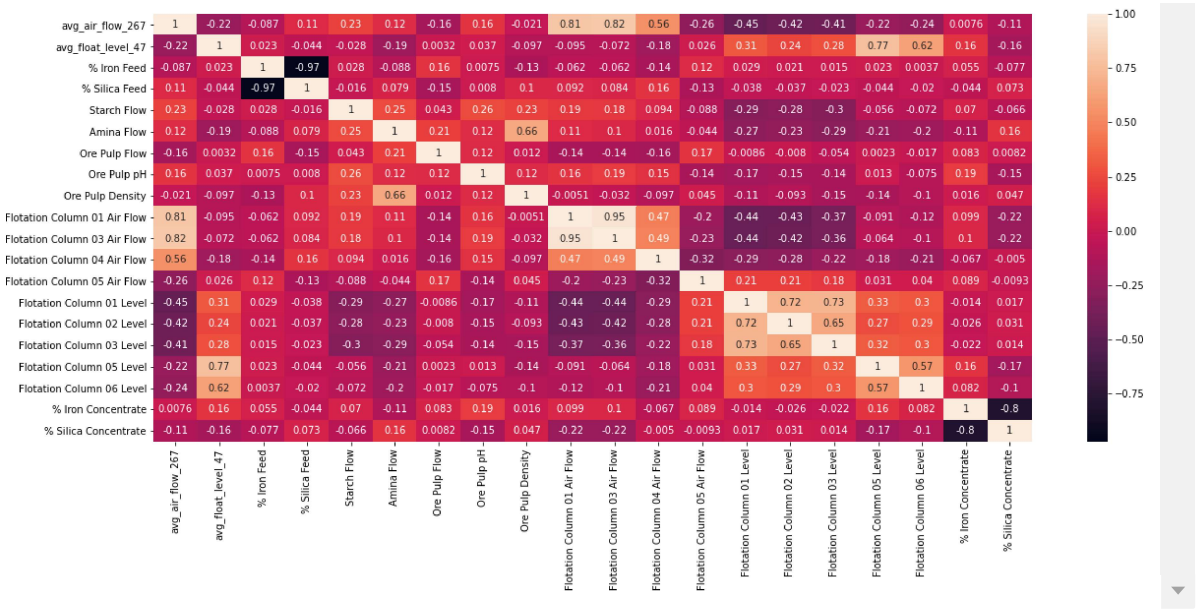
```
#combining remaining air flow features by taking their mean, and similarly for the level da
# This would also help later on when building the machine learning models given the less co


df.head()

# Average of the colomns
df['avg_air_flow_267'] = df[['Flotation Column 02 Air Flow','Flotation Column 06 Air Flow',
                             'Flotation Column 07 Air Flow']].mean(axis=1)
df['avg_float_level_47'] = df[['Flotation Column 04 Level', 'Flotation Column 07 Level']].m
df.head()
plt.figure(figsize=(20, 10))
p = sns.heatmap(df.corr(), annot=True);
```

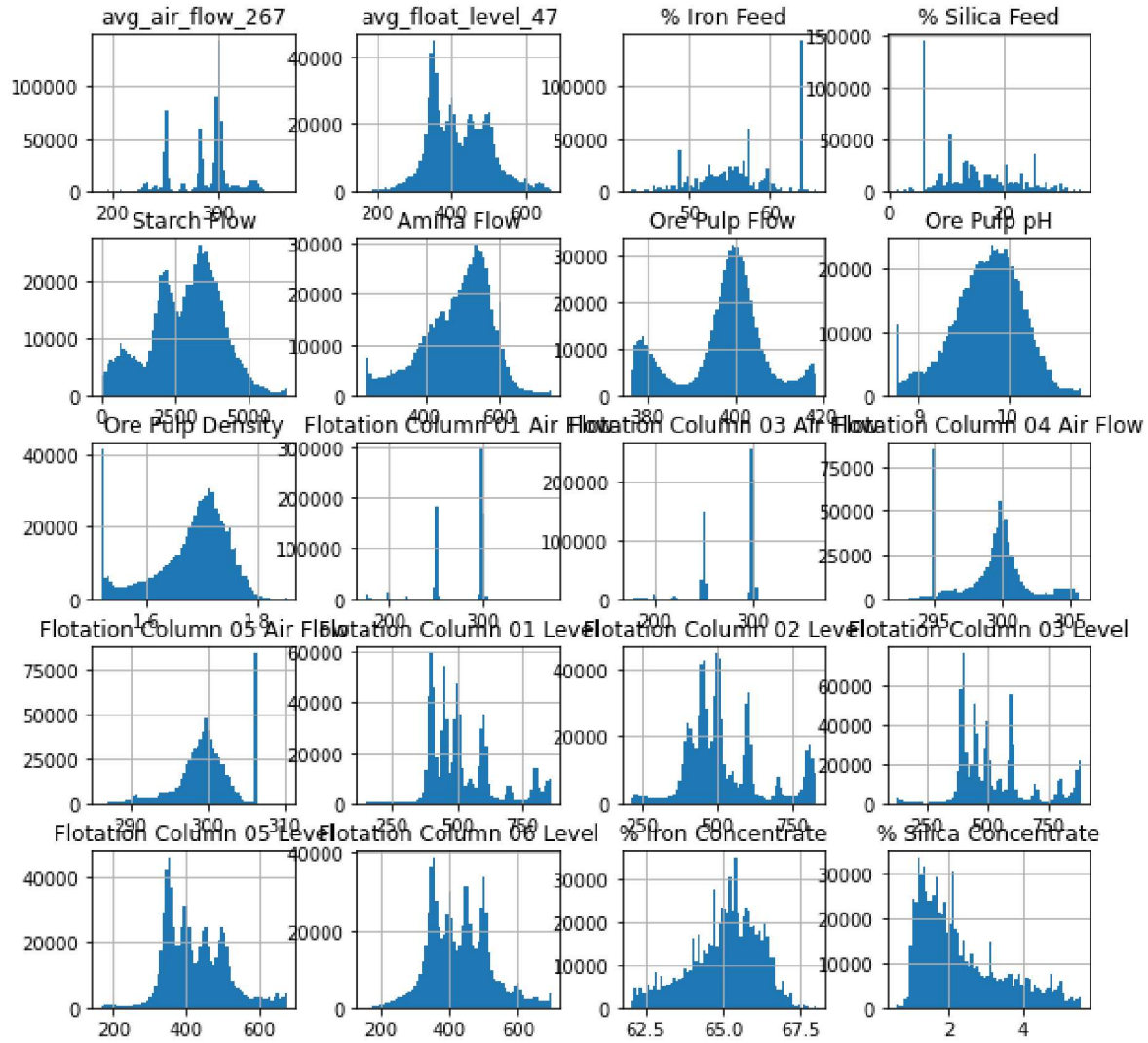


In [46]:

```
#Further cleaning of dataset and final view of correlations

df_ml=df.drop(['Flotation Column 02 Air Flow','Flotation Column 06 Air Flow','Flotation Col
               'Flotation Column 04 Level', 'Flotation Column 07 Level'], axis = 1)
df_ml.head()
cols = df_ml.columns.tolist()
cols
cols = cols[-1:] + cols[:-1]
cols = cols[-1:] + cols[:-1]
df_ml = df_ml[cols]
df_ml.head()
df_ml.info()
plt.figure(figsize=(20, 8))
p = sns.heatmap(df_ml.corr(), annot=True);

df_ml.hist(bins = 70, figsize = (10,10))
plt.show()
plt.suptitle('figure title', color='w')
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 737453 entries, 2017-03-10 01:00:00 to 2017-09-09 23:00:00
Data columns (total 20 columns):
 #   Column                         Non-Null Count   Dtype  
---  ------                         --------------   -----  
 0   avg_air_flow_267               737453 non-null  float64
 1   avg_float_level_47             737453 non-null  float64
 2   % Iron Feed                    737453 non-null  float64
 3   % Silica Feed                  737453 non-null  float64
 4   Starch Flow                    737453 non-null  float64
 5   Amina Flow                     737453 non-null  float64
 6   Ore Pulp Flow                  737453 non-null  float64
 7   Ore Pulp pH                    737453 non-null  float64
 8   Ore Pulp Density               737453 non-null  float64
 9   Flotation Column 01 Air Flow   737453 non-null  float64
 10  Flotation Column 03 Air Flow   737453 non-null  float64
 11  Flotation Column 04 Air Flow   737453 non-null  float64
 12  Flotation Column 05 Air Flow   737453 non-null  float64
 13  Flotation Column 01 Level      737453 non-null  float64
 14  Flotation Column 02 Level      737453 non-null  float64
 15  Flotation Column 03 Level      737453 non-null  float64
 16  Flotation Column 05 Level      737453 non-null  float64
 17  Flotation Column 06 Level      737453 non-null  float64
 18  % Iron Concentrate             737453 non-null  float64
 19  % Silica Concentrate           737453 non-null  float64
dtypes: float64(20)
memory usage: 118.2 MB
```





Out[46]:

```
Text(0.5, 0.98, 'figure title')
```

```
<Figure size 432x288 with 0 Axes>
```

In [48]:

```
#change"<mention your location path>"with the file path to where you want to save your exce
export_csv = df_ml.to_csv(r'C:\Users\91965\OneDrive\Desktop\Enhanced_Mining_dataset.csv',
                          header= True, index=True)
```

Model Building & Finding out the Accuracy

In [49]:

```
# Importing of Libraries
import pandas as pd
import numpy as np
```

In [50]:

```
# Importing of Dataset
# Indexing of the Date
# view top 5 row of the dataset using head function
# Change your dataset name

dataset=pd.read_csv(r'C:\Users\91965\OneDrive\Desktop\Enhanced_Mining_dataset.csv',sep=',')
dataset=dataset.set_index('date')
dataset.head()
```

Out[50]:

date	avg_air_flow_267	avg_float_level_47	% Iron Feed	% Silica Feed	Starch Flow	Amina Flow	Ore Pulp Flow	Ore Pulp pH
2017-03-10 01:00:00	251.448000	483.4510	55.2	16.98	3019.53	557.434	395.713	10.0664
2017-03-10 01:00:00	249.887667	473.0805	55.2	16.98	3024.41	563.965	397.383	10.0672
2017-03-10 01:00:00	249.096667	454.1275	55.2	16.98	3043.46	568.054	399.668	10.0680
2017-03-10 01:00:00	252.018667	436.9395	55.2	16.98	3047.36	568.665	397.939	10.0689
2017-03-10 01:00:00	250.349000	439.6745	55.2	16.98	3033.69	558.167	400.254	10.0697

In [51]:

```
dataset=dataset.set_index('date')
dataset.head()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-51-de63fa181aff> in <module>
----> 1 dataset=dataset.set_index('date')
      2 dataset.head()

E:\91965\lib\site-packages\pandas\core\frame.py in set_index(self, keys, dro
p, append, inplace, verify_integrity)
   4725
   4726         if missing:
-> 4727             raise KeyError(f"None of {missing} are in the columns")
   4728
   4729         if inplace:

KeyError: "None of ['date'] are in the columns"
```

In [53]:

```
# Spliting of Dataset into Independent & Dependent Variables

#Independent Variables
x=dataset.iloc[:,:-1].values

#Dependent Variable
y=dataset.iloc[:,-1].values
```

In [6]:

```
# Independent Variables
x
```

Out[6]:

```
array([[ 66.91 ,   1.31 ,  55.2  , ..., 432.962, 424.954, 502.255],
       [ 66.91 ,   1.31 ,  55.2  , ..., 429.56 , 432.939, 496.363],
       [ 66.91 ,   1.31 ,  55.2  , ..., 468.927, 434.61 , 484.411],
       ...,
       [ 64.27 ,   1.71 ,  49.75 , ..., 399.316, 867.598, 503.414],
       [ 64.27 ,   1.71 ,  49.75 , ..., 466.832, 876.591, 502.301],
       [ 64.27 ,   1.71 ,  49.75 , ..., 514.143, 881.323, 500.1  ]])
```

In [7]:

```
# Dependent Variable
y
```

Out[7]:

```
array([446.37 , 445.922, 447.826, ..., 336.035, 340.844, 374.354])
```

In [54]:

```
# Spliting of Dataset into Trainset & Testset

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [55]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [ ]:

```
# Importing the Random forest regressor model
from sklearn.ensemble import RandomForestRegressor
model=RandomForestRegressor(n_estimators=42,criterion='mse')
# Fitting the Random forest regressor to x_train & y_train
model.fit(x_train,y_train)
```

In [ ]:

```
# Prediction

y_pred=model.predict(x_test)
y_pred
```

In [ ]:

```
# Finding Accuracy of the model
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

In [ ]:

```
import pickle
pickle.dump(model,open('mining.pkl','wb'))
```

In [ ]:

```
model.predict([[251.448000,483.4510,55.20,557.434,10.06640,1.74000]])
```

In [ ]:

```

```