In [1]:
```
#Impoting the libraries
import os
import time
import pandas as pd
import numpy as np
from sklearn.neighbors import LocalOutlierFactor
from sklearn.ensemble import IsolationForest
import matplotlib.pyplot as plt
import mysql.connector
```

In [2]:
```
cnx = mysql.connector.connect(
  user='root', host='localhost', port=3306,passwd = 'passwd',
  database="Grahnumb"
)
```

In [4]:
```
# Functions to detect anamoly(outliers)
def model_1(data):
        time_series = data['value'].values.reshape(-1, 1)
    # Fit the Isolation Forest model to the data
        model_1 = IsolationForest(contamination=0.005)
        model_1.fit(time_series)
        data['outliers_1']= pd.Series(model_1.predict(data[['value']])).apply(lambda x: 'yes' if (x==-1) else 'no')
    # returs a dataframe that contains all the anomaly datapoints.
        return data.query('outliers_1=="yes"')

def model_2(data):
        time_series = data.iloc[:,0].values.reshape(-1, 1)
    # using LocalOutlierFactor method
        model_2 = LocalOutlierFactor(n_neighbors=20)
        data['outliers_2']= pd.Series(model_2.fit_predict(data[['value']])).apply(lambda x: 'yes' if (x==-1) else 'no')
    # returs a dataframe that contains all the anomaly datapoints.
        return data.query('outliers_2=="yes"')
```

In [13]:
```
def detect_anomalies(file_path, database_connection):
    #reading the data from the folder
    data = pd.read_excel(file_path)

    anomalies_1 = model_1(data)
    anomalies_2 = model_2(data)

    if len(anomalies_1) > len(anomalies_2):         # basic intiution if a model predicts more anamolies then its bette  !!!haha!!
        insert_stmt = ("INSERT INTO anomaly_present(filepath, no_of_anomaly)"
                       "VALUES (%s, %s)")
        data = (file_path,len(anomalies_1))
        database_connection.execute(insert_stmt, data) # inserting the file path and no_of anamoles to a table called anamoly_present
        for index, row in anomalies_1.iterrows():
            database_connection.execute("INSERT INTO anomalies (filepath,anomalies) VALUES (%s,%s)", (file_path,str(row.to_dict()),))
                # storing the file path and all the anamoly datapoint to a table called anomales.
        cnx.commit()

    else:
        insert_stmt = ("INSERT INTO anomaly_present(filepath, no_of_anomaly)"
                       "VALUES (%s, %s)")
        data = (file_path,len(anomalies_2))
        database_connection.execute(insert_stmt, data)
        for index, row in anomalies_2.iterrows():
            database_connection.execute("INSERT INTO anomalies (filepath,anomalies) VALUES (%s,%s)", (file_path,str(row.to_dict()),))
        cnx.commit()
```

In [15]:

```
def monitor_folder(folder_path, database_connection):
# Function to continuously monitor a folder for new files
    while True:
        for file in os.listdir(folder_path):             # list all the files in the directory
            file_path = os.path.join(folder_path, file)  # join the folder path with the file name
            if os.path.isfile(file_path):                # if theres a file then do the following
                detect_anomalies(file_path, database_connection)
        time.sleep(60)
```

In [40]:

```
if __name__ == "__main__":
    folder_path = r'C:\Users\91775\Desktop\Grahnumb_assignment\data'
    database_connection = cnx.cursor()
    monitor_folder(folder_path, database_connection)
```