PositionEmailBatch class

global class PositionEmailBatch implements Database.Batchable<sObject>{

global Database.QueryLocator start(Database.BatchableContext bc)

{

return Database.getQueryLocator('select Position\_\_c.OwnerId, Status\_\_c, Position\_owner\_Email\_\_c from Position\_\_c where Status\_\_c=\'Open\' and CreatedDate != LAST\_N\_DAYS:2');

}

global void execute(Database.BatchableContext bc, List<Position\_\_c> records)

{

for(Position\_\_c pos:records)

{

// test=[select Position\_owner\_Email\_\_c from pos];

string address=pos.Position\_owner\_Email\_\_c;

EmailManager.sendMail(address, 'Position is opened', 'Dear user your position request is open for more than 2 days');

}

}

global void finish(Database.BatchableContext bc)

{

AsyncApexJob job = [SELECT Id, Status, NumberOfErrors,

JobItemsProcessed,

TotalJobItems, CreatedBy.Email

FROM AsyncApexJob

WHERE Id = :bc.getJobId()];

// EmailUtils.sendMessage(job, owner);

}

}

PositionEmailSchedule class

global class PostionEmailSchedule implements Schedulable{

global void execute(System.SchedulableContext sc)

{

PositionEmailBatch pe= new PositionEmailBatch();

database.executeBatch(pe);

}

}

EmailManager class

public class EmailManager {

// Public method

public static void sendMail(String address, String subject, String body) {

// Create an email message object

Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();

String[] toAddresses = new String[] {address};

mail.setToAddresses(toAddresses);

mail.setSubject(subject);

mail.setPlainTextBody(body);

// Pass this email message to the built-in sendEmail method

// of the Messaging class

Messaging.SendEmailResult[] results = Messaging.sendEmail(

new Messaging.SingleEmailMessage[] { mail });

// Call a helper method to inspect the returned results

inspectResults(results);

}

// Helper method

private static Boolean inspectResults(Messaging.SendEmailResult[] results) {

Boolean sendResult = true;

// sendEmail returns an array of result objects.

// Iterate through the list to inspect results.

// In this class, the methods send only one email,

// so we should have only one result.

for (Messaging.SendEmailResult res : results) {

if (res.isSuccess()) {

System.debug('Email sent successfully');

}

else {

sendResult = false;

System.debug('The following errors occurred: ' + res.getErrors());

}

}

return sendResult;

}

}