This cloud function aims to get the file from gcs and then modify it, then finally send it to was s3 bucket. This process is automatic as soon as in the certain gcs we get a hew file (set it in trigger)

main.py

```
import pandas as pd
import gcsfs
import csv
import google.auth
import numpy as np
from datetime import datetime
import boto3

currentMonth = datetime.now().month
currentYear = datetime.now().year
currentDay = datetime.now().day
ACCESS_KEY = 'XXX'
SECRET_KEY = ‘YYY'
## get the current project
project = google.auth.default()[1]
fs = gcsfs.GCSFileSystem(project=project)

def main(event, context):
    file= event['name']
    file_path="gs://psa-bucket/%s" %file
    csvfile = fs.open(file_path)
    file_modification(csvfile)

def file_modification(file):
    df= pd.read_csv(file,sep='|', dtype=str)
    df_grouped=df[['ADRESSE_LIGNE_1','ADRESSE_LIGNE_2','ADRESSE_LIGNE_3']]
    df['ADDRESS1'] = df_grouped.apply(lambda x: ' '.join(x.dropna().astype(str)),axis=1)
    df_deleted1=df.drop('CIVILITE',axis=1)
    df_deleted2=df_deleted1.drop('ADRESSE_LIGNE_1',axis=1)
    df_deleted3=df_deleted2.drop('ADRESSE_LIGNE_2',axis=1)
```

```
    df_deleted=df_deleted3.drop('ADRESSE_LIGNE_3',axis=1)

df_header_modified=df_deleted.rename(columns={'IDCRM':'CustomerID','NOM':'LASTNAME
','PRENOM':'FIRSTNAME','ADRESSE_LIGNE_4':'ADDRESS4','CODE_POSTAL':'POSTCO
DE','VILLE':'TOWN','EMAIL':'EMAIL1','TEL_PORT':'MOBILE1','TEL_FIXE':'LANDLINE1'}
)
    df_final= df_header_modified[['CustomerID',
'FIRSTNAME','LASTNAME','ADDRESS1','ADDRESS4','TOWN','POSTCODE','EMAIL1','M
OBILE1','LANDLINE1','DSS1','DSS2','DSS3','DSS4','DSS5','DSS6','DSS7','DSS8','DSS9','DSS
10','DSS11','DSS12','DSS13','DSS14','DSS15','DSS16','SEG1','SEG2']]
    csvnew="gs://bucketname/filename_%s%s%s.psv" % (currentDay,currentMonth,currentYear)
    df_final.to_csv(csvnew, sep='|',index = False, header=True)
    CPL(df_final)
    local_file= fs.open(csvnew)
    bucket='bucketname'
    s3_file="filename_%s%s%s.psv" % (currentDay,currentMonth,currentYear)
    upload_to_aws(local_file, bucket,s3_file)

def CPL(dataframe):
    ##select the interested columns and #ofnotnull, %ofnotnull, #ofnotnull, %ofnotzero,
fieldtotal(only for num), fieldaverage(only for num),maxvaluelength, minvaluelength
    ###select the interested columns (e.g TOWN,EMAIL1,MOBILE1,SEG1,SEG2)
    df_selected=dataframe[['TOWN','EMAIL1','MOBILE1','SEG1','SEG2']]

    ###max&min length of value
    measurer = np.vectorize(len)
    maxlength = measurer(df_selected.values.astype(str)).max(axis=0)
    minlength = measurer(df_selected.values.astype(str)).min(axis=0)

    ### #ofnotnull, %ofnotnull
    df_missing = df_selected.isna()
    df_num_missing = df_missing.sum()
    cpl_No_notnull= len(df_selected)-df_num_missing
    cpl_percen_notnull=100.0-(df_num_missing / len(df_selected)*100.0)
    cpl_No_notzero = (df_selected != 0).sum()
```

```
    cpl_No_notzero_notnull= (pd.concat([ cpl_No_notnull, cpl_No_notzero], axis=1).min(axis=1))
    cpl_percen_notzero_notnull=cpl_No_notzero_notnull/len(df_selected)*100.0

    # if necessary, change some columns to numeric,e.g SEG1 and SEG2, to calculate notzero, sum and avg
    num_list = ['SEG1', 'SEG2']
    for element in num_list:
        df_selected[element]=pd.to_numeric(df_selected[element])

    ### fieldtotal and avg (numerical calculation)
    df_selected_num=df_selected[num_list]
    Field_total=df_selected_num.sum()
    Field_average=df_selected_num.mean()

    ###CPL-combine all stats together
    dtype=pd.DataFrame(df_selected.dtypes)
    cpl_No_notzero_notnull=pd.DataFrame(cpl_No_notzero_notnull)
    cpl_percen_notzero_notnull=pd.DataFrame(cpl_percen_notzero_notnull)
    cpl_No_notnull=pd.DataFrame(cpl_No_notnull)
    cpl_percen_notnull=pd.DataFrame(cpl_percen_notnull)
    Field_total=pd.DataFrame(Field_total)
    Field_average=pd.DataFrame(Field_average)
    MaxLen =pd.DataFrame(maxlength,index=dtype.index)
    MinLen =pd.DataFrame(minlength,index=dtype.index)
    result = pd.concat([dtype, cpl_No_notzero_notnull,cpl_percen_notzero_notnull,cpl_No_notnull,cpl_percen_notnull,Field_total,Field_average,MaxLen,MinLen], axis=1,sort=False).reindex(dtype.index)
    result.columns =[ 'dtype','cpl_No_notzero_notnull','cpl_percen_notzero_notnull','cpl_No_notnull','cpl_percen_notnull','Field_total','Field_average','MaxLen','MinLen']
    ###CPL-send to bucket
    cplfile_csv="gs://bucketname/CPL_%s%s%s.csv" % (currentDay,currentMonth,currentYear)
    result.to_csv(cplfile_csv,index = True, header= True)
    #cplfile_html="gs://bucektname/CPL_%s%s%s.html" % (currentDay,currentMonth,currentYear)
```

```
    #result.to_html(cplfile_html)
    #cpl_fig=result.plot(kind='bar').get_figure()
    #cpl_fig.savefig("gs://bucketname/cpl_image.pdf")
    local_cpl_file= fs.open(cplfile_csv)
    bucket='psajinn'
    cpl_file="CPL_csv_%s%s%s.csv" % (currentDay,currentMonth,currentYear)
    upload_to_aws(local_cpl_file, bucket,cpl_file)


def upload_to_aws(local_file, bucket, s3_file):
    s3 = boto3.client('s3',
aws_access_key_id=ACCESS_KEY,aws_secret_access_key=SECRET_KEY)
    s3.upload_fileobj(local_file, bucket,s3_file)
```

requirements.txt

```
# Function dependencies, for example:
# package>=version

boto3==1.16.43
pandas==0.25.2
gcsfs==0.6.0
numpy==1.17.3
```