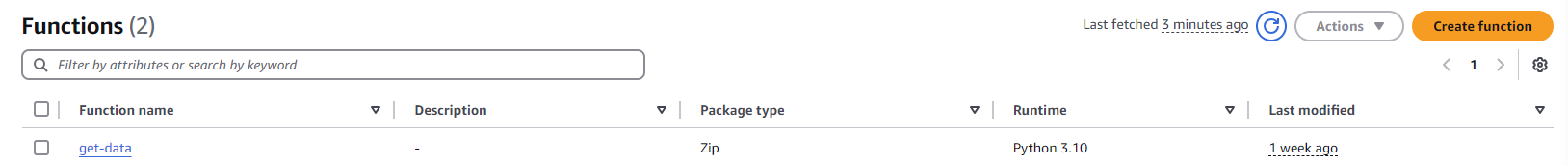
**Step 1: Create a S3 Bucket and upload the dataset into the bucket.**   
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated  
Engagement Overview is the uploaded excel file

**Step 2: Search for Lambda service in AWS management Console and create a lambda function.**   
This function will read the data from S3 bucket, clean and transform the data. Finally the data is again stored in the same S3 bucket in a different folder, here in the folder processed data.   
  
  
Code snippet -   
import json  
import urllib.parse  
import boto3  
import pandas as pd  
from io import BytesIO   
print('Loading function')  
s3 = boto3.client('s3')  
def lambda\_handler(event, context):  
 bucket = event['Records'][0]['s3']['bucket']['name']  
 key = urllib.parse.unquote\_plus(event['Records'][0]['s3']['object']['key'], encoding='utf-8') try:

# Download the Excel file from S3  
 response = s3.get\_object(Bucket=bucket, Key=key)  
 print("File retrieved successfully!")

# Convert the S3 file content to a file-like object  
 file\_stream = BytesIO(response['Body'].read())  
 # Read the second sheet (sheet index 1) into a DataFrame  
 data = pd.read\_excel(file\_stream, sheet\_name=1, engine='openpyxl')  
 print(f"Data preview before conversion: {data.head()}")  
 # Convert `Visits`, `Unique Visitors`, and `Total Page Views` to numeric values  
 for column in ['Visits', 'Unique Visitors', 'Total Page Views']:  
 if column in data.columns and data[column].dtype == 'object':  
 data[column] = data[column].str.replace(',', '').astype(float)  
 # Convert `Desktop Share` and `Bounce Rate` to decimals  
 for column in ['Desktop Share', 'Bounce Rate']:  
 if column in data.columns and data[column].dtype == 'object':  
 data[column] = data[column].str.rstrip('%').astype(float) / 100  
 # Convert `Avg. Visit Duration` to seconds  
 if 'Avg. Visit Duration' in data.columns:  
 data['Avg. Visit Duration'] = pd.to\_timedelta(data['Avg. Visit Duration']).dt.total\_seconds()  
 # Retain decimal precision for `Pages / Visit`  
 if 'Pages / Visit' in data.columns and data['Pages / Visit'].dtype == 'object':  
 data['Pages / Visit'] = data['Pages / Visit'].astype(float)  
 # Standardize `Date` column format  
 if 'Date' in data.columns:  
 data['Date'] = pd.to\_datetime(data['Date']).dt.strftime('%Y-%m-%d')  
 print(f"Data preview after conversion: {data.head()}")  
 # Save the DataFrame to S3 as a CSV file  
 output\_key = 'processed-data/trend\_analysis.csv'  
 csv\_buffer = BytesIO()  
 data.to\_csv(csv\_buffer, index=False)  
 csv\_buffer.seek(0)  
 s3.put\_object(Bucket=bucket, Key=output\_key, Body=csv\_buffer.getvalue())  
 return {  
 'statusCode': 200,  
 'body': json.dumps({  
 "message": "Data processed and saved to S3 for QuickSight.",  
 "csv\_location": f"s3://{bucket}/{output\_key}"  
 })  
 }  
 except Exception as e:  
 print(e)  
 print(f'Error processing file {key} from bucket {bucket}.')  
 raise e

This code requires numpy and pandas layers/modules.

Test Event –   
{  
 "Records": [  
 {  
 "s3": {  
 "bucket": {  
 "name": "website--data"  
 },  
 "object": {  
 "key": "EngagementOverview-sap.com.xlsx"  
 }  
 }  
 }  
 ]  
}

**Step 3 – Install Numpy, and Pandas libraries and upload zip files in AWS Layers.**A screenshot of a computer

Description automatically generated

After uploading required layers go back to the get\_data lambda function and click on Add Layers to add the created layers.

**Step 4 – Setup AWS Quicksight. The processed data gets visualized**.   
A screenshot of a computer

Description automatically generated

**Step 5- Setting up alerts on KPI visual.**   
The alerts are triggered when the weekly page views exceed 20Million views. Email notifications are sent to teams.

To set alert, go to the dashboard, select the KPT visual (example KPI, Gauge, Table, and Pivot table visuals), click on the create alert option, edit the threshold value from the create alert dialogue box, and finally click Save.

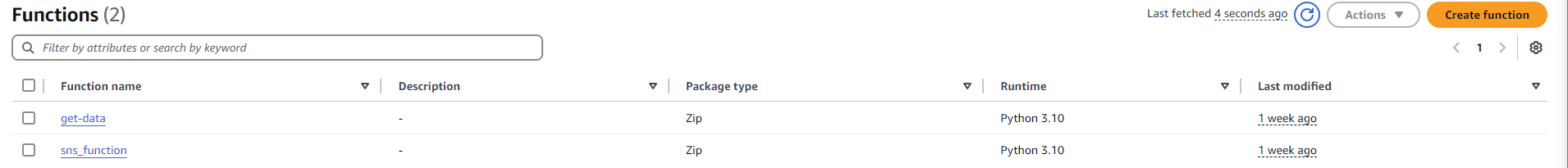
A screenshot of a computer

Description automatically generated

Step 6 – Setting up Simple Notification Service(SNS)  
First we create a topic for the Alerts. Here – Page\_Views\_Alerts is created.

A white object with a black line

Description automatically generated with medium confidence

In the topic we set up subscriptions or email id of person who should receive alerts/notifications.   
Since we are not using real-time data, we create a another lambda function to trigger alerts.   
  
Code snippet –   
import json  
import urllib.parse  
import boto3  
import pandas as pd  
from io import BytesIO  
print('SNS Notification Lambda function loaded')  
s3 = boto3.client('s3')  
sns = boto3.client('sns')  
# SNS topic ARN  
sns\_topic\_arn = 'arn:aws:sns:us-east-1:054037136070:Page\_Views\_Alert'  
def lambda\_handler(event, context):  
 bucket = event['Records'][0]['s3']['bucket']['name']  
 key = urllib.parse.unquote\_plus(event['Records'][0]['s3']['object']['key'], encoding='utf-8')   
 try:  
 # Download the file from S3  
 response = s3.get\_object(Bucket=bucket, Key=key)  
 print("File retrieved successfully!")  
 # Convert the S3 file content to a file-like object  
 file\_stream = BytesIO(response['Body'].read())  
 # Read the csv file  
 data = pd.read\_csv(file\_stream)  
 print(f"Data preview before processing: {data.head()}")  
 # Convert `Total Page Views` to numeric values   
 if 'Total Page Views' in data.columns and data['Total Page Views'].dtype == 'object':  
 data['Total Page Views'] = data['Total Page Views'].str.replace(',', '').astype(float)  
 # Convert `Date` column to datetime  
 if 'Date' in data.columns:  
 data['Date'] = pd.to\_datetime(data['Date'])  
 # Group data by week (using ISO week number)  
 data['Week'] = data['Date'].dt.isocalendar().week  
 weekly\_views = data.groupby('Week')['Total Page Views'].sum().reset\_index()  
 print(f"Weekly Views Data: {weekly\_views}")  
 # Check if any week exceeds 20 million views  
 alert\_triggered = False  
 for index, row in weekly\_views.iterrows():  
 if row['Total Page Views'] > 20000000: # 20 Million views threshold  
 alert\_triggered = True  
 # Get the start and end dates for the week  
 week\_start = data[data['Week'] == row['Week']]['Date'].min()  
 week\_end = data[data['Week'] == row['Week']]['Date'].max()  
 # Format the date range as MM/DD - MM/DD  
 date\_range = f"{week\_start.strftime('%m/%d')} - {week\_end.strftime('%m/%d')}"  
 # Convert views to millions  
 views\_in\_millions = row['Total Page Views'] / 1\_000\_000  
 # Create the message with the new format  
 message = f"Alert: Week {row['Week']} ({date\_range}) has exceeded 20 million views. Total views are {views\_in\_millions:.2f} million!"  
 print(message)  
 # Send SNS Notification  
 sns.publish(  
 TopicArn=sns\_topic\_arn,  
 Message=message,  
 Subject="Weekly Page View Alert"  
 )  
 if not alert\_triggered:  
 print("No breaches found. No notifications sent.")  
 return {  
 'statusCode': 200,  
 'body': json.dumps({  
 "message": "SNS notifications processed successfully."  
 })  
 }  
 except Exception as e:  
 print(e)  
 print(f'Error processing file {key} from bucket {bucket}.')  
 raise e

Test Event -   
{  
 "Records": [  
 {  
 "s3": {  
 "bucket": {  
 "name": "website--data"  
 },  
 "object": {  
 "key": "processed-data/trend\_analysis.csv"  
 }  
 }  
 }  
 ]  
}

Finally a snapshot of how alerts looks like when received in email.

A close-up of a white background

Description automatically generated **A white background with black text

Description automatically generated**