**Stream Processing and Analytics**

**Assignment-2 (Pizzario Delivery Chain)**

**Prepared and Executed by – Group 079**

|  |  |
| --- | --- |
| 2020FC04101 | AMAL SHAJI |
| 2020FC04102 | GNANESWAR S V VIJJAPU |
| 2020FC04103 | UMANG MISHRA |

**TABLE OF CONTENTS**

[1 Scope, Purpose, Audience 3](#_Toc96586121)

[2 Problem Statement 3](#_Toc96586122)

[3 Customer Segmentation Approach – Python Program 3](#_Toc96586123)

[3.1 Approach 3](#_Toc96586124)

[3.2 Python Program 5](#_Toc96586125)

[4 Customer Base – Cluster Nomenclature 6](#_Toc96586126)

[4.1 Cluster 1 – Above Average Income & Less Spend Group 6](#_Toc96586127)

[4.2 Cluster 2 – Less Income & Average Spend Group 7](#_Toc96586129)

[4.3 Cluster 3 – Average Income & Average Spend Group (Young Middle Age) 7](#_Toc96586130)

[4.4 Cluster 4 – Low Income & High Spend Group 8](#_Toc96586131)

[4.5 Cluster 5 – Average Income &Above Average Spend Group (Older Age) 8](#_Toc96586132)

[4.6 Cluster 6 – High Income & High Spend Group 9](#_Toc96586133)

[5 Customer Movement Simulation 10](#_Toc96586134)

[5.1 Approach 10](#_Toc96586135)

[5.2 Python Program 10](#_Toc96586136)

[6 Streaming Data Pipeline Architecture 12](#_Toc96586137)

[6.1 Architecture 12](#_Toc96586138)

[6.2 Step-by-Step Approach 12](#_Toc96586139)

[6.3 Design Consideration & Coding 13](#_Toc96586140)

[6.3.1 S3 Configuration 13](#_Toc96586141)

[7 Demo 14](#_Toc96586142)

# Scope, Purpose, Audience

The purpose of this document is to provide the information needed to understand the approach, high level design solution and developmentintegration that were considered while solving the Assignment 2– Pizzario Delivery Chain for Streaming Data Analytics. The primary audience of this document is the members of BITS Pilani WILP who evaluate and assess this problem statement.

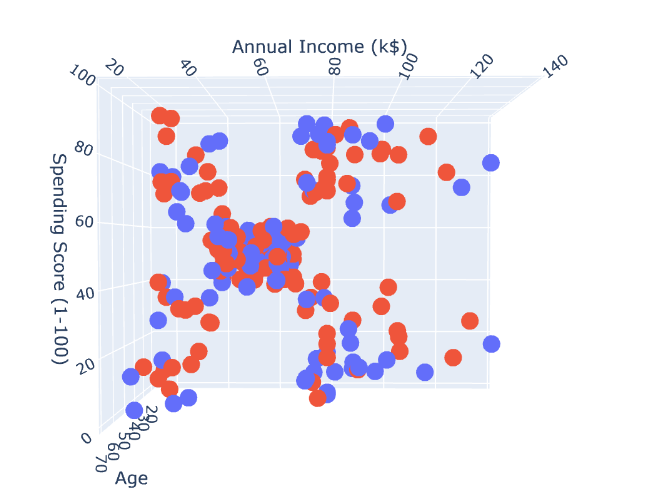
# Problem Statement

Assume that you are working as analyst for “Pizzario”, a pizza delivery chain. The group has collected some interesting characteristics of customers who had purchased their pizza earlier. (Refer the attached pizza\_customers.csv file for the same). The marketing team is planning a campaign to increase the sales of a newlylaunched pizza. Before that they want to analyze the segmentation of existing customers so that they can have the clearer picture about the customer categories.

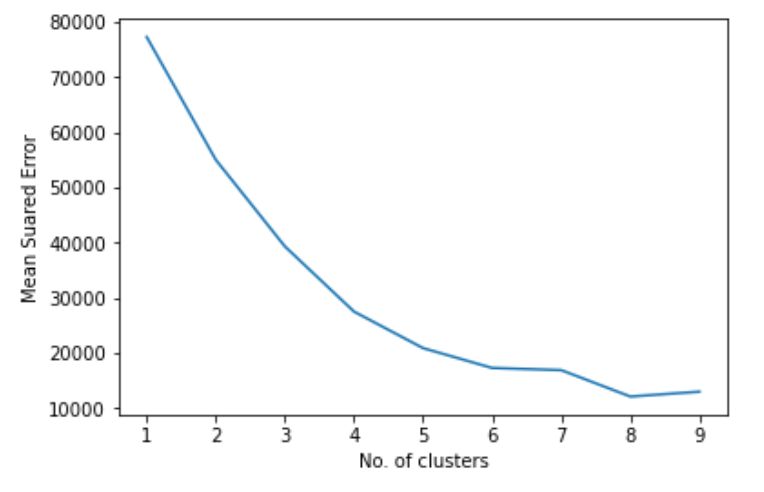
# Customer Segmentation Approach – Python Program

## Approach

* For the given pizza\_customer.csv file, we loaded the CSV into the data-frame and using the plotly scatter plot with x-axis as Age, y-axis as Annual Income and z-axis as Spending Score and differentiated the gender with color. (For evaluation purpose, we need the evaluator to install plotly library for user friendly navigation of the plotly graph – **pip install plotly**)



* As we observe that in each of these above clusters, we see gender play a minor role as the data is distributed among all the clusters. For easy reference, we label encoded the gender column using get dummies function of pandas (1,0).
* Using cluster algorithm custom implementation, we have the below methods written –
  + ***def k\_means(self k:int)🡪 list*** : Function used to find the clusters in the dataframe where k is the number of clusters needed and the function returns multidimensional array of clusters.
  + ***def finding\_k(self) 🡪 None:***Function is used to find the number of clusters with cluster number as key and mean square error as values to draw the graph between both in order to find the elbow point which will help to decide the number of clusters.
* We have plotted the graph between number of clusters takenin x-axis and mean square error due to that setup in y-axis,



* A general rule of thumb is to take the elbow of this curve as the optimum number of clusters. Hence 6 is taken as the number of clusters for the k-means.

## Python Program

Below is the snippets of the jupyter notebook containing the custom implementation of the customer segmentation. *(the actual notebook will be attached while submitting the assignment)*

|  |
| --- |
|  |
|  |
|  |

The output of this python program will be a CSV file with the corresponding customer ID and the Cluster ID that customer belongs to.

|  |  |
| --- | --- |
|  |  |

# Customer Base – Cluster Nomenclature

## Cluster 1 – Above Average Income & Less Spend Group

## As per Cluster 1, we have 35 total entries ranging from age group – 19 to 59. We can consider this group contains majority of working-class group who don’t spend more (Spending Mean – 17.28). However, they are potential buyers and has a capacity to spend. If we give better offers, they should be getting attracted and can improve the purchase power. We treat them as Above Average Income and Less Spend Group.

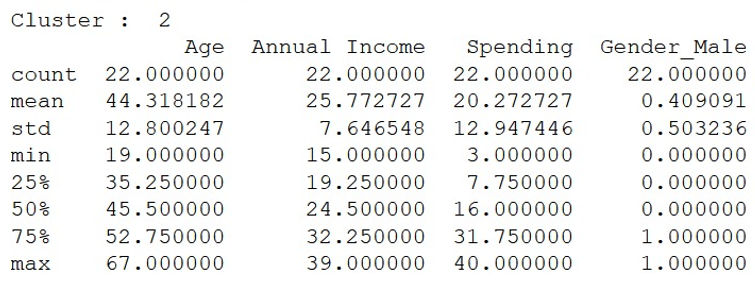
Proposed Offer: -**BUY1GET1**

## 

## Cluster 2 – Less Income & Average Spend Group

As per Cluster 2, having 22 total entries ranging from age group – 19 to 67. This group can be considered in the similar age group like Cluster 1 but they are moderately exposed to spending. Even though when compared with Cluster 1, their earnings are less, they are better buyers. However, we assume their spending at maximum can equalize up to their income we should focus on retaining them as they are better buyers. We treat them as **Less Income and Average Spend** Group.

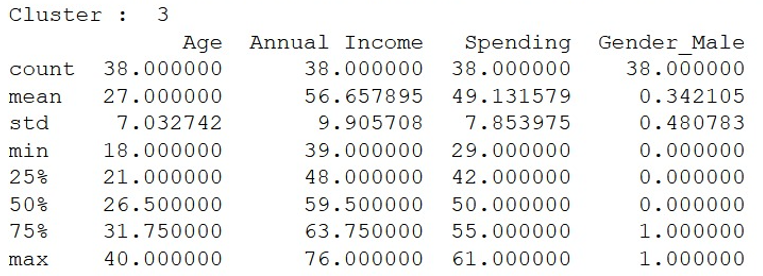
Proposed Offer: -**GET10%OFF**



## Cluster 3 – Average Income & Average Spend Group (Young Middle Age)

As per Cluster 3, having 38 entries has majority of middle young age group ranging from 18-40. This group falls under average buying capacity compared with other clusters. They don’t need any great offers, as they frequent pizza customer base and will be easily attracted to any offers for young age group. We treat this group as **Average Income andAverage Spend** Group.

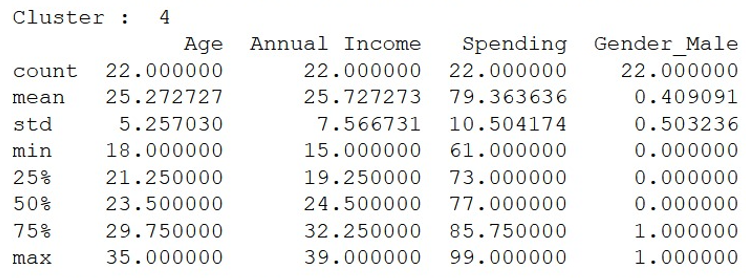
Proposed Offer: -**GET10%ON1\_GET20%ON2**



## Cluster 4 – Low Income & High Spend Group

As per Cluster 4, having 22 entries has majority of them are below 30 years ranging from age group – 18 to 35. This group falls under very high purchasing power, but they don’t have good income (income similar to Cluster 2). So, we will consider as they will visit the store with minimum offers and deals to buy pizza. We treat this group as **Low Income and High Spend** Group

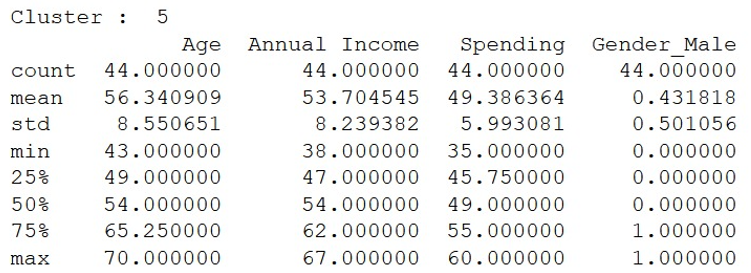
Proposed Offer: -**GET10%OFF**



## Cluster 5 – Average Income &Above Average Spend Group (Older Age)

As per Cluster 5, having 44 entries has majority of the population 50+ years ranging from 43 to 70 years. This group falls under moderately average purchasing power similar to cluster 3. If good deals are offered to them, they may buy more and hence can attract more customers in this group (more customer base is in this cluster compared with others). We treat this group as **Average Income and Above Average Spend** Group.

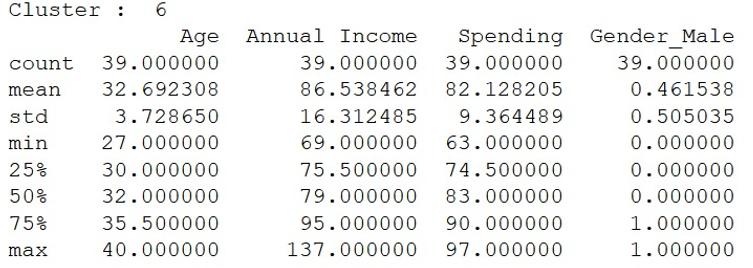
Proposed Offer: -**GET15%OFF**



## Cluster 6 – High Income & High Spend Group

As per Cluster 6, having 39 entries has age group between 27 to 40 years with maximum population below 36 years. This group can be treated as middle age family group which are more excited to spend with their high income possess high expenditure. So they must be enforced or enticed to buying more since they can afford. We treat this group as **High Income and High Spend** Group.

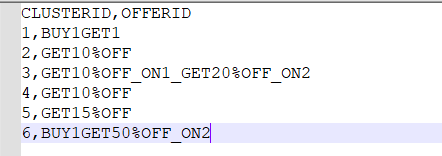
Proposed Offer: -**BUY1GET50%OFF\_ON2**



Below is the consolidated offer list that will be further used for each customer based on their cluster classification.

|  |  |
| --- | --- |
| **CLUSTERID** | **OFFER** |
| 1 | BUY1GET1 |
| 2 | GET10%OFF |
| 3 | GET10%OFF\_ON1\_GET20%OFF\_ON2 |
| 4 | GET10%OFF |
| 5 | GET15%OFF |
| 6 | BUY1GET50%OFF\_ON2 |

Snippet of the offer.csv file that will be used as a lookup to find which customer is eligible for which offer based on the cluster ID –



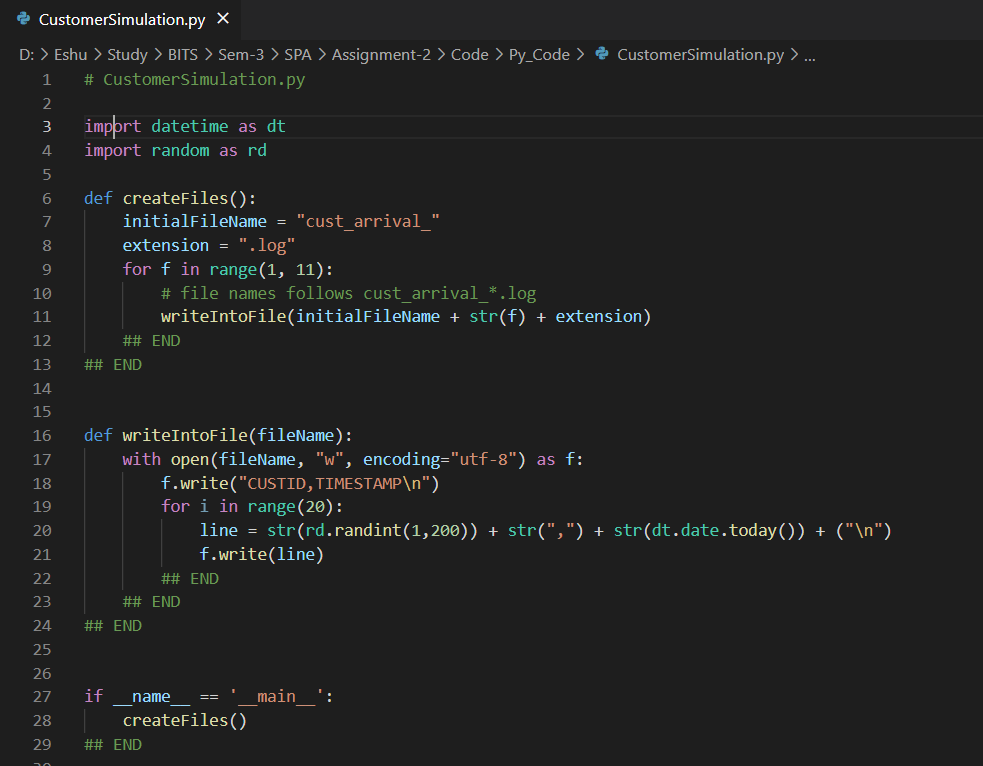
# Customer Movement Simulation

## Approach

* This python program randomly generates multiple log files which captures the Customer ID and the Timestamp when the customer entered the mall premises.
* Below are the methods defined for the customer movement simulation.
  + ***def createFile() :***This method is used the generate multiple files (10 files) into the same location with file naming convention as cust\_arrival\_\*.log
  + ***def writeIntoFile(filename) :*** This method is used to generate customers randomly and with the corresponding date when the customer is generated. Each file is limited to 20 customers between the range of 1 to 200.

## Python Program

Below is the Python snippet code that is used to generate the customer movement simulation. *(the actual python code will be attached while submitting the assignment)*

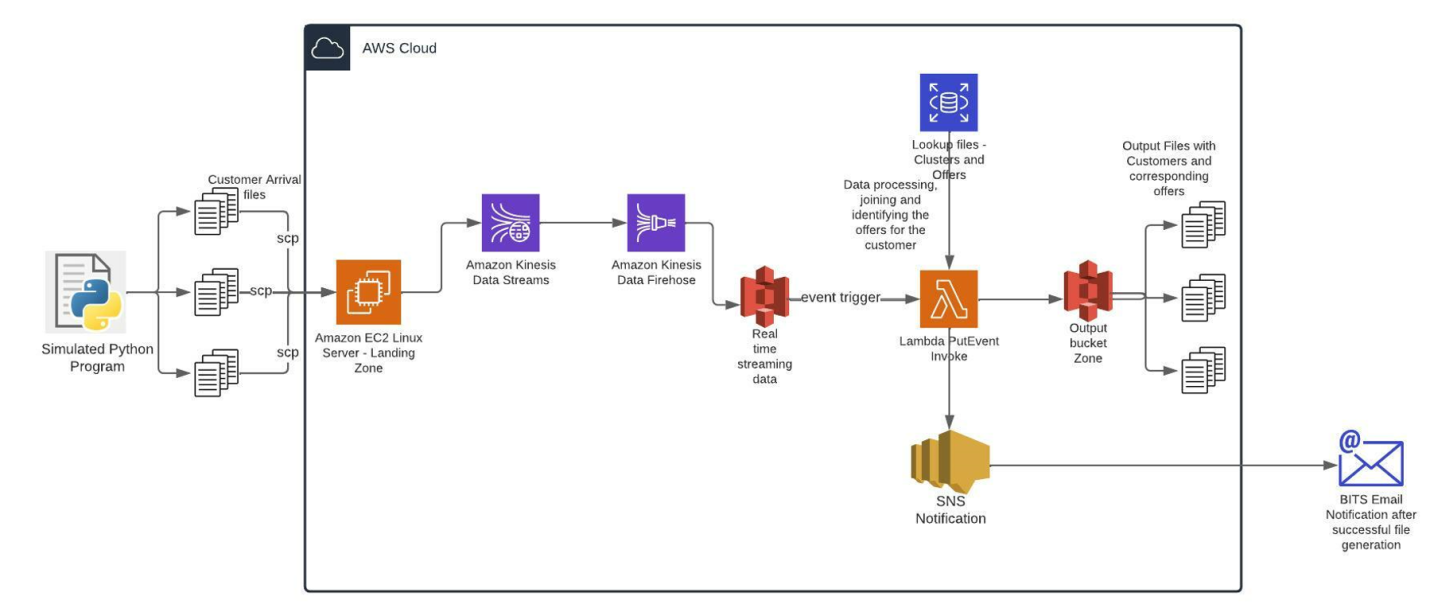


The outcome of this python program is the log files generated in CSV format with customer and the timestamp details.

|  |  |
| --- | --- |
|  |  |

# Streaming Data Pipeline Architecture

## Architecture



## Step-by-Step Approach

* As part of exercise 1 and 2, we have generated the customer\_file.csv (containing customer ID and Cluster ID) and offer.csv file (Cluster ID and Offer) that will be stored in the AWS instance as lookup files.
* An Amazon EC2 AMI Linux instance has been created which is used as a Landing Zone for the files to arrival in AWS.
* The EC2 instance is configured to run a Kinesis agent using the command – “sudo service aws-kinesis-agent start”
* Once this agent started OK, we will be using a Simulated Python program that will transfer (SCP) files into this AWS EC2 instance.
* The AWS Kinesis agent will listen to the simulated files that are being pooled to the landing zone.
* AWS Kinesis Firehose is configured to read the data from the AWS Kinesis data stream as a source and loaded into the AWS S3 bucket which stores all the simulated files in the real time streaming data with a minimal latency.
* AWS Lambda trigger is configured to pool any PutObject in this S3 bucket and executes the core AWS Lambda function that will identify offer related to each customer that is being sent by the real time streaming data from the Kinesis datastream.
* The output of this AWS Lambda function is stored in a Output Bucket Zone in AWS S3 which can be further utilized for sending offer messages to the corresponding users.
* An Amazon SNS notification is used to send a notification for the subscribers to their corresponding BITS accounts after the successful execution of the Lambda function.

## Design Consideration & Coding

## S3 Configuration

**arn:aws:s3:::kinesis-demo-gv** : This is the S3 bucket where AWS Kinesis Firehose will process the real time data that is coming from AWS Kinesis Data Stream into the Landing Zone in AWS S3. The S3 bucket is also configured to an PutObject Event to trigger a Lambda function whenever there is any upload that happens in this S3 bucket.

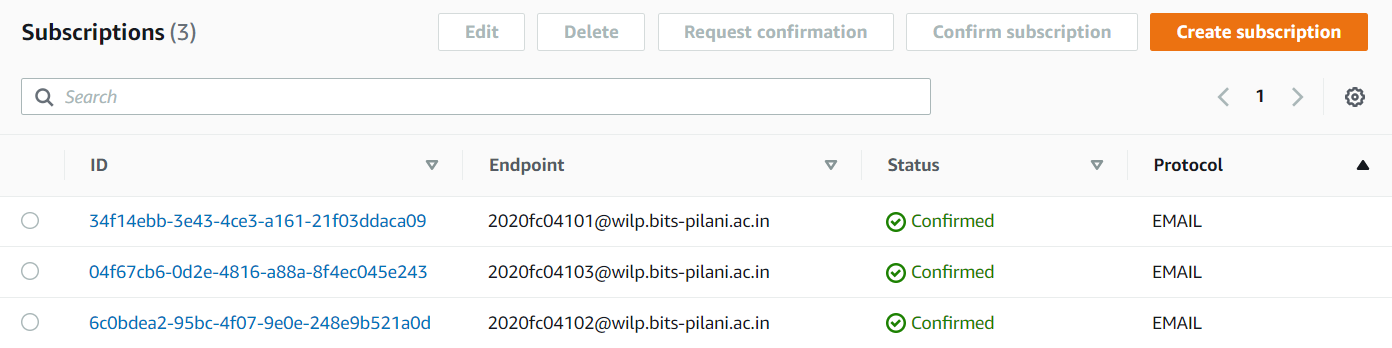
**arn:aws:s3:::customer-offers-zone** : This is the S3 bucket for the target output location of the lambda function places the output files with customers and the corresponding offers that the customer is eligible.

**arn:aws:lambda:ap-south-1:339298396291:function:bits\_spa\_generate\_customer\_offers\_lambda :**

This lambda function joins with the customer\_arrival\_\*.log files that are loaded into the landing zone with other lookup files – offer.csv and customer\_file.csv and generate the output file containing the customer and the offer information. This lambda once successfully completed will send an SNS notification to the BITS accounts.

**arn:aws:sns:ap-south-1:339298396291:kinesis-spa**

This is the SNS notification service that is configured to send emails after successful completion of the lambda job to the 3 subscriptions –



**arn:aws:sns:ap-south-1:339298396291:kinesis-demo**

This is the Kinesis data stream service which pools the EC2 instance files in a real time data streaming through the kinesis agent and delivery log streams to the destination

**arn:aws:sns:ap-south-1:339298396291:kinesis-firehose-demo**

This Kinesis Firehose streaming service will source the data from the Kinesis data stream – kinesis-demo and delivers to the destination – AWS S3 Landing Zone for further processing.

# Demo

Below is the link for the mp4 file used for the demo. It demonstrates the end to end real time data streaming using AWS Kinesis Data Stream and Kinesis Firehose integrated with other AWS components (like Lambda, SNS and AWS S3).

Google Drive Link - <https://drive.google.com/file/d/1q_pZStdF0LGaM6jYiuTKu_csLNHus0Ch/view?usp=sharing>