Smart\_Ops Task

Accelerator (Python)

13th December 2023

Table of Contents

[Introduction 4](#_Toc90309638)

[Smart Ops 4](#_Toc90309639)

[Algorithm 6](#_Toc90309640)

[Abstract of Accelerator 6](#_Toc90309641)

[Flow Chart 7](#_Toc90309642)

[Regular Expression 8](#_Toc90309643)

[Pattern’s Info 9](#_Toc90309644)

[Code Template 11](#_Toc90309645)

[Input Files 15](#_Toc90309646)

[Code with Explanation 16](#_Toc90309647)

List of Images

[Figure 1.Smart Ops 4](#_Toc90309664)

[Figure 2.Revision History 5](#_Toc90309665)

[Figure 3.Pattern1 9](#_Toc90309666)

[Figure 4.Pattern2 9](#_Toc90309667)

[Figure 5.Pattern3 10](#_Toc90309668)

[Figure 6.Pattern 4 10](#_Toc90309669)

[Figure 7.Sample Input Data 15](#_Toc90309670)

[Figure 8.KeyValuePair Details 15](#_Toc90309671)

# Introduction

The Internet of things (IoT) describes physical objects (or groups of such objects) that are embedded with sensors, processing ability, software, and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks. Increasingly, organizations in a variety of industries are using IoT to operate more efficiently, better understand customers to deliver enhanced customer service, improve decision-making and increase the value of the business.Generally, IoT is most abundant in manufacturing, transportation and utility organizations, making use of sensors and other IoT devices; however, it has also found use cases for organizations within the agriculture, infrastructure and home automation industries, leading some organizations toward digital transformation.

An IoT ecosystem consists of web-enabled smart devices that use embedded systems, such as processors, sensors and communication hardware, to collect, send and act on data they acquire from their environments. IoT devices share the sensor data they collect by connecting to an IoT gateway or other edge device where data is either sent to the cloud to be analyzed or analyzed locally. Sometimes, these devices communicate with other related devices and act on the information they get from one another. The devices do most of the work without human intervention, although people can interact with the devices -- for instance, to set them up, give them instructions or access the data.

## Smart Ops

SmartOps is an intelligent process automation project that delivers end-to-end business outcomes using low-code solution.

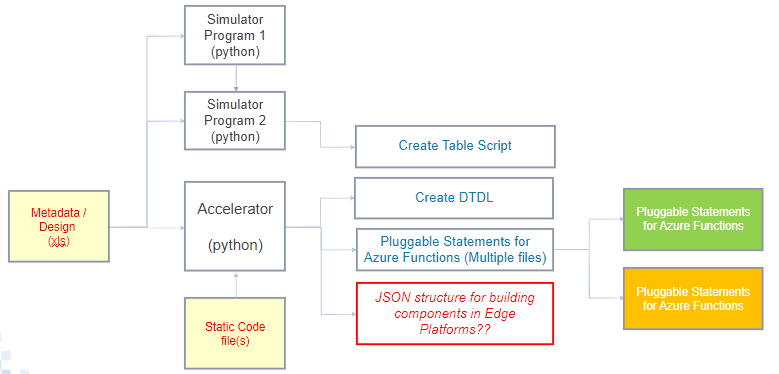


Figure .Smart Ops

**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Effective Date (DD/MM/YYYY) | Phase | Brief Description of Change | Comments |
| 1. | 02/11/2023 | Project initiation  phase | Generating the one output using code template | New functionality added, Pattern Matchings |
| 2. | 15/11/2023 | Project- mid phase | Preprocessing the excel sheet to enhance performance | Convert excel sheet to dataframe, replace default values, reduce repeated tasks |
| 3. | 25/11/2023 | Project- mid phase | Adding new functionality namely Regular Expression to match the pattern | Improving the Regular Expressions Pattern to enhance for better Performance |
| 4. | 03/12/2023 | Project- end phase | Replaced all Contents from excel with the help of Code Template | All functionality attained. Redundant conditional statements and loops avoided. Version of better performance obtained |

Figure .Revision History

# Algorithm

1. Inserting Pandas and Time and Regular Expression modules
2. Accessing The Excel Sheet from Local PC and Reading the Code Template
3. Declaring the Pattern Logically using re.compile method
4. Reading a line and Splitting the line into word by word
5. Iterating the pattern’s and checking for the conditions and replace the contents with repect to code template
6. After, acheving the condition’s writing down the contents into the file
7. Closing the reading as well as writing files to avoid exceptions
8. Calculating the run time of a code

## Abstract of Accelerator

The purpose of Accelerator task to generate a C# code for given code template. The generated output file. are meaningful and logically correct. The input for the accelerator is an excel sheet, which provides the metadata . The excel sheet hold the defined set of variables and a set of data types that can used by the accelerator for generating meaningful code. These generated code will help in providing a deeper understanding on the need, to improve productivity and drive growth and revenue.

# Flow Chart

Checking Patterns

END

Output File

Throw An Pattern Error and Pass the condition

Replace The Contents

Input, Excel File &

Code Template

YES NO

# Regular Expression

A regular expression (sometimes called a rational expression) is **a sequence of characters that define a search pattern**, mainly for use in pattern matching with strings, or string matching, i.e. “find and replace”-like operations. Regular expressions are a generalized way to match patterns with sequences of characters.

**(...)**

Matches whatever regular expression is inside the parentheses, and indicates the start and end of a group; the contents of a group can be retrieved after a match has been performed

\w

**For Unicode (str) patterns:**

Matches Unicode word characters; this includes most characters that can be part of a word in any language, as well as numbers and the underscore. If the [ASCII](https://docs.python.org/3/library/re.html#re.ASCII) flag is used, only [a-zA-Z0-9\_] is matched.

**re.finditer(*pattern*, *string*, *flags=0*)**

Return an [iterator](https://docs.python.org/3/glossary.html#term-iterator) yielding [match objects](https://docs.python.org/3/library/re.html#match-objects) over all non-overlapping matches for the RE *pattern* in *string*. The *string* is scanned left-to-right, and matches are returned in the order found. Empty matches are included in the result.

+

Causes the resulting RE to match 1 or more repetitions of the preceding RE. ab+ will match ‘a’ followed by any non-zero number of ‘b’s; it will not match just ‘a’.

\d

**For Unicode (str) patterns:**

Matches any Unicode decimal digit (that is, any character in Unicode character category [Nd]). This includes [0-9], and also many other digit characters. If the [ASCII](https://docs.python.org/3/library/re.html#re.ASCII) flag is used only [0-9] is matched.

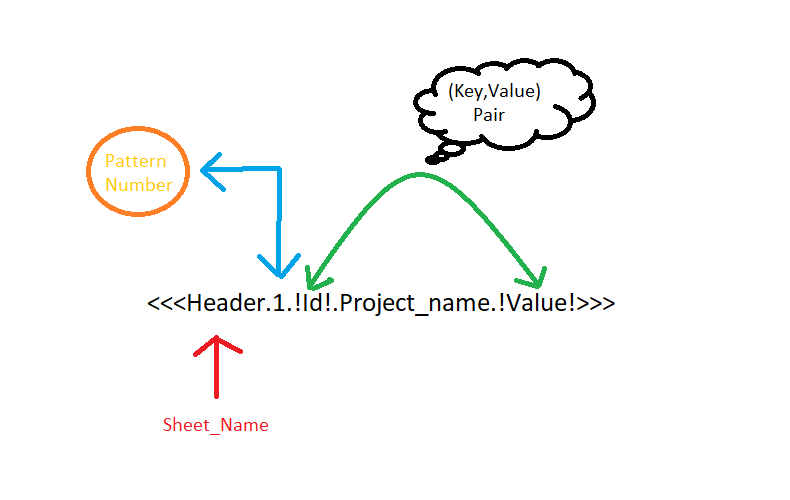
Pattern’s Info****

Figure .Pattern1

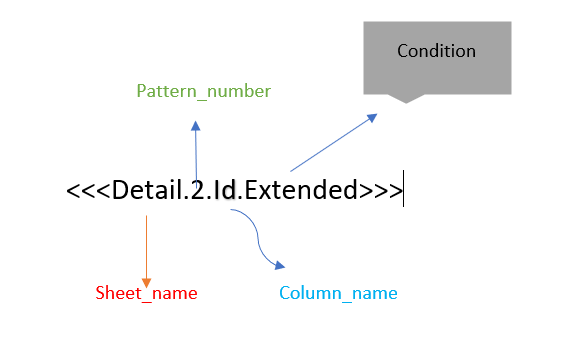
****

Figure .Pattern2

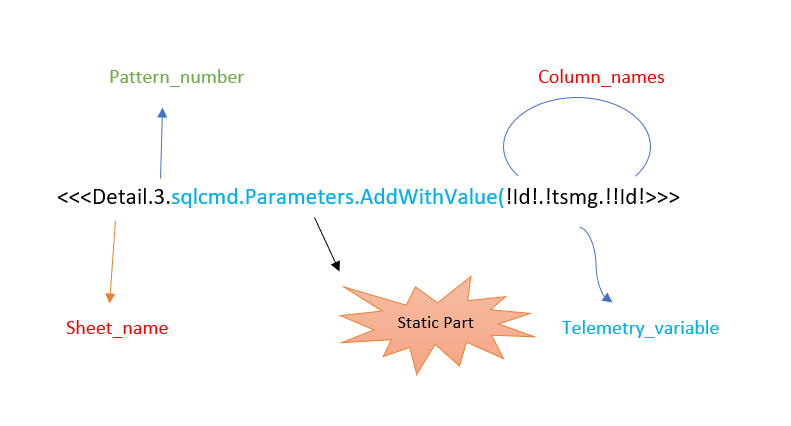
****

Figure .Pattern3

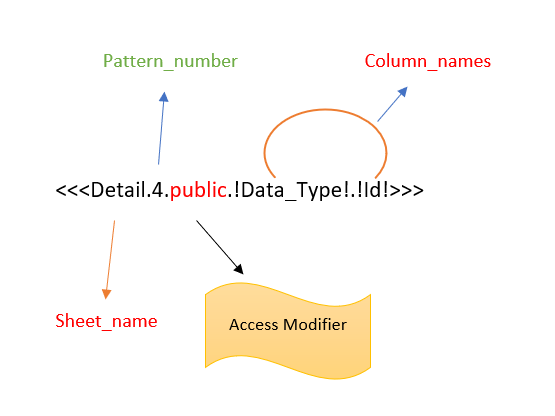
****

Figure .Pattern 4

# Code Template

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

/\*

Project Name : <<<Header.1.!Id!.Project\_name.!Value!>>>

Task : Reading data From IoT Hub to FunctionApp

Developed on : <<<Header.1.!Id!.Developed\_on.!Value!>>>

Developed by : <<<Header.1.!Id!.Developed\_by.!Value!>>>

Version : <<<Header.1.!Id!.Version.!Value!>>>

Program Language : C#

Reviewed by : <<<Header.1.!Id!.Reviewed\_name.!Value!>>> <<<Header.1.!Id!.Reviewed\_date.!Value!>>>

\*/

using IoTHubTrigger = Microsoft.Azure.WebJobs.EventHubTriggerAttribute;

using System.Data;

//using System.Data.SqlClient;

//using Microsoft.Data.SqlClient;

using Microsoft.Azure.WebJobs;

using Microsoft.Azure.WebJobs.Host;

using Microsoft.Azure.EventHubs;

using System.Text;

using System.Net.Http;

using Microsoft.Extensions.Logging;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using Microsoft.Azure.Services.AppAuthentication;

using System.Security;

using Microsoft.IdentityModel.Clients.ActiveDirectory;

using Microsoft.Data.SqlClient;

namespace <<<Header.1.!Id!.NameSpace.!Value!>>>

{

public static class <<<Header.1.!Id!.ClassName.!Value!>>>

{

private static HttpClient client = new HttpClient();

[FunctionName(" <<<Header.1.!Id!.ClassName.!Value!>>> ")]

public static void Run([IoTHubTrigger(" <<<Header.1.!Id!.IOT\_Hubname.!Value!>>> ", Connection = " <<<Header.1.!Id!.IOT\_Connection.!Value!>>> ")] EventData message, ILogger log)

{

// Read IoT hub receiving data

log.LogInformation($"C# IoT Hub trigger function processed a message: {Encoding.UTF8.GetString(message.Body.Array)}");

var deviceid = message.SystemProperties["iothub-connection-device-id"].ToString();

try

{

//parsing the input json data

Telemetry tsmg = JsonConvert.DeserializeObject<Telemetry>(Encoding.UTF8.GetString(message.Body.Array));

// Providing SQL credentials

<<<Header.1.!Id!.SQl\_Connection.!Value!>>>

using (SqlConnection con = new SqlConnection(strcon))

{

con.Open();

if (con.State == ConnectionState.Open)

{

String query = $"INSERT INTO <<<Header.1.!Id!.Table\_Name.!Value!>>> (<<<Detail.2.Id.ALL>>> )" +

$"VALUES (<<<Detail.2.Id.Extended>>> )";

SqlCommand sqlcmd1 = new SqlCommand(query, con);

<<<Detail.3.sqlcmd.Parameters.AddWithValue(!Id!.!tsmg.!!Id!>>>

// Need to add some more code//

}

con.Close();

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DONE\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

}

catch (Exception e)

{

log.LogInformation(e.Message);

}

}

}

// input parameters maping to sql db from input data

public class Telemetry

{

<<<Detail.4.public.!Data\_Type!.!Id!>>>

}

}

Reviewed by : <<<Header.1.!Id!.Reviewed\_name.!Value!>>>

## Input Files

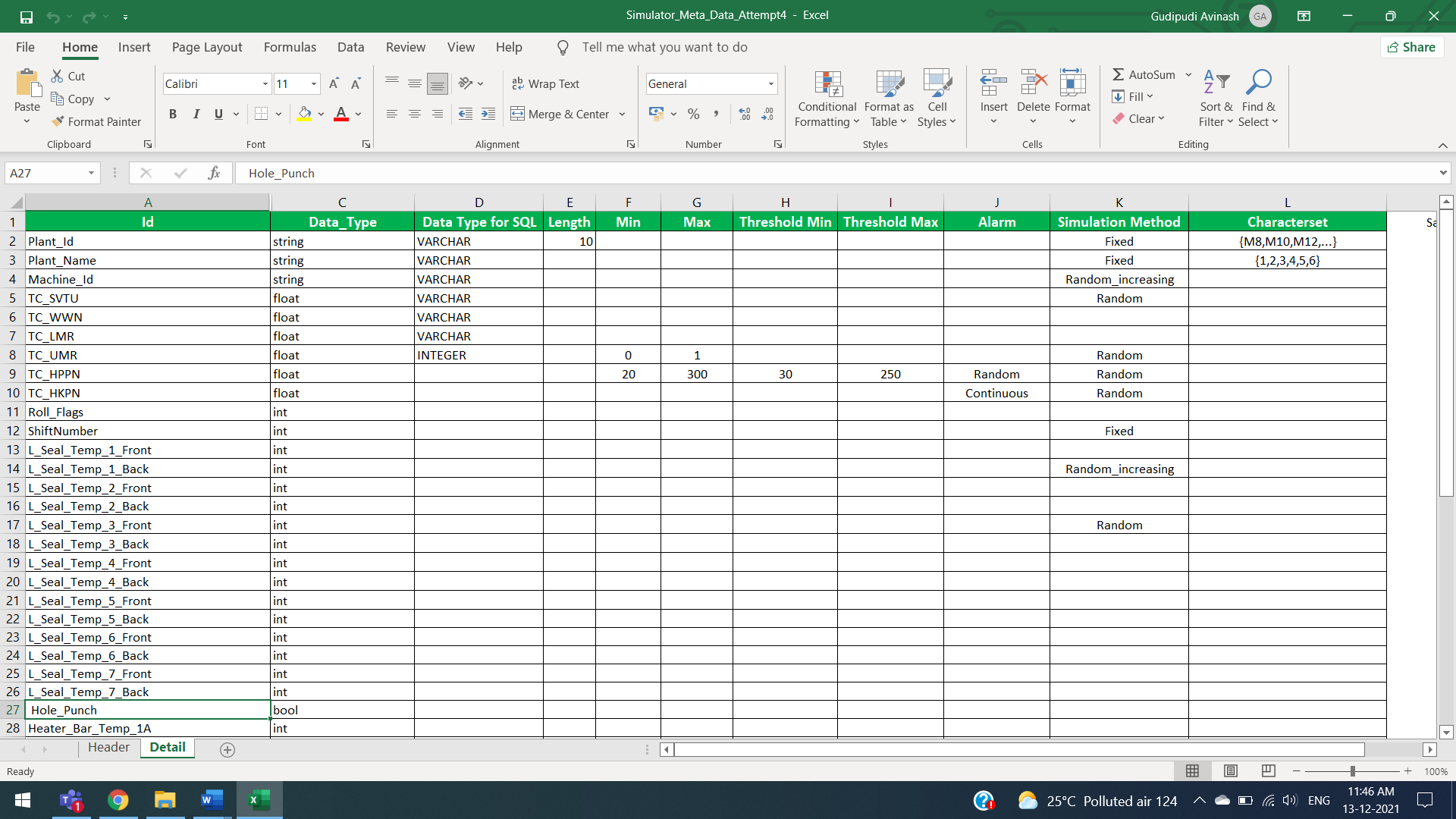


Figure .Sample Input Data

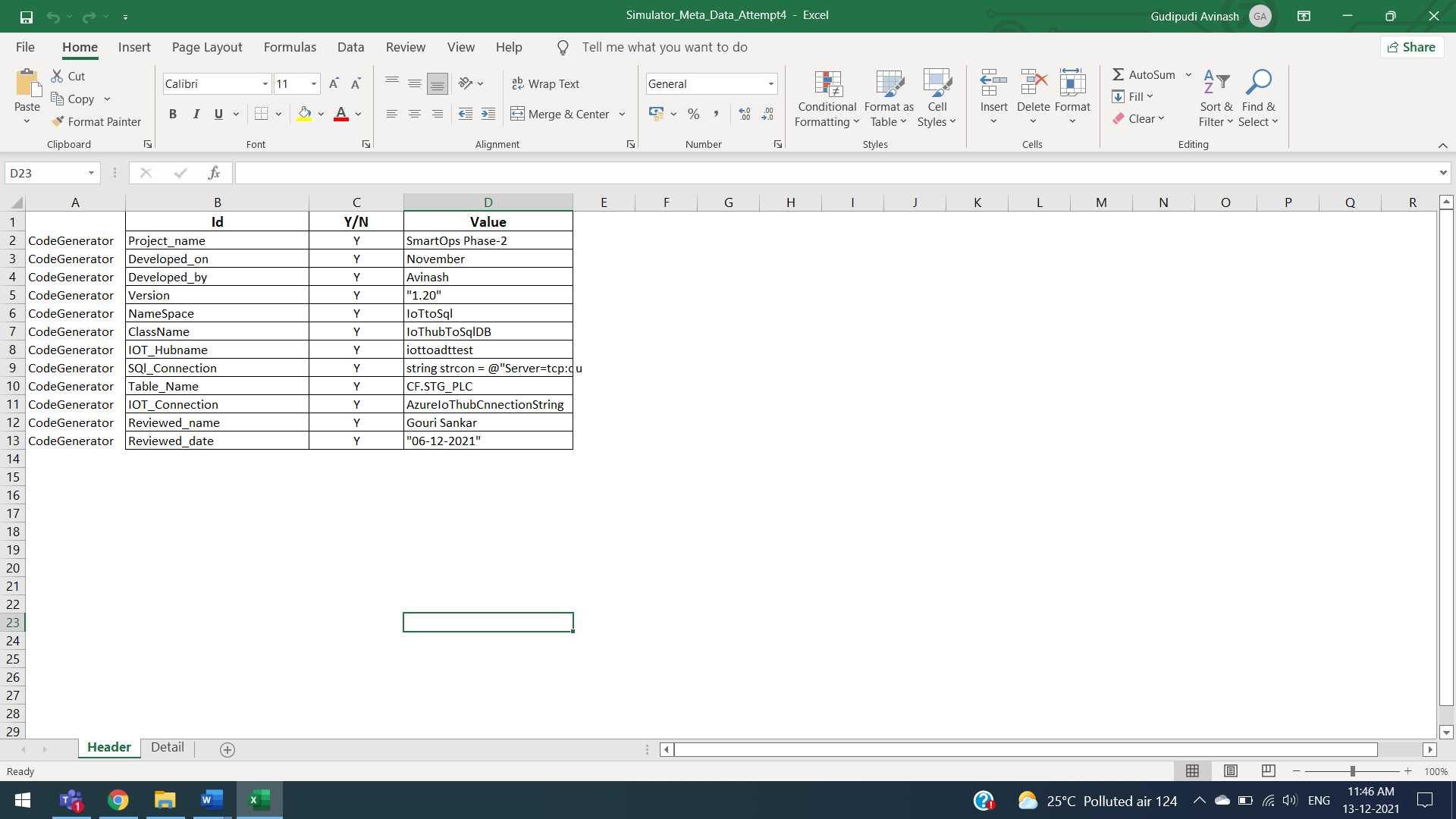


Figure .KeyValuePair Details

# Code with Explanation

*import pandas as pd*

*import time*

*import re*

*begin=time.time()*

*file\_path ="F:/Attempt\_4/Simulator\_Meta\_Data\_Attempt4.xlsx" #filePath from Local Computer*

*file\_reading= open("F:/Attempt\_4/Code\_Template\_Final.txt",'r') #Reading Code Template*

*file\_writing= open("F:/Attempt\_4/output\_Final.txt",'w') #Writing a Code*

*pattern1=re.compile(r"<<<(\w+).\d+.!(\w+)!.(\w+).!(\w+)!>>>")*

*pattern2=re.compile(r"<<<(\w+).\d+.(\w+).(\w+)>>>")*

*#(\d+) Represent a group of one or more digits*

*pattern3= re.compile(r"<<<(\w+).\d+.([\w.(]+)!(\w+)!.!([\w.]+)!!(\w+)!>>>")*

*#(\w+) Represent a group of one or more words*

*pattern4= re.compile(r"<<<(\w+).\d+.(\w+).!(\w+)!.!(\w+)!>>>")*

*for line in file\_reading:*

*for word in line.split(): #Spliting a Line By Word by Word*

*try:*

*#In Regular Expression Group 0 Represents a whole Pattern and Group 1 represents first items of '()' and so on...*

*for key1 in re.finditer(pattern1, word):*

*Sheet\_name=key1[1]*

*column1=key1[2]*

*finding\_value=key1[3]*

*column2=key1[4]*

*Reading\_Excel = pd.read\_excel(file\_path,sheet\_name=Sheet\_name) #Reading\_Sheet*

*Reading\_Excel.dropna(subset = [column1,column2], inplace=True) #Droping NAN Cells*

*Key\_column=Reading\_Excel[column1]*

*ValuePair\_column=Reading\_Excel[column2]*

*for check,rep in zip(Key\_column,ValuePair\_column):*

*if check==finding\_value:*

*word = word.replace(key1[0], rep) #Replacing Contents From Sheet Using (Key,Value) Pair*

*except:*

*print("Pattern1 ERROR!!! Please Re-Verify The Code-Template")*

*try:*

*for key2 in re.finditer(pattern2, word):*

*Sheet\_name=key2[1] #Sheet\_name through Dynamic*

*Column\_name=key2[2] #Extracting a column through Dynamic*

*Condition\_operation=key2[3]*

*Reading\_Excel = pd.read\_excel(file\_path,sheet\_name=Sheet\_name) #Reading Excel Based upon Template*

*if Condition\_operation=="ALL": # 'All' Accessing all Variables*

*Respective\_column=Reading\_Excel[Column\_name].tolist()*

*Removing\_Quotes\_for\_Respectivecolumn=",".join([str(s) for s in Respective\_column[:]])*

*word=word.replace(key2[0],str(Removing\_Quotes\_for\_Respectivecolumn))*

*if Condition\_operation=="Extended": # 'Extended' is Adding Values via Code Syntax*

*Respective\_column="@"+Reading\_Excel[Column\_name]*

*Respective\_column\_toList=Respective\_column.tolist()*

*Removing\_Quotes\_for\_Respectivecolumnvalues=",".join([str(s) for s in Respective\_column\_toList[:]])*

*word=word.replace(key2[0],str(Removing\_Quotes\_for\_Respectivecolumnvalues)) #Replacing Contents*

*except:*

*print("Pattern2 ERROR!!! Please Re-Verify The Code-Template")*

*try:*

*for key3 in re.finditer(pattern3, word):*

*Sheet\_name=key3[1]*

*variable\_name=key3[2]*

*Column\_name\_for\_Parameters=key3[3]*

*Telemetry\_variablename=key3[4]*

*Telemetry\_Column\_Parameters=key3[5]*

*Reading\_Excel = pd.read\_excel(file\_path,sheet\_name=Sheet\_name)*

*For\_SQL\_AddValue=pd.DataFrame(variable\_name+'"@'+Reading\_Excel[Column\_name\_for\_Parameters]+'",'+Telemetry\_variablename+Reading\_Excel[Telemetry\_Column\_Parameters]+");")*

*sql\_addvalue\_Dataframe=For\_SQL\_AddValue.to\_string(header=False,index=False)*

*sql\_addvalue\_Dataframe=sql\_addvalue\_Dataframe.replace(" ","")*

*word=word.replace(key3[0],sql\_addvalue\_Dataframe) #Replacing Contents*

*except:*

*print("Pattern3 ERROR!!! Please Re-Verify The Code-Template")*

*try:*

*for key4 in re.finditer(pattern4, word):*

*Sheet\_name=key4[1]*

*Access\_modifier=key4[2]*

*data\_type\_column=key4[3]*

*Column\_name=key4[4]*

*Reading\_Excel = pd.read\_excel(file\_path,sheet\_name=Sheet\_name)*

*For\_GetAndSet=Access\_modifier+" "+Reading\_Excel[data\_type\_column]+" "+Reading\_Excel[Column\_name]+" "+"{get;set;}"*

*GetAndSet=For\_GetAndSet.to\_string(index=False)*

*GetAndSet=GetAndSet.replace(" ","") #Removing Spaces*

*word=word.replace(key4[0],str(GetAndSet))*

*except:*

*print("Pattern4 ERROR!!! Please Re-Verify The Code-Template")*

*file\_writing.write(' ')*

*file\_writing.write(word)*

*file\_writing.write("\n")*

*file\_reading.close() #Closing The Code Template*

*file\_writing.close() #Closing The Output Write File*

*time.sleep(1)*

*end=time.time()*

*print(f"Time Execution {end-begin}") #Displaying The Time*

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