# PROJECT REPORT

PROJECT TITLE: CORPORATE EMPLOYEE ATTRITION ANALYTICS

TEAM ID:PNT2022TMID49703

***TEAM LEAD:NEILJOHNSON.J***

***TEAM MEMBER:1.VISHNU KUMAR.J***

***TEAM MEMBER:2.MANIKANDA HARISH.M***

***TEAM MEMBER:3.GNANA JEYASINGH.P***

***TEAM MEMBER:4.ASHLY ISHWIN.***

# 1. INTRODUCTION

FOR OUR IBM PROJECT ,WE CHOSE DATA ANALYTICS OUR DYNAMIC FOR THE NALAIYA THIRAN INITIATIVE OUR TOPIC CORPORATE EMPLOYEE ATRITION

ATTRITION REFERS TO THE REDUCTION OF STRENGTHNESS IN AN ORGANIZATION I.E, EMPLOYEE SUDDENLY RESIGNING FROM THE POST DUE TO THEIR OWN REASONS ,WHICH LEADS TO THE ORGANISATIONAL NOT BEING ABLE TO COMPLETE THEIR DUE WORK TIMELY .IN SENSE IT REPRESENTS THE LACK OF COMPETANCY IN A COMPANY TO RETAIN THEIR EMPLOYEE WHICH NECESSARY

WE INTEND TO ANALYSE SUCH ORGANISATION EMPLOYEE DATA PROVIDE THEM WITH A SOLUTION FOR PREVENTING SUCH HAPPENINGS AND IF POSSIBLE ,BE ABLE TO EVEN MOTIVATE SAID EM[PLOYEE TO WORK MORE EFF ICIENTLY

1.1 PROJECT OVERVIEW

1.1 PROJECT OVERVIEW

# TO IDENTIFY AND RETAIN EXPERIENCED TALENTED AND INTRESTED EMPLOYEE

#UNDERSTANDING EMPLOYEES INTRESTED OR LACK THERE OF IN ORDER TO PROVIDE THEM DESERVING RAISE AND INCENTIVE FOR FUTHER PROGRESS

#REFERS TO THE TECHNIQUES IMPLEMENTED BY THE MANAGMMENT TO HELP EMPLOYEES STAY WITH THE ORGANISATION FOR A LONGER PERIOD

1.2 PURPOSE

THE PURPOSE OF OUR PROJECT IS TO HELP ORGANIZATI0ON TO RETAIN THEM EMPLOYEE WITHIN AND PROVIDE THEMWITH SOLUTIONS WHICH OFFER PROJECT INCENTIVES FOR THE EMPLOYEE TO WORK COMMENLY EVEN FURTHER

# LITERATURE SURVEY

#### 2.1 EXISTING PROBLEM

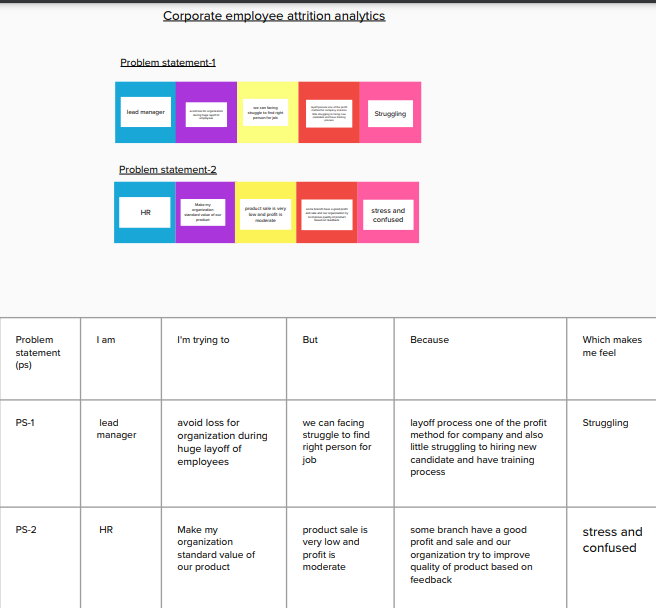
MORE ALONG THE LINES OF PREDECTION ,BASED ON PAST BEHAVIOR AND CHOICES PROBABLY EFFECTING THE ORGANISATION AS WELL

2.2 REFERENCE

https://www.aihr.com/blog/employee-attrition/ 2. http://www.zenworkplace.com/2014/07/01/costemployee-turnover/ 3. http://www.compensationforce.com/2016/04/2015-turnoverrates-by-industry.html 4. http://www.shrm.org/Research/SurveyFindings/Documents

#### 2.3 PROBLEM STATE DEFINATION

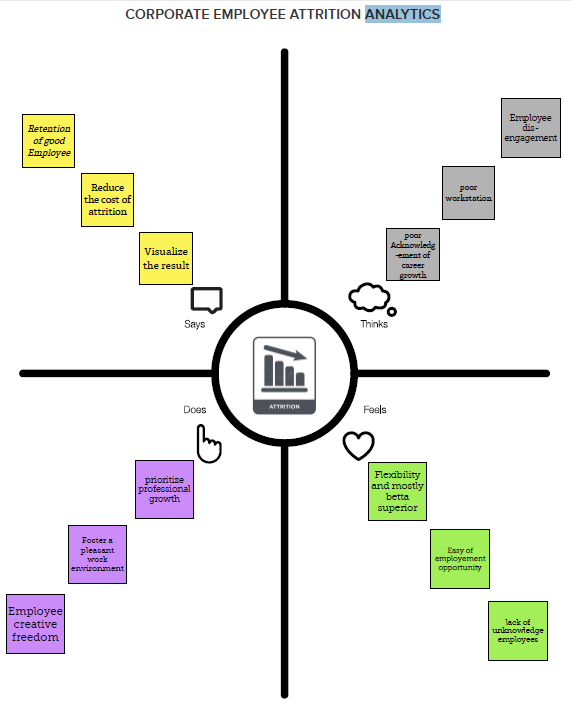
#### PROBLEM STATEMENT



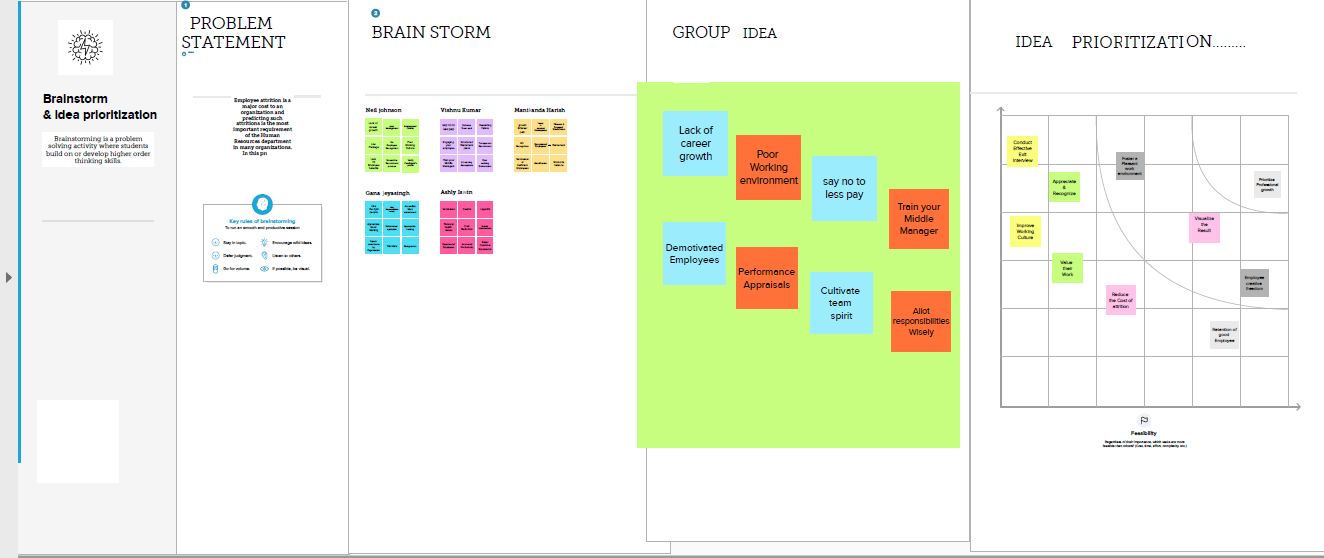
#### 

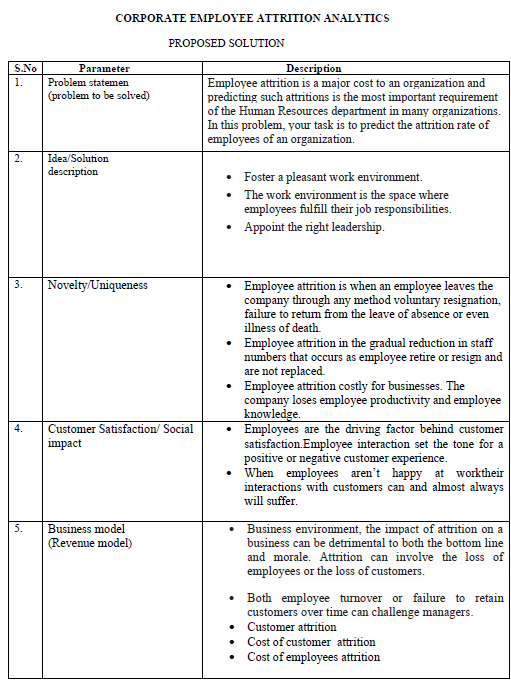
# 3.IDEATION AND PROPOSED SOLUTION

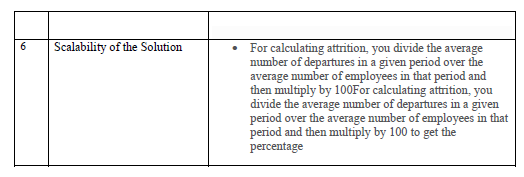
#### 3.1 EMPATHY MAP CANVAS



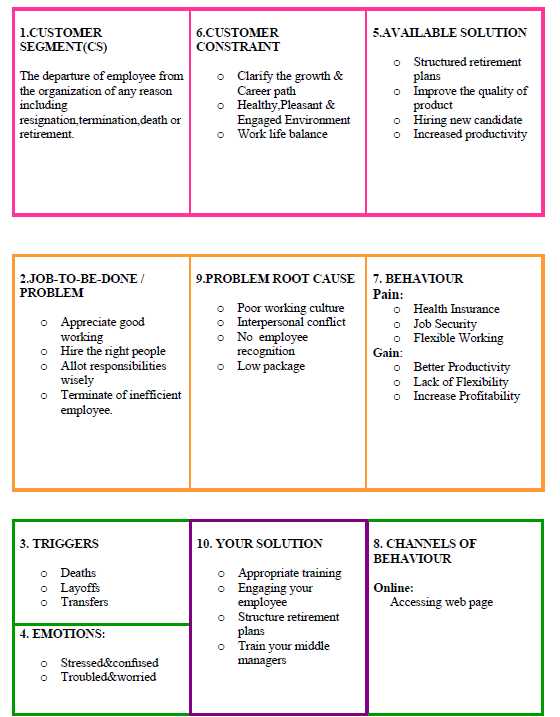
#### 3.2 IDEATION AND BRAINSTORMING





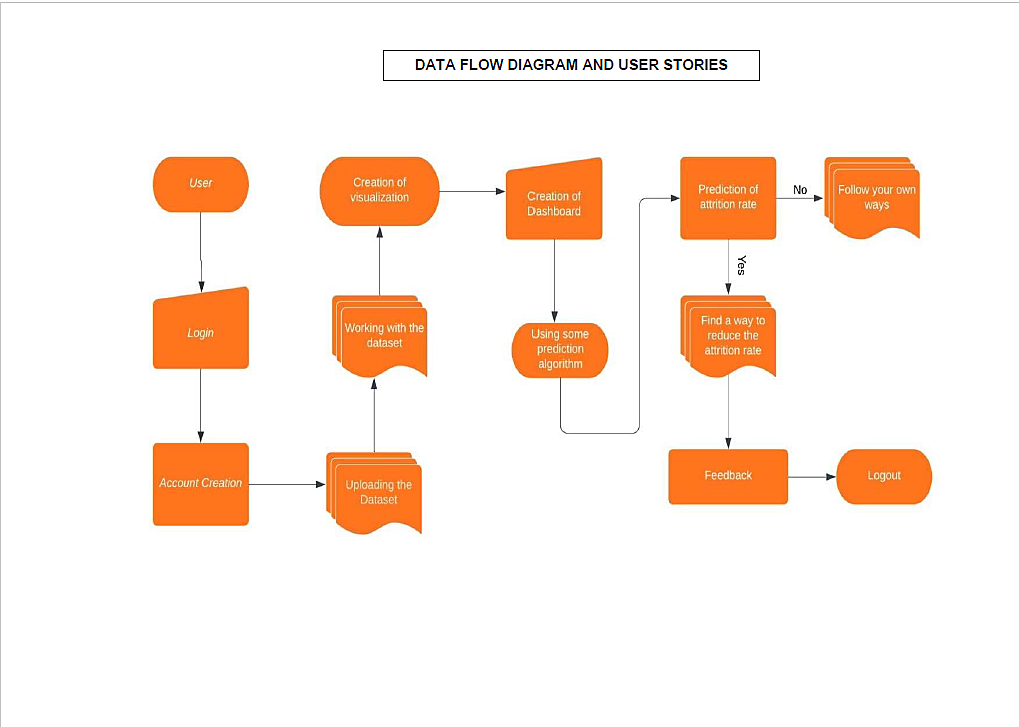


### 3.3 PROBLEM SOLUTION FIT



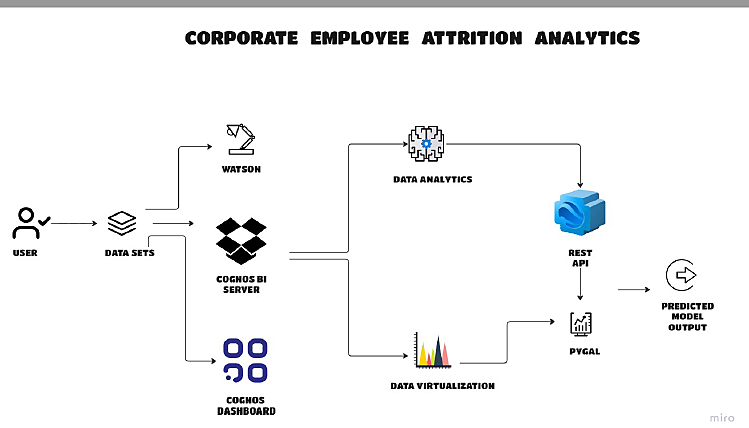
# **PROJECT DESIGN**

4.1 DATA FLOW DIAGRAM

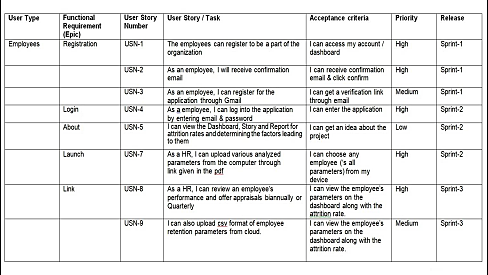


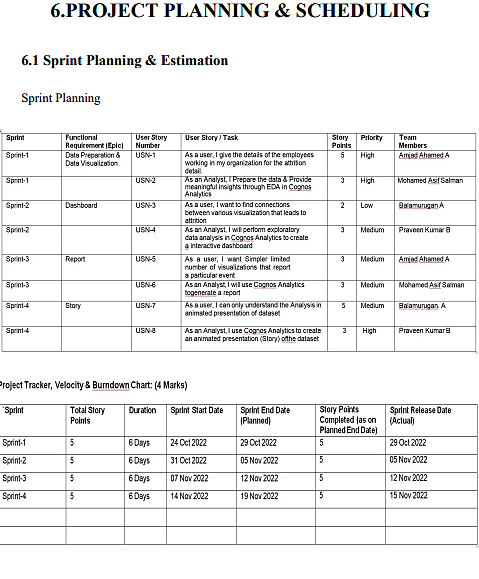
#### 5.2 SOLUTION AND TECHNICAL ARCHITECTURE

#### TECHNICAL ARCHITECTURE

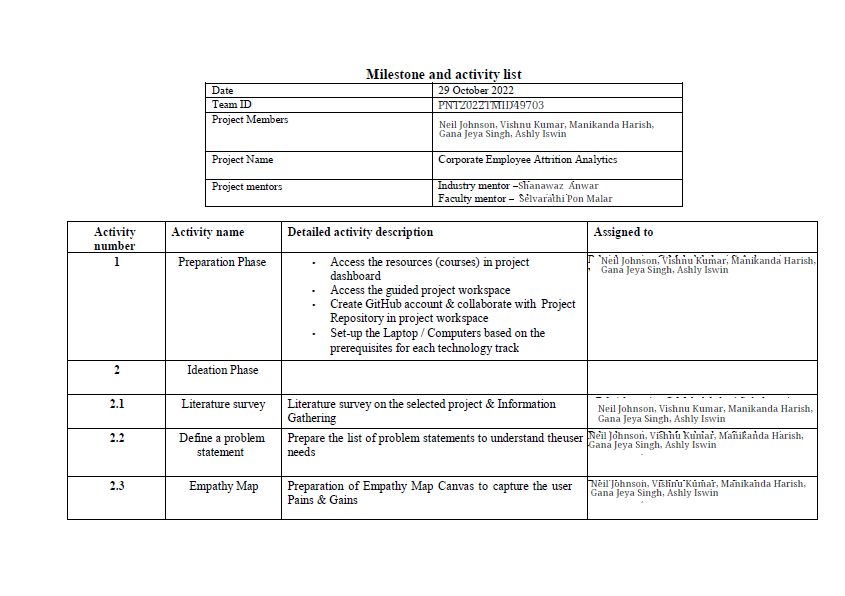


# User Stories





# 



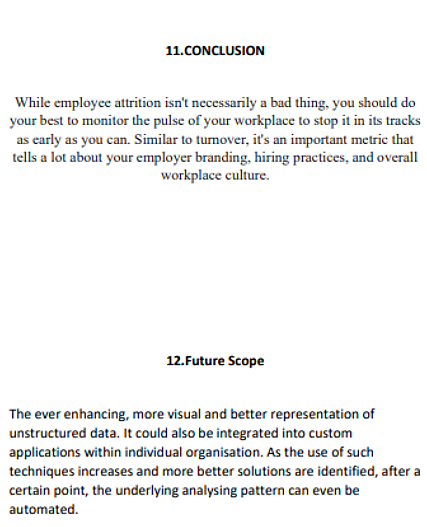
#### Report

#### 

## 

## 

## 



## APPENDIX

SOURCE CODE

{  
 "nbformat": 4,  
 "nbformat\_minor": 0,  
 "metadata": {  
 "colab": {  
 "provenance": []  
 },  
 "kernelspec": {  
 "name": "python3",  
 "display\_name": "Python 3"  
 },  
 "language\_info": {  
 "name": "python"  
 }  
 },  
 "cells": [  
 {  
 "cell\_type": "markdown",  
 "source": [  
 "\*\*Exercises\*\*\n",  
 "\n",  
 "\n",  
 "\n",  
 "```\n",  
 "# This is formatted as code\n",  
 "```\n",  
 "\n",  
 "\n",  
 "Answer the questions or complete the tasks outlined in bold below, use the specific method described if applicable.\n",  
 "\n",  
 "\*\* What is 7 to the power of 4?\*\*"  
 ],  
 "metadata": {  
 "id": "3bjl4UytNmwK"  
 }  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "7\*\*4\n",  
 "print(7\*\*4)\n"  
 ],  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/"  
 },  
 "id": "71B2UnbbN03I",  
 "outputId": "03ff8443-5c54-4157-d6cd-beb3b93b9940"  
 },  
 "execution\_count": 1,  
 "outputs": [  
 {  
 "output\_type": "stream",  
 "name": "stdout",  
 "text": [  
 "2401\n"  
 ]  
 }  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "source": [  
 "\*\* Split this string:\*\*\n",  
 "\n",  
 "s = \"Hi there Sam!\"\n",  
 "\n",  
 "into a list."  
 ],  
 "metadata": {  
 "id": "JcZQw93eOGLk"  
 }  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "s=\"Hi there sam!\"\n",  
 "s.split()\n"  
 ],  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/"  
 },  
 "id": "WQIrbMUaOIJX",  
 "outputId": "d090659f-519e-46d7-baf8-03a4321ff097"  
 },  
 "execution\_count": 2,  
 "outputs": [  
 {  
 "output\_type": "execute\_result",  
 "data": {  
 "text/plain": [  
 "['Hi', 'there', 'sam!']"  
 ]  
 },  
 "metadata": {},  
 "execution\_count": 2  
 }  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "source": [  
 "\*\* Given the variables:\*\*\n",  
 "\n",  
 "planet = \"Earth\"\n",  
 "diameter = 12742\n",  
 "\n",  
 "\*\* Use .format() to print the following string: \*\*\n",  
 "\n",  
 "The diameter of Earth is 12742 kilometers"  
 ],  
 "metadata": {  
 "id": "S5eaKI0gORer"  
 }  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "planet=\"Earth\"\n",  
 "diameter=\"12742\"\n",  
 "print(\"The diameter of{} is {} kilometers.\".format(planet,diameter))"  
 ],  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/"  
 },  
 "id": "Iy\_84\_dzOWy4",  
 "outputId": "4a149c30-27b5-4546-bb28-eb4ca72f6716"  
 },  
 "execution\_count": 3,  
 "outputs": [  
 {  
 "output\_type": "stream",  
 "name": "stdout",  
 "text": [  
 "The diameter ofEarth is 12742 kilometers.\n"  
 ]  
 }  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "source": [  
 "\*\* Given this nested list, use indexing to grab the word \"hello\" \*\*\n",  
 "\n",  
 "\*\* Given this nest dictionary grab the word \"hello\". Be prepared, this will be annoying/tricky \*\*\n"  
 ],  
 "metadata": {  
 "id": "ZgUL5MbxN6i5"  
 }  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "I = [1,2,[3,4],[5,[100,200,['hello']],23,11],1,7]\n",  
 "\n",  
 "d= {'k1':[1,2,3,{'tricky':['oh','man','inception',{'target':[1,2,3,'hello']}]}]}\n",  
 "I[3][1][2][0]\n",  
 "d['k1'][3]['tricky'][3]['target'][3]"  
 ],  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/",  
 "height": 35  
 },  
 "id": "5kqlqLNPOkMv",  
 "outputId": "1788aff4-7239-4852-9a16-2738accce316"  
 },  
 "execution\_count": 4,  
 "outputs": [  
 {  
 "output\_type": "execute\_result",  
 "data": {  
 "text/plain": [  
 "'hello'"  
 ],  
 "application/vnd.google.colaboratory.intrinsic+json": {  
 "type": "string"  
 }  
 },  
 "metadata": {},  
 "execution\_count": 4  
 }  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "source": [  
 "\*\* What is the main difference between a tuple and a list? \*\*"  
 ],  
 "metadata": {  
 "id": "DwJ6\_fy\_O1gK"  
 }  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "#tuple-im mutable\n",  
 "#list -mutable"  
 ],  
 "metadata": {  
 "id": "IJR\_mBNYO6sl"  
 },  
 "execution\_count": 6,  
 "outputs": []  
 },  
 {  
 "cell\_type": "markdown",  
 "source": [  
 "\*\* Create a function that grabs the email website domain from a string in the form: \*\*"  
 ],  
 "metadata": {  
 "id": "a22o5uQxPftu"  
 }  
 },  
 {  
 "cell\_type": "markdown",  
 "source": [  
 "user@domain.com\n",  
 "\n",  
 "So for example, passing \"user@domain.com\" would return: domain.com"  
 ],  
 "metadata": {  
 "id": "TGmj-U\_7PNhu"  
 }  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "def domainGet(email):\n",  
 " return email.split('@')[-1]"  
 ],  
 "metadata": {  
 "id": "7FM-VXCaPPmo"  
 },  
 "execution\_count": 7,  
 "outputs": []  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "domainGet('user@domain.com')"  
 ],  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/",  
 "height": 35  
 },  
 "id": "KMk4bgEwPtK0",  
 "outputId": "210876f9-65ff-4a1a-a53c-929290c41784"  
 },  
 "execution\_count": 8,  
 "outputs": [  
 {  
 "output\_type": "execute\_result",  
 "data": {  
 "text/plain": [  
 "'domain.com'"  
 ],  
 "application/vnd.google.colaboratory.intrinsic+json": {  
 "type": "string"  
 }  
 },  
 "metadata": {},  
 "execution\_count": 8  
 }  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "source": [  
 "\*\* Create a basic function that returns True if the word 'dog' is contained in the input string. Don't worry about edge cases like a punctuation being attached to the word dog, but do account for capitalization. \*\*"  
 ],  
 "metadata": {  
 "id": "7S0ucI8YPy8N"  
 }  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "def findDog(st):\n",  
 " return 'dog' in st.lower().split()"  
 ],  
 "metadata": {  
 "id": "hMOQ5kjtP6Zz"  
 },  
 "execution\_count": 9,  
 "outputs": []  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "findDog('Isthere a dog here')"  
 ],  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/"  
 },  
 "id": "YhEhZLzbQCuN",  
 "outputId": "52ec609d-e3f6-45e2-9909-a22d84218479"  
 },  
 "execution\_count": 10,  
 "outputs": [  
 {  
 "output\_type": "execute\_result",  
 "data": {  
 "text/plain": [  
 "True"  
 ]  
 },  
 "metadata": {},  
 "execution\_count": 10  
 }  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "source": [  
 "\n",  
 "\*\* Create a function that counts the number of times the word \"dog\" occurs in a string. Again ignore edge cases. \*\*"  
 ],  
 "metadata": {  
 "id": "TyGy31y9QIOj"  
 }  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "def countDog(st):\n",  
 " count=0\n",  
 " for word in st.lower().split():\n",  
 " if word=='dog':\n",  
 " count+= 1\n",  
 " return count"  
 ],  
 "metadata": {  
 "id": "lMmqrDn9QJji"  
 },  
 "execution\_count": 14,  
 "outputs": []  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "countDog('This dogs runs faster than the other dog dude!')\n"  
 ],  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/"  
 },  
 "id": "sG4TnD\_IQVxe",  
 "outputId": "38196796-15a7-4135-9c11-346d5a46d9ae"  
 },  
 "execution\_count": 13,  
 "outputs": [  
 {  
 "output\_type": "execute\_result",  
 "data": {  
 "text/plain": [  
 "1"  
 ]  
 },  
 "metadata": {},  
 "execution\_count": 13  
 }  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "source": [  
 "\n",  
 "Problem\n",  
 "You are driving a little too fast, and a police officer stops you. Write a function to return one of 3 possible results: \"No ticket\", \"Small ticket\", or \"Big Ticket\". If your speed is 60 or less, the result is \"No Ticket\". If speed is between 61 and 80 inclusive, the result is \"Small Ticket\". If speed is 81 or more, the result is \"Big Ticket\". Unless it is your birthday (encoded as a boolean value in the parameters of the function) -- on your birthday, your speed can be 5 higher in all cases."  
 ],  
 "metadata": {  
 "id": "Gwd0quguQhsF"  
 }  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "def caught\_speeding(speed, is\_birthday):\n",  
 " \n",  
 " if is\_birthday:\n",  
 " speeding = speed - 5\n",  
 " else:\n",  
 " speeding = speed\n",  
 " \n",  
 " if speeding > 80:\n",  
 " return 'Big Ticket'\n",  
 " elif speeding > 60:\n",  
 " return 'Small Ticket'\n",  
 " else:\n",  
 " return 'No Ticket'"  
 ],  
 "metadata": {  
 "id": "1TZnSSlrQklZ"  
 },  
 "execution\_count": 15,  
 "outputs": []  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "caught\_speeding(81,False)"  
 ],  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/",  
 "height": 35  
 },  
 "id": "Vu4ZRKDcQyxZ",  
 "outputId": "ff428158-8dad-40f4-c0a9-7938cfbd3292"  
 },  
 "execution\_count": 16,  
 "outputs": [  
 {  
 "output\_type": "execute\_result",  
 "data": {  
 "text/plain": [  
 "'Big Ticket'"  
 ],  
 "application/vnd.google.colaboratory.intrinsic+json": {  
 "type": "string"  
 }  
 },  
 "metadata": {},  
 "execution\_count": 16  
 }  
 ]  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "caught\_speeding(81,True)"  
 ],  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/",  
 "height": 35  
 },  
 "id": "gSk\_7zBLQ2pJ",  
 "outputId": "057e5057-d732-4116-cdef-5bafb32ab7b4"  
 },  
 "execution\_count": 17,  
 "outputs": [  
 {  
 "output\_type": "execute\_result",  
 "data": {  
 "text/plain": [  
 "'Small Ticket'"  
 ],  
 "application/vnd.google.colaboratory.intrinsic+json": {  
 "type": "string"  
 }  
 },  
 "metadata": {},  
 "execution\_count": 17  
 }  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "source": [  
 "Create an employee list with basic salary values(at least 5 values for 5 employees) and using a for loop retreive each employee salary and calculate total salary expenditure."  
 ],  
 "metadata": {  
 "id": "q2rpbb1bQ77q"  
 }  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "employee\_salary = [2000,3000,4000,5000,6000]\n",  
 "total\_salary = 0\n",  
 "for salary in employee\_salary:\n",  
 " total\_salary+=salary\n",  
 " print(salary)"  
 ],  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/"  
 },  
 "id": "KptMfDKdQ9ut",  
 "outputId": "1b4e5838-e268-4098-894b-3b8788d570cd"  
 },  
 "execution\_count": 18,  
 "outputs": [  
 {  
 "output\_type": "stream",  
 "name": "stdout",  
 "text": [  
 "2000\n",  
 "3000\n",  
 "4000\n",  
 "5000\n",  
 "6000\n"  
 ]  
 }  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "source": [],  
 "metadata": {  
 "id": "EGuvaYd4QaZC"  
 }  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "print(total\_salary)"  
 ],  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/"  
 },  
 "id": "yNeNcep4RFSK",  
 "outputId": "8e67b053-ca5a-4d91-a24a-ba7aa7fd6937"  
 },  
 "execution\_count": 19,  
 "outputs": [  
 {  
 "output\_type": "stream",  
 "name": "stdout",  
 "text": [  
 "20000\n"  
 ]  
 }  
 ]  
 },  
 {  
 "cell\_type": "markdown",  
 "source": [  
 "Create two dictionaries in Python:\n",  
 "\n",  
 "First one to contain fields as Empid, Empname, Basicpay\n",  
 "\n",  
 "Second dictionary to contain fields as DeptName, DeptId.\n",  
 "\n",  
 "Combine both dictionaries."  
 ],  
 "metadata": {  
 "id": "MXs7AI62RRPY"  
 }  
 },  
 {  
 "cell\_type": "code",  
 "source": [  
 "dict1 = {'Empid':1, 'Empname':'kalai','Basicpay':15000}\n",  
 "dict2 = {'DeptName':'computer science','DepId':104}\n",  
 "dict3 = {\*\*dict1,\*\*dict2}\n",  
 "print(dict3)"  
 ],  
 "metadata": {  
 "colab": {  
 "base\_uri": "https://localhost:8080/"  
 },  
 "id": "MB6LwNOxRSdY",  
 "outputId": "83b849f1-80cf-44bc-b90e-97f44d98075c"  
 },  
 "execution\_count": 20,  
 "outputs": [  
 {  
 "output\_type": "stream",  
 "name": "stdout",  
 "text": [  
 "{'Empid': 1, 'Empname': 'kalai', 'Basicpay': 15000, 'DeptName': 'computer science', 'DepId': 104}\n"  
 ]  
 }  
 ]  
 }  
 ]

#### VIDEO LINK

#### GITHUB LINK :

https://github.com/IBM-EPBL/IBM-Project-50197-1660899663.git