THE ASCY AIRPLANE DATABASE

Computer Science Project

Table of Contents

Introduction	**3**
Applications	**4**
Logic	**5**
Flowcharts	**8**
Code	**12**

Introduction

ASCY stand for an application created by **A**dvay, **C**hirag and **S**rijan which would go on to store **Y**ottabytes of data.

ASCY is a database system tailored for the usage of a commercial airline company. It boasts multiple tables with ease of access, with a 2-factor secured admin account that can assign security levels to the viewing and editing of different tables. The application can also automatically update a lot of data values if connected to its respective source, for example, an input to the projected fuel costs will help it to automatically generate ticket prices for the all the flights on that particular day, depending on the number of stops, distance travelled, payload, pro t margin etc. Similarly an input on the weather conditions is helpful to predict the Estimated Time of Arrival.

The software was built with a small to medium scale company in mind, but has been coded in a way so as to not hinder scalability. A new table according to company demands can easily be added, assigned a security level and integrated as if it were there from the very beginning.

The program will help reduce man hours spent on manually entering data values, projecting prices and other menial tasks, drastically helping the company to reduce expenses as well as focus its monetary aide towards the safety and advancement of air travel.

Applications

Each and every successful company over the last years, strives to achieve a level of excellence by being extremely efficient across all fields of work. Our program uses the simple applications of a Database management system and allows seamless and smooth connectivity among company employees, executives and infrastructure. In order to make things even simpler, we use python to connect a DBMS to the user interface that is very easy to understand.

The database management system (DBMS) handles the data; the database engine allows data to be viewed, locked, and updated; and the database schema specifies the logical structure of the database. These three fundamental features contribute to concurrency, security, data integrity, and consistent data administration methods. Many common database administration functions are supported by the DBMS, including change management, performance monitoring and tuning, security, and backup and recovery. Most database management systems are also in charge of automatic rollbacks and restarts, as well as recording and auditing of database and application activities.

The DBMS provides a centralised view of data that may be accessed in a controlled way by numerous users from different places. A database management system (DBMS) can limit what data end users see and how they see the data by offering many views of a single database schema. Because the DBMS handles all requests, end users and software programmes are not required to understand where the data is physically housed or on what sort of storage media it resides.

The DBMS can provide both logical and physical data independence, shielding users and applications from needing to know where data is kept or being worried about changes to data's physical form. Developers will not have to adjust programmes simply because the database has changed if applications use the application programming interface (API) provided by the DBMS.

Python has bindings for many database systems including MySQL, Postgresql, Oracle, Microsoft SQL Server and Maria DB.

Some other examples of DBMS with Python connectivity are:

- Railway stations services
- Human Resource services in any company
- Social Media experiences
- Credit Card management (banks)
- E-Commerce
- Manufacturing

Logic

Login

A multi-level login system with the following classifications, based on usage and ease of access. This can be customized to the company's needs as well: ● A-Level Clearance

This would grant the user access to the entire company database.

- B-Level Clearance

  A clearance tailored to the needs of the ticket **booking staff** and websites, it grants access solely to the ticket prices and availability records.

- C-Level Clearance

  A special clearance for the **crew** (pilots and cabin crew), it would grant access to the Flight and Plane Details.

- E-Level Clearance

  The E-Level Clearance grants access to the Plane Maintenance database for **engineers** working on snags.

- F-Level Clearance

  The user would have access to all **Financial** details of the company. Ideal for employees in the financial department.

Plane Management/Maintenance

A database that would record the planes owned by the company, their specific details, abilities and engineers will approve the plane for take o in the Status column.

ID	Model	YoM	Passenger Capacity	Fuel Consumption per HO	Endurance	Hours own	Status (Airworthy/Check/ Grounded/Retired)

ID - An identification code unique to each plane to be used for efficient communication (integer)

Model - All general and specific repairs will be done keeping the plane model in mind (string)

YoM - Year of Manufacture helps determine the extent of repair and retirement date (string)

Passenger capacity - Helps to determine the demand-supply ratio (integer)

Fuel Consumption - The average fuel consumption per Hour of Operation (HO) (integer (liters))

Endurance - The maximum distance that the plane is capable of ying on a full fuel tank and at full capacity (integer (nautical miles))

Hours own - The number of hours it has been in use would help determine the extent of repairs

(integer)

Status - Indicates what state the plane is in; it will be filled out by the Chief Engineer

Airworthy means that the plan is ready for its next flight

Grounded means that the plane is currently unavailable for flights. This could be for repairs, tests, inspections, reserves or any other reason

Retired means that the plane is now useless to the company due to outdated equipment, the cost of updating which will be more than buying a new plane

---

Passenger Database

Passenger Code	Flight Number	Economy or First Class	Name	Sex	Phone	Email

Passenger Code - This will be a unique 3 digit code assigned to each individual passenger

Flight Number - Linked to the flight details table, it would help to identify which flight the passenger is on

Class - Economy or First depending on personal preference

Passenger Details (Name, Sex, Phone, Email) - All required details and will be necessary when contacting the person

---

Flight Details + Ticket Costs

Flight ID	Destination	Date	Boarding Time	Boarding Gate	Ticket Cost

A database which will contain all ticket and departure details

Flight ID - An identification code unique to each plane to be used for efficient communication (integer)

Destination - The place to which the fight is ying, the starting point is always XYZ terminal

Date - The date of departure of the fight

Boarding Time- The boarding time, passengers must be seated, one hour before the fight departure

Boarding Gate - The gate at which passengers will board the aircraft

Ticket Cost - The total pricing of a singular ticket on the fight

---

Financials

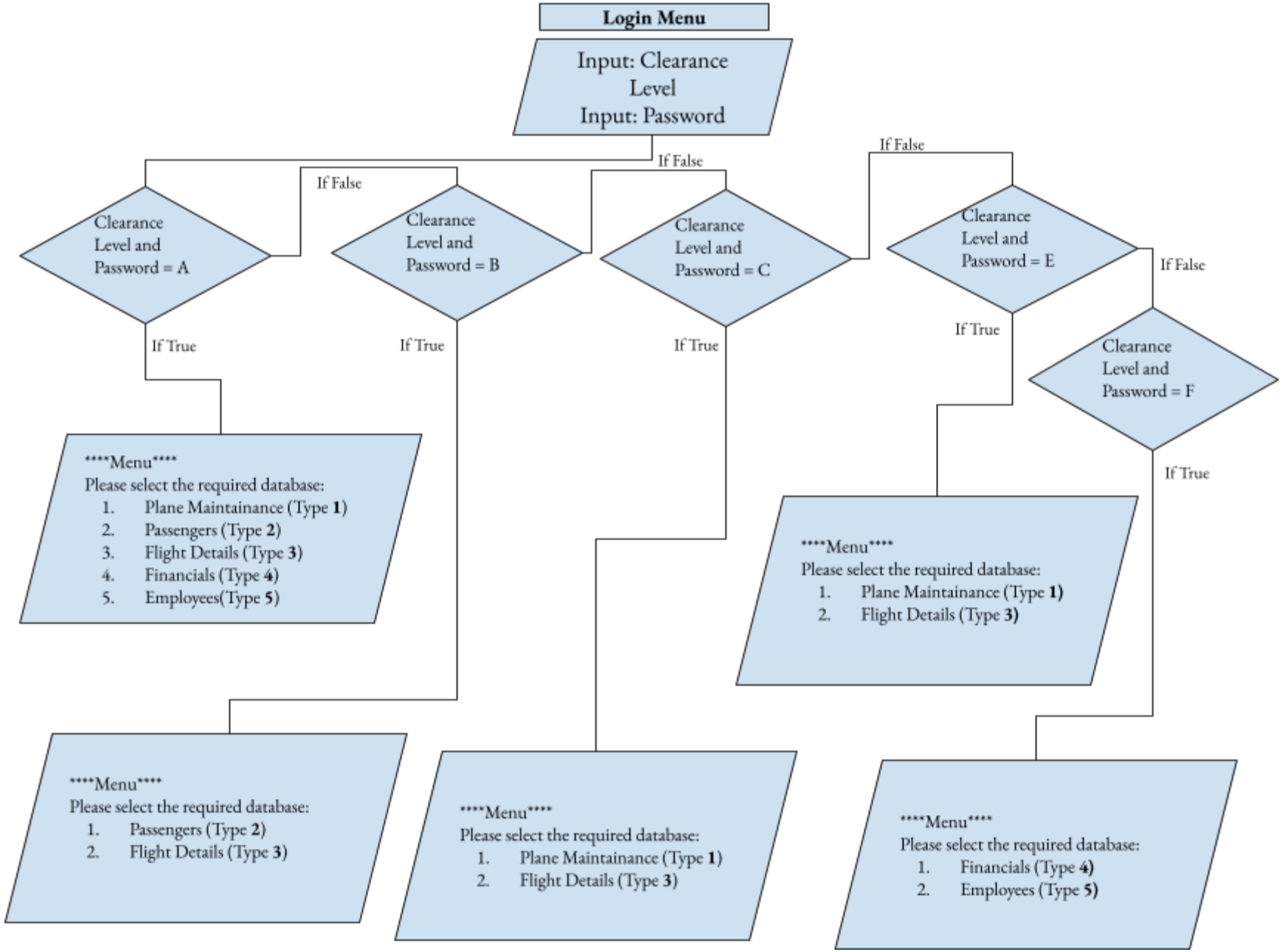
Employee Details

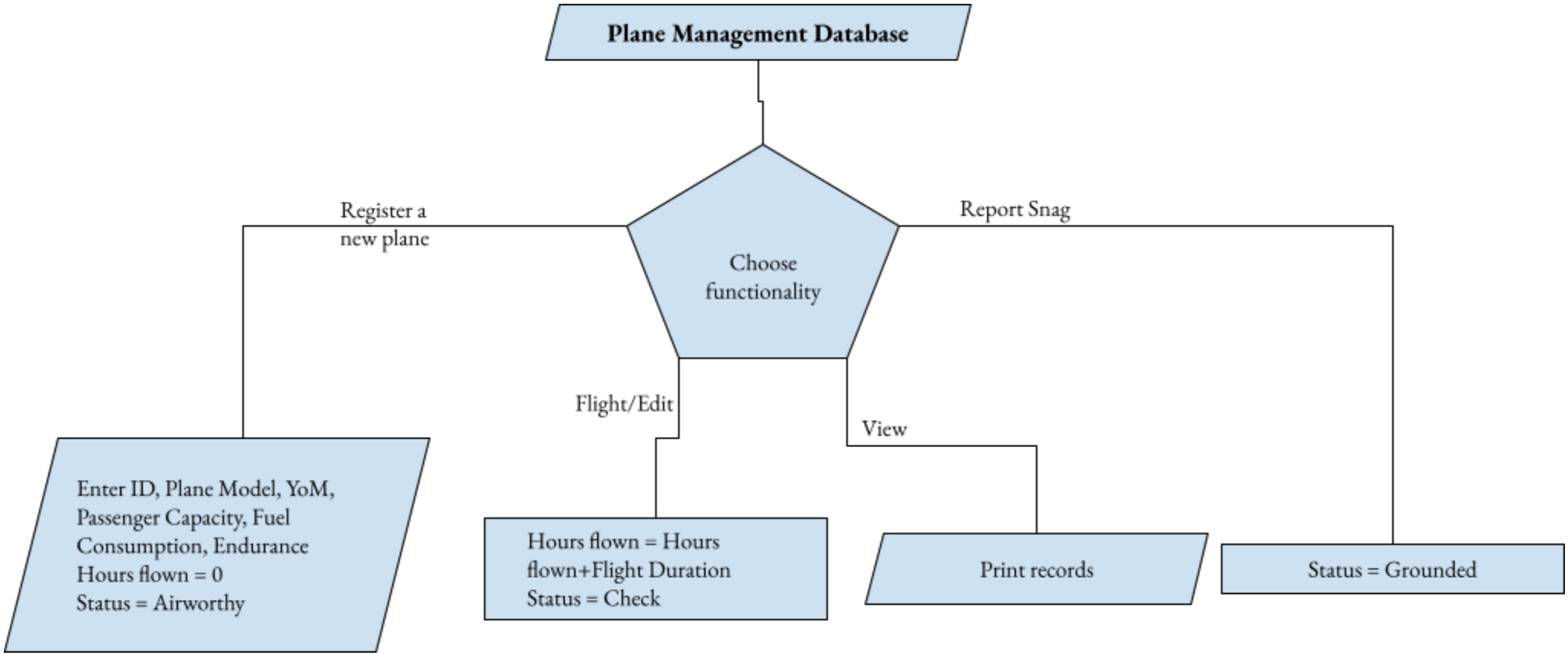
EmpID	Name	Age	Date of Joining	Designation	Salary

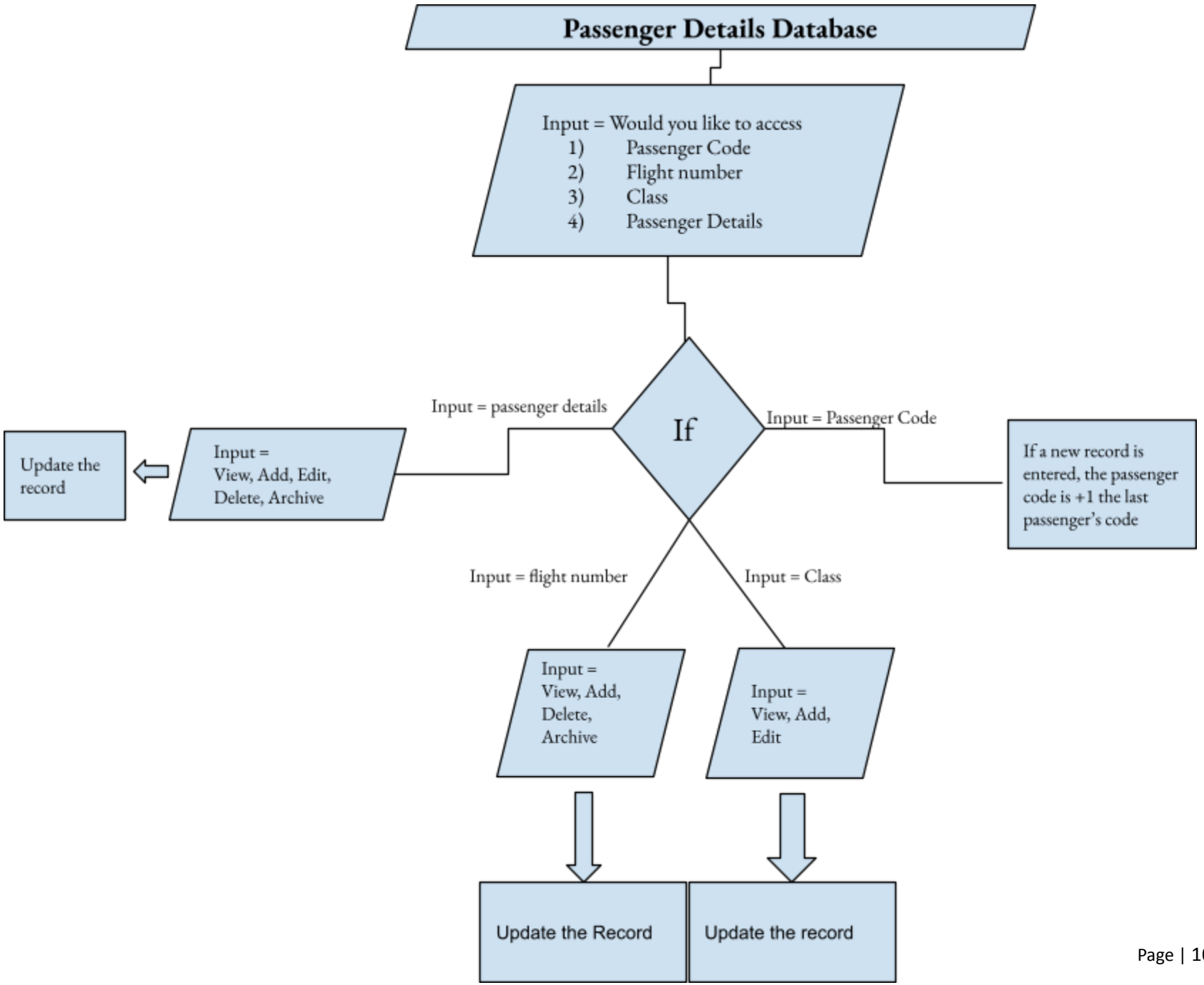
Will have an option to select category of employee (high executive or a normal employee) Input salary of the person

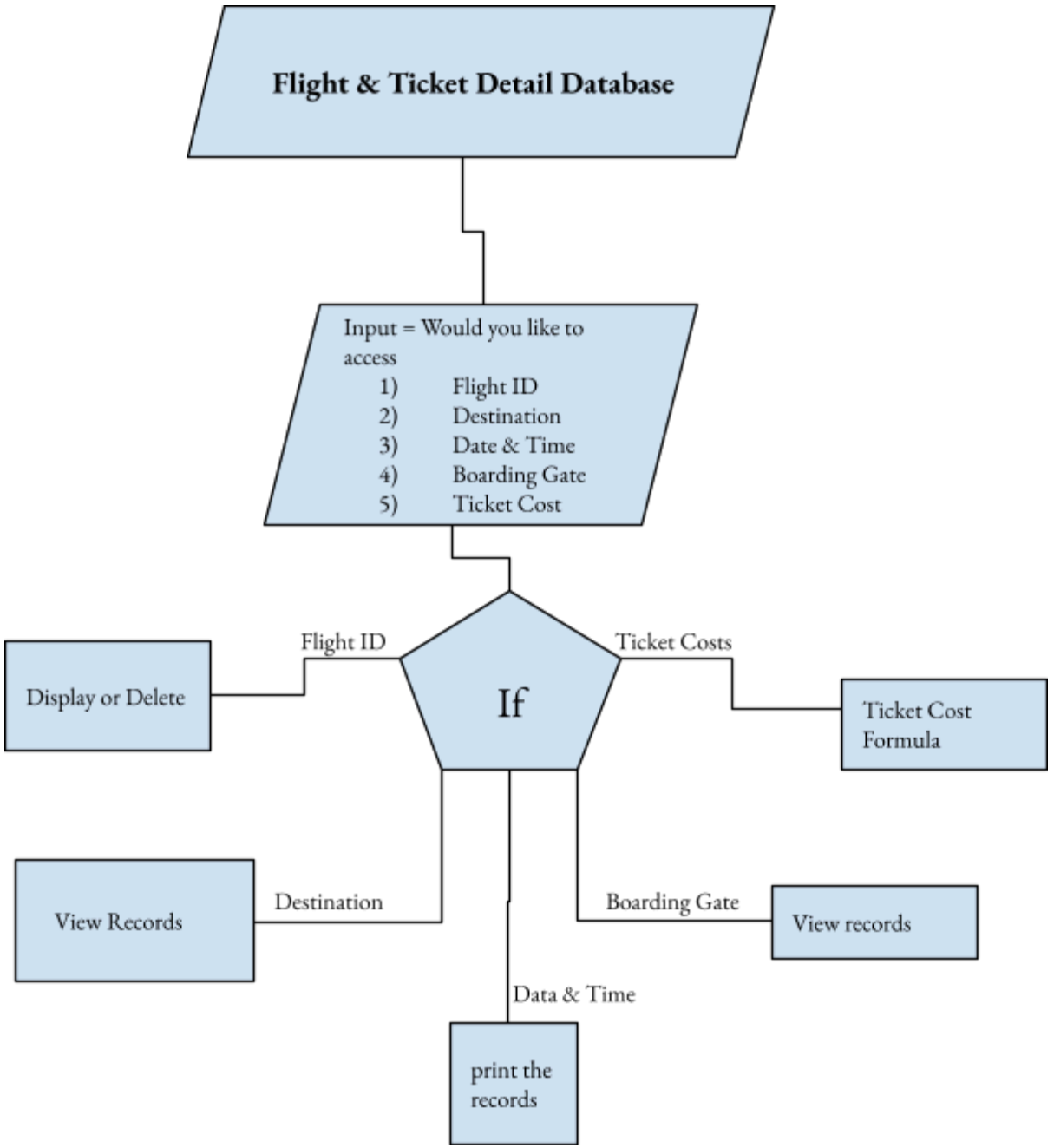
Optional feature - A formula for an increment of salaries by a particular percentage

---


Login Menu
Input: Clearance Level
Input: Password
Clearance Level and Password = A
If False
Clearance Level and Password = B
If False
Clearance Level and Password = C
If False
Clearance Level and Password = E
If False
Clearance Level and Password = F
If True
If True
If True
If True
If True
****Menu****
Please select the required database:
1. Plane Maintainance (Type 1)
2. Passengers (Type 2)
3. Flight Details (Type 3)
4. Financials (Type 4)
5. Employees(Type 5)
****Menu****
Please select the required database:
1. Passengers (Type 2)
2. Flight Details (Type 3)
****Menu****
Please select the required database:
1. Plane Maintainance (Type 1)
2. Flight Details (Type 3)
****Menu****
Please select the required database:
1. Plane Maintainance (Type 1)
2. Flight Details (Type 3)
****Menu****
Please select the required database:
1. Financials (Type 4)
2. Employees (Type 5)


Plane Management Database
Choose functionality
Register a new plane
Report Snag
Flight/Edit
View
Enter ID, Plane Model, YoM, Passenger Capacity, Fuel Consumption, Endurance
Hours flown = 0
Status = Airworthy
Hours flown = Hours flown+Flight Duration
Status = Check
Print records
Status = Grounded


Passenger Details Database
Input = Would you like to access
1) Passenger Code
2) Flight number
3) Class
4) Passenger Details
If
Input = passenger details
Input = Passenger Code
Input = flight number
Input = Class
Update the record
Input = View, Add, Edit, Delete, Archive
If a new record is entered, the passenger code is +1 the last passenger's code
Input = View, Add, Delete, Archive
Input = View, Add, Edit
Update the Record
Update the record


Flight & Ticket Detail Database
Input = Would you like to access
1) Flight ID
2) Destination
3) Date & Time
4) Boarding Gate
5) Ticket Cost
If
Flight ID
Display or Delete
Ticket Costs
Ticket Cost Formula
Destination
View Records
Boarding Gate
View records
Data & Time
print the records

CODE

Creating the Databases

This will be inclusive of the initial setup when a company opts to use ASCY.

Dummy records have been added to these databases for demonstration purposes.

Plane Mananagement

```
create database PM;
use PM;
create table P_Management (Id int primary key,
 Model varchar(20), YoM int (4), Capacity int(3),
 Fuel_Hour int(5), Endurance int (5), Hours_F own int (5),
Status varchar (20));

Insert into P_Management values ( 1, 'Airbus 320',2000 , 150, 350, 6300, 20000,
'Grounded');
Insert into P_Management values ( 2, 'Airbus 321',2010 , 180, 450, 8000, 25000,
'Airworthy');
Insert into P_Management values ( 3, 'Airbus 330',2015 , 220, 750, 10000, 12000,
'Check');
Insert into P_Management values( 5, 'Boeing 737',1998 , 150, 100, 2500, 25000,
'Retired');
```

Passenger Details

```
create database PAX;
use PAX;
create table DETAILS (Pax_Code int(5) primary key, F
 ight_No int (5),
Class varchar(20),Name varchar (20),
Phone_No int (20), Sex varchar (20),
Email varchar (30));

Insert into DETAILS values (12345, 22222, 'First', 'Chirag', 9981928182 , ' Male',
'abc@gmail.com');
Insert into DETAILS values (13345, 22223, 'First', 'Advay', 8981928182 , ' Male',
'xyz@gmail.com');
Insert into DETAILS values (14345, 22224, 'First', 'Srijan', 7981928182 , ' Male',
'pqr@gmail.com');
Insert into DETAILS values (15345, 22225, 'Economy', 'Tarini', 6981928182 , '
Female', 'aaa@gmail.com');
```

F ight and Ticket Details create

```
database DETAILS;
use DETAILS;
```

```
create table DEETS (F ightID varchar(5) primary key,
Destination varchar (20), Date_Departure Date, Gate_No
int(3), Boarding_Time time,
Ticket_Cost int(5));

Insert into DEETS values ( 'ABC21', 'Bombay', '2022-12-10', 35, '02 00 00', 10000);
Insert into DEETS values ( 'ABC22', 'Delhi', '2022-12-12, 12',' 06 00 00', 9000);
Insert into DEETS values ( 'ABC23', 'London', '2022-12-11', 9, '12 30 00', 90000);
Insert into DEETS values( 'PQR21', 'New York',' 2022-12-15', 8, '12 45 00', 80000);
```

Employee Mananagement

```
create database EMP;
use EMP;
create table INFO ( Emp_Id int (5) primary key, Name
 varchar(20) , D_O_J date ,
Designation varchar (20),
 Salary int (6));

Insert into INFO values (77821, 'Rajesh' , '2010-05-01', 'Sales Head', 75000);
 Insert into INFO values(77822, 'Ram' , '2012-06-01', 'HR Head', 90000);
 Insert into INFO values (77823, 'Rohini' ,' 2010-01-01', 'Marketing Head', 80000);
 Insert into INFO values(77824, 'Rose' , '2009-02-01', 'CEO',95000);
```

The Programme

This is the code that will run the application. It is written in Python and linked to a MySQL user with the following details:**Host =** localhost

User=root

Password=welcome

In order to run this code on another system, the above details will have to be modi ed manually in lines 3, 53, 89 and 124.

```
import mysql.connector
#accessing Plane Management
def Table1() mycon = mysql.connector.connect(host='localhost',user='root',
    passwd='welcome',
database='PM')
    c=mycon.cursor()
    print("What would you like to do in this database")
    print("1. Update Record") print("2. New
    Record")   print("3.   Delete   Record")
```

```
    print("4. Show Records") print("5. Exit
    Program")          while          True:
    ans=int(input("Please  choose  no."))  if
    ans 1
            identity=int(input("Plane ID"))
            FuelHour=int(input("ENTER FUEL HOUR"))
            HoursF own=int(input("ENTER HOURS FLOWN")) STATUS=input("ENTER
            STATUS")
            x="update P_Management set Fuel_hour=" str(FuelHour)+" where
id=" str(identity)
            c.execute(x)
            y="update P_Management set Hours_F own=" str(HoursF own)+" where
id=" str(identity)
            c.execute(y)
            z="update P_Management set Status='" str(STATUS)+"' where
id=" str(identity)
            c.execute(z)
            mycon.commit()
        elif ans 2
        identity=int(input("PL
        ANE ID"))
            Model=input("ENTER MODEL")
            YoM=int(input("ENTER YEAR OF MANAFACTURING"))
            Capacity=int(input("ENTER CAPACITY"))
            ENDURANCE=int(input("ENTER ENDURANCE"))
            FuelHour=int(input("ENTER FUEL HOUR"))
            HoursF own=int(input("ENTER HOURS FLOWN"))
            STATUS=input("ENTER STATUS") x=f" insert
            into P_Management
values({identity},'{Model}',{YoM},{Capacity},{FuelHour},{ENDURANCE},{HoursF own},'{
STATUS}')"
            c.execute(x)
            mycon.commit()
        elif ans 3
            identity=int(input("PLANE ID TO DELETE FROM DATABASE"))
            x="delete from P_Management where id=" str(identity)
            c.execute(x) mycon.commit()
            print("SUCCESS")
        elif ans 4
            c.execute("select * from P_Management")
            x=c.fetchall()
            for u in x:

print(u[0],':',u[1],':',u[2],':',u[3],':',u[4],':',u[5],':',u[6],":",u[7])
        else: print("EXITING PROGRAM, THANK YOU") break
```

```
#accessing Passenger Details
def Table2()
    mycon=mysql.connector.connect(host='localhost',user='root',passwd='welcome',
database='PAX')
    c=mycon.cursor()
    print("What would you like to do in this database")
    print("1. New Record") print("2. Delete
    Record")   print("3.   Show   Records")
    print("4.  Exit  Program")  while  True:
    ans=int(input("Please  choose  no."))  if
    ans 1
            code=int(input("PAX CODE")) Fno=int(input("FLIGHT
            NO.")) classx=input("CLASS") name=input("NAME")
            phone=int(input("PHONE NO."))
            sex=input("sex")
            email=input("email")
            x=f"insert into details
values({code},{Fno},'{classx}','{name}',{phone},'{sex}','{email}')"
            c.execute(x)
            mycon.commit()
        elif ans 2
            code=int(input("Enter pax code"))
            x="delete from details where pax_code=" str(code) c.execute(x)
            mycon.commit()
        elif ans 3
            c.execute("select * from details")
            x=c.fetchall() for u in x:
            print(u[0],':',u[1],':',u[2],':',u[3],':',u[4],':',u[5],':',u[6])
            mycon.commit()
        else: print("EXITING PROGRAM, THANK
            YOU") break
#accessing F ight and Ticket Details def Table3()
mycon=mysql.connector.connect(host='localhost',user='root',passwd='welcome',
database='details')
    c=mycon.cursor()
    print("What would you like to do in this database")
    print("1. New Record") print("2. Delete
    Record")   print("3.   Show   Records")
    print("4.  Exit  Program")  while  True:
    ans=int(input("Please  choose  no."))  if
    ans 1
            flightid=input("F ight ID")
            Destination=input("Enter destination")
            Departure=input("Enter date") gateno=int(input("Enter
            gate no.")) time=input("Enter time formatted")
            ticketcost=int(input("Enter ticket cost"))
```

```
            x=f"insert into deets
values('{flightid}','{Destination}','{Departure}',{gateno},'{time}',{ticketcost})"
            c.execute(x)
            mycon.commit()
        elif ans 2
            code=input("Enter flight id")
            x="delete from deets where flightid='" str(code)+"'" c.execute(x)
            mycon.commit()
        elif ans 3
            c.execute("select * from deets")
            x=c.fetchall() for u in x:
            print(u[0],':',u[1],':',u[2],':',u[3],':',u[4],':',u[5])
            mycon.commit()
        else: print("EXITING PROGRAM, THANK
            YOU") break
```

#accessing Employee Management

```
def Table4()
    mycon=mysql.connector.connect(host='localhost',user='root',passwd='welcome',
database='emp')
    c=mycon.cursor()
    print("What would you like to do in this database")
    print("1. New Record") print("2. Delete
    Record")   print("3.   Show   Records")
    print("4.  Update  Records")  print("5.
    Exit     Program")     while     True:
    ans=int(input("Please  choose  no."))  if
    ans 1
            empid=int(input("ENTER emp id"))
            name=input("enter name") date=input("enter
            formatted date") designation=input("enter
            designation") salary=int(input("enter
            salary"))
            x=f"insert into info
values({empid},'{name}','{date}','{designation}',{salary})"
            c.execute(x)
            mycon.commit()
        elif ans 2
            code=int(input("Enter empid"))
            x="delete from info where emp_id=" str(code) c.execute(x)
            mycon.commit()
        elif ans 3
            c.execute("select * from info")
            x=c.fetchall() for u in x:
            print(u[0],':',u[1],':',u[2],':',u[3],':',u[4])
            mycon.commit()
        elif ans 4
```

```
            code=int(input("enter empid"))
            new1=input("enter new name") new2=input("enter
            new date") new3=input("enter new designation")
            new4=int(input("enter new salary")) x=f"update
            info set name='{new1}',
D_O_J='{new2}',designation='{new3}',salary={new4} where emp_id=" str(code)
            c.execute(x)
            mycon.commit()
        else: print("EXITING PROGRAM, THANK
YOU") break #Login Program while True:
      login={1 ['A','admin@A'],2 ['B','admin@B'],3 ['C','admin@C'],4 ['E','admin@E'],
5 ['F','admin@F']} print(' WELCOME
    TO ASCY ') print('
    LOGIN        ')
    user=input('Enter Password:')
    for k,v in login.items() if
    v[1] user:
            assign=k
            print('{}-Level Clearance granted'.format(v[0])) msg='''

Please select the database
1. PM
2. PAX
3. DETAILS
4. EMP
'''
      err='You do not have access to this database. Kindly press (0) to login and (1)
to exit the program.' print(msg)
    choice=int(input(' )) if choice 1 if
    assign 1 or assign 3 or assign 4
            Table1()
        else:
            print(err) f
            nal=int(input(' )) if
            f nal 0
                continue
            elif f nal 1 break
    elif choice 2 if assign 1 or assign 2 or
        assign 5
            Table2()
        else:
            print(err) f
            nal=int(input(' )) if
            f nal 0
                continue
            elif f nal 1 break
```

```
    elif choice 3 if assign 1 or assign 2 or
        assign 3
            Table3()
        else:
            print(err) f
            nal=int(input(' '))
            if f nal 0
                continue
            elif f nal 1 break
    elif choice 4 if assign 1 or
        assign 5
            Table4()
        else:
            print(err) f
            nal=int(input(' )) if
            f nal 0
                continue
            elif f nal 1 break
print('Thank you for using ASCY!')
```
