1) Create a python program on the scenario given below: Create an Employee table with attributes such as emp_ssn, emp_name, emp_category, gross_sal, basic_sal. Insert atleast three values in to the database. Demonstrate the database concepts for the following scenario: A company management wants to compute the net salary of each group of employee based on the category of the employee such as Category A, Category B, Category C. Compute the net salary based on the following table.

In [1]:

```python
import sqlite3
con = sqlite3.connect('XYZCOMPANY.db')
cur = con.cursor()
```

In [2]:

```python
try:
    cur.execute("""
        CREATE TABLE Employees(
            ssn INTEGER PRIMARY KEY,
            name TEXT,
            category TEXT,
            gross_sal REAL,
            basic_sal REAL
        )
    """)
except sqlite3.OperationalError as e:
    print(e)
```

```
table Employees already exists
```

In [3]:

```python
employees = [
    (1, 'Rohan', 'A', 'NULL', 1000),
    (2, 'Mohan', 'B', 'NULL', 1000),
    (3, 'Sohan', 'C', 'NULL', 1000),
]
```

In [4]:

```python
cur.executemany("INSERT INTO Employees VALUES (?, ?, ?, ?, ?)", employees)
```

Out[4]:

```
<sqlite3.Cursor at 0x1940738e490>
```

In [5]:

```python
con.commit()
```

In [6]:

```python
cur.execute("SELECT * FROM Employees")
empTuple = cur.fetchall()
```

In [7]:

```
for record in empTuple:
    ssn, name, category, g_sal, b_sal = record
    if category == 'A':
        da = 0.8 * b_sal
        g_sal = b_sal + da
        taxamt = 0.3 * g_sal
    elif category == 'A':
        da = 0.5 * b_sal
        g_sal = b_sal + da
        taxamt = 0.2 * g_sal
    else:
        da = 0.3 * b_sal
        g_sal = b_sal + da
        taxamt = 0.1 * g_sal
    finalsal = b_sal + da - taxamt
    cur.execute("UPDATE Employees SET gross_sal = ? where ssn = ?", (g_sal, ssn))
    print(name, 'with ssn', ssn, 'has net salary', finalsal)
```

```
Rohan with ssn 1 has net salary 1260.0
Mohan with ssn 2 has net salary 1170.0
Sohan with ssn 3 has net salary 1170.0
```

In [8]:

```
cur.execute("SELECT * FROM Employees")
empTuple = cur.fetchall()

for item in empTuple:
    print(*item, sep=", ") # VV IMP, instead of using for loop, u can use *item
```

```
1, Rohan, A, 1800.0, 1000.0
2, Mohan, B, 1300.0, 1000.0
3, Sohan, C, 1300.0, 1000.0
```

In [9]:

```
cur.close()
```

In [11]:

```
# cur.execute("Delete from employees")
# con.commit() # con.commit() and not cur.commit()
```

Implement Library management where students can borrow as well as donate books. Books table:

1 id INTEGER PRIMARY KEY 2 name TEXT 3 total_count INTEGER

a) Insert values to the table

34,king,5

123,Harry Potter,3

b) Update the table based on user inputs: based on book id I) BORROW II) RETURN

In [12]:

```
import sqlite3
con = sqlite3.connect("Library.db")
cur = con.cursor()
```

In [13]:

```
try:
    cur.execute("""
        CREATE TABLE Books (
            id INTEGER PRIMARY KEY,
            name TEXT,
            amt INTEGER
        )
    """)
except sqlite3.IntegrityError:
    print("DB already exists")
```

In [15]:

```
books = [
    (34, "king", 5),
    (123, "Harry Potter", 3)
]

cur.executemany("INSERT INTO Books VALUES (?, ?, ?)", books)
```

Out[15]:

```
<sqlite3.Cursor at 0x1940738ef10>
```

In [16]:

```
cur.execute("SELECT * FROM Books")
books = cur.fetchall()
# for book in books:
#     id_, name, amt = book
#     print(f"{id_}, {name}, {amt}")
for book in books:
    print(*book, sep=", ")
```

```
34, king, 5
123, Harry Potter, 3
```

In [24]:

```
while 1:
    print("\n\nWhat do you want to do \n 1. borrow \n 2. return \n 3. exit")
    opt = int(input("Enter option: "))
    if opt == 3:
        break
    book_id = int(input("Enter book id: "))
#     amt = int(input("Enter book amt: ")) # not necessary
    if opt == 1:
        cur.execute("UPDATE Books SET amt = amt - (?) WHERE id = (?)", (1, book_id))
        print(book_id, "Book borrowed")
        # instead of 1 it would be amt
    elif opt == 2:
        cur.execute("UPDATE Books SET amt = amt + (?) WHERE id = (?)", (1, book_id))
        print(book_id, "Book returned")
        # instead of 1 it would be amt, for many no of books
    else:
        print("Wrong Input")
```

```


What do you want to do 
 1. borrow 
 2. return 
 3. exit
Enter option: 2
Enter book id: 123
123 Book returned


What do you want to do 
 1. borrow 
 2. return 
 3. exit
Enter option: 2
Enter book id: 123
123 Book returned
```

In [25]:

```
cur.execute("SELECT * FROM Books")
books = cur.fetchall()
for book in books:
    print(*book, sep=", ")
```

```
34, king, 3
123, Harry Potter, 7
```

In [26]:

```
con.commit()
```

In [27]:

```
con.close()
```

In [ ]: