SQL ASSIGNMENT - LONDON METROPOLITAN POLICE DATABASE

(Antony Roy Nellangara, 22019057)

Introduction:

The London Metropolitan Police Database has been designed to store the details of the police officials who work for the department. The database contains 4 different tables with 4 different types of data about personal details, service details and health details. The database contains information about 1000 police officials who are employed under the London metropolitan police under different police stations and designations. The database has been designed keeping security, accessibility and different constraints in mind. This database design allows the database admin to query and perform various join operations very easily. The database also has no data redundancy. The database has 4 different tables and they are names, service_record, personal_details and health_details.

Database Generation using Python :

The database has been generated using Python with mainly numpy, pandas and random packages. In order to create 4 tables for the database, 4 pandas dataframes were created and were later converted and saved into csv files which will be uploaded into DB Browser for sqlite. The 4 dataframes (df_names, df_service_record, df_health_details, df_personal_details) represents the 4 tables and contains the data for the 4 tables in the database.

Step 1: Designing the database:

The 4 tables and the columns were selected after identifying the most useful and essential information that is related to police officials.

Table1: names:
Table names contains 2 columns. empid column contains 1000 unique employee ids and names column contains 1000 names of the police officials.

Table 2: service_record:
Table service_record contains 8 columns. empid column contains 1000 unique employee ids and rank column contains 11 ranks that are distributed among 1000 officials. Ranks are distributed in a way that the top ranks are given to a very less proportion of the total police force and the bottom ranks are given to a very large proportion of the police force. police_station column contains 31 police station names that is distributed among 1000 officials and office_timings column contains the time intervals that the 1000 police officials work for. age column contains age for 1000 police officials and no_of_arrests column contains number of arrests for each police officials. no_of_medals column contains number of medals won by each police official. Both no_of_arrests and no_of_medals are distributed in a way that the top ranking officials have a higher number of arrests and medals compared to the lower ranking officials. service_exp column contains number of years of service experience for each official.

Table 3: personal_details :

Table personal_details has 5 columns. empid column contains 1000 unique employee ids and names column contains 1000 names of the police officials. Email column has the email ids of 1000 officials. Mobileno column has the mobile numbers of 1000 officials. Address column has the residential postcodes of 1000 officials.

Table 4: health_record:
Table health_record has 5 columns. empid column contains 1000 unique employee ids. age column contains the age of the police officials. height column contains the height of the officials and weight column contains the weight of the officials. next_of_kin column has the names of the closest relative or friends who are the emergency contacts for each official.

Step 2 : Using Python to build the columns :

Python was used to create 4 csv files which will be later loaded into the database.

Libraries used :

Numpy : numpy package has been imported and numpy.random.choice function was used to assign 1000 unique ids for police officials, ranks for police officials and police stations for police officials. Numpy.random.randint function was used to assign mobile number, age , height, no_of_arrests, no_of_medals, height and weight values to the police officials.

Pandas : pandas dataframes were used to create 4 dataframes which has the columns of each the 4 tables. Later pandas has also been used to convert dataframes to csv files.

Names : get_full_names function under the names package was used to assign random generated names to the police officials

Random : shuffle function under the random packages was used to shuffle the employee id list of the police officials.

Table 1 : **Column names and their mode of creation :**

Table Name	Column Name	Mode of creation
names	emp_id	Numpy.random.choice() was used to create 1000 unique empids.
	names	get_full_names() function under names package was used to create 1000 names.
personal_details	empid	Numpy.random.choice was used to create 1000 unique empids.
	name	get_full_names() function under names package was used to create 1000 names.
	email	Email was generates with the names list which was generated above and appending the string '@gmail.com' at the end of every name and removing the spaces in the name.

	mobileno	Numpy.random.randit() was used to create 1000 ten digit mobile numbers.
	address	Address was generated unsing F strings in python.
health_record	empid	Numpy.random.choice() was used to create 1000 unique empids.
	age	Based on the rank of the police officers, age was assigned to each of them using the numpy.random.randint() function.
	height	Numpy.random.randit() was used to create 1000 values for heights between 140 and 200 centimeters.
	weight	Numpy.random.randit() was used to create 1000 values for weights between 50 kg and 120 kg.
	next_of_kin	Numpy.random.randit was used to create 1000 ten digit mobile numbers.
service_record	empid	Numpy.random.choice was used to create 1000 unique empids.
	rank	Based on a list containing the actual 11 ranks under the police department, and a list assigning the probabilities for each ranks, 1000 values for 11 ranks were created using numpy.random.choice().
	police_station	Based on a list containing 31 police station names in London, 1000 values for 31 stations were created using numpy.random.choice() function.
	office_timing	Based on a list containing different office timings, 1000 values for different timings were created using numpy.random.choice() function.
	age	Based on the ranks of the police officers, age is assigned to police officials using numpy.random.randint().
	no_of_arrests	Based on the ranks of the police officers, number of arrests is assigned to police officials using numpy.random.randint().
	no_of_medals	Based on the ranks of the police officers, number of medals is assigned to police officials using numpy.random.randint().
	service_exp	Based on the age and rank of the police officials, service_exp is assigned to the 1000 police officers using numpy.random.randint() function.

Python code used in the generation of dataframes and csv files :

```
#importing packages necessary for the program
import numpy as np
import pandas as pd
#importing package names which creates random names
import names as nm
import random as rd
```

```
#Declaring the number of rows for our database
n = 1000
#creating all the dataframes for 4 tables
df_names = pd.DataFrame()
df_service_record = pd.DataFrame()
df_personal_details = pd.DataFrame()
df_health_data= pd.DataFrame()
```

```
#creating 1000 random names for the database.
def names():
    global names
    names = []
    for i in range(0, n) :
        names.append(nm.get_full_name())


#creating 1000 unique employee ids for the database.
def empid():
    global empid
    empid = np.random.choice(10000000, n, replace = False)
    rd.shuffle(empid)


empid()
names()
```

```
#Creating the dataframe for names table
df_name = pd.DataFrame({
    'emp_id' : empid,
    'names' : names})


#Converting the dataframe to csv file
df_name.to_csv('C:/Users/anton/OneDrive/Desktop/names.csv', index = False)
```

```
#creating rank for 1000 police officers
def ranks():
    #Assigning all the ranks in the police department
    ranks = ['Commissioner', 'Deputy commissioner',  'Assistant commissioner',
             'Deputy assistant commissioner',  'Commander' ,
             'Chief superintendent', 'Superintendent' ,'Chief inspector',
             'Inspector', 'Police sergeant', 'Police constable' ]
    global ranks_data
    #Assigning the probabilities for each rank using p .
    ranks_data = np.random.choice(ranks, 1000,
    p = [0.01, 0.02, 0.035, 0.035, 0.05, 0.05, 0.1, 0.1, 0.15, 0.15, 0.3 ])

ranks()
```

```
def police_station():
    police_stations = ['26 Shepherds Bush Road', 'Acton Police Station',
    '2 Town Square', 'Bethnal Green Police Station', 'Bexleyheath Police Station',
    'Brixton Police Station', 'High Street', 'Charing Cross Police Station ',
    'Chingford Police Station', 'Colindale Police Station',
     'Croydon Police Station', 'Dagenham Police Station', 'Edmonton Police Station'
     'Forest Gate Police Station',
    'Harrow Police Station & Annexes' ,'Hayes Police Station',
    'Hounslow Police Station', 'Ilford Police Station' ,'Islington Police Station' ,
    'Kensington Police Station',
    'Kentish Town Police Station' ,'Kingston Police Station' ,
    'Lavender Hill Police Station',
    '43 Lewisham High Street' ,'Mitcham Police Station' ,
    'Plumstead Police Station',
    'Romford Police Station' ,'Stoke Newington Police Station',
     'Sutton Police Station' ,
    'Tottenham Police Station' ,'Twickenham Police Station' ,
    'Walworth Police Station',
    'Wembley Police Station' ,'Wimbledon Police Station' ,'88 Church Street']
    global stations_data
    stations_data = np.random.choice(police_stations, 1000)

police_station()
```

```
#Assigning office timings to the police officials
def office_timings():
    office_timing = ['12:00 - 07:00', '06:00 - 13:00', '13:00 - 19:00', '19:00 - 01:00']
    p = [0.2, 0.2, 0.3, 0.3]
    global office_timings_data
    office_timings_data = np.random.choice(office_timing, n, p)

office_timings()

#Creating the dataframe for service record
df_service_record = pd.DataFrame({
    'empid' : empid,
    'rank' : ranks_data,
    'police_staion' : stations_data,
    'office_timing' : office_timings_data
    })
```

```
#Function to assign ages to the police officers
def age():
    df_service_record['age'] = np.random.randint(20, 60 , n)
    #Assigning age to the top tier police officials
    df_service_record.loc[(df_service_record['rank'] == 'Commissioner') |
    (df_service_record['rank'] == 'Deputy commissioner') |
    (df_service_record['rank'] == 'Assistant commissioner') , 'age']
    =  np.random.randint(40,61,(len(df_service_record[(df_service_record['rank'] == 'Commissioner') |
    (df_service_record['rank'] == 'Deputy commissioner') |
    (df_service_record['rank'] == 'Assistant commissioner')])))


    #Assigning age for mid level officials
    df_service_record.loc[(df_service_record['rank'] == 'Deputy assistant commissioner') |
    (df_service_record['rank'] == 'Commander') |
    (df_service_record['rank'] == 'Chief superintendent') |
    (df_service_record['rank'] == 'Superintendent'), 'age']
    = np.random.randint(28,40, (len(df_service_record[(df_service_record['rank']
    == 'Deputy assistant commissioner') | (df_service_record['rank'] == 'Commander')
    | (df_service_record['rank'] == 'Chief superintendent')
     | (df_service_record['rank'] == 'Superintendent')])))


    #Assigning age for lower level officials
    df_service_record.loc[(df_service_record['rank'] == 'Chief inspector') |
    (df_service_record['rank'] == 'Inspector') | (df_service_record['rank'] == 'Police sergeant')
    | (df_service_record['rank']   == 'Police constable'), 'age']
    = np.random.randint(20,32, (len(df_service_record[(df_service_record['rank'] == 'Chief inspector') |
    (df_service_record['rank'] == 'Inspector') | (df_service_record['rank'] == 'Police sergeant') |
    (df_service_record['rank'] == 'Police constable')])))
    global age
    age = df_service_record['age'].tolist()
```

```
#Function to assign age to the police officials
def no_of_arrests():
    global no_of_arrests

    df_service_record['no_of_arrests'] = np.random.randint(0,150, n)
    #Assigning the number of arrests for top level officials
    df_service_record.loc[(df_service_record['rank'] == 'Commissioner') |
    (df_service_record['rank'] == 'Deputy commissioner') |
    (df_service_record['rank'] == 'Assistant commissioner') ,'no_of_arrests']
    = np.random.randint(100,150, (len(df_service_record[(df_service_record['rank'] == 'Commissioner') |
    (df_service_record['rank'] == 'Deputy commissioner') | (df_service_record['rank'] == 'Assistant commissioner')])))


    #Assigning the number of arrests for mid level officials
    df_service_record.loc[(df_service_record['rank'] == 'Deputy assistant commissioner') |
    (df_service_record['rank'] == 'Commander') | (df_service_record['rank'] == 'Chief superintendent') |
    (df_service_record['rank'] == 'Superintendent'), 'no_of_arrests'] =
    np.random.randint(50,100, (len(df_service_record[(df_service_record['rank']=='Deputy assistant commissioner') |
    (df_service_record['rank'] == 'Commander') | (df_service_record['rank'
    == 'Chief superintendent') | (df_service_record['rank'] == 'Superintendent')])))


    #Assigning the number of arrests for lower level officials
    df_service_record.loc[(df_service_record['rank'] == 'Chief inspector') |
    (df_service_record['rank'] == 'Inspector') | (df_service_record['rank'] == 'Police sergeant') |
    (df_service_record['rank'] == 'Police constable'), 'no_of_arrests'] =
    np.random.randint(0,50, (len(df_service_record[(df_service_record['rank'] == 'Chief inspector') |
    (df_service_record['rank'] == 'Inspector') | (df_service_record['rank'] == 'Police sergeant') |
    (df_service_record['rank'] == 'Police constable')])))
```

```
#Function to assign number of medals to the police officials
def no_of_medals():
    global no_of_medals
    df_service_record['no_of_medals'] = np.random.randint(0, 7, n)

    #Assigning the number of medals to the top tier police officials
    df_service_record.loc[(df_service_record['rank'] == 'Commissioner') |
    (df_service_record['rank'] == 'Deputy commissioner') |
    (df_service_record['rank'] == 'Assistant commissioner') , 'no_of_medals']
    = np.random.randint(5,7, (len(df_service_record[(df_service_record['rank'] == 'Commissioner') |
    (df_service_record['rank'] == 'Deputy commissioner') | (df_service_record['rank'] == 'Assistant commissioner')])))


    #Assigning the number of medals for mid level officials
    df_service_record.loc[(df_service_record['rank'] == 'Deputy assistant commissioner') |
    (df_service_record['rank'] == 'Commander') | (df_service_record['rank'] == 'Chief superintendent')
    | (df_service_record['rank'] == 'Superintendent'), 'no_of_medals'] =
    np.random.randint(3,6, (len(df_service_record[(df_service_record['rank']=='Deputy assistant commissioner') |
    (df_service_record['rank'] == 'Commander') | (df_service_record['rank'] ==
    'Chief superintendent') | (df_service_record['rank'] == 'Superintendent')])))


    #Assigning the number of arrests for lower level officials
    df_service_record.loc[(df_service_record['rank'] == 'Chief inspector') |
    (df_service_record['rank'] == 'Inspector') | (df_service_record['rank'] == 'Police sergeant') |
    (df_service_record['rank'] == 'Police constable'), 'no_of_medals'] =
    np.random.randint(0,3, (len(df_service_record[(df_service_record['rank'] == 'Chief inspector') |
    (df_service_record['rank'] == 'Inspector') | (df_service_record['rank'] == 'Police sergeant') |
    (df_service_record['rank'] == 'Police constable')])))

age()
```

```
#Function to assign service experience to the police officials
def service_experience():
    df_service_record['service_exp'] = np.random.randint(0, 40, n)
    for i in range(len(age)):
        df_service_record.iloc[i,6] = age[i] - 20
```

```
#Calling the above functions
no_of_arrests()
no_of_medals()
service_experience()

#Converting the dataframe to csv files
df_service_record.to_csv('C:/Users/anton/OneDrive/Desktop/service_record.csv', index = False)
```

```
#Function to assign email ids to the police officials
def emailid():
    global email
    email = []
    #print(names)
    #print(df_personal_details)
    for i in names:
        email.append(((i+"@gmail.com").replace(" ", "")).lower())
    #print(email)

#Function to assign mobile numbers to police officials
def mobileno():
    global mobile
    mobile = []
    for i in range(0,1000):
        a = []
        b = ''
        for i in range(0,10):
            a.append(np.random.randint(1,9))
        for i in a:
            b = b+str(i)
        mobile.append(b)
```

```
#Function to assign addresses to the police officials
def address():
    global address
    address = []
    postcodes1 = [f'AL{str(i).zfill(2)}' for i in range(1, 2000)]
    postcodes2 = [f'EG{str(i).zfill(2)}' for i in range(1, 2000)]
    postcodes3 = [f'KL{str(i).zfill(2)}' for i in range(1, 2000)]
    postcodes4 = [f'NI{str(i).zfill(2)}' for i in range(1, 2000)]
    postcodes5 = [f'DS{str(i).zfill(2)}' for i in range(1, 2000)]
    address.extend(np.unique(np.random.choice(postcodes1, 200)))
    address.extend(np.unique(np.random.choice(postcodes2, 200)))
    address.extend(np.unique(np.random.choice(postcodes3, 200)))
    address.extend(np.unique(np.random.choice(postcodes4, 500)))
    address.extend(np.unique(np.random.choice(postcodes5, 400)))
    address = address[:1000]
    rd.shuffle(address)


#Calling the above functions.
emailid()
mobileno()
address()
```

```
#Creating the dataframe for personal details
df_personal_details = pd.DataFrame({
    'empid' : empid,
    'name' : names,
    'email' : email,
    'mobileno' : mobile,
    'address' : address
    })

#Converting the dataframe to csv
df_personal_details.to_csv('C:/Users/anton/OneDrive/Desktop/personal_details.csv', index = False)
```

```
#Function to assign height to the police officials
def height():
    global height
    height = []
    height = (np.random.randint(140, 200, n))


#Function to assign weights to the police officials
def weight():
    global weight
    weight = []
    weight = np.random.randint(50, 120, n)


#Function to assign next of kin to the police officials
def next_of_kin():
    global next_of_kin
    next_of_kin = []
    for i in range(0, n) :
        next_of_kin.append(nm.get_full_name())
```

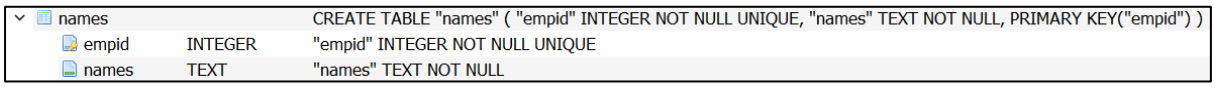
```
#Calling all the above functions
height()
weight()
next_of_kin()


#Creating the dataframe for health_data
df_health_data = pd.DataFrame({
    'empid' : empid,
    'age' : age,
    'height' : height,
    'weight' : weight,
    'next_of_kin' : next_of_kin
    })


#Converting the dataframe to csv
df_health_data.to_csv('C:/Users/anton/OneDrive/Desktop/health_record.csv', index = False)
```

The above python code generates the dataframes and will later convert them into csv files.

Step 3 : Creating a database and creating tables inside DB Browser for sqlite :

Open sqlite and create a new database called metropolitan_police_db and create 4 tables inside it.

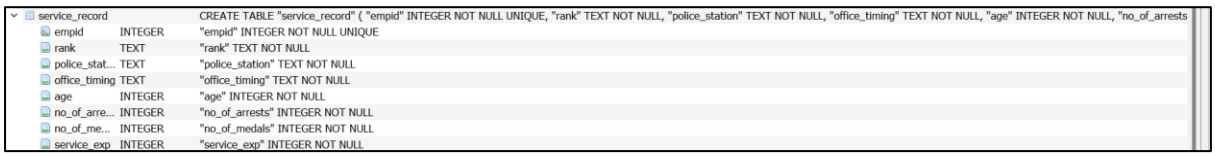
names :



names		CREATE TABLE "names" ("empid" INTEGER NOT NULL UNIQUE, "names" TEXT NOT NULL, PRIMARY KEY("empid"))
empid	INTEGER	"empid" INTEGER NOT NULL UNIQUE
names	TEXT	"names" TEXT NOT NULL

```
CREATE TABLE "names" ( "empid" INTEGER NOT NULL UNIQUE, "names" TEXT NOT NULL,
PRIMARY KEY("empid") )
```

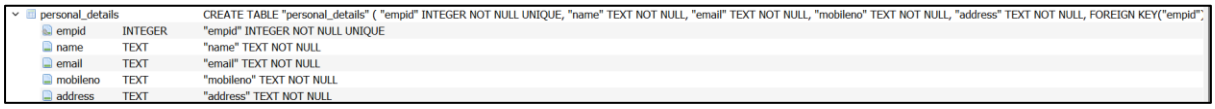
service_record :



service_record		CREATE TABLE "service_record" ("empid" INTEGER NOT NULL UNIQUE, "rank" TEXT NOT NULL, "police_station" TEXT NOT NULL, "office_timing" TEXT NOT NULL, "age" INTEGER NOT NULL, "no_of_arrests
empid	INTEGER	"empid" INTEGER NOT NULL UNIQUE
rank	TEXT	"rank" TEXT NOT NULL
police_stat...	TEXT	"police_station" TEXT NOT NULL
office_timing	TEXT	"office_timing" TEXT NOT NULL
age	INTEGER	"age" INTEGER NOT NULL
no_of_arre...	INTEGER	"no_of_arrests" INTEGER NOT NULL
no_of_me...	INTEGER	"no_of_medals" INTEGER NOT NULL
service_exp	INTEGER	"service_exp" INTEGER NOT NULL

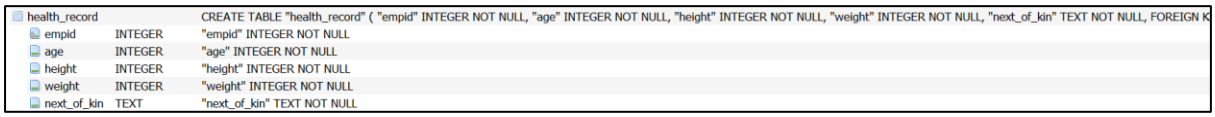
```
CREATE TABLE "service_record" ( "empid" INTEGER NOT NULL UNIQUE,
"rank" TEXT NOT NULL, "police_station" TEXT NOT NULL, "office_timing" TEXT NOT NULL,
"age" INTEGER NOT NULL, "no_of_arrests" INTEGER NOT NULL, "no_of_medals" INTEGER NOT NULL,
"service_exp" INTEGER NOT NULL , FOREIGN KEY("empid") REFERENCES "names"("empid"))
```

Personal_details: :




```
CREATE TABLE "personal_details" ( "empid" INTEGER NOT NULL UNIQUE,
 "name" TEXT NOT NULL, "email" TEXT NOT NULL, "mobileno" TEXT NOT NULL,
"address" TEXT NOT NULL, FOREIGN KEY("empid") REFERENCES "names"("empid") )
```

Health_record :




```
CREATE TABLE "health_record" ( "empid" INTEGER NOT NULL, "age" INTEGER NOT NULL,
"height" INTEGER NOT NULL, "weight" INTEGER NOT NULL,
"next_of_kin" TEXT NOT NULL, FOREIGN KEY("empid") REFERENCES "names"("empid") )
```

After the tables were created, the csv files that was created earlier can be uploaded into the tables above and the database is now ready to be used and queried.

Datatypes and constaints of every columns in the tables :

Table 2 : Column names, data types and constraints :

Table Name	Column Name	Data type	Constraints
names	emp_id	INTEGER	NOT NULL, UNIQUE, PRIMARY KEY
	names	TEXT	NOT NULL
personal_details	empid	INTEGER	NOT NULL, UNIQUE, FOREIGN KEY
	name	TEXT	NOT NULL
	email	TEXT	NOT NULL
	mobileno	TEXT	NOT NULL
	address	TEXT	NOT NULL
health_record	empid	INTEGER	NOT NULL, UNIQUE, FOREIGN KEY
	age	INTEGER	NOT NULL
	height	INTEGER	NOT NULL
	weight	INTEGER	NOT NULL
	next_of_kin	TEXT	NOT NULL
	empid	INTEGER	NOT NULL, UNIQUE, FOREIGN KEY
	rank	TEXT	NOT NULL

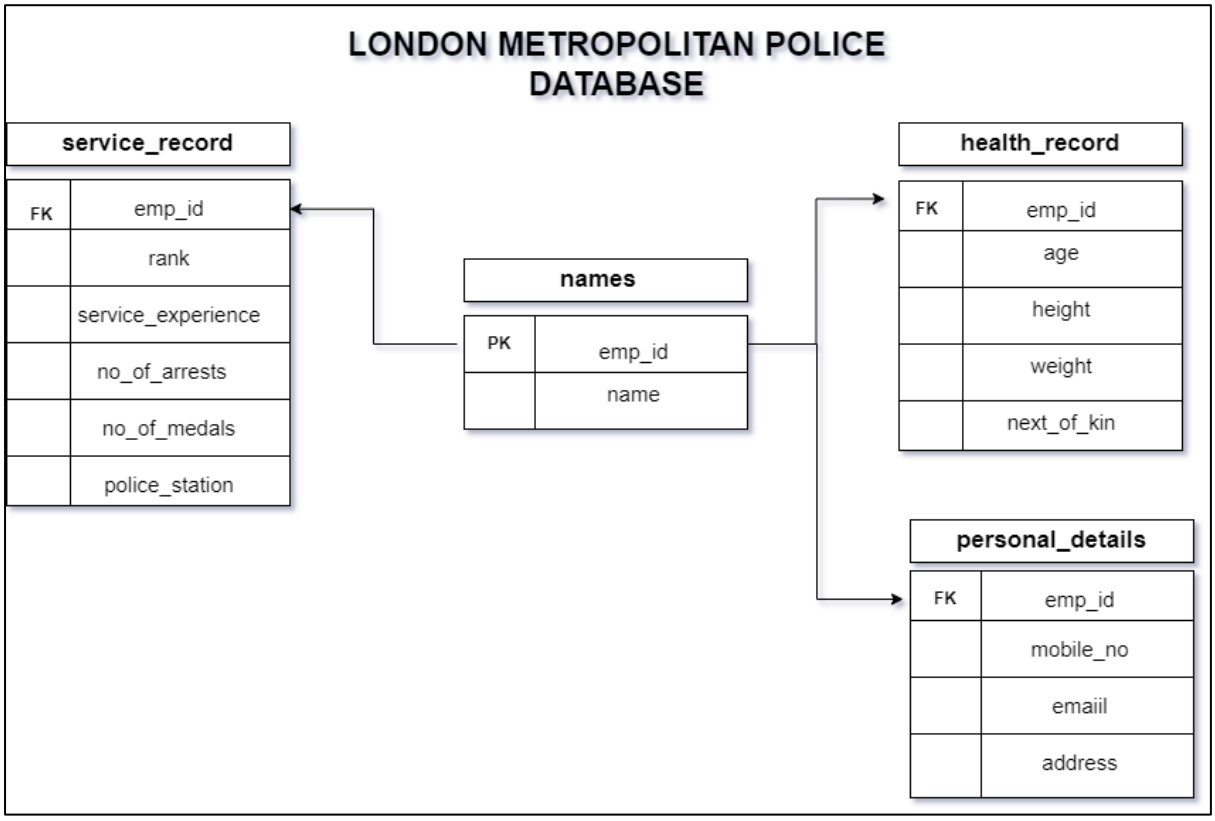
service_record	police_station	TEXT	NOT NULL
	office_timing	TEXT	NOT NULL
	age	INTEGER	NOT NULL
	no_of_arrests	INTEGER	NOT NULL
	no_of_medals	INTEGER	NOT NULL
	service_exp	INTEGER	NOT NULL

Column names and their data category :

Table 3 : Column names and their category of data :

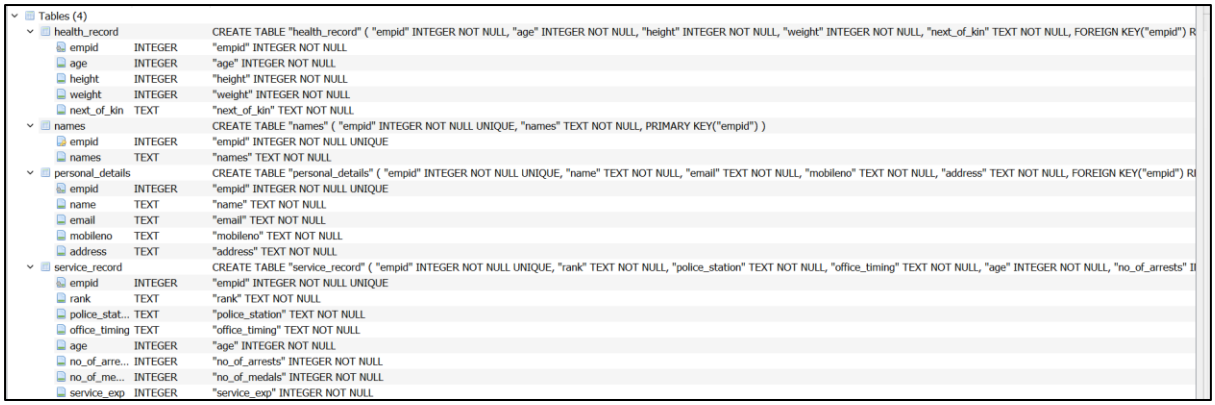
Table Name	Column Name	Nominal	Ordinal	Interval	Ratio
names	empid	✓			
names	names	✓			
service_record	empid	✓	✓		
service_record	rank		✓		
service_record	police_station	✓			
service_record	age				✓
service_record	no_of_arrests				✓
service_record	no_of_medals				✓
service_record	service_exp				✓
service_record	office_timing			✓	
personal_details	empid	✓	✓		
personal_details	name	✓			
personal_details	email	✓			
personal_details	mobileno	✓			
personal_details	address	✓			
health_record	empid	✓	✓		
health_record	age				✓
health_record	height				✓
health_record	weight				✓
health_record	next_of_kin	✓			

Database schema :



The above schema diagram shows how 4 tables are connected with each other using primary keys and foreign keys. Empid acts as the primary key in the names table. Empid acts as the foreign key in the servicea_record, health_record and personal_record tables.

Schema image from DB Browser for sqlite :



SQL QUERIES :

names table :

1.Displaying the first ten names alphabetically.

```
4   select * from names order by names limit 10;
```

	empid	names
1	1312497	Aaron Smolik
2	4625529	Adam Heiliger
3	9043918	Adam Ramirez
4	7892727	Adelina English
5	5350384	Adrian Rivera
6	2302307	Adriana Mitchell
7	6246965	Adriane Bomar
8	4248559	Agnes Hankins
9	3144270	Agnes Simmons
10	1623003	Agustin Enochs

2.Counting the number of names in the table.

```
4   select count(*) from names limit 10;
```

	count(*)
1	1000

3.Finding the names that has ‘Tony’ in their name.

```
4   select * from names where names like '%Tony%';
```

	empid	names
1	1264236	Tonya Pulley
2	2060728	Tony Bravo
3	3225300	Tony Hayes

Service_record table :

1.Displaying the details of commissioners in the department.

```
4   select * from service_record where rank = 'Commissioner';
```

	empid	rank	police_station	office_timing	age	no_of_arrests	no_of_medals	service_exp
1	2686518	Commissioner	Kingston Police Station	13:00 - 19:00	47	112	5	27
2	6952319	Commissioner	Croydon Police Station	12:00 - 07:00	55	120	5	35
3	9600881	Commissioner	Lavender Hill Police Station	12:00 - 07:00	54	138	6	34
4	1758406	Commissioner	Stoke Newington Police Station	19:00 - 01:00	47	110	6	27
5	1753054	Commissioner	Romford Police Station	13:00 - 19:00	55	126	5	35
6	2855300	Commissioner	Wembley Police Station	13:00 - 19:00	56	135	5	36

2.Counting the number of records in the table.

```
4   select count(*) from service_record;
```

	count(*)
1	1000

3.Finding out the details of the officer with most number of arrests.

```
4   select * from service_record where no_of_arrests = (select  max(no_of_arrests) from service_record);
```

	empid	rank	police_station	office_timing	age	no_of_arrests	no_of_medals	service_exp
1	8010718	Deputy commissioner	Mitcham Police Station	13:00 - 19:00	46	149	5	26

4.Finding out the personal details of the officer with service experience greater than 37 years by joining the personal_details table and the service_record table.

```
2   select a.*,  b.service_exp  from personal_details a
3   inner join service_record b
4   where a.empid = b.empid and b.service_exp > 37;
```

	empid	name	email	mobileno	address	service_exp
1	5139582	Nicholas Griffen	nicholasgriffen@gmail.com	7458163553	NI1394	40
2	7725661	Donald Mossman	donaldmossman@gmail.com	5142385731	KL272	38
3	4717734	Daniel Martin	danielmartin@gmail.com	3351121538	NI1906	40
4	5675360	Donna Haines	donnahaines@gmail.com	6261713677	NI516	40
5	243114	David Villanveva	davidvillanveva@gmail.com	1773131551	KL1999	38
6	1759498	Nancy Mays	nancymays@gmail.com	6786522674	NI1703	38
7	5564134	Gerry Stlouis	gerrystlouis@gmail.com	7453832366	NI1850	40
8	4248559	Agnes Hankins	agneshankins@gmail.com	1584218842	NI76	40

Health_record table :

1.Finding out the names of the tallest police officers.

```
2   select a.*, b.height from  names a
3   inner join health_record b
4   where a.empid = b.empid
5   order by b.height desc;
```

	empid	names	height
1	7324809	Virginia Gonzalez	199
2	1524417	Eugene Sowers	199
3	2686518	Chuck Wolverton	199
4	1180502	Stephen Barnes	199
5	8981880	Catherine Lopez	199
6	2927334	Kenneth Gagnon	199
7	5302904	Elizabeth Boland	199

2.Find out the youngest police officers and their service records.

```
select a.* from service_record a
inner join health_record b
where a.empid = b.empid
order by b.age asc;
```

	empid	rank	police_station	office_timing	age	no_of_arrests	no_of_medals	service_exp
1	3580990	Police constable	Brixton Police Station	06:00 - 13:00	20	21	0	0
2	712067	Police constable	Stoke Newington Police Station	12:00 - 07:00	20	29	2	0
3	4377821	Inspector	Kensington Police Station	19:00 - 01:00	20	18	0	0
4	6411725	Police constable	Kentish Town Police Station	06:00 - 13:00	20	33	1	0
5	9519027	Police constable	Kingston Police Station	06:00 - 13:00	20	39	1	0
6	8161651	Police constable	Kensington Police Station	12:00 - 07:00	20	18	0	0
7	3088293	Chief inspector	Chingford Police Station	13:00 - 19:00	20	16	1	0
8	9529004	Police constable	Ilford Police Station	13:00 - 19:00	20	20	1	0
9	387318	Police constable	Charing Cross Police Station	12:00 - 07:00	20	45	2	0
10	5298976	Police constable	Wembley Police Station	19:00 - 01:00	20	30	1	0

3.Find out the personal details of 5 most experienced police officers.

```
select a.* , b.age from personal_details a
inner join health_record b
where a.empid = b.empid and age =(select max(age) from health_record);
```

	empid	name	email	mobileno	address	age
1	5139582	Nicholas Griffen	nicholasgriffen@gmail.com	7458163553	NI1394	60
2	4717734	Daniel Martin	danielmartin@gmail.com	3351121538	NI1906	60
3	5675360	Donna Haines	donnahaines@gmail.com	6261713677	NI516	60
4	5564134	Gerry Stlouis	gerrystlouis@gmail.com	7453832366	NI1850	60
5	4248559	Agnes Hankins	agneshankins@gmail.com	1584218842	NI76	60

Personal_details table :

1.Find out the names and the contact details of the police officers who are Deputy Commissioners in the department.

```
select a.*, b.rank, b.police_station from personal_details a
inner join service_record b
where a.empid = b.empid and rank = 'Deputy commissioner';
```

	empid	name	email	mobileno	address	rank	police_station
1	6005822	Marjorie Wells	marjoriewells@gmail.com	4887777814	NI1866	Deputy commissioner	Kingston Police Station
2	4717734	Daniel Martin	danielmartin@gmail.com	3351121538	NI1906	Deputy commissioner	43 Lewisham High Street
3	9173944	Gayle Kurtz	gaylekurtz@gmail.com	7856523627	KL78	Deputy commissioner	Twickenham Police Station
4	5675360	Donna Haines	donnahaines@gmail.com	6261713677	NI516	Deputy commissioner	Hounslow Police Station
5	5372124	Marian Schaefer	marianschaefer@gmail.com	2666453657	EG1544	Deputy commissioner	High Street
6	4506081	Janet Matsuno	janetmatsuno@gmail.com	8535428875	AL1892	Deputy commissioner	26 Shepherds Bush Road
7	5296122	Pamela Unrein	pamelaunrein@gmail.com	6742125162	EG1280	Deputy commissioner	Chingford Police Station
8	6783800	Karen Martinez	karenmartinez@gmail.com	7223514758	EG373	Deputy commissioner	Harrow Police Station & Annexes
9	243114	David Villanveva	davidvillanveva@gmail.com	1773131551	KL1999	Deputy commissioner	Romford Police Station
10	6186328	Larry Banks	larrybanks@gmail.com	3323865743	NI620	Deputy commissioner	Croydon Police Station

2.Find out the details of the officers whose post code start with EG.

```
4   select * from personal_details where address like 'EG%';
```

	empid	name	email	mobileno	address
1	4996294	Judy Perry	judyperry@gmail.com	5254218121	EG1647
2	1718384	Elisha Rhodes	elisharhodes@gmail.com	8842156563	EG95
3	4996164	Larry Rudman	larryrudman@gmail.com	7717233538	EG529
4	2208403	Eric Marcoux	ericmarcoux@gmail.com	6714527138	EG475
5	7406905	Larry Styers	larrystyers@gmail.com	1445813374	EG774
6	3580990	Robin Cuthbert	robincuthbert@gmail.com	3554271776	EG1842
7	9881544	Kimberly Nicholas	kimberlynicholas@gmail.com	3173256683	EG578
8	5331587	Steven Cutwright	stevencutwright@gmail.com	6273462144	EG1207
9	7324809	Virginia Gonzalez	virginiagonzalez@gmail.com	2163584114	EG1504
10	5365833	Devon Roberts	devonroberts@gmail.com	1644438874	EG1961
11	5020878	Susanna Williams	susannawilliams@gmail.com	6417626522	EG338

3.Find out the details of the police officials who works as Police Constable in Kingston Police Station who has atleast 3 years of service experience.

```
2   select a.*, b.rank, b.police_station, b.service_exp from personal_details a
3   inner join service_record b
4   where a.empid = b.empid and b.police_station = 'Kingston Police Station'
5   and b.service_exp >= 3 and rank = 'Police constable';
```

	empid	name	email	mobileno	address	rank	police_station	service_exp
1	2633050	Thelma Morales	thelmamorales@gmail.com	3332223884	NI276	Police constable	Kingston Police Station	7
2	8690836	Jack Mcmonagle	jackmcmonagle@gmail.com	8812521354	NI935	Police constable	Kingston Police Station	4
3	4899374	Kathryn Beaton	kathrynbeaton@gmail.com	4742323742	KL1723	Police constable	Kingston Police Station	6
4	9027832	Maurice Perez	mauriceperez@gmail.com	4451425553	NI1193	Police constable	Kingston Police Station	5
5	9774845	Kenny Wilson	kennywilson@gmail.com	6323671833	AL166	Police constable	Kingston Police Station	4
6	9174166	William Randall	williamrandall@gmail.com	4118744416	NI874	Police constable	Kingston Police Station	6
7	9112148	Marie Juett	mariejuett@gmail.com	5755278332	EG1152	Police constable	Kingston Police Station	8

Ethical discussion and Justification :

The database has been designed for London Metropolitan Police department to manage the data of 1000 police officials. Database and tables were created in way respecting the confidentiality and security. Each of the 4 tables designed has their own specific purpose and this makes sure that no data is being misused. Only the admin of the database has access to all the tables and the admin can also limit access to tables that contains personal information. Access should be given to only the roles that work closely with the data that the table contains. This ensures that only the right roles have the right access to the right tables. Ethical practices can include providing the access to limited trusted roles.

Names table : This table contains 2 columns names and empid. The table was designed in a way that anybody can easily access the employee id and the names of the police officials. It is very important this table only contains 2 columns and no other personal data as this is meant to be accessed easily by the department officials just to identify the names of the police officials based on their employee id. Employee id also has to be unique for accurate identification of the employees.

Service_record table : This table contains 8 columns which are empid, rank, police_station, no_of_arrest, no_of_medals, service_exp, office_timing and age. This is a very important table as it is very important to track progress and record it frequently in this work of line. This table does not contain any personal or confidential data and is meant for easy access to the service record details of the officers. The 8 columns in the table clearly describes the service record of officers. In order to evaluate the performance of the officers, this table and the information can be accessed by just using the empids. Here ethical practices can include fact checking the service records and frequent updation of the records. Officers should be able to track their service record with ease and track only their service record. Higher officials should be given access to the service records of their subordinates.

Health_record table : This table contains 5 columns and they are empid, age, height, weight and next_of_kin. This table is really important as the profession they are involved is dangerous and challenging and health data is really important. In case of any emergency, the health data can be used for identification. The data has to be accurate and secure and updated regularly.

Personal_details table : This table contains 5 columns and they are empid, name, email, mobileno and address. This table should be treated carefully and the access to this table should be only given the to people with the right role. This table contains personal and confidential information about the police officers and can be misused. Storing personal details separately in a different table ensures that the data stays confidential and that it does not fall in the wrong hands as this data can be easily misused. Ethical practices can include ensuring the access to this table is with the right people, fact checking the details in the table, asking the individual police officers to confirm their personal details.

The 4 tables created contains all the necessary details of the police officers.