VISA APPLICATION DATABASE

Vaibhav Ramakrishnan

I. Introduction

1.1. Project purpose

The project aims to design a robust and user-friendly database system using Unified Modeling Language (UML) to create a conceptual design and the Relational Database Model for the logical design. By leveraging these tools, I intend to ensure the database's integrity, flexibility, and scalability, thereby optimizing its performance for both current and future requirements.

The proposed database system will be specifically tailored to handle visa applications, ensuring that each applicant's data is accurately recorded and easily accessible. We then implement SQL queries on this database, which allow us to efficiently retrieve relevant application information, such as applicant details, visa types, processing status, and historical data.

1.2. Data structure

In many developing countries, Applicants need to submit Visa Applications in the U.S. Consulate General for applying to the U.S. visa. The U.S. Officers (aka Officers) will process the applications and decide to approve or reject. Each applicant has only one valid visa, meaning that if they applied for both visa types and got approval, they would only grant one visa. Visas granted after meeting the requirements will be printed and delivered to Applicants through Courier.

Each Applicant is represented.

- The information about an applicant is passport number (Passport_no : VARCHAR(20)) which is unique within the system, the name (Name: VARCHAR (50)), data of birth (DOB: Date), address (Address: VARCHAR(100)), and the nationality (Nationality: VARCHAR(50)) of the applicant.
- An applicant has zero to many dependants.
- One applicant can apply to many visas' applications.
- Each applicant can have only one valid visa at a time.
- Each applicant corresponds to one courier.

Each Dependent (if any) is represented.

- The information about a dependent is his or her passport number (Dependent_passport_num: VARCHAR(20)) which is unique within an applicant, date of birth (DOB: Date), relationship to the main applicant (Relationship: VARCHAR(50)), and name (Name: VARCHAR(50)) of a dependent.

Each Visa_application is represented.

- The information about a visa application is the application id (Application_id: INT) which is unique within the system, the location where an application is proceed (location: VARCHAR(50)), and the date it is created (application_date: DATE).
- Visa application consists of Immigrant and Migrant visa type.
- Visa applications are processed by Officers, 1 Officer can process many applications.

- After processing, visa applications can be rejected or approved. If it is approved, visa information is recorded.
- Each visa application will become only one valid visa.

Each Immigrant application is represented.

- The information about an immigrant application is the visa class (Visa_class: VARCHAR(20)) and the purpose of the trip (Purpose: VARCHAR(50)).
- The Immigrant application is unique defined by the visa application number.

Each Migrant application is represented.

- The information about a migrant application is the visa class (Visa_class: VARCHAR(20)) and the Petitioner (Petitioner: VARCHAR(50)) of the application.
- The Migrant application is unique defined by the visa application number.

Each Visa is represented.

- The information about a visa is the visa number (Visa_no: INT) which is unique within the system, approved date (Approved_date:DATE), expired date (Expired_date: DATE), and the visa class (Visa_class: VARCHAR(20)).
- A visa is issued by an officer.
- A visa is printed by a local employee.
- After printing, visa is shipped to courier. One visa is delivered by one courier.
- One visa is gained by one applicant.

Each Officer is represented.

- The information about a U.S. Officer is Officer id (Officer_id: INT) which is unique within the system, name (VARCHAR(50)), email (VARCHAR(50)), department (VARCHAR(50)), and title (VARCHAR(50)).
- Every Officer needs to report to other Officers in the same department or in different departments.
- One Officer can be a manager of zero or many Officers.
- One Officer can issue many visas.
- One Officer can be a manager of many local employees. It is possible that an Officer does not mange anyone.

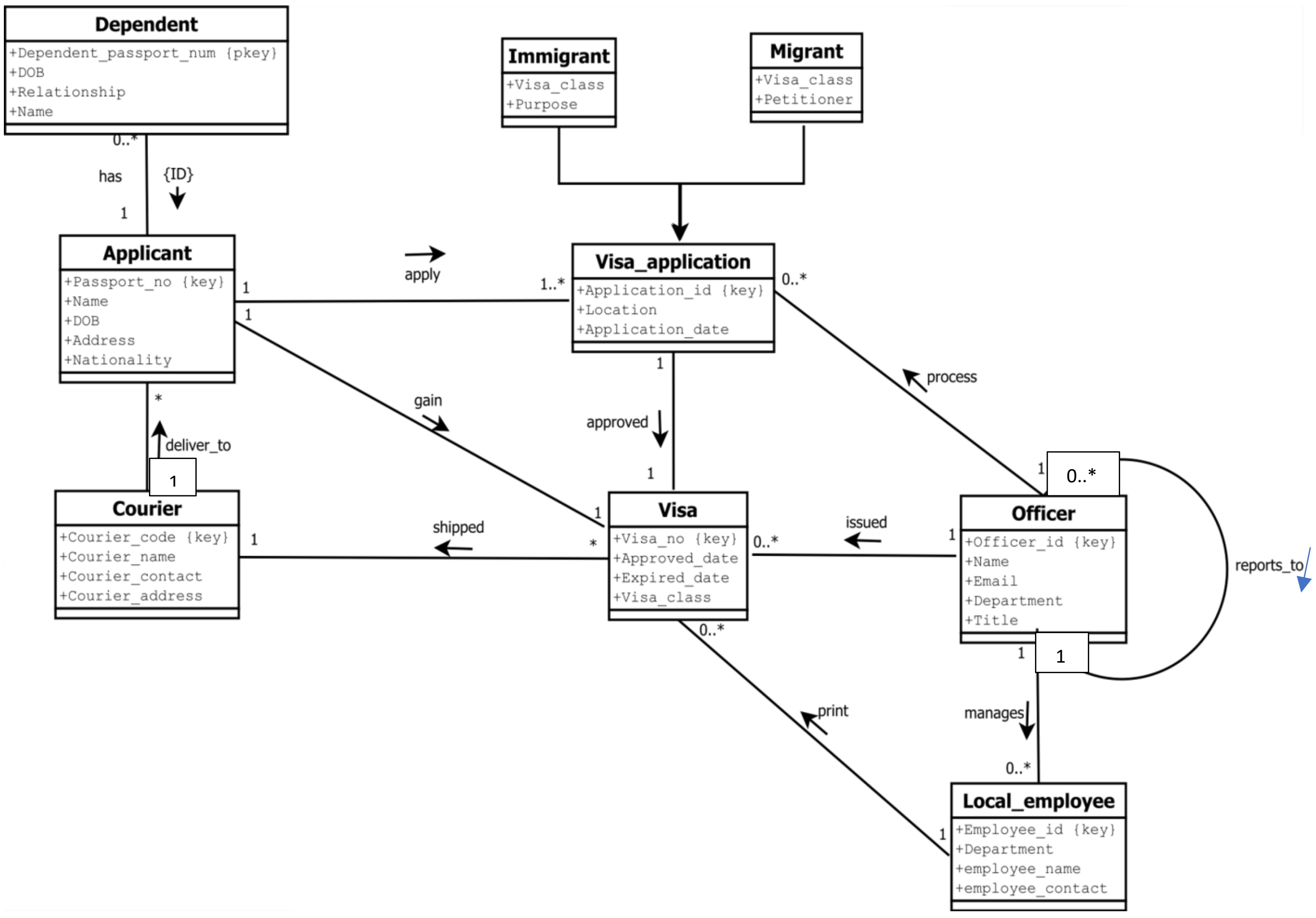
Each Local_employee is represented.

- The information about a local employee is employee id (Employee_id: INT) which is unique within the system, department (Department: VARCHAR(50)), name (employee_name: VARCHAR(50)), and contact (employee_contact: VARCHAR(50)) of a local employee.
- A local employee is managed by an Officer.
- A local employee prints visa after approval, many visas can be handled by one local employee.

Each Courier is represented.

- The information about a courier is courier code (Courier_code: INT) which is unique within the system, courier name (Courier_name: VARCHAR(50)), contact (Courier_contact: VARCHAR(50)), and address (Courier_address: VARCHAR(100)) of a courier.
- One Courier can deliver many visas to many applicants.

II. Creating UML of our Visa Database



III. Convert the UML to Database Schemas

Applicant (<u>Passport_no</u>, Name, DOB, Address, Nationality, Courier_code)

```
                    PK                                    FK
```

- **Passport_no** is the Primary Key.
- **Courier_code** is the Foreign Key references Courier.

```
Visa_application ( Application_id,  Location, Application_date, Passport_no, Officer_id )
                                                                 ---------------------------------
                    PK                                                     FK
```

- **Application_id** is the Primary Key.
- **Passport_no** is the Foreign Key references Applicant.
- **Officer_id** is the Foreign Key references Officer.

```
Dependent ( Dependent_passport_num, Passport_no, Name, DOB, Relationship)
                                    ---------------
            PK                           PK, FK
```

- **Dependent_passport_num** and **Passport_no** is the Primary Key.
- **Passport_no** is the Foreign Key references Applicant.

```
Immigrant ( Application_id, Visa_class, Purpose)
           --------------------
            PK, FK
```

- **Application_id** is the Primary Key and the Foreign Key references Visa_application.

```
Migrant ( Application_id, Visa_class, Petitioner)
        --------------------
            PK, FK
```

- **Application_id** is the Primary Key and the Foreign Key references Visa_application.

```
Courier ( Courier_code, Courier_name, Courier_contact, Courier_address )
               PK
```

- **Courier_code** is the Primary Key.

```
Visa ( Visa_no, Approved_date, Expired_date, Visa_class, Officer_id, Courier_code, Employee_id )
                                                         ----------------------------------------------------
        PK                                                                FK
```

- **Visa_no** is the Primary Key.
- **Officer_id** is the Foreign Key references Officer.
- **Courier_code** is the Foreign Key references Courier.
- **Employee_id** is the Foreign Key references Local_employee

Officer (Officer_id, Name, Email, Department, Title)
PK

- **Officer_id** is the Primary Key.

Reports_to (supervisor_id, subordinate_id, Officer_id)

PK FK

- **supervisor_id** and **subordinate_id** are the Primary Keys.
- **Officer_id** is the Foreign Key references Officer.

Local_employee (Employee_id, Department, employee_name, employee_contact, Officer_id)

PK FK

- **Employee_id** is the Primary Key.
- **Officer_id** is the Foreign Key references Officer.

IV. Design necessary queries

(a) Three queries for one table

We will create 3 queries for the Applicant Table-

- Retrieve all applicants of a specific Nationality.
- Retrieve the address of an applicant with a given Passport_no.
- Retrieve the count of dependents for respective applicants.

(b) Three queries for two tables.

We will create 3 queries for the Applicant and Visa_application tables-

- Retrieve all visa applications submitted by an applicant.
- Retrieve the count of visa applications for visa applicants.
- Retrieve the most recent application date for visa applicants.

(c) Three queries for three tables.

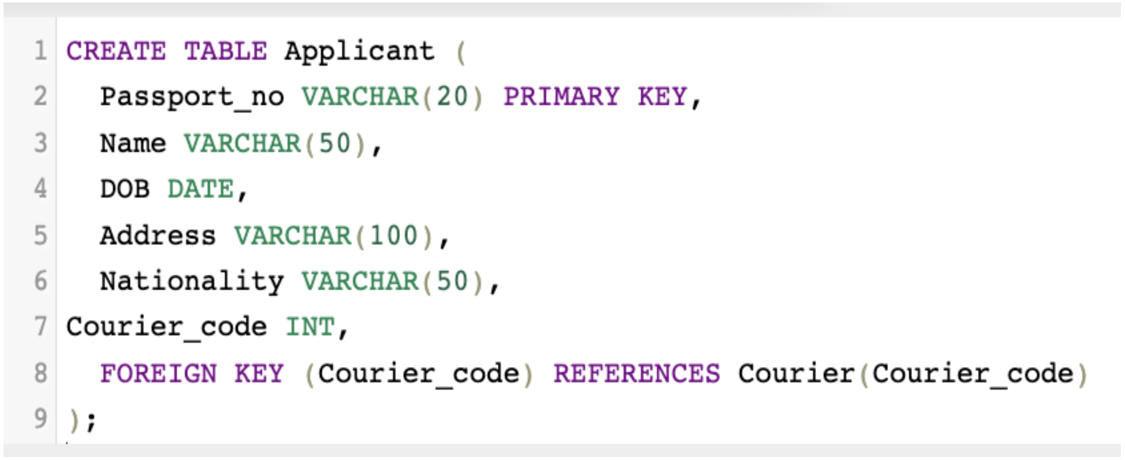
We will create 3 queries for the Visa, Officer, and Local_Employee tables-

- Retrieve all approved visas for a specific officer using the Visa, Officer, and Local_Employee tables.
- Retrieve the number of approved visas for each officer using the Officer, Local_Employee, and Visa tables.
- Retrieve the Visa_no and approved_date for each approved visa along with the corresponding officer's name using the Visa, Officer, and Local_Employee tables.

V. Write necessary SQL commands to create tables and input data to each table.

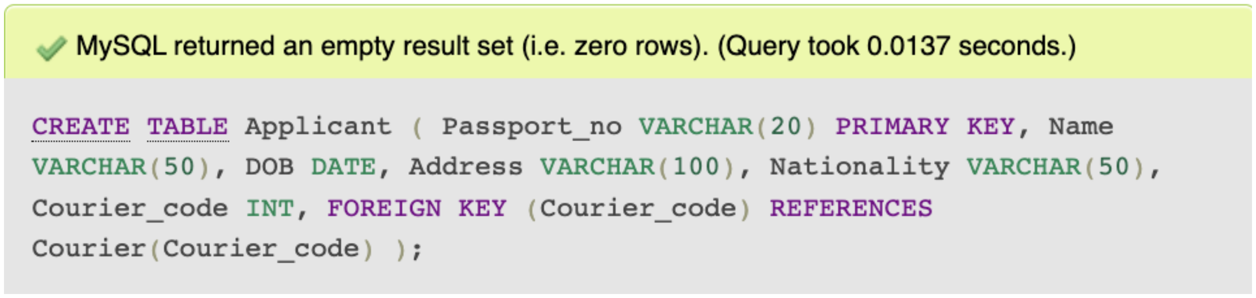
5.1. Create tables

```
CREATE TABLE Applicant (

 Passport_no VARCHAR(20) PRIMARY KEY,

 Name VARCHAR(50),

 DOB DATE,

 Address VARCHAR(100),

 Nationality VARCHAR(50),

 Courier_code INT,

 FOREIGN KEY (Courier_code) REFERENCES Courier(Courier_code)

);
```



```
CREATE TABLE Applicant (
  Passport_no VARCHAR(20) PRIMARY KEY,
  Name VARCHAR(50),
  DOB DATE,
  Address VARCHAR(100),
  Nationality VARCHAR(50),
Courier_code INT,
  FOREIGN KEY (Courier_code) REFERENCES Courier(Courier_code)
);
```

Output-



```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0137 seconds.)

CREATE TABLE Applicant ( Passport_no VARCHAR(20) PRIMARY KEY, Name
VARCHAR(50), DOB DATE, Address VARCHAR(100), Nationality VARCHAR(50),
Courier_code INT, FOREIGN KEY (Courier_code) REFERENCES
Courier(Courier_code) );
```

CREATE TABLE Courier (

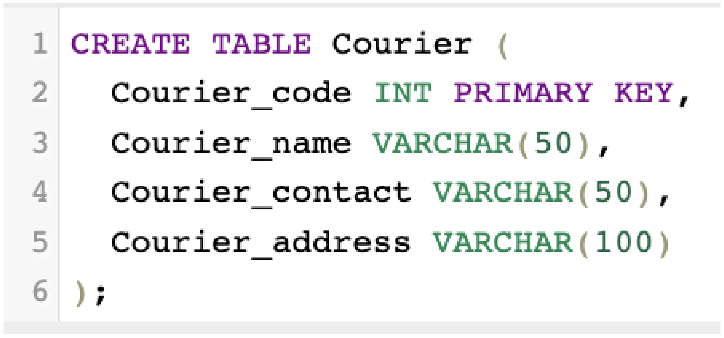
Courier_code INT PRIMARY KEY,

Courier_name VARCHAR(50),

Courier_contact VARCHAR(50),

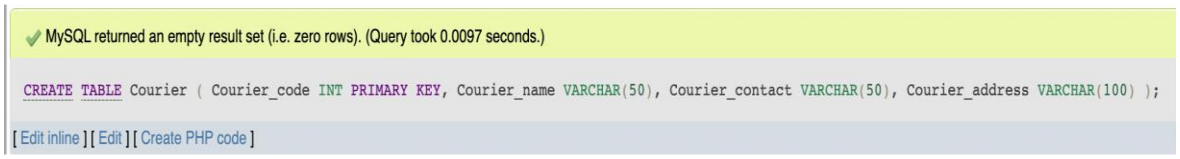
Courier_address VARCHAR(100)

);



```
CREATE TABLE Courier (
  Courier_code INT PRIMARY KEY,
  Courier_name VARCHAR(50),
  Courier_contact VARCHAR(50),
  Courier_address VARCHAR(100)
);
```

<u>Output-</u>



```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0097 seconds.)

CREATE TABLE Courier ( Courier_code INT PRIMARY KEY, Courier_name VARCHAR(50), Courier_contact VARCHAR(50), Courier_address VARCHAR(100) );
```

CREATE TABLE Dependent (

Dependent_passport_num VARCHAR(20),

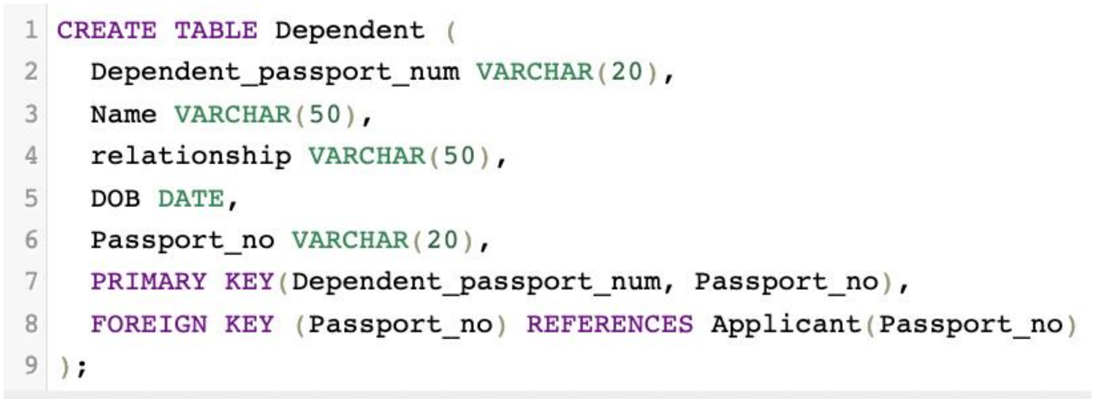
Name VARCHAR(50),

relationship VARCHAR(50),

DOB DATE,

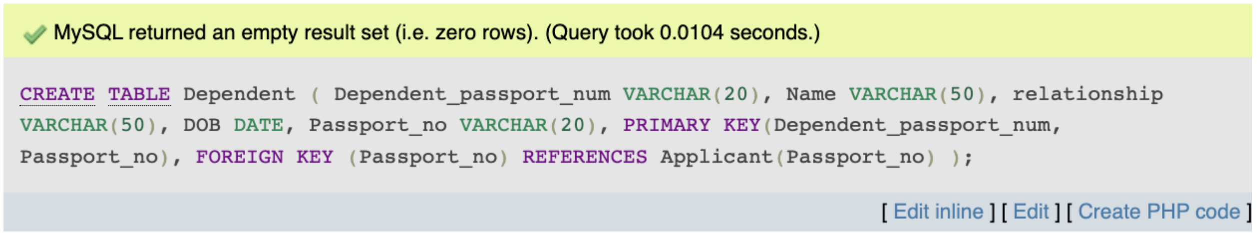
Passport_no VARCHAR(20),

```
 PRIMARY KEY(Dependent_passport_num, Passport_no),
 FOREIGN KEY (Passport_no) REFERENCES Applicant(Passport_no)
);
```



```
CREATE TABLE Dependent (
  Dependent_passport_num VARCHAR(20),
  Name VARCHAR(50),
  relationship VARCHAR(50),
  DOB DATE,
  Passport_no VARCHAR(20),
  PRIMARY KEY(Dependent_passport_num, Passport_no),
  FOREIGN KEY (Passport_no) REFERENCES Applicant(Passport_no)
);
```

<u>Output-</u>

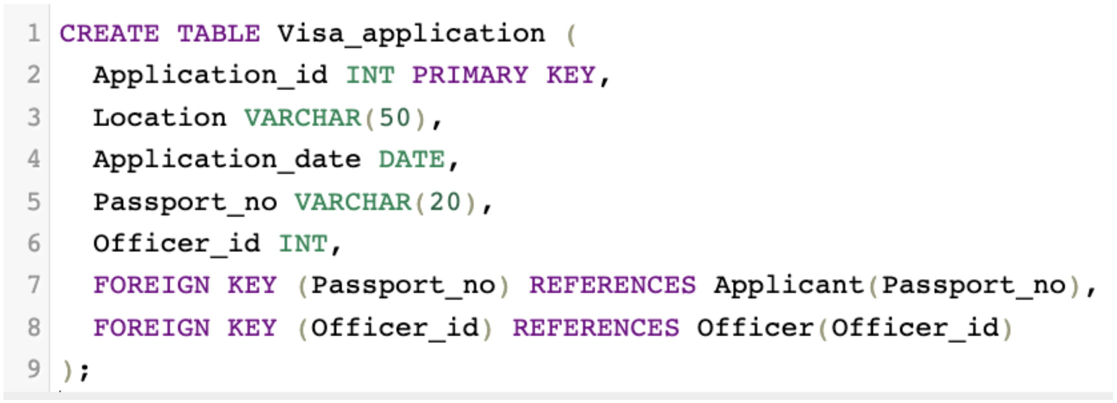


MySQL returned an empty result set (i.e. zero rows). (Query took 0.0104 seconds.)

```
CREATE TABLE Dependent ( Dependent_passport_num VARCHAR(20), Name VARCHAR(50), relationship
VARCHAR(50), DOB DATE, Passport_no VARCHAR(20), PRIMARY KEY(Dependent_passport_num,
Passport_no), FOREIGN KEY (Passport_no) REFERENCES Applicant(Passport_no) );
```

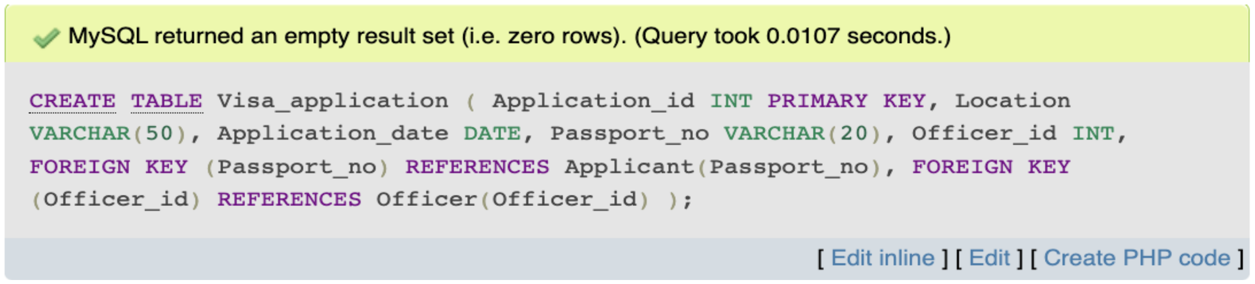
[Edit inline] [Edit] [Create PHP code]

```
CREATE TABLE Visa_application (
 Application_id INT PRIMARY KEY,
 Location VARCHAR(50),
 Application_date DATE,
 Passport_no VARCHAR(20),
 Officer_id INT,
 FOREIGN KEY (Passport_no) REFERENCES Applicant(Passport_no),
 FOREIGN KEY (Officer_id) REFERENCES Officer(Officer_id)
);
```



```sql
CREATE TABLE Visa_application (
  Application_id INT PRIMARY KEY,
  Location VARCHAR(50),
  Application_date DATE,
  Passport_no VARCHAR(20),
  Officer_id INT,
  FOREIGN KEY (Passport_no) REFERENCES Applicant(Passport_no),
  FOREIGN KEY (Officer_id) REFERENCES Officer(Officer_id)
);
```

<u>Output-</u>



```
✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0107 seconds.)

CREATE TABLE Visa_application ( Application_id INT PRIMARY KEY, Location
VARCHAR(50), Application_date DATE, Passport_no VARCHAR(20), Officer_id INT,
FOREIGN KEY (Passport_no) REFERENCES Applicant(Passport_no), FOREIGN KEY
(Officer_id) REFERENCES Officer(Officer_id) );
```

CREATE TABLE Immigrant (

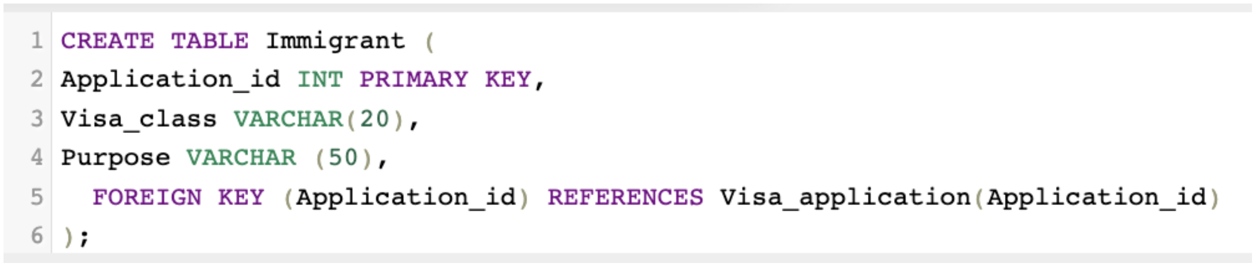
Application_id INT PRIMARY KEY,

Visa_class VARCHAR(20),

Purpose VARCHAR (50),

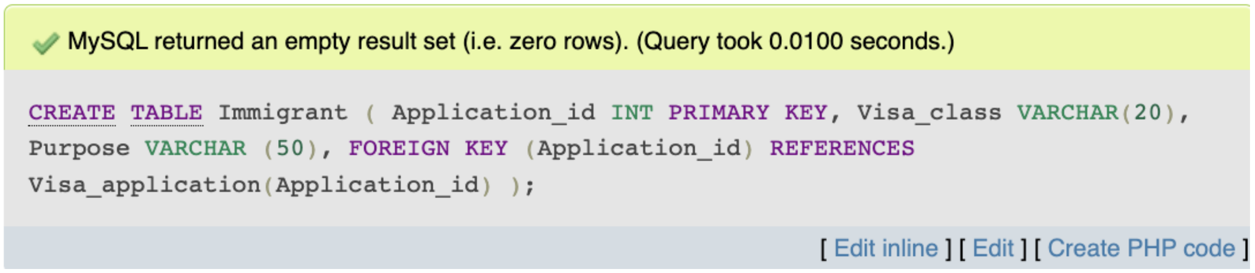
FOREIGN KEY (Application_id) REFERENCES Visa_application(Application_id)

);



```sql
CREATE TABLE Immigrant (
Application_id INT PRIMARY KEY,
Visa_class VARCHAR(20),
Purpose VARCHAR (50),
  FOREIGN KEY (Application_id) REFERENCES Visa_application(Application_id)
);
```

<u>Output-</u>



```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0100 seconds.)

CREATE TABLE Immigrant ( Application_id INT PRIMARY KEY, Visa_class VARCHAR(20),
Purpose VARCHAR (50), FOREIGN KEY (Application_id) REFERENCES
Visa_application(Application_id) );
```

CREATE TABLE Migrant (

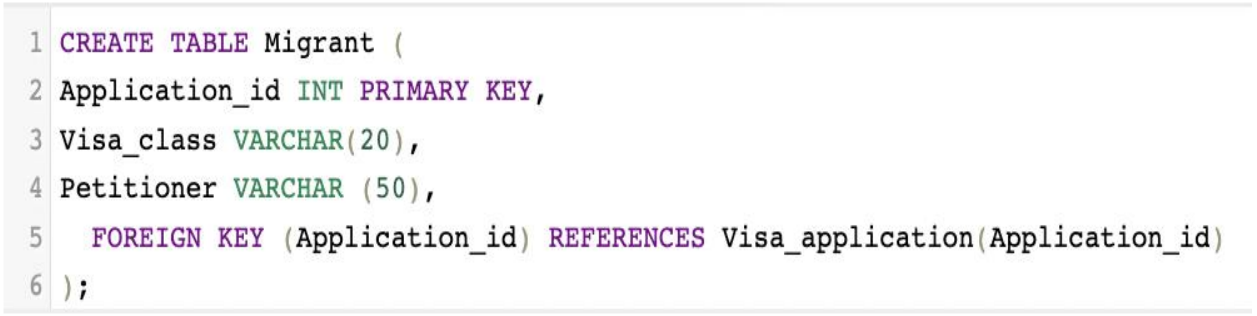
Application_id INT PRIMARY KEY,

Visa_class VARCHAR(20),

Petitioner VARCHAR (50),

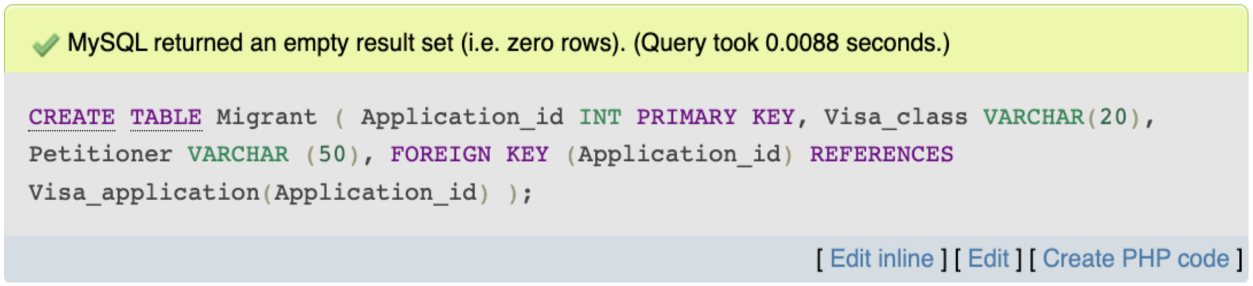
 FOREIGN KEY (Application_id) REFERENCES Visa_application(Application_id)

);



```
CREATE TABLE Migrant (
Application_id INT PRIMARY KEY,
Visa_class VARCHAR(20),
Petitioner VARCHAR (50),
  FOREIGN KEY (Application_id) REFERENCES Visa_application(Application_id)
);
```

<u>Output-</u>



```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0088 seconds.)

CREATE TABLE Migrant ( Application_id INT PRIMARY KEY, Visa_class VARCHAR(20),
Petitioner VARCHAR (50), FOREIGN KEY (Application_id) REFERENCES
Visa_application(Application_id) );
```

CREATE TABLE Visa (

 Visa_no INT PRIMARY KEY,

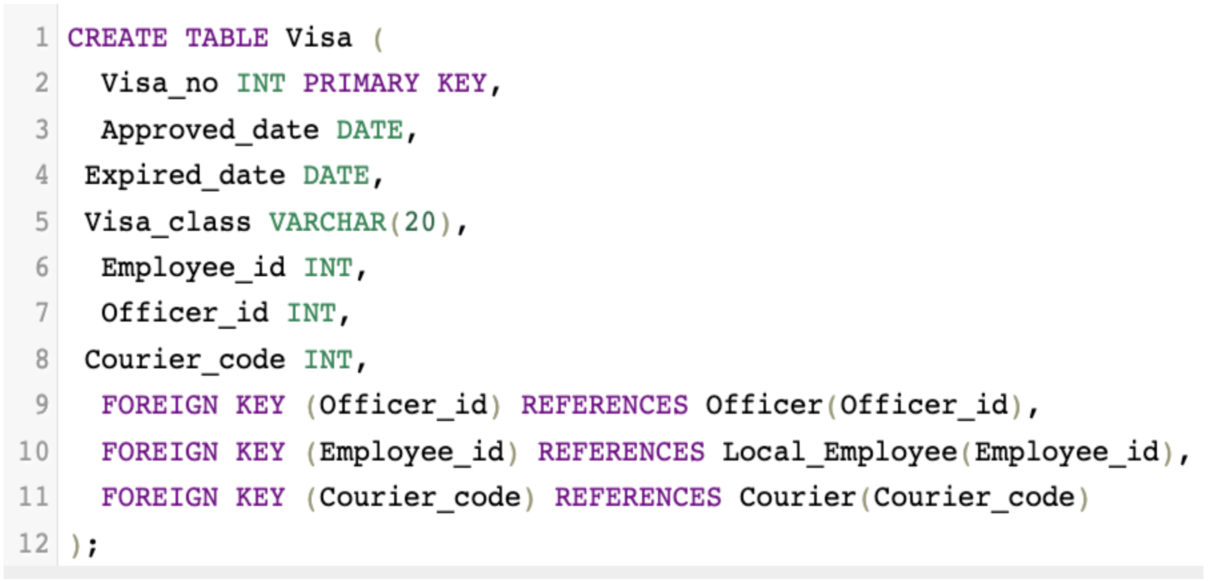
 Approved_date DATE,

 Expired_date DATE,

 Visa_class VARCHAR(20),

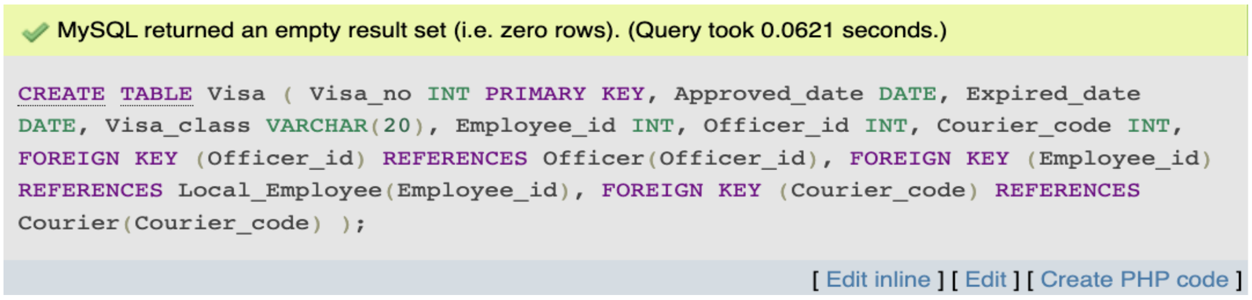
 Employee_id INT,

```
 Officer_id INT,
Courier_code INT,
 FOREIGN KEY (Officer_id) REFERENCES Officer(Officer_id),
 FOREIGN KEY (Employee_id) REFERENCES Local_Employee(Employee_id),
 FOREIGN KEY (Courier_code) REFERENCES Courier(Courier_code)
);
```



```
CREATE TABLE Visa (
  Visa_no INT PRIMARY KEY,
  Approved_date DATE,
 Expired_date DATE,
 Visa_class VARCHAR(20),
  Employee_id INT,
  Officer_id INT,
 Courier_code INT,
  FOREIGN KEY (Officer_id) REFERENCES Officer(Officer_id),
  FOREIGN KEY (Employee_id) REFERENCES Local_Employee(Employee_id),
  FOREIGN KEY (Courier_code) REFERENCES Courier(Courier_code)
);
```

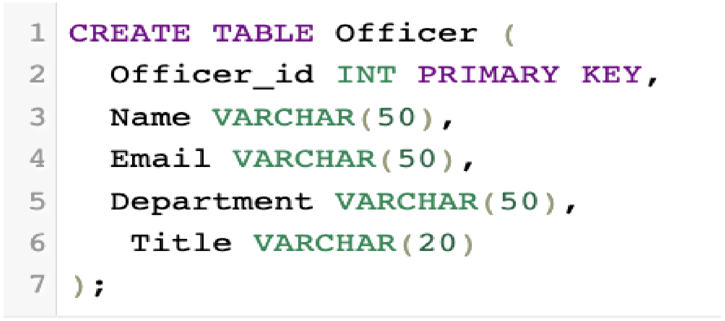
<u>Output-</u>



```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0621 seconds.)

CREATE TABLE Visa ( Visa_no INT PRIMARY KEY, Approved_date DATE, Expired_date
DATE, Visa_class VARCHAR(20), Employee_id INT, Officer_id INT, Courier_code INT,
FOREIGN KEY (Officer_id) REFERENCES Officer(Officer_id), FOREIGN KEY (Employee_id)
REFERENCES Local_Employee(Employee_id), FOREIGN KEY (Courier_code) REFERENCES
Courier(Courier_code) );
```

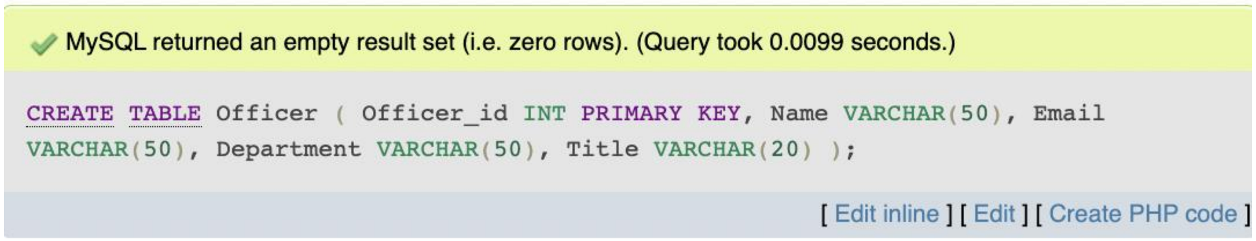
```
CREATE TABLE Officer (
 Officer_id INT PRIMARY KEY,
 Name VARCHAR(50),
 Email VARCHAR(50),
 Department VARCHAR(50),
  Title VARCHAR(20)
```

);



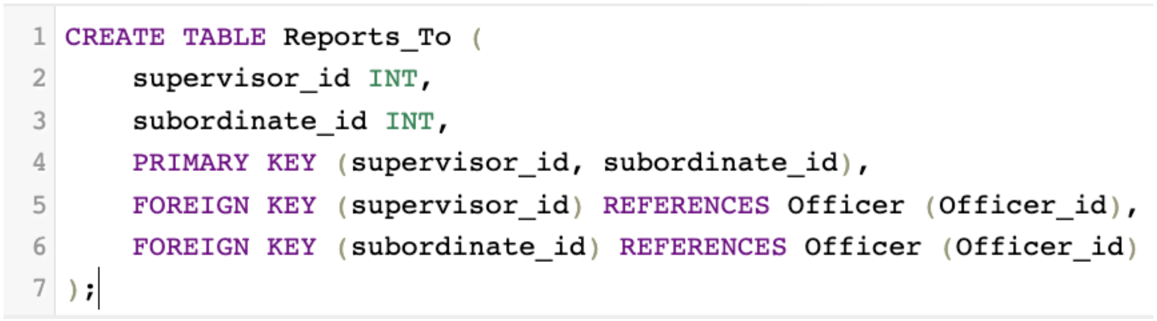
```
CREATE TABLE Officer (
  Officer_id INT PRIMARY KEY,
  Name VARCHAR(50),
  Email VARCHAR(50),
  Department VARCHAR(50),
   Title VARCHAR(20)
);
```

<u>Output-</u>



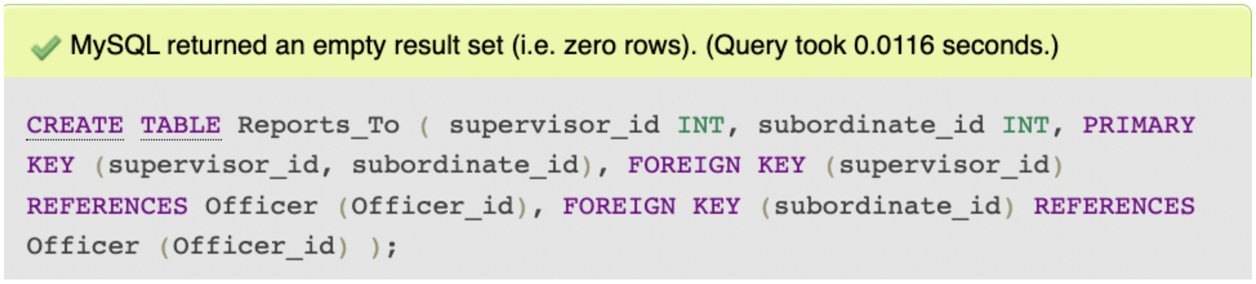
```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0099 seconds.)

CREATE TABLE Officer ( Officer_id INT PRIMARY KEY, Name VARCHAR(50), Email
VARCHAR(50), Department VARCHAR(50), Title VARCHAR(20) );
```

```
CREATE TABLE Reports_To (

          supervisor_id INT,

          subordinate_id INT,

          PRIMARY KEY (supervisor_id, subordinate_id),

          FOREIGN KEY (supervisor_id) REFERENCES Officer (Officer_id),

          FOREIGN KEY (subordinate_id) REFERENCES Officer (Officer_id)

);
```



```
CREATE TABLE Reports_To (
    supervisor_id INT,
    subordinate_id INT,
    PRIMARY KEY (supervisor_id, subordinate_id),
    FOREIGN KEY (supervisor_id) REFERENCES Officer (Officer_id),
    FOREIGN KEY (subordinate_id) REFERENCES Officer (Officer_id)
);
```

<u>Output-</u>


```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0116 seconds.)

CREATE TABLE Reports_To ( supervisor_id INT, subordinate_id INT, PRIMARY
KEY (supervisor_id, subordinate_id), FOREIGN KEY (supervisor_id)
REFERENCES Officer (Officer_id), FOREIGN KEY (subordinate_id) REFERENCES
Officer (Officer_id) );
```

CREATE TABLE Local_Employee (

Employee_id INT PRIMARY KEY,

Department VARCHAR(50),

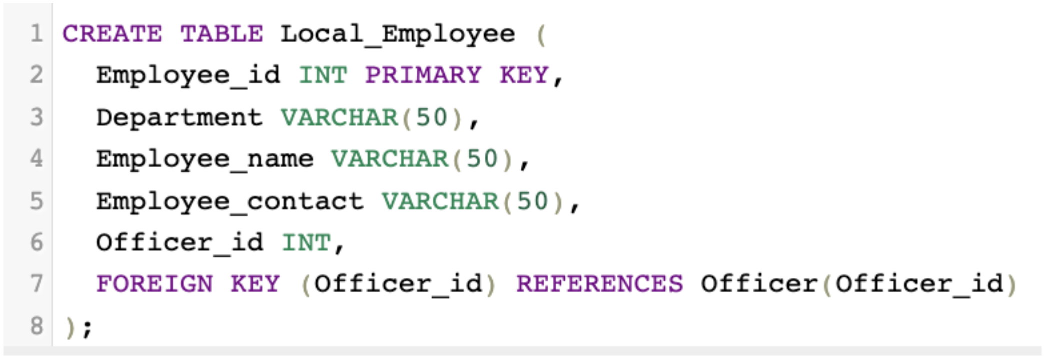
Employee_name VARCHAR(50),

Employee_contact VARCHAR(50),

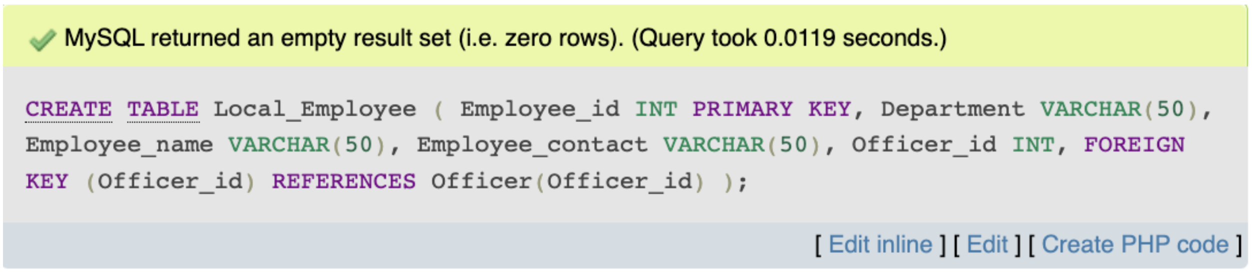
Officer_id INT,

FOREIGN KEY (Officer_id) REFERENCES Officer(Officer_id)

);


```
CREATE TABLE Local_Employee (
  Employee_id INT PRIMARY KEY,
  Department VARCHAR(50),
  Employee_name VARCHAR(50),
  Employee_contact VARCHAR(50),
  Officer_id INT,
  FOREIGN KEY (Officer_id) REFERENCES Officer(Officer_id)
);
```

<u>Output-</u>


```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0119 seconds.)

CREATE TABLE Local_Employee ( Employee_id INT PRIMARY KEY, Department VARCHAR(50),
Employee_name VARCHAR(50), Employee_contact VARCHAR(50), Officer_id INT, FOREIGN
KEY (Officer_id) REFERENCES Officer(Officer_id) );
```

5.2. Inserting Data into the tables

Input data into **Applicant table**

INSERT INTO Applicant (Passport_no, Name, DOB, Address, Nationality, Courier_code) VALUES

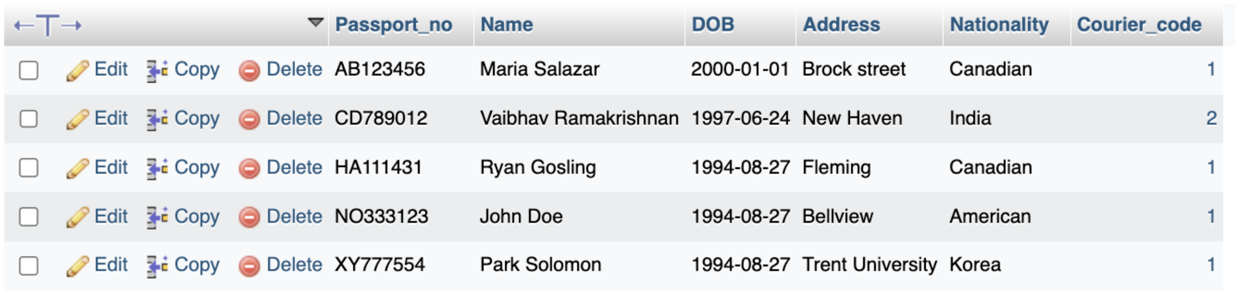
('AB123456', ‘Maria Salazar’, ‘2000-01-01’, 'Brock Street', 'Canadian', 1),

('CD789012', 'Vaibhav Ramakrishnan', '1997-06-24', 'New Haven', 'India', 2),

('XY777554', 'Park Solomon', '1994-08-27', 'Trent University', 'Korea', 1),

('NO333123', 'John Doe', '1994-08-27', 'Bellview', 'American', 1),

('HA111431', 'Ryan Gosling', '1994-08-27', 'Fleming', 'Canadian', 1);



	Passport_no	Name	DOB	Address	Nationality	Courier_code
Edit Copy Delete	AB123456	Maria Salazar	2000-01-01	Brock street	Canadian	1
Edit Copy Delete	CD789012	Vaibhav Ramakrishnan	1997-06-24	New Haven	India	2
Edit Copy Delete	HA111431	Ryan Gosling	1994-08-27	Fleming	Canadian	1
Edit Copy Delete	NO333123	John Doe	1994-08-27	Bellview	American	1
Edit Copy Delete	XY777554	Park Solomon	1994-08-27	Trent University	Korea	1

The above command helps us insert 5 rows into the Applicant table. The data includes the Passport number, name of the applicant, date of birth of the applicant, his/her address, nationality and the courier code used by the applicant.

Input data into **Courier table**

INSERT INTO Courier (Courier_code, Courier_name, Courier_contact, Courier_address) VALUES

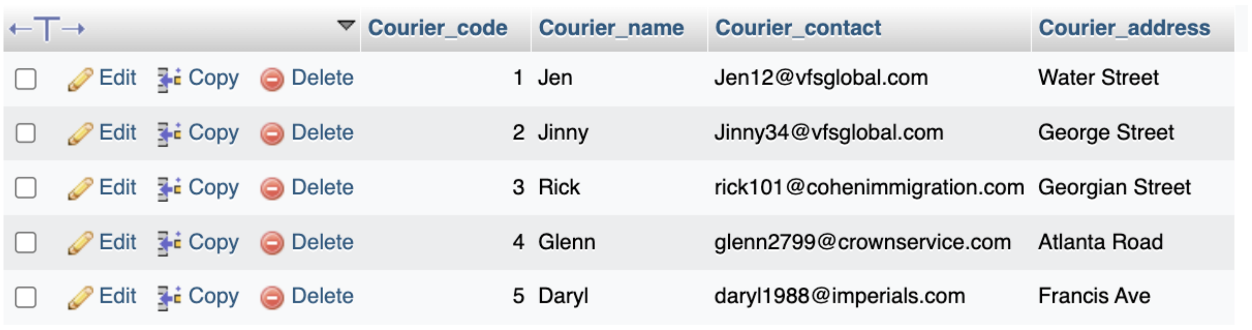
(1, 'Jen’, 'Jen12@vfsglobal.com', 'Water Street'),

(2, 'Jinny', 'Jinny34@vfsglobal.com’, 'George Street');

(3, 'Rick', 'rick101@cohenimmigration.com', 'Georgian Street');

(4, 'Glenn', 'glenn2799@crownservice.com', 'Atlanta Road');

(5, 'Daryl', 'daryl1988@imperials.com', 'Francis Ave');



	Courier_code	Courier_name	Courier_contact	Courier_address
Edit Copy Delete	1	Jen	Jen12@vfsglobal.com	Water Street
Edit Copy Delete	2	Jinny	Jinny34@vfsglobal.com	George Street
Edit Copy Delete	3	Rick	rick101@cohenimmigration.com	Georgian Street
Edit Copy Delete	4	Glenn	glenn2799@crownservice.com	Atlanta Road
Edit Copy Delete	5	Daryl	daryl1988@imperials.com	Francis Ave

The above command helps us insert 5 rows into the Courier table. The data includes the courier code, the name of the courier, the email contact of the courier and the address of each courier.

Input data into **Dependent table**

INSERT INTO Dependent (Dependent_passport_num, Name, Relationship, DOB, Passport_no) VALUES

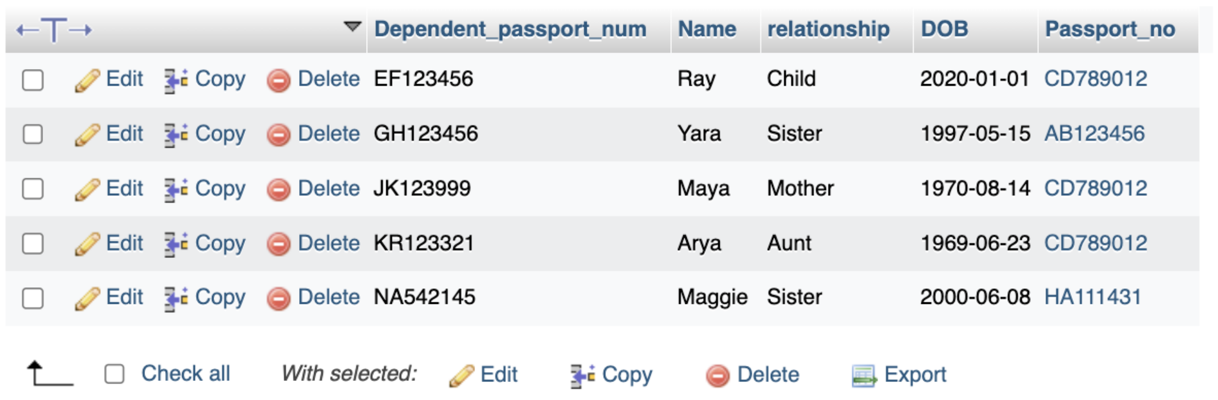
('EF123456', 'Ray', 'Child', '2020-01-01', 'CD789012'),

('JK123999', 'Maya', 'Mother', '1970-08-14', 'CD789012'),

('GH123456', 'Yara', 'Sister', '1997-05-15', 'AB123456'),

('KR123321', 'Arya', 'Aunt', '1969-06-23', 'CD789012'),

('NA542145', 'Maggie', 'Sister', '2000-06-08', 'HA111431');



	Dependent_passport_num	Name	relationship	DOB	Passport_no
Edit Copy Delete	EF123456	Ray	Child	2020-01-01	CD789012
Edit Copy Delete	GH123456	Yara	Sister	1997-05-15	AB123456
Edit Copy Delete	JK123999	Maya	Mother	1970-08-14	CD789012
Edit Copy Delete	KR123321	Arya	Aunt	1969-06-23	CD789012
Edit Copy Delete	NA542145	Maggie	Sister	2000-06-08	HA111431

Check all *With selected:* Edit Copy Delete Export

The above command helps us insert respective dependents' information which includes their passport number, the name of the dependent, the relationship they have with the primary applicant, their date of birth and passport number of the applicant (primary applicant).

Input data into **Visa_application table**

INSERT INTO Visa_application (Application_id, Location, Application_date, Passport_no, Officer_id) VALUES

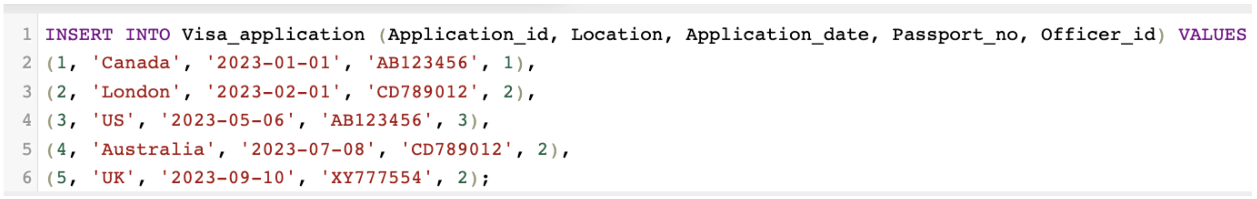
(1, 'Canada', '2023-01-01', 'AB123456', 1),

(2, 'London', '2023-02-01', 'CD789012', 2),

(3, 'US', '2023-05-06', 'AB123456', 3),

(4, 'Australia', '2023-07-08', 'CD789012', 2),

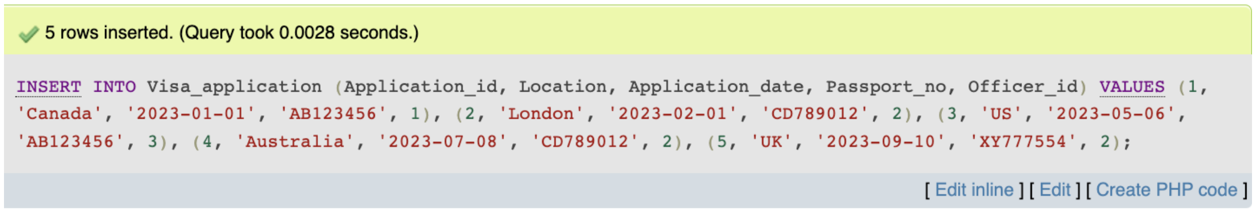
(5, 'UK', '2023-09-10', 'XY777554', 2);



```
INSERT INTO Visa_application (Application_id, Location, Application_date, Passport_no, Officer_id) VALUES
(1, 'Canada', '2023-01-01', 'AB123456', 1),
(2, 'London', '2023-02-01', 'CD789012', 2),
(3, 'US', '2023-05-06', 'AB123456', 3),
(4, 'Australia', '2023-07-08', 'CD789012', 2),
(5, 'UK', '2023-09-10', 'XY777554', 2);
```

The above command helps us insert information about the visa applicant for respective primary applicant, which includes their application id, location from where they are applying, their date of application, their passport number and the officer id their application belongs to.

<u>Output-</u>



```
✓ 5 rows inserted. (Query took 0.0028 seconds.)

INSERT INTO Visa_application (Application_id, Location, Application_date, Passport_no, Officer_id) VALUES (1,
'Canada', '2023-01-01', 'AB123456', 1), (2, 'London', '2023-02-01', 'CD789012', 2), (3, 'US', '2023-05-06',
'AB123456', 3), (4, 'Australia', '2023-07-08', 'CD789012', 2), (5, 'UK', '2023-09-10', 'XY777554', 2);

[ Edit inline ] [ Edit ] [ Create PHP code ]
```

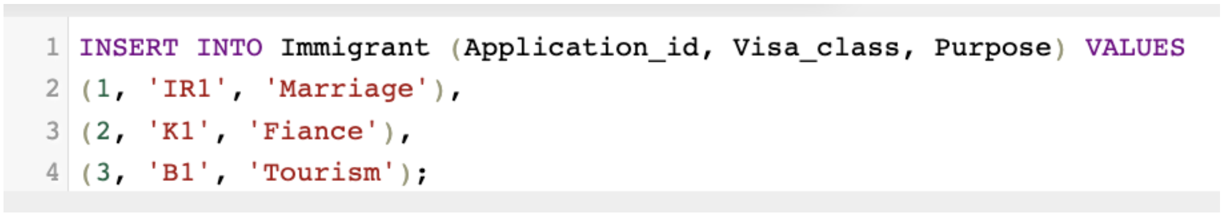
<u>Input data into **Immigrant table**</u>

INSERT INTO Immigrant (Application_id, Visa_class, Purpose) VALUES

(1, 'IR1', 'Marriage'),

(2, 'K1', 'Fiance'),

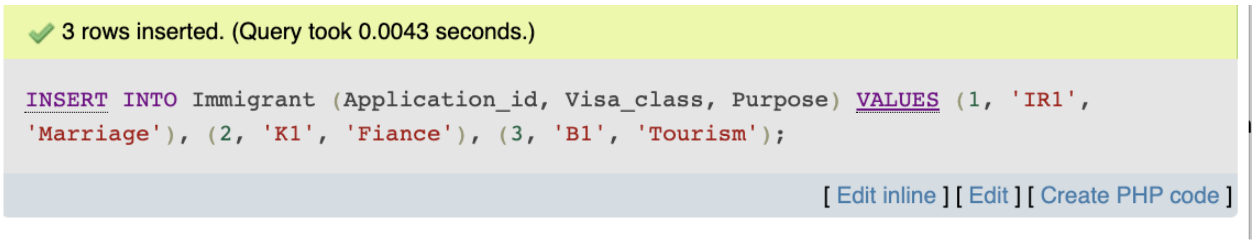
(3, 'B1', 'Tourism');



```
INSERT INTO Immigrant (Application_id, Visa_class, Purpose) VALUES
(1, 'IR1', 'Marriage'),
(2, 'K1', 'Fiance'),
(3, 'B1', 'Tourism');
```

The above table contains information about immigrants which includes various visa classes tagged to respective application ids. The application classes belong to their respective purpose (IR1- Marriage, K1-Fiancé, B1- Tourism).

<u>Output-</u>



```
✓ 3 rows inserted. (Query took 0.0043 seconds.)

INSERT INTO Immigrant (Application_id, Visa_class, Purpose) VALUES (1, 'IR1',
'Marriage'), (2, 'K1', 'Fiance'), (3, 'B1', 'Tourism');
```

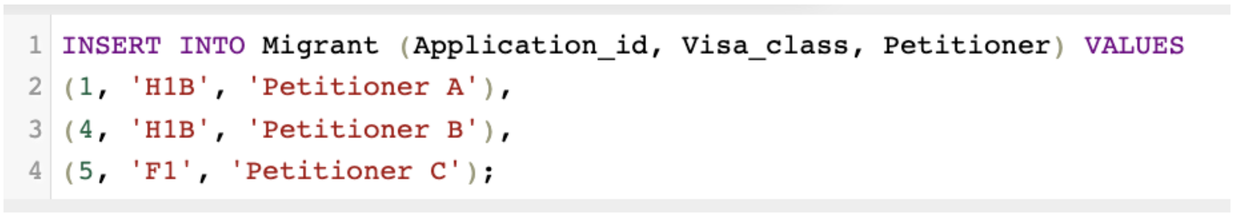
Input data into **Migrant table**

INSERT INTO Migrant (Application_id, Visa_class, Petitioner) VALUES

(1, 'H1B', 'Petitioner A'),

(4, 'H1B', 'Petitioner B'),

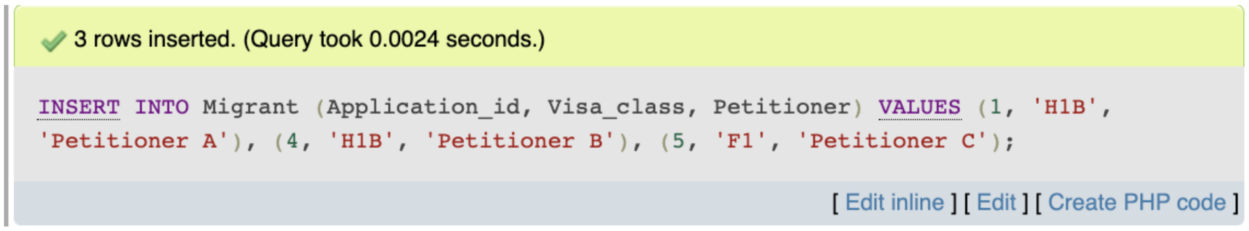
(5, 'F1', 'Petitioner C');



```
INSERT INTO Migrant (Application_id, Visa_class, Petitioner) VALUES
(1, 'H1B', 'Petitioner A'),
(4, 'H1B', 'Petitioner B'),
(5, 'F1', 'Petitioner C');
```

The above table contains information about migrants which includes various visa classes tagged to respective application ids. The applications are also tagged to their respective Petitioners. For simplicity, the Petitioners here are named A, B and C.

Output-



```
✓ 3 rows inserted. (Query took 0.0024 seconds.)

INSERT INTO Migrant (Application_id, Visa_class, Petitioner) VALUES (1, 'H1B',
'Petitioner A'), (4, 'H1B', 'Petitioner B'), (5, 'F1', 'Petitioner C');
```

Input data into **Visa table**

INSERT INTO Visa (Visa_no, Approved_date, Expired_date, Visa_class, Employee_id, Officer_id, Courier_code) VALUES

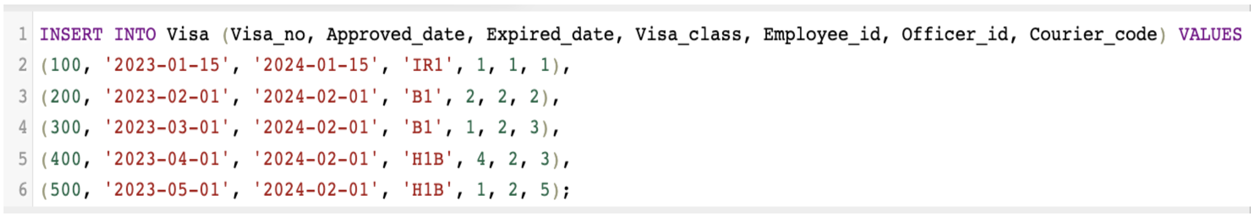
(100, '2023-01-15', '2024-01-15', 'IR1', 1, 1, 1),

(200, '2023-02-01', '2024-02-01', 'B1', 2, 2, 2),

(300, '2023-03-01', '2024-02-01', 'B1', 1, 2, 3),

(400, '2023-04-01', '2024-02-01', 'H1B', 4, 2, 3),

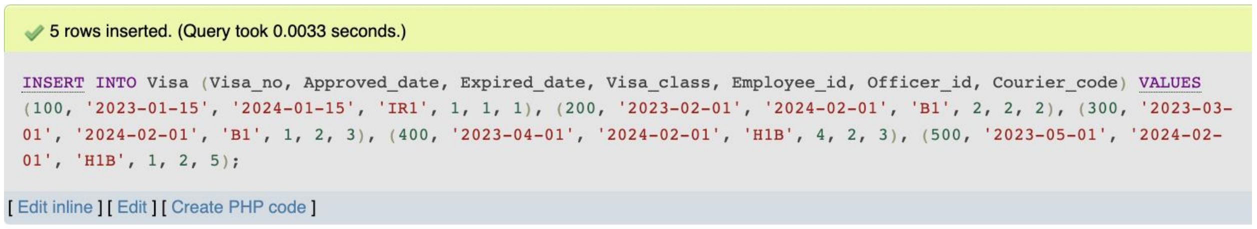
(500, '2023-05-01', '2024-02-01', 'H1B', 1, 2, 5);



```
INSERT INTO Visa (Visa_no, Approved_date, Expired_date, Visa_class, Employee_id, Officer_id, Courier_code) VALUES
(100, '2023-01-15', '2024-01-15', 'IR1', 1, 1, 1),
(200, '2023-02-01', '2024-02-01', 'B1', 2, 2, 2),
(300, '2023-03-01', '2024-02-01', 'B1', 1, 2, 3),
(400, '2023-04-01', '2024-02-01', 'H1B', 4, 2, 3),
(500, '2023-05-01', '2024-02-01', 'H1B', 1, 2, 5);
```

The above command holds information about Visa number, the date the application is approved and the date the application expires. The Visa class and the employee id, officer id and courier id tagged to each of these files respectively.

<u>Output-</u>



```
✓ 5 rows inserted. (Query took 0.0033 seconds.)

INSERT INTO Visa (Visa_no, Approved_date, Expired_date, Visa_class, Employee_id, Officer_id, Courier_code) VALUES
(100, '2023-01-15', '2024-01-15', 'IR1', 1, 1, 1), (200, '2023-02-01', '2024-02-01', 'B1', 2, 2, 2), (300, '2023-03-
01', '2024-02-01', 'B1', 1, 2, 3), (400, '2023-04-01', '2024-02-01', 'H1B', 4, 2, 3), (500, '2023-05-01', '2024-02-
01', 'H1B', 1, 2, 5);

[ Edit inline ] [ Edit ] [ Create PHP code ]
```

<u>Input data into **Officer table**</u>

INSERT INTO Officer (Officer_id, Name, Email, Department, Title) VALUES

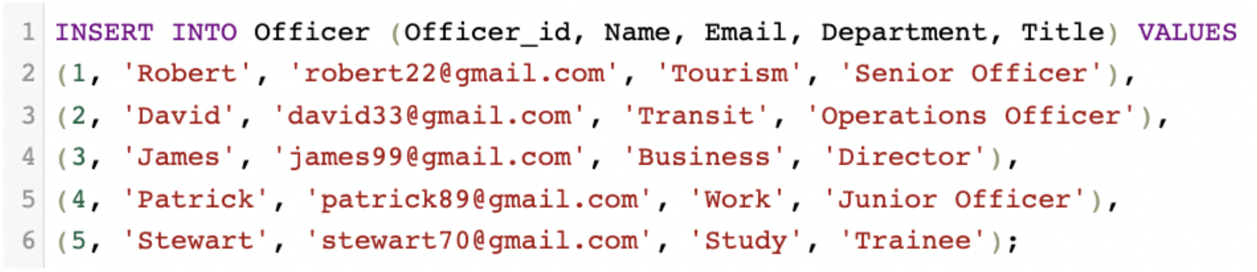
(1, 'Robert', 'robert22@gmail.com', 'Tourism', 'Senior Officer'),

(2, 'David', 'david33@gmail.com', 'Transit', 'Operations Officer'),

(3, 'James', 'james99@gmail.com', 'Business', 'Director'),

(4, 'Patrick', 'patrick89@gmail.com', 'Work', 'Junior Officer'),

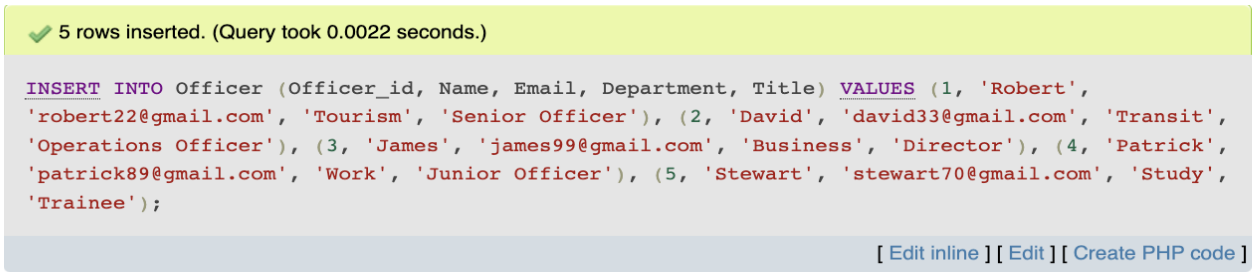
(5, 'Stewart', 'stewart70@gmail.com', 'Study', 'Trainee');



```
INSERT INTO Officer (Officer_id, Name, Email, Department, Title) VALUES
(1, 'Robert', 'robert22@gmail.com', 'Tourism', 'Senior Officer'),
(2, 'David', 'david33@gmail.com', 'Transit', 'Operations Officer'),
(3, 'James', 'james99@gmail.com', 'Business', 'Director'),
(4, 'Patrick', 'patrick89@gmail.com', 'Work', 'Junior Officer'),
(5, 'Stewart', 'stewart70@gmail.com', 'Study', 'Trainee');
```

The above command helps us insert information about 5 officers Robert, David, James, Patrick and Stewart. The 5 officers have their respective officer ids, Email addresses, Departments and titles.

<u>Output-</u>



✓ 5 rows inserted. (Query took 0.0022 seconds.)

```
INSERT INTO Officer (Officer_id, Name, Email, Department, Title) VALUES (1, 'Robert',
'robert22@gmail.com', 'Tourism', 'Senior Officer'), (2, 'David', 'david33@gmail.com', 'Transit',
'Operations Officer'), (3, 'James', 'james99@gmail.com', 'Business', 'Director'), (4, 'Patrick',
'patrick89@gmail.com', 'Work', 'Junior Officer'), (5, 'Stewart', 'stewart70@gmail.com', 'Study',
'Trainee');
```

[Edit inline] [Edit] [Create PHP code]

Input data into **Reports_to table**

INSERT INTO Reports_To (supervisor_id, subordinate_id)

VALUES

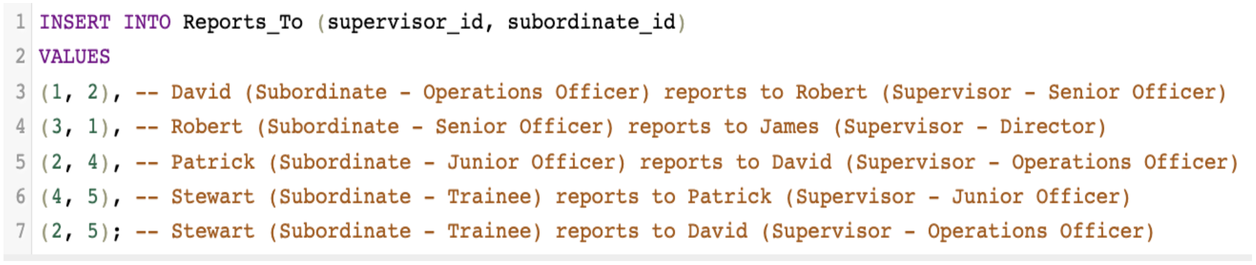
(1, 2),-- David (Subordinate- Operations Officer) reports to Robert (Supervisor- Senior Officer)

(3, 1),-- Robert (Subordinate- Senior Officer) reports to James (Supervisor- Director)

(2, 4),-- Patrick (Subordinate- Junior Officer) reports to David (Supervisor- Operations Officer)

(4, 5),-- Stewart (Subordinate- Trainee) reports to Patrick (Supervisor- Junior Officer)

(2, 5);-- Stewart (Subordinate- Trainee) reports to David (Supervisor- Operations Officer)



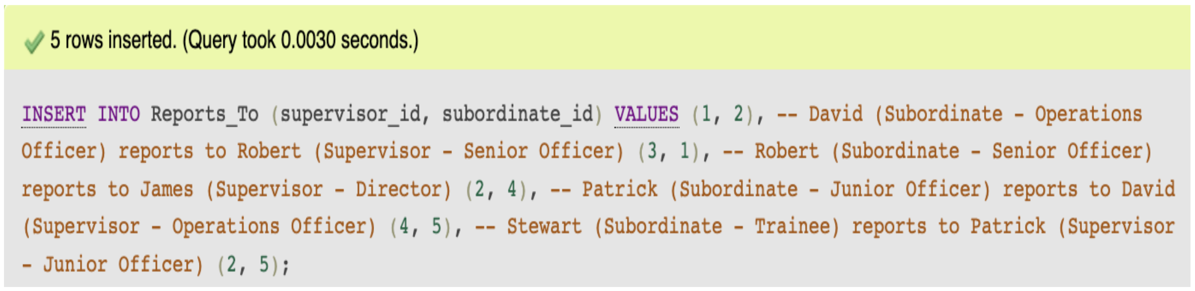
```
INSERT INTO Reports_To (supervisor_id, subordinate_id)
VALUES
(1, 2), -- David (Subordinate - Operations Officer) reports to Robert (Supervisor - Senior Officer)
(3, 1), -- Robert (Subordinate - Senior Officer) reports to James (Supervisor - Director)
(2, 4), -- Patrick (Subordinate - Junior Officer) reports to David (Supervisor - Operations Officer)
(4, 5), -- Stewart (Subordinate - Trainee) reports to Patrick (Supervisor - Junior Officer)
(2, 5); -- Stewart (Subordinate - Trainee) reports to David (Supervisor - Operations Officer)
```

The above command helps us create a hierarchy for the officers where each officer reports to one officer. As per the command above, below is the hierarchy -

Stewart -> Patrick -> David -> Robert -> James

Here, Stewart (Trainee) reports to both Patrick (Junior Officer) and David (Operations Officer).

Output



✓ 5 rows inserted. (Query took 0.0030 seconds.)

```
INSERT INTO Reports_To (supervisor_id, subordinate_id) VALUES (1, 2), -- David (Subordinate - Operations
Officer) reports to Robert (Supervisor - Senior Officer) (3, 1), -- Robert (Subordinate - Senior Officer)
reports to James (Supervisor - Director) (2, 4), -- Patrick (Subordinate - Junior Officer) reports to David
(Supervisor - Operations Officer) (4, 5), -- Stewart (Subordinate - Trainee) reports to Patrick (Supervisor
- Junior Officer) (2, 5);
```

Input data into **Local_Employee table**

INSERT INTO Local_Employee (Employee_id, Department, Employee_name, Employee_contact, Officer_id) VALUES

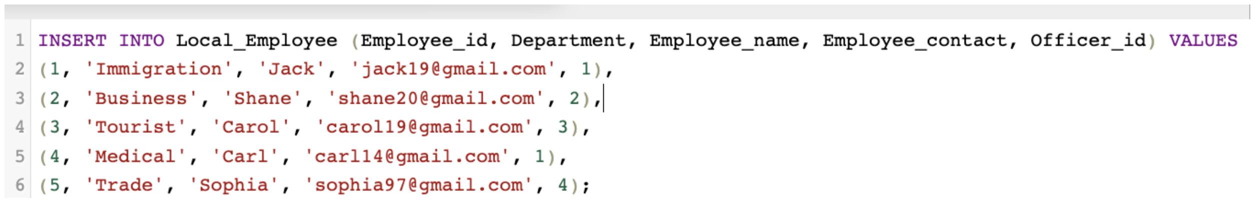
(1, 'Immigration', 'Jack', 'jack19@gmail.com', 1),

(2, 'Business', 'Shane', 'shane20@gmail.com', 2),

(3, 'Tourist', 'Carol', 'carol19@gmail.com', 3),

(4, 'Medical', 'Carl', 'carl14@gmail.com', 1),

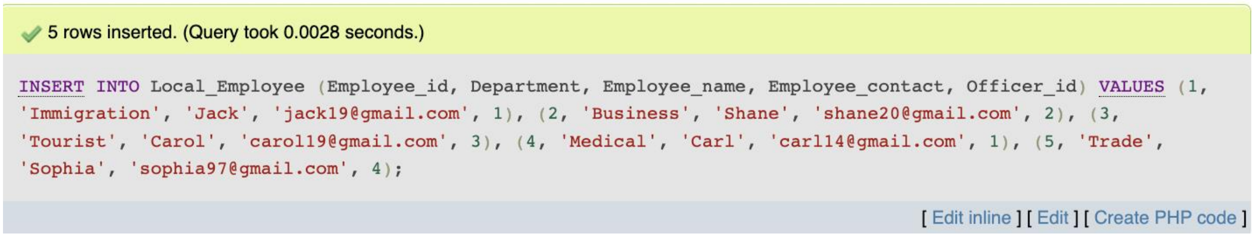
(5, 'Trade', 'Sophia', 'sophia97@gmail.com', 4);



```
INSERT INTO Local_Employee (Employee_id, Department, Employee_name, Employee_contact, Officer_id) VALUES
(1, 'Immigration', 'Jack', 'jack19@gmail.com', 1),
(2, 'Business', 'Shane', 'shane20@gmail.com', 2),
(3, 'Tourist', 'Carol', 'carol19@gmail.com', 3),
(4, 'Medical', 'Carl', 'carl14@gmail.com', 1),
(5, 'Trade', 'Sophia', 'sophia97@gmail.com', 4);
```

The above command helps us hold information about local employees who have their respective employee ids, departments they belong to, their email contact and officer ids that they work with.

<u>Output-</u>



```
✓ 5 rows inserted. (Query took 0.0028 seconds.)

INSERT INTO Local_Employee (Employee_id, Department, Employee_name, Employee_contact, Officer_id) VALUES (1,
'Immigration', 'Jack', 'jack19@gmail.com', 1), (2, 'Business', 'Shane', 'shane20@gmail.com', 2), (3,
'Tourist', 'Carol', 'carol19@gmail.com', 3), (4, 'Medical', 'Carl', 'carl14@gmail.com', 1), (5, 'Trade',
'Sophia', 'sophia97@gmail.com', 4);
```

VI. <u>Write the queries using SQL statements and the output of the SQL queries using screen shot</u>

(a) Three queries for one table

We will create 3 queries for the Applicant Table-

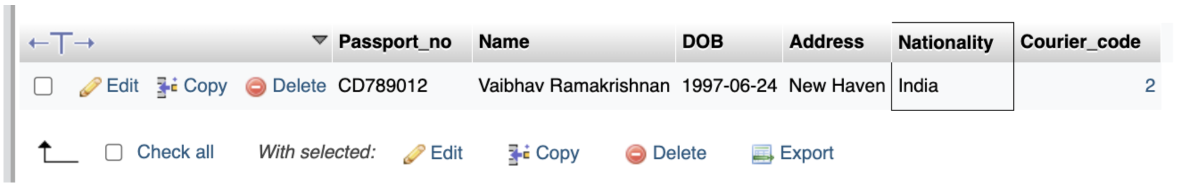
- Retrieve all applicants of a specific Nationality.

SELECT * FROM Applicant WHERE Nationality = 'India';

```
SELECT * FROM Applicant WHERE Nationality = 'India';
```

<u>Output-</u>

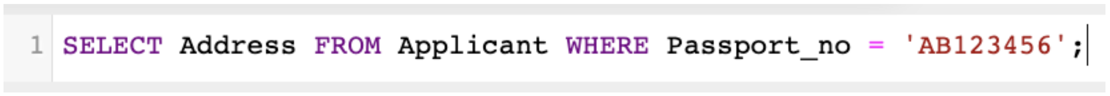
The provided query retrieves all applicants from the Applicant table who have a specific nationality, in this case, 'India'.



Passport_no	Name	DOB	Address	Nationality	Courier_code
CD789012	Vaibhav Ramakrishnan	1997-06-24	New Haven	India	2

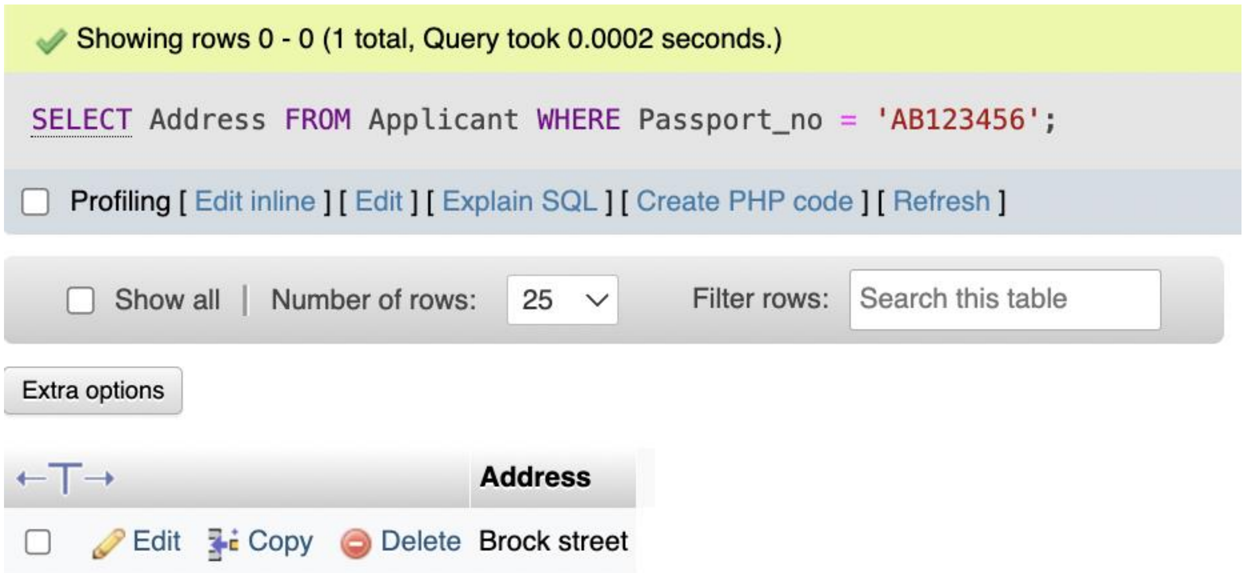
- Retrieve the address of an applicant with a given Passport_no.

SELECT Address FROM Applicant WHERE Passport_no = 'AB123456';



```
SELECT Address FROM Applicant WHERE Passport_no = 'AB123456';
```

Output-



```
Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)

SELECT Address FROM Applicant WHERE Passport_no = 'AB123456';
```

Address
Brock street

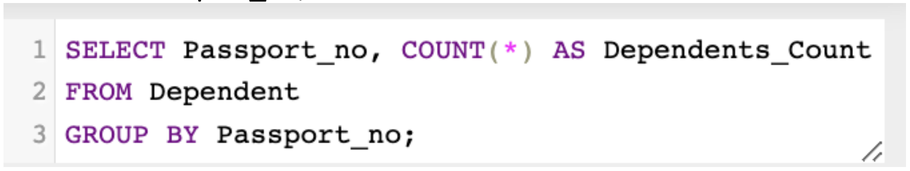
The provided query retrieves the address of an applicant with a specific passport number ('AB123456') from the Applicant table.

- Retrieve the count of dependents for each applicant.

SELECT Passport_no, COUNT(*) AS Dependents_Count

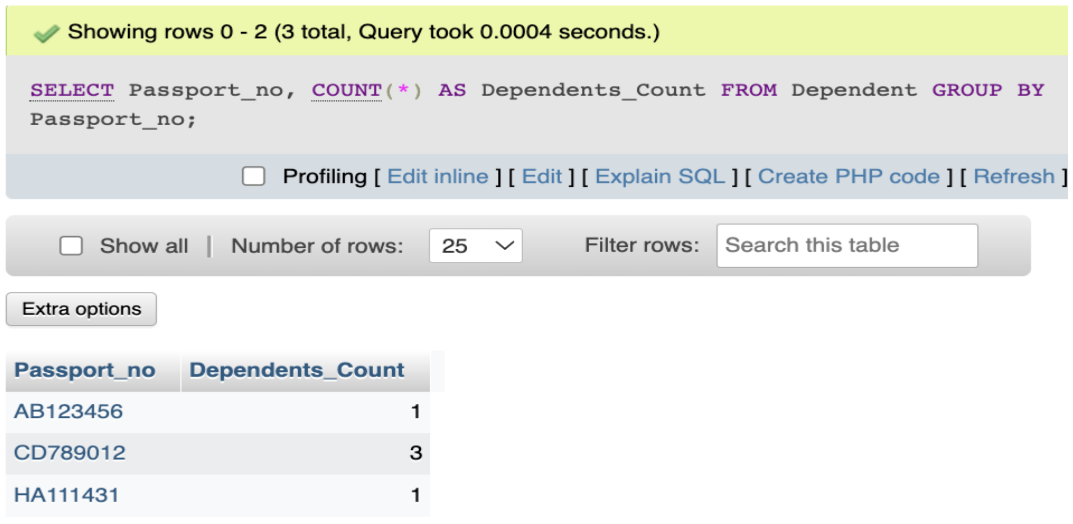
FROM Dependent

GROUP BY Passport_no;



```
SELECT Passport_no, COUNT(*) AS Dependents_Count
FROM Dependent
GROUP BY Passport_no;
```

Output-



Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)

```
SELECT Passport_no, COUNT(*) AS Dependents_Count FROM Dependent GROUP BY
Passport_no;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25
Filter rows: Search this table
Extra options

Passport_no	Dependents_Count
AB123456	1
CD789012	3
HA111431	1

This query counts the number of dependents for each unique Passport_no in the Dependent table and returns the Dependents_count for each.

(b) Three queries for two tables.

We will create 3 queries for the **Applicant** and **Visa_application** tables-

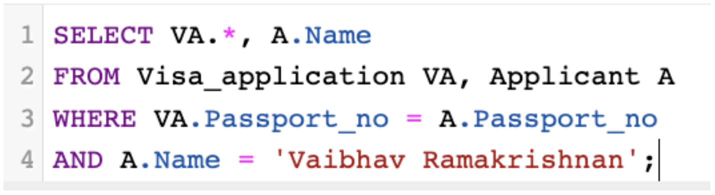
- Retrieve all visa applications submitted by an applicant.

  SELECT VA.*, A.Name

  FROM Visa_application VA, Applicant A

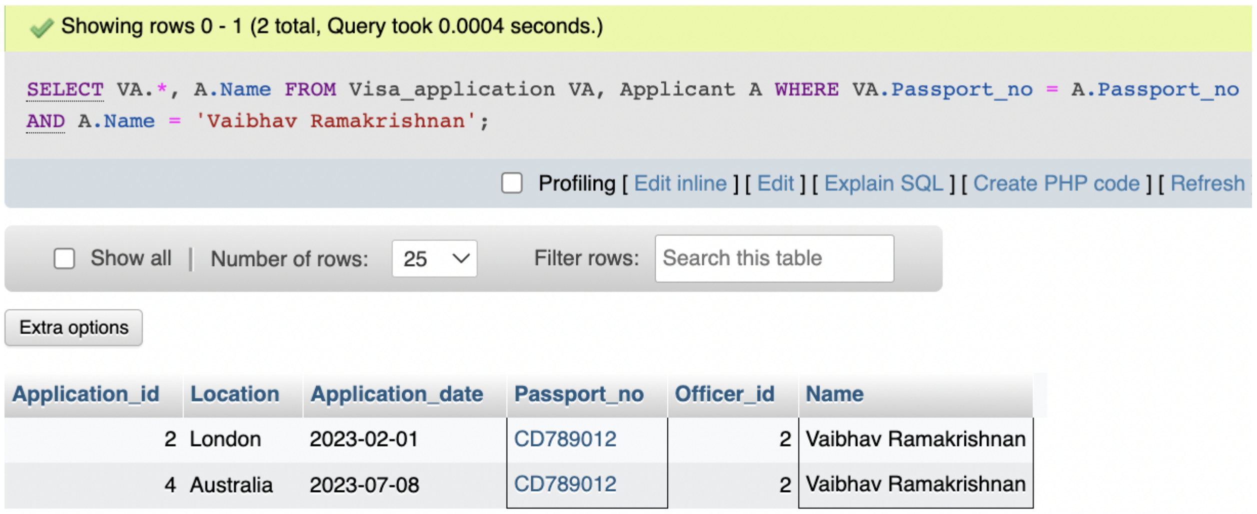
  WHERE VA.Passport_no = A.Passport_no

  AND A.Name = 'Vaibhav Ramakrishnan';



```
SELECT VA.*, A.Name
FROM Visa_application VA, Applicant A
WHERE VA.Passport_no = A.Passport_no
AND A.Name = 'Vaibhav Ramakrishnan';
```

Output-


Showing rows 0 - 1 (2 total, Query took 0.0004 seconds.)

```
SELECT VA.*, A.Name FROM Visa_application VA, Applicant A WHERE VA.Passport_no = A.Passport_no
AND A.Name = 'Vaibhav Ramakrishnan';
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

Application_id	Location	Application_date	Passport_no	Officer_id	Name
2	London	2023-02-01	CD789012	2	Vaibhav Ramakrishnan
4	Australia	2023-07-08	CD789012	2	Vaibhav Ramakrishnan

The given query retrieves all the visa applications that have been submitted by a specific applicant named 'Vaibhav Ramakrishnan'. The query involves two tables: Visa_application and Applicant. It performs a join operation by matching the Passport_no column between the Visa_application and Applicant tables. This ensures that the visa applications are linked to the corresponding applicant.

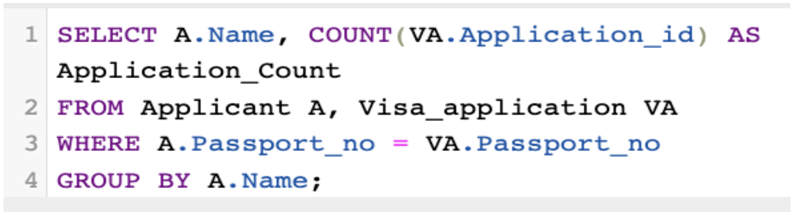
- Retrieve the count of visa applications for visa applicants.

SELECT A.Name, COUNT(VA.Application_id) AS Application_Count

FROM Applicant A, Visa_application VA

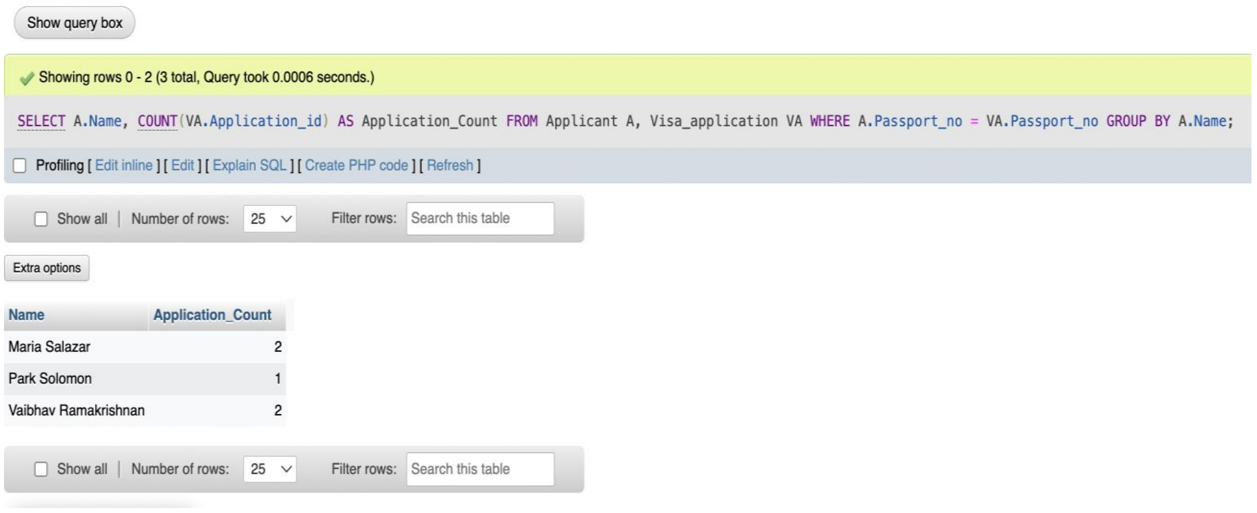
WHERE A.Passport_no = VA.Passport_no

GROUP BY A.Name;



```
SELECT A.Name, COUNT(VA.Application_id) AS
Application_Count
FROM Applicant A, Visa_application VA
WHERE A.Passport_no = VA.Passport_no
GROUP BY A.Name;
```

<u>Output-</u>


Show query box
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
Extra options

Showing rows 0 - 2 (3 total, Query took 0.0006 seconds.)

```sql
SELECT A.Name, COUNT(VA.Application_id) AS Application_Count FROM Applicant A, Visa_application VA WHERE A.Passport_no = VA.Passport_no GROUP BY A.Name;
```

Name	Application_Count
Maria Salazar	2
Park Solomon	1
Vaibhav Ramakrishnan	2

The provided query retrieves the count of visa applications for each applicant. It involves two tables: Applicant and Visa_application. The query performs a join operation by matching the Passport_no column between the two tables.

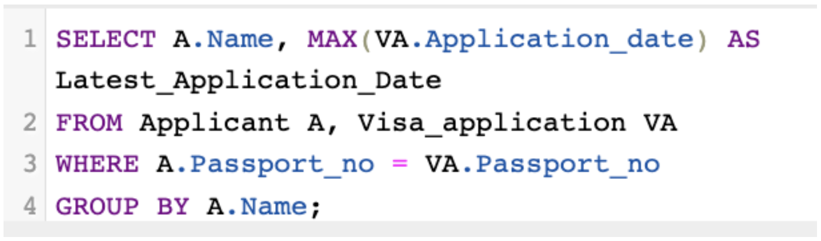
- Retrieve the most recent application date for visa applicants.

SELECT A.Name, MAX(VA.Application_date) AS Latest_Application_Date

FROM Applicant A, Visa_application VA

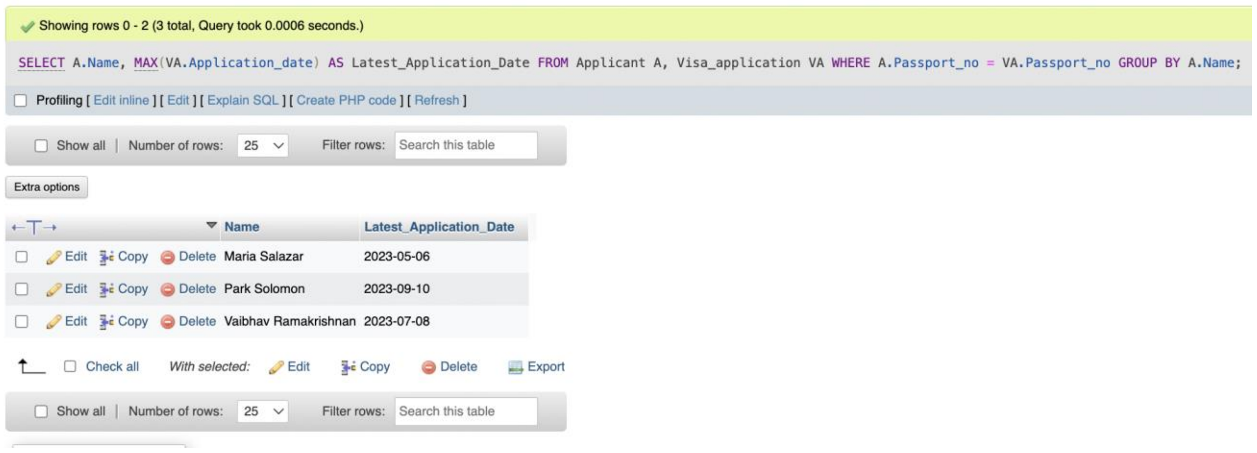
WHERE A.Passport_no = VA.Passport_no

GROUP BY A.Name;



```sql
SELECT A.Name, MAX(VA.Application_date) AS
Latest_Application_Date
FROM Applicant A, Visa_application VA
WHERE A.Passport_no = VA.Passport_no
GROUP BY A.Name;
```

<u>Output-</u>


```
✓ Showing rows 0 - 2 (3 total, Query took 0.0006 seconds.)

SELECT A.Name, MAX(VA.Application_date) AS Latest_Application_Date FROM Applicant A, Visa_application VA WHERE A.Passport_no = VA.Passport_no GROUP BY A.Name;
```

Name	Latest_Application_Date
Maria Salazar	2023-05-06
Park Solomon	2023-09-10
Vaibhav Ramakrishnan	2023-07-08

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
Extra options
Edit Copy Delete
Check all With selected: Edit Copy Delete Export

The provided query retrieves the latest application date for each applicant from the Applicant and Visa_application tables. It performs a join operation based on the Passport_no column, connecting the two tables.

(c) Three queries for three tables.

We will create 3 queries for the **Visa, Officer**, and **Local_Employee** tables-

- Retrieve all approved visas for a specific applicant.

  SELECT V.*

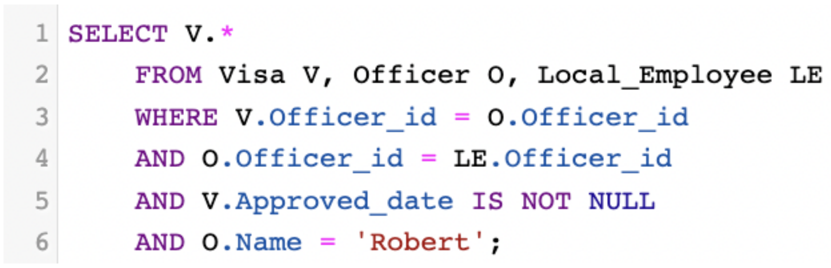
  FROM Visa V, Officer O, Local_Employee LE

  WHERE V.Officer_id = O.Officer_id

  AND O.Officer_id = LE.Officer_id

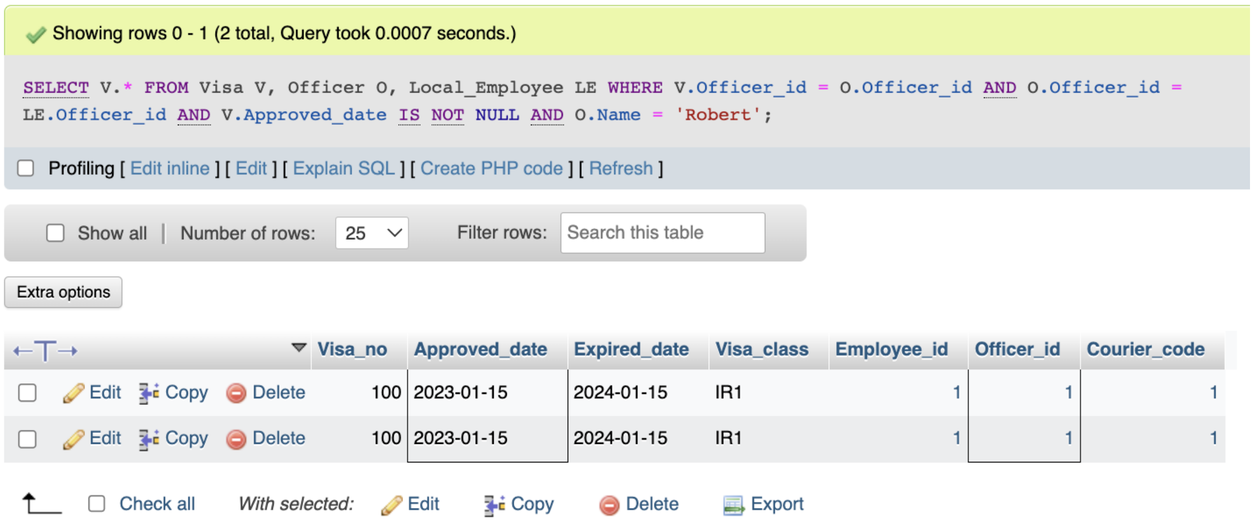
  AND V.Approved_date IS NOT NULL

  AND O.Name = 'Robert';


```
SELECT V.*
    FROM Visa V, Officer O, Local_Employee LE
    WHERE V.Officer_id = O.Officer_id
    AND O.Officer_id = LE.Officer_id
    AND V.Approved_date IS NOT NULL
    AND O.Name = 'Robert';
```

The provided query retrieves all the approved visas for a specific applicant, in this case, the applicant associated with the officer named 'Robert'. The query involves three tables: Visa, Officer, and Local_Employee. It performs a join operation by matching the Officer_id between the Visa and Officer tables, and also between the Officer and Local_Employee tables. This ensures that the corresponding records are linked correctly.

<u>Output-</u>


Showing rows 0 - 1 (2 total, Query took 0.0007 seconds.)

```
SELECT V.* FROM Visa V, Officer O, Local_Employee LE WHERE V.Officer_id = O.Officer_id AND O.Officer_id =
LE.Officer_id AND V.Approved_date IS NOT NULL AND O.Name = 'Robert';
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
Extra options

	Visa_no	Approved_date	Expired_date	Visa_class	Employee_id	Officer_id	Courier_code
Edit Copy Delete	100	2023-01-15	2024-01-15	IR1	1	1	1
Edit Copy Delete	100	2023-01-15	2024-01-15	IR1	1	1	1

- Retrieve the number of approved visas and show each visa class.

SELECT V.Visa_class, COUNT(V.Visa_no) AS Approved_Visas

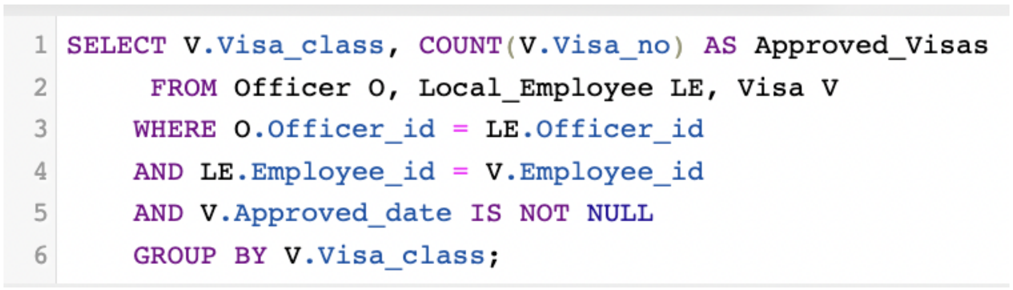
FROM Officer O, Local_Employee LE, Visa V

WHERE O.Officer_id = LE.Officer_id

AND LE.Employee_id = V.Employee_id

AND V.Approved_date IS NOT NULL

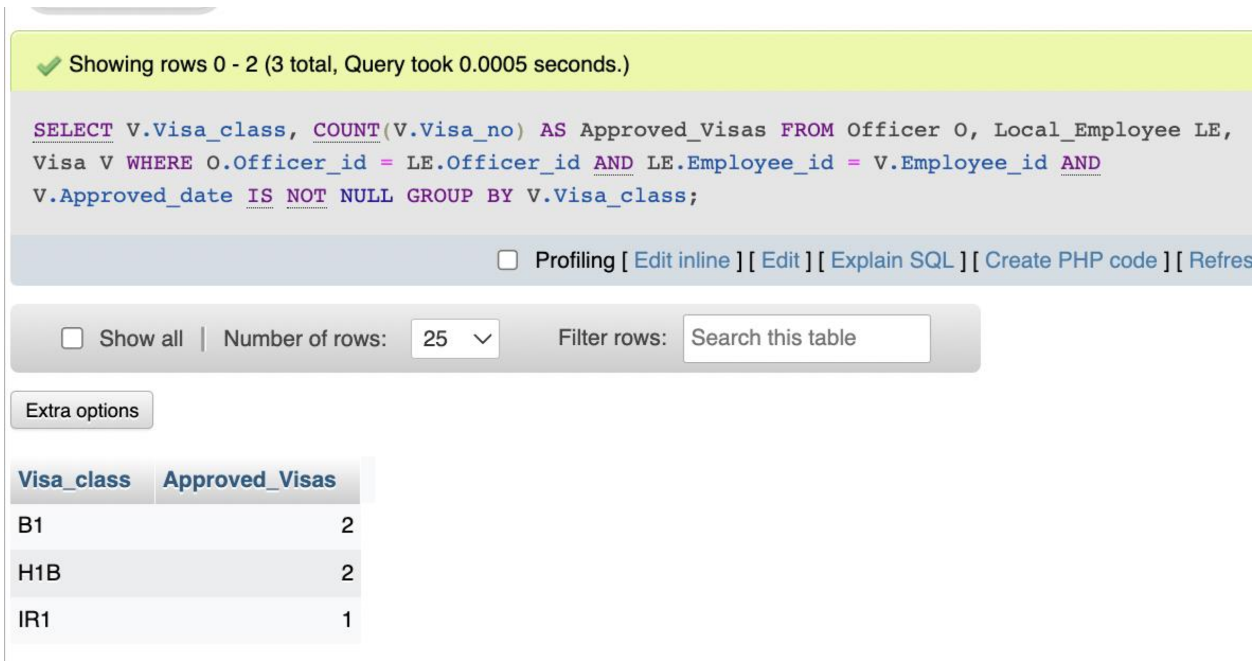
GROUP BY V.Visa_class;



```
SELECT V.Visa_class, COUNT(V.Visa_no) AS Approved_Visas
     FROM Officer O, Local_Employee LE, Visa V
    WHERE O.Officer_id = LE.Officer_id
    AND LE.Employee_id = V.Employee_id
    AND V.Approved_date IS NOT NULL
    GROUP BY V.Visa_class;
```

The provided query retrieves the number of approved visas and shows each visa class. It involves three tables: Officer, Local_Employee, and Visa. The query joins these tables using the Officer_id and Employee_id columns, ensuring that the corresponding records are linked. It further filters the data by checking that the Approved_date in the Visa table is not NULL, indicating that the visa has been approved.

<u>Output-</u>


Showing rows 0 - 2 (3 total, Query took 0.0005 seconds.)

```
SELECT V.Visa_class, COUNT(V.Visa_no) AS Approved_Visas FROM Officer O, Local_Employee LE,
Visa V WHERE O.Officer_id = LE.Officer_id AND LE.Employee_id = V.Employee_id AND
V.Approved_date IS NOT NULL GROUP BY V.Visa_class;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refres
Show all | Number of rows: 25
Filter rows: Search this table
Extra options

Visa_class	Approved_Visas
B1	2
H1B	2
IR1	1

- Retrieve the Visa_no and approved_date for each approved visa along with the corresponding officer's name using the Visa, Officer, and Local_Employee tables.

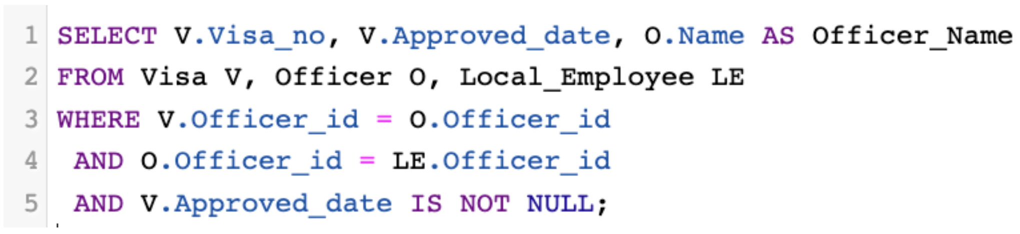
SELECT V.Visa_no, V.Approved_date, O.Name AS Officer_Name

FROM Visa V, Officer O, Local_Employee LE

WHERE V.Officer_id = O.Officer_id

 AND O.Officer_id = LE.Officer_id

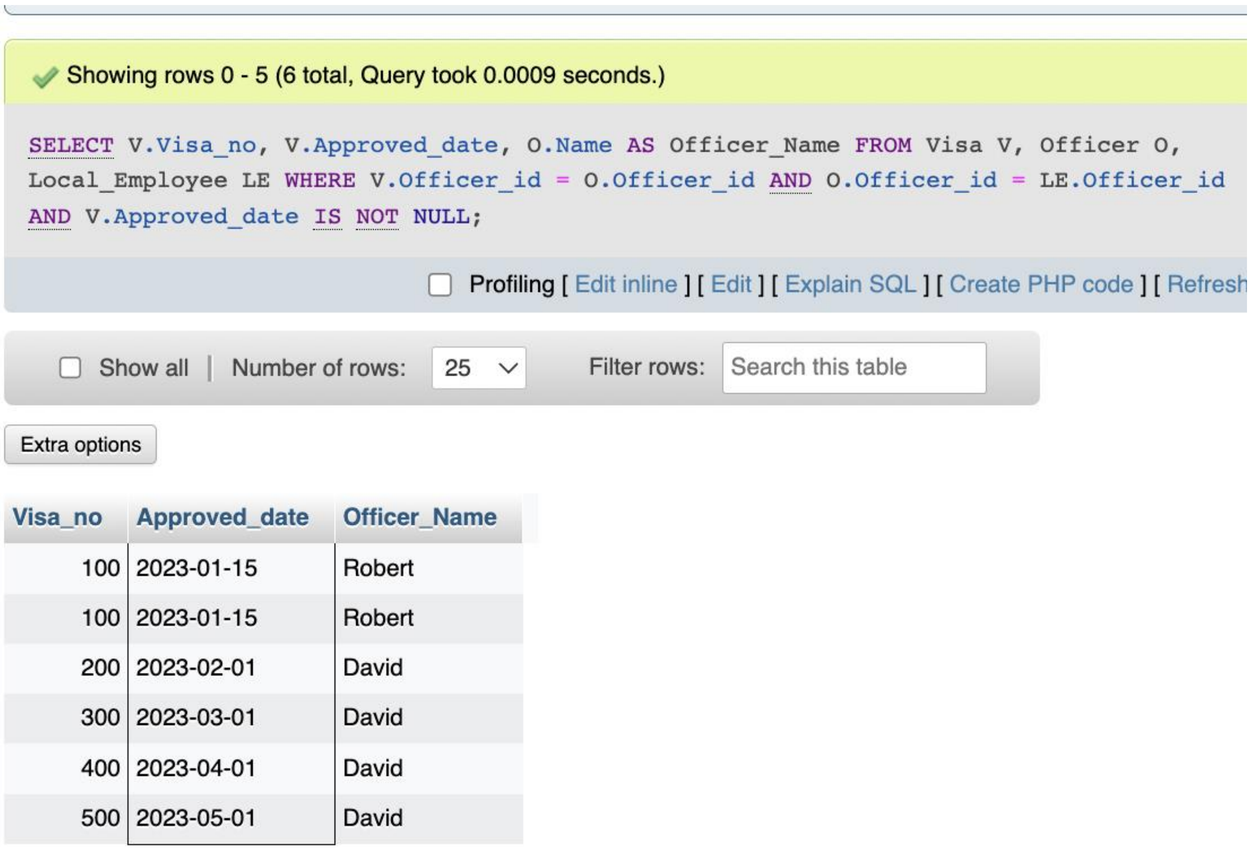
 AND V.Approved_date IS NOT NULL;



```
SELECT V.Visa_no, V.Approved_date, O.Name AS Officer_Name
FROM Visa V, Officer O, Local_Employee LE
WHERE V.Officer_id = O.Officer_id
 AND O.Officer_id = LE.Officer_id
 AND V.Approved_date IS NOT NULL;
```

This query retrieves the Visa number, Approved date, and the corresponding officer's name for each approved visa by joining the Visa, Officer, and Local_Employee tables based on their respective foreign key relationships.

<u>Output-</u>



Showing rows 0 - 5 (6 total, Query took 0.0009 seconds.)

```
SELECT V.Visa_no, V.Approved_date, O.Name AS Officer_Name FROM Visa V, Officer O,
Local_Employee LE WHERE V.Officer_id = O.Officer_id AND O.Officer_id = LE.Officer_id
AND V.Approved_date IS NOT NULL;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh
Show all | Number of rows: 25
Filter rows: Search this table
Extra options

Visa_no	Approved_date	Officer_Name
100	2023-01-15	Robert
100	2023-01-15	Robert
200	2023-02-01	David
300	2023-03-01	David
400	2023-04-01	David
500	2023-05-01	David

VII. Summary

Overall, our project aims to make a significant contribution to the U.S. Consulate General's operations by providing a sophisticated and reliable Visa Application Database. By streamlining the data management process through UML and the Relational Database Model, I tried to enhance the Consulate's operational efficiency and provide better support for decision-making and reporting tasks. Furthermore, a well-organized database system will reduce the processing times, leading to improved customer service for visa applicants. I believe that with our proposed solution, the Consulate will have a powerful tool which enables seamless management and effectively process visa application information.