

Group 7

Visa Processing Management Database

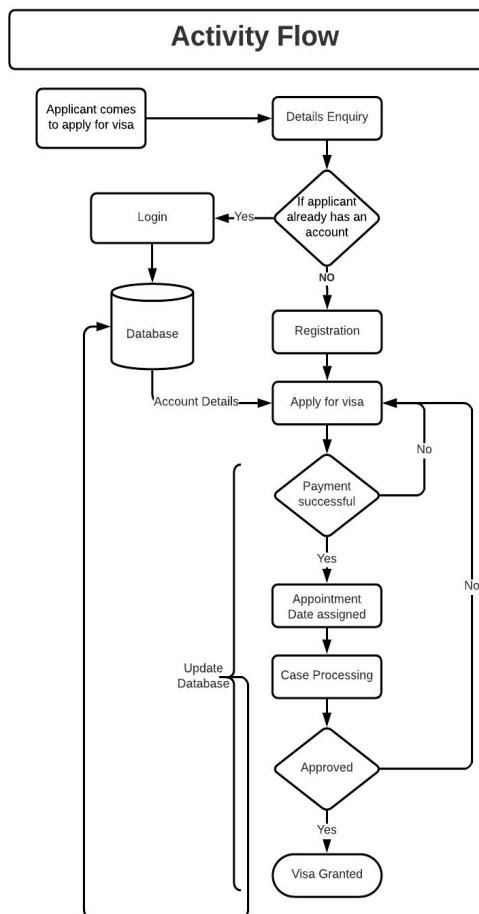
Mentor TA : Deepak Tarkunde

ID	Name
201801123	Dhyey Gami
201801199	Krutik Parmar
201801203	Ayushi Patel
201801457	Shivangi Zala

Section1: Final version of SRS

Product Perspectives:

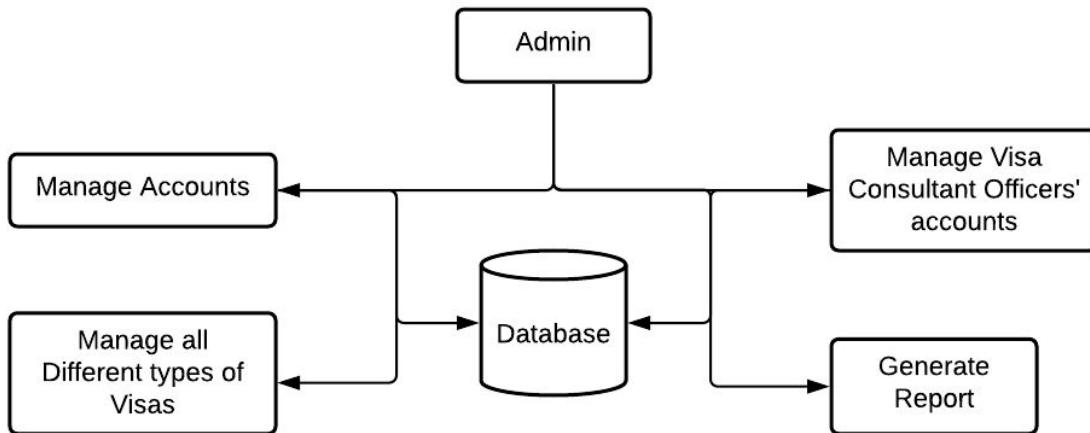
The project “Visa Processing Management Database” is a system that is designed to securely manage the visa application. This application is designed to be used by people who are applying for visas, visa application centers, etc. This application covers every step in the visa application process like fee payment to booking an appointment. Users can create an account and using that account they can apply for an online visa, keep track of their visa application process. The user details are stored in the database so he doesn't need to enter the same details again. Users can apply for different types of visas and based on those types users have to submit different information. From the server-side, the Visa Consultant Officer (visa approver) accesses the data of users and then assigns the appointment date to the user. Users can see the status of his/her application. Visa Consultant Officer also has an account using which he/she can keep track of all users whom he/she will be handling. In addition to this, there is a Admin who supervises this process and can access all users and approver's accounts.



Product Functions:

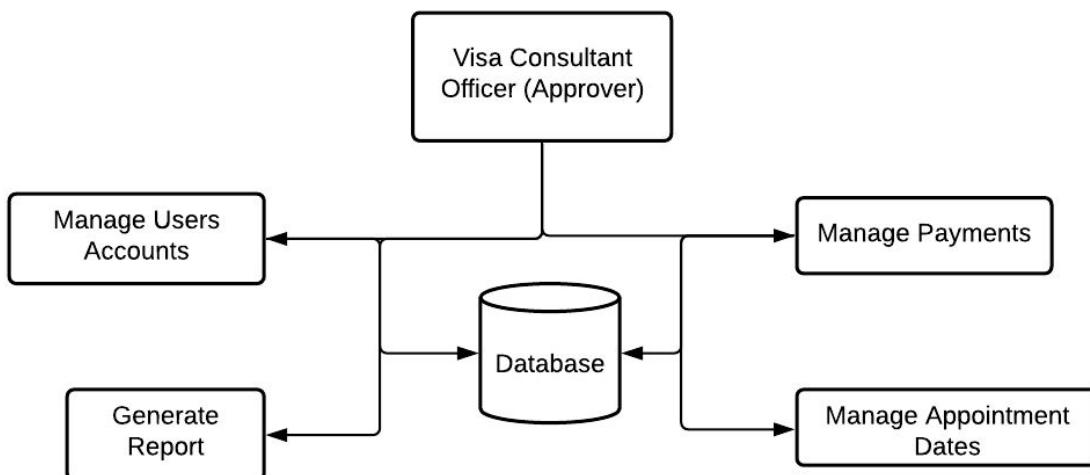
Admin Subsystem :

Admin can access all user accounts. Admin can manage all payments which are made and assign appointment dates to them. Admin can see what types of visas are applied more often. Admin can manage all appointment dates. Admin can also access all Visa Consultant Officers' accounts. Admin can manage roles of Visa Consultant Officer. Admin can also generate reports like for which visa users are applying more, based on the data which is there in the database.



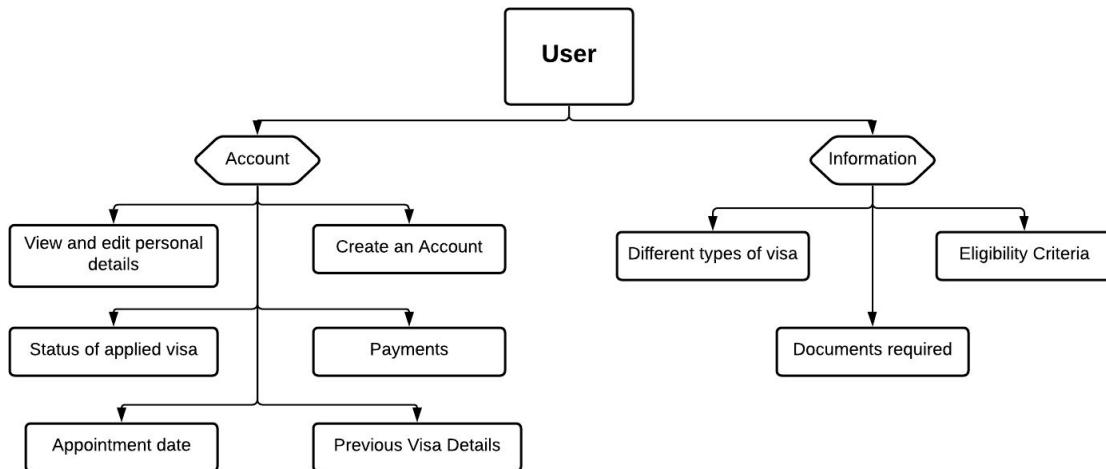
Visa Consultant Officer Subsystem:

Visa consultant officer can view visa applications, can check logs about applications like approval status, payment status, etc. Visa consultant officers can generate logs about the status of applications and can approve or reject applications for visas.

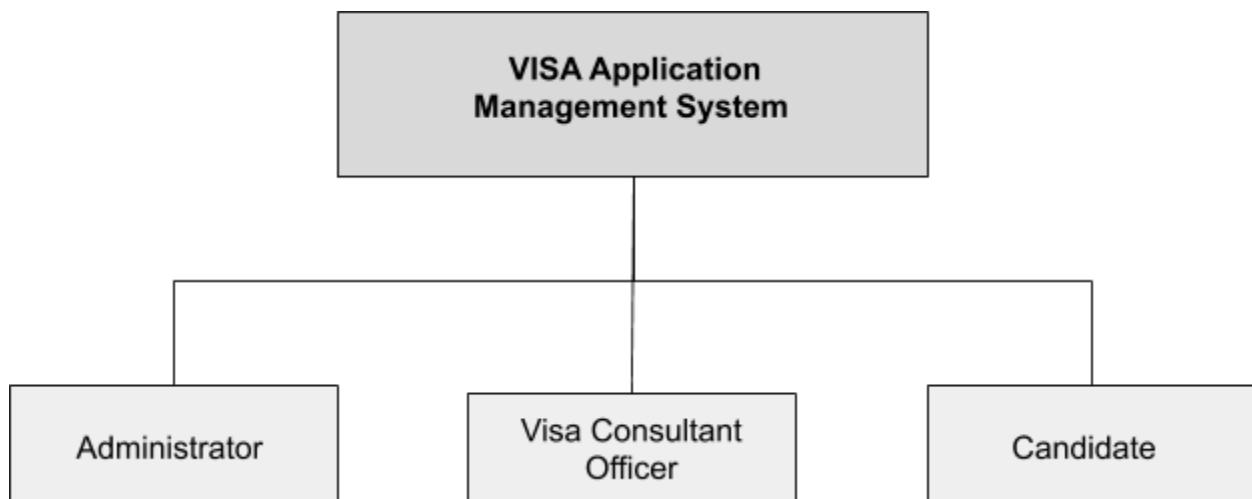


User Subsystem :

User can create a new account or can view and edit all account details. User can also see the status of the applied visa, previous visa details, payments he made and appointment date. User can also get information about different types of visa, documents required and eligibility criteria, etc.



User classes and characteristics:



There are 3 core classes in the system:

1. Administrator (Government)
2. Visa Consultant Officer (Approver)
3. Candidate (Applicant)

1. Administrator:

- This is the most privileged user class
- Can access all application data
- Maintain details about logs in the database
- Can remove any application or access to data to approvers
- Can assign the role of Visa Consultant officer
- Maintain Visa consultant officers
- Can check and generate logs
- Can accept or reject Visa applications

2. Visa Consultant Officer :

- This user class is authorized by the admin for the role of approval
- Can access Visa application for viewing
- Can accept or reject Visa applications
- Can generate logs about application status
- Can schedule appointment dates
- Can manage payments

3. Candidate:

- Can apply for multiple Visas.
- Can check their application status
- Can get information about the procedure
- Can make updates in the current application when required
- Can view application details
- Can see process fees

Operating Environment:

- Software:**

Database:

PostgreSQL (v10.14)

Programming Language:

PL/pgSQL

Supported Operating System:

Windows 8.0 or above

macOS

Linux

- Hardware requirements:**

1 GHz processor

2 GB RAM minimum (4GB recommended)

512 MB HardDisk (For software)

10GB HardDisk (For storing data of applicants and logs)

Privileges:

There are 4 key privileges:

- View
- Add
- Delete
- Update

Administrator	
View	Application Details Logs about applications

	Status of any application
Add	Generate logs Give authority to approvers
Delete	Remove applications Delete logs
Update	Update status of applications Update roles

Visa Consultant Officer	
View	Application Details Logs about applications
Add	Generate logs about the status Schedule appointments
Delete	-
Update	Update status of applications Manage payments

Candidates	
View	Application Details Status of any application View appointments
Add	Add applications for visas
Delete	Delete application Delete his/her account
Update	Update current application

Assumptions:

- Basic knowledge of computers because the process is online
- Familiarity with English language and terminology of the Visa system
- Admin is familiar with the management of system and database
- System server should be online 24 × 7 hrs
- System server is having good internet connectivity always
- We are assuming that the user will add appropriate input because we didn't add input validations

Business Constraints:

Initial development and gathering of requirements are time-consuming.
COVID pandemic situation is a major issue for VISA approvals so have to apply strict visa regulations for preventing spread.

Background Readings:

[India Visa Online](#)

[Writing a software requirements specification document | by Eni Sinanaj](#)

[Visa Project /India](#)

- Got an insight about a current facility for visa on arrival
- Get to know about Validities for visas and application fees

Section2: Final Noun Analysis

Accepted Noun and Verb

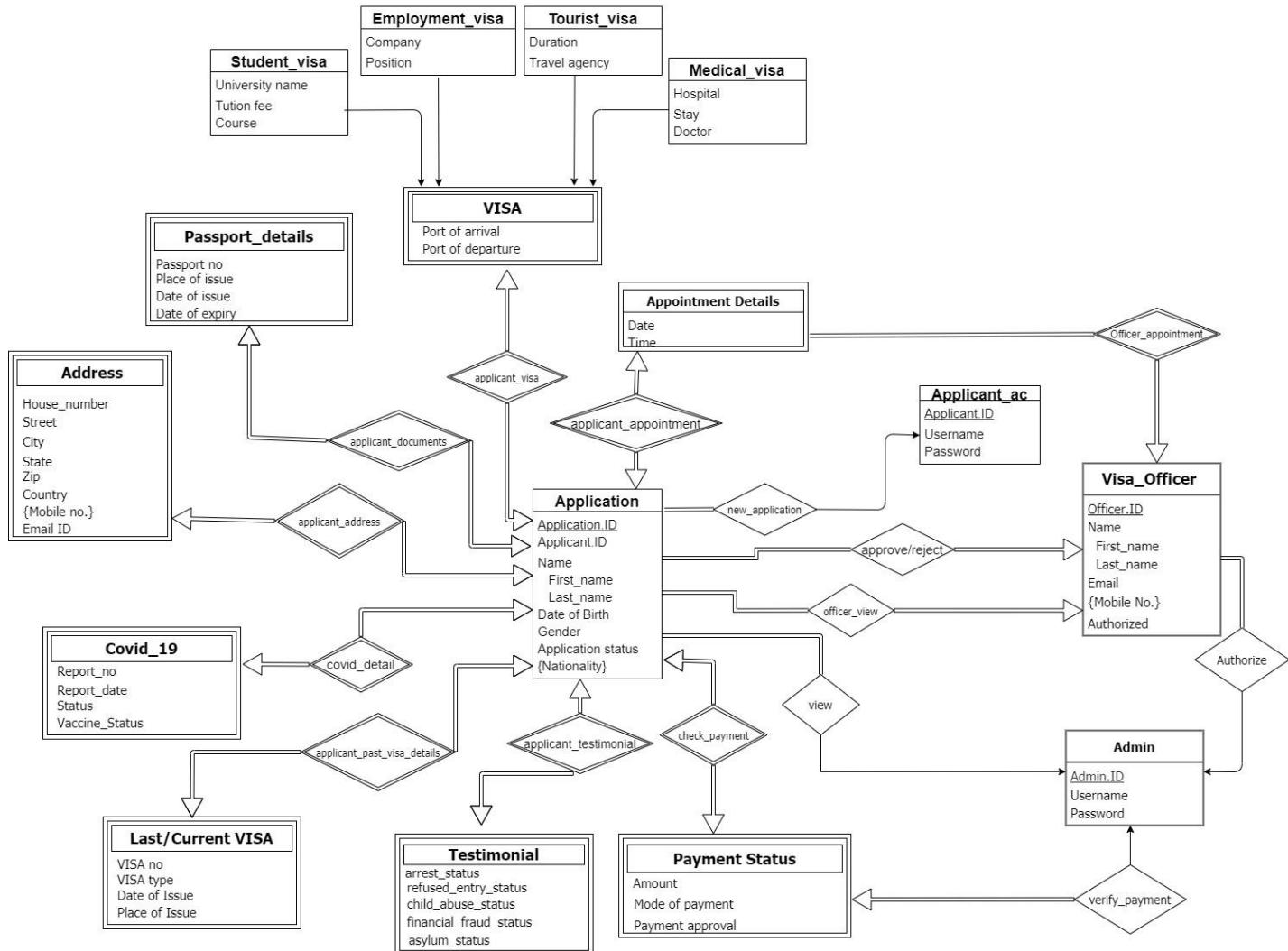
Entity	Attribute	Relations (Verb)
Applicant	Applicant ID	Create
Application	Application ID	Approve
Visa Officer	Username	Reject
Admin	Password	View
Address	First name	Verify
VISA	Last name	Authorize
Appointment Details	Date of birth	
Passport Details	Gender	
Student visa	Application status	
Employment visa	Nationality	
Tourist visa	Email	
Medical visa	Mobile no.	
Covid 19	Nationality	
Last/Current visa	Visa no.	
Testimonial	Visa type	
Payment status	Date of issue	
	Place of issue	
	Amount	
	Mode of payment	
	Report no.	
	Report date	
	Report status	
	Vaccine status	

Rejected Noun & Verbs list

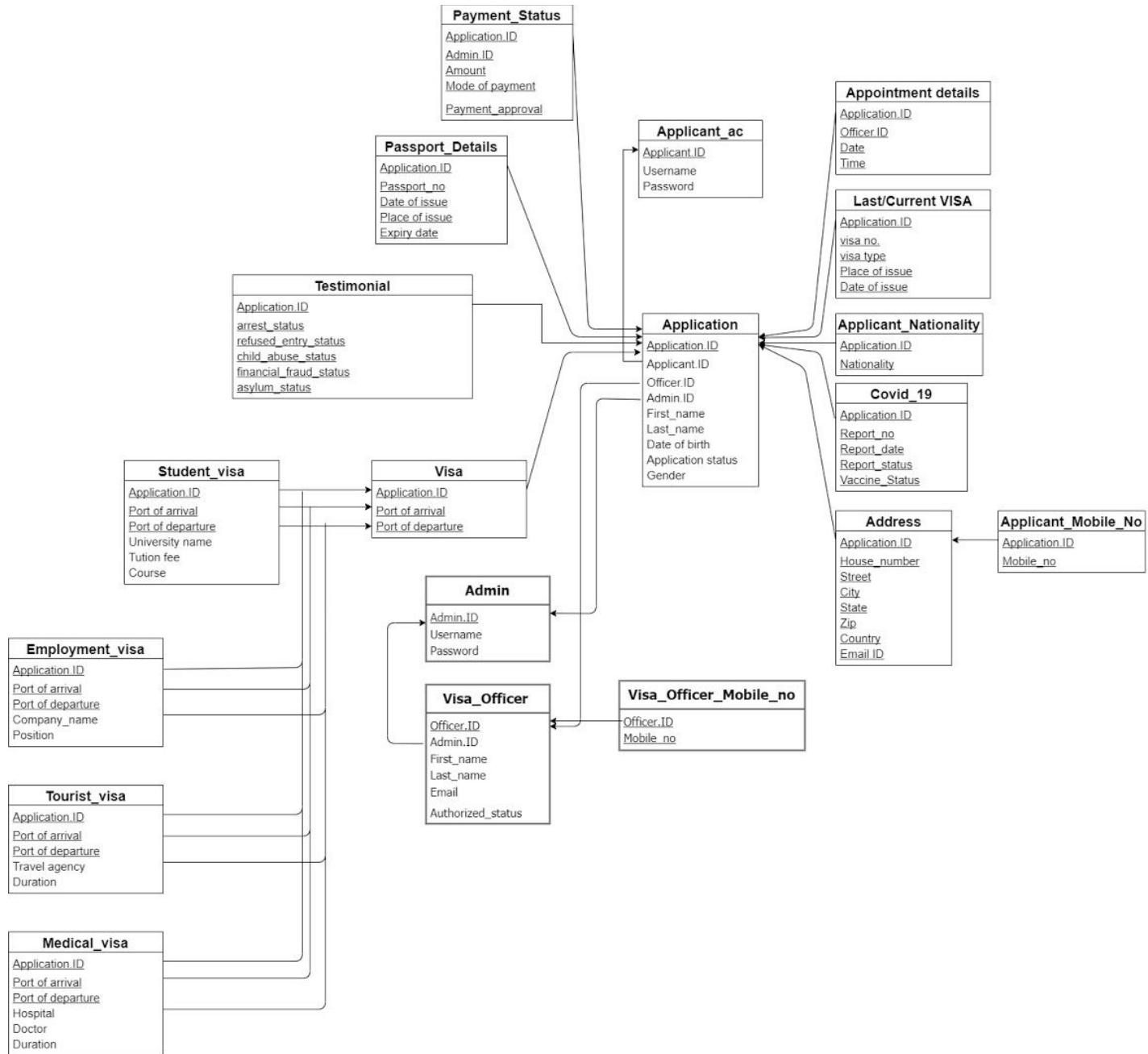
Noun	Verb
VISA	Redundant
Passport	Attribute
Document	Attribute
VISA Application	Redundant

Applicant	Redundant
Status	Attribute
Courier	Irrelevant
Case	General
Stamp	Irrelevant
Authorization Letter	General
Reason	Redundant
VISA application center	Redundant
VISA application receipt	Redundant
Authorized person	Unclear
Photocopy	Irrelevant
Email	Attribute
India	Irrelevant
Apply	Association
Collect	Irrelevant
Accept	Association
Reject	Association
Grant	Redundant
Authorize	Unclear
Receive	Redundant
Attach	Irrelevant
Scan	Irrelevant
See	Association

Section 3 : Final ER-Diagram



Section4: Final Relational Model



Section5: Normalization and Schema Refinement

Update, Insert, Delete anomalies for every schema

1. applicant_ac(Applicant_ID,Username,Password)
 - Insertion Anomaly:not present
 - Deletion Anomaly:not present
 - Updation Anomaly:not present
2. application(Application_ID,Applicant_ID,Officer_ID,Admin_ID,First_name,Last_name,Date_of_birth,Application_status,Gender)
 - Insertion Anomaly:not present
 - Deletion Anomaly:not present
 - Updation Anomaly:not present
3. admin(Admin_ID, Username, Password)
 - Insertion Anomaly:not present
 - Deletion Anomaly:not present
 - Updation Anomaly:not present
4. visa_officer(Officer_ID, Admin_ID, First_name, Last_name, Email, Authorized_status)
 - Insertion Anomaly:not present
 - Deletion Anomaly:not present
 - Updation Anomaly:not present
5. visa_officer_mobile_no(Officer_ID, Mobile_no)
 - Insertion Anomaly:not present
 - Deletion Anomaly:not present
 - Updation Anomaly:not present
6. address(Application_ID, house_number, Street, City, State, zipcode, Country, Email_ID)
 - Insertion Anomaly:not present
 - Deletion Anomaly:not present
 - Updation Anomaly:not present
7. applicant_mobile_no(Application_ID, Mobile_no)
 - Insertion Anomaly:not present

- Deletion Anomaly:not present
 - Updation Anomaly:not present
8. covid19(Application_ID, Report_no, Report_date, report_tatus, Vaccine_status)
- Insertion Anomaly:not present
 - Deletion Anomaly:not present
 - Updation Anomaly:not present
9. applicant_nationality(Application_ID, Nationality)
- Insertion Anomaly:not present
 - Deletion Anomaly:not present
 - Updation Anomaly:not present
10. Last/Current_visa(Application_ID, visa_no,visa type, Place of issue, Date of issue)
- Insertion Anomaly:not present
 - Deletion Anomaly:not present
 - Updation Anomaly:not present
11. appointment_details(Application_ID, Officer_ID, Date, Time)
- Insertion Anomaly:not present
 - Deletion Anomaly:not present
 - Updation Anomaly:not present
12. payment_status(Application_ID, Admin_ID, Amount, Mode_of_Payment,Payment_approval)
- Insertion Anomaly:not present
 - Deletion Anomaly:not present
 - Updation Anomaly:not present
13. passport_details(Application_ID, Passport_No, Date_of_issue, Place_of_issue, Expiry_date, Nationality)
- Insertion Anomaly:not present
 - Deletion Anomaly:not present
 - Updation Anomaly:not present
14. testimonial(Application_ID,arrested, refused_entry_status, human_drug_trafficing_involvement , child_abuse , financial_fraud,asylum_status)
- Insertion Anomaly:not present
 - Deletion Anomaly:not present
 - Updation Anomaly:not present

15. visa(Application_ID, Port_of_arrival, Port_of_departure)

- Insertion Anomaly:not present
- Deletion Anomaly:not present
- Updation Anomaly:not present

16. student_visa(Application_ID, Port_of_arrival, Port_of_departure, Validity, University_name, Tution_fee, Course)

- Insertion Anomaly:not present
- Deletion Anomaly:not present
- Updation Anomaly:not present

17. employment_visa(Application_ID, Port_of_arrival, Port_of_departure, Company name, Position, Validity)

- Insertion Anomaly:not present
- Deletion Anomaly:not present
- Updation Anomaly:not present

18. tourist_visa(Application_ID, Port_of_arrival, Port_of_departure, Travel agency, Duration, Validity)

- Insertion Anomaly:not present
- Deletion Anomaly:not present
- Updation Anomaly:not present

19. medical_visa(Application_ID, Port_of_arrival, Port_of_departure, Hospital, Doctor, Duration, Validity)

- Insertion Anomaly:not present
- Deletion Anomaly:not present
- Updation Anomaly:not present

- The schema was already in 3NF form.

Schema

1. applicant_ac(Applicant_ID, Username, Password)
2. application(Application_ID, Applicant_ID, Officer_ID, Admin_ID, First_name, Last_name, Date_of_birth, Application_status, Gender)
3. admin(Admin_ID, Username, Password)
4. visa_officer(Officer_ID, Admin_ID, First_name, Last_name, Email, Authorized_status)
5. visa_officer_mobile_no(Officer_ID, Mobile_no)

6. address(Application_ID,house_number,Street,City,State,zipcode,Country,Email_ID)
7. applicant_mobile_no(Application_ID,Mobile_no)
8. covid19(Application_ID,Report_no,Report_date,report_tatus,Vaccine_status)
9. applicant_nationality(Application_ID,Nationality)
10. Last/Current_visa(Application_ID,visa_no,visa type,Place of issue,Date of issue)
11. appointment_details(Application_ID,Officer_ID,Date,Time)
12. payment_status(Application_ID,Admin_ID,Amount,
Mode_of_Payment,Payment_approval)
13. passport_details(Application_ID,Passport_No,Date_of_issue,Place_of_issue,
Expiry_date,Nationality)
14. testimonial(Application_ID,arrested,refused_entry_status,
human_drug_trafficing_involvement,child_abuse,financial_fraud,asylum_status)
15. visa(Application_ID,Port_of_arrival,Port_of_departure)
16. student_visa(Application_ID,Port_of_arrival,Port_of_departure,Validity,University_name,
Tution_fee,Course)
17. employment_visa(Application_ID,Port_of_arrival,Port_of_departure,Company name,
Position,Validity)
18. tourist_visa(Application_ID,Port_of_arrival,Port_of_departure,Travel agency,Duration,
Validity)
19. medical_visa(Application_ID,Port_of_arrival,Port_of_departure,Hospital,Doctor,
Duration,Validity)

Section6: SQL: Final DDL Scripts, Insert statements, 40 SQL Queries, Snapshots of output of each query

DDL Script

```
SET SEARCH_PATH TO visa_db

create table visa_db.applicant_ac
(
    applicant_id int unique primary key,
    username varchar(20),
    password varchar(20)
);

create table visa_db.admin
(
    admin_id int unique primary key,
    username varchar(20),
    password varchar(20)
);

create table visa_db.visa_officer
(
    officer_id int unique primary key,
    admin_id int,
    first_name varchar(20),
    last_name varchar(20),
    email varchar(100),
    authorized_status varchar(20),
    foreign key(admin_id) references admin(admin_id) on delete set default on update cascade
);

create table visa_db.visa_officer_mobile
(
    officer_id int,
    mobile_no varchar(20),
    primary key(officer_id,mobile_no),
```

```
foreign key(officer_id) references visa_officer(officer_id) on delete set default on update
cascade
);

create table visa_db.application
(
    application_id int unique,
    applicant_id int,
    officer_id int,
    admin_id int,
    first_name varchar(20),
    last_name varchar(20),
    date_of_birth date,
    gender varchar(10),
    application_status varchar(20),
    primary key(application_id),
    foreign key (applicant_id) references applicant_ac(applicant_id) on delete set default
    on update cascade,
    foreign key (officer_id) references visa_officer(officer_id) on delete set default on
    update cascade,
    foreign key (admin_id) references admin(admin_id) on delete set default on update
    cascade
);

create table visa_db.applicant_nationality
(
    application_id int,
    nationality varchar(50),
    Primary key(application_id,nationality),
    foreign key (application_id) references application(application_id) on delete set default
    on update cascade
);

create table visa_db.applicant_mobile_no
(
    application_id int,
    mobile_no varchar(20),
    Primary key(application_id,mobile_no),
    foreign key (application_id) references application(application_id) on delete set default
    on update cascade
);
```

```

create table visa_db.covid19
(
    application_id int,
    report_no int,
    report_date date,
    report_status varchar(20),
    vaccine_status varchar(20),
    Primary key(application_id,report_no,report_date,report_status,vaccine_status),
    foreign key (application_id) references application(application_id) on delete set default
    on update cascade
);

create table visa_db.appointment_details
(
    application_id int,
    officer_id int,
    date date,
    time varchar(20),
    primary key(application_id,officer_id,date,time),
    foreign key(application_id) references application(application_id) on delete set default
    on update cascade,
    foreign key(officer_id) references visa_officer(officer_id) on delete set default on update
    cascade
);

create table visa_db.last_current_visa
(
    application_id bigint,
    visa_no bigint,
    visa_type varchar(20),
    place_of_issue varchar(50),
    date_of_issue date,
    primary key(application_id,visa_no,visa_type,place_of_issue,date_of_issue),
    foreign key(application_id) references application(application_id) on delete set default
    on update cascade
);

create table visa_db.address
(
    application_id int,
    house_name varchar(50),
    street varchar(50),

```

```
        city varchar(50),
        state varchar(50),
        zipcode varchar(20),
        country varchar(50),
        email_id varchar(100),
        primary key(application_id,house_name,street,city,state,zipcode,country,email_id),
        foreign key(application_id) references application(application_id) on delete set default
on          update cascade
);

```

```
create table visa_db.passport_Details
(
    application_id int,
    passport_no bigint,
    date_of_issue date,
    place_of_issue varchar(50),
    expiry_date date,
Primary Key ( application_id, passport_no, date_of_issue, place_of_issue, expiry_date),
    foreign key(application_id) references application(application_id) on delete set default
on update cascade
);

```

```
create table visa_db.testimonial
(
    application_id int,
    arrest_status varchar(20),
    refused_entrance_status varchar(20),
    human_drug_trafficing_involvement varchar(20),
    child_abuse varchar(20),
    financial_fraud varchar(20),
    asylum_status varchar(20),
    Primary key ( application_id, arrest_status, refused_Entrance_status,
human_Drug_trafficing_involvement, child_Abuse,financial_Fraud, asylum_status),
    foreign key(application_id) references application(application_id) on delete set default
on update cascade
);

```

```
create table visa_db.payment_Status
(
    application_id int,
    admin_id int,
    amount float,
```

```

        mode_of_payment varchar(20),
        payment_approval varchar(20),
        primary key(application_id,admin_id,amount,mode_of_payment,payment_approval),
        foreign key(application_id) references application(application_id) on delete set default
on update cascade,
        foreign key(admin_id) references admin(admin_id) on delete set default on update
cascade
);

create table visa_db.visa
(
    application_id int,
    port_of_Arrival varchar(50),
    port_of_departure varchar(50),
    primary key(application_id,port_of_Arrival,port_of_Departure),
    foreign key(application_id) references application(application_id) on delete set default
on update cascade
);

create table visa_db.student_visa
(
    application_id int,
    port_of_Arrival varchar(50),
    port_of_departure varchar(50),
    university_name varchar(50),
    tuition_fee varchar(20),
    course varchar(20),
    primary key(application_id,port_of_Arrival,port_of_Departure),
    foreign key(application_id) references application(application_id) on delete set default
on update cascade
);

create table visa_db.employment_visa
(
    application_id int,
    port_of_Arrival varchar(50),
    port_of_departure varchar(50),
    company_name varchar(20),
    position varchar(20),
    primary key(application_id,port_of_Arrival,port_of_Departure),
    foreign key(application_id) references application(application_id) on delete set default
on update cascade
);

```

```

);

create table visa_db.tourist_visa
(
    application_id int,
    port_of_Arrival varchar(50),
    port_of_departure varchar(50),
    travel_Agency varchar(50),
    duration varchar(20),
    primary key(application_id,port_of_Arrival,port_of_Departure),
    foreign key(application_id) references application(application_id) on delete set default
on update cascade
);

create table visa_db.medical_visa
(
    application_id int,
    port_of_Arrival varchar(50),
    port_of_departure varchar(50),
    hospital varchar(50),
    doctor varchar(50),
    duration varchar(20),
    primary key(application_id,port_of_Arrival,port_of_Departure),
    foreign key(application_id) references application(application_id) on delete set default
on update cascade
);

```

Data insertion queries :

```
COPY visa_db.applicant_ac(Applicant_ID,Username,Password)
FROM 'E:\Academic\dbms_data\applicant_ac.csv' DELIMITER '|' CSV HEADER;
```

```
COPY visa_db.admin/Admin_ID, Username, Password)
FROM 'E:\Academic\dbms_data\admin.csv' DELIMITER '|' CSV HEADER;
```

```
COPY visa_db.visa_officer(Officer_ID, First_name, Last_name, Email,
Authorized_status,Admin_ID)
FROM 'E:\Academic\dbms_data\visa_officer.csv' DELIMITER '|' CSV HEADER;
```

```
COPY
visa_db.application(Application_ID,Applicant_ID,Officer_ID,Admin_ID,First_name,Last_name,Date
_of_birth, Gender,Application_status)
FROM 'E:\Academic\dbms_data\application.csv' DELIMITER '|' CSV HEADER;

COPY visa_db.visa_officer_mobile(Officer_ID, Mobile_no)
FROM 'E:\Academic\dbms_data\visa_officer_mobile.csv' DELIMITER ',' CSV HEADER;

COPY visa_db.address(Application_ID, house_name, street, City, State, zipcode, Country,
Email_ID)
FROM 'E:\Academic\dbms_data\address.csv' DELIMITER '|' CSV HEADER;

COPY visa_db.applicant_mobile_no(Application_ID, Mobile_no)
FROM 'E:\Academic\dbms_data\applicant_mobile_no.csv' DELIMITER ',' CSV HEADER;

COPY visa_db.covid19(Application_ID, Report_no, Report_date, report_status, Vaccine_status)
FROM 'E:\Academic\dbms_data\covid19.csv' DELIMITER '|' CSV HEADER;

COPY visa_db.applicant_nationality(Application_ID, Nationality)
FROM 'E:\Academic\dbms_data\applicant_nationality.csv' DELIMITER ',' CSV HEADER;

COPY visa_db.Last_current_visa(Application_ID, visa_no, visa_type, Place_of_issue,
Date_of_issue)
FROM 'E:\Academic\dbms_data\Last_Current_visa.csv' DELIMITER '|' CSV HEADER;

COPY visa_db.appointment_details(Application_ID, Officer_ID, Date, Time)
FROM 'E:\Academic\dbms_data\appointment_details.csv' DELIMITER '|' CSV HEADER;

COPY visa_db.payment_status(Application_ID, Admin_ID, Amount,
Mode_of_Payment,Payment_approval)
FROM 'E:\Academic\dbms_data\payment_status.csv' DELIMITER '|' CSV HEADER;

COPY visa_db.passport_details(Application_ID, Passport_No, Date_of_issue, Place_of_issue,
Expiry_date)
FROM 'E:\Academic\dbms_data\passport_details.csv' DELIMITER '|' CSV HEADER;

COPY visa_db.testimonial(Application_ID,arrest_status, refused_entrance_status,
human_drug_trafficing_involvement , child_abuse , financial_fraud,asylum_status )
FROM 'E:\Academic\dbms_data\testimonial.csv' DELIMITER '|' CSV HEADER;

COPY visa_db.visa(Application_ID, Port_of_arrival, Port_of_departure)
FROM 'E:\Academic\dbms_data\visa.csv' DELIMITER '|' CSV HEADER;
```

```
COPY visa_db.student_visa(Application_ID, Port_of_arrival, Port_of_departure, University_name,  
Tuition_fee, Course)
```

```
FROM 'E:\Academic\dbms_data\student_visa.csv' DELIMITER '|' CSV HEADER;
```

```
COPY visa_db.employment_visa(Application_ID, Port_of_arrival, Port_of_departure  
, Company_name, Position)
```

```
FROM 'E:\Academic\dbms_data\employment_visa.csv' DELIMITER '|' CSV HEADER;
```

```
COPY visa_db.tourist_visa(Application_ID, Port_of_arrival, Port_of_departure, Travel_agency,  
Duration)
```

```
FROM 'E:\Academic\dbms_data\tourist_visa.csv' DELIMITER '|' CSV HEADER;
```

```
COPY visa_db.medical_visa(Application_ID, Port_of_arrival, Port_of_departure, Hospital, Doctor,  
Duration)
```

```
FROM 'E:\Academic\dbms_data\medical_visa.csv' DELIMITER '|' CSV HEADER;
```

All select queries :

```
select * from applicant_ac
```

```
select * from admin
```

```
select * from visa_officer
```

```
select * from application
```

```
select * from visa_officer_mobile
```

```
select * from address
```

```
select * from applicant_mobile_no
```

```
select * from covid19
```

```
select * from applicant_nationality
```

```
select * from last_current_visa
```

```
select * from appointment_details
```

```
select * from payment_status
```

```
select * from passport_details
```

```
select * from testimonial
```

```
select * from visa
```

```
select * from student_visa
```

```
select * from medical_visa
```

```
select * from tourist_visa
```

```
select * from employment_visa
```

Simple Queries

1. Find application ID of applicants whose fees payment will be done in Offline mode and isn't approved.

A:

```
select application_id  
from payment_status  
where mode_of_payment='Offline' and payment_approval='Pending'
```

No. of tuples=17

The screenshot shows the PostgreSQL Query Editor interface. The query in the editor is:

```
1 select application_id  
2 from payment_status  
3 where mode_of_payment='Offline' and payment_approval='Pending'  
4
```

The results are displayed in a table titled "Data Output". The table has one column labeled "application_id" with a data type of "integer". The data consists of 17 rows, numbered 8 to 17, with values ranging from 100054 to 100090.

	application_id
8	100054
9	100060
10	100063
11	100067
12	100071
13	100073
14	100075
15	100079
16	100086
17	100090

2. Count the total number of appointments held in last three months of 2020

A:

```
select count(application_id)  
from appointment_details  
where date>= '1/10/2020' and date<='31/12/2020'
```

No. of tuples=1

The screenshot shows the PostgreSQL Query Editor interface. The query in the editor is:

```
1 select count(application_id)  
2 from appointment_details  
3 where date>= '1/10/2020' and date<='31/12/2020'  
4
```

The results are displayed in a table titled "Data Output". The table has one column labeled "count" with a data type of "bigint". There is a single row with a value of 14.

	count
1	14

3. Find details of applicants with any type type criminal record

A:

```
select *
from testimonial
where arrest_status ='Yes' or refused_entrance_status='Yes'
      or human_drug_trafficing_involvement='Yes' or child_abuse='Yes'
      or financial_fraud ='Yes' or asylum_status='Yes'
```

No. of tuples=54

	application_id [PK] integer	arrest_status [PK] character varying (20)	refused_entrance_status [PK] character varying (20)	human_drug_trafficing_involvement [PK] character varying (20)	child_abuse [PK] character varying (20)	financial_fraud [PK] c
46	100084	No	Yes	No	No	No
47	100085	No	Yes	No	No	No
48	100086	No	Yes	No	No	No
49	100088	No	Yes	No	No	No
50	100089	No	Yes	No	No	No
51	100093	No	Yes	No	No	No
52	100096	No	Yes	No	No	No
53	100097	No	Yes	No	No	No
54	100098	No	Yes	No	No	No

4. Find the count of total number of applications from each country

A:

```
select address.country, count(application_id)
from address
group by country
```

No. of tuples

	country character varying (50)	count bigint
3	England	10
4	UK	10
5	Germany	10
6	China	9
7	Russia	10
8	Canada	10
9	Nationality	1
10	USA	10
11	France	11

5. Find the count of total number of applications from each country for each state in ascending order

A:

```
select country,state,count(application_id)
from address
group by country,state
order by (country,state) asc
```

No.of tuples:72

country	state	count
character varying (50)	character varying (50)	bigint
64 UK	Ontario	1
65 UK	South Gyeongsang	2
66 USA	Alberta	1
67 USA	Essex	2
68 USA	Gyeonggi	1
69 USA	Hamburg	2
70 USA	Kostroma Oblast	1
71 USA	New South Wales	2
72 USA	South Gyeongsang	1

6. select all applicants with negative covid report.

select * from covid19

where report_status='Negative'

No of tuples : 52

application_id	report_no	report_date	report_status	vaccine_status
[PK] integer	[PK] integer	[PK] character varying(20)	[PK] character varying(20)	[PK] character varying(20)
1	100000	13629063	09/06/2021	Negative
2	100002	16589639	14/01/2021	Negative
3	100004	19291032	08/06/2021	Negative
4	100008	13121429	27/08/2021	Negative
5	100010	10577896	29/02/2020	Negative
6	100011	11687569	28/05/2020	Negative
7	100015	15225494	11/06/2021	Negative
8	100016	19275448	06/01/2021	Negative
9	100017	14262739	20/05/2020	Negative
10	100019	12468560	14/12/2019	Negative
11	100021	13613866	13/04/2021	Negative
12	100022	10875233	26/04/2020	Negative
13	100025	11005898	06/05/2021	Negative
14	100026	14110856	21/07/2021	Negative
15	100029	19253980	11/11/2020	Negative

7. select all applications involved in human/drug trafficking.

```
select * from testimonial  
where human_drug_trafficking_involvement='Yes'  
No of tuples: 0
```

The screenshot shows a PostgreSQL query editor interface. The title bar says "visa_db/postgres@PostgreSQL 10". The main area contains the following SQL code:

```
1 select * from testimonial  
2 where human_drug_trafficking_involvement='Yes'  
3
```

Below the code, there are tabs for "Data Output", "Explain", "Messages", and "Notifications". The "Data Output" tab is selected, showing a table with the following columns and data:

	application_id [PK] integer	arrest_status [PK] character varying (20)	refused_entrance_status [PK] character varying (20)	human_drug_trafficking_involvement [PK] character varying (20)	child_abuse [PK] character varying (20)	financial_fraud [PK] character varying (20)
1						

8. Select all visa officers authorized by admin with ID 2.

```
select * from visa_officer  
where admin_id=2 and authorized_status='Yes'  
No. of tuples: 2
```

The screenshot shows a PostgreSQL query editor interface. The title bar says "visa_db/postgres@PostgreSQL 10". The main area contains the following SQL code:

```
1 select * from visa_officer  
2 where admin_id=2 and authorized_status='Yes'  
3
```

Below the code, there are tabs for "Data Output", "Explain", "Messages", and "Notifications". The "Data Output" tab is selected, showing a table with the following columns and data:

	officer_id [PK] integer	admin_id integer	first_name character varying (20)	last_name character varying (20)	email character varying (100)
1	1001	2	Fritz	Kim	Nulla.interdum@semvitaealiqu...
2	1005	2	Coby	Hogan	eros@nec.org

9. count no of applicants who made online payment for visa procedure.

```
select count(application_id)  
from payment_status  
where mode_of_payment='Online'
```

No. of tuples=1

visa_db/postgres@PostgreSQL 10

Query Editor Query History

```

1 select count(application_id)
2 from payment_status
3 where mode_of_payment='Online'
4

```

Data Output Explain Messages Notifications

	count	bigint
1	49	

10. select all applicants who are applying for the CEO position on an employment visa.

```

select *
from employment_visa
where position='CEO'

```

No of tuples : 3

```

1 select *
2 from employment_visa
3 where position='CEO'
4

```

Data Output Explain Messages Notifications

	application_id	port_of_arrival	port_of_departure	company_name	position
	[PK] integer	[PK] character varying (50)	[PK] character varying (50)	character varying (20)	character varying (20)
1	100004	Amritsur	Pune	JIO	CEO
2	100013	Amritsur	Pune	Vipro	CEO
3	100020	Delhi	Kolkata	Zomato	CEO

11. Find the total number of visa officers who are not authorized.

```

select count(officer_id)
from visa_officer
where authorized_status='No'

```

No. of tuples = 1

```

visa_project/postgres@PostgreSQL 10
Query Editor Query History
1 select count(officer_id)
2 from visa_officer
3 where authorized_status='No'

```

Data Output Explain Messages Notifications

	count	bigint
1	5	

12. Find the names of the students who have applied for student visa for university 'IIT-B'

```

select first_name,last_name
from application
where application_id in (select student_visa.application_id
                           from student_visa
                           where university_name = 'IIT-B')

```

No. of tuples = 6

```

visa_project/postgres@PostgreSQL 10
Query Editor Query History
1 select first_name,last_name
2 from application
3 where application_id in (select student_visa.application_id
                           from student_visa
                           where university_name = 'IIT-B')

```

Data Output Explain Messages Notifications

	first_name	last_name
1	Graham	Kelly
2	Lane	Fleming
3	Oscar	Bowman
4	Ciaran	Bauer
5	Troy	Wiggins
6	Rigel	Carver

13. Find all applications whose applicant's age is more than 30.

```
select distinct *
from application
where extract(year from age(current_date,date_of_birth)) > 30
```

No. of tuples = 46

The screenshot shows a PostgreSQL query editor window titled "visa_project/postgres@PostgreSQL 10". The query in the editor is:

```
1 select distinct *
2 from application
3 where extract(year from age(current_date,date_of_birth)) > 30
```

The results are displayed in a table titled "Data Output". The table has 12 columns corresponding to the fields in the application table. The columns are:

	application_id	applicant_id	officer_id	admin_id	first_name	last_name	date_of_birth	gender	application_status
31	100096	12818240	1009	4	Ulric	Massey	1984-03-29	Male	Approved
32	100069	67393090	1002	4	Lars	Sampson	1988-12-25	Male	Pending
33	100018	34777989	1001	4	Knox	Nguyen	1989-03-15	Female	Pending
34	100051	23400816	1007	2	Giacomo	Spencer	1989-03-22	Male	Pending
35	100089	29074650	1005	5	Ulysses	Cross	1986-08-12	Male	Pending
36	100080	36181992	1005	5	Quinlan	Salas	1989-09-24	Female	Pending
37	100023	35967735	1001	3	Lev	Myers	1981-06-17	Female	Pending
38	100004	10361676	1009	5	George	Chambers	1989-10-13	Female	Pending
39	100000	26030746	1001	2	Declan	Mcconnell	1984-02-17	Female	Approved
40	100046	67232295	1007	2	Lucian	Monroe	1980-08-18	Male	Approved
41	100013	51444818	1001	3	Patrick	Larson	1989-05-23	Male	Approved
42	100083	47259552	1002	2	Gil	Harding	1985-12-10	Male	Approved
43	100036	59145564	1007	2	Griffith	Henson	1987-07-28	Female	Pending
44	100031	69084573	1002	4	Brett	Ramos	1989-05-07	Female	Pending
45	100062	60249077	1009	4	Christian	Gamble	1983-01-10	Male	Pending
46	100088	39769641	1002	3	Elijah	Cline	1980-02-03	Female	Approved

14. Find the total number of student visa applications.

```
select count(*)
from student_visa
```

No. of tuples = 27

The screenshot shows a PostgreSQL query editor window titled "visa_project/postgres@PostgreSQL 10". The query in the editor is:

```
1 select count(*)
2 from student_visa
```

The results are displayed in a table titled "Data Output". The table has one column labeled "count".

count
27

15. Find all applications whose applicant is female.

```
select *  
from application  
where gender = 'Female'
```

No. of tuple = 52

	application_id	applicant_id	officer_id	admin_id	first_name	last_name	date_of_birth	gender	application_status
38	100068	61433340	1001	3	Xavier	Gaines	1997-07-09	Female	Approved
39	100070	57198575	1005	5	Holmes	Tyson	1990-03-17	Female	Pending
40	100073	10660021	1001	2	Josiah	Pratt	1994-12-16	Female	Pending
41	100076	12411340	1007	1	Ryan	Johns	1993-10-19	Female	Pending
42	100078	20186217	1001	3	Andrew	Burns	1993-03-13	Female	Pending
43	100079	66996793	1002	4	Nigel	Caldwell	2002-10-14	Female	Approved
44	100080	36181992	1005	5	Quinlan	Salas	1989-09-24	Female	Pending
45	100081	20576510	1007	2	Burke	Coffey	1995-08-06	Female	Pending
46	100085	35447209	1007	2	Tad	Woodward	1999-10-01	Female	Approved
47	100088	39769641	1002	3	Elijah	Cline	1980-02-03	Female	Approved
48	100092	48605713	1001	3	Marsden	Puckett	2004-08-29	Female	Pending
49	100093	38761783	1002	4	Thor	Mcintosh	1986-05-30	Female	Approved
50	100094	12821168	1005	5	Lee	House	1984-12-13	Female	Approved
51	100095	48469592	1007	2	Travis	Woods	1981-12-09	Female	Pending
52	100098	67975198	1002	2	Ahmed	Tyler	1980-07-07	Female	Pending

16. Count all the German applicants applying for visas.

```
select application_id, country from address
```

```
where country='Germany';
```

No, of tuples: 10

	application_id	country
1	100001	Germany
2	100011	Germany
3	100021	Germany
4	100031	Germany
5	100041	Germany
6	100051	Germany
7	100061	Germany
8	100073	Germany
9	100083	Germany
10	100093	Germany

17. Find all the canadian applicants applying for tourist visa.

```
select application_id,country from address  
where country='Canada' and application_id in (select tourist_visa.application_id  
from tourist_visa);
```

No. of tuples: 3

The screenshot shows a PostgreSQL Query Editor window. The title bar says 'visaproject/postgres@PostgreSQL 10'. The 'Query Editor' tab is selected. The query is:1 select application_id,country from address
2 where country='Canada' and application_id in (select tourist_visa.application_id
3 from tourist_visa);

```
Below the query, there are tabs for 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is selected, showing a table with two columns: 'application_id' (integer) and 'country' (character varying(50)). The data is:|  | application_id | country |
| --- | --- | --- |
| 1 | 100076 | Canada |
| 2 | 100086 | Canada |
| 3 | 100096 | Canada |

```

18. Count all the tourist visa applicants whose travel agency is Make my trip.

```
select count(*)  
from tourist_visa  
where travel_agency='Make My Trip'
```

No. of tuples: 1

The screenshot shows a PostgreSQL Query Editor window. The title bar says 'visaproject/postgres@PostgreSQL 10'. The 'Query Editor' tab is selected. The query is:1 select count(*)
2 from tourist_visa
3 where travel_agency='Make My Trip'

```
Below the query, there are tabs for 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is selected, showing a table with one column 'count' (bigint). The data is:|  | count |
| --- | --- |
| 1 | 5 |

```

19. Find all the applicants who already have a medical visa and are now applying for a tourist visa.

```
select application_id  
from last_current_visa  
where visa_type='Medical' and application_id in(select tourist_visa.application_id  
from tourist_visa)
```

No.of tuples: 5

The screenshot shows a PostgreSQL query editor window. The title bar says "visaproject/postgres@PostgreSQL 10". The query tab contains the following SQL code:

```
1 select application_id
2 from last_current_visa
3 where visa_type='Medical' and application_id in(select tourist_visa.application_id
4                                                 from tourist_visa)
5
```

The results tab shows a table with one column "application_id" and five rows:

	application_id
1	100076
2	100083
3	100088
4	100090
5	100093

20. Find the total no. of visa applications that are pending for visa officer with ID 1002.

```
select count(*)
from application
where application_status='Pending' and officer_id in(select visa_officer.officer_id
                                                       from visa_officer
                                                       where officer_id=1002)
```

The screenshot shows a PostgreSQL query editor window. The title bar says "visaproject/postgres@PostgreSQL 10". The query tab contains the following SQL code:

```
1 select count(*)
2 from application
3 where application_status='Pending' and officer_id in(select visa_officer.officer_id
                                                       from visa_officer
                                                       where officer_id=1002)
4
5
6
```

The results tab shows a table with one column "count" and one row:

	count
1	9

Complex Queries

1. Find the Officer who has been scheduled highest number of appointments in 2020

A:

```
select officer_id,count(application_id)
from appointment_details
where date>='1/01/2020' and date<='31/12/2020'
group by officer_id
having count(application_id) in
    (select max(counts)
     from(
      select count(application_id) counts
      from appointment_details
      where date>='1/01/2020' and date<='31/12/2020'
      group by officer_id
     ) as bb)
```

No.of tuples:3

The screenshot shows a PostgreSQL query editor interface. The title bar says "visa/postgres@PostgreSQL 10". The main area is titled "Query Editor" and contains the following SQL code:

```
1 select officer_id, count(application_id)
2 from appointment_details
3 where date>='1/01/2020' and date<='31/12/2020'
4 group by officer_id
5 having count(application_id) in
6     (select max(counts)
7      from(
8       select count(application_id) counts
9       from appointment_details
10      where date>='1/01/2020' and date<='31/12/2020'
11      group by officer_id
12     ) as bb)
```

Below the code, there are tabs for "Data Output", "Explain", "Messages", "Query History", and "Notifications". The "Data Output" tab is selected, showing a table with three rows of data:

	officer_id	count
1	1007	8
2	1005	8
3	1009	8

2. Find the details of applicants who has applied for employment visa and are SDE

A:

```
select *  
from application  
join employment_visa on application.application_id=employment_visa.application_id  
where position='SDE-3'or position='SDE-2'or position='SDE-1'
```

No.of tuples:9

Query Editor

```
1 select *  
2 from application  
3 join employment_visa on application.application_id=employment_visa.application_id  
4 where position='SDE-3'or position='SDE-2'or position='SDE-1'  
5
```

Data Output Explain Messages Query History Notifications

	application_id	applicant_id	officer_id	admin_id	first_name	last_name	date_of_birth	gender
3	100002	37881566	1003	1003	Z Marsnair	Bradford	1999-01-29	Male
4	100009	39893281	1009	5	Gary	Baldwin	1990-09-01	Male
5	100010	65607022	1001	2	Josiah	Bishop	1987-08-24	Male
6	100011	24852854	1002	4	Wayne	Rivers	2001-02-09	Male
7	100016	40187104	1007	2	Logan	Blackburn	1992-12-22	Male
8	100017	23160552	1009	3	Patrick	Sawyer	2003-09-17	Male
9	100018	34777989	1001	4	Knox	Nguyen	1989-03-15	Female

3. Find report status of covid 19 of all approved medical visa applications.

A:

```
select aa.application_id,report_status  
from application aa  
inner join medical_visa m_v on aa.application_id=m_v.application_id  
inner join covid19 cc on m_v.application_id=cc.application_id  
where application_status='Approved'
```

No. of tuples:11

Query Editor

```
1 select aa.application_id,report_status  
2 from application aa  
3 inner join medical_visa m_v on aa.application_id=m_v.application_id  
4 inner join covid19 cc on m_v.application_id=cc.application_id  
5 where application_status='Approved'  
6
```

Data Output Explain Messages Query History Notifications

	application_id	report_status
5	100035	Negative
6	100038	Positive
7	100039	Negative
8	100041	Negative
9	100044	Negative
10	100046	Positive
11	100049	Positive

4. Find the email ID of all the officers who have an appointment with applicants applying for medical_visa.

A:

```
select distinct(email)
```

```
from visa_officer v inner join appointment_details a_d on v.officer_id=a_d.officer_id  
inner join medical_visa m_v on m_v.application_id=a_d.application_id
```

No. of tuples:10

```
1 select distinct(email)  
2 from visa_officer v inner join appointment_details a_d on v.officer_id=a_d.officer_id  
3 inner join medical_visa m_v on m_v.application_id=a_d.application_id  
4
```

Data Output Explain Messages Query History Notifications

email
character varying (100)
4 Nulla.interdum@semvitaealiqu...
5 amet.luctus.vulputate@elitse...
6 eros@nec.org
7 iacus.Nulla.tincidunt@utnulla...
8 luctus.vulputate.nisi@Integert...
9 felis.purus@Nullafacilisis.co.uk
10 pretium@risuslnmi.com

✓ Successfully run. Total

5. Find the total number applicants for each country, who have applied for an employment visa to work for Google , Microsoft or Amazon .

A:

```
select country, count(employment_visa.application_id)
```

```
from address
```

```
inner join employment_visa on address.application_id=employment_visa.application_id  
where company_name='Google' or company_name='Microsoft' or  
company_name='Amazon'
```

```
group by country
```

No. of tuples:3

```

1 select country, count(employment_visa.application_id)
2 from address
3 inner join employment_visa on address.application_id=employment_visa.application_id
4 where company_name='Google' or company_name='Microsoft' or company_name='Amazon'
5 group by country
6

```

Data Output Explain Messages Query History Notifications

country	count
character varying (50)	bigint
1 China	2
2 Nepal	1
3 South Africa	1

6. Group all the cities by no of students coming there for student visa for course "Computer Science".

```

select port_of_arrival , count(application_id)
from student_visa
where course='CS'
group by port_of_arrival

```

No. of Tuples = 9

```

1 select port_of_arrival , count(application_id)
2 from student_visa
3 where course='CS'
4 group by port_of_arrival

```

Data Output Explain Messages Notifications

port_of_arrival	count
character varying (50)	bigint
1 Amritsur	1
2 Banglore	1
3 Chennai	3
4 Delhi	1
5 Hydrabad	2
6 Kolkata	1
7 Mumbai	2
8 Pune	1
9 Rajkot	2

7. Find full name of applicants who applied for tourist visa and not having covid vaccine.

```
select application.first_name,application.last_name  
from tourist_visa  
join application on tourist_visa.application_id = application.application_id  
join covid19 on covid19.application_id = application.application_id  
where vaccine_status='No'
```

No of Tuples = 10

	first_name	last_name	
1	Ryan	Johns	
2	Nigel	Caldwell	
3	Quinlan	Salas	
4	Gil	Harding	
5	Tad	Woodward	
6	Ulysses	Cross	
7	Donovan	Pope	
8	Travis	Woods	
9	Kamal	Ford	
10	Tucker	Trevino	

8. Show how many applicants are coming to AIIMS on a medical visa and group them by their nationality.

```
select applicant_nationality.nationality,count(application.application_id)  
from medical_visa  
join application on medical_visa.application_id = application.application_id  
join applicant_nationality on application.application_id = applicant_nationality.application_id  
where hospital='AIIMS'  
group by applicant_nationality.nationality
```

No of tuples : 4

```

1 select applicant_nationality.nationality, count(application.application_id)
2 from medical_visa
3 join application on medical_visa.application_id = application.application_id
4 join applicant_nationality on application.application_id = applicant_nationality.application_id
5 where hospital='AIIMS'
6 group by applicant_nationality.nationality
7
8

```

Data Output Explain Messages Notifications

	nationality	count
	character varying (50)	bigint
1	England	1
2	Germany	1
3	Nepal	1
4	UK	1

9. Find the full name of applicants whose covid report is negative and they did covid test before their appointment with a consultant officer.

```

select first_name, last_name
from application
join appointment_details on application.application_id = appointment_details.application_id
join covid19 on application.application_id = covid19.application_id
where report_status='Negative' and covid19.report_date < appointment_details.date

```

No of Tuples : 19

```

1 select first_name, last_name
2 from application
3 join appointment_details on application.application_id = appointment_details.application_id
4 join covid19 on application.application_id = covid19.application_id
5 where report_status='Negative' and covid19.report_date < appointment_details.date

```

Data Output Explain Messages Notifications

	first_name	last_name
	character varying (20)	character varying (20)
5	Arthur	Logan
6	Carl	Adams
7	Devin	Hull
8	Cole	Leach
9	Fitzgerald	Donaldson
10	Warren	Guzman
11	Clinton	Skinner
12	Holmes	Tyson
13	Ryan	Johns
14	Nigel	Caldwell
15	Quinlan	Salas
16	Burke	Coffey
17	Merrill	Sanchez
18	Gil	Harding
19	Kamal	Ford

10. List all universities according to course offered and no. of students applied for student visa for that course.

```
select university_name, course, count(application_id)
from student_visa
group by university_name, course
order by university_name
```

No of Tuples: 18

1	select university_name, course, count(application_id)
2	from student_visa
3	group by university_name, course
4	order by university_name

Data Output Explain Messages Notifications

	university_name character varying (50)	course character varying (20)	count bigint
1	AMITYA	Civil	2
2	BITS	CS	2
3	DAIICT	CS	2
4	IIT-B	Civil	2
5	IIT-B	CS	3
6	IIT-B	Mechanical	1
7	IIT-KH	CS	1
8	IIT-KH	Mechanical	2
9	Nirma	CS	1
10	Nirma	Mechanical	1
11	NITK	CS	2
12	PDPU	CS	1
13	PDPU	Mechanical	1
14	SVNIT	Civil	1
15	SVNIT	CS	1

11. Create a trigger function which will ensure that insert, update , delete operations performed on the visa_officer table are recorded in the visa_officer_audit table.

```
create table visa_db.visa_officer_audit
(
operation char(1),
stamp timestamp,
officer_id int,
admin_id int
)
```

```

create or replace function visa_db.audit()
returns trigger
language 'plpgsql'
as $body$
begin
    if (TG_OP = 'DELETE') then
        insert into visa_officer_audit(operation,stamp,officer_id,admin_id)
        values ('D',current_timestamp,OLD.officer_id,OLD.admin_id);
        return OLD;
    elsif (TG_OP = 'UPDATE') then
        insert into visa_officer_audit(operation,stamp,officer_id,admin_id)
        values ('U',current_timestamp,NEW.officer_id,NEW.admin_id);
        return NEW;
    elsif (TG_OP = 'INSERT') then
        insert into visa_officer_audit(operation,stamp,officer_id,admin_id)
        values ('I',current_timestamp,NEW.officer_id,NEW.admin_id);
        return NEW;
    end if;
    RETURN NULL;
end
$body$;

```

```

create trigger "trigger_visa_officer_audit"
AFTER INSERT OR UPDATE OR DELETE ON visa_officer
for each row
execute procedure visa_db.audit();

```

```

insert into visa_officer values ('123','1','krutik','parmar','as@gmail.com','No')
select * from visa_officer_audit

```

```

update visa_officer set admin_id = 2 where officer_id = 123
select * from visa_officer_audit

```

```

delete from visa_officer where officer_id = 123
select * from visa_officer_audit

```

Insert :

visa_project/postgres@PostgreSQL 10

Query Editor Query History

```

24     values ('I',current_timestamp,NEW.officer_id,NEW.admin_id);
25     return NEW;
26   end if;
27 RETURN NULL;
28 end
29 $body$;

31
32 create trigger "trigger_visa_officer_audit"
33 AFTER INSERT OR UPDATE OR DELETE ON visa_officer
34 for each row
35 execute procedure visa_db.audit();
36
37 insert into visa_officer values ('123','1','krutik','parmar','as@gmail.com','No')
38 select * from visa_officer_audit

```

Data Output Explain Messages Notifications

operation	stamp	officer_id	admin_id
character(1)	timestamp without time zone	integer	integer
I	2020-12-04 11:39:52.140734	123	1

✓ Successfully run. Total query runtime: 79 msec. 1 rows affected.

Update:

visa_project/postgres@PostgreSQL 10

Query Editor Query History

```

28 end
29 $body$;
30
31
32 create trigger "trigger_visa_officer_audit"
33 AFTER INSERT OR UPDATE OR DELETE ON visa_officer
34 for each row
35 execute procedure visa_db.audit();
36
37 insert into visa_officer values ('123','1','krutik','parmar','as@gmail.com','No')
38 select * from visa_officer_audit
39
40 update visa_officer set admin_id = 2 where officer_id = 123
41 select * from visa_officer_audit
42

```

Data Output Explain Messages Notifications

operation	stamp	officer_id	admin_id
character(1)	timestamp without time zone	integer	integer
I	2020-12-04 11:39:52.140734	123	1
U	2020-12-04 11:41:51.542038	123	2

✓ Successfully run. Total query runtime: 66 msec. 2 rows affected.

Delete :

The screenshot shows the pgAdmin interface with a query editor tab open. The title bar says "visa_project/postgres@PostgreSQL 10". The query editor contains the following SQL code:

```
31
32 create trigger "trigger_visa_officer_audit"
33 AFTER INSERT OR UPDATE OR DELETE ON visa_officer
34 for each row
35 execute procedure visa_db.audit();
36
37 insert into visa_officer values ('123','1','krutik','parmar','as@gmail.com','No')
38 select * from visa_officer_audit
39
40 update visa_officer set admin_id = 2 where officer_id = 123
41 select * from visa_officer_audit
42
43 delete from visa_officer where officer_id = 123
44 select * from visa_officer_audit
45
```

Below the code, there are four tabs: Data Output, Explain, Messages, and Notifications. The Data Output tab shows the results of the audit trigger execution:

operation	stamp	officer_id	admin_id
I	2020-12-04 11:39:52.140734	123	1
U	2020-12-04 11:41:51.542038	123	2
D	2020-12-04 11:42:58.08173	123	2

12. Find all applicant's names and application ids who have applied for medical visa and whose age is less than 20.

```
select application_id,first_name,last_name
from application
where exists (select application_id from medical_visa)
and
extract(year from age(current_date,date_of_birth)) < 20
```

No. of tuples = 16

The screenshot shows the pgAdmin interface with a query editor tab open. The title bar says "visa_project/postgres@PostgreSQL 10". The query editor contains the following SQL code:

```
1 select application_id,first_name,last_name
2 from application
3 where exists (select application_id from medical_visa)
4 and
5 extract(year from age(current_date,date_of_birth)) < 20
6
```

Below the code, there are four tabs: Data Output, Explain, Messages, and Notifications. The Data Output tab shows the results of the query:

application_id	first_name	last_name
3	Fitzgerald	Lindsey
4	Patrick	Sawyer
5	Brody	Wyatt
6	Gregory	Zamora
7	August	Stevens
8	Brendan	Moses
9	Nasim	Mays
10	Oscar	Bowman
11	Beck	Neal
12	Nigel	Caldwell
13	Merrill	Sanchez
14	Donovan	Pope
15	Marsden	Puckett
16	Tucker	Trevino

13. What is the average age of student applying for same course.

```
select course,avg(extract(year from age(current_date,date_of_birth)))
from application join student_visa on application.application_id =
student_visa.application_id
group by course
```

No. of tuples : 3

Query Editor Query History

```
1 select course,avg(extract(year from age(current_date,date_of_birth)))
2 from application join student_visa on application.application_id = student_visa.application_id
3 group by course
4
```

Data Output Explain Messages Notifications

course	avg
character varying (20)	double precision
Mechanical	26.2857142857143
Civil	30
CS	31.5

14. Find the name of those patients and doctors whose name starts with same letter and the patient has applied for medical visa to get treatment from same doctor.

```
select first_name,doctor
from application join medical_visa on application.application_id =
medical_visa.application_id
where left(first_name,1) like left(doctor,1)
```

No of tuples = 1

visa_project/postgres@PostgreSQL 10

Query Editor Query History

```
1 select first_name,doctor
2 from application join medical_visa on application.application_id = medical_visa.application_id
3 where left(first_name,1) like left(doctor,1)
```

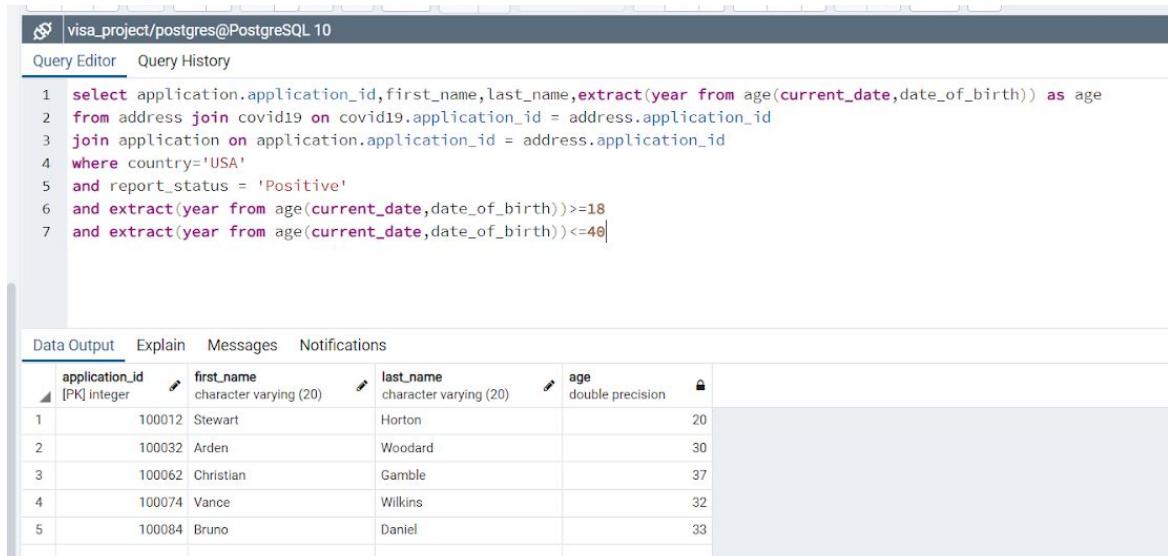
Data Output Explain Messages Notifications

first_name	doctor
character varying (20)	character varying (50)
Raphael	Raj Bhatteja

15. Find the application_id, name and age of patients who is COVID19 positive and age is between 18 to 40 and living in USA.

```
select application.application_id,first_name,last_name,extract(year from
age(current_date,date_of_birth)) as age
from address join covid19 on covid19.application_id = address.application_id
join application on application.application_id = address.application_id
where country='USA'
      and report_status = 'Positive'
      and extract(year from age(current_date,date_of_birth))>=18
      and extract(year from age(current_date,date_of_birth))<=40
```

No. of tuples = 5



The screenshot shows a PostgreSQL query editor window titled "visa_project/postgres@PostgreSQL 10". The "Query Editor" tab is selected. The query text is as follows:

```
1 select application.application_id,first_name,last_name,extract(year from age(current_date,date_of_birth)) as age
2 from address join covid19 on covid19.application_id = address.application_id
3 join application on application.application_id = address.application_id
4 where country='USA'
5 and report_status = 'Positive'
6 and extract(year from age(current_date,date_of_birth))>=18
7 and extract(year from age(current_date,date_of_birth))<=40
```

Below the query, there is a table titled "Data Output" with the following data:

	application_id	first_name	last_name	age
1	100012	Stewart	Horton	20
2	100032	Arden	Woodard	30
3	100062	Christian	Gamble	37
4	100074	Vance	Wilkins	32
5	100084	Bruno	Daniel	33

16. Find the total no. of applicants who got approved for a tourist visa this year by each visa officer.

```
select application.officer_id, count(application.application_id)
from application join tourist_visa on
application.application_id=tourist_visa.application_id
where application_status='Approved'
group by application.officer_id
```

No.of tuples: 5

The screenshot shows a PostgreSQL Query Editor interface. At the top, there are tabs for 'Query Editor' and 'Query History'. Below the tabs is a code editor containing the following SQL query:

```

1 select application.officer_id, count(application.application_id)
2 from application join tourist_visa on application.application_id=tourist_visa.application_id
3 where application_status='Approved'
4 group by application.officer_id

```

Below the code editor is a navigation bar with tabs: 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is selected. A table is displayed with the following data:

	officer_id	count
	integer	bigint
1	1002	4
2	1001	1
3	1007	2
4	1009	2
5	1005	2

17. Find application status of all canadian applications scheduled to visa officer with ID 1007.

```

select application_status,visa_officer.officer_id,country
from application join address on application.application_id=address.application_id
join visa_officer on visa_officer.officer_id=application.officer_id
where address.country='Canada' and visa_officer.officer_id=1007

```

No. of tuples: 1

The screenshot shows a PostgreSQL Query Editor interface. At the top, there are tabs for 'Query Editor' and 'Query History'. The title bar indicates the session is 'visaproject/postgres@PostgreSQL 10'. Below the tabs is a code editor containing the following SQL query:

```

1 select application_status,visa_officer.officer_id,country
2 from application join address on application.application_id=address.application_id
3 join visa_officer on visa_officer.officer_id=application.officer_id
4 where address.country='Canada' and visa_officer.officer_id=1007
5

```

Below the code editor is a navigation bar with tabs: 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is selected. A table is displayed with the following data:

	application_status	officer_id	country
	character varying (20)	integer	character varying (50)
1	Pending	1007	Canada

18. Find total number of pending applications for each country.

```
select address.country,count(application.application_id)
from address join application on application.application_id=address.application_id
where application.application_status='Pending'
group by address.country
```

No.of tuples: 10

Query Editor Query History

```
1 select address.country,count(application.application_id)
2 from address join application on application.application_id=address.application_id
3 where application.application_status='Pending'
4 group by address.country
```

Data Output Explain Messages Notifications

	country character varying (50)	count bigint
1	South Africa	5
2	Nepal	6
3	England	6
4	UK	6
5	Germany	5
6	Russia	5
7	China	4
8	Canada	7
9	USA	5
10	France	5

19. Find the total number of applicants whose covid19 report is positive from each country.

```
select country,count(address.application_id)
from address join covid19 on covid19.application_id = address.application_id
group by (report_status,country)
having report_status = 'Positive'
```

No. of tuples = 11

Query Editor Query History

```

1 select country,count(address.application_id)
2 from address join covid19 on covid19.application_id = address.application_id
3 group by (report_status,country)
4 having report_status = 'Positive'

```

Data Output Explain Messages Notifications

	country character varying (50)	count bigint
1	Nepal	4
2	Nationality	1
3	France	5
4	China	7
5	Germany	5
6	Canada	5
7	England	3
8	Russia	5
9	USA	5
10	South Africa	3
11	UK	5

20. Find the number of applicants from each country who have applied for tourist visa and have taken the covid19 vaccine.

```

select country,count(address.application_id)
from address join covid19 on covid19.application_id = address.application_id
join tourist_visa on tourist_visa.application_id = address.application_id
group by (vaccine_status,country)
having vaccine_status = 'Yes'

```

No. of tuples = 9

Query Editor Query History

```

1 select country,count(address.application_id)
2 from address join covid19 on covid19.application_id = address.application_id
3 join tourist_visa on tourist_visa.application_id = address.application_id
4 group by (vaccine_status,country)
5 having vaccine_status = 'Yes'

```

Data Output Explain Messages Notifications

	country character varying (50)	count bigint
1	France	2
2	China	1
3	Russia	3
4	Nepal	1
5	England	2
6	Germany	1
7	Canada	2
8	USA	2
9	UK	1