IMMIGRATION AND VISA MANAGEMENT SYSTEM

Team Members:

Bharath Kumar Tamilselvam
Rupam Patra
Sakshi Mohan Tapkir
Sonal Sunil Takalikar

Professor:

Manuel Montrond

Group 10

OBJECTIVES

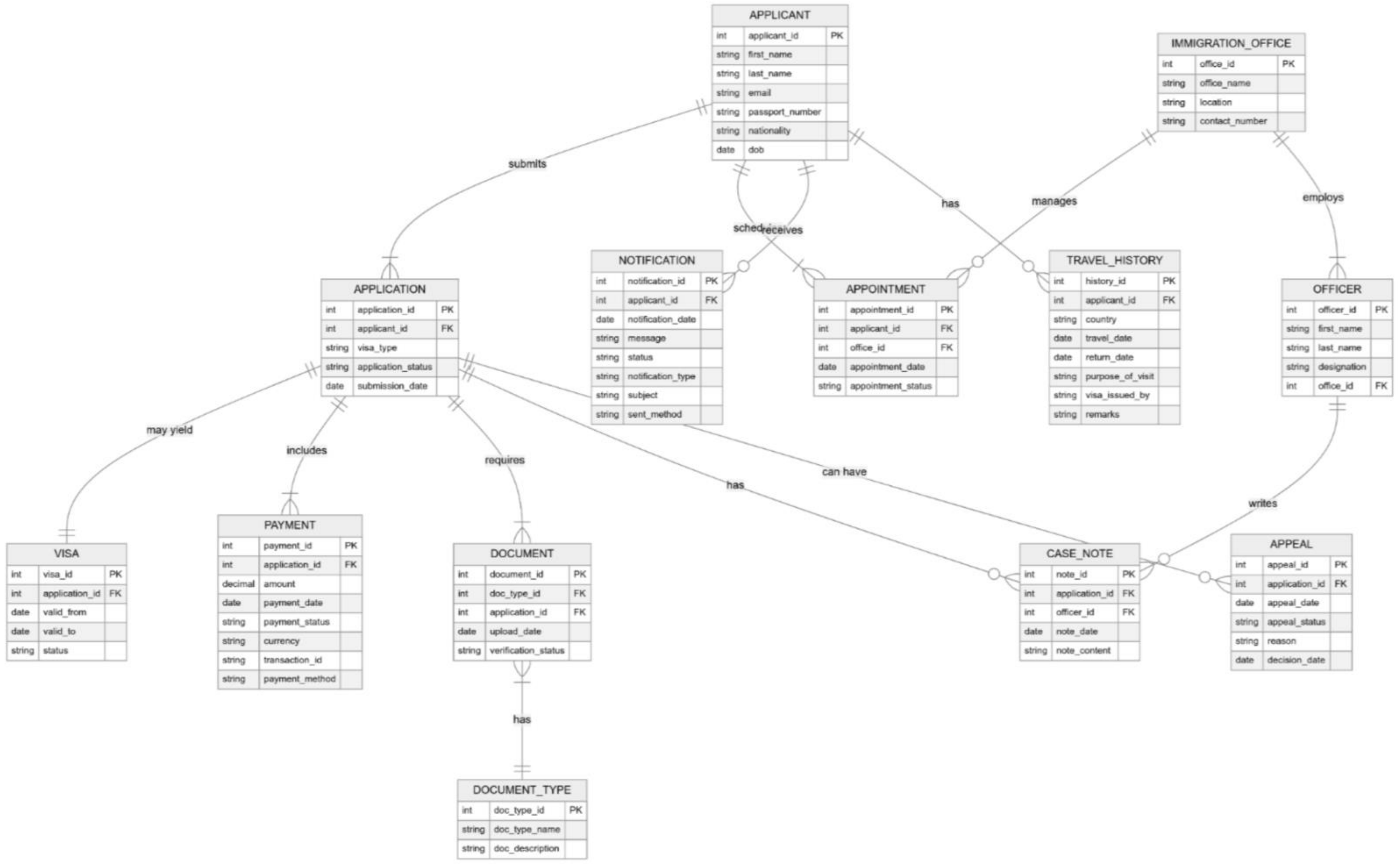
PROBLEMS BEING ADDRESSED

- Lack of a centralized system for managing immigration and visa processes.
- Manual paperwork leads to delays, errors, and lack of transparency.
- Limited access control and data security in traditional systems.
- Difficulty for applicants to track visa status or communicate with officers.

Our Project Goals

- Design a secure, normalized database to manage immigration and visa records.
- Automate key processes like application submission, review, and status updates.
- Provide features like data encryption, triggers, and views for reporting and integrity.

ENTITY RELATIONSHIP DIAGRAM (ERD)



APPLICANT		
int	applicant_id	PK
string	first_name	
string	last_name	
string	email	
string	passport_number	
string	nationality	
date	dob	

IMMIGRATION_OFFICE		
int	office_id	PK
string	office_name	
string	location	
string	contact_number	

APPLICATION		
int	application_id	PK
int	applicant_id	FK
string	visa_type	
string	application_status	
date	submission_date	

NOTIFICATION		
int	notification_id	PK
int	applicant_id	FK
date	notification_date	
string	message	
string	status	
string	notification_type	
string	subject	
string	sent_method	

APPOINTMENT		
int	appointment_id	PK
int	applicant_id	FK
int	office_id	FK
date	appointment_date	
string	appointment_status	

TRAVEL_HISTORY		
int	history_id	PK
int	applicant_id	FK
string	country	
date	travel_date	
date	return_date	
string	purpose_of_visit	
string	visa_issued_by	
string	remarks	

OFFICER		
int	officer_id	PK
string	first_name	
string	last_name	
string	designation	
int	office_id	FK

VISA		
int	visa_id	PK
int	application_id	FK
date	valid_from	
date	valid_to	
string	status	

PAYMENT		
int	payment_id	PK
int	application_id	FK
decimal	amount	
date	payment_date	
string	payment_status	
string	currency	
string	transaction_id	
string	payment_method	

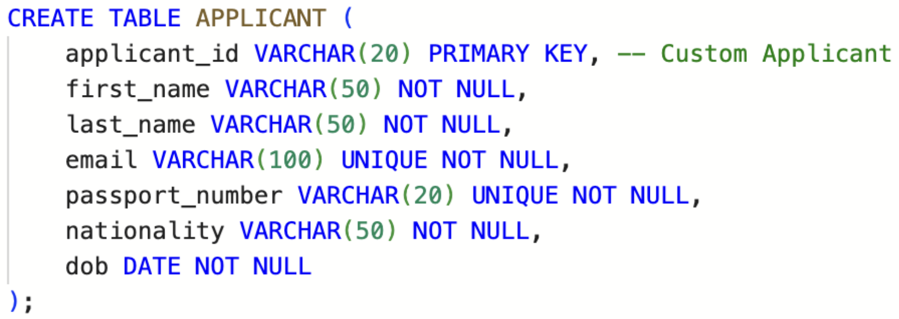
DOCUMENT		
int	document_id	PK
int	doc_type_id	FK
int	application_id	FK
date	upload_date	
string	verification_status	

CASE_NOTE		
int	note_id	PK
int	application_id	FK
int	officer_id	FK
date	note_date	
string	note_content	

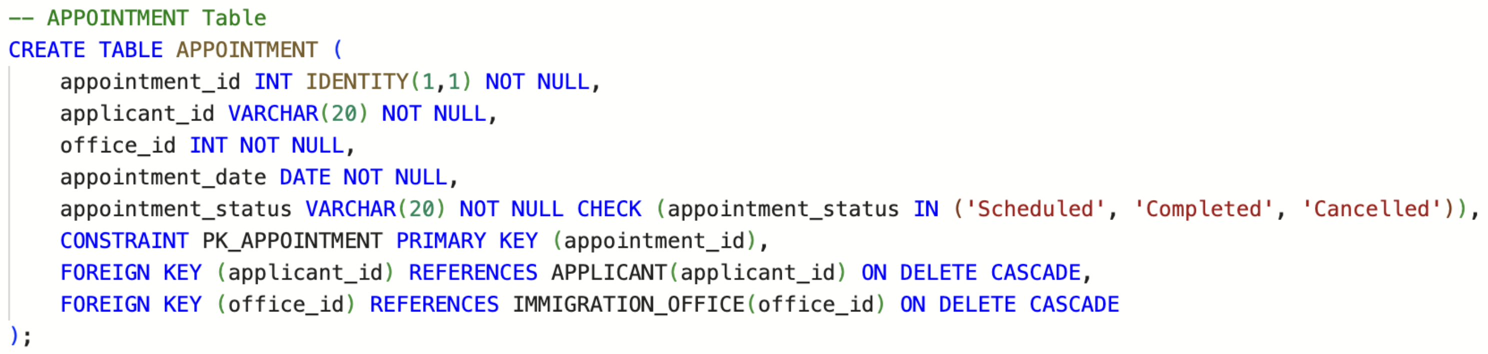
APPEAL		
int	appeal_id	PK
int	application_id	FK
date	appeal_date	
string	appeal_status	
string	reason	
date	decision_date	

DOCUMENT_TYPE		
int	doc_type_id	PK
string	doc_type_name	
string	doc_description	

DATABASE OBJECTS

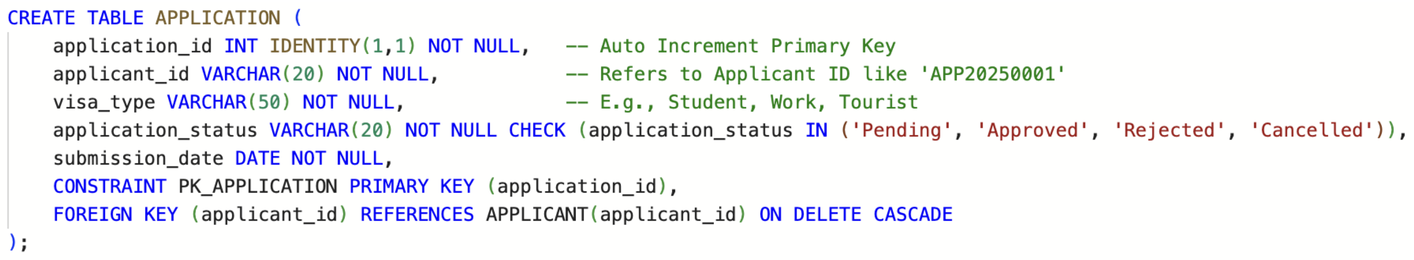


```sql
CREATE TABLE APPLICANT (
    applicant_id VARCHAR(20) PRIMARY KEY, -- Custom Applicant
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    passport_number VARCHAR(20) UNIQUE NOT NULL,
    nationality VARCHAR(50) NOT NULL,
    dob DATE NOT NULL
);
```

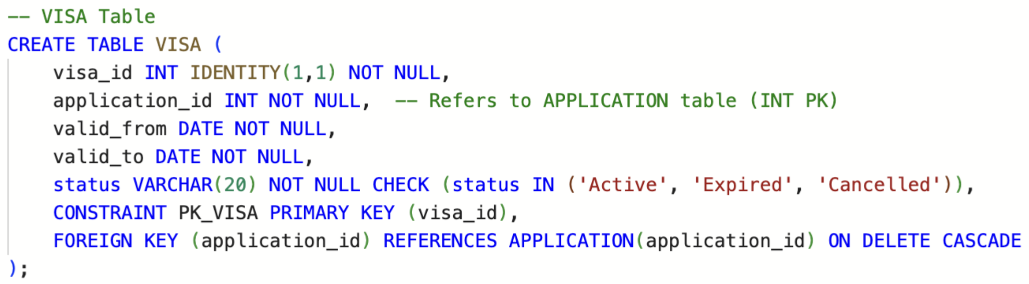


```sql
-- APPOINTMENT Table
CREATE TABLE APPOINTMENT (
    appointment_id INT IDENTITY(1,1) NOT NULL,
    applicant_id VARCHAR(20) NOT NULL,
    office_id INT NOT NULL,
    appointment_date DATE NOT NULL,
    appointment_status VARCHAR(20) NOT NULL CHECK (appointment_status IN ('Scheduled', 'Completed', 'Cancelled')),
    CONSTRAINT PK_APPOINTMENT PRIMARY KEY (appointment_id),
    FOREIGN KEY (applicant_id) REFERENCES APPLICANT(applicant_id) ON DELETE CASCADE,
    FOREIGN KEY (office_id) REFERENCES IMMIGRATION_OFFICE(office_id) ON DELETE CASCADE
);
```

We implemented key database tables including **Applicant**, **Visa**, **Application**, and **Appointments**. The *Applicant* table holds personal details with proper constraints and a unique primary key. The *Visa* table defines visa types and validity. The *Application* table links applicants to visa types using foreign keys and tracks status with value constraints. The *Appointments* table handles scheduling between applicants and officers, ensuring no time conflicts. These DDLs highlight the core structure and relationships in our system, focusing on integrity, normalization, and scalability.

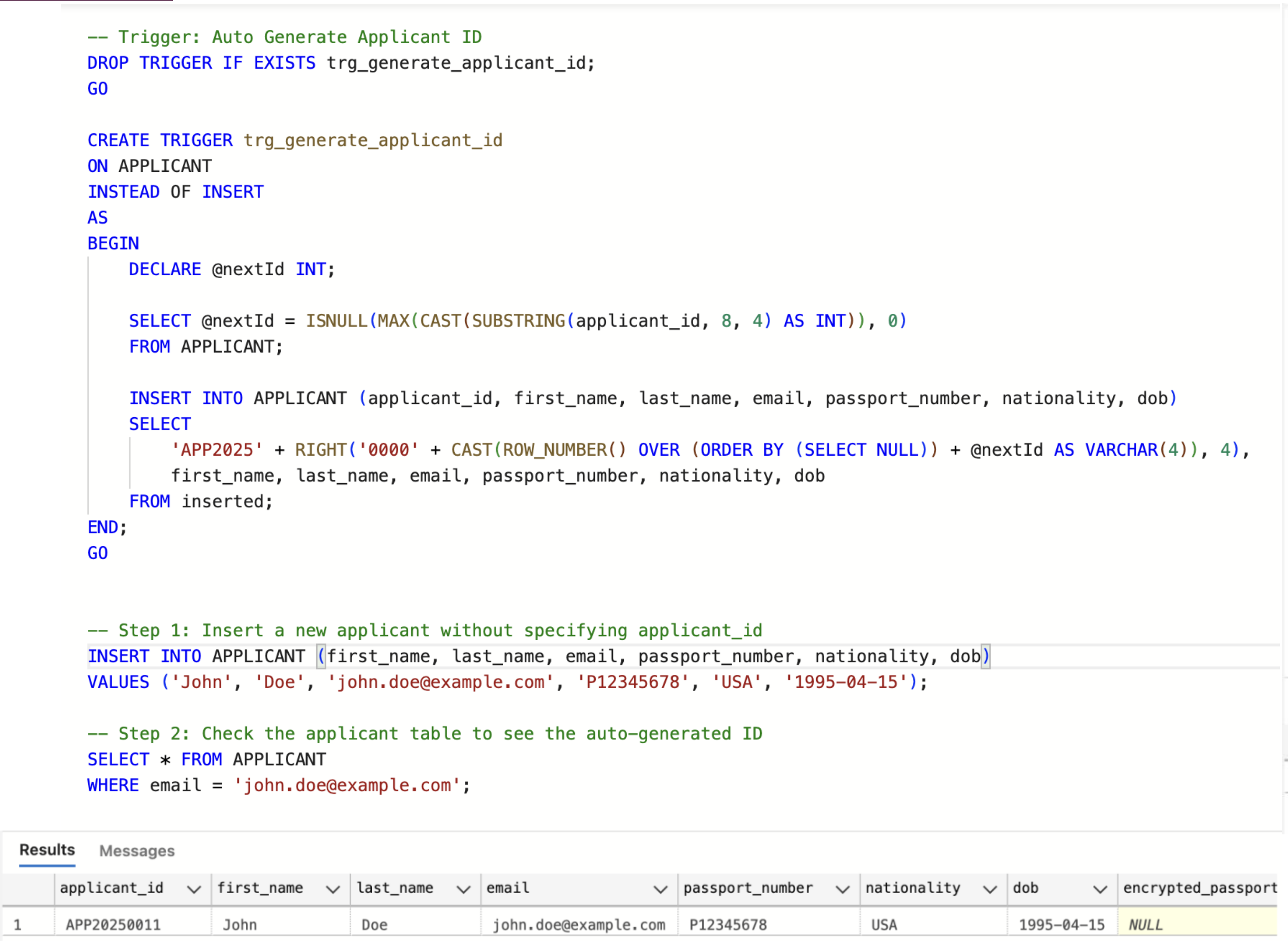


```sql
CREATE TABLE APPLICATION (
    application_id INT IDENTITY(1,1) NOT NULL,   -- Auto Increment Primary Key
    applicant_id VARCHAR(20) NOT NULL,           -- Refers to Applicant ID like 'APP20250001'
    visa_type VARCHAR(50) NOT NULL,              -- E.g., Student, Work, Tourist
    application_status VARCHAR(20) NOT NULL CHECK (application_status IN ('Pending', 'Approved', 'Rejected', 'Cancelled')),
    submission_date DATE NOT NULL,
    CONSTRAINT PK_APPLICATION PRIMARY KEY (application_id),
    FOREIGN KEY (applicant_id) REFERENCES APPLICANT(applicant_id) ON DELETE CASCADE
);
```



```sql
-- VISA Table
CREATE TABLE VISA (
    visa_id INT IDENTITY(1,1) NOT NULL,
    application_id INT NOT NULL,  -- Refers to APPLICATION table (INT PK)
    valid_from DATE NOT NULL,
    valid_to DATE NOT NULL,
    status VARCHAR(20) NOT NULL CHECK (status IN ('Active', 'Expired', 'Cancelled')),
    CONSTRAINT PK_VISA PRIMARY KEY (visa_id),
    FOREIGN KEY (application_id) REFERENCES APPLICATION(application_id) ON DELETE CASCADE
);
```

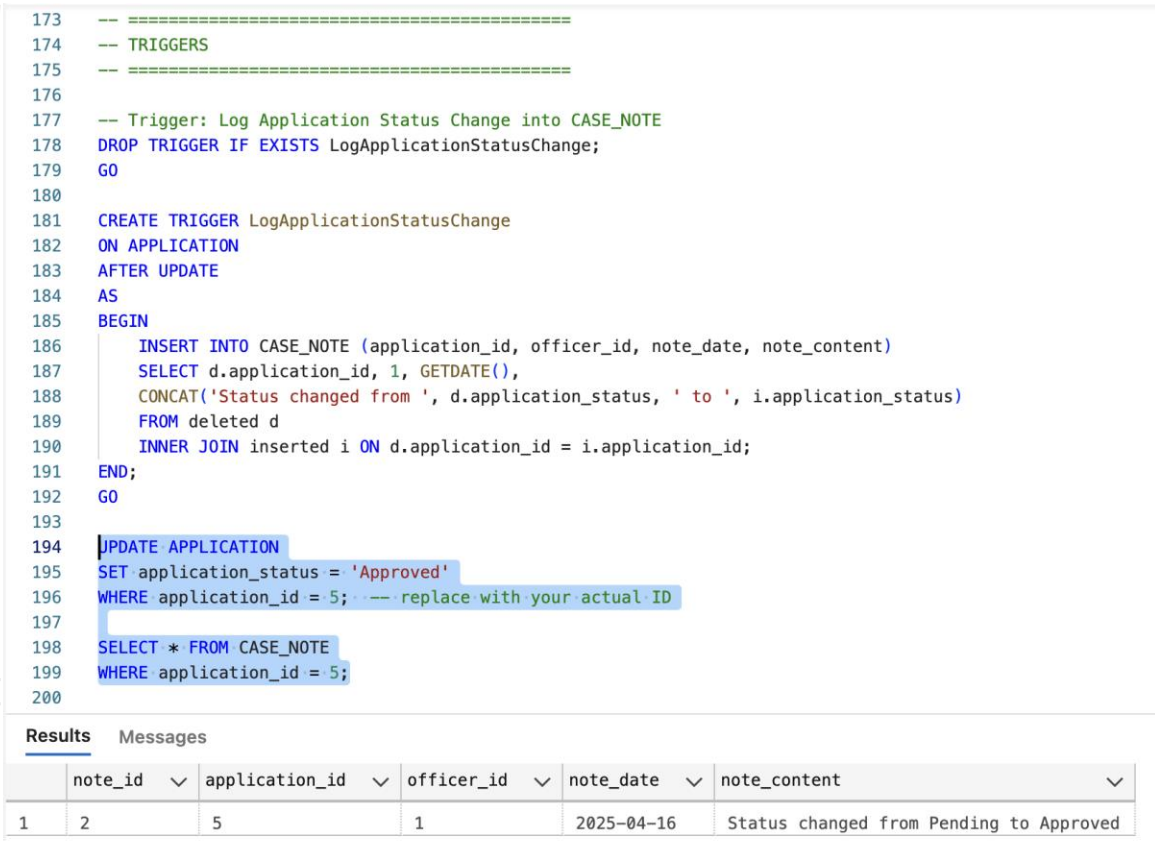
TRIGGERS



```sql
-- Trigger: Auto Generate Applicant ID
DROP TRIGGER IF EXISTS trg_generate_applicant_id;
GO

CREATE TRIGGER trg_generate_applicant_id
ON APPLICANT
INSTEAD OF INSERT
AS
BEGIN
    DECLARE @nextId INT;

    SELECT @nextId = ISNULL(MAX(CAST(SUBSTRING(applicant_id, 8, 4) AS INT)), 0)
    FROM APPLICANT;

    INSERT INTO APPLICANT (applicant_id, first_name, last_name, email, passport_number, nationality, dob)
    SELECT
        'APP2025' + RIGHT('0000' + CAST(ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) + @nextId AS VARCHAR(4)), 4),
        first_name, last_name, email, passport_number, nationality, dob
    FROM inserted;
END;
GO


-- Step 1: Insert a new applicant without specifying applicant_id
INSERT INTO APPLICANT (first_name, last_name, email, passport_number, nationality, dob)
VALUES ('John', 'Doe', 'john.doe@example.com', 'P12345678', 'USA', '1995-04-15');

-- Step 2: Check the applicant table to see the auto-generated ID
SELECT * FROM APPLICANT
WHERE email = 'john.doe@example.com';
```

Results Messages

	applicant_id	first_name	last_name	email	passport_number	nationality	dob	encrypted_passport
1	APP20250011	John	Doe	john.doe@example.com	P12345678	USA	1995-04-15	NULL



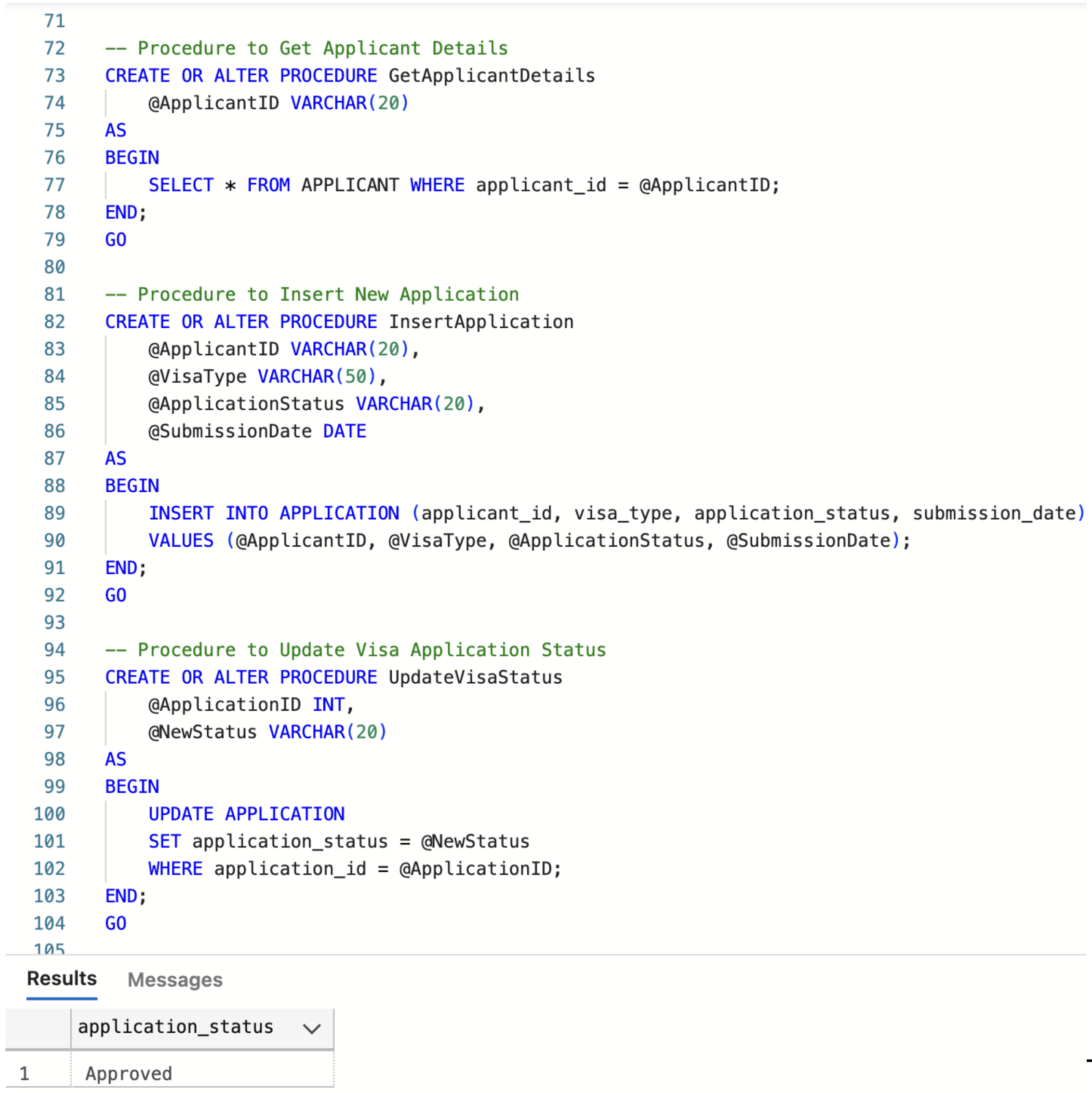
```sql
-- ===========================================
-- TRIGGERS
-- ===========================================

-- Trigger: Log Application Status Change into CASE_NOTE
DROP TRIGGER IF EXISTS LogApplicationStatusChange;
GO

CREATE TRIGGER LogApplicationStatusChange
ON APPLICATION
AFTER UPDATE
AS
BEGIN
    INSERT INTO CASE_NOTE (application_id, officer_id, note_date, note_content)
    SELECT d.application_id, 1, GETDATE(),
    CONCAT('Status changed from ', d.application_status, ' to ', i.application_status)
    FROM deleted d
    INNER JOIN inserted i ON d.application_id = i.application_id;
END;
GO

UPDATE APPLICATION
SET application_status = 'Approved'
WHERE application_id = 5;  -- replace with your actual ID

SELECT * FROM CASE_NOTE
WHERE application_id = 5;
```

Results Messages

	note_id	application_id	officer_id	note_date	note_content
1	2	5	1	2025-04-16	Status changed from Pending to Approved

- **Trigger 1 – `LogApplicationStatusChange`**: Automatically logs a note in the `CASE_NOTE` table whenever an application's status is updated.
- **Trigger 2 – trg_generate_applicant_id**: This trigger auto-generates a unique `applicant_id` whenever a new applicant is inserted, following a fixed format like 'APP2025xxxx'.

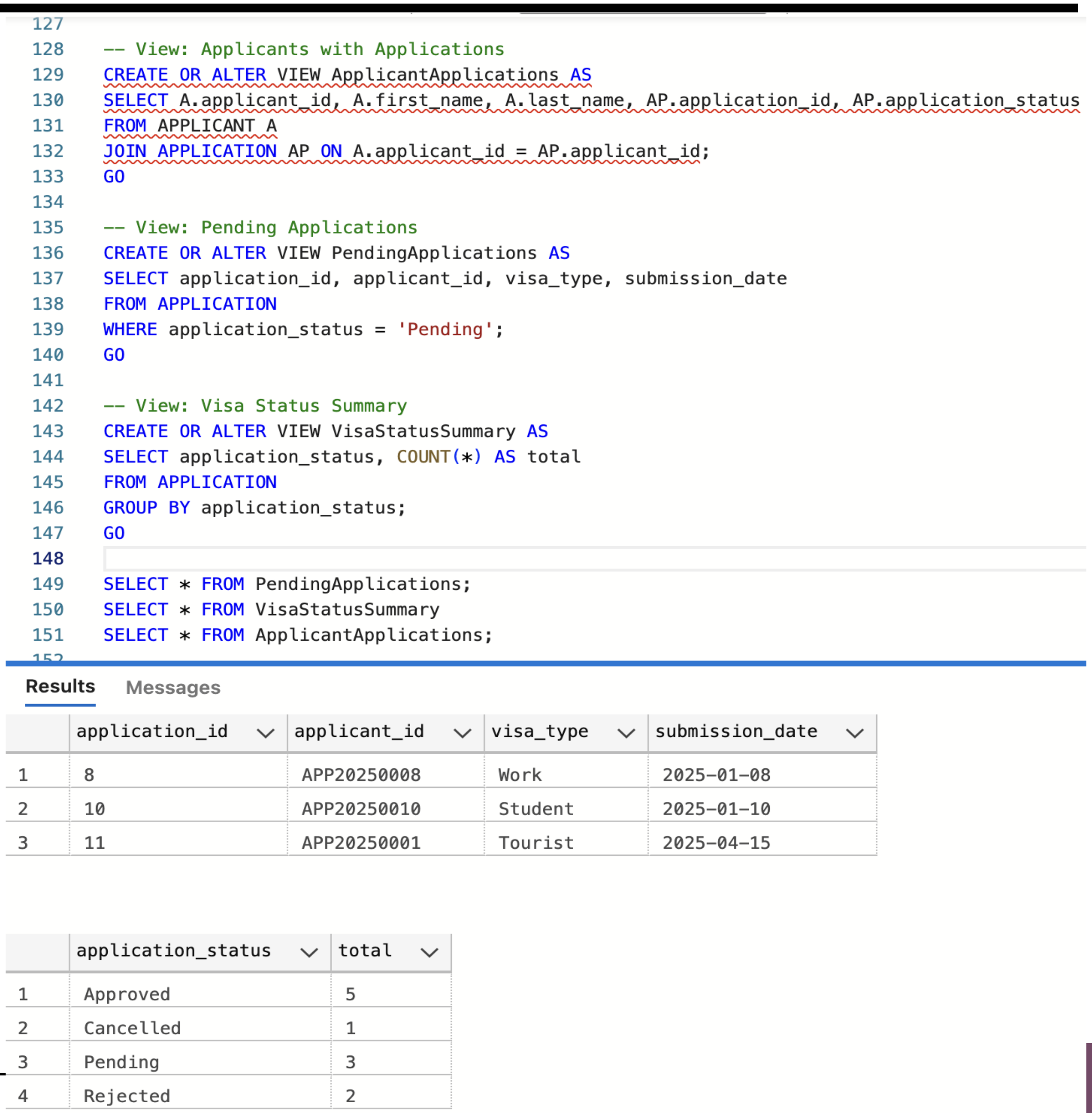
STORED PROCEDURES AND VIEWS



```sql
-- Procedure to Get Applicant Details
CREATE OR ALTER PROCEDURE GetApplicantDetails
    @ApplicantID VARCHAR(20)
AS
BEGIN
    SELECT * FROM APPLICANT WHERE applicant_id = @ApplicantID;
END;
GO

-- Procedure to Insert New Application
CREATE OR ALTER PROCEDURE InsertApplication
    @ApplicantID VARCHAR(20),
    @VisaType VARCHAR(50),
    @ApplicationStatus VARCHAR(20),
    @SubmissionDate DATE
AS
BEGIN
    INSERT INTO APPLICATION (applicant_id, visa_type, application_status, submission_date)
    VALUES (@ApplicantID, @VisaType, @ApplicationStatus, @SubmissionDate);
END;
GO

-- Procedure to Update Visa Application Status
CREATE OR ALTER PROCEDURE UpdateVisaStatus
    @ApplicationID INT,
    @NewStatus VARCHAR(20)
AS
BEGIN
    UPDATE APPLICATION
    SET application_status = @NewStatus
    WHERE application_id = @ApplicationID;
END;
GO
```

Results Messages

	application_status
1	Approved



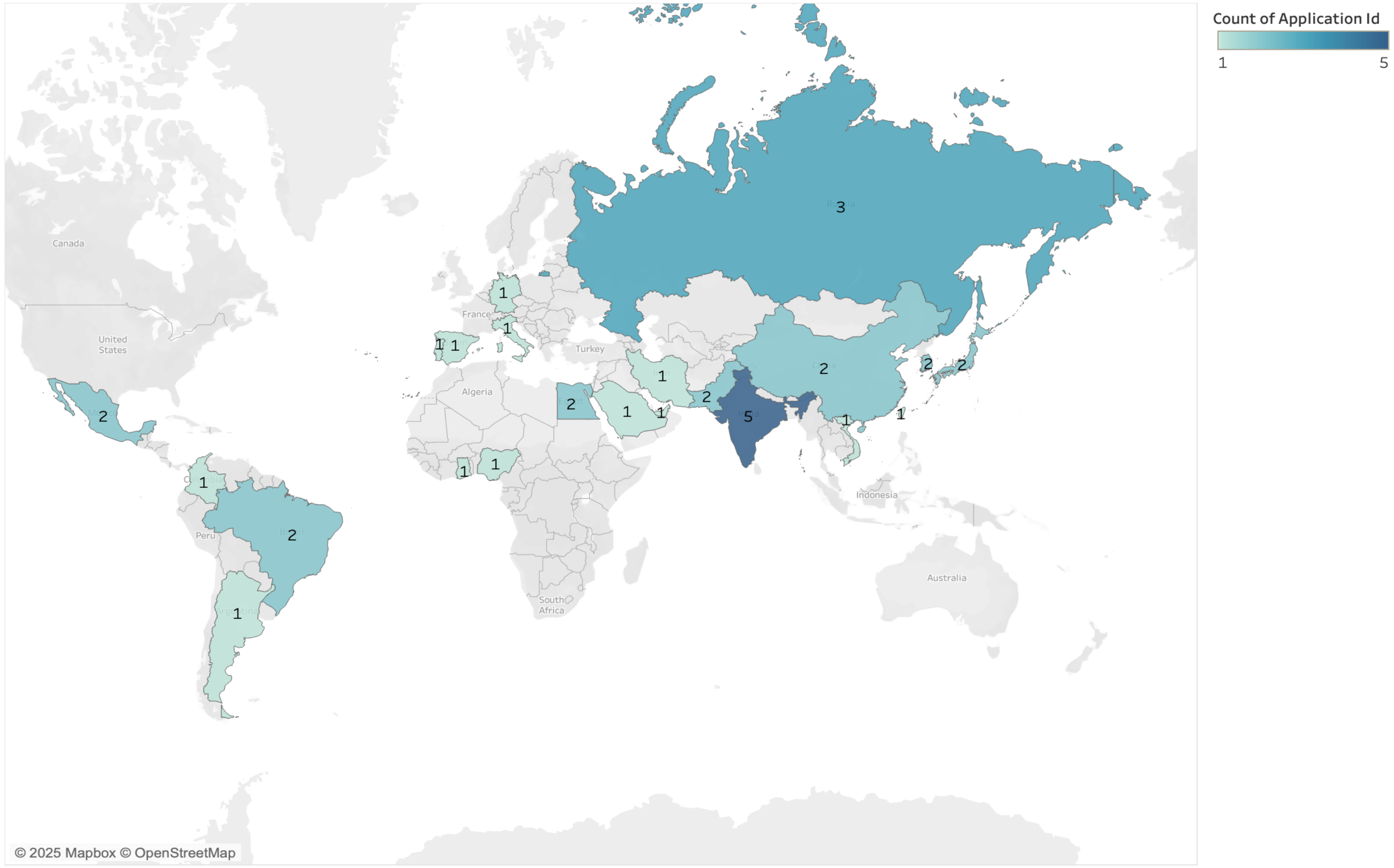
```sql
-- View: Applicants with Applications
CREATE OR ALTER VIEW ApplicantApplications AS
SELECT A.applicant_id, A.first_name, A.last_name, AP.application_id, AP.application_status
FROM APPLICANT A
JOIN APPLICATION AP ON A.applicant_id = AP.applicant_id;
GO

-- View: Pending Applications
CREATE OR ALTER VIEW PendingApplications AS
SELECT application_id, applicant_id, visa_type, submission_date
FROM APPLICATION
WHERE application_status = 'Pending';
GO

-- View: Visa Status Summary
CREATE OR ALTER VIEW VisaStatusSummary AS
SELECT application_status, COUNT(*) AS total
FROM APPLICATION
GROUP BY application_status;
GO

SELECT * FROM PendingApplications;
SELECT * FROM VisaStatusSummary
SELECT * FROM ApplicantApplications;
```

Results Messages

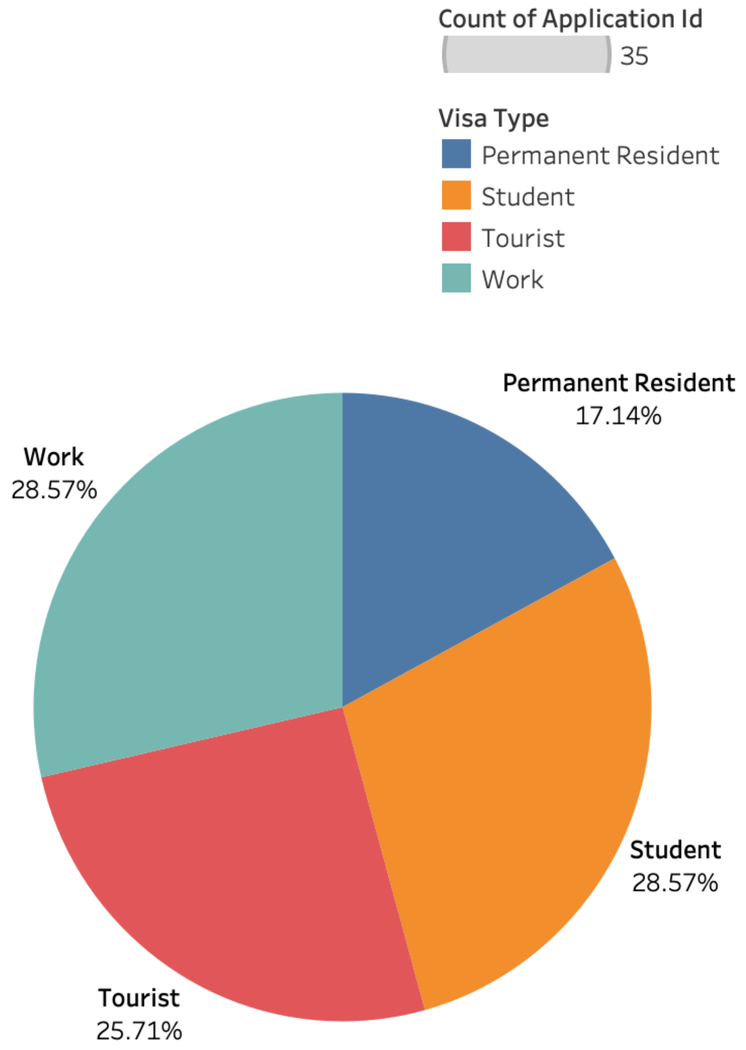
	application_id	applicant_id	visa_type	submission_date
1	8	APP20250008	Work	2025-01-08
2	10	APP20250010	Student	2025-01-10
3	11	APP20250001	Tourist	2025-04-15

	application_status	total
1	Approved	5
2	Cancelled	1
3	Pending	3
4	Rejected	2

DATA VISUALIZATIONS - TABLEAU

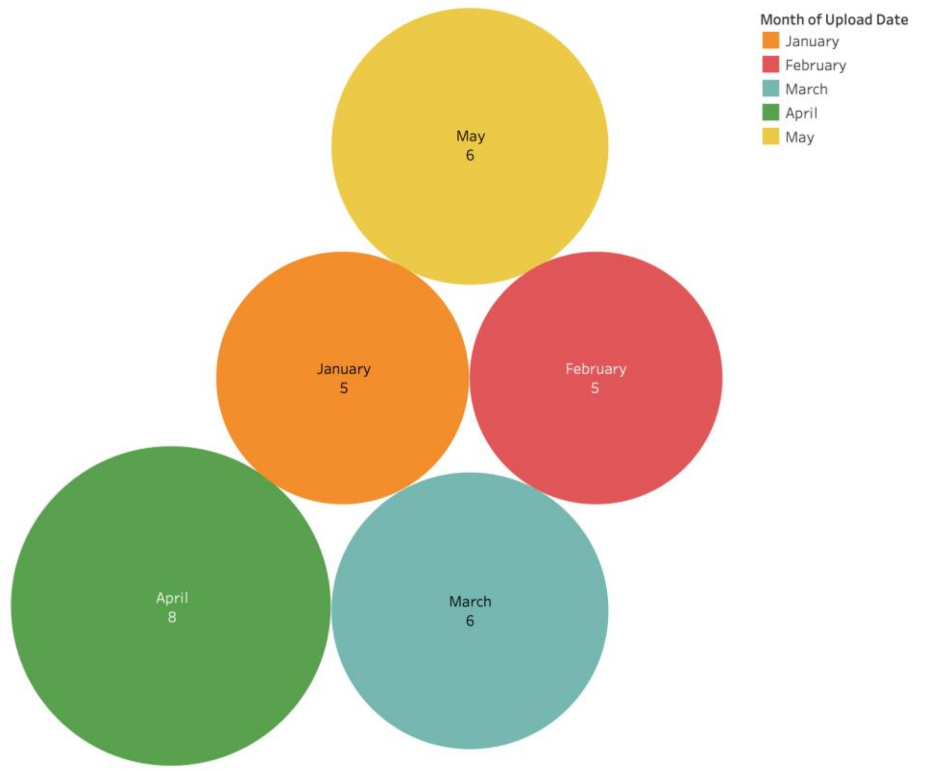


Map based on Longitude (generated) and Latitude (generated). Color shows count of Application Id. Details are shown for Nationality.



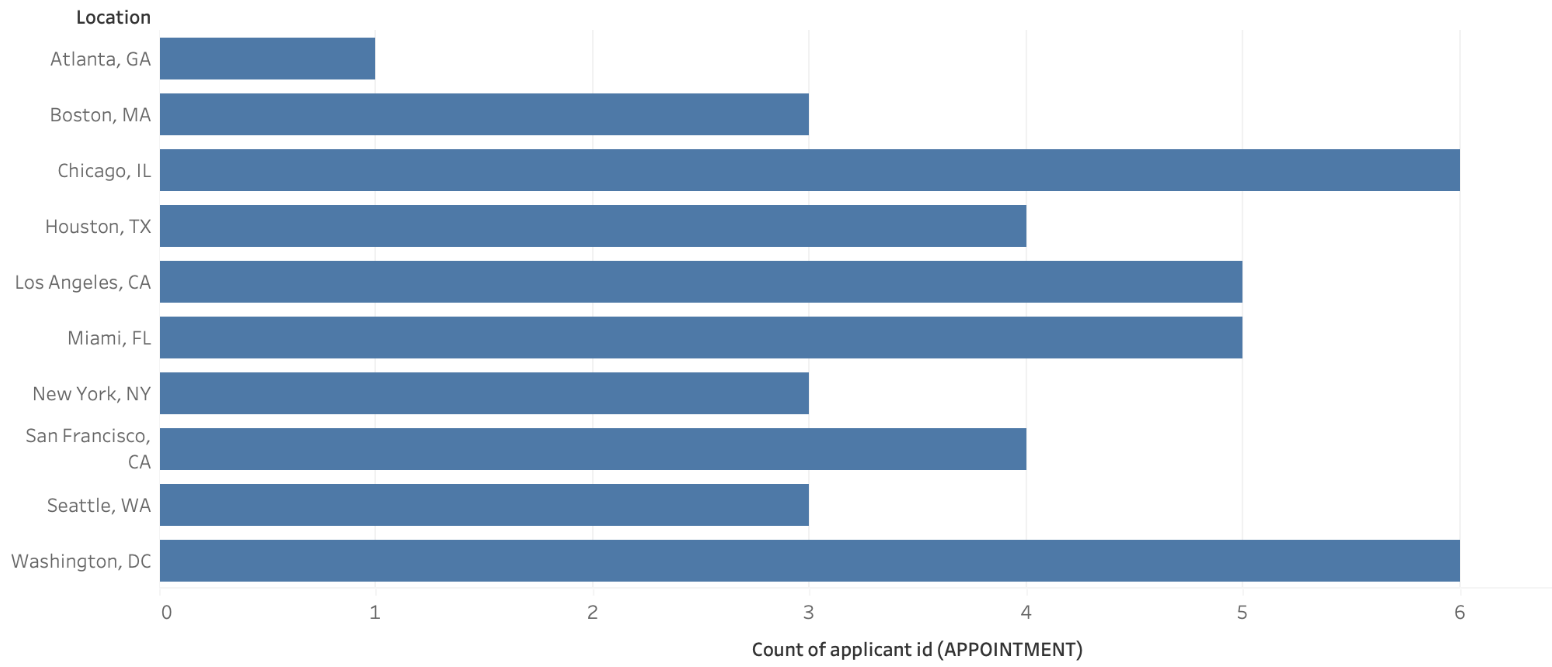
DATA VISUALIZATIONS - TABLEAU

Documents



Upload Date Month and distinct count of Doc Type Name. Color shows details about Upload Date Month. Size shows count of Doc Type Name. The marks are labeled by Upload Date Month and distinct count of Doc Type Name. The view is filtered on Upload Date Month, which excludes Null.

Sheet 4



Count of applicant id (APPOINTMENT) for each Location.

GUI DEMO FOR OUR PROJECT

THANK YOU