DAMG6210 - Data Management And Database Design

RECRUITMENT MANAGEMENT SYSTEM

Under the guidance of Manuel D. Montrond

Group No.: 12
Group Members: Gopal Anil Pillai | Vandana Rangaswamy | Nisarg Sheth | Sanjana Srikanth Tikotikar | Vimala Suram

PURPOSE & OBJECTIVES

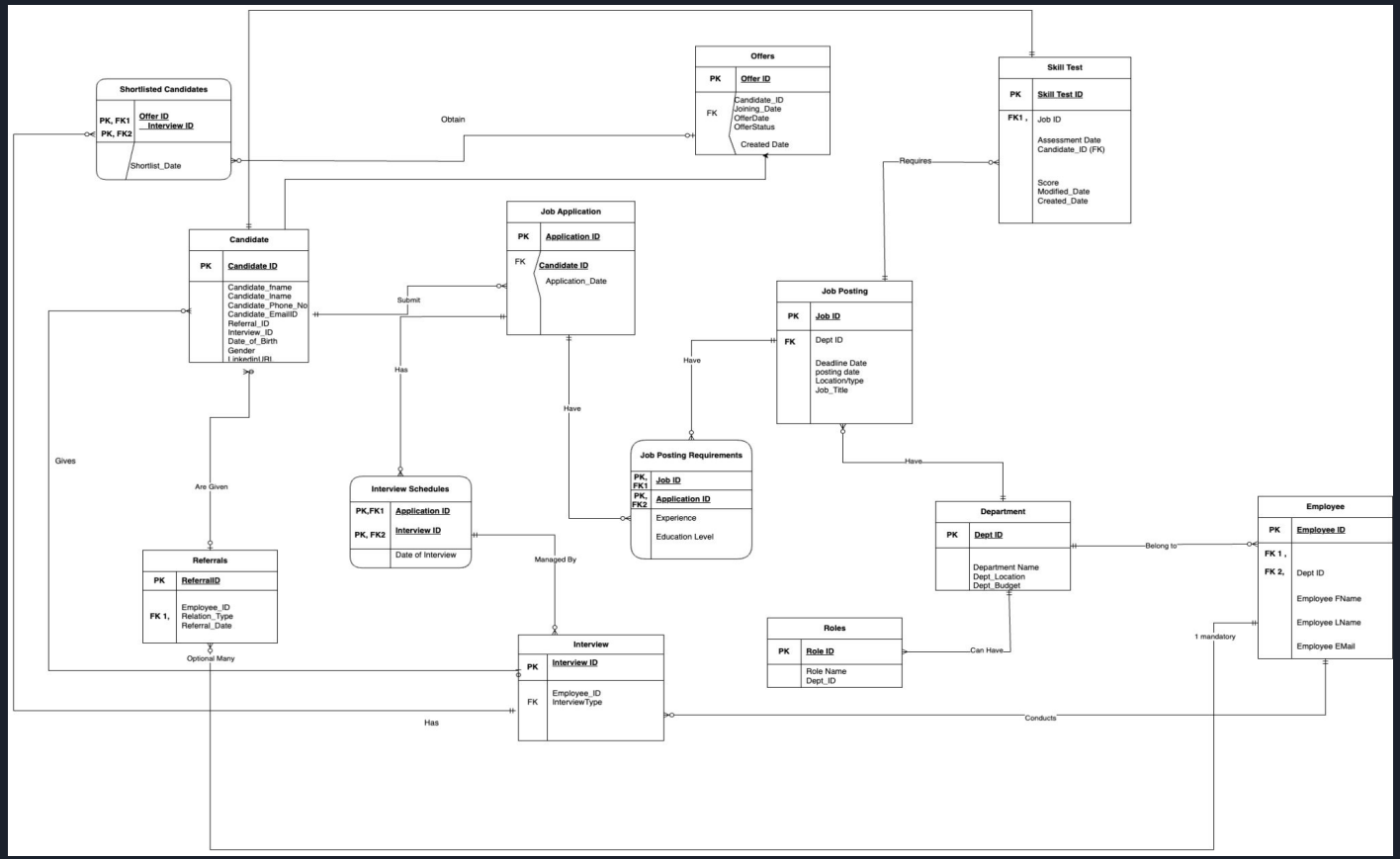
PURPOSE:

The objective is to create an all-inclusive Recruitment Management System that optimizes and enriches the recruiting procedure for organizations, guaranteeing effective candidate handling, enhanced decision-making, and a smooth experience for both hiring managers and candidates.

MISSION OBJECTIVES:

- **Streamlined Recruitment**: Automate the entire hiring process, from job posting to onboarding, reducing administrative costs and manual effort.
- **Optimizing internal hiring process:** Making the the hiring process in the company more easier and convenient.
- **Efficient Client Management**: Enable advanced candidate search and filtering based on criteria like experience and education. Maintain organized records of applicants, track progress through recruitment stages, and assist in shortlist creation.
- **Data-Driven Insights**: Provide analytics and dashboards to monitor metrics like hiring source efficacy, cost per hire, and time to hire. Leverage insights to optimize recruitment strategies.

E-R DIAGRAM


Shortlisted Candidates
PK, FK1 Offer ID
PK, FK2 Interview ID
Shortlist_Date
Obtain
Offers
PK Offer ID
FK Candidate_ID
Joining_Date
OfferDate
OfferStatus
Created Date
Skill Test
PK Skill Test ID
FK1 , Job ID
Assessment Date
Candidate_ID (FK)
Score
Modified_Date
Created_Date
Requires
Candidate
PK Candidate ID
Candidate_fname
Candidate_lname
Candidate_Phone_No
Candidate_EmailID
Referral_ID
Interview_ID
Date_of_Birth
Gender
LinkedinURL
Submit
Job Application
PK Application ID
FK Candidate ID
Application_Date
Job Posting
PK Job ID
FK Dept ID
Deadline Date
posting date
Location/type
Job_Title
Have
Has
Have
Gives
Are Given
Interview Schedules
PK,FK1 Application ID
PK, FK2 Interview ID
Date of Interview
Job Posting Requirements
PK, FK1 Job ID
PK, FK2 Application ID
Experience
Education Level
Have
Department
PK Dept ID
Department Name
Dept_Location
Dept_Budget
Belong to
Employee
PK Employee ID
FK 1 ,
FK 2, Dept ID
Employee FName
Employee LName
Employee EMail
1 mandatory
Referrals
PK ReferralID
FK 1, Employee_ID
Relation_Type
Referral_Date
Optional Many
Managed By
Interview
PK Interview ID
FK Employee_ID
InterviewType
Roles
PK Role ID
Role Name
Dept_ID
Can Have
Has
Conducts

STORED PROCEDURES

```sql
-- 2.  Stored Procedure to Insert a New Candidate and Return Their ID
CREATE PROCEDURE AddNewCandidate
    @CandidateFName NVARCHAR(100),
    @CandidateLName NVARCHAR(100),
    @CandidatePhoneNo NVARCHAR(20),
    @CandidateEmail NVARCHAR(100),
    @ReferralID INT,
    @InterviewID INT,
    @DOB DATE,
    @Gender CHAR(1),
    @NewCandidateID INT OUTPUT
AS
BEGIN
    -- Insert a new candidate
    INSERT INTO Candidate (Candidate_FName, Candidate_LName, Candidate_Phone_No, Candidate_EmailID, Referral_ID, Interview_ID, Date_of_Birth, Gender)
    VALUES (@CandidateFName, @CandidateLName, @CandidatePhoneNo, @CandidateEmail, @ReferralID, @InterviewID, @DOB, @Gender);

    -- Get the ID of the newly inserted candidate
    SET @NewCandidateID = SCOPE_IDENTITY();
END;
GO

DECLARE @NewCandidateID INT;
EXEC AddNewCandidate 'John', 'Doe', '8573975545', 'john.doe@example.com', 1, 2, '1990-05-15', 'M', @NewCandidateID OUTPUT;
SELECT @NewCandidateID AS 'New Candidate ID';
```

Results | Messages

	New Candidate…
1	11

AddNewCandidate inserts a new candidate's details into the Candidate table and outputs the newly generated Candidate_ID using SCOPE_IDENTITY(). When executed with candidate details, it adds the record to the database and returns the ID of the newly created candidate through the @NewCandidateID variable.

```sql
CREATE PROCEDURE UpdateCandidateContactInfo
    @CandidateID INT,
    @NewPhone VARCHAR(20),
    @NewEmail VARCHAR(100)
AS
BEGIN
    -- Update the phone and email for the specified candidate
    UPDATE Candidate
    SET Candidate_Phone_No = @NewPhone,
        Candidate_EmailID = @NewEmail
    WHERE Candidate_ID = @CandidateID;
END;

EXEC UpdateCandidateContactInfo
    @CandidateID = 1,
    @NewPhone = '9999999999999',
    @NewEmail = 'johnson.doe@example.com';

select * from Candidate
```

Results | Messages

	Candidate_ID	Candidate_FName	Candidate_LName	Candidate_Phone_No	Candidate_EmailID	Referral_ID	Interview_ID	Date_of_Birth	Gender
1	1	Tom	Baker	1234567890	tombaker@example.com	1	NULL	1990-05-15	M
2	2	Anna	Clark	2345678901	annaclark@example.com	2	NULL	1992-07-22	F
3	3	Peter	Doyle	3456789012	peterdoyle@example.com	3	NULL	1988-10-05	M
4	4	Sarah	Evans	4567890123	sarahevans@example.com	4	NULL	1995-11-12	F
5	5	Mike	Foster	5678901234	mikefoster@example.com	5	NULL	1987-09-21	M
6	6	Lucy	Green	6789012345	lucygreen@example.com	6	NULL	1993-03-08	F
7	7	George	Hill	7890123456	georgehill@example.com	7	NULL	1991-04-25	M
8	8	Emily	Ivy	8901234567	emilyivy@example.com	8	NULL	1990-08-30	F
9	9	James	Johnson	9012345678	jamesjohnson@example.…	9	NULL	1989-02-18	M
10	10	Olivia	King	0123456789	oliviaking@example.com	10	NULL	1994-06-14	F
11	11	John	Doe	8573975545	john.doe@example.com	1	2	1990-05-15	M

UpdateCandidateContactInfo updates the phone number and email address for a specified candidate in the Candidate table. It takes CandidateID, NewPhone, and NewEmail as input parameters, and modifies the Candidate_Phone_No and Candidate_EmailID fields for the candidate with the given CandidateID.

VIEWS

Users > sanjanatikotikar > Downloads > psm_script (3).sql

Results grid | Cancel | Disconnect | Change | Database: DAMG

```sql
CREATE VIEW Department_Job_Openings AS
SELECT
    j.Job_Title,
    d.Department_Name
FROM
    Job_Posting j
JOIN
    Department d ON j.DeptID = d.DeptID;
SELECT * from Department_Job_Openings
```

Results | Messages

	Job_Title	Department_Name
1	Senior Software Engineer	Engineering
2	Marketing Manager	Marketing
3	Sales Executive	Sales
4	Financial Consultant	Finance
5	Operations Analyst	Operations
6	IT Project Manager	IT
7	Customer Support Specialist	Customer Service
8	Paralegal	Legal
9	Office Administrator	Admin
1…	HR Specialist	HR

The Department_job_Openings view provides a detailed report of the job openings in each departments. The view includes Job_Title and Department_name.

Run | Cancel | Disconnect | Change | Database: DAMG | Estimated Plan | Enable Actual Plan | Parse | Enable SQLCMD | To Notebook

```sql
CREATE VIEW Candidate_Skills_Report AS
SELECT
    C.Candidate_FName,
    C.Candidate_LName,
    C.Candidate_EmailID,
    JP.Job_Title,
    ST.Score,
    JP.Location,
    JP.Deadline_Date,
    JR.Experience,
    JR.Education_Level
FROM
    Candidate C
JOIN
    Job_Application JA ON C.Candidate_ID = JA.Candidate_ID
JOIN
    Job_Posting JP ON JA.Application_ID = JP.DeptID
JOIN
    Skill_Test ST ON C.Candidate_ID = ST.Candidate_ID
JOIN
    Job_Posting_Requirement JR ON JP.Job_ID = JR.Job_ID;

SELECT * from Candidate_Skills_Report
```

Results | Messages

	Candidate_FName	Candidate_LName	Candidate_EmailID	Job_Title	Score	Location	Deadline_Date	Experience	Education_Level
1	Tom	Baker	tombaker@example.com	HR Specialist	85.50	New York	2024-06-01	0	Diploma
2	Anna	Clark	annaclark@example.com	Senior Software Engineer	78.00	San Francisco	2024-06-01	5	Bachelor
3	Peter	Doyle	peterdoyle@example.com	Marketing Manager	92.30	Chicago	2024-06-01	3	Master
4	Sarah	Evans	sarahevans@example.com	Sales Executive	88.90	Los Angeles	2024-06-01	2	Bachelor
5	Mike	Foster	mikefoster@example.com	Financial Consultant	79.50	Boston	2024-06-01	4	Doctorate
6	Lucy	Green	lucygreen@example.com	Operations Analyst	82.70	Dallas	2024-06-01	1	Bachelor
7	George	Hill	georgehill@example.com	IT Project Manager	91.20	Seattle	2024-06-01	5	Master
8	Emily	Ivy	emilyivy@example.com	Customer Support Specialist	76.30	Miami	2024-06-01	2	Associate
9	James	Johnson	jamesjohnson@example.com	Paralegal	84.40	Washington DC	2024-06-01	1	High School
10	Olivia	King	oliviaking@example.com	Office Administrator	89.10	Denver	2024-06-01	3	Bachelor

The Candidate_Skills_Report view provides a detailed report of candidates' skills and performance by combining information from multiple tables (Candidate, Job_Application, Job_Posting, Skill_Test, and Job_Posting_Requirement). It includes details like the candidate's name, email, job title, skill test score, job location, application deadline, required experience, and education level, offering a comprehensive insight into candidates' qualifications and job suitability.

ENCRYPTION

```
    DECLARE @decrypted_data VARCHAR(50)

    SELECT @decrypted_data =
        CONVERT(VARCHAR(50),
            DecryptByKey(@encrypted_data))

    RETURN @decrypted_data
END;

-- Step 7: Decrypt and retrieve employee data
-- Open the symmetric key again, as it needs to be open to decrypt the data
OPEN SYMMETRIC KEY EmpDataKey
DECRYPTION BY CERTIFICATE EmpCert;
GO

SELECT
    Employee_ID,
    dbo.DecryptEmployeeData1(EncryptedEmpFirstName) AS Decrypted_FirstName,
    dbo.DecryptEmployeeData1(EncryptedEmpLastName) AS Decrypted_LastName
FROM employee;

-- Step 8: Close the symmetric key after use
CLOSE SYMMETRIC KEY EmpDataKey;

select * from employee;
```

Results | Messages

Employee_FName	Employee_LName	Employee_Email	EncryptedEmpFirstName	EncryptedEmpLastName
John	Doe	johndoe@example.com	0x0052293DB28DBB4FBEC95816773200FA0200000060C57805D4…	0x0052293DB28DBB4FBEC95816773200FA0200000B751C710F2…
Alice	Smith	alicesmith@example.com	0x0052293DB28DBB4FBEC95816773200FA02000001ABA63A4C4…	0x0052293DB28DBB4FBEC95816773200FA0200000187A592554…
Bob	Brown	bobbrown@example.com	0x0052293DB28DBB4FBEC95816773200FA020000000954 0B6F2…	0x0052293DB28DBB4FBEC95816773200FA02000005C64BB732E…
Charlie	Davis	charliedavis@example.com	0x0052293DB28DBB4FBEC95816773200FA020000059F7C10C3A…	0x0052293DB28DBB4FBEC95816773200FA0200000AB5D6AA9A4…
Diana	Evans	dianaevans@example.com	0x0052293DB28DBB4FBEC95816773200FA02000005C6FA11180…	0x0052293DB28DBB4FBEC95816773200FA0200000CAD68E95BA…
Frank	Garcia	frankgarcia@example.com	0x0052293DB28DBB4FBEC95816773200FA020000075F0084D85…	0x0052293DB28DBB4FBEC95816773200FA0200000777CFE8BF9…
Grace	Harris	graceharris@example.com	0x0052293DB28DBB4FBEC95816773200FA0200000DC7E6C068D…	0x0052293DB28DBB4FBEC95816773200FA0200000076E88FCA0…
Hank	Ivy	hankivy@example.com	0x0052293DB28DBB4FBEC95816773200FA020000056336AA818…	0x0052293DB28DBB4FBEC95816773200FA0200000F921E3BE38…
Ivy	Jones	ivyjones@example.com	0x0052293DB28DBB4FBEC95816773200FA02000000DFC8DD2327…	0x0052293DB28DBB4FBEC95816773200FA0200000FDEFF4D566…
Jack	King	jackking@example.com	0x0052293DB28DBB4FBEC95816773200FA020000007885F6B410…	0x0052293DB28DBB4FBEC95816773200FA020000095CBD08754…

This script encrypts sensitive employee data (First Name and Last Name) using AES-256 with a symmetric key (EmpDataKey), secured by a certificate (EmpCert) and a master key. It also includes a decryption function (DecryptEmployeeData1) to retrieve the data once the symmetric key is unlocked.

USER DEFINED FUNCTIONS

Run | Cancel | Disconnect | Change | Database: DAMG | Estimate

```
-- USER DEFINED FUNCTIONS
-- 1.   Get Candidate's Total Applications Count
CREATE FUNCTION dbo.fn_GetTotalApplications (@Candidate_ID INT)
RETURNS INT
AS
BEGIN
    DECLARE @TotalApplications INT;

    SELECT @TotalApplications = COUNT(*)
    FROM Job_Application
    WHERE Candidate_ID = @Candidate_ID;

    RETURN @TotalApplications;
END;

SELECT dbo.fn_GetTotalApplications(1) AS TotalApplications;
```

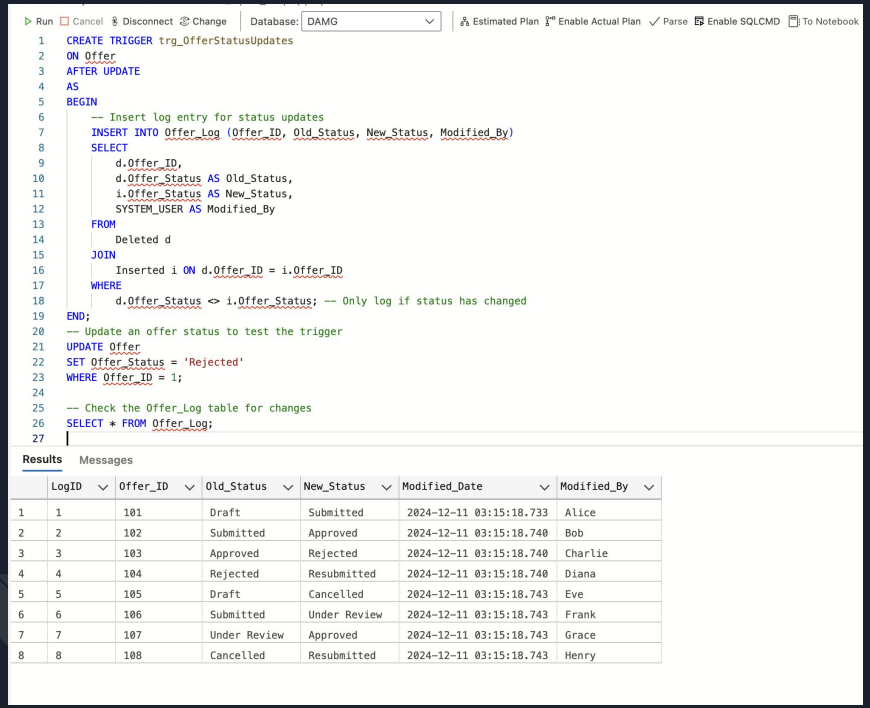
Results | Messages

	TotalAppli…
1	1

This script defines a function, fn_GetTotalApplications, which takes a candidate's ID as input and returns the total number of job applications submitted by that candidate, by querying the Job_Application table. It can be called using a SELECT statement with the candidate's ID.

TRIGGERS

This trigger trg_OfferStatusUpdates automatically logs changes to the Offer_Status column in the Offer_Log table whenever the status of an offer is updated in the Offer table. It captures the old and new status, the offer ID, and the user making the change, ensuring only actual status changes are recorded.



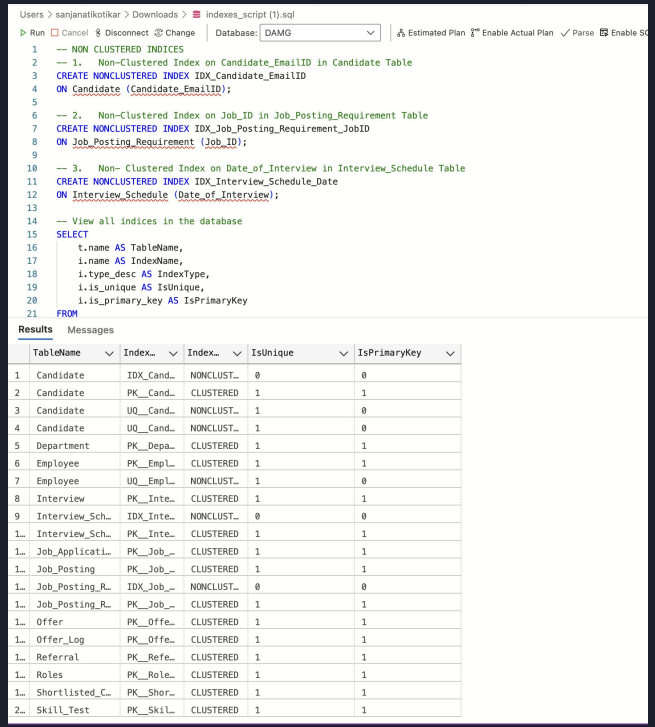
```sql
CREATE TRIGGER trg_OfferStatusUpdates
ON Offer
AFTER UPDATE
AS
BEGIN
    -- Insert log entry for status updates
    INSERT INTO Offer_Log (Offer_ID, Old_Status, New_Status, Modified_By)
    SELECT
        d.Offer_ID,
        d.Offer_Status AS Old_Status,
        i.Offer_Status AS New_Status,
        SYSTEM_USER AS Modified_By
    FROM
        Deleted d
    JOIN
        Inserted i ON d.Offer_ID = i.Offer_ID
    WHERE
        d.Offer_Status <> i.Offer_Status; -- Only log if status has changed
END;
-- Update an offer status to test the trigger
UPDATE Offer
SET Offer_Status = 'Rejected'
WHERE Offer_ID = 1;

-- Check the Offer_Log table for changes
SELECT * FROM Offer_Log;
```

Results | Messages

	LogID	Offer_ID	Old_Status	New_Status	Modified_Date	Modified_By
1	1	101	Draft	Submitted	2024-12-11 03:15:18.733	Alice
2	2	102	Submitted	Approved	2024-12-11 03:15:18.740	Bob
3	3	103	Approved	Rejected	2024-12-11 03:15:18.740	Charlie
4	4	104	Rejected	Resubmitted	2024-12-11 03:15:18.740	Diana
5	5	105	Draft	Cancelled	2024-12-11 03:15:18.743	Eve
6	6	106	Submitted	Under Review	2024-12-11 03:15:18.743	Frank
7	7	107	Under Review	Approved	2024-12-11 03:15:18.743	Grace
8	8	108	Cancelled	Resubmitted	2024-12-11 03:15:18.743	Henry

INDEXES

The SQL script creates three non-clustered indexes on the Candidate_EmailID, Job_ID, and Date_of_Interview columns to enhance query performance. It also includes a query to list all indexes in the database, showing details like table name, index name, type, and whether it's unique or a primary key.

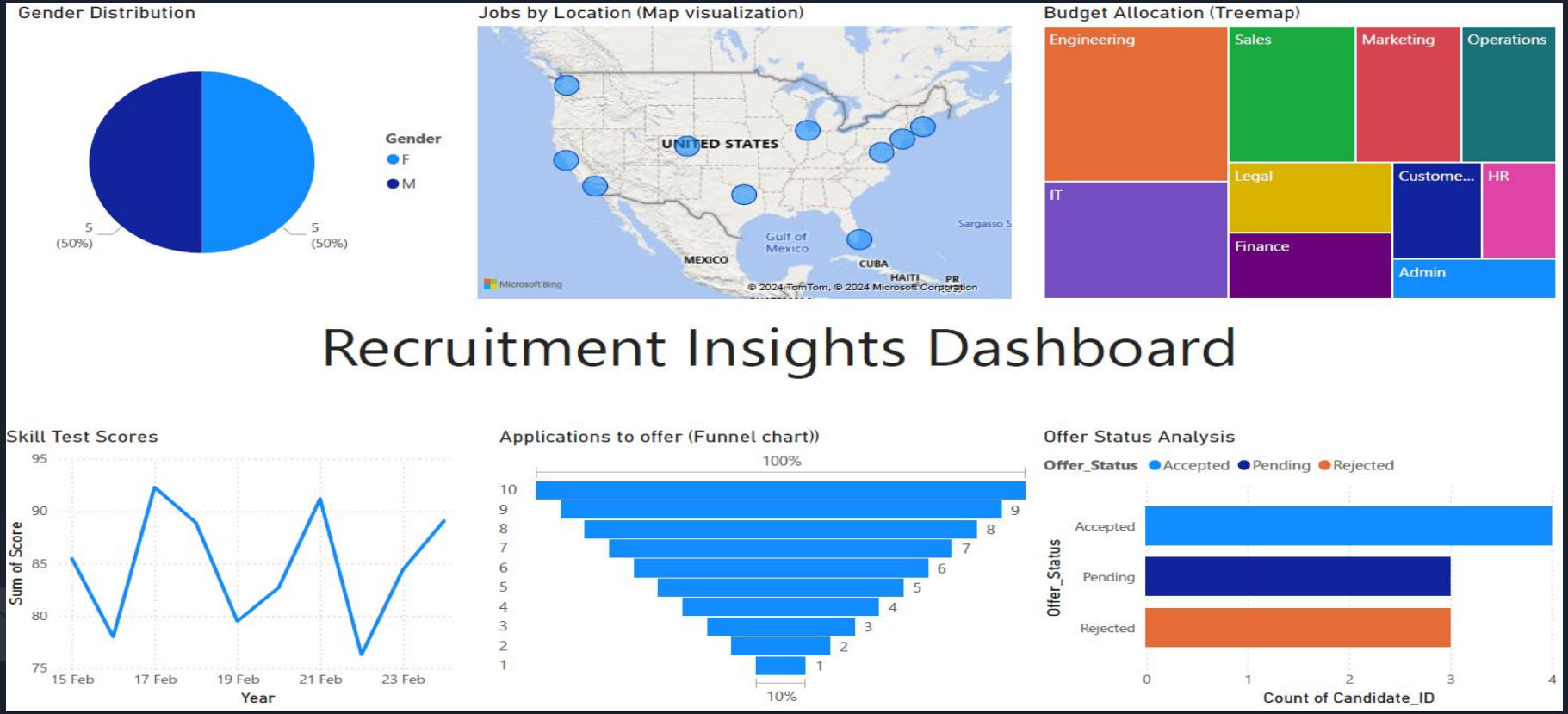


```sql
-- NON CLUSTERED INDICES
-- 1.  Non-Clustered Index on Candidate_EmailID in Candidate Table
CREATE NONCLUSTERED INDEX IDX_Candidate_EmailID
ON Candidate (Candidate_EmailID);

-- 2.  Non-Clustered Index on Job_ID in Job_Posting_Requirement Table
CREATE NONCLUSTERED INDEX IDX_Job_Posting_Requirement_JobID
ON Job_Posting_Requirement (Job_ID);

-- 3.  Non- Clustered Index on Date_of_Interview in Interview_Schedule Table
CREATE NONCLUSTERED INDEX IDX_Interview_Schedule_Date
ON Interview_Schedule (Date_of_Interview);

-- View all indices in the database
SELECT
    t.name AS TableName,
    i.name AS IndexName,
    i.type_desc AS IndexType,
    i.is_unique AS IsUnique,
    i.is_primary_key AS IsPrimaryKey
FROM
```

Results | Messages

	TableName	Index…	Index…	IsUnique	IsPrimaryKey
1	Candidate	IDX_Cand…	NONCLUST…	0	0
2	Candidate	PK__Cand…	CLUSTERED	1	1
3	Candidate	UQ__Cand…	NONCLUST…	1	0
4	Candidate	UQ__Cand…	NONCLUST…	1	0
5	Department	PK__Depa…	CLUSTERED	1	1
6	Employee	PK__Empl…	CLUSTERED	1	1
7	Employee	UQ__Empl…	NONCLUST…	1	0
8	Interview	PK__Inte…	CLUSTERED	1	1
9	Interview_Sch…	IDX_Inte…	NONCLUST…	0	0
1…	Interview_Sch…	PK__Inte…	CLUSTERED	1	1
1…	Job_Applicati…	PK__Job_…	CLUSTERED	1	1
1…	Job_Posting	PK__Job_…	CLUSTERED	1	1
1…	Job_Posting_R…	IDX_Job_…	NONCLUST…	0	0
1…	Job_Posting_R…	PK__Job_…	CLUSTERED	1	1
1…	Offer	PK__Offe…	CLUSTERED	1	1
1…	Offer_Log	PK__Offe…	CLUSTERED	1	1
1…	Referral	PK__Refe…	CLUSTERED	1	1
1…	Roles	PK__Role…	CLUSTERED	1	1
1…	Shortlisted_C…	PK__Shor…	CLUSTERED	1	1
2…	Skill_Test	PK__Skil…	CLUSTERED	1	1

DATA INSIGHTS & VISUALIZATION



THANK YOU