

CSE 581 – Intro To Database Management Systems

Project 2

Rachana Mohan Fulsundar

SUID : 512373928

12/11/2022

Abstract -

In this project, we will design, implement, and test a database for the Recruitment branch of the HR Department in a company.

The interview process will update a candidate's status from time-to-time. Focus is on tracking the status of candidate's application.

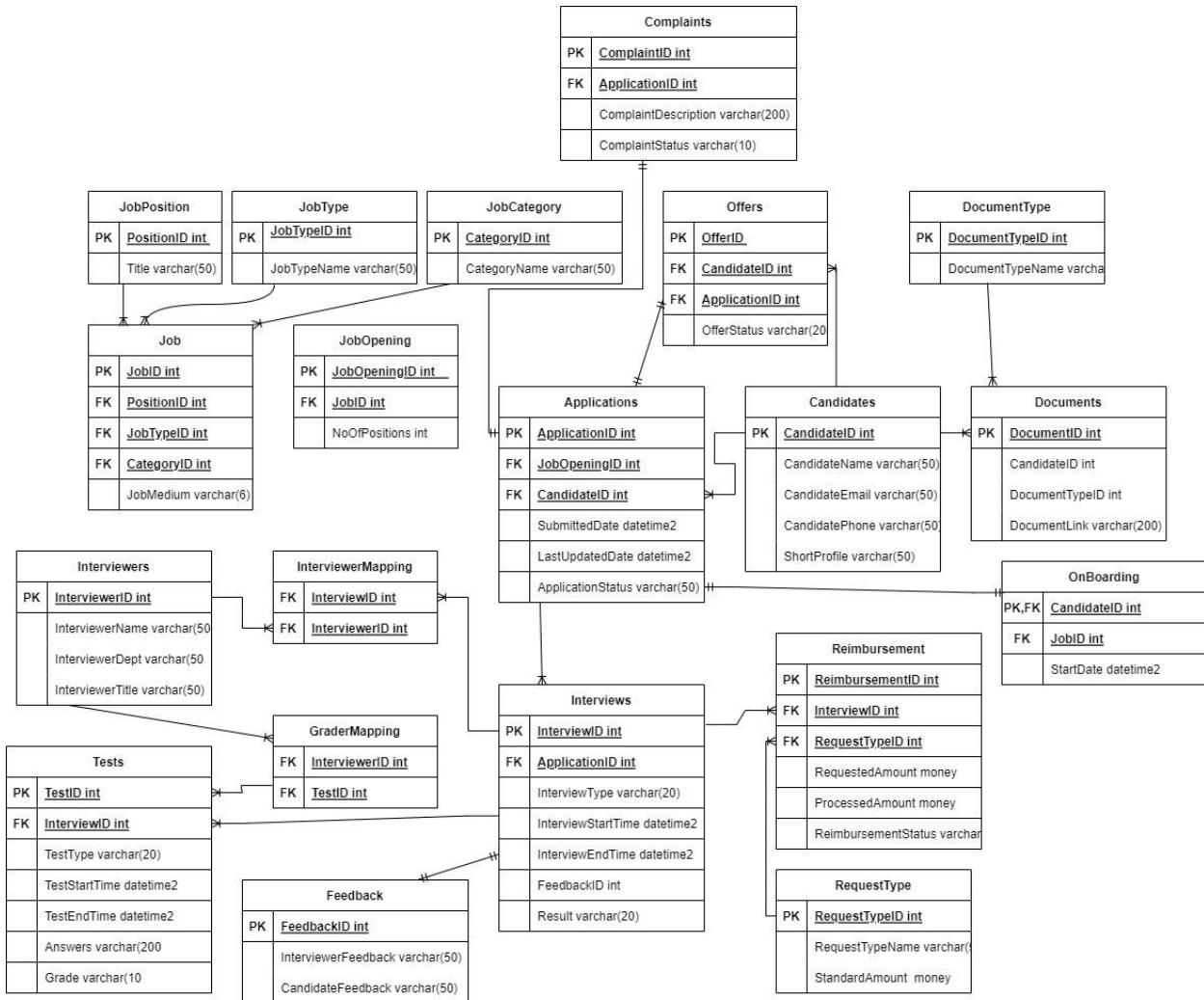
We will write views, stored procedures, user defined functions, transactions, triggers, scripts for this database.

Contents

Section A : Design	5
Database Schema -	5
Section B : Implementation	6
Creating Tables -.....	6
Inserting Data Into Tables -	11
Displaying data from all the tables -	17
Section C : Testing	22
Views -.....	22
Stored Procedures -	25
User Defined Functions -.....	30
Transactions -	34
Scripts -	38
Triggers -	41
Security Roles –	45
Section D : Conclusion	50

Section A : Design

Database Schema -



Section B : Implementation

Creating Database -

```
USE master
```

```
GO
```

```
IF DB_ID('Recruitment') IS NOT NULL
```

```
    DROP DATABASE Recruitment
```

```
GO
```

```
CREATE DATABASE Recruitment
```

```
GO
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "Project2.sql - DESKTOP-5MGKLUU.master (DESKTOP-5MGKLUU\rfulsund (70)) - Microsoft SQL Server Management Studio". The left pane is the Object Explorer, showing a tree structure of databases, system databases, and other objects under "DESKTOP-5MGKLUU (SQL Server 15.0.2000.5 - DESKTOP)". The right pane contains a query editor window titled "Project2.sql - DESKTOP-5MGKLUU (SQL Server 15.0.2000.5 - DESKTOP)". The query is:

```
USE master
GO
IF DB_ID('Recruitment') IS NOT NULL
    DROP DATABASE Recruitment
GO
CREATE DATABASE Recruitment
GO
```

The status bar at the bottom indicates "Commands completed successfully." and "Completion time: 2022-12-10T19:44:76Z". The taskbar at the bottom shows the Windows Start button, a search bar, and several pinned application icons.

Creating Tables -

```
USE Recruitment
```

```
GO
```

```
CREATE TABLE JobPosition
```

```
    PositionID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    Title varchar(50) NOT NULL
```

```
)
```

```
GO
```

```

CREATE TABLE JobType(
    JobTypeID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    JobTypeName varchar(50) NOT NULL
)
GO

CREATE TABLE JobCategory(
    CategoryID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    CategoryName varchar(50) NOT NULL
)
GO

CREATE TABLE Job(
    JobID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    PositionID int NOT NULL REFERENCES JobPosition(PositionID),
    JobTypeID int NOT NULL REFERENCES JobType(JobTypeID),
    CategoryID int NOT NULL REFERENCES JobCategory(CategoryID),
    JobMedium varchar(6) NOT NULL,
    CONSTRAINT CHK_Medium CHECK (lower(JobMedium) IN ('onsite','online'))
)
GO

CREATE TABLE JobOpening(
    JobOpeningID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    JobID int NOT NULL REFERENCES Job(JobID),
    NoOfPositions int NOT NULL
)
GO

CREATE TABLE Candidates(
    CandidateID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    CandidateName varchar(50) NOT NULL,
    CandidateEmail varchar(50) NOT NULL,
    CandidatePhone varchar(50) NOT NULL,
    ShortProfile varchar(50) NULL
)
GO

CREATE TABLE Applications(
    ApplicationID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    JobOpeningID int NOT NULL REFERENCES JobOpening(JobOpeningID),
    CandidateID int NOT NULL REFERENCES Candidates(CandidateID),
    SubmittedDate datetime2 NOT NULL,

```

```

LastUpdatedDate datetime2 NOT NULL,
ApplicationStatus varchar(50) NOT NULL
)
GO

CREATE TABLE Offers(
OfferID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
CandidateID int NOT NULL REFERENCES Candidates(CandidateID),
ApplicationID int NOT NULL REFERENCES Applications(ApplicationID),
OfferStatus varchar(20) NOT NULL,
CONSTRAINT CHK_OfferStatus CHECK (lower(OfferStatus) IN
('accepted','declined','negotiating','actionneeded'))
)
GO

CREATE TABLE Feedback(
FeedbackID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
InterviewerFeedback varchar(50) NOT NULL,
CandidateFeedback varchar(50) NULL
)
GO

CREATE TABLE Interviews(
InterviewID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
ApplicationID int NOT NULL REFERENCES Applications(ApplicationID),
InterviewType varchar(20) NOT NULL,
CONSTRAINT CHK_InterviewType CHECK (lower(InterviewType) IN ('onsite','online')),
InterviewStartTime datetime2 NOT NULL,
InterviewEndTime datetime2 NOT NULL,
FeedbackID int NOT NULL REFERENCES Feedback(FeedbackID),
Result varchar(20) NOT NULL,
CONSTRAINT CHK_Result CHECK (lower(Result) IN ('accepted','rejected'))
)
GO

CREATE TABLE Interviewers(
InterviewerID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
InterviewerName varchar(50) NOT NULL,
InterviewerDept varchar(50) NOT NULL,
InterviewerTitle varchar(50) NOT NULL,
)
GO

CREATE TABLE InterviewerMapping(

```

```

InterviewID int NOT NULL REFERENCES Interviews(InterviewID),
InterviewerID int NOT NULL REFERENCES Interviewers(InterviewerID)
)
GO

CREATE TABLE Tests(
    TestID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    InterviewID int NOT NULL REFERENCES Interviews(InterviewID),
    TestType varchar(20) NOT NULL,
    CONSTRAINT CHK_TestType CHECK (lower(TestType) IN ('onsite','online')),
    TestStartTime datetime2 NOT NULL,
    TestEndTime datetime2 NOT NULL,
    Answers varchar(200) NOT NULL,
    Grade varchar(10) NOT NULL,
    CONSTRAINT CHK_Grade CHECK (lower(Grade) IN ('passed','failed'))
)
GO

CREATE TABLE GraderMapping(
    InterviewerID int NOT NULL REFERENCES Interviewers(InterviewerID),
    TestID int NOT NULL REFERENCES Tests(TestID)
)
GO

CREATE TABLE RequestType(
    RequestTypeID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    RequestTypeName varchar(50) NOT NULL,
    StandardAmount money NOT NULL
)
GO

CREATE TABLE Reimbursement(
    ReimbursementID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    InterviewID int NOT NULL REFERENCES Interviews(InterviewID),
    RequestTypeID int NOT NULL REFERENCES RequestType(RequestTypeID),
    RequestedAmount money DEFAULT 0,
    ProcessedAmount money DEFAULT 0,
    ReimbursementStatus varchar(20) NOT NULL,
    CONSTRAINT CHK_Status CHECK (lower(ReimbursementStatus) IN ('processed','submitted')),
    ReceiptLink varchar(200) NOT NULL
)
GO

```

```
CREATE TABLE DocumentType(
    DocumentTypeID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    DocumentTypeName varchar(50) NOT NULL
)
GO

CREATE TABLE Documents(
    DocumentID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    CandidateID int NOT NULL REFERENCES Candidates(CandidateID),
    DocumentTypeID int NOT NULL REFERENCES DocumentType(DocumentTypeID),
    DocumentLink varchar(200) NOT NULL
)
GO

CREATE TABLE Complaints(
    ComplaintID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    ApplicationID int NOT NULL REFERENCES Applications(ApplicationID),
    ComplaintDescription varchar(200) NOT NULL,
    ComplaintStatus varchar(10) NOT NULL
)
GO

CREATE TABLE OnBoarding(
    CandidateID int NOT NULL REFERENCES Candidates(CandidateID) PRIMARY KEY,
    JobID int NOT NULL REFERENCES Job(JobID),
    StartDate datetime2 NOT NULL
)
GO
```

```

Project2.sql - DESKTOP-5MGKLN0.Recruitment (DESKTOP-5MGKLN0\rfulsund (70)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
Quick Launch (Ctrl+Q) ×
Object Explorer
Connect ▾
DESKTOP-5MGKLN0 (SQL Server 15.0.2000.5 - DESKTOP-5MGKLN0)
Databases
System Databases
Database Snapshots
AP
College
Examples
MyCollege
ProductOrders
Recruitment
School
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabled)
XEvent Profiler
Project Explorer
Project2.sql - DESKTOP-5MGKLN0.Recruitment (DESKTOP-5MGKLN0\rfulsund (70)) ×
CREATE TABLE Complaints(
    CandidateID int NOT NULL REFERENCES Candidates(CandidateID),
    DocumentTypeID int NOT NULL REFERENCES DocumentType(DocumentTypeID),
    DocumentLink varchar(200) NOT NULL
)
GO

CREATE TABLE OnBoarding(
    CandidateID int NOT NULL REFERENCES Candidates(CandidateID) PRIMARY KEY,
    JobID int NOT NULL REFERENCES Job(JobID),
    StartDate datetime2 NOT NULL
)
GO

CREATE TABLE DocumentLink(
    DocumentLinkID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    DocumentType nvarchar(50) NOT NULL,
    DocumentLink varchar(200) NOT NULL,
    CONSTRAINT CHK_DocumentLink CHECK (lower(DocumentLink) IN ('valid','invalid'))
)
GO
100 %
Messages
Commands completed successfully.

Completion time: 2022-12-10T22:57:23.986368-06:00

```

Inserting Data Into Tables -

```

INSERT INTO JobPosition VALUES
('IT Manager'),
('Software Developer'),
('Project Manager'),
('Team Lead'),
('System Engineer'),
('Associate Developer'),
('Senior Software Developer'),
('Business Analyst'),
('Senior Business Analyst'),
('Product Manager'),
('Database Engineer'),
('QA Engineer'),
('Cyber Security Analyst');

```

```

INSERT INTO JobCategory VALUES
('Information Security'),
('Software Development'),
('Management'),
('QA'),
('Analyst'),

```

```
('Database Administrator'),  
('Cyber Security');
```

```
INSERT INTO JobType VALUES  
('Summer Internship'),  
('Full-time Job'),  
('Part-time job'),  
('Contract-based');
```

```
INSERT INTO Job VALUES  
(7,1,2,'onsite'),  
(4,2,3,'online'),  
(1,3,3,'online'),  
(3,4,3,'onsite'),  
(2,1,2,'online'),  
(8,2,5,'onsite'),  
(11,3,6,'online'),  
(10,4,3,'onsite'),  
(6,1,2,'online'),  
(3,2,3,'onsite');
```

```
INSERT INTO JobOpening VALUES  
(3,2),  
(5,1),  
(2,3),  
(8,1),  
(4,1),  
(1,3),  
(9,2),  
(7,1);
```

```
INSERT INTO Candidates VALUES  
('Rachana Fulsundar','rachana.fulsundar@gmail.com','315433567','Java Developer'),  
('Pallavi Kale','pallavi.kale@gmail.com','315477567','Front End Developer'),  
('Pritesh Baria','priteshbaria@gmail.com','315498567','Business Analyst'),  
('Shubham Gangwal','sgangwal@gmail.com','315223567','Database Engineer'),  
('Priti Badhe','priti.badhe@gmail.com','311133567','UI Developer'),  
('Kimaya Khilare','kimkhilare@gmail.com','315433560','Management'),  
('Pratiksha Gadia','pratikgadia@gmail.com','315433587','Java Developer'),  
('Siddharth Joshi','joshisiddharth@gmail.com','315993567','Front End Developer'),  
('Alok Tupe','aloktupe@gmail.com','315400567','Business Analyst'),  
('Surabhi Metre','surabhimetre@gmail.com','315422567','Database Engineer'),  
('Janhavi Hinge','janhavi.hinge@gmail.com','315455567','UI Developer'),  
('Pragati Nikam','pragatinikam@gmail.com','315433767','Front End Developer');
```

```
INSERT INTO Applications VALUES  
(1,1,dateadd(w, -3, GETDATE()),GETDATE(),'offer extended'),  
(2,1,dateadd(w, -4, GETDATE()),GETDATE(),'interview1'),  
(7,4,dateadd(w, -8, GETDATE()),GETDATE(),'applied'),  
(1,8,dateadd(w, -1, GETDATE()),GETDATE(),'offer accepted'),  
(3,7,dateadd(w, -2, GETDATE()),GETDATE(),'applied'),  
(5,1,dateadd(w, -3, GETDATE()),GETDATE(),'interview 2'),  
(1,9,dateadd(w, -3, GETDATE()),GETDATE(),'rejected'),  
(5,10,dateadd(w, -5,GETDATE()),GETDATE(),'interview 2'),  
(8,4,dateadd(w, -3, GETDATE()),GETDATE(),'applied'),  
(7,2,dateadd(w, -4, GETDATE()),GETDATE(),'onboarding');
```

INSERT INTO Offers VALUES

```
(1,1,'actionneeded'),  
(4,1,'accepted');
```

INSERT INTO Offers VALUES

```
(1,1,'actionneeded'),  
(4,8,'accepted'),  
(10,2,'accepted');
```

INSERT INTO Feedback VALUES

```
('Interview 1 completed, proceed to interview 2',null),  
('Interview 1 completed, candidate is rejected',null),  
('Interview 1 completed, proceed to interview 2',null),  
('Interview 1 completed, proceed to interview 2',null),  
('Interview 2 completed, candidate is selected',null),  
('Interview 2 completed, proceed to interview 3',null),  
('Interview 1 completed, proceed to interview 2',null),  
('Interview 1 completed, proceed to interview 2',null),  
('Interview 1 completed, proceed to interview 2',null),  
('Interview 2 completed, candidate is selected',null),  
('Interview 2 completed, proceed to interview 3',null),  
('Interview 3 completed, candidate is selected',null),  
('Interview 2 completed, proceed to interview 3',null);
```

INSERT INTO Interviews VALUES

```
(1,'onsite',dateadd(hh, -12, dateadd(w,-7 ,GETDATE()))),dateadd(hh, -10, dateadd(w,-7 ,GETDATE())),1,'accepted'),  
(7,'online',dateadd(hh, -12, dateadd(w,-8 ,GETDATE()))),dateadd(hh, -10, dateadd(w,-8 ,GETDATE())),2,'rejected'),
```

```

(8,'online',dateadd(hh,-10,dateadd(w,-9,GETDATE())),dateadd(hh,-8,dateadd(w,-9
,GETDATE())),3,'accepted'),
(6,'online',dateadd(hh,-9,dateadd(w,-8,GETDATE())),dateadd(hh,-7,dateadd(w,-8
,GETDATE())),4,'accepted'),
(1,'onsite',dateadd(hh,-12,dateadd(w,-5,GETDATE())),dateadd(hh,-10,dateadd(w,-5
,GETDATE())),5,'accepted'),
(8,'online',dateadd(hh,-12,dateadd(w,-7,GETDATE())),dateadd(hh,-10,dateadd(w,-7
,GETDATE())),6,'accepted'),
(2,'onsite',dateadd(hh,-4,dateadd(w,-7,GETDATE())),dateadd(hh,-2,dateadd(w,-7
,GETDATE())),7,'accepted'),
(10,'online',dateadd(hh,-12,dateadd(w,-15,GETDATE())),dateadd(hh,-10,dateadd(w,-
15,GETDATE())),8,'accepted'),
(4,'onsite',dateadd(hh,-12,dateadd(w,-14,GETDATE())),dateadd(hh,-10,dateadd(w,-14
,GETDATE())),9,'accepted'),
(10,'online',dateadd(hh,-12,dateadd(w,-13,GETDATE())),dateadd(hh,-10,dateadd(w,-
13,GETDATE())),10,'accepted'),
(4,'onsite',dateadd(hh,-12,dateadd(w,-12,GETDATE())),dateadd(hh,-10,dateadd(w,-12
,GETDATE())),11,'accepted'),
(4,'online',dateadd(hh,-12,dateadd(w,-10,GETDATE())),dateadd(hh,-10,dateadd(w,-10
,GETDATE())),12,'accepted'),
(6,'onsite',dateadd(hh,-6,dateadd(w,-7,GETDATE())),dateadd(hh,-4,dateadd(w,-7
,GETDATE())),13,'accepted');

```

```

INSERT INTO Interviewers VALUES
('Mrunmayee Mokashi','Management','HR'),
('Dhanshree Lonkar','Senior Software Developer','Software Development'),
('Rahul Bankar','Senior Software Developer','Software Development'),
('Luvy Sharma','Management','HR'),
('Pratik Gadia','Management','HR');

```

```

INSERT INTO InterviewerMapping VALUES
(2,2),
(3,3),
(4,1),
(2,2),
(5,5),
(6,3),
(2,5),
(8,2),
(8,1),
(9,3),
(10,4),
(12,2),
(14,1),

```

(7,3),
(13,4),
(11,4);

INSERT INTO Tests VALUES

(2,'online',dateadd(hh,-9,dateadd(w,-7,GETDATE())),dateadd(hh,-8,dateadd(w,-7,GETDATE()))),'1-c 2-d 3-d 4-a 5-b','passed'),
(3,'onsite',dateadd(hh,-9,dateadd(w,-8,GETDATE())),dateadd(hh,-8,dateadd(w,-8,GETDATE()))),' ','failed'),
(6,'onsite',dateadd(hh,-8,dateadd(w,-9,GETDATE())),dateadd(hh,-7,dateadd(w,-9,GETDATE()))),'1-c 2-d 3-d 4-a','passed'),
(2,'online',dateadd(hh,-6,dateadd(w,-8,GETDATE())),dateadd(hh,-5,dateadd(w,-8,GETDATE()))),'1-c 3-d 4-a 5-b','passed'),
(7,'online',dateadd(hh,-10,dateadd(w,-4,GETDATE())),dateadd(hh,-9,dateadd(w,-3,GETDATE()))),'1-d 2-d 3-d 5-b','passed'),
(10,'online',dateadd(hh,-9,dateadd(w,-7,GETDATE())),dateadd(hh,-8,dateadd(w,-7,GETDATE()))),'1-c 2-d 3-d 4-a 5-b','passed'),
(4,'onsite',dateadd(hh,-3,dateadd(w,-7,GETDATE())),dateadd(hh,-2,dateadd(w,-2,GETDATE()))),'1-d 2-d 3-d 5-b','passed'),
(8,'onsite',dateadd(hh,-9,dateadd(w,-15,GETDATE())),dateadd(hh,-8,dateadd(w,-15,GETDATE()))),'1-c 2-d 3-d 4-a 5-b','passed'),
(5,'online',dateadd(hh,-9,dateadd(w,-14,GETDATE())),dateadd(hh,-8,dateadd(w,-14,GETDATE()))),'1-c 2-d 3-d 4-a 5-b','passed'),
(9,'onsite',dateadd(hh,-9,dateadd(w,-13,GETDATE())),dateadd(hh,-8,dateadd(w,-13,GETDATE()))),'1-d 2-d 3-d 5-b','passed'),
(2,'onsite',dateadd(hh,-9,dateadd(w,-12,GETDATE())),dateadd(hh,-8,dateadd(w,-12,GETDATE()))),'1-d 2-d 3-d 5-b','passed'),
(10,'online',dateadd(hh,-9,dateadd(w,-10,GETDATE())),dateadd(hh,-8,dateadd(w,-10,GETDATE()))),'1-c 2-d 3-d 4-a 5-b','passed'),
(12,'onsite',dateadd(hh,-3,dateadd(w,-7,GETDATE())),dateadd(hh,-2,dateadd(w,-7,GETDATE()))),'1-c 2-d 3-d 4-a 5-b','passed'),
(11,'online',dateadd(hh,-9,dateadd(w,-7,GETDATE())),dateadd(hh,-8,dateadd(w,-7,GETDATE()))),'1-c 2-d 3-d 4-a 5-b','passed'),
(3,'onsite',dateadd(hh,-9,dateadd(w,-5,GETDATE())),dateadd(hh,-8,dateadd(w,-5,GETDATE()))),'1-d 2-d 3-d 5-b','failed'),
(13,'online',dateadd(hh,-9,dateadd(w,-5,GETDATE())),dateadd(hh,-8,dateadd(w,-5,GETDATE()))),'1-c 2-d 3-d 4-a 5-b','passed');

INSERT INTO GraderMapping VALUES

(2,1),
(1,3),
(1,2),
(4,1),
(1,6),

(3,5),
(4,2),
(1,1),
(3,3),
(2,7),
(1,9),
(1,12),
(3,8),
(1,11),
(2,14),
(4,10),
(3,15),
(2,14),
(1,13),
(2,16);

INSERT INTO RequestType VALUES
('Airline reservation',500.0000),
('Hotel reservation',250.0000),
('Car Rental',150.0000),
('Food Expenses',100.0000);

INSERT INTO Reimbursement VALUES
(2,1,700.0000,500.0000, 'processed', 'receipt.docx'),
(6,2,150.0000,150.0000,'processed', 'bill.docx'),
(8,3,140.0000,0, 'submitted', 'uber1.docx'),
(10,4,80.0000, 80.0000,'processed', 'receipt.docx'),
(12,4,120.0000,100.0000, 'processed', 'dunkin.docx'),
(14,4,74.0000,74.0000, 'processed', 'subway.docx'),
(2,4,90.0000, 0,'submitted', 'dunkin.docx'),
(10,3,140.0000,0, 'submitted', 'uber1.docx'),
(12,2,150.0000,150.0000,'processed', 'bill1.docx'),
(12,4,74.0000,74.0000, 'processed', 'subway1.docx');

INSERT INTO DocumentType VALUES
('CV'), ('Reference letters'),('Cover letter');

INSERT INTO Documents VALUES
(1,1,'CV1.pdf'),
(6,3,'COver1.pdf'),
(2,1,'resume.pdf'),
(8,1,'resume2.pdf'),

```
(5,1,'res.pdf'),
(1,2,'ref2.pdf'),
(2,1,'resume2.pdf'),
(7,1,'res2.pdf'),
(2,3,'CoverPage.pdf'),
(10,2,'ref2.pdf'),
(9,1,'resume2.pdf');
```

```
INSERT INTO Complaints VALUES
(7,'Complex questions were asked','Reviewing');
```

```
INSERT INTO OnBoarding VALUES
```

```
(1,1,dateadd(w,30 ,GETDATE())),
(8,4,dateadd(w,-60 ,GETDATE())),
(2,10,dateadd(w,-30 ,GETDATE()));
```

Displaying data from all the tables -

The screenshot shows the Microsoft SQL Server Management Studio interface with three result sets displayed in the Results tab:

- JobPosition:**

PositionID	Title
1	IT Manager
2	Software Developer
3	Project Manager
4	Team Lead
5	System Engineer
6	Associate Developer
7	Senior Software Developer
8	Business Analyst
- JobType:**

JobTypeID	JobTypeName
1	Summer Internship
2	Full-time Job
3	Part-time job
4	Contract-based
- JobCategory:**

CategoryID	CategoryName
1	Information Security
2	Software Development
3	Management
4	QA
5	Analyst
6	Database Administrator

At the bottom of the results pane, it says "Query executed successfully."

Project2.sql - DESKTOP-5MGKLUU.Recruitment (DESKTOP-5MGKLUU\rfulsund (70)) - Microsoft SQL Server Management Studio

```
File Edit View Query Project Tools Window Help
Recruitment Execute
Object Explorer
Project2.sql - DESKTOP-5MGKLUU.Recruitment (DESKTOP-5MGKLUU\rfulsund (70)) x
SELECT * FROM Job
SELECT * FROM JobOpening
```

Results Messages

JobID	PositionID	JobTypeID	CategoryID	JobMedium
1	7	1	2	onsite
2	4	2	3	online
3	1	3	3	online
4	4	3	3	onsite
5	2	1	2	online
6	8	2	5	onsite
7	11	3	6	online
8	10	4	3	onsite
9	6	1	2	online
10	3	2	3	onsite

JobOpeningID	JobID	NoOfPositions
1	3	2
2	5	1
3	2	3
4	8	1
5	4	1
6	1	3
7	9	2
8	7	1

Query executed successfully.

DESKTOP-5MGKLUU (15.0 RTM) | DESKTOP-5MGKLUU\rfulsu... | Recruitment | 00:00:00 | 18 rows

Ln 437 Col 1 Ch 1 INS

Item(s) Saved Type here to search

Project2.sql - DESKTOP-5MGKLUU.Recruitment (DESKTOP-5MGKLUU\rfulsund (70)) - Microsoft SQL Server Management Studio

```
File Edit View Query Project Tools Window Help
Recruitment Execute
Object Explorer
Project2.sql - DESKTOP-5MGKLUU.Recruitment (DESKTOP-5MGKLUU\rfulsund (70)) x
SELECT * FROM Interviews
SELECT * FROM Interviewers
SELECT * FROM InterviewerMapping
```

Results Messages

InterviewID	ApplicationID	InterviewType	InterviewStartTime	InterviewEndTime	FeedbackID	Result
1	2	onsite	2022-12-03 15:40:09.8000000	2022-12-03 17:40:09.8000000	1	accepted
2	3	online	2022-12-02 15:40:09.8000000	2022-12-02 17:40:09.8000000	2	rejected
3	4	online	2022-12-01 17:40:09.8000000	2022-12-01 19:40:09.8000000	3	accepted
4	5	online	2022-12-02 18:40:09.8000000	2022-12-02 20:40:09.8000000	4	accepted
5	6	onsite	2022-12-05 15:40:09.8000000	2022-12-05 17:40:09.8000000	5	accepted
6	7	online	2022-12-03 15:40:09.8000000	2022-12-03 17:40:09.8000000	6	accepted
7	8	onsite	2022-12-03 23:40:09.8000000	2022-12-04 01:40:09.8000000	7	accepted
8	9	online	2022-11-25 15:40:09.8000000	2022-11-25 17:40:09.8000000	8	accepted

InterviewerID	InterviewerName	InterviewerDept	InterviewerTitle
1	Mrunmayee Mokashi	Management	HR
2	Dhanshree Lonkar	Senior Software Developer	Software Development
3	Rahul Bankar	Senior Software Developer	Software Development
4	Luvy Shama	Management	HR
5	Pratik Gadia	Management	HR

InterviewID	InterviewerID
1	2
2	3
3	1
4	2
5	5

Query executed successfully.

DESKTOP-5MGKLUU (15.0 RTM) | DESKTOP-5MGKLUU\rfulsu... | Recruitment | 00:00:00 | 34 rows

Ln 439 Col 33 Ch 33 INS

Item(s) Saved Type here to search

Project2.sql - DESKTOP-5MGKLNNU.Recruitment (DESKTOP-5MGKLNNU\rfulsund (70)) - Microsoft SQL Server Management Studio

```
File Edit View Query Project Tools Window Help
Recruitment Execute
Object Explorer
Project2.sql - DESKTOP-5MGKLNNU.Recruitment (DESKTOP-5MGKLNNU\rfulsund (70))
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Applications
dbo.Candidates
dbo.Complaints
Columns
Keys
Constraints
Triggers
Indexes
Statistics
dbo.Documents
dbo.DocumentType
dbo.Feedback
dbo.GraderMapping
dbo.InterviewerMapping
dbo.Interviewers
dbo.Interviews
dbo.Job
dbo.JobCategory
dbo.JobOpening
dbo.JobPosition
dbo.JobType
Item(s) Saved
Type here to search
Project2.sql - DESKTOP-5MGKLNNU.Recruitment (DESKTOP-5MGKLNNU\rfulsund (70))
SELECT * FROM Interviews
SELECT * FROM Feedback
```

Results Messages

InterviewID	ApplicationID	InterviewType	InterviewStartTime	InterviewEndTime	FeedbackID	Result
1	2	onsite	2022-12-03 15:40:09.8000000	2022-12-03 17:40:09.8000000	1	accepted
2	3	online	2022-12-02 15:40:09.8000000	2022-12-02 17:40:09.8000000	2	rejected
3	4	online	2022-12-01 17:40:09.8000000	2022-12-01 19:40:09.8000000	3	accepted
4	5	online	2022-12-02 18:40:09.8000000	2022-12-02 20:40:09.8000000	4	accepted
5	6	onsite	2022-12-05 15:40:09.8000000	2022-12-05 17:40:09.8000000	5	accepted
6	7	online	2022-12-03 15:40:09.8000000	2022-12-03 17:40:09.8000000	6	accepted
7	8	onsite	2022-12-03 23:40:09.8000000	2022-12-04 01:40:09.8000000	7	accepted
8	9	online	2022-11-25 15:40:09.8000000	2022-11-25 17:40:09.8000000	8	accepted
9	10	onsite	2022-11-26 15:40:09.8000000	2022-11-26 17:40:09.8000000	9	accepted
10	11	online	2022-11-27 15:40:09.8000000	2022-11-27 17:40:09.8000000	10	accepted

FeedbackID	InterviewFeedback	CandidateFeedback
1	Interview 1 completed, proceed to interview 2	NULL
2	Interview 1 completed, candidate is rejected	NULL
3	Interview 1 completed, proceed to interview 2	NULL
4	Interview 1 completed, proceed to interview 2	NULL
5	Interview 2 completed, candidate is selected	NULL
6	Interview 2 completed, proceed to interview 3	NULL
7	Interview 1 completed, proceed to interview 2	NULL
8	Interview 1 completed, proceed to interview 2	NULL
9	Interview 1 completed, proceed to interview 2	NULL
10	Interview 2 completed, candidate is selected	NULL

Query executed successfully.

DESKTOP-5MGKLNNU (15.0 RTM) DESKTOP-5MGKLNNU\rfulsu... Recruitment 00:00:00 26 rows

Ln 438 Col 23 Ch 23 INS

Item(s) Saved Type here to search

Project2.sql - DESKTOP-5MGKLNNU.Recruitment (DESKTOP-5MGKLNNU\rfulsund (70)) - Microsoft SQL Server Management Studio

```
File Edit View Query Project Tools Window Help
Recruitment Execute
Object Explorer
Project2.sql - DESKTOP-5MGKLNNU.Recruitment (DESKTOP-5MGKLNNU\rfulsund (70))
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Applications
dbo.Candidates
dbo.Complaints
Columns
Keys
Constraints
Triggers
Indexes
Statistics
dbo.Documents
dbo.DocumentType
dbo.Feedback
dbo.GraderMapping
dbo.InterviewerMapping
dbo.Interviewers
dbo.Interviews
dbo.Job
dbo.JobCategory
dbo.JobOpening
dbo.JobPosition
dbo.JobType
Item(s) Saved
Type here to search
Project2.sql - DESKTOP-5MGKLNNU.Recruitment (DESKTOP-5MGKLNNU\rfulsund (70))
SELECT * FROM Tests
SELECT * FROM GraderMapping
```

Results Messages

TestID	InterviewID	TestType	TestStartTime	TestEndTime	Answers	Grade
1	2	online	2022-12-03 19:27:07.6233333	2022-12-03 20:27:07.6233333	1-c 2-d 3-d 4-a 5-b	passed
2	3	onsite	2022-12-02 19:27:07.6233333	2022-12-02 20:27:07.6233333		failed
3	3	onsite	2022-12-01 20:27:07.6233333	2022-12-01 21:27:07.6233333	1-c 2-d 3-d 4-a	passed
4	4	online	2022-12-02 22:27:07.6233333	2022-12-02 23:27:07.6233333	1-c 3-d 4-a 5-b	passed
5	5	online	2022-12-06 19:27:07.6233333	2022-12-07 19:27:07.6233333	1-d 2-d 3-d 5-b	passed
6	6	online	2022-12-03 19:27:07.6233333	2022-12-03 20:27:07.6233333	1-c 2-d 3-d 4-a 5-b	passed
7	7	online	2022-12-04 01:27:07.6233333	2022-12-09 02:27:07.6233333	1-d 2-d 3-d 5-b	passed
8	8	onsite	2022-11-25 19:27:07.6233333	2022-11-25 20:27:07.6233333	1-c 2-d 3-d 4-a 5-b	passed
9	9	online	2022-11-26 19:27:07.6233333	2022-11-26 20:27:07.6233333	1-c 2-d 3-d 4-a 5-b	passed
10	9	onsite	2022-11-27 19:27:07.6233333	2022-11-27 20:27:07.6233333	1-d 2-d 3-d 5-b	passed

InterviewerID	TestID
1	2
2	1
3	3
4	2
5	1
6	3
7	4
8	1
9	3
10	2

Query executed successfully.

DESKTOP-5MGKLNNU (15.0 RTM) DESKTOP-5MGKLNNU\rfulsu... Recruitment 00:00:00 36 rows

Ln 438 Col 28 Ch 28 INS

Item(s) Saved Type here to search

Project2.sql - DESKTOP-5MGKLNNU.Recruitment (DESKTOP-5MGKLNNU\rfulsund (70)) - Microsoft SQL Server Management Studio

```
File Edit View Query Project Tools Window Help
Recruitment Execute
Object Explorer
Project2.sql - DESKTOP-5MGKLNNU.Recruitment (DESKTOP-5MGKLNNU\rfulsund (70)) - Results
SELECT * FROM Candidates
SELECT * FROM Offers
SELECT * FROM DocumentType
SELECT * FROM Documents
```

CandidateID	CandidateName	CandidateEmail	CandidatePhone	ShortProfile
1	Rachana Fulunder	rachana.fulunder@gmail.com	315433567	Java Developer
2	Pallavi Kale	pallavi.kale@gmail.com	315477567	Front End Developer
3	Priteesh Bana	priteebana@gmail.com	315498567	Business Analyst
4	Shubham Gangwal	sgangwal@gmail.com	315223567	Database Engineer
5	Priti Badhe	pritibadhe@gmail.com	311133567	UI Developer
6	Kimaya Khilare	kimkhilare@gmail.com	315433560	Management
7	Pratiksha Gadia	pratkshagadia@gmail.com	315433587	Java Developer
8	Siddharth Joshi	johaisdharth@gmail.com	315993567	Front End Developer

OfferID	CandidateID	ApplicationID	OfferStatus
1	3	1	actionneeded
2	4	4	accepted
3	5	10	accepted

DocumentTypeID	DocumentTypeName
1	CV
2	Reference letters
3	Cover letter

DocumentID	CandidateID	DocumentTypeID	DocumentLink
1	1	1	CV1.pdf
2	2	6	CV01.pdf
3	3	2	resume.pdf

Query executed successfully.

DESKTOP-5MGKLNNU (15.0 RTM) | DESKTOP-5MGKLNNU\rfulsu... | Recruitment | 00:00:00 | 29 rows

Item(s) Saved

Type here to search

Ln 440 Col 24 Ch 24 INS

05:35 11-12-2022

Project2.sql - DESKTOP-5MGKLNNU.Recruitment (DESKTOP-5MGKLNNU\rfulsund (70)) - Microsoft SQL Server Management Studio

```
File Edit View Query Project Tools Window Help
Recruitment Execute
Object Explorer
Project2.sql - DESKTOP-5MGKLNNU.Recruitment (DESKTOP-5MGKLNNU\rfulsund (70)) - Results
SELECT * FROM Reimbursement
SELECT * FROM RequestType
```

ReimbursementID	InterviewID	RequestTypeID	RequestedAmount	ProcessedAmount	ReimbursementStatus	ReceiptLink	
1	2	1	700.00	500.00	processed	receipt.docx	
2	2	6	150.00	150.00	processed	bill.docx	
3	3	8	140.00	0.00	submitted	uber1.docx	
4	4	10	4	80.00	processed	receipt.docx	
5	5	12	4	120.00	100.00	processed	dunkin.docx
6	6	14	4	74.00	74.00	processed	subway.docx
7	7	2	4	90.00	0.00	submitted	dunkin.docx
8	8	10	3	140.00	0.00	submitted	uber1.docx
9	9	12	2	150.00	150.00	processed	bill1.docx
10	10	12	4	74.00	74.00	processed	subway1.docx

RequestTypeID	RequestTypeName	StandardAmount
1	Airline reservation	500.00
2	Hotel reservation	250.00
3	Car Rental	150.00
4	Food Expenses	100.00

Query executed successfully.

DESKTOP-5MGKLNNU (15.0 RTM) | DESKTOP-5MGKLNNU\rfulsu... | Recruitment | 00:00:00 | 14 rows

Item(s) Saved

Type here to search

Ln 438 Col 26 Ch 26 INS

05:35 11-12-2022

Project2.sql - DESKTOP-5MGKLN0.Recruitment (DESKTOP-5MGKLN0\rfulsund (70)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Recruitment Execute

Object Explorer

Project2.sql - DESKTOP-5MGKLN0.Recruitment (DESKTOP-5MGKLN0\rfulsund (70))

```
SELECT * FROM Complaints
SELECT * FROM Onboarding
```

Results Messages

	ComplaintID	ApplicationID	ComplaintDescription	ComplaintStatus
1	2	7	Complex questions were asked	Reviewing

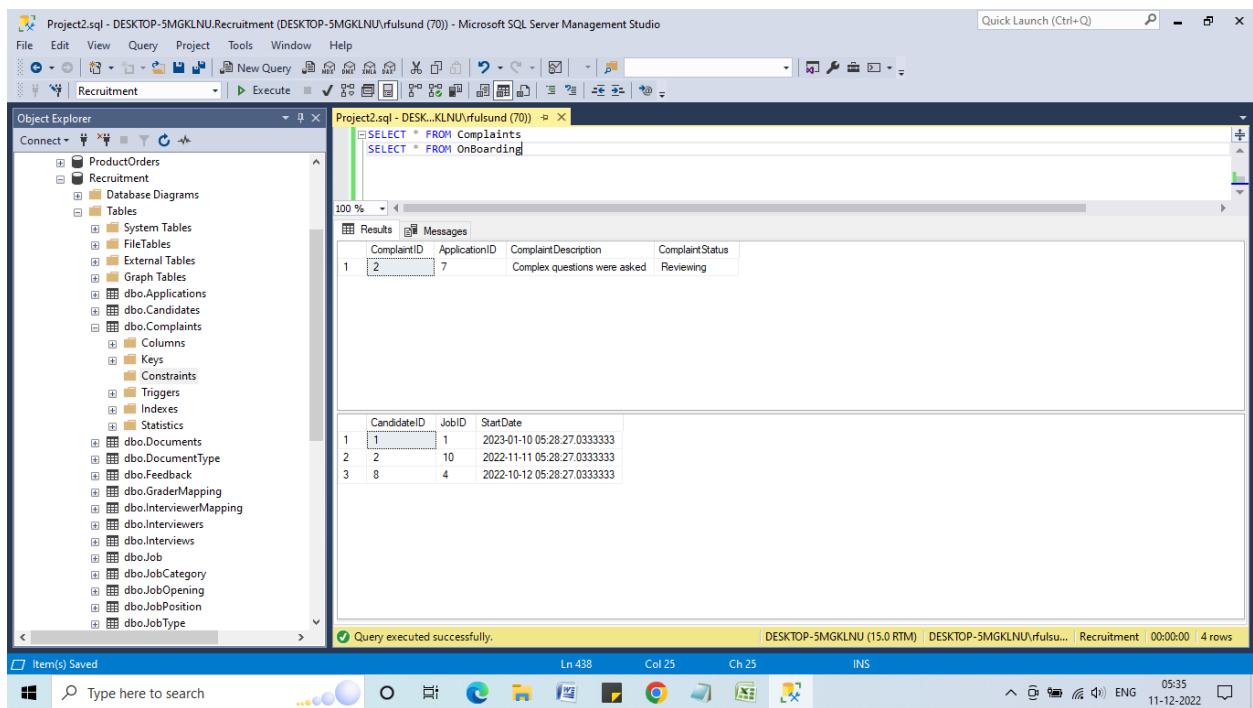
	CandidateID	JobID	StartDate
1	1	1	2023-01-10 05:28:27.033333
2	2	10	2022-11-11 05:28:27.033333
3	8	4	2022-10-12 05:28:27.033333

Query executed successfully.

DESKTOP-5MGKLN0 (15.0 RTM) DESKTOP-5MGKLN0\rfulsund... Recruitment 00:00:00 4 rows

Ln 438 Col 25 Ch 25 INS

Item(s) Saved Type here to search 05:35 11-12-2022



Section C : Testing

Views -

1. Write a view named SelectedCandidates that returns three columns: CandidateID, CandidateName and ApplicationID. Then, write a SELECT statement that returns all the columns in the view.
Displays all the candidates whose onboarding is going on.

```
CREATE VIEW onboardCandidates AS  
SELECT c.CandidateID, c.CandidateName, a.ApplicationID  
FROM Candidates c JOIN Applications a ON c.CandidateID = a.CandidateID WHERE  
a.ApplicationStatus = 'onboarding'
```

```
SELECT * FROM onboardCandidates
```

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane displays a database structure with several databases listed. In the center, a query window titled 'Project2.sql' contains the SQL code for creating a view and selecting from it. The results pane at the bottom shows a single row of data: CandidateID 1, CandidateName Pallavi Kale, and ApplicationID 10. A status bar at the bottom indicates the query was executed successfully.

CandidateID	CandidateName	ApplicationID
1	Pallavi Kale	10

2. Write a view named InterviewCleared that returns three columns: CandidateID, ApplicationID and InterviewID. Then, write a SELECT statement that returns all the columns in the view.
Displays all the candidates who cleared the interviews.

```

CREATE VIEW InterviewCleared AS
SELECT a.CandidateID, a.ApplicationID, i.InterviewID
FROM Applications a JOIN Interviews i ON a.ApplicationID = i.ApplicationID WHERE
i.Result = 'accepted'
SELECT * FROM InterviewCleared

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "Project2.sql - DESKTOP-5MGKLU.Recruitment (DESKTOP-5MGKLU\rfulsund (55)) - Microsoft SQL Server Management Studio". The Object Explorer sidebar shows a database named "Recruitment" containing various objects like Databases, Tables, and Views. The main query window contains the T-SQL code for creating the "InterviewCleared" view. The results window below shows a table with three columns: CandidateID, ApplicationID, and InterviewID, with 12 rows of data. The status bar at the bottom indicates "Query executed successfully." and "12 rows".

CandidateID	ApplicationID	InterviewID
1	1	2
2	10	8
3	1	6
4	1	1
5	10	8
6	1	2
7	2	10
8	8	4
9	2	10
10	8	4
11	8	4
12		13

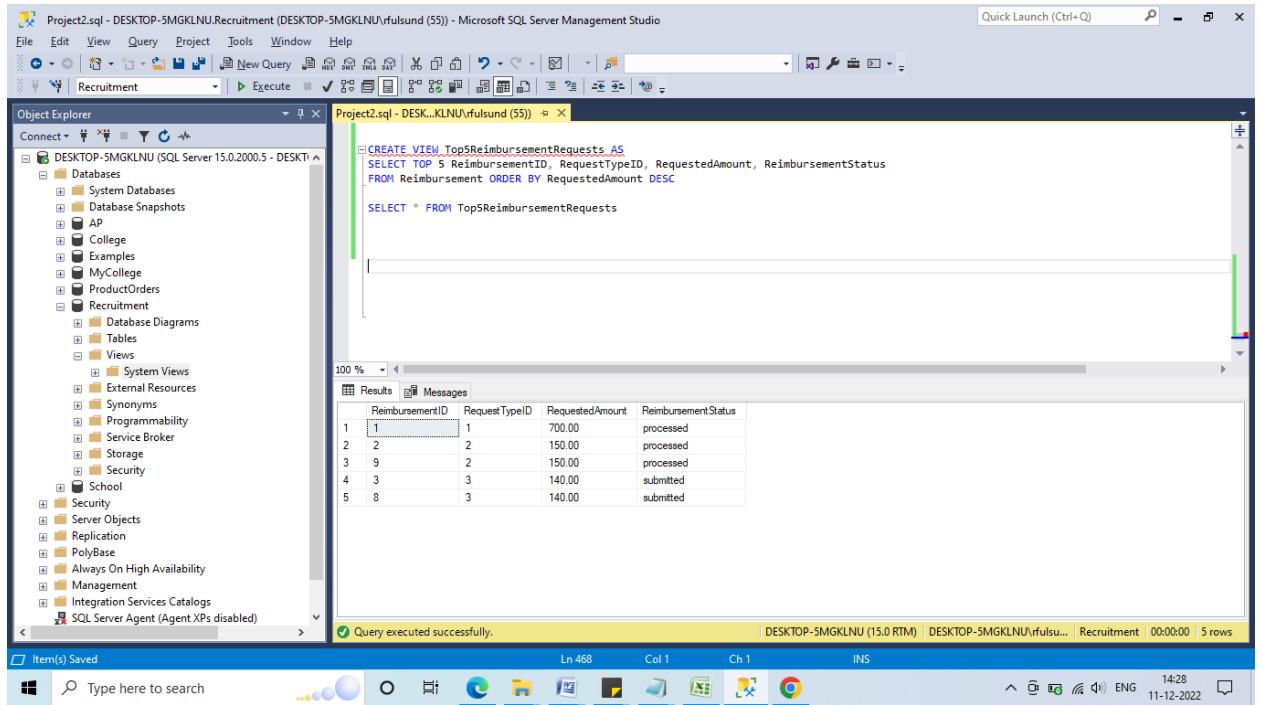
3. Write a view named Top5ReimbursementRequests that returns three columns: ReimbursementID, RequestTypeID, RequestedAmount, ReimbursementStatus. Then, write a SELECT statement that returns all the columns in the view.
Displays top 5 reimbursements requested and their status.

```

CREATE VIEW Top5ReimbursementRequests AS
SELECT TOP 5 ReimbursementID, RequestTypeID, RequestedAmount,
ReimbursementStatus
FROM Reimbursement ORDER BY RequestedAmount DESC

```

```
SELECT * FROM Top5ReimbursementRequests
```



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists databases, tables, and views under the 'Recruitment' database. The central pane displays a query window with the following code:

```
CREATE VIEW Top5ReimbursementRequests AS
SELECT TOP 5 ReimbursementID, RequestTypeID, RequestedAmount, ReimbursementStatus
FROM Reimbursement ORDER BY RequestedAmount DESC

SELECT * FROM Top5ReimbursementRequests
```

Below the code, a results grid shows the output of the query:

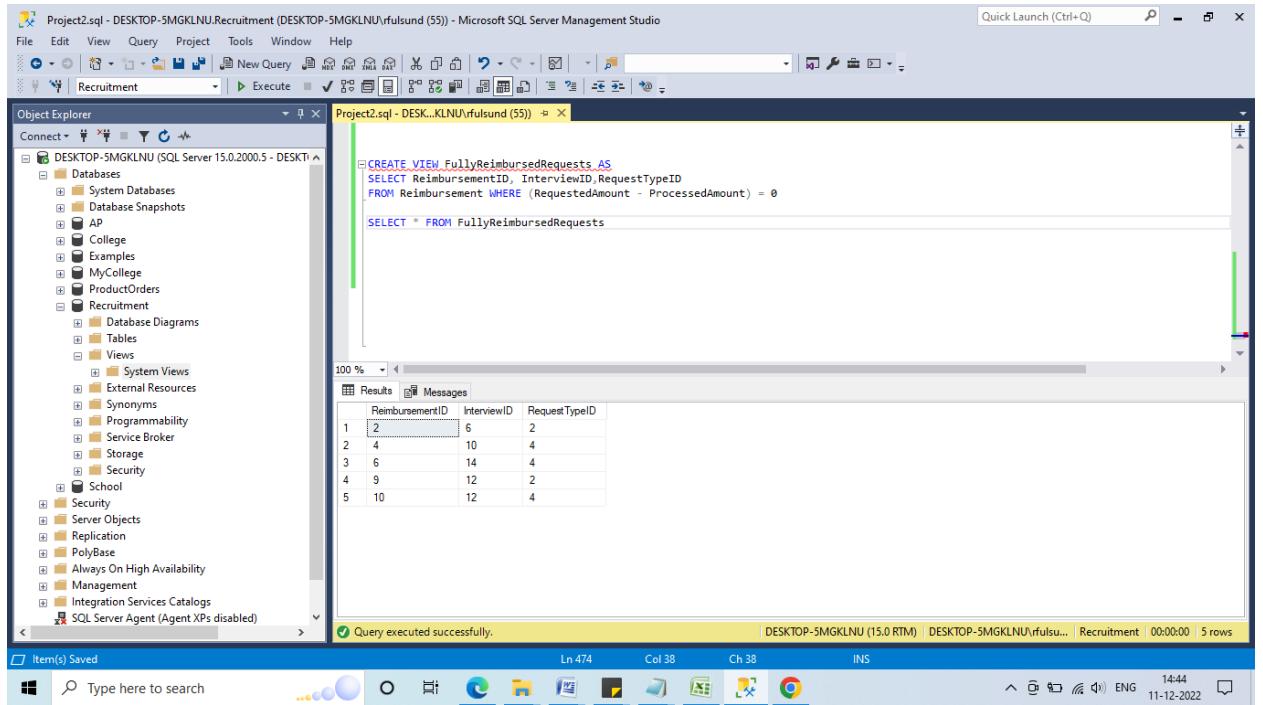
ReimbursementID	RequestTypeID	RequestedAmount	ReimbursementStatus
1	1	700.00	processed
2	2	150.00	processed
3	9	150.00	processed
4	3	140.00	submitted
5	8	140.00	submitted

The status bar at the bottom indicates "Query executed successfully." and shows system information like the computer name, user, and timestamp.

4. Write a view named FullyReimbursedRequests that returns three columns: ReimbursementID, InterviewID, RequestTypeID. Then, write a SELECT statement that returns all the columns in the view.

```
CREATE VIEW FullyReimbursedRequests AS
SELECT ReimbursementID, InterviewID, RequestTypeID
FROM Reimbursement WHERE (RequestedAmount - ProcessedAmount) = 0
```

```
SELECT * FROM FullyReimbursedRequests
```



The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'Recruitment' is selected. In the center pane, a query window titled 'Project2.sql - DESKTOP-5MGKLN0.Recruitment (DESKTOP-5MGKLN0\rfulsund (55)) - Microsoft SQL Server Management Studio' contains the following SQL code:

```
CREATE VIEW FullyReimbursedRequests AS
SELECT ReimbursementID, InterviewID, RequestTypeID
FROM Reimbursement WHERE (RequestedAmount - ProcessedAmount) = 0

SELECT * FROM FullyReimbursedRequests
```

The 'Results' tab is selected, displaying a table with five rows of data:

ReimbursementID	InterviewID	RequestTypeID
1	2	6
2	4	10
3	6	14
4	9	12
5	10	12

At the bottom of the results pane, a message says 'Query executed successfully.' The status bar at the bottom right shows 'DESKTOP-5MGKLN0 (15.0 RTM) | DESKTOP-5MGKLN0\rfulsu... | Recruitment | 00:00:00 | 5 rows'.

Stored Procedures -

1. Create a stored procedure named spCountApplication that accepts ApplicationStatus. The procedure should return count of applications for that status.

```
CREATE PROC spCountApplication
@Status varchar(25) = NULL,
@CountCandidates int = 0
AS
IF @Status IS NULL
SET @Status = ''
IF @CountCandidates = 0
SELECT @CountCandidates = COUNT(*) FROM Applications WHERE ApplicationStatus =
@Status

SELECT @Status AS ApplicationStatus, @CountCandidates AS NoOfApplications;
```

```
EXEC spCountApplication @Status = 'applied';
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a database named 'Recruitment'. The central pane displays a query window with the following code:

```
CREATE PROC spCountApplication
@Status varchar(25) = NULL,
@CountCandidates int = 0
AS
IF @Status IS NULL
SET @Status = ''
IF @CountCandidates =0
SELECT @CountCandidates = COUNT(*) FROM Applications WHERE ApplicationStatus= @Status
SELECT @Status AS ApplicationStatus, @CountCandidates AS NoOfApplications;
EXEC spCountApplication @Status = 'applied';
```

The results pane below shows a single row of data:

ApplicationStatus	NoOfApplications
applied	3

At the bottom of the screen, the taskbar shows various open applications and the system clock indicates it is 15:15 on 11-12-2022.

2. Create a stored procedure named spDateRange that accepts date range. The procedure should return count of applications in that date range.

```
CREATE PROC spDateRange
@DateMin varchar(50) = NULL,
@DateMax varchar(50) = NULL
AS
IF @DateMin IS NULL
THROW 50002, ' Date Minimum not provided ',1;
IF @DateMax IS NULL
THROW 50003, ' Date Maximum not provided ',1;
IF CONVERT(datetime2, @DateMin) > CONVERT(datetime2, @DateMax)
THROW 50004, ' Date Minimum greater than Date Maximum ',1;

SELECT ApplicationID, ApplicationStatus
FROM Applications a
WHERE a.SubmittedDate > @DateMin
    and a.SubmittedDate < @DateMax
ORDER BY a.SubmittedDate;
```

```
EXEC spDateRange @DateMin ='2022/12/01', @DateMax ='2022/12/10';
```

```

CREATE PROC spDateRange
    @DateMin varchar(50) = NULL,
    @DateMax varchar(50) = NULL
AS
BEGIN
    IF (@DateMin IS NULL)
        THROW 50002, 'Date Minimum not provided ',1;
    IF (@DateMax IS NULL)
        THROW 50003, 'Date Maximum not provided ',1;
    IF CONVERT(datetime2, @DateMin) > CONVERT(datetime2, @DateMax)
        THROW 50004, 'Date Minimum greater than Date Maximum ',1;

    SELECT ApplicationID, ApplicationStatus
    FROM Applications a
    WHERE a.SubmittedDate > @DateMin
        AND a.SubmittedDate < @DateMax
    ORDER BY a.SubmittedDate;

    EXEC spDateRange @DateMin ='2022/12/01', @DateMax ='2022/12/10';
END

```

The screenshot shows the SQL Server Management Studio interface. The Object Explorer pane on the left lists various database objects like Databases, Tables, and Views. The central pane displays the T-SQL code for the stored procedure spDateRange. Below the code is a results grid showing a list of applications with their IDs and statuses. The status column includes values like 'applied', 'interview 2', 'interview1', 'onboarding', 'offer extended', 'applied', 'interview 2', 'rejected', and 'applied'. At the bottom of the results grid, a message indicates 'Query executed successfully.'

3. Create a stored procedure named spInterviewsCountByType that accepts InterviewType. The procedure should return count of interviews for that interview type.

```

CREATE PROC spInterviewsCountByType
    @Type varchar(25) = NULL,
    @CountInterviews int = 0
AS
IF @Type IS NULL
SET @Type = ''
IF @CountInterviews = 0
SELECT @CountInterviews = COUNT(*) FROM Interviews WHERE InterviewType= @Type
SELECT @Type AS InterviewType, @CountInterviews AS NoOfInterviews;

```

```
EXEC spInterviewsCountByType @Type = 'online';
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a database named 'DESKTOP-5MGKLN0' with various objects like Databases, Tables, and Views. The central pane displays a query window titled 'Project2.sql - DESKTOP-5MGKLN0.Recruitment (DESKTOP-5MGKLN0\rfulsund (55)) - Microsoft SQL Server Management Studio'. The query is:

```
CREATE PROC spInterviewsCountByType
@Type varchar(25) = NULL,
@CountInterviews int = 0
AS
IF @Type IS NULL
SET @Type = ''
IF @CountInterviews = 0
SELECT @CountInterviews = COUNT(*) FROM Interviews WHERE InterviewType= @Type
SELECT @Type AS InterviewType, @CountInterviews AS NoOfInterviews;
EXEC spInterviewsCountByType @Type = 'online' parameter @CountInterviews int = 0
```

The results pane shows a single row of data:

InterviewType	NoOfInterviews
online	7

At the bottom, a message bar indicates 'Query executed successfully.'

4. Create a stored procedure named spCountInterviewsAndTests that accepts InterviewerID. The procedure should return count of interviews taken and count of tests graded by that interviewer.

```
CREATE PROC spCountInterviewsAndTests
@InterviewerID int = 0,
@CountInterviews int = 0,
@CountTests int = 0
AS
IF @InterviewerID = 0
SET @InterviewerID = 0

IF @CountInterviews = 0
SELECT @CountInterviews = COUNT(*) FROM Interviews JOIN InterviewerMapping
ON Interviews.InterviewerID = InterviewerMapping.InterviewerID
WHERE InterviewerID= @InterviewerID

IF @CountTests = 0
SELECT @CountTests = COUNT(*) FROM Tests JOIN GraderMapping ON Tests.TestID =
GraderMapping.TestID
WHERE InterviewerID= @InterviewerID
```

```

SELECT @InterviewerID AS InterviewerID,@CountInterviews AS
CountOfInterviewsTaken,@CountTests AS CountOfTestsGraded;

```

```

EXEC spCountInterviewsAndTests @InterviewerID = 2 ;

```

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer displays the database structure of 'DESKTOP-5MGKLN0' (SQL Server 15.0.2000.5). In the center, the 'Query Editor' window contains the following T-SQL code:

```

CREATE PROC spCountInterviewsAndTests
    @InterviewerID int = 0,
    @CountInterviews int = 0,
    @CountTests int = 0
    AS
    IF @InterviewerID = 0
        SET @InterviewerID = 0

    IF @CountInterviews = 0
    BEGIN
        SELECT @CountInterviews = COUNT(*) FROM Interviews JOIN InterviewerMapping ON Interviews.InterviewID = InterviewerMapping.InterviewID
        WHERE InterviewerID= @InterviewerID
    END

    IF @CountTests = 0
    BEGIN
        SELECT @CountTests = COUNT(*) FROM Tests JOIN GraderMapping ON Tests.TestID = GraderMapping.TestID
        WHERE InterviewerID= @InterviewerID
    END

    SELECT @InterviewerID AS InterviewerID,@CountInterviews AS CountOfInterviewsTaken,@CountTests AS CountOfTestsGraded;

    EXEC spCountInterviewsAndTests @InterviewerID = 2 ;

```

The 'Results' tab shows the output of the execution:

InterviewerID	CountOfInterviewsTaken	CountOfTestsGraded
1	2	4

Below the results, a message indicates: "Query executed successfully." The status bar at the bottom right shows the session details: DESKTOP-5MGKLN0 (15.0 RTM) | DESKTOP-5MGKLN0\rfulsund | Recruitment | 00:00:00 | 1 rows.

User Defined Functions -

1. Create a function named fnLatestInterviewDetails that returns application ID of most recent interview.

```
CREATE FUNCTION fnLatestInterviewDetails()
RETURNS INT
BEGIN
RETURN
(SELECT ApplicationID
FROM Interviews
ORDER BY InterviewEndTime DESC
OFFSET 0 ROWS
FETCH FIRST 1 ROWS ONLY);
END
```

```
SELECT a.CandidateID, CandidateName, ApplicationID, ApplicationStatus FROM
Applications a JOIN Candidates c
ON a.CandidateID = c.CandidateID WHERE ApplicationID =
dbo.fnLatestInterviewDetails();
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists databases, tables, and other objects under the 'Recruitment' database. The main pane displays two queries. The top query is the function definition:

```
CREATE FUNCTION fnLatestInterviewDetails()
RETURNS INT
BEGIN
RETURN
(SELECT ApplicationID
FROM Interviews
ORDER BY InterviewEndTime DESC
OFFSET 0 ROWS
FETCH FIRST 1 ROWS ONLY);
END
```

The bottom query is the select statement:

```
SELECT a.CandidateID, CandidateName, ApplicationID, ApplicationStatus FROM Applications a JOIN Candidates c ON a.CandidateID = c.CandidateID WHERE ApplicationID = dbo.fnLatestInterviewDetails();
```

The results pane shows a single row from the query:

CandidateID	CandidateName	ApplicationID	ApplicationStatus
1	Rachana Fulsunar	1	offer extended

At the bottom of the results pane, it says "Query executed successfully."

2. Create a function named fnDate1 that returns application details for all the applications submitted after a particular date.

```
CREATE FUNCTION fnDate1 (@DateValue smalldatetime = NULL)
RETURNS TABLE
RETURN
(SELECT ApplicationID, ApplicationStatus
FROM Applications a
WHERE a.SubmittedDate > CONVERT(datetime2,@DateValue));
SELECT * FROM dbo.fnDate1(2022/12/01);
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists databases, tables, and other objects under the 'Recruitment' database. The central pane contains the T-SQL code for the fnDate1 function. Below the code, the results pane displays a table with two columns: ApplicationID and ApplicationStatus. The data shows seven rows of application details. At the bottom of the results pane, a message indicates the query was executed successfully.

ApplicationID	ApplicationStatus
1	offer extended
2	interview1
3	applied
4	offer accepted
5	applied
6	interview 2
7	rejected

3. Create a function named fnInterviews that returns interview details for all the interviews taken by an interviewer.

```
CREATE FUNCTION fnInterviews (@InterviewerID int = 0)
RETURNS TABLE
RETURN
(SELECT Interviews.InterviewID, Interviews.ApplicationID, Interviews.Result
FROM Interviews JOIN InterviewerMapping ON Interviews.InterviewID =
InterviewerMapping.InterviewID
WHERE InterviewerMapping.InterviewerID = @InterviewerID);
```

```
SELECT * FROM dbo.fnInterviews(4);
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'Recruitment' is selected. In the center pane, a query window titled 'Project2.sql - DESKTOP-5MGKLN... (55) - Microsoft SQL Server Management Studio' contains the following code:

```
CREATE FUNCTION fnInterviews (@InterviewerID int = 0)
RETURNS TABLE
RETURN
(SELECT Interviews.InterviewID, Interviews.ApplicationID, Interviews.Result
FROM Interviews JOIN InterviewerMapping ON Interviews.InterviewID = InterviewerMapping.InterviewID
WHERE InterviewerMapping.InterviewerID = @InterviewerID);

SELECT * FROM dbo.fnInterviews(4);
```

The 'Results' tab shows the output of the query:

InterviewID	ApplicationID	Result
1	10	4
2	13	4
3	11	10

A message at the bottom of the results pane says 'Query executed successfully.' The status bar at the bottom right indicates the session ID is 1654 and the date is 11-12-2022.

4. Create a function named fnOffers that returns offer details held by that particular candidate

```
CREATE FUNCTION fnOffers (@CandidateID int = 0)
RETURNS TABLE
RETURN
(SELECT c.CandidateID, c.CandidateName, o.OfferStatus
FROM Candidates c JOIN Offers o ON c.CandidateID = o.CandidateID
WHERE c.CandidateID = @CandidateID);
```

```
SELECT * FROM dbo.fnOffers(4);
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists databases, including 'RECRUITMENT' which contains tables like AP, COLLEGE, EXAMPLES, MYCOLLEGE, PRODUCTORDERS, RECRUITMENT, and SCHOOL. The central pane displays a query window with the following code:

```
CREATE FUNCTION fnOffers (@CandidateID int = 0)
RETURNS TABLE
RETURN
(SELECT c.CandidateID, c.CandidateName, o.OfferStatus
FROM Candidates c JOIN Offers o ON c.CandidateID = o.CandidateID
WHERE c.CandidateID = @CandidateID);

SELECT * FROM dbo.fnOffers(4);
```

The Results tab shows the output of the query:

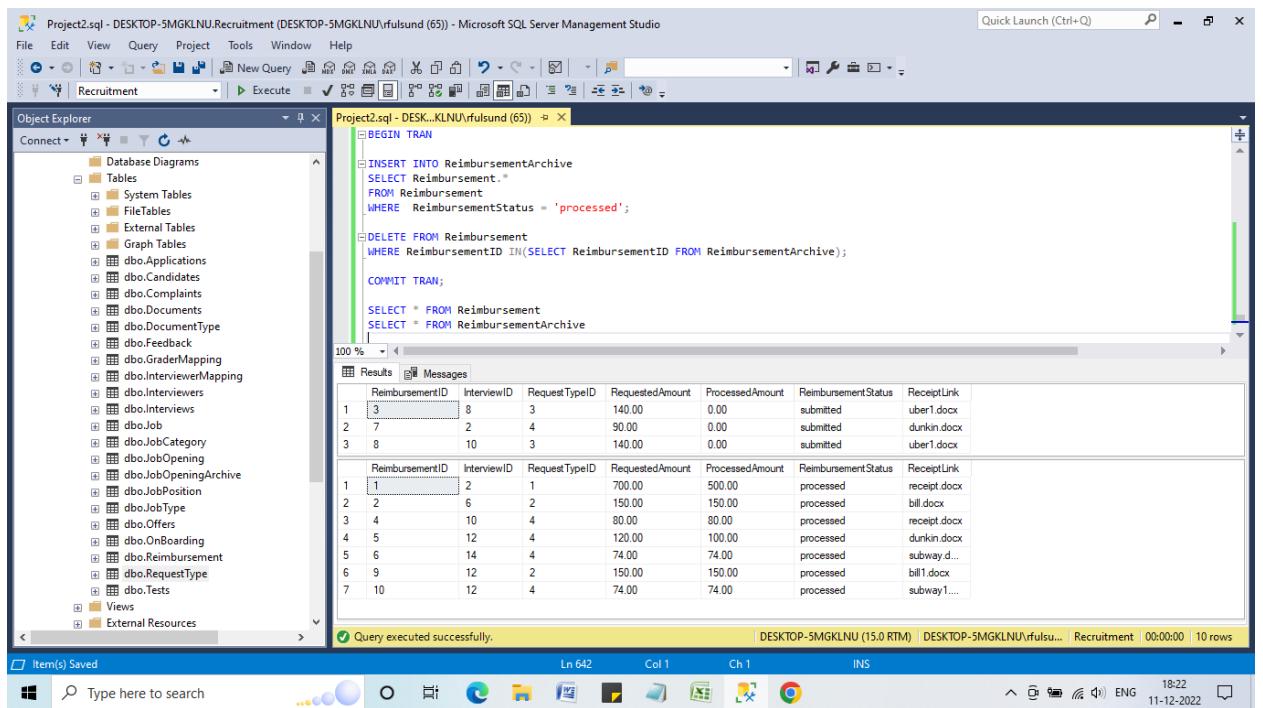
CandidateID	CandidateName	OfferStatus
1	4	Shubham Gangwal accepted

A status bar at the bottom indicates "Query executed successfully." and provides system information.

Transactions -

1. Delete all processed reimbursements from reimbursement table.

```
BEGIN TRAN  
INSERT INTO ReimbursementArchive  
SELECT Reimbursement.*  
FROM Reimbursement  
WHERE ReimbursementStatus = 'processed';  
  
DELETE FROM Reimbursement  
WHERE ReimbursementID IN(SELECT ReimbursementID FROM ReimbursementArchive);  
  
COMMIT TRAN;  
  
SELECT * FROM Reimbursement  
SELECT * FROM ReimbursementArchive
```



The screenshot shows the Microsoft SQL Server Management Studio interface. The left pane displays the Object Explorer with the 'Recruitment' database selected, showing various tables like InterviewerMapping, JobOpening, and ReimbursementArchive. The right pane contains a query window titled 'Project2.sql - DESKTOP-5MGKLU\rfulsund (65)'. It contains the following T-SQL code:

```
BEGIN TRAN  
INSERT INTO ReimbursementArchive  
SELECT Reimbursement.*  
FROM Reimbursement  
WHERE ReimbursementStatus = 'processed';  
  
DELETE FROM Reimbursement  
WHERE ReimbursementID IN(SELECT ReimbursementID FROM ReimbursementArchive);  
  
COMMIT TRAN;  
  
SELECT * FROM Reimbursement  
SELECT * FROM ReimbursementArchive
```

Below the code, the 'Results' tab is open, showing two sets of data. The first set is the result of the SELECT statement on the Reimbursement table, and the second is the result of the SELECT statement on the ReimbursementArchive table. Both sets show columns: ReimbursementID, InterviewerID, RequestTypeID, RequestedAmount, ProcessedAmount, ReimbursementStatus, and ReceiptLink. The data is as follows:

ReimbursementID	InterviewerID	RequestTypeID	RequestedAmount	ProcessedAmount	ReimbursementStatus	ReceiptLink
1	3	3	140.00	0.00	submitted	uber1.docx
2	7	4	90.00	0.00	submitted	dunkin.docx
3	8	3	140.00	0.00	submitted	uber1.docx

ReimbursementID	InterviewerID	RequestTypeID	RequestedAmount	ProcessedAmount	ReimbursementStatus	ReceiptLink	
1	2	1	700.00	500.00	processed	receipt.docx	
2	6	2	150.00	150.00	processed	bill.docx	
3	4	10	80.00	80.00	processed	receipt.docx	
4	5	12	4	120.00	100.00	processed	dunkin.docx
5	6	14	4	74.00	74.00	processed	subway.d...
6	9	12	2	150.00	150.00	processed	bill1.docx
7	10	12	4	74.00	74.00	processed	subway1....

At the bottom of the results pane, a message states "Query executed successfully." The status bar at the bottom of the screen shows the system information: DESKTOP-5MGKLU\rfulsu..., Recruitment, 00:00:00, 10 rows, and the date/time: 11-12-2022 18:22.

2. Update all JobMedium as online from Job table for which there is an opening.

```
BEGIN TRAN
```

```
UPDATE Job SET JobMedium = 'online' WHERE JobID IN (SELECT DISTINCT JobID FROM JobOpening)
```

```
COMMIT TRAN;
```

```
SELECT * FROM Job JOIN JobOpening ON Job.JobID = JobOpening.JobID
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "Project2.sql - DESKTOP-5MGKLNNU.Recruitment (DESKTOP-5MGKLNNU\rfulsund (65)) - Microsoft SQL Server Management Studio". The left pane is the Object Explorer, displaying the database structure with tables like Job, JobOpening, and Applications. The right pane contains a query window with the following T-SQL code:

```
BEGIN TRAN  
UPDATE Job SET JobMedium = 'online' WHERE JobID IN (SELECT DISTINCT JobID FROM JobOpening)  
COMMIT TRAN;  
  
SELECT * FROM Job JOIN JobOpening ON Job.JobID = JobOpening.JobID
```

Below the code, a results grid displays the output of the query. The columns are JobID, PositionID, JobTypeID, CategoryID, JobMedium, JobOpeningID, JobID, and NoOfPositions. The data is as follows:

JobID	PositionID	JobTypeID	CategoryID	JobMedium	JobOpeningID	JobID	NoOfPositions
1	3	1	3	online	1	3	2
2	5	2	1	online	2	5	0
3	2	4	2	online	3	2	3
4	8	10	4	3	4	8	1
5	4	3	4	online	5	4	0
6	1	7	1	2	6	1	3
7	9	6	1	2	7	9	2
8	7	11	3	6	online	8	7

The status bar at the bottom indicates "Query executed successfully".

3. Update application status to 'onboarding' whenever a new entry in onboarding table.

```
BEGIN TRAN
```

```
INSERT INTO OnBoarding VALUES (4,7,dateadd(w,-30 ,GETDATE()));
```

```
UPDATE Applications SET ApplicationStatus = 'onboarding' WHERE CandidateID = 4 and JobOpeningID = (SELECT JobOpeningID FROM JobOpening WHERE JobID=7)
```

```
COMMIT TRAN;
```

```
SELECT * FROM OnBoarding  
SELECT * FROM Applications
```

```

Project2.sql - DESKTOP-5MGKLNNU.Recruitment (DESKTOP-5MGKLNNU\rfulsund (53)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
Quick Launch (Ctrl+Q) 
Recruitment Execute 
Object Explorer
Project2.sql - DESKTOP-5MGKLNNU.Recruitment (DESKTOP-5MGKLNNU\rfulsund (53))
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Applications
dbo.Candidates
dbo.Complaints
dbo.Documents
dbo.DocumentType
dbo.Employees
dbo.Feedback
dbo.GradeMapping
dbo.InterviewerMapping
dbo.Interviewers
dbo.Interviews
dbo.Job
dbo.JobCategory
dbo.JobOpening
dbo.JobOpeningArchive
dbo.JobPosition
dbo.JobType
dbo.Offers
dbo.OnBoarding
dbo.Reimbursement
dbo.ReimbursementArchive
dbo.RequestType
dbo.Tests
Views
Results Messages
100 %
1 2 10 2022-11-11 05:28:27.0333333
2 4 7 2022-11-11 20:52:34.8166667
3 8 4 2022-10-12 05:28:27.0333333
4 10 5 2022-11-11 20:36:59.4233333
5 2 1 2022-12-07 02:24:02.0200000
6 3 7 2022-12-03 02:24:02.0200000
7 4 1 2022-12-10 02:24:02.0200000
8 5 8 2022-12-09 02:24:02.0200000
9 6 1 2022-12-08 02:24:02.0200000
10 7 9 2022-12-08 02:24:02.0200000
11 8 5 2022-12-06 02:24:02.0200000
12 9 4 2022-12-08 02:24:02.0200000
13 2 2 1 2022-12-11 02:24:02.0200000
14 3 7 4 2022-12-11 02:24:02.0200000
15 4 1 8 2022-12-11 02:24:02.0200000
16 5 3 7 2022-12-11 02:24:02.0200000
17 6 5 1 2022-12-11 02:24:02.0200000
18 7 1 9 2022-12-11 02:24:02.0200000
19 8 5 10 2022-12-11 02:24:02.0200000
20 9 8 4 2022-12-11 02:24:02.0200000
21 2 2 10 2022-11-11 05:28:27.0333333
22 3 4 7 2022-11-11 20:52:34.8166667
23 4 8 4 2022-10-12 05:28:27.0333333
24 5 10 5 2022-11-11 20:36:59.4233333
25 2 1 2022-12-07 02:24:02.0200000
26 3 7 2022-12-03 02:24:02.0200000
27 4 1 2022-12-10 02:24:02.0200000
28 5 8 2022-12-09 02:24:02.0200000
29 6 1 2022-12-08 02:24:02.0200000
30 7 9 2022-12-08 02:24:02.0200000
31 8 5 2022-12-06 02:24:02.0200000
32 9 4 2022-12-08 02:24:02.0200000
33 2 2 1 2022-12-11 02:24:02.0200000
34 3 7 4 2022-12-11 02:24:02.0200000
35 4 1 8 2022-12-11 02:24:02.0200000
36 5 3 7 2022-12-11 02:24:02.0200000
37 6 5 1 2022-12-11 02:24:02.0200000
38 7 1 9 2022-12-11 02:24:02.0200000
39 8 5 10 2022-12-11 02:24:02.0200000
40 9 8 4 2022-12-11 02:24:02.0200000
41 2 2 10 2022-11-11 05:28:27.0333333
42 3 4 7 2022-11-11 20:52:34.8166667
43 4 8 4 2022-10-12 05:28:27.0333333
44 5 10 5 2022-11-11 20:36:59.4233333
45 2 1 2022-12-07 02:24:02.0200000
46 3 7 2022-12-03 02:24:02.0200000
47 4 1 2022-12-10 02:24:02.0200000
48 5 8 2022-12-09 02:24:02.0200000
49 6 1 2022-12-08 02:24:02.0200000
50 7 9 2022-12-08 02:24:02.0200000
51 8 5 2022-12-06 02:24:02.0200000
52 9 4 2022-12-08 02:24:02.0200000
53 2 2 1 2022-12-11 02:24:02.0200000
54 3 7 4 2022-12-11 02:24:02.0200000
55 4 1 8 2022-12-11 02:24:02.0200000
56 5 3 7 2022-12-11 02:24:02.0200000
57 6 5 1 2022-12-11 02:24:02.0200000
58 7 1 9 2022-12-11 02:24:02.0200000
59 8 5 10 2022-12-11 02:24:02.0200000
60 9 8 4 2022-12-11 02:24:02.0200000
61 2 2 10 2022-11-11 05:28:27.0333333
62 3 4 7 2022-11-11 20:52:34.8166667
63 4 8 4 2022-10-12 05:28:27.0333333
64 5 10 5 2022-11-11 20:36:59.4233333
65 2 1 2022-12-07 02:24:02.0200000
66 3 7 2022-12-03 02:24:02.0200000
67 4 1 2022-12-10 02:24:02.0200000
68 5 8 2022-12-09 02:24:02.0200000
69 6 1 2022-12-08 02:24:02.0200000
70 7 9 2022-12-08 02:24:02.0200000
71 8 5 2022-12-06 02:24:02.0200000
72 9 4 2022-12-08 02:24:02.0200000
73 2 2 1 2022-12-11 02:24:02.0200000
74 3 7 4 2022-12-11 02:24:02.0200000
75 4 1 8 2022-12-11 02:24:02.0200000
76 5 3 7 2022-12-11 02:24:02.0200000
77 6 5 1 2022-12-11 02:24:02.0200000
78 7 1 9 2022-12-11 02:24:02.0200000
79 8 5 10 2022-12-11 02:24:02.0200000
80 9 8 4 2022-12-11 02:24:02.0200000
81 2 2 10 2022-11-11 05:28:27.0333333
82 3 4 7 2022-11-11 20:52:34.8166667
83 4 8 4 2022-10-12 05:28:27.0333333
84 5 10 5 2022-11-11 20:36:59.4233333
85 2 1 2022-12-07 02:24:02.0200000
86 3 7 2022-12-03 02:24:02.0200000
87 4 1 2022-12-10 02:24:02.0200000
88 5 8 2022-12-09 02:24:02.0200000
89 6 1 2022-12-08 02:24:02.0200000
90 7 9 2022-12-08 02:24:02.0200000
91 8 5 2022-12-06 02:24:02.0200000
92 9 4 2022-12-08 02:24:02.0200000
93 2 2 1 2022-12-11 02:24:02.0200000
94 3 7 4 2022-12-11 02:24:02.0200000
95 4 1 8 2022-12-11 02:24:02.0200000
96 5 3 7 2022-12-11 02:24:02.0200000
97 6 5 1 2022-12-11 02:24:02.0200000
98 7 1 9 2022-12-11 02:24:02.0200000
99 8 5 10 2022-12-11 02:24:02.0200000
100 9 8 4 2022-12-11 02:24:02.0200000

```

Query executed successfully.

- Update status of application to 'waiting' if candidate registers a compliant

BEGIN TRAN

INSERT INTO Complaints VALUES (5,'Very rude behavior from interviewer', 'Reviewing');

UPDATE Applications

SET ApplicationStatus = 'waiting'

WHERE ApplicationID = 5;

COMMIT TRAN;

SELECT * FROM Complaints

SELECT * FROM Applications

Project2.sql - DESKTOP-5MGKLN0.Recruitment (DESKTOP-5MGKLN0)\rfulsund (S3) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Recruitment | Execute | Quick Launch (Ctrl+Q)

Object Explorer

Project2.sql - DESKTOP-5MGKLN0\rfulsund (S3) - X

```

BEGIN TRAN
INSERT INTO Complaints VALUES (5,'Very rude behavior from interviewer', 'Reviewing');

UPDATE Applications
SET ApplicationStatus = 'waiting'
WHERE ApplicationID = 5;

COMMIT TRAN;

SELECT * FROM Complaints
SELECT * FROM Applications

```

Results Messages

ComplaintID	ApplicationID	ComplaintDescription	ComplaintStatus
1	2	Complex questions were asked	Reviewing
2	3	Very rude behavior from interviewer	Reviewing
3	4	Very rude behavior from interviewer	Reviewing

ApplicationID	JobOpeningID	CandidateID	SubmittedDate	LastUpdatedDate	ApplicationStatus
1	1	1	2022-12-08 02:24:02.020000	2022-12-11 02:24:02.020000	offer extended
2	2	2	2022-12-07 02:24:02.020000	2022-12-11 02:24:02.020000	onboarding
3	3	7	2022-12-03 02:24:02.020000	2022-12-11 02:24:02.020000	applied
4	4	1	2022-12-10 02:24:02.020000	2022-12-11 02:24:02.020000	offer accepted
5	5	3	2022-12-09 02:24:02.020000	2022-12-11 02:24:02.020000	waiting
6	6	5	2022-12-08 02:24:02.020000	2022-12-11 02:24:02.020000	waiting
7	7	1	2022-12-08 02:24:02.020000	2022-12-11 02:24:02.020000	rejected
8	8	5	2022-12-06 02:24:02.020000	2022-12-11 02:24:02.020000	onboarding

Query executed successfully.

DESKTOP-5MGKLN0 (15.0 RTM) | DESKTOP-5MGKLN0\rfulsu... | Recruitment | 00:00:00 | 13 rows

Item(s) Saved Type here to search Ln 808 Col 1 Ch 1 INS 20:57 ENG 11-12-2022

Scripts -

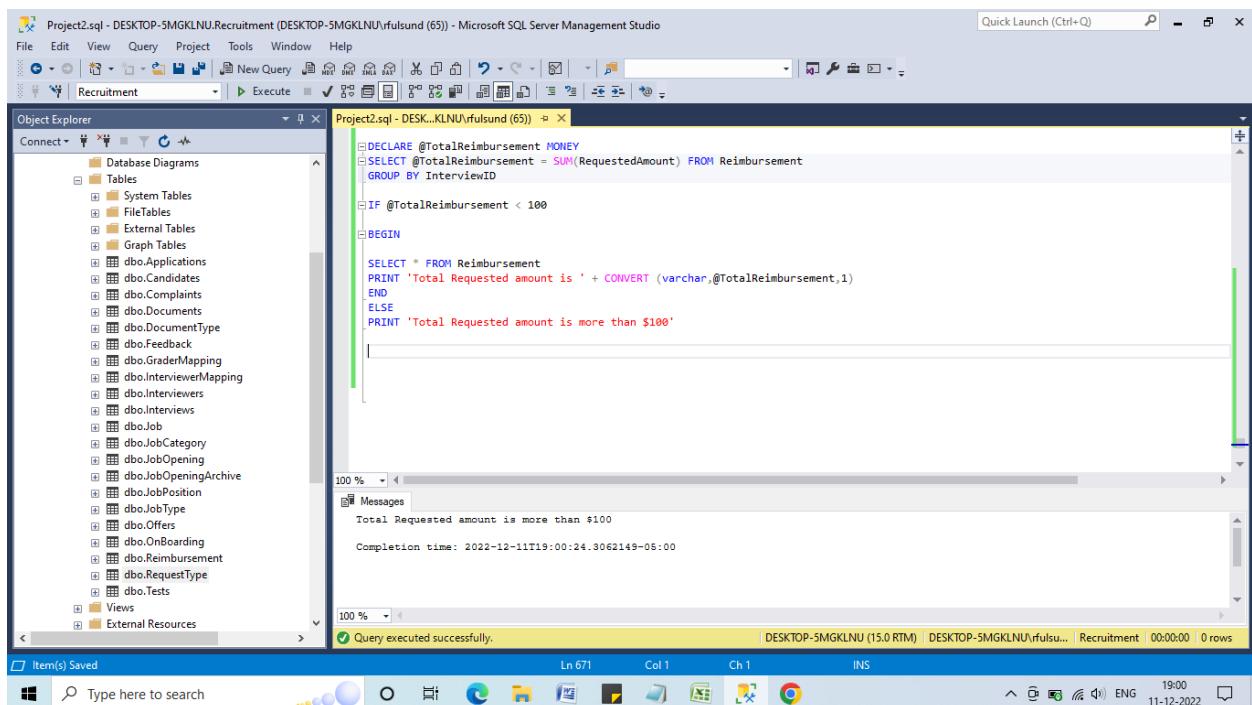
1. Script to check total requested reimbursed amount is greater than \$100 or not.

```
DECLARE @TotalReimbursement MONEY
SELECT @TotalReimbursement = SUM(RequestedAmount) FROM Reimbursement
GROUP BY InterviewID

IF @TotalReimbursement < 100

BEGIN

SELECT * FROM Reimbursement
PRINT 'Total Requested amount is ' + CONVERT (varchar,@TotalReimbursement,1)
END
ELSE
PRINT 'Total Requested amount is more than $100'
```



2. Script to display recent interview details taken by each interview

```
DECLARE @recent VARCHAR(4000)
IF EXISTS(select 1 FROM SYS.VIEWS WHERE NAME ='Latest_Interview')
BEGIN
```

```

SELECT*
FROM Latest_Interview;
END
ELSE
BEGIN
    SET @recent =
'CREATE VIEW Latest_Interview
AS
SELECT InterviewerID, MAX(InterviewEndTime) AS Latest_Interview
FROM Interviews JOIN InterviewerMapping ON Interviews.InterviewID = InterviewerMapping.InterviewerID
GROUP BY InterviewerID'
EXEC(@recent)
END

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Query Editor window contains a T-SQL script for creating a view named 'Latest_Interview'. The script uses dynamic SQL to check if the view exists, and if not, it creates it by selecting the latest interview end time for each interviewer from the 'Interviews' and 'InterviewerMapping' tables. The Results pane displays the output of the query, which is a table with two columns: 'InterviewerID' and 'Latest_Interview'. The data shows five rows with InterviewerIDs 1 through 5 and their corresponding latest interview end times.

InterviewerID	Latest_Interview
1	2022-12-04 01:40:09.800000
2	2022-12-04 01:40:09.800000
3	2022-12-05 17:40:09.800000
4	2022-11-30 17:40:09.800000
5	2022-12-03 17:40:09.800000

3. Script to display recent application submitted by candidate.

```

DECLARE @recentApplication VARCHAR(4000)
IF EXISTS(select 1 FROM SYS.VIEWS WHERE NAME ='Latest_Application')
BEGIN
    SELECT *
    FROM Latest_Application;
END
ELSE

```

```

BEGIN
    SET @recentApplication =
    'CREATE VIEW Latest_Application
    AS
        SELECT CandidateID, MAX(LastUpdatedDate) AS LatestApplicationDate
        FROM Applications
        GROUP BY CandidateID'
    EXEC(@recentApplication)
END

```

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane displays the database structure of 'RECRUITMENT'. In the center, the 'Project2.sql' file is open in the 'Script Editor' window. The script creates a view named 'Latest_Application' that selects the maximum 'LastUpdatedDate' for each 'CandidateID' from the 'Applications' table. The 'Results' tab at the bottom shows a grid of data with two columns: 'CandidateID' and 'LatestApplicationDate'. The data consists of 7 rows, each with a CandidateID from 1 to 10 and a corresponding LastUpdatedDate.

CandidateID	LatestApplicationDate
1	2022-12-11 02:24:02.020000
2	2022-12-11 02:24:02.020000
3	2022-12-11 02:24:02.020000
4	2022-12-11 02:24:02.020000
7	2022-12-11 02:24:02.020000
8	2022-12-11 02:24:02.020000
9	2022-12-11 02:24:02.020000
10	2022-12-11 02:24:02.020000

4. Script that uses dynamic SQL to return a single column that represents the number of rows in the first table in the current database

```

DECLARE @First_tab varchar(4000);
SET @First_tab =(SELECT TOP 1 name
FROM sys.tables
WHERE name <> 'dtproperties' OR name <> 'sysdiagrams'
ORDER BY name);

```

```
EXEC ('SELECT COUNT(*) AS CountOf' + @First_tab + ' FROM ' + @First_tab + ');
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various database objects like Databases, Tables, and Views. The central Query Editor window contains a dynamic SQL script. The script declares a variable @First_tab, sets it to the name of the first table in sys.tables, and then executes a query to count all rows in that table. The Results pane below shows the output: 'CountOfApplications' with a value of 10. A status bar at the bottom indicates the query was executed successfully.

```
DECLARE @First_tab varchar(4000);
SET @first_tab =(SELECT TOP 1 name
                 FROM sys.tables
                WHERE name <> 'dtproperties' OR name <> 'sysdiagrams'
                  ORDER BY name);
EXEC ('SELECT COUNT(*) AS CountOf' + @First_tab + ' FROM ' + @First_tab + ');
```

CountOfApplications
10

Query executed successfully.

Triggers -

1. Trigger for CREATE TABLE statement

```
CREATE TRIGGER CreateTable_DropTable
ON DATABASE
AFTER CREATE_TABLE, DROP_TABLE
AS
DECLARE @EventData xml;
SELECT @EventData = EVENTDATA();
DECLARE @EventType varchar(100);
SET @EventType =
@EventData.value('/EVENT_INSTANCE/EventType)[1]',
'verchar(100)');
IF @EventType = 'CREATE_TABLE'
PRINT 'A new table has been created.';
ELSE
PRINT 'A table has been dropped.';
```

PRINT CONVERT(varchar(max), @EventData);

```
CREATE TABLE Employees
```

```

( EmployeeID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
  EmployeeName varchar(50) NOT NULL,
  JoiningDate datetime2 NOT NULL
)

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure, including the 'Recruitment' database which contains tables like 'Employees', 'InterviewCleared', 'Latest_Interview', 'onboardingCandidates', and 'TopReimbursementRequests'. The central pane displays a T-SQL script for creating a trigger named 'CreateTable_DropTable' that logs table creation and drop events to the 'Messages' table. The 'Messages' pane shows the successful execution of the command, indicating a new table was created. The status bar at the bottom right shows the date and time as 11-12-2022 19:52.

```

CREATE TRIGGER CreateTable_DropTable
ON DATABASE
AFTER CREATE_TABLE, DROP_TABLE
AS
DECLARE @EventData xml;
SELECT @EventData = EVENTDATA();
DECLARE @EventType varchar(100);
SET @EventType =
@EventData.value('/EVENT_INSTANCE/EventType[1]',
'verchar(100)');
IF @EventType = 'CREATE_TABLE'
PRINT 'A new table has been created.';
ELSE
PRINT 'A table has been dropped.';
PRINT CONVERT(varchar(max), @EventData);

CREATE TABLE Employees
(
  EmployeeID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
  EmployeeName varchar(50) NOT NULL,
  JoiningDate datetime2 NOT NULL
)

```

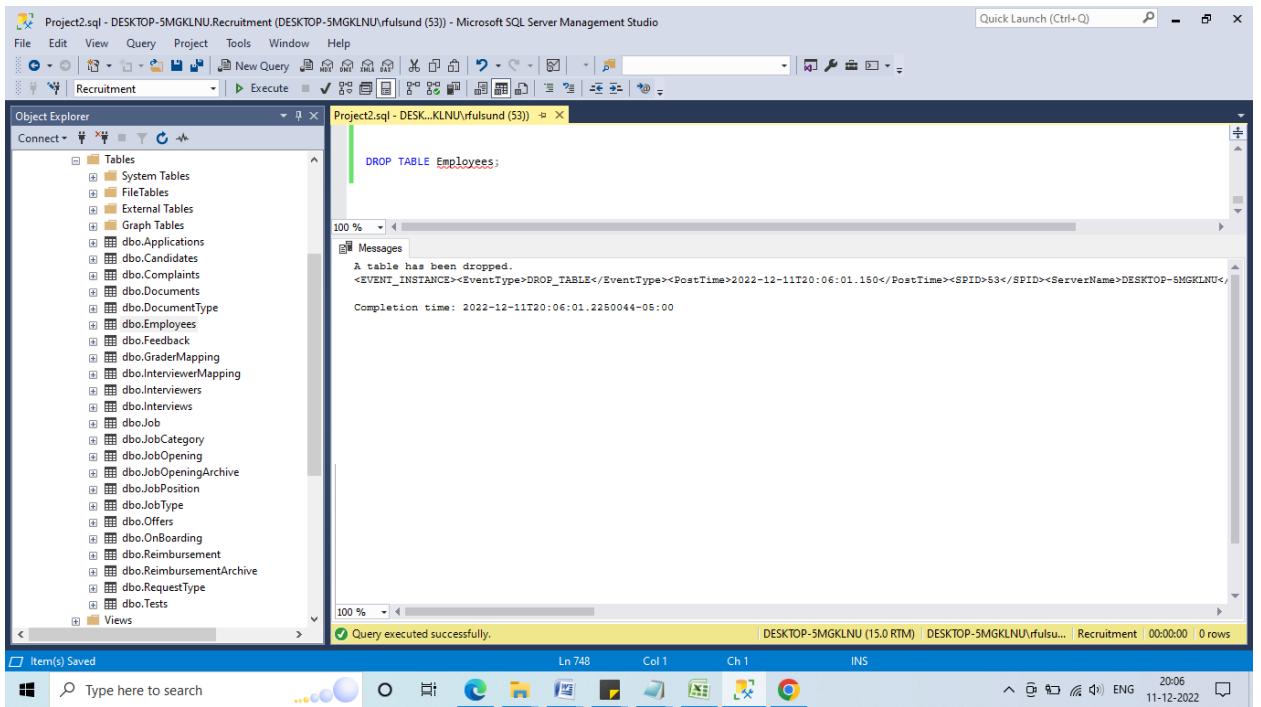
2. Trigger for DROP TABLE statement

```

CREATE TRIGGER CreateTable_DropTable
ON DATABASE
AFTER CREATE_TABLE, DROP_TABLE
AS
DECLARE @EventData xml;
SELECT @EventData = EVENTDATA();
DECLARE @EventType varchar(100);
SET @EventType =
@EventData.value('/EVENT_INSTANCE/EventType[1]',
'verchar(100)');
IF @EventType = 'CREATE_TABLE'
PRINT 'A new table has been created.';
ELSE
PRINT 'A table has been dropped.';
PRINT CONVERT(varchar(max), @EventData);

```

```
DROP TABLE Employees;
```



3. AFTER INSERT trigger which updates status of application to 'waiting' if candidate registers a compliant

```
CREATE TRIGGER Complaint_INSERT
ON Complaints
AFTER INSERT
AS
UPDATE Applications
SET ApplicationStatus = 'waiting'
WHERE ApplicationID IN (SELECT ApplicationID FROM Complaints WHERE ComplaintID =
(SELECT MAX(ComplaintID) FROM Complaints));

INSERT INTO Complaints VALUES
(6,'Very rude behavior from interviewer', 'Reviewing')
SELECT * FROM Applications
```

```

CREATE_TRIGGER Complaint_INSERT
ON Complaints
AFTER INSERT
AS
UPDATE Applications
SET ApplicationStatus = 'waiting'
WHERE ApplicationID IN (SELECT ApplicationID FROM Complaints WHERE ComplaintID = (SELECT MAX(ComplaintID) FROM Complaints));

INSERT INTO Complaints VALUES
(6, 'Very rude behavior from interviewer', 'Reviewing')

```

Messages

(1 row affected)

(1 row affected)

Completion time: 2022-12-11T20:15:05.7265108-05:00

Query executed successfully.

```

CREATE_TRIGGER Complaint_INSERT
ON Complaints
AFTER INSERT
AS
UPDATE Applications
SET ApplicationStatus = 'waiting'
WHERE ApplicationID IN (SELECT ApplicationID FROM Complaints WHERE ComplaintID = (SELECT MAX(ComplaintID) FROM Complaints));

INSERT INTO Complaints VALUES
(6, 'Very rude behavior from interviewer', 'Reviewing')

SELECT * FROM Applications

```

Results

ApplicationID	JobOpeningID	CandidateID	SubmittedDate	LastUpdatedDate	ApplicationStatus
1	1	1	2022-12-08 02:24:02.020000	2022-12-11 02:24:02.020000	offer extended
2	2	1	2022-12-07 02:24:02.020000	2022-12-11 02:24:02.020000	interview1
3	3	7	2022-12-03 02:24:02.020000	2022-12-11 02:24:02.020000	applied
4	4	1	2022-12-10 02:24:02.020000	2022-12-11 02:24:02.020000	offer accepted
5	5	3	2022-12-09 02:24:02.020000	2022-12-11 02:24:02.020000	applied
6	6	5	2022-12-08 02:24:02.020000	2022-12-11 02:24:02.020000	waiting
7	7	1	2022-12-08 02:24:02.020000	2022-12-11 02:24:02.020000	rejected
8	8	5	2022-12-06 02:24:02.020000	2022-12-11 02:24:02.020000	interview 2
9	9	8	2022-12-08 02:24:02.020000	2022-12-11 02:24:02.020000	applied
10	10	7	2022-12-07 02:24:02.020000	2022-12-11 02:24:02.020000	onboarding

Messages

(10 rows)

Query executed successfully.

- Trigger to update application status to 'onboarding' whenever a new entry in onboarding table.

**CREATE TRIGGER Onboarding_INSERT
ON Onboarding**

AFTER INSERT

AS

UPDATE Applications

SET ApplicationStatus = 'onboarding'

WHERE JobOpeningID IN (SELECT JobOpeningID FROM JobOpening WHERE JobID = (SELECT JobID FROM inserted));

INSERT INTO OnBoarding VALUES

(10,5,dateadd(w,-30 ,GETDATE()))

SELECT * FROM Applications

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, there is a 'Tables' node under 'Recruitment' which contains various tables like Applications, Candidates, Complaints, etc. In the main pane, a script named 'Project2.sql' is open. The script contains a trigger definition:

```
CREATE TRIGGER Onboarding_INSERT
ON OnBoarding
AFTER INSERT
AS
UPDATE Applications
SET ApplicationStatus = 'onboarding'
WHERE JobOpeningID IN (SELECT JobOpeningID FROM JobOpening WHERE JobID = (SELECT JobID FROM inserted));

INSERT INTO OnBoarding VALUES
(10,5,dateadd(w,-30 ,GETDATE()));

SELECT * FROM Applications
```

Below the script, the 'Results' tab shows the output of the 'SELECT * FROM Applications' query. The table has columns: ApplicationID, JobOpeningID, CandidateID, SubmittedDate, LastUpdatedDate, ApplicationStatus. The data is as follows:

ApplicationID	JobOpeningID	CandidateID	SubmittedDate	LastUpdatedDate	ApplicationStatus
1	1	1	2022-12-08 02:24:02.020000	2022-12-11 02:24:02.020000	offer extended
2	2	1	2022-12-07 02:24:02.020000	2022-12-11 02:24:02.020000	onboarding
3	3	4	2022-12-03 02:24:02.020000	2022-12-11 02:24:02.020000	applied
4	4	8	2022-12-10 02:24:02.020000	2022-12-11 02:24:02.020000	offer accepted
5	5	3	2022-12-09 02:24:02.020000	2022-12-11 02:24:02.020000	applied
6	6	5	2022-12-08 02:24:02.020000	2022-12-11 02:24:02.020000	waiting
7	7	1	2022-12-08 02:24:02.020000	2022-12-11 02:24:02.020000	rejected
8	8	5	2022-12-06 02:24:02.020000	2022-12-11 02:24:02.020000	interview 2
9	9	8	2022-12-08 02:24:02.020000	2022-12-11 02:24:02.020000	applied
10	10	7	2022-12-07 02:24:02.020000	2022-12-11 02:24:02.020000	onboarding

At the bottom of the results pane, it says 'Query executed successfully.'

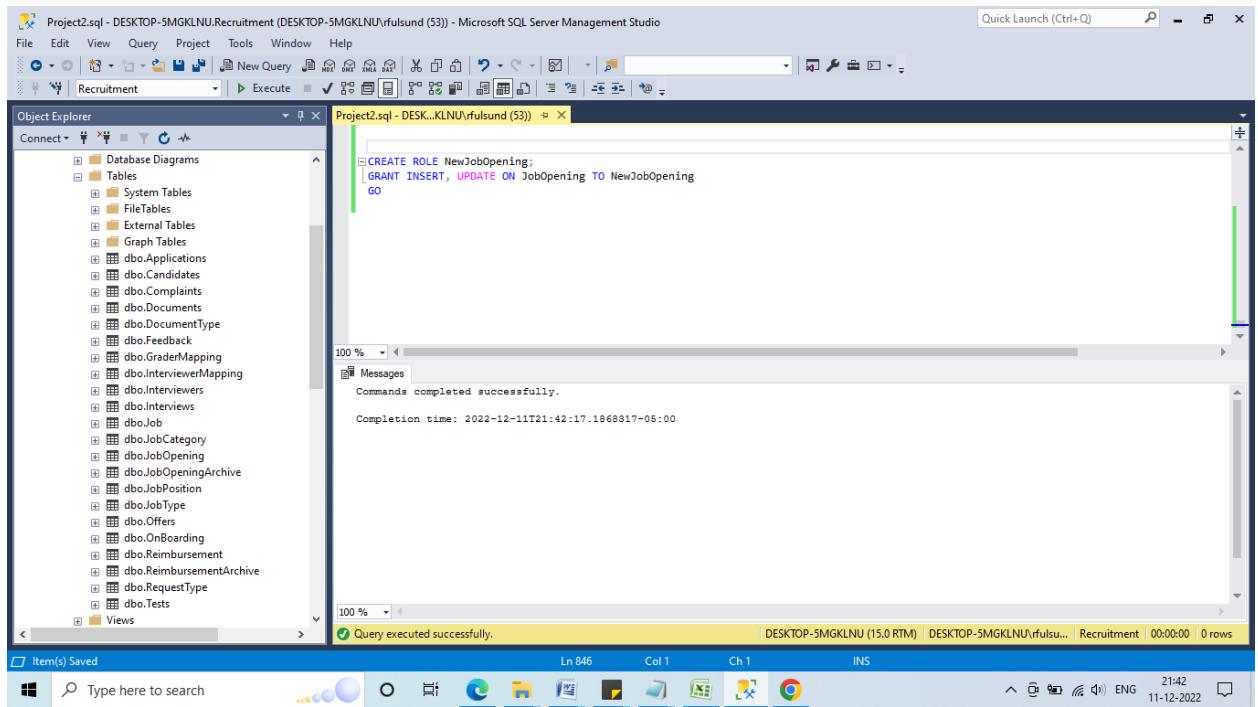
Security Roles -

1. Role 1 :

CREATE ROLE NewJobOpening;

GRANT INSERT, UPDATE ON JobOpening TO NewJobOpening

GO



The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various database objects like Database Diagrams, Tables, System Tables, etc., under the 'Recruitment' database. The central pane displays a SQL script window titled 'Project2.sql - DESKTOP-5MGKLN0.Recruitment (DESKTOP-5MGKLN0\rfulsund (53)) - Microsoft SQL Server Management Studio'. The script contains the following T-SQL code:

```
CREATE ROLE NewJobOpening;
GRANT INSERT, UPDATE ON JobOpening TO NewJobOpening
GO
```

The 'Messages' pane below the script shows the output of the command:

```
Commands completed successfully.
Completion time: 2022-12-11T21:42:17.1868817-05:00
```

The status bar at the bottom indicates 'Query executed successfully.' and provides system information.

2. Role 2 :

```
CREATE ROLE CandidateEntry;
GRANT UPDATE ON Candidates TO CandidateEntry
GO
CREATE LOGIN Rachana WITH PASSWORD = 'r@ch@ana',
DEFAULT_DATABASE = Recruitment;
CREATE USER RACHANA FOR LOGIN Rachana;
ALTER ROLE CandidateEntry ADD MEMBER Rachana;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists various database objects like Database Diagrams, Tables, System Tables, etc. The central pane displays a SQL script for creating a role and a login:

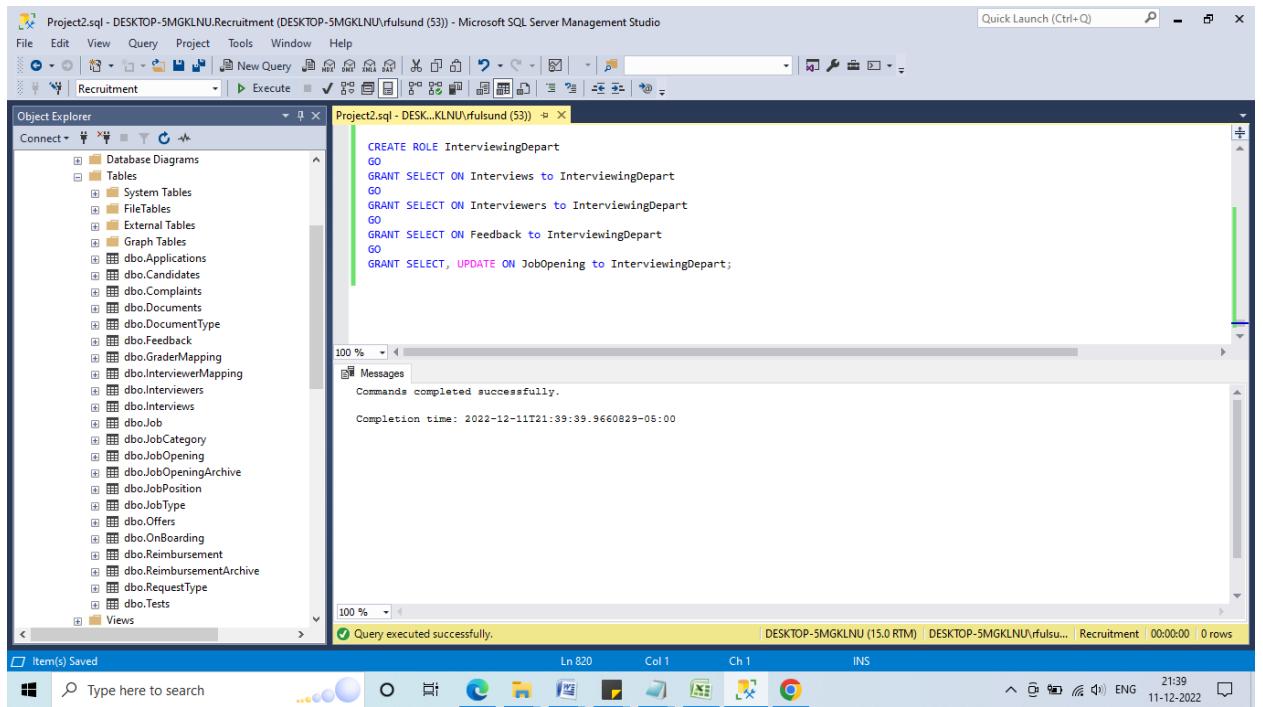
```
CREATE ROLE CandidateEntry;
GRANT UPDATE ON Candidates TO CandidateEntry
GO
CREATE LOGIN Rachana WITH PASSWORD = 'r@ch@na',
    DEFAULT_DATABASE = Recruitment;
CREATE USER RACHANA FOR LOGIN Rachana;
ALTER ROLE CandidateEntry ADD MEMBER Rachana;
```

The status bar at the bottom indicates "Query executed successfully." and "Completion time: 2022-12-11T21:41:33.1850265-05:00".

3. Role 3 :

```
CREATE ROLE InterviewingDepart
GO
GRANT SELECT ON Interviews to InterviewingDepart
GO
GRANT SELECT ON Interviewers to InterviewingDepart
GO
GRANT SELECT ON Feedback to InterviewingDepart
GO
```

GRANT SELECT, UPDATE ON JobOpening to InterviewingDepart;



```
CREATE ROLE InterviewingDepart
GO
GRANT SELECT ON Interviews to InterviewingDepart
GO
GRANT SELECT ON Interviewers to InterviewingDepart
GO
GRANT SELECT ON Feedback to InterviewingDepart
GO
GRANT SELECT, UPDATE ON JobOpening to InterviewingDepart;
```

Messages
Commands completed successfully.
Completion time: 2022-12-11T21:39:39.9660829-05:00

4. Role 4 :

```
CREATE ROLE ComplaintDept
GO
GRANT INSERT,UPDATE,DELETE on Complaints to ComplaintDept
GO
GRANT SELECT on DATABASE:: Recruitment to ComplaintDept;
```

Project2.sql - DESKTOP-5MGKLN0.Recruitment (DESKTOP-5MGKLN0\rfulsund (53)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Quick Launch (Ctrl+Q)

Recruitment Execute

Object Explorer

Project2.sql - DESKTOP-5MGKLN0\fulsund (53)

```
CREATE ROLE ComplaintDept
GO
GRANT INSERT,UPDATE,DELETE on Complaints to ComplaintDept
GO
GRANT SELECT on DATABASE:: Recruitment to ComplaintDept;
```

100 % Messages
Commands completed successfully.
Completion time: 2022-12-11T21:37:53.2507113-05:00

100 % Output
Query executed successfully.

DESKTOP-5MGKLN0 (15.0 RTM) | DESKTOP-5MGKLN0\fulsund (53) | Recruitment | 00:00:00 | 0 rows

Ready Type here to search Ln 819 Col 1 Ch 1 INS

21:37 ENG 11-12-2022

Section D : Conclusion

In this project, we designed database schema, ER relationship diagram for recruitment department. We designed tables and columns as per the specifications given along with the constraints.

We used views, stored procedures, user defined functions, transactions, triggers, security roles on tables in the database.