CSE 581 – Intro To Database Management Systems

Project 2

Rachana Mohan Fulsundar

SUID : 512373928

12/11/2022

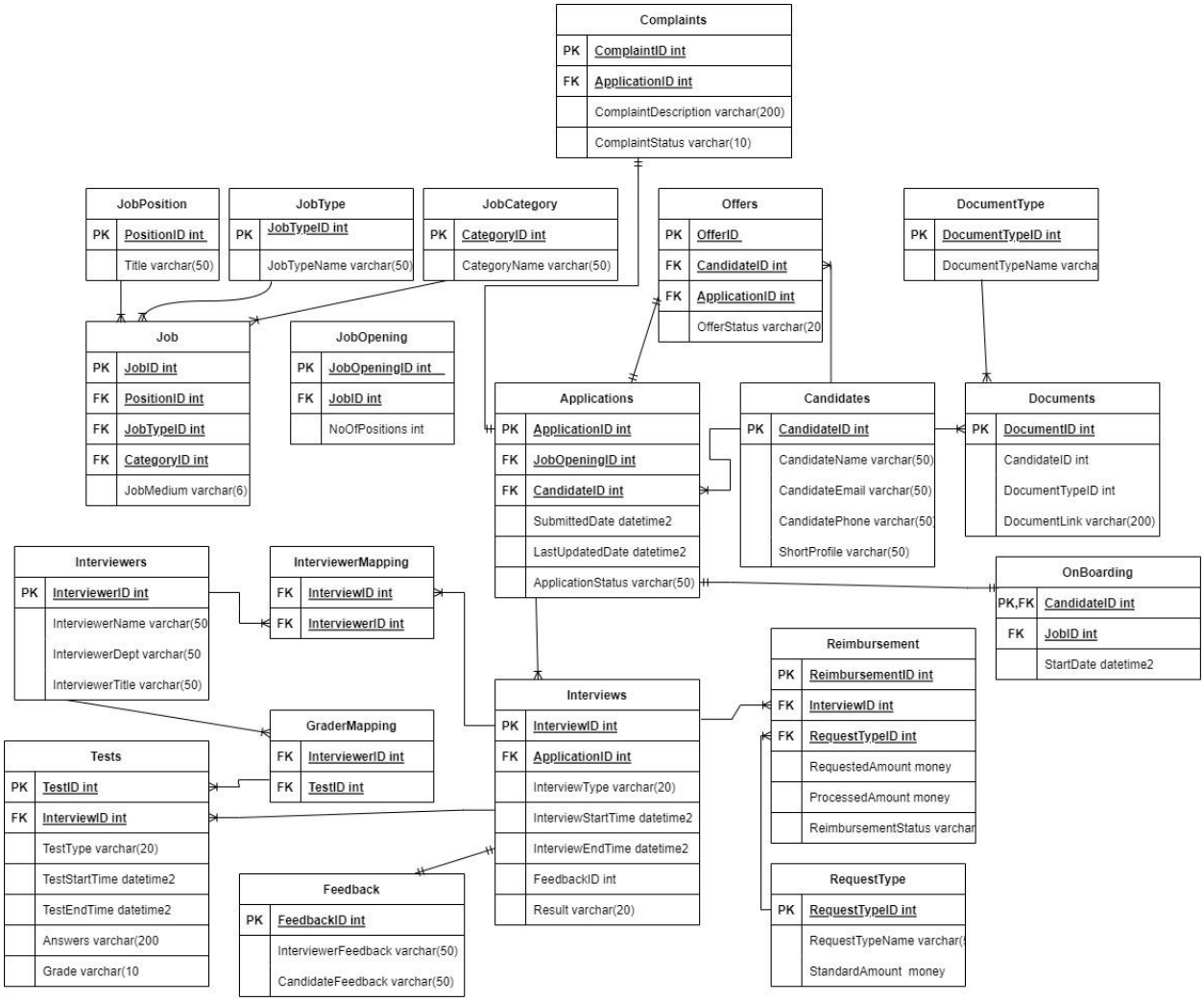
Abstract -

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

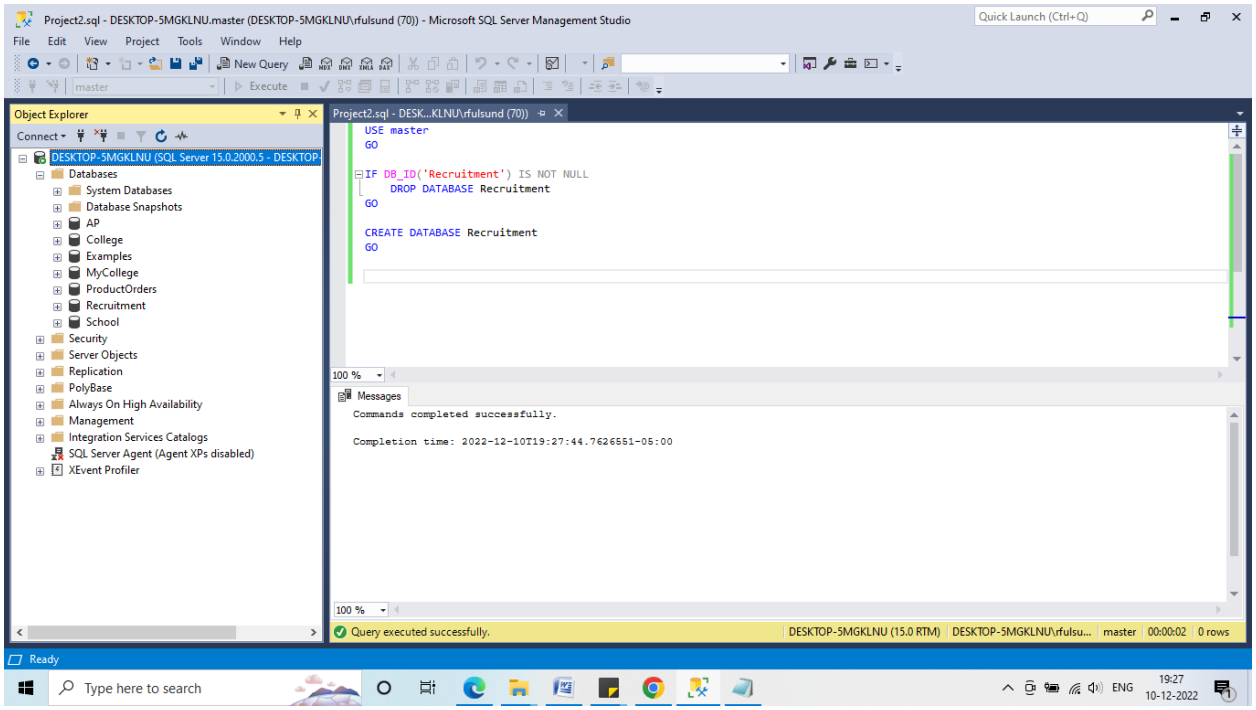
Database Schema -



Section B : Implementation

Creating Database -

```
USE master
GO

IF DB_ID('Recruitment') IS NOT NULL
   DROP DATABASE Recruitment
GO

CREATE DATABASE Recruitment
GO
```



Creating Tables -

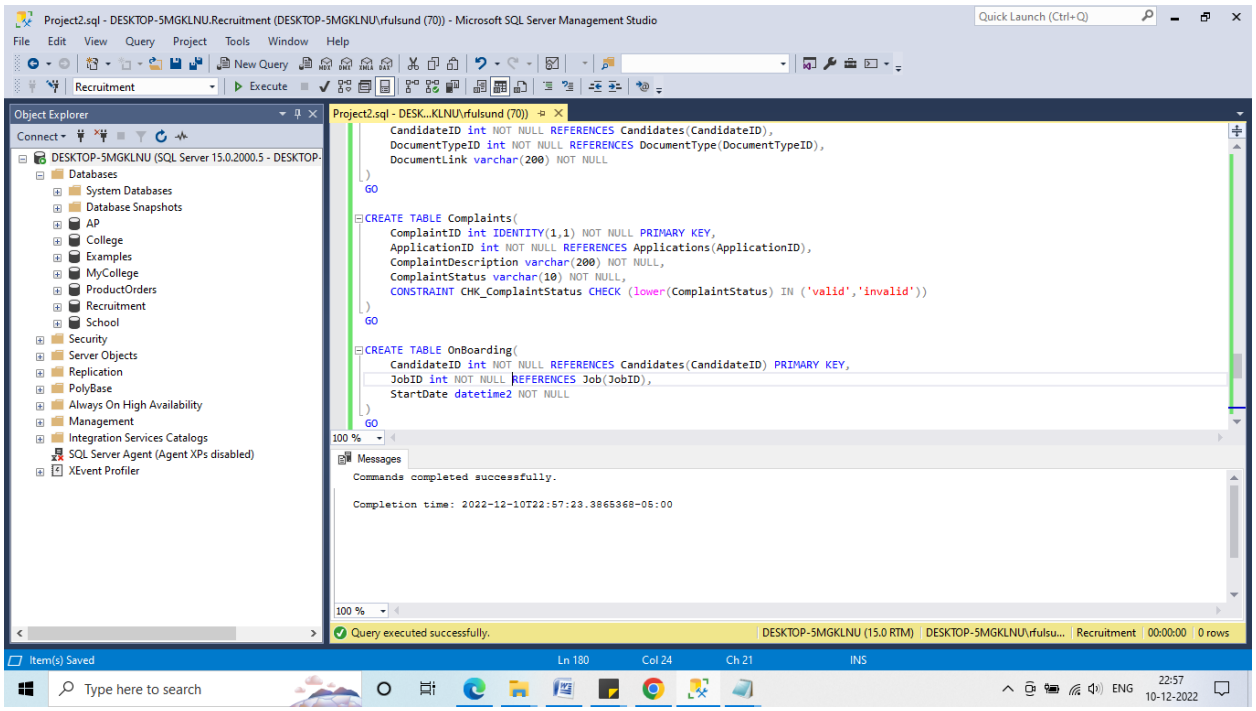
```
USE Recruitment
GO

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
```



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'),
```

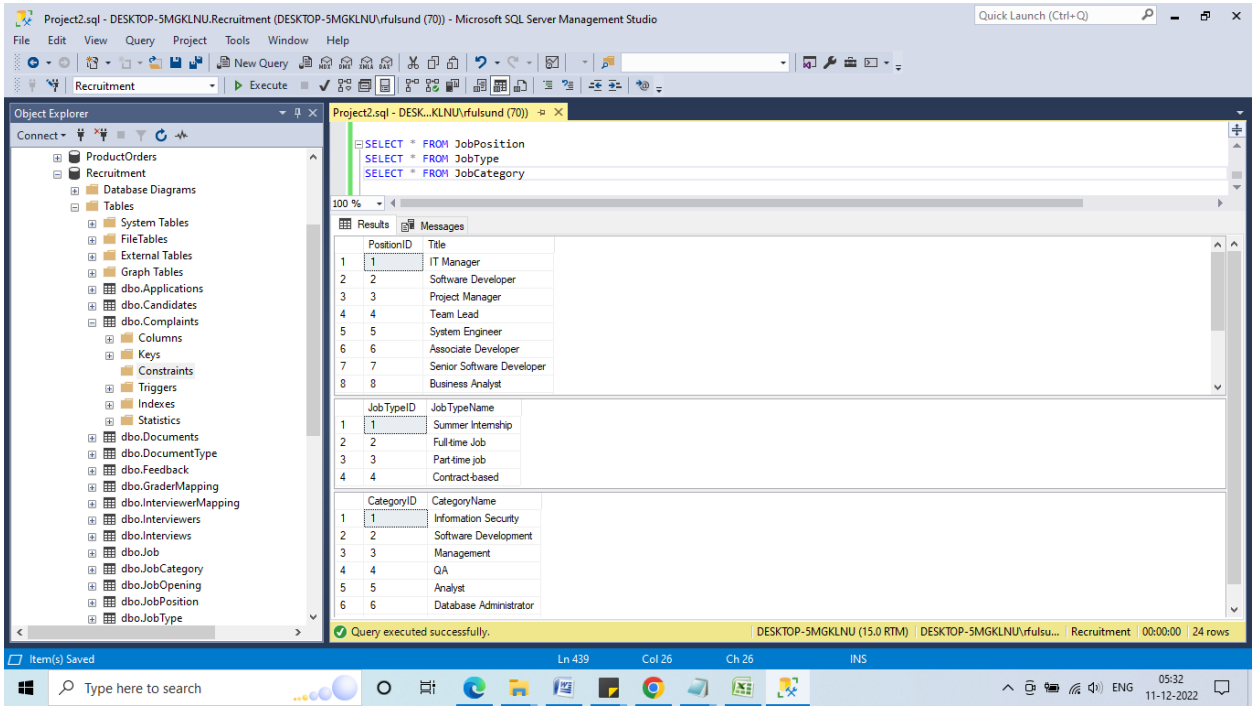
```
(8,'online',dateadd(hh, -10, dateadd(w,-9 ,GETDATE())),dateadd(hh, -8, dateadd(w,-9
,GETDATE())),3,'accepted'),
(6,'online',dateadd(hh, -9, dateadd(w,-8 ,GETDATE())),dateadd(hh, -7, dateadd(w,-8
,GETDATE())),4,'accepted'),
(1,'onsite',dateadd(hh, -12, dateadd(w,-5 ,GETDATE())),dateadd(hh, -10, dateadd(w,-5
,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),
```

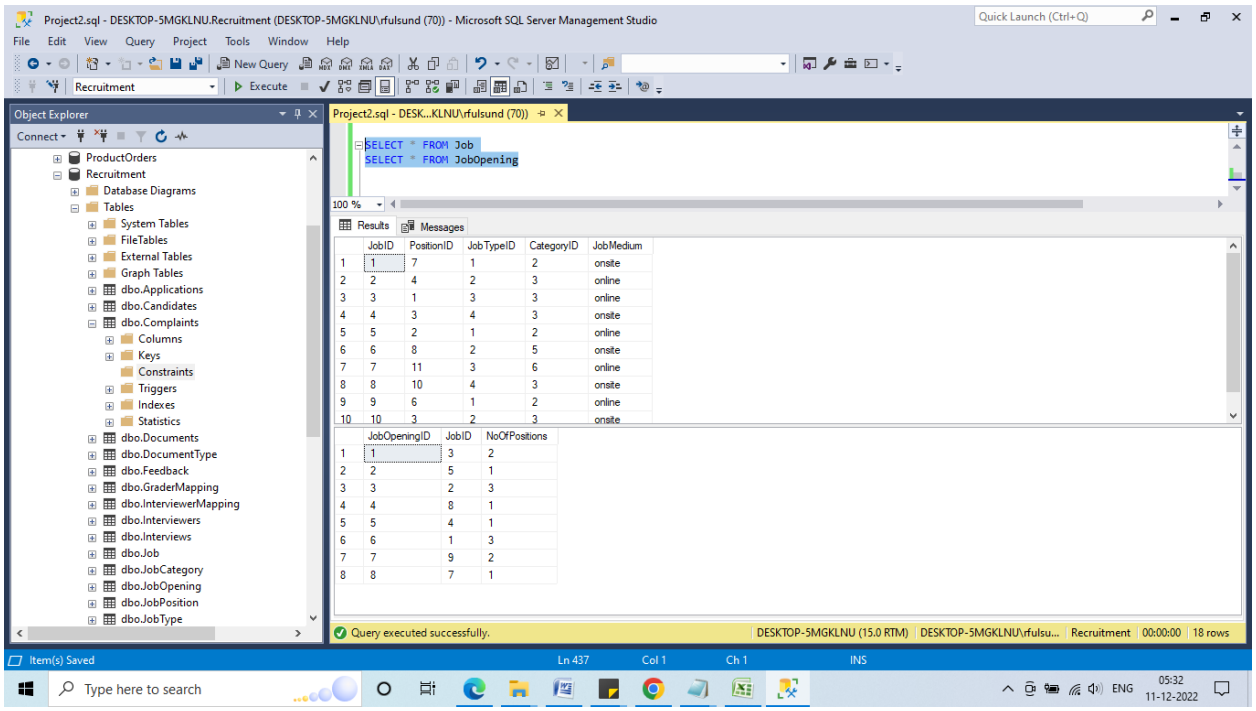
```
(7,3),
(13,4),
(11,4);

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),
```

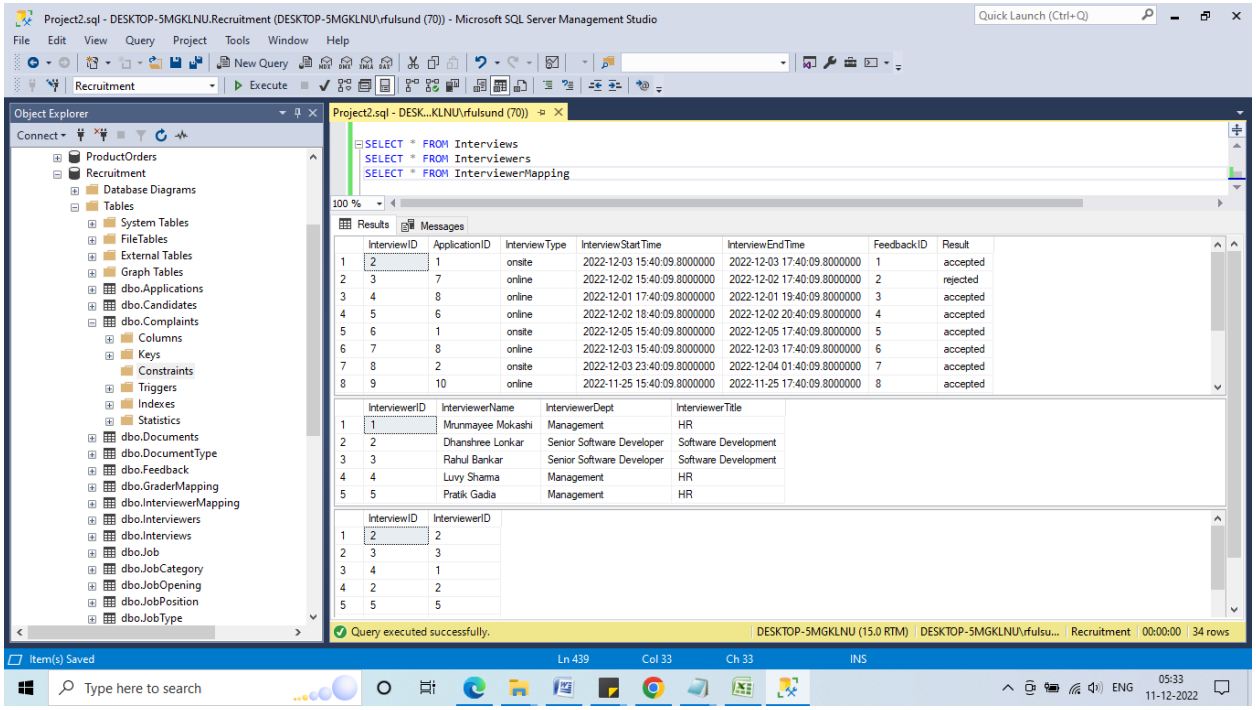
```
(3,5),
(4,2),
(1,1),
(3,3),
(2,7),
(1,9),
(1,12),
(3,8),
(1,11),
(2,14),
(4,10),
(3,15),
(2,14),
(1,13),
(2,16);


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'),
```

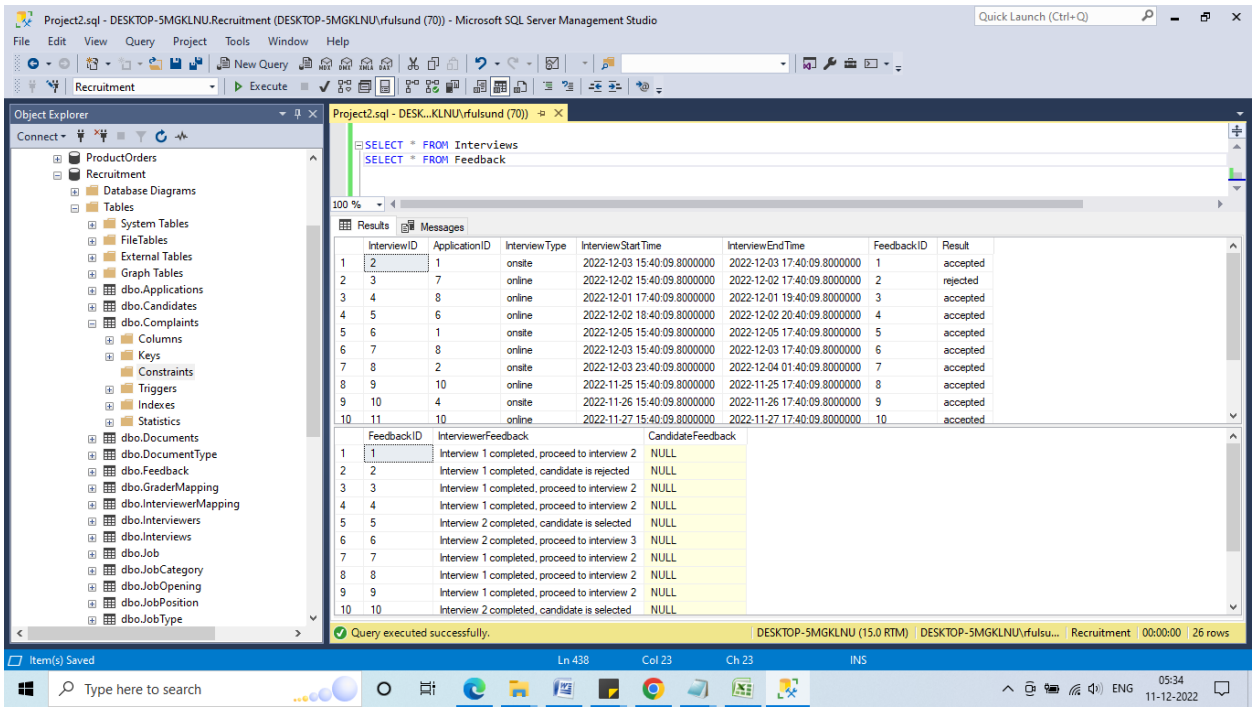
```
(5,1,'res.pdf'),
(1,2,'ref2.pdf'),
(2,1,'resume2.pdf'),
(7,1,'res2.pdf'),
(2,3,'CoverPage.pdf'),
(10,2,'ref2.pdf'),
(9,1,'resume2.pdf');


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()));
```

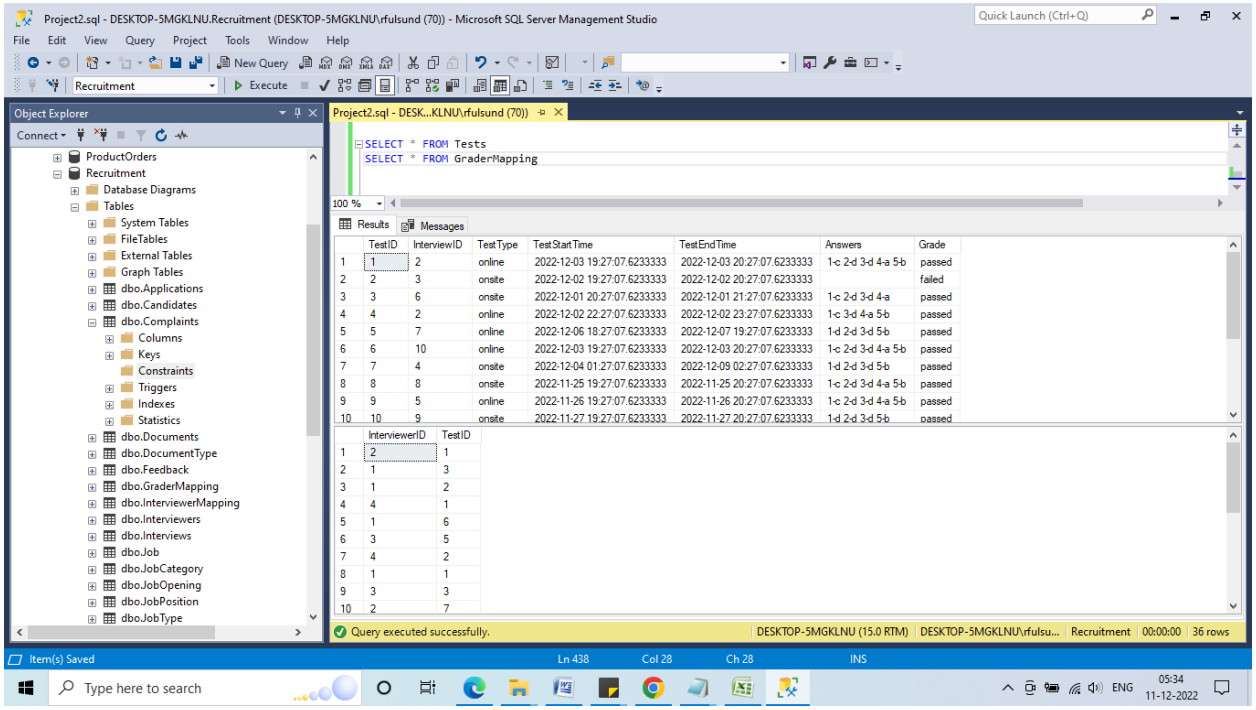
Displaying data from all the tables -

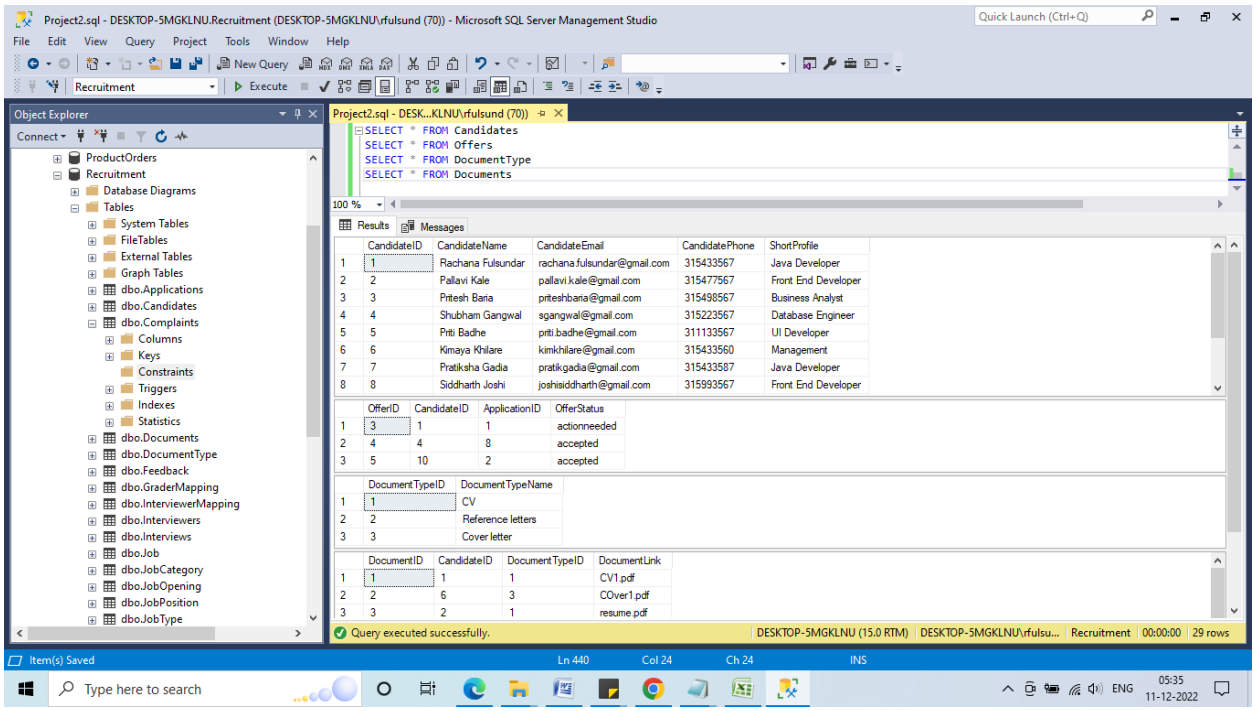


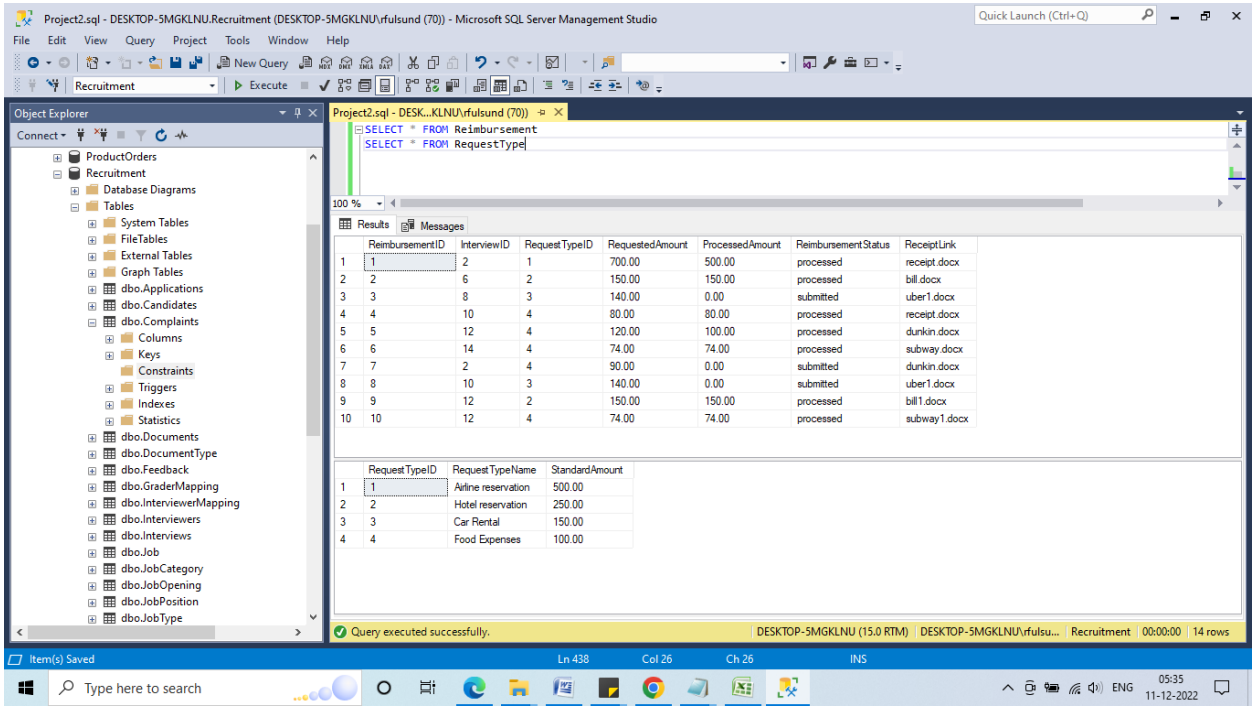

Project2.sql - DESKTOP-5MGKLNU.Recruitment (DESKTOP-5MGKLNU\rfulsund (70)) - Microsoft SQL Server Management Studio
SELECT * FROM Job
SELECT * FROM JobOpening
Query executed successfully.

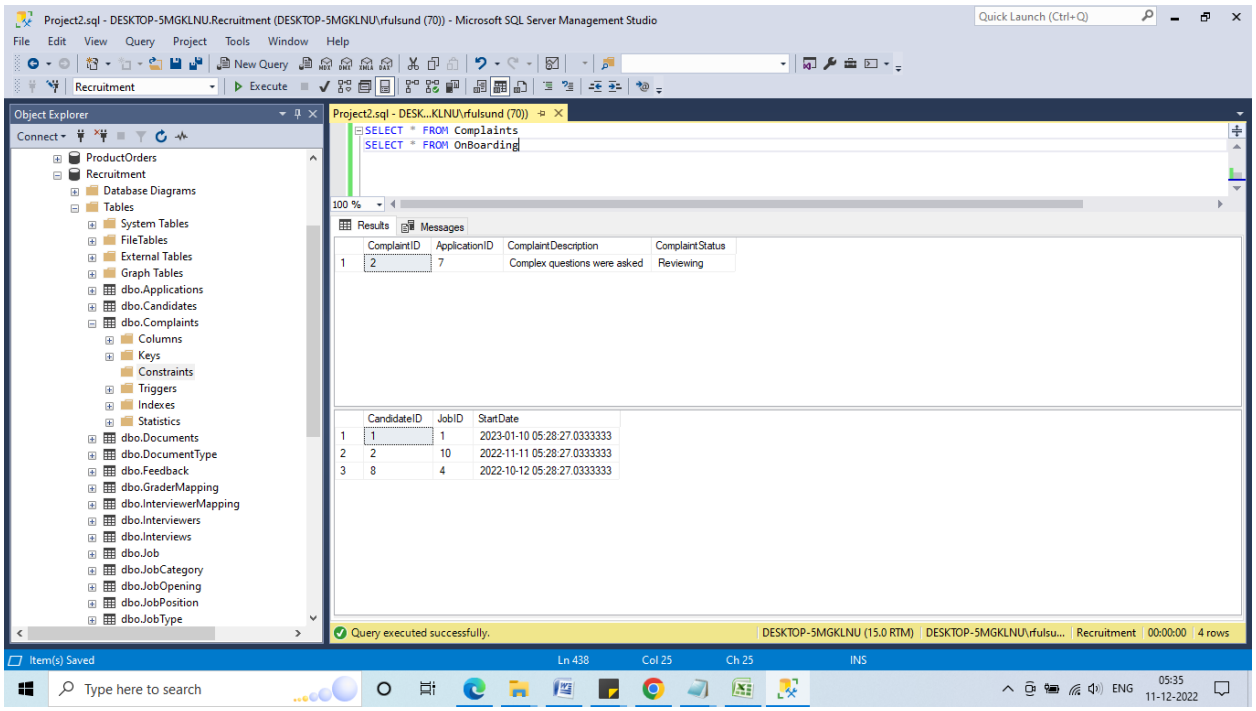

Project2.sql - DESKTOP-5MGKLNU.Recruitment (DESKTOP-5MGKLNU\rfulsund (70)) - Microsoft SQL Server Management Studio
SELECT * FROM Interviews
SELECT * FROM Interviewers
SELECT * FROM InterviewerMapping
Query executed successfully.


Project2.sql - DESKTOP-5MGKLNU.Recruitment (DESKTOP-5MGKLNU\rfulsund (70)) - Microsoft SQL Server Management Studio
Object Explorer
Project2.sql - DESK...KLNU\rfulsund (70))
SELECT * FROM Interviews
SELECT * FROM Feedback
Results
Messages
Query executed successfully.
Item(s) Saved
Type here to search


Project2.sql - DESKTOP-5MGKLNU.Recruitment (DESKTOP-5MGKLNU\rfulsund (70)) - Microsoft SQL Server Management Studio
Object Explorer
Project2.sql - DESK...KLNU\rfulsund (70))
SELECT * FROM Tests
SELECT * FROM GraderMapping
Results
Messages
Query executed successfully.
Item(s) Saved
Type here to search


Project2.sql - DESKTOP-5MGKLNU.Recruitment (DESKTOP-5MGKLNU\rfulsund (70)) - Microsoft SQL Server Management Studio
SELECT * FROM Candidates
SELECT * FROM Offers
SELECT * FROM DocumentType
SELECT * FROM Documents
Query executed successfully.


Project2.sql - DESKTOP-5MGKLNU.Recruitment (DESKTOP-5MGKLNU\rfulsund (70)) - Microsoft SQL Server Management Studio
SELECT * FROM Reimbursement
SELECT * FROM RequestType
Query executed successfully.

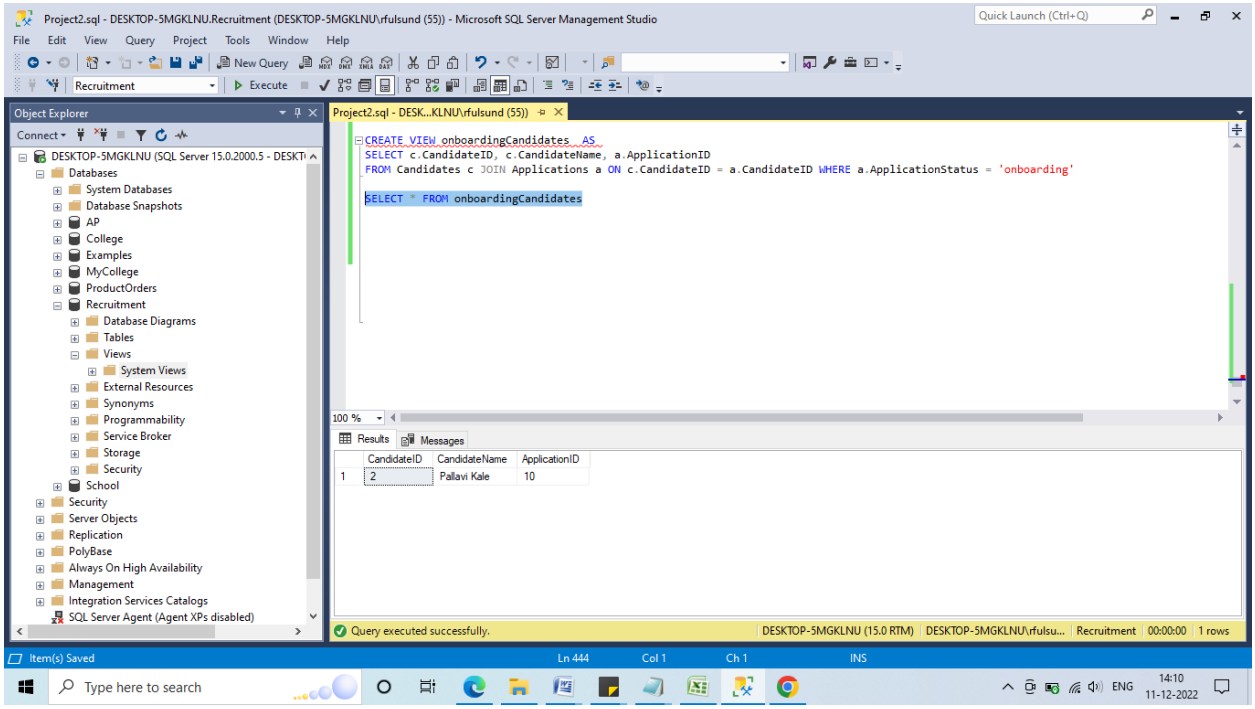

Project2.sql - DESKTOP-5MGKLNU.Recruitment (DESKTOP-5MGKLNU\rfulsund (70)) - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
Recruitment
Execute
Object Explorer
Connect
ProductOrders
Recruitment
Database Diagrams
Tables
System Tables
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
Project2.sql - DESK...KLNU\rfulsund (70))
SELECT * FROM Complaints
SELECT * FROM OnBoarding
100 %
Results
Messages
ComplaintID ApplicationID ComplaintDescription ComplaintStatus
1 2 7 Complex questions were asked Reviewing
CandidateID JobID StartDate
1 1 1 2023-01-10 05:28:27.0333333
2 2 10 2022-11-11 05:28:27.0333333
3 8 4 2022-10-12 05:28:27.0333333
Query executed successfully.
DESKTOP-5MGKLNU (15.0 RTM)
DESKTOP-5MGKLNU\rfulsu...
Recruitment
00:00:00
4 rows
Item(s) Saved
Ln 438
Col 25
Ch 25
INS
Type here to search
ENG
05:35
11-12-2022

Section C : Testing

Views -

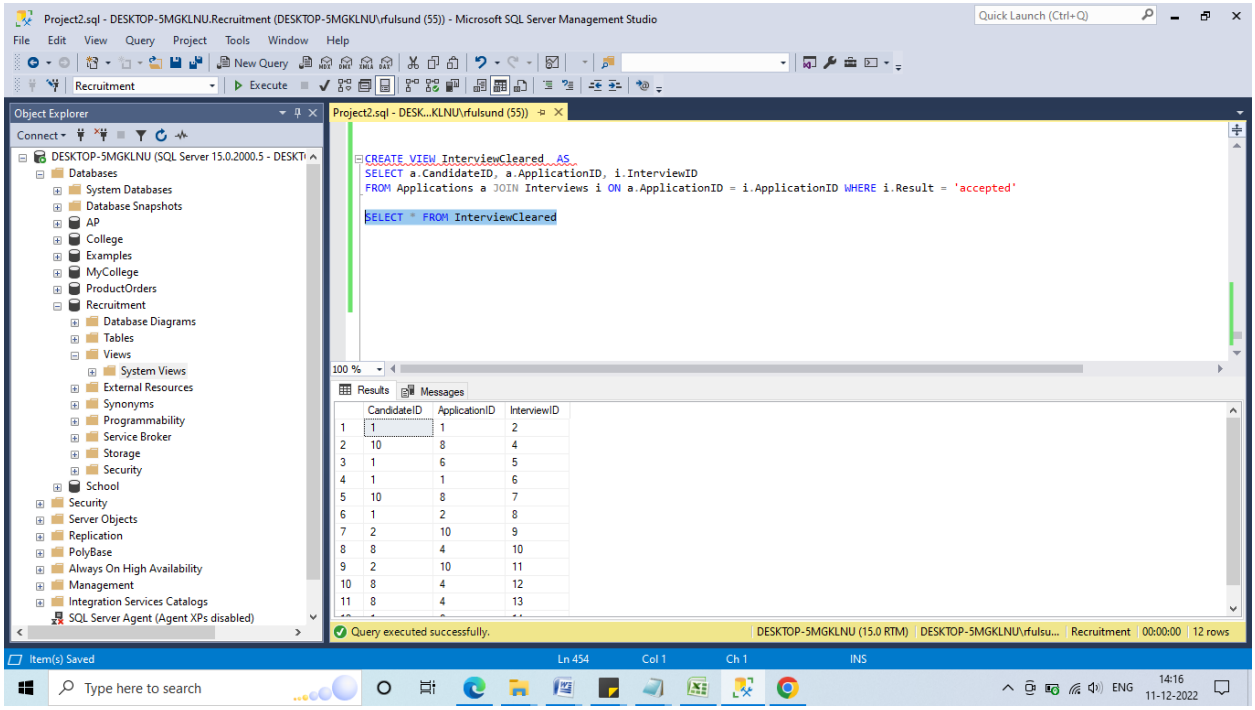
1. Write a view named SelectedCandidates that returns three columns: CandidateID, CandidateName and ApplicationID. Then, write a SELECT statement that returns all the columns in the view.
   Displays all the candidates whose onboarding is going on.

```
CREATE VIEW onboardingCandidates  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 onboardingCandidates
```



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.

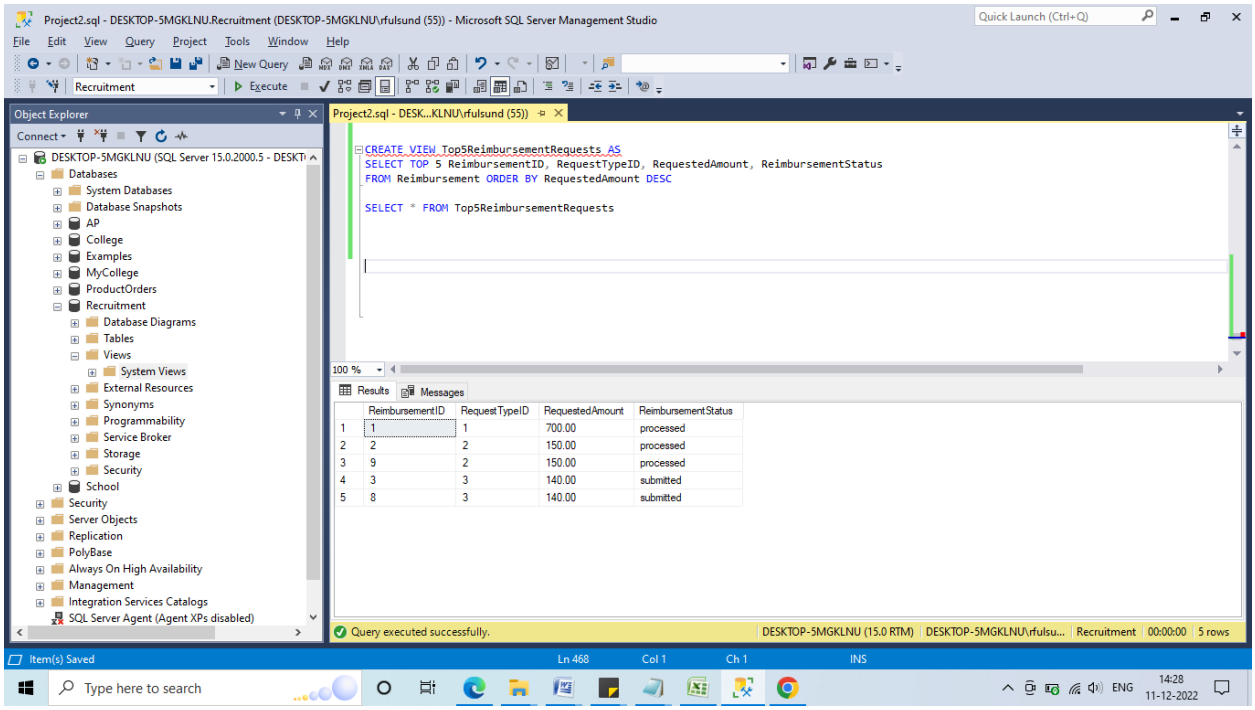
```
CREATE VIEW InterviewCleared  AS
SELECT a.CandidateID, a.ApplicationID, i.InterviewID
FROM Applications a JOIN Interviews i ON a.ApplicationID = i.ApplicationID WHERE
i.Result = 'accepted'
SELECT * FROM InterviewCleared
```



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
```

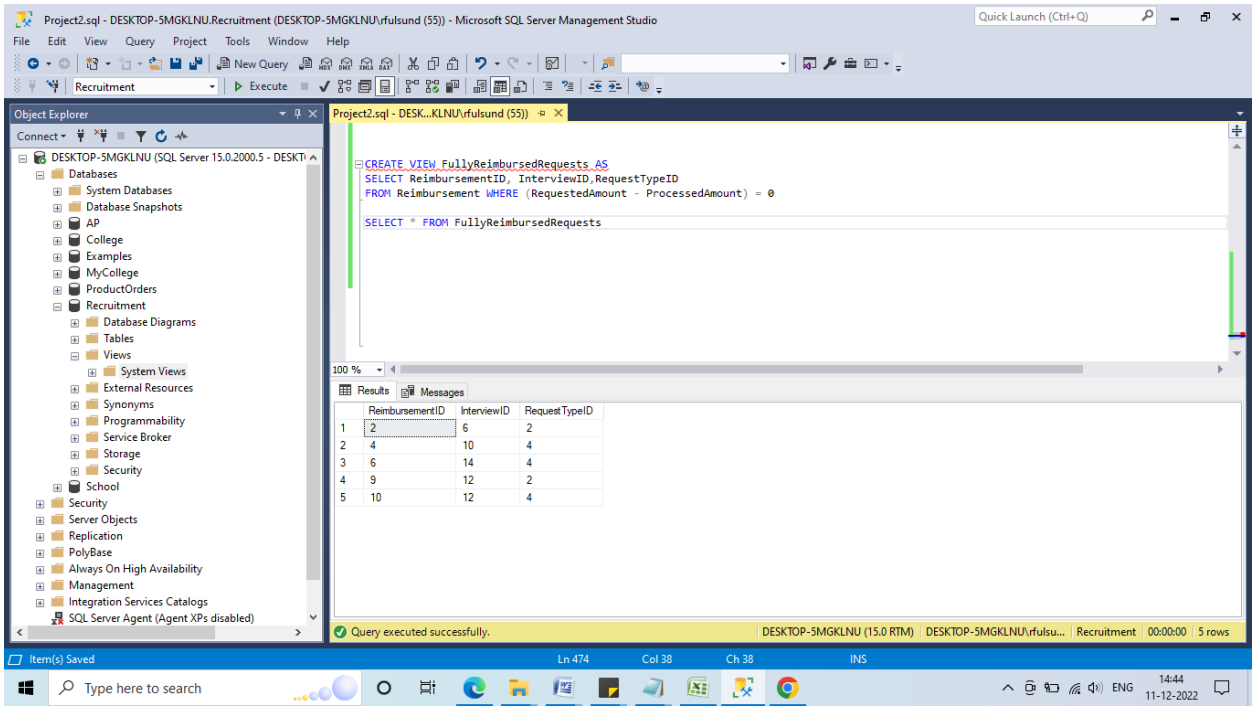
```
SELECT * FROM Top5ReimbursementRequests
```



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
```

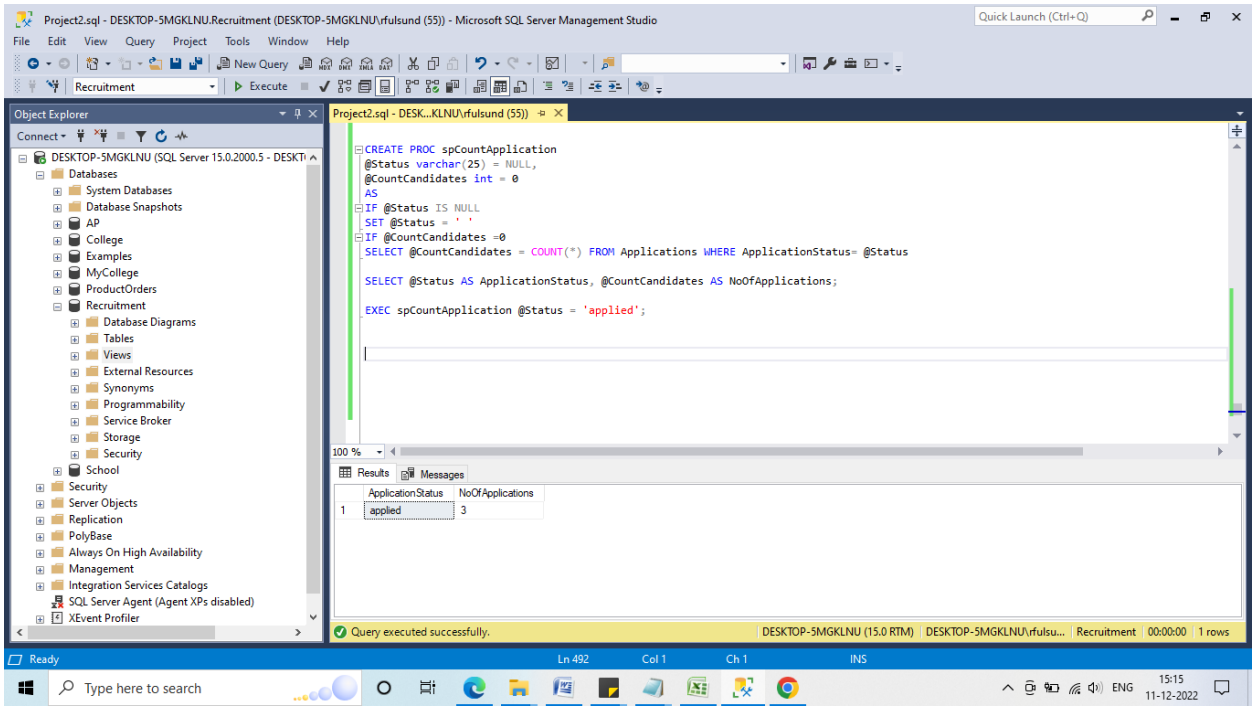


Stored Procedures -

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;
```

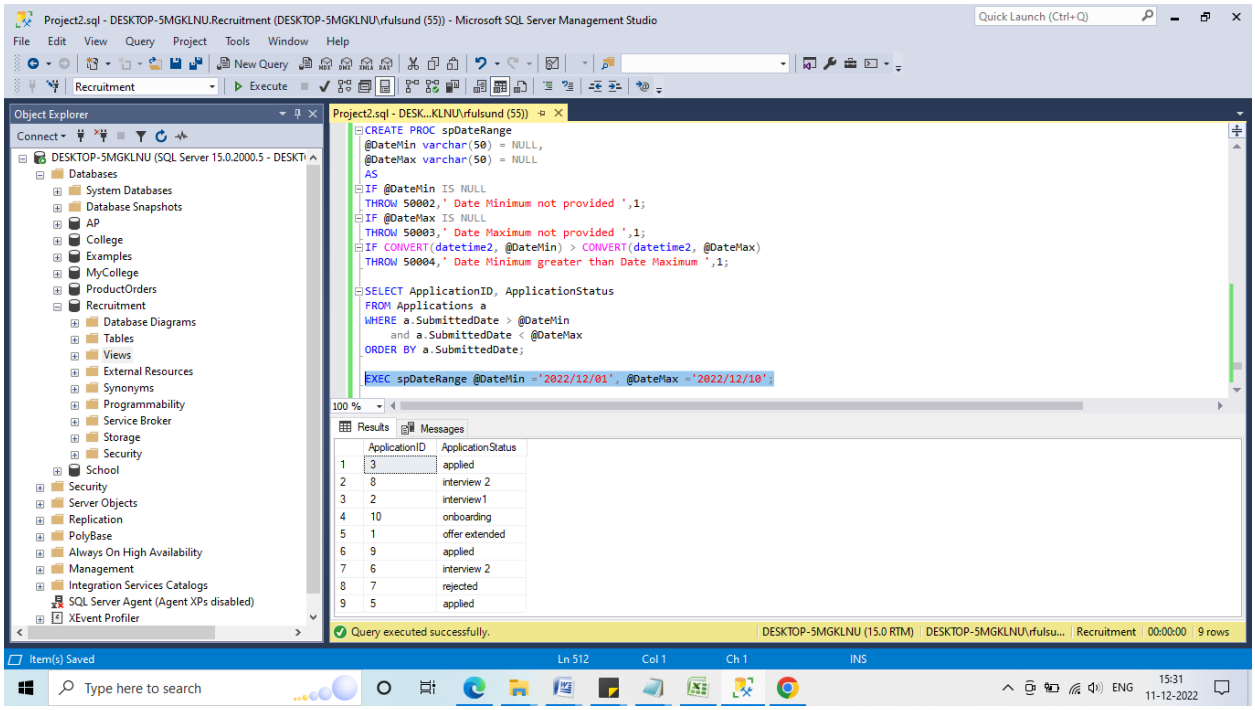
```
EXEC spCountApplication @Status = 'applied';
```



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;
```

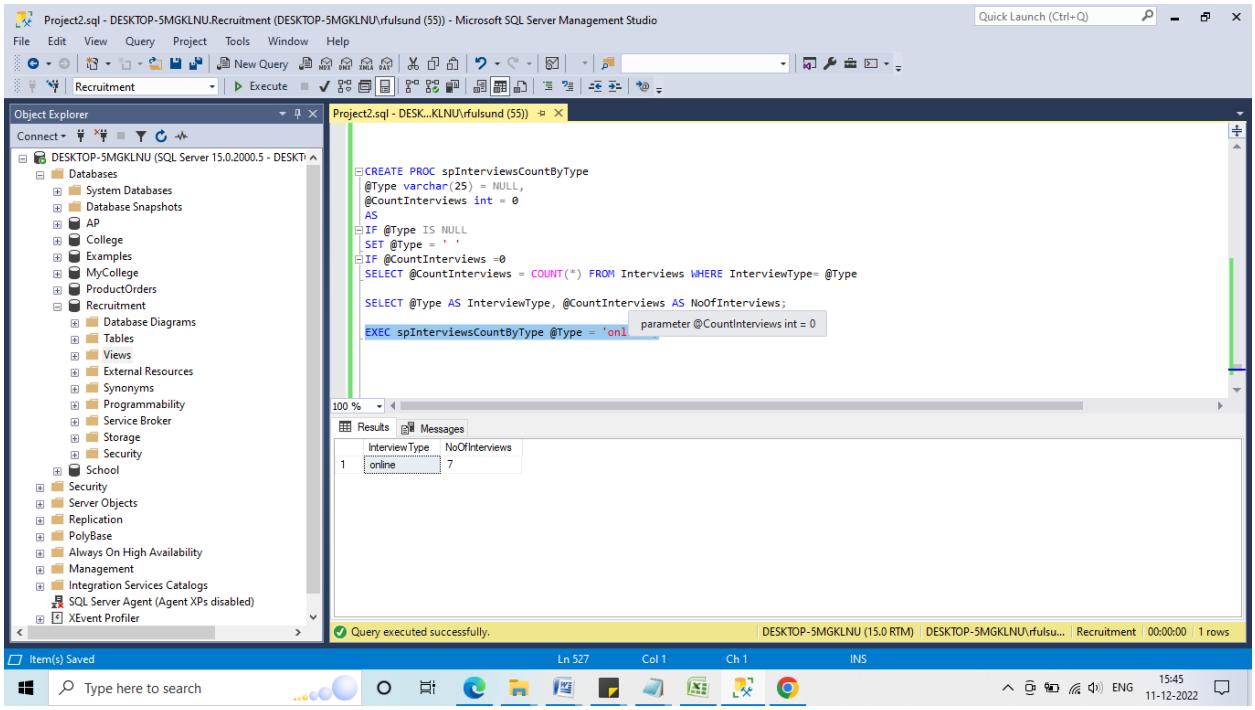
```
EXEC spDateRange @DateMin ='2022/12/01', @DateMax ='2022/12/10';
```



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;
```

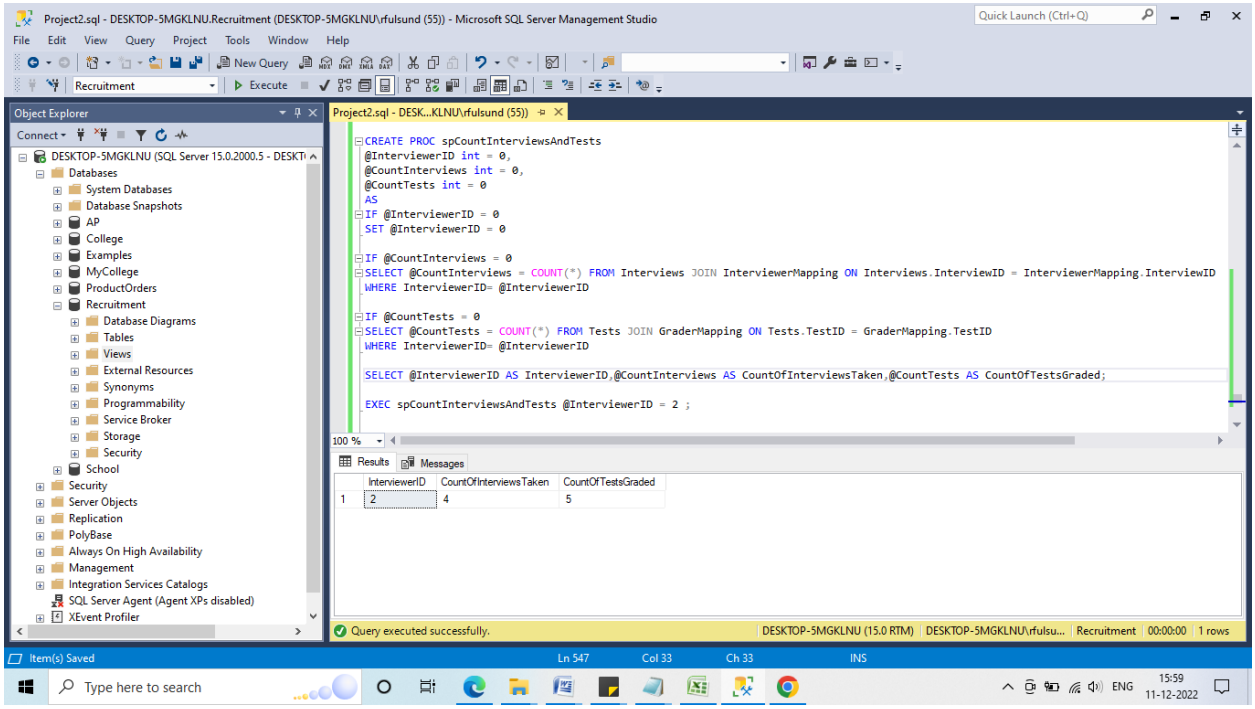
```
EXEC spInterviewsCountByType @Type = 'online';
```



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
ONInterviews.InterviewID = InterviewerMapping.InterviewID
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 ;
```

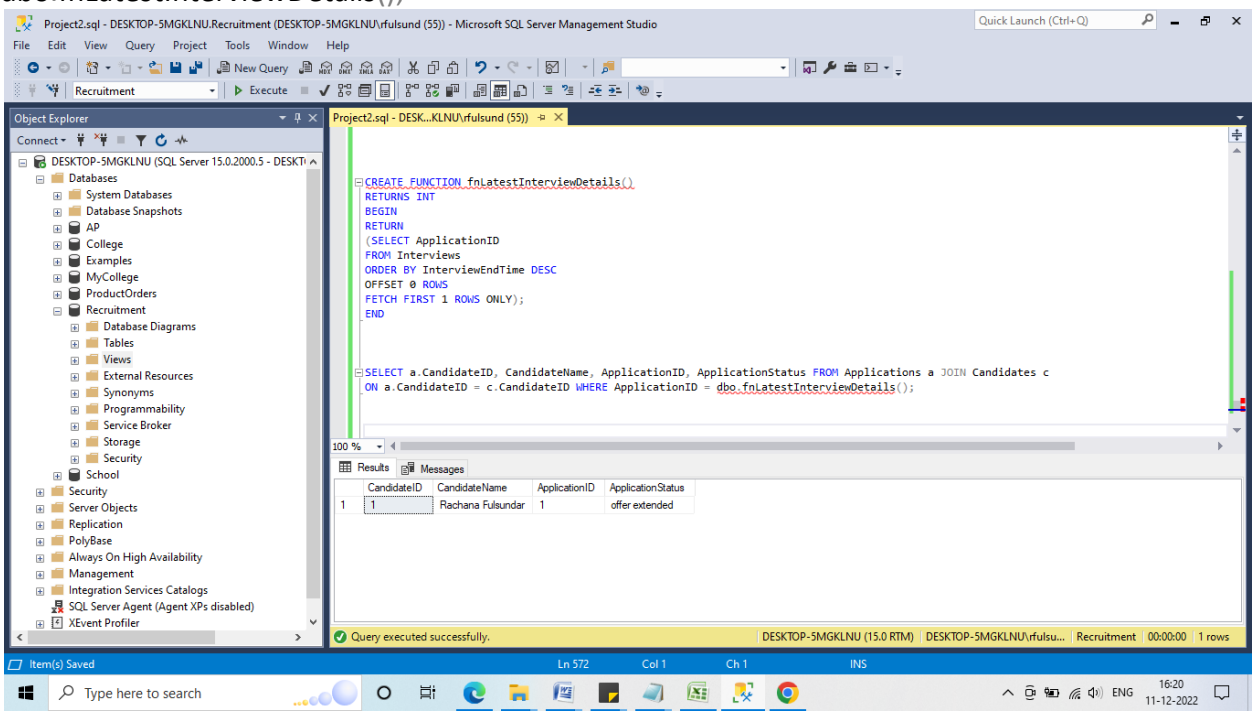


User Defined Functions -

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

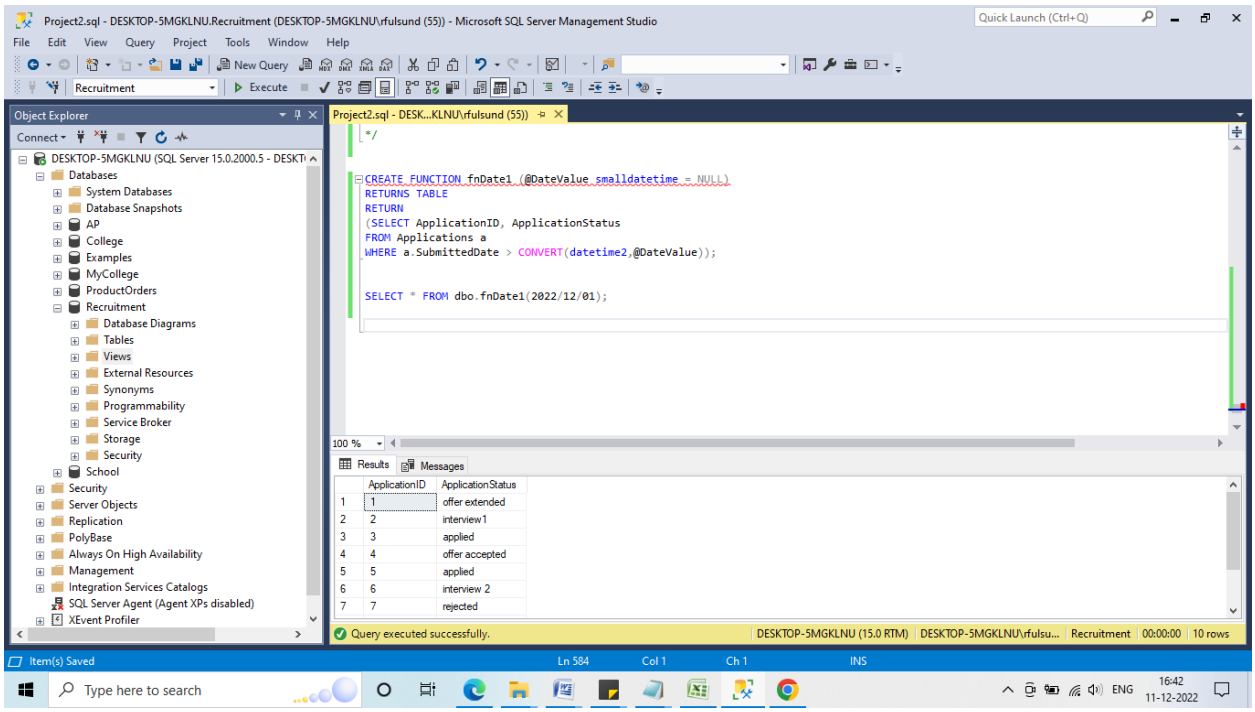
```
CREATE FUNCTION fnLatestInterviewDetails()
RETURNS INT
BEGIN
RETURN
(SELECT ApplicationID
FROM Interviews
ORDER BY InterviewEndTime DESC
OFFSET 0 ROWS
FETCH FIRST 1 ROWS ONLY);
END
```

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



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

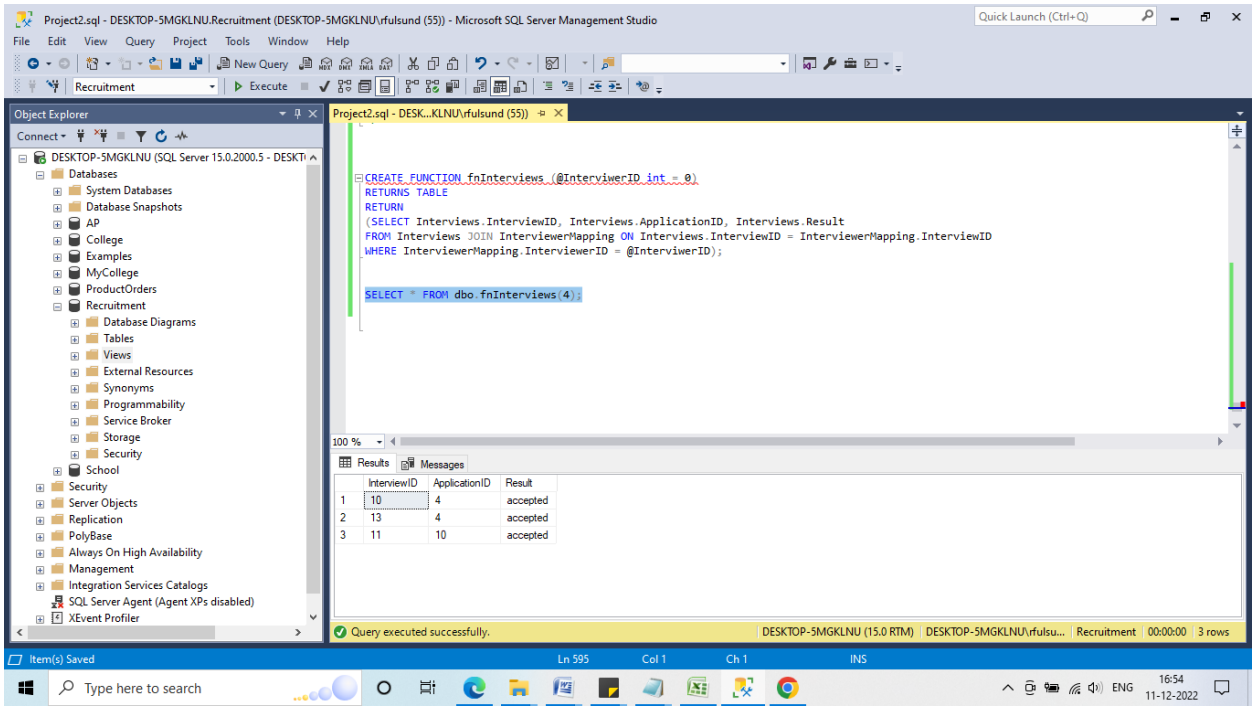
```
CREATE FUNCTION fnDate1 (@DateValue smalldatetime = NULL)
RETURNS TABLE
RETURN
(SELECT ApplicationID, ApplicationStatus
FROM Applications a
WHERE a.SubmittedDate > CONVERT(datetime2,@DateValue));
SELECT * FROM dbo.fnDate1(2022/12/01);
```



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

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

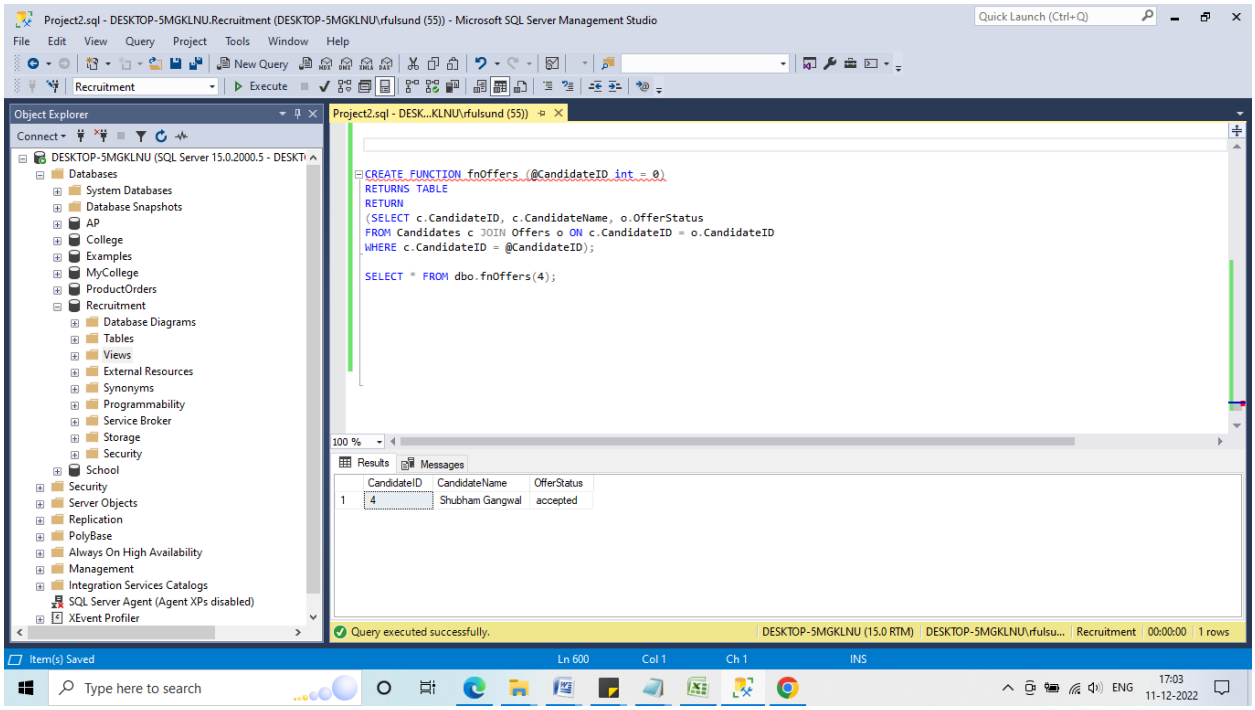
```
SELECT * FROM dbo.fnInterviews(4);
```



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);
```

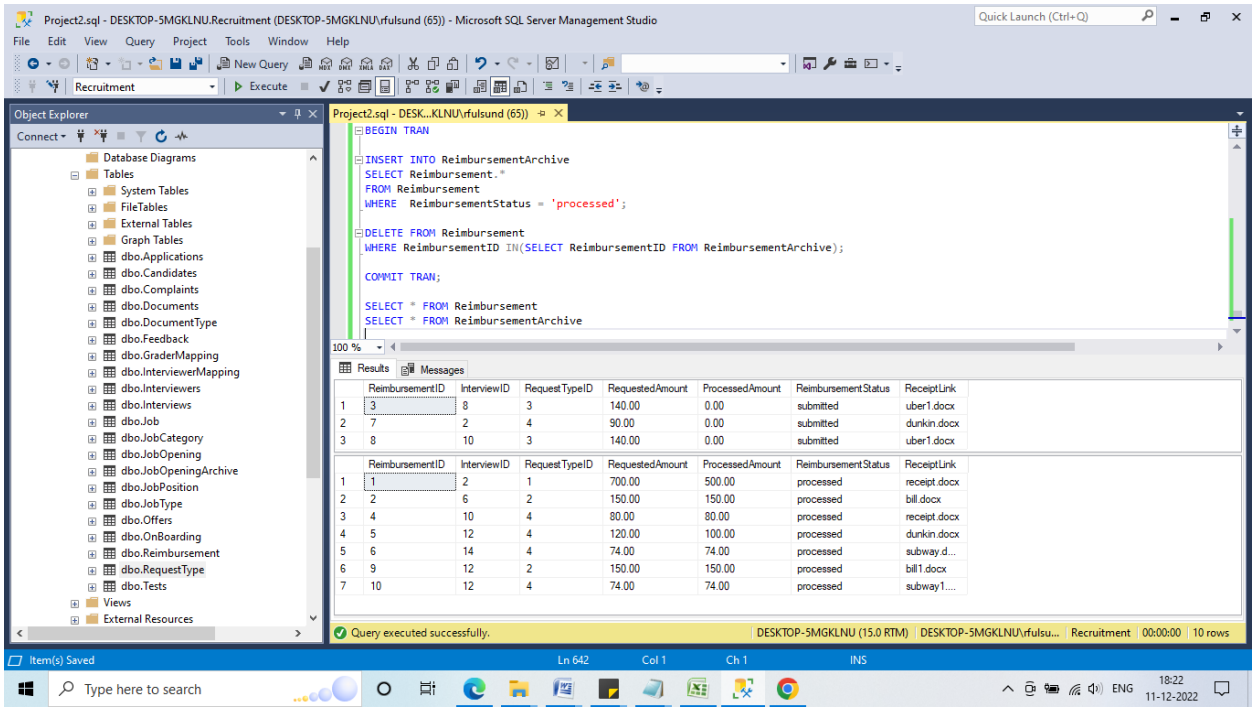
```
SELECT * FROM dbo.fnOffers(4);
```



Transactions -

1. Delete all processed reimbursements from reimbursement table.

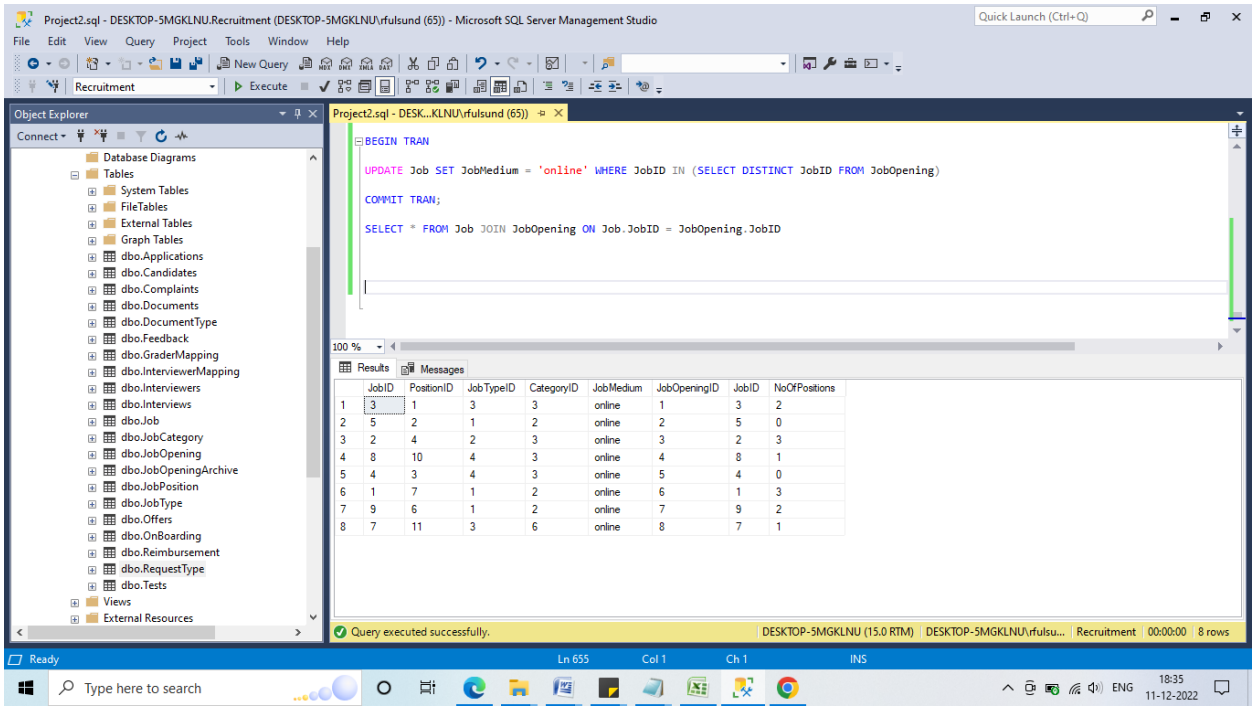
```
BEGIN TRAN
INSERT INTO ReimbursementArchive
SELECT Reimbursement.*
FROM Reimbursement
WHERE  ReimbursementStatus = 'processed';

DELETE FROM Reimbursement
WHERE ReimbursementID IN(SELECT ReimbursementID FROM ReimbursementArchive);

COMMIT TRAN;

SELECT * FROM Reimbursement
SELECT * FROM ReimbursementArchive
```



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

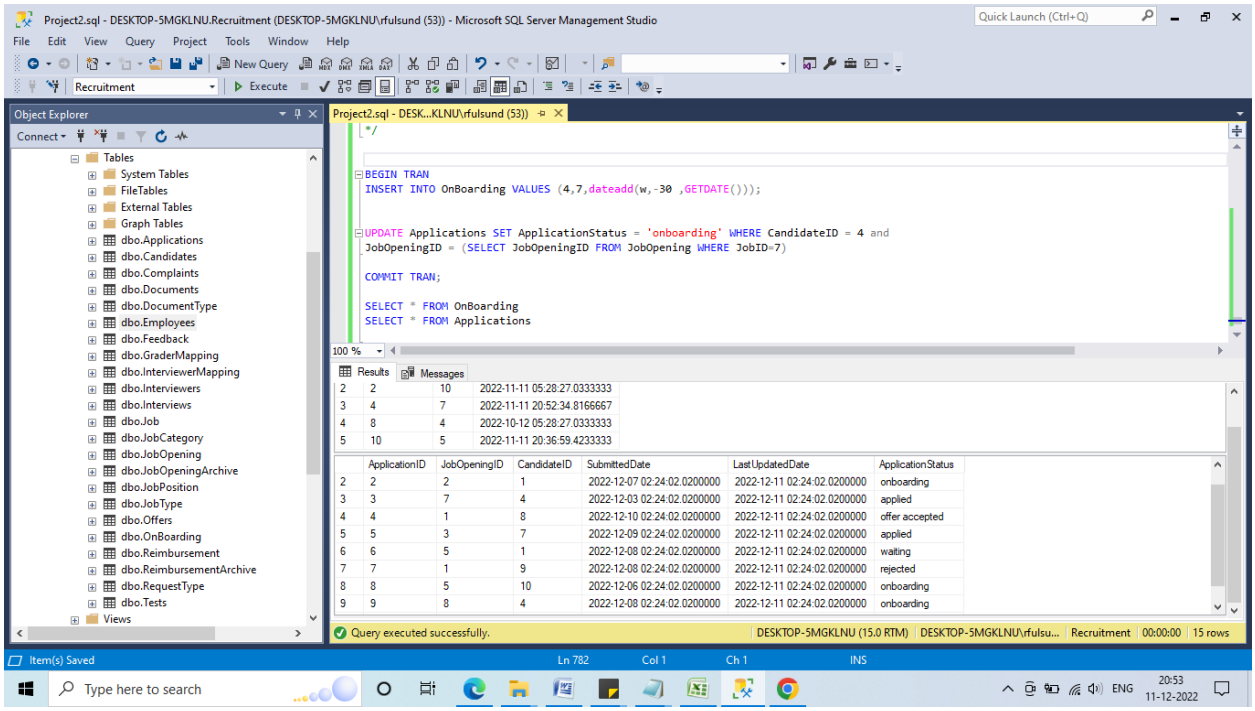
```
BEGIN TRAN
```

```
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
```



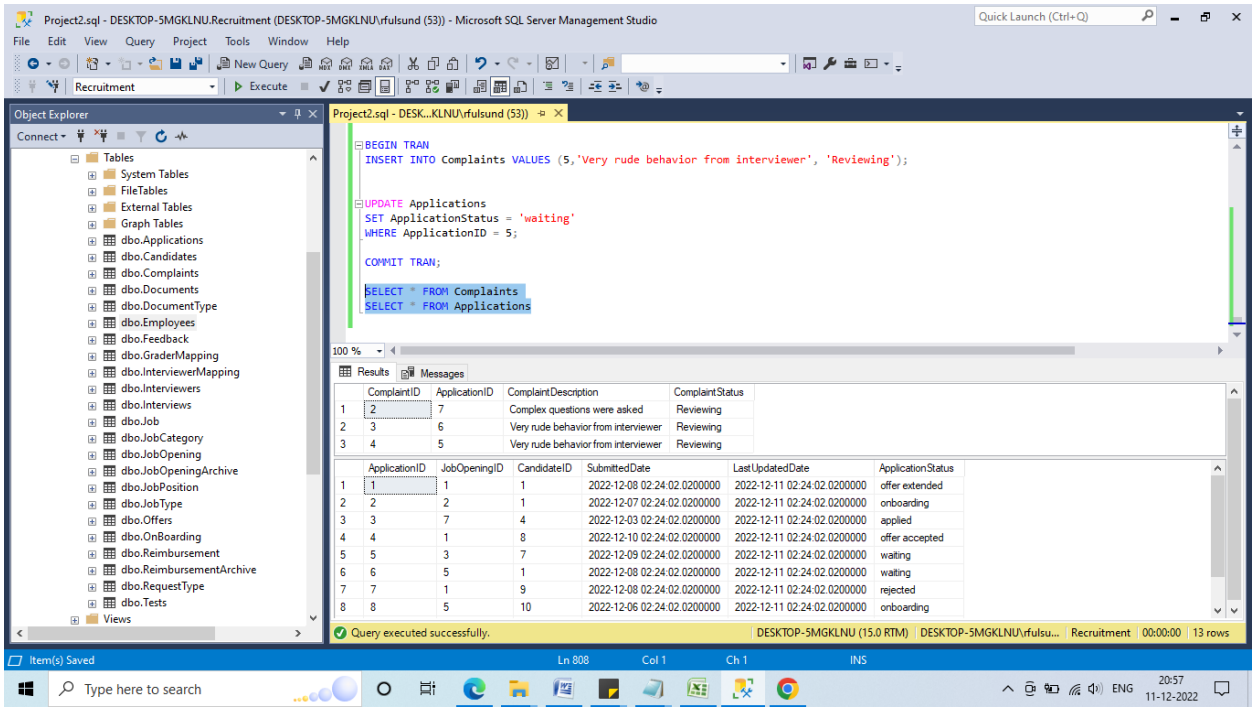
3. Update application status to ‘onboarding’ whenever a new entry in onboarding table.

```
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
```



4. Update status of application to ‘waiting’ if candidate registers a compliant

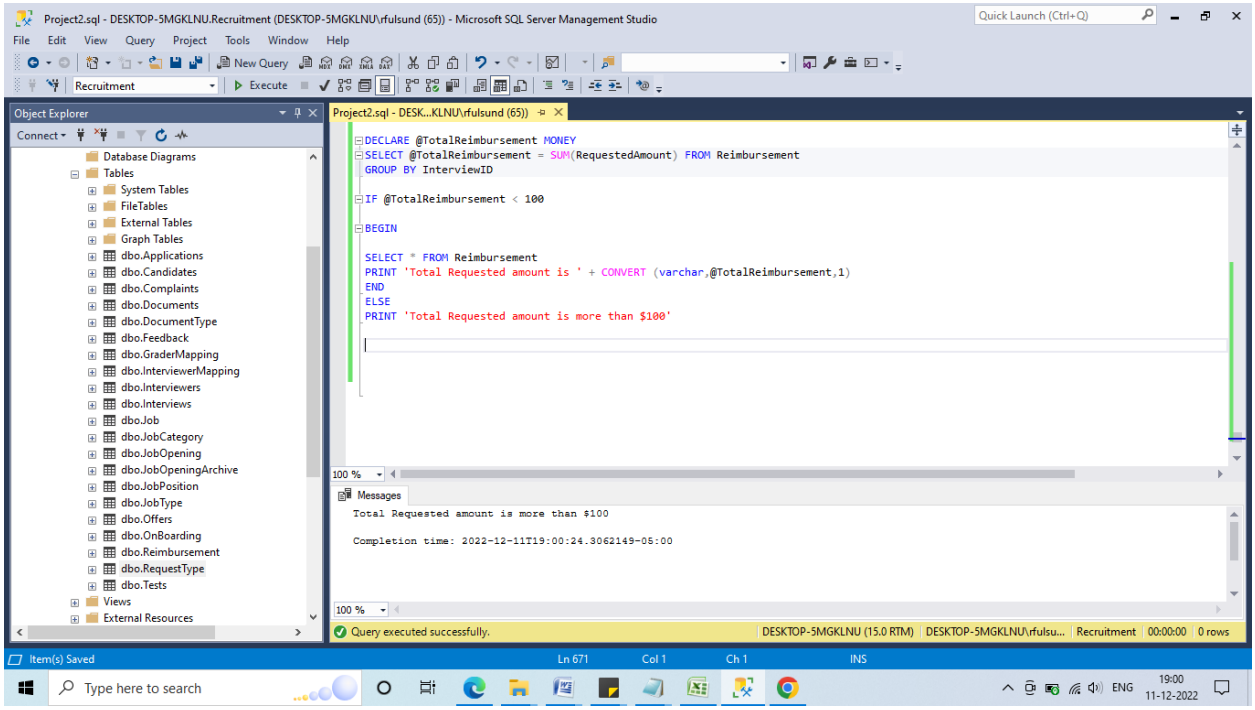
```
BEGIN TRAN
INSERT INTO Complaints VALUES (5,'Very rude behavior from interviewer', 'Reviewing');

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

COMMIT TRAN;

SELECT * FROM Complaints
SELECT * FROM Applications
```


Project2.sql - DESKTOP-5MGKLNU.Recruitment (DESKTOP-5MGKLNU\rfulsund (53)) - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
Execute
Recruitment
Object Explorer
Connect
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Applications
dbo.Candidates
dbo.Complaints
dbo.Documents
dbo.DocumentType
dbo.Employees
dbo.Feedback
dbo.GraderMapping
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
Project2.sql - DESK...KLNU\rfulsund (53))
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
100 %
Results
Messages
ComplaintID ApplicationID ComplaintDescription ComplaintStatus
1 2 7 Complex questions were asked Reviewing
2 3 6 Very rude behavior from interviewer Reviewing
3 4 5 Very rude behavior from interviewer Reviewing
ApplicationID JobOpeningID CandidateID SubmittedDate LastUpdatedDate ApplicationStatus
1 1 1 1 2022-12-08 02:24:02.0200000 2022-12-11 02:24:02.0200000 offer extended
2 2 2 1 2022-12-07 02:24:02.0200000 2022-12-11 02:24:02.0200000 onboarding
3 3 7 4 2022-12-03 02:24:02.0200000 2022-12-11 02:24:02.0200000 applied
4 4 1 8 2022-12-10 02:24:02.0200000 2022-12-11 02:24:02.0200000 offer accepted
5 5 3 7 2022-12-09 02:24:02.0200000 2022-12-11 02:24:02.0200000 waiting
6 6 5 1 2022-12-08 02:24:02.0200000 2022-12-11 02:24:02.0200000 waiting
7 7 1 9 2022-12-08 02:24:02.0200000 2022-12-11 02:24:02.0200000 rejected
8 8 5 10 2022-12-06 02:24:02.0200000 2022-12-11 02:24:02.0200000 onboarding
Query executed successfully.
DESKTOP-5MGKLNU (15.0 RTM)
DESKTOP-5MGKLNU\rfulsu...
Recruitment
00:00:00
13 rows
Item(s) Saved
Ln 808
Col 1
Ch 1
INS
Type here to search
ENG
20:57
11-12-2022

Scripts -

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

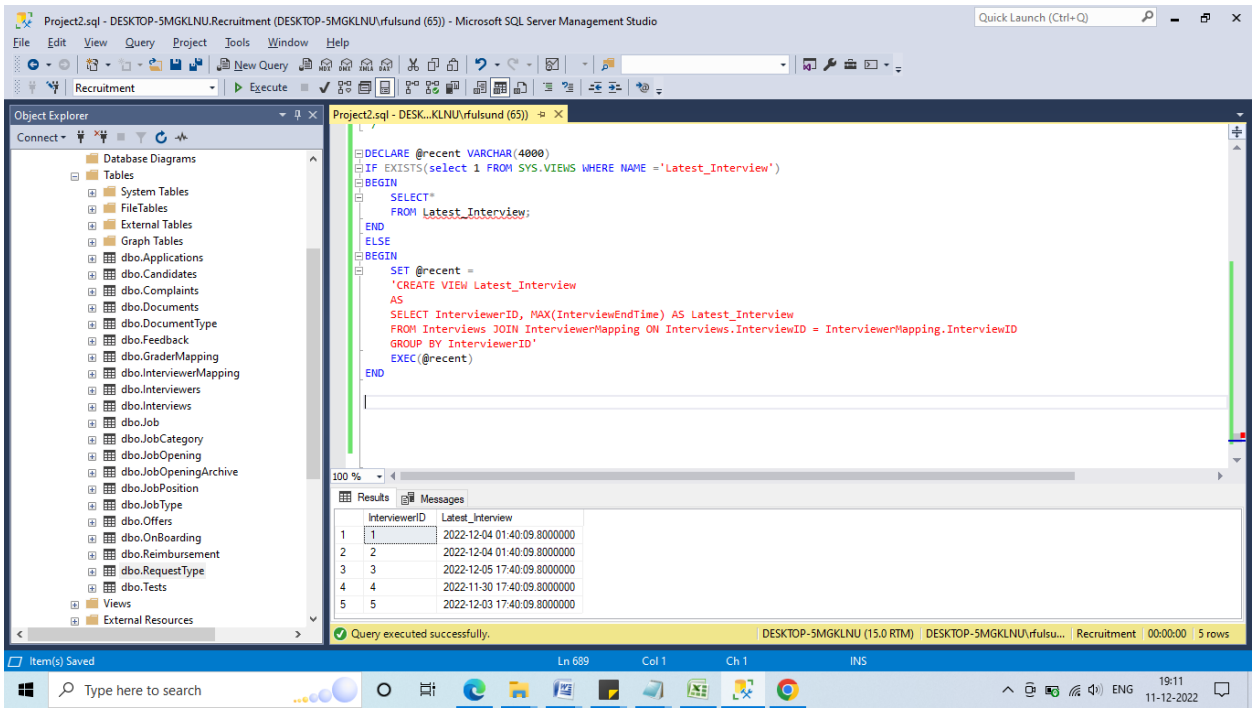
```
DECLARE @TotalReimbursement MONEY
SELECT @TotalReimbursement = SUM(RequestedAmount) FROM Reimbursement
GROUP BY InterviewID

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
```

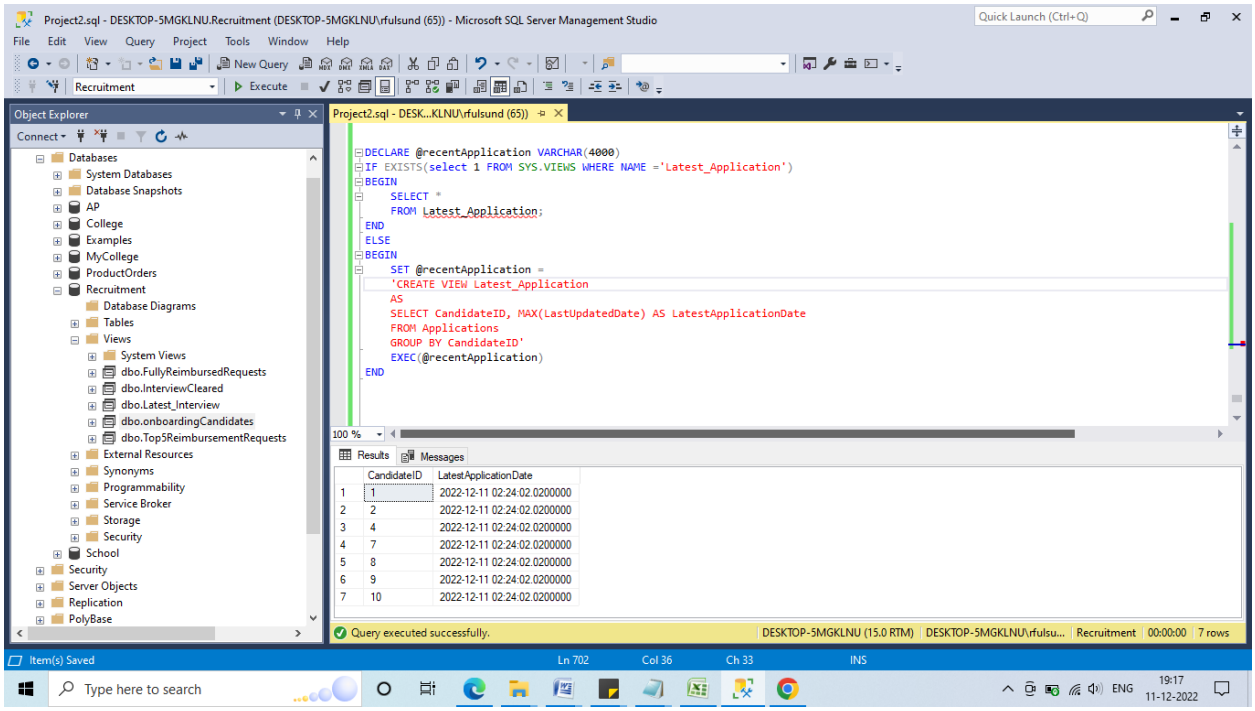
```
        SELECT*
        FROM Latest_Interview;
END
ELSE
BEGIN
        SET @recent =
        'CREATE VIEW Latest_Interview
        AS
        SELECT InterviewerID, MAX(InterviewEndTime) AS Latest_Interview
        FROM Interviews JOIN InterviewerMapping ON Interviews.InterviewID
InterviewerMapping.InterviewID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



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
```

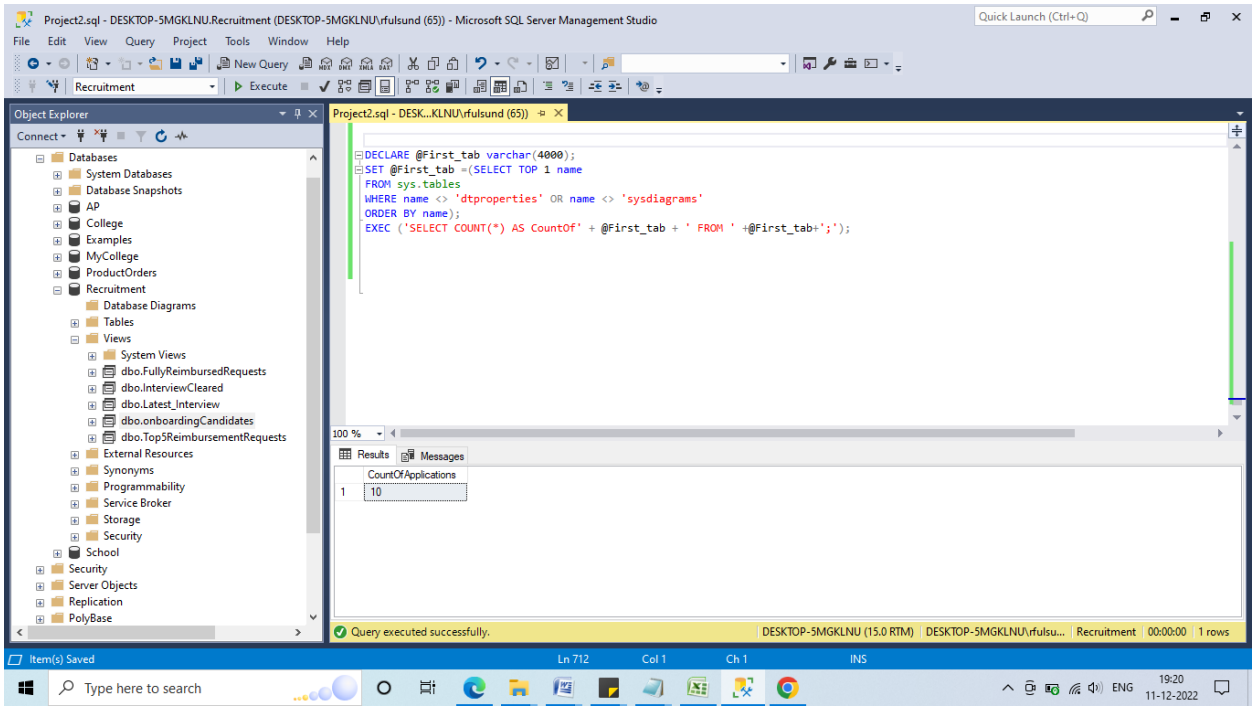
```
BEGIN
        SET @recentApplication =
        'CREATE VIEW Latest_Application
        AS
        SELECT CandidateID, MAX(LastUpdatedDate) AS LatestApplicationDate
        FROM Applications
        GROUP BY CandidateID'
        EXEC(@recentApplication)
END
```



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+';');
```

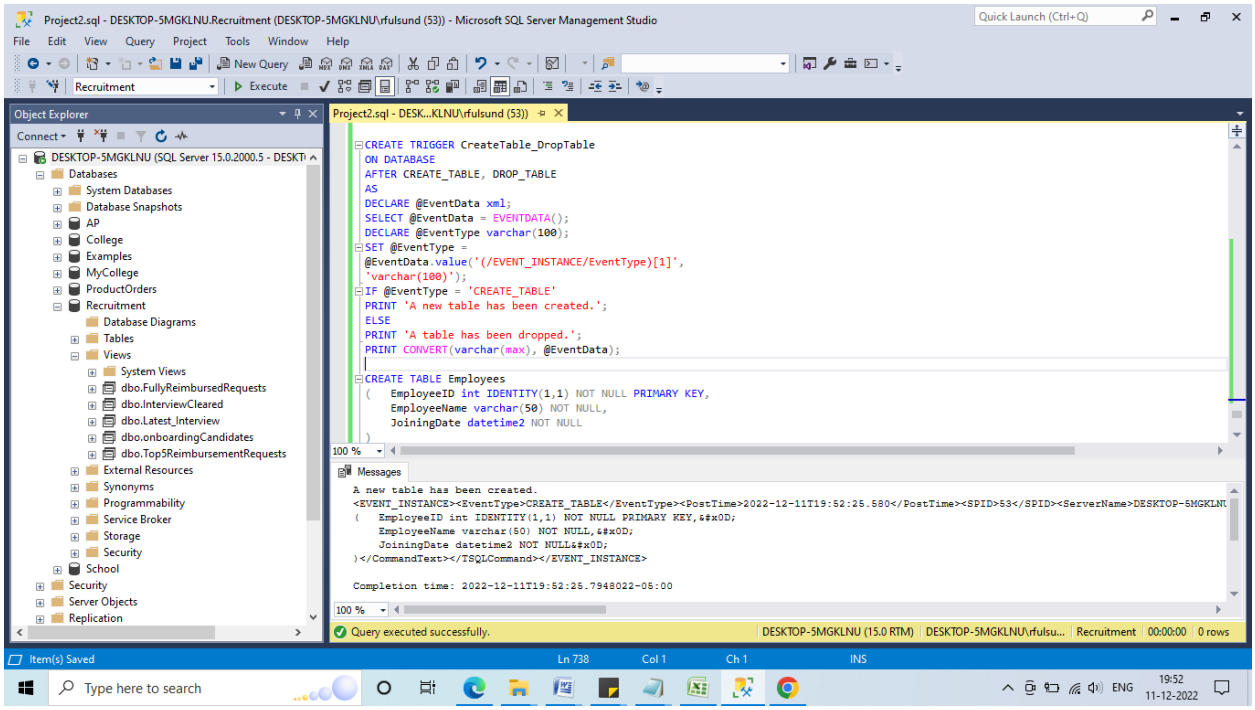


Triggers -

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]',
'varchar(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
```

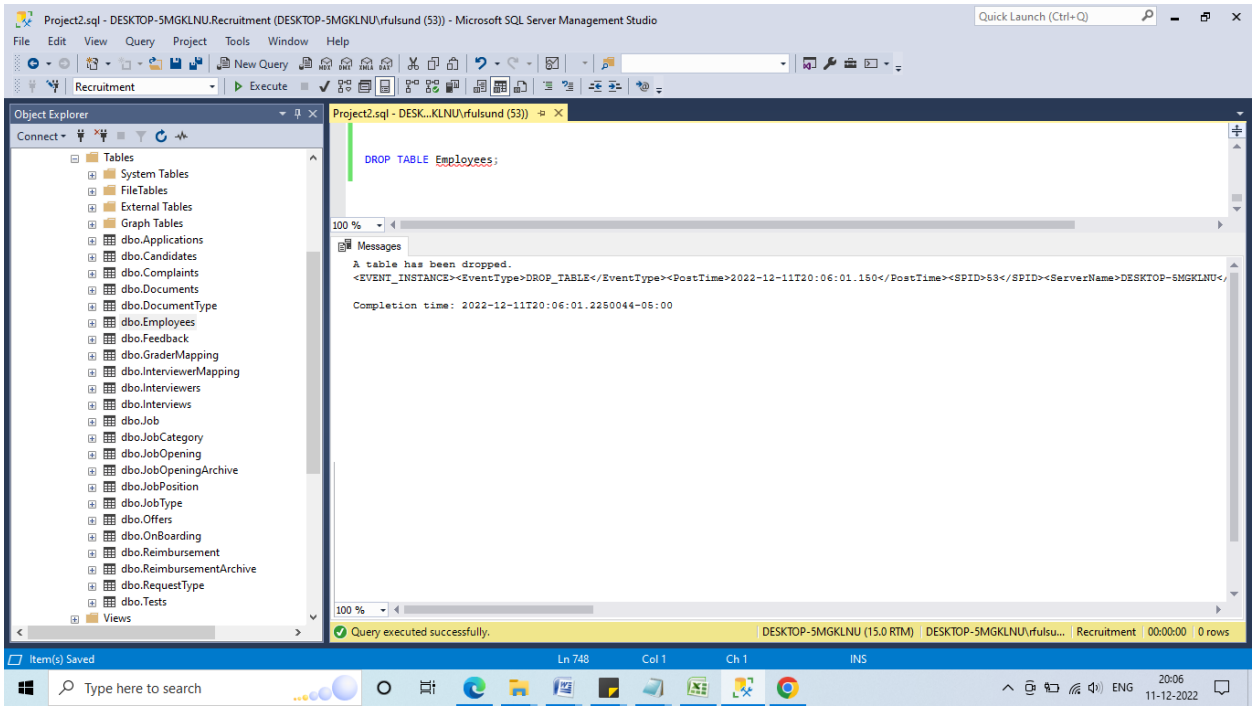
```
(   EmployeeID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
        EmployeeName varchar(50) NOT NULL,
        JoiningDate datetime2 NOT NULL
)
```



2. Trigger for DROP TABLE statement

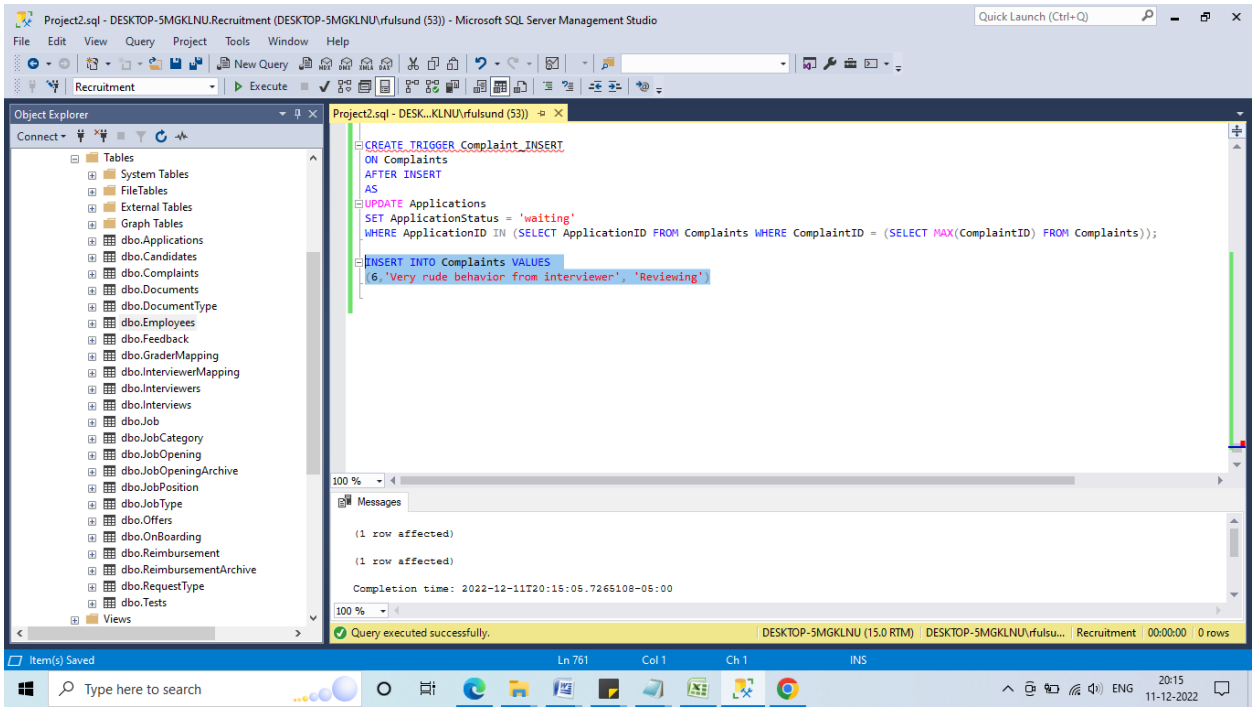
```
CREATE TRIGGER CreateTable_DropTable
ON DATABASE
AFTER CREATE_TABLE, DROP_TABLE
AS
DECLARE @EventData xml;
SELECT @EventData = EVENTDATA();
DECLARE @EventType varchar(100);
SET @EventType =
@EventData.value('(/EVENT_INSTANCE/EventType)[1]',
'varchar(100)');
IF @EventType = 'CREATE_TABLE'
PRINT 'A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

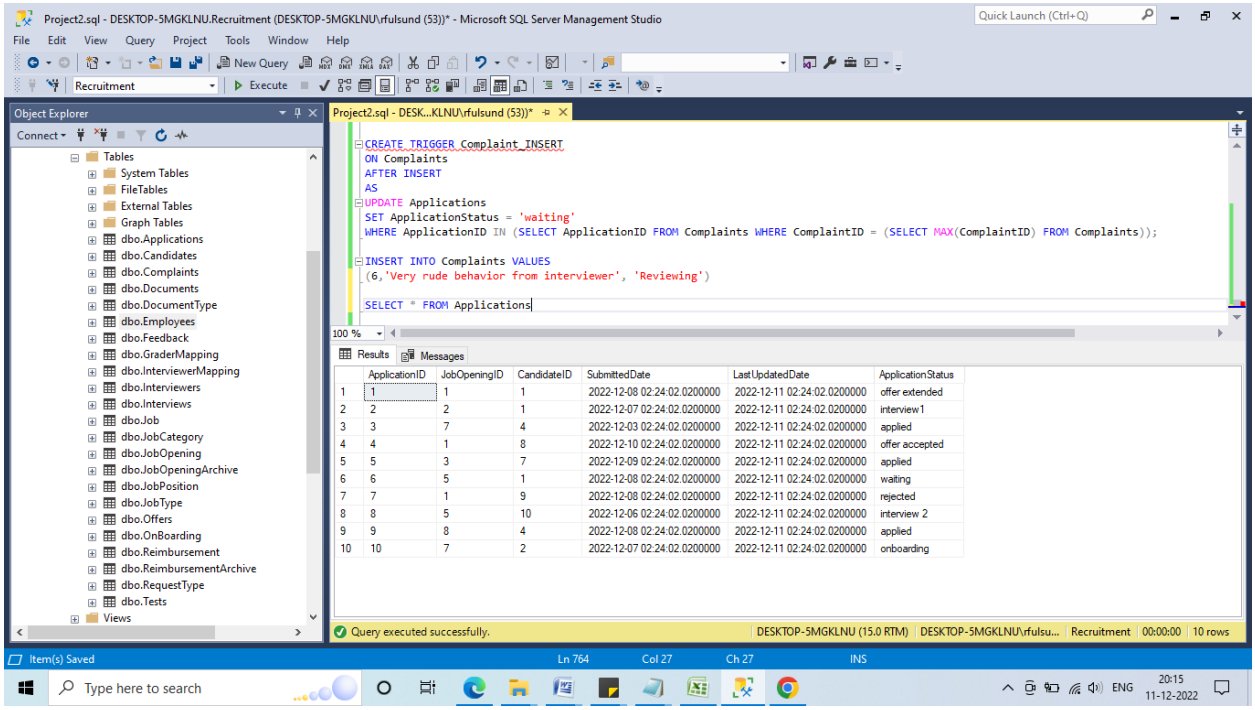
```
DROP TABLE Employees;
```



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
```

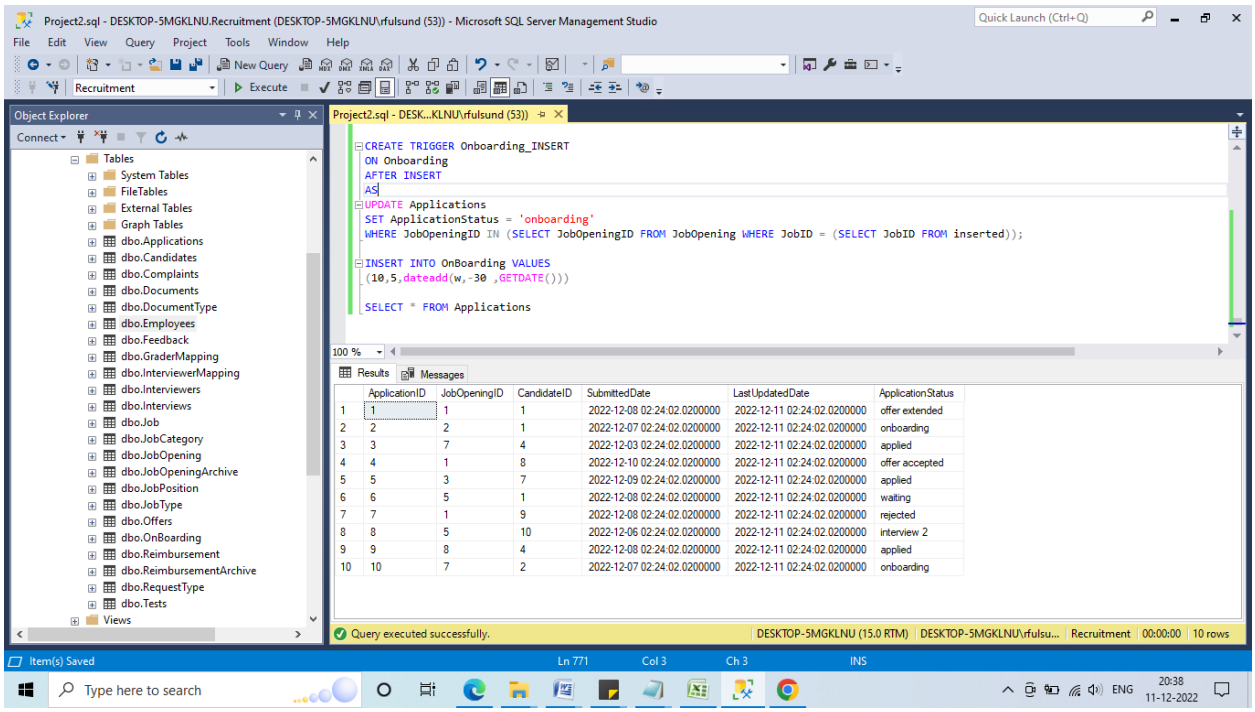




4. Trigger to update application status to ‘onboarding’ whenever a new entry in onboarding table.

```
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
```

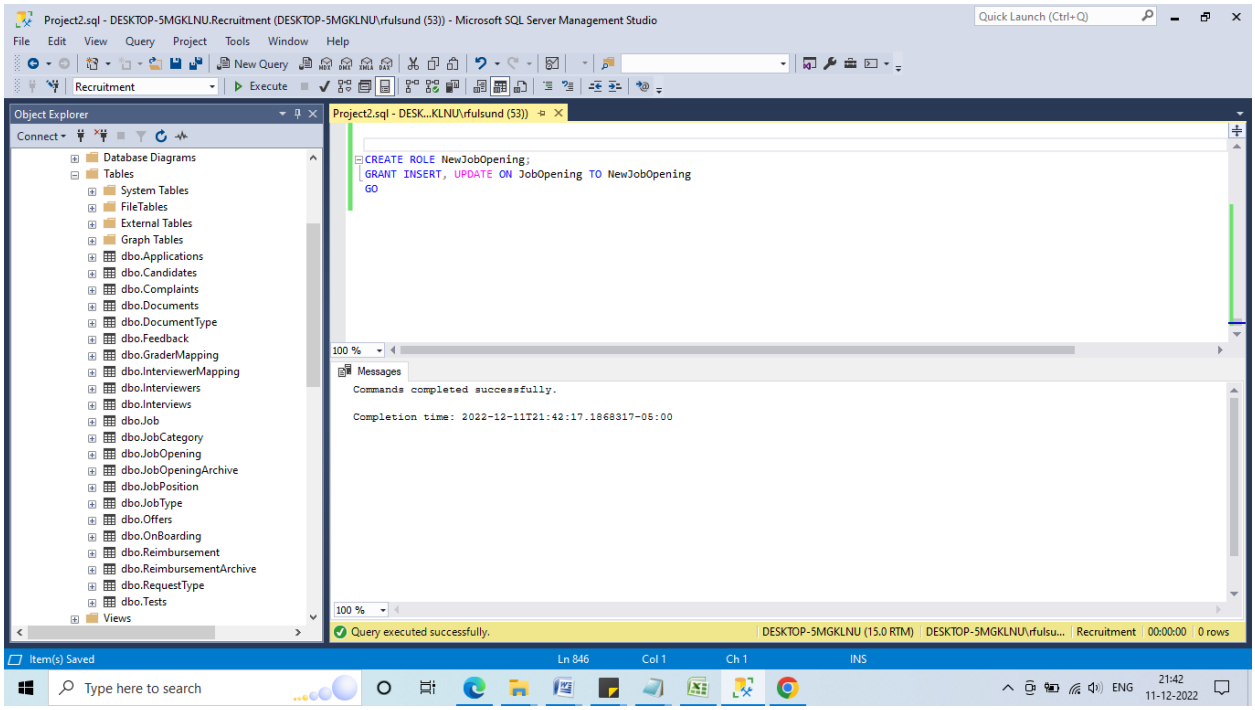


Security Roles –

1. Role 1 :

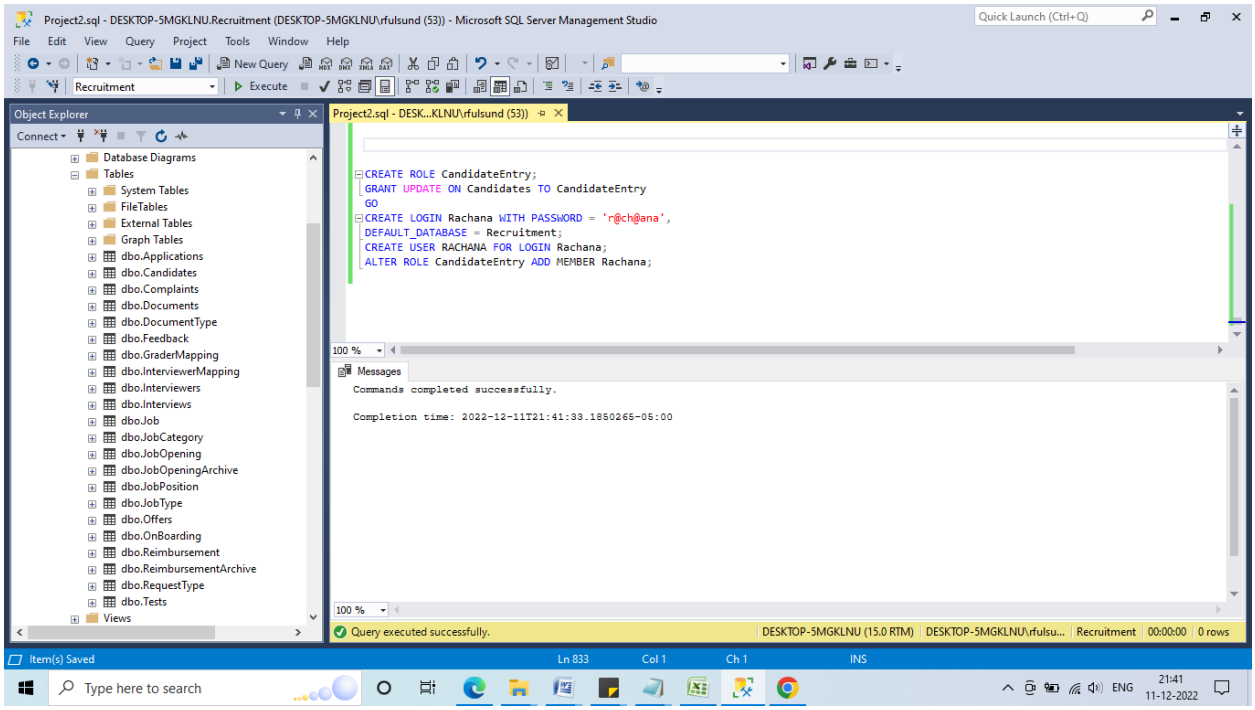
```
CREATE ROLE NewJobOpening;
GRANT INSERT, UPDATE ON JobOpening TO NewJobOpening
```

```
GO
```



2. Role 2 :

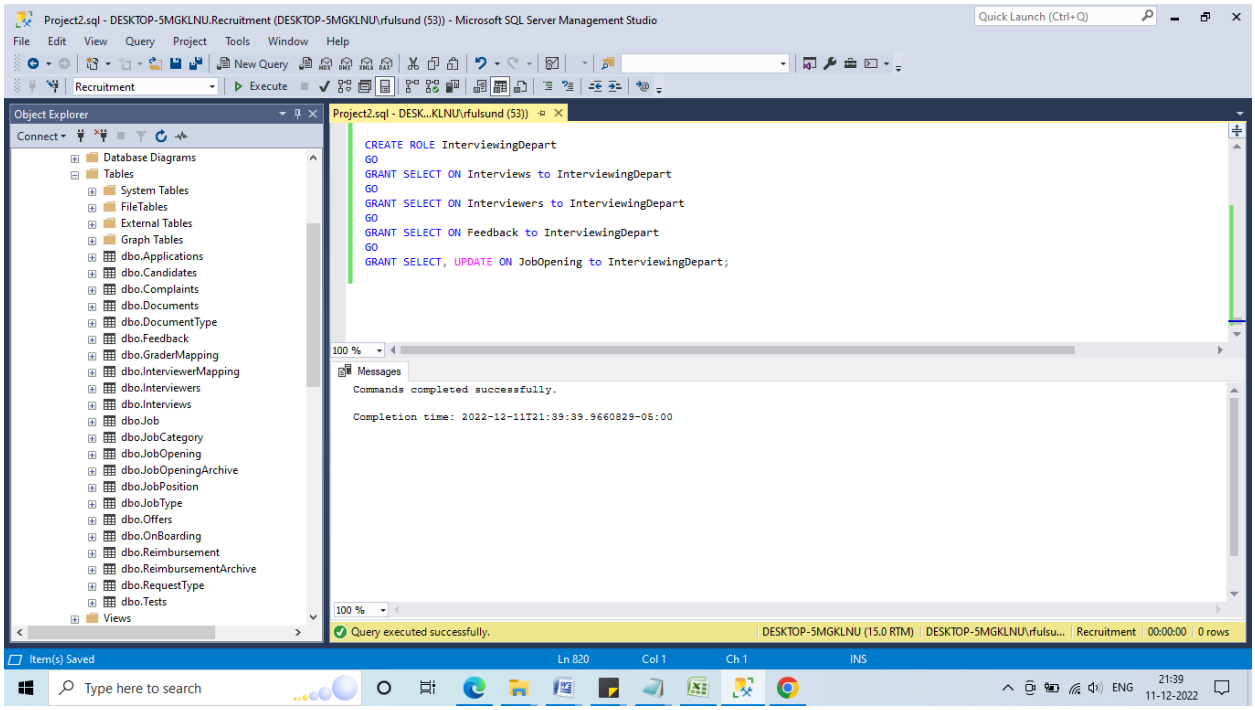
```
CREATE ROLE CandidateEntry;
GRANT UPDATE ON Candidates TO CandidateEntry
GO
CREATE LOGIN Rachana WITH PASSWORD = 'r@ch@ana',
DEFAULT_DATABASE = Recruitment;
CREATE USER RACHANA FOR LOGIN Rachana;
ALTER ROLE CandidateEntry ADD MEMBER Rachana;
```



3. Role 3 :

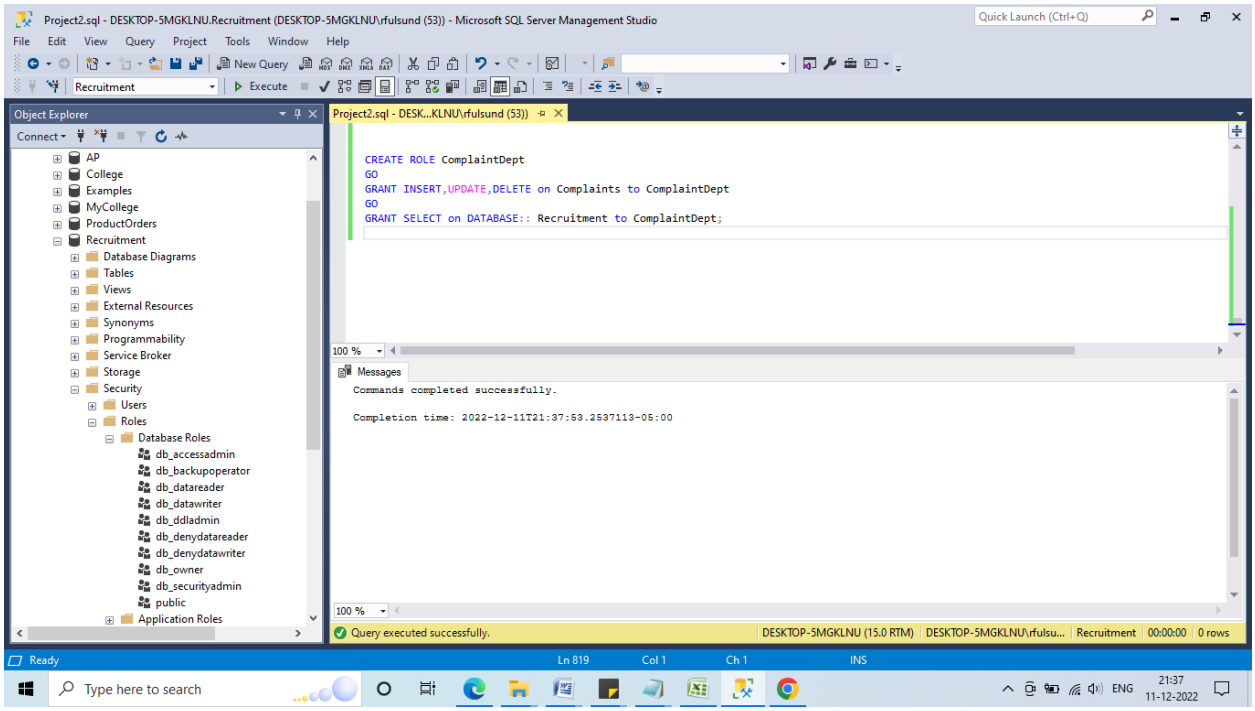
```
CREATE ROLE InterviewingDepart
GO
GRANT SELECT ON Interviews to InterviewingDepart
GO
GRANT SELECT ON Interviewers to InterviewingDepart
GO
GRANT SELECT ON Feedback to InterviewingDepart
GO
```

```
GRANT SELECT, UPDATE ON JobOpening to InterviewingDepart;
```



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-5MGKLNU.Recruitment (DESKTOP-5MGKLNU\rfulsund (53)) - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
Recruitment
Execute
Object Explorer
Connect
AP
College
Examples
MyCollege
ProductOrders
Recruitment
Database Diagrams
Tables
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
Users
Roles
Database Roles
db_accessadmin
db_backupoperator
db_datareader
db_datawriter
db_ddladmin
db_denydatareader
db_denydatawriter
db_owner
db_securityadmin
public
Application Roles
Project2.sql - DESK...KLNU\rfulsund (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: 2022-12-11T21:37:53.2537113-05:00
100 %
Query executed successfully.
DESKTOP-5MGKLNU (15.0 RTM)
DESKTOP-5MGKLNU\rfulsu...
Recruitment
00:00:00
0 rows
Ready
Ln 819
Col 1
Ch 1
INS
Type here to search
ENG
21:37
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.