CSE 581 INTRO TO DATABASE MANAGEMENT SYSTEMS

Project 2

DECEMBER 11, 2022

SIDDHITA NIKAM
SUID - 471330539

ABSTRACT

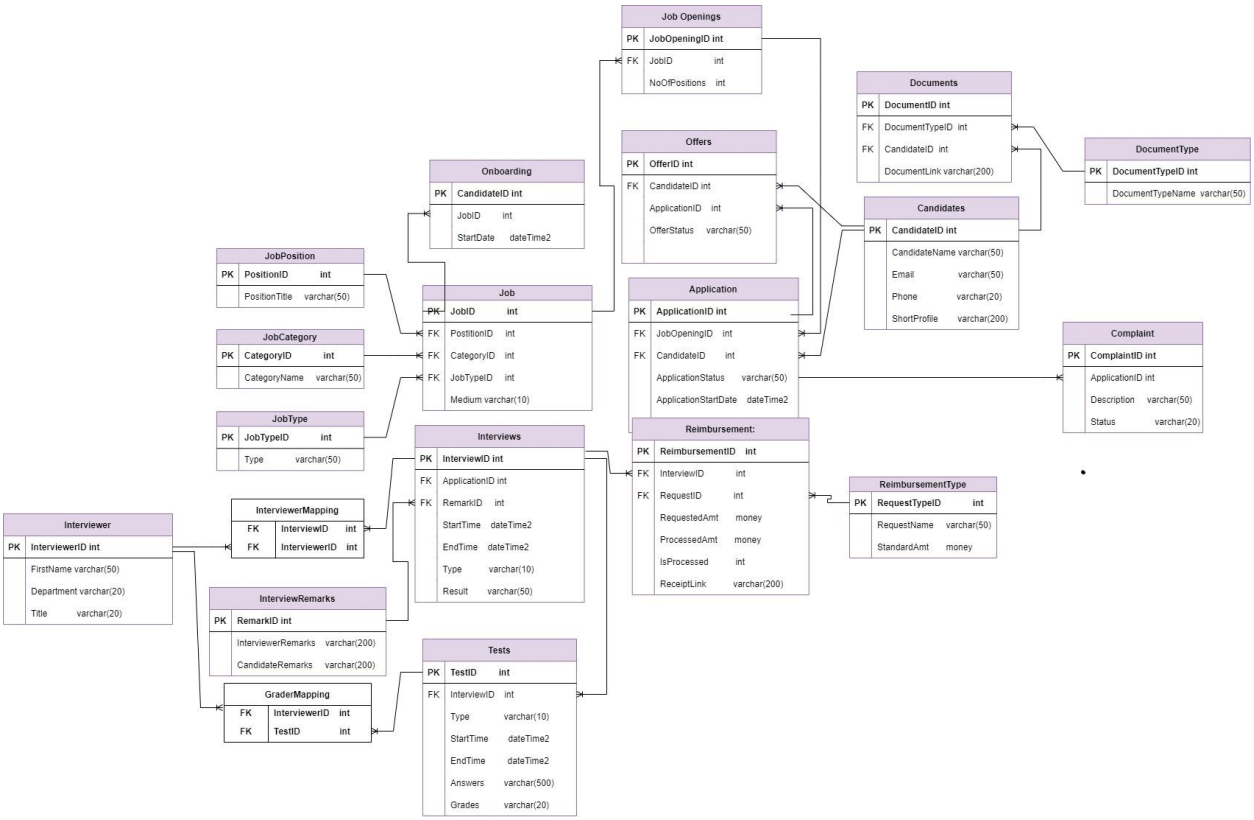
In this project, we will design, implement, and test a database for the Recruitment branch of the HR Department in a company.

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

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

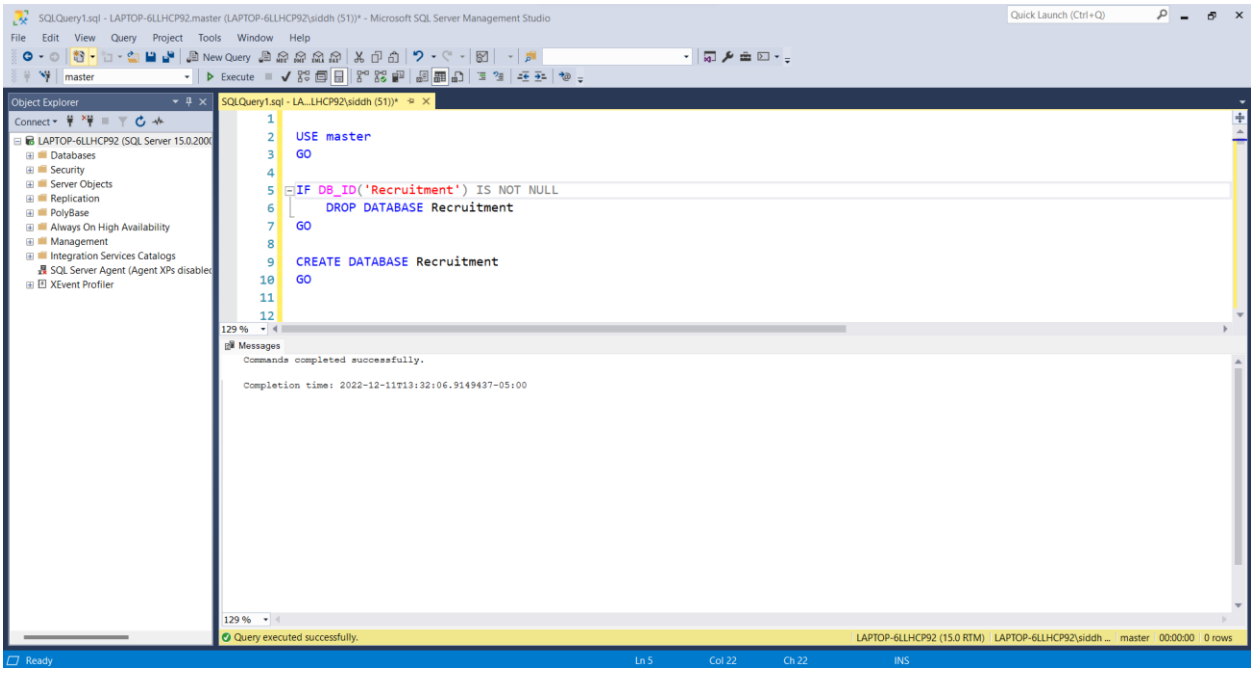
Contents

DESIGN



IMPLEMENTATION

```
USE master
GO

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

CREATE DATABASE Recruitment
GO
```



Create Table Statements:

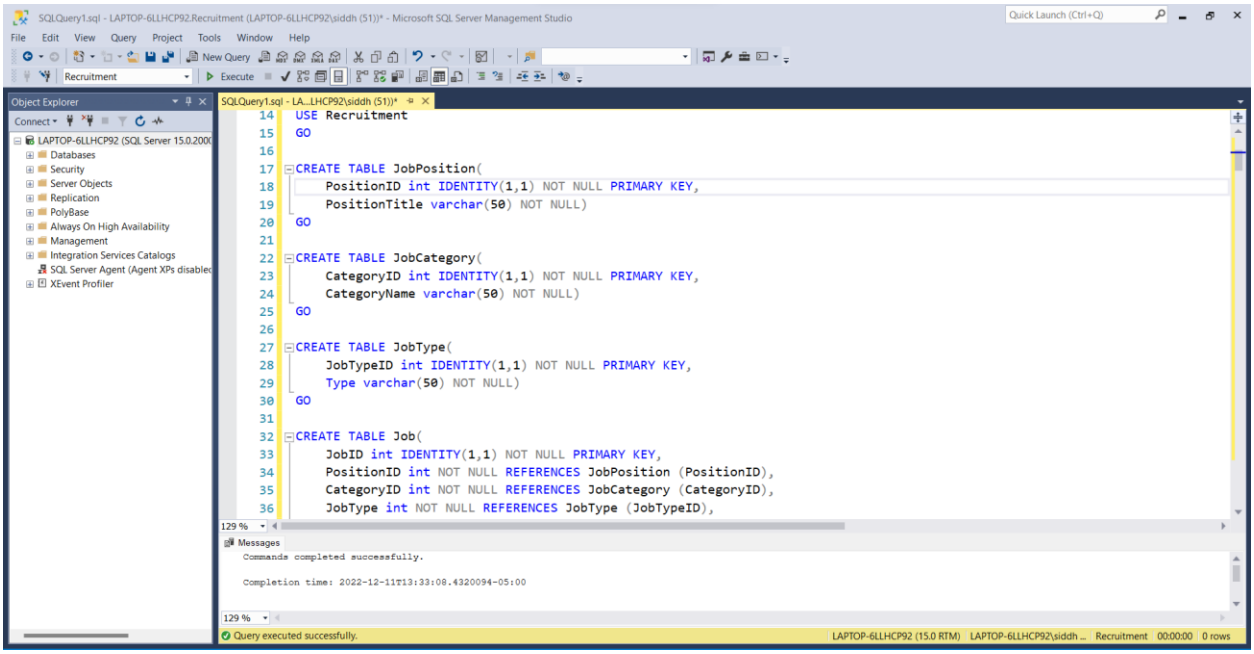
```
USE Recruitment
GO

CREATE TABLE JobPosition(
      PositionID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
      PositionTitle 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 JobType(
      JobTypeID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
      Type varchar(50) NOT NULL)
GO

CREATE TABLE Job(
      JobID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
      PositionID int NOT NULL REFERENCES JobPosition (PositionID),
      CategoryID int NOT NULL REFERENCES JobCategory (CategoryID),
      JobType int NOT NULL REFERENCES JobType (JobTypeID),
      Medium varchar(10) NOT NULL,
```

```
      CONSTRAINT mediumCheck CHECK(lower(ltrim(rtrim(Medium))) IN
('online','onsite')))
GO

CREATE TABLE JobOpenings(
      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,
      Email varchar(50) NOT NULL,
      Phone varchar(20) NOT NULL,
      ShortProfile varchar(200)  NULL
)
GO

CREATE TABLE Applications(
      ApplicationID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
      JobOpeningID int NOT NULL REFERENCES JobOpenings (JobOpeningID),
      CandidateID int NOT NULL REFERENCES Candidates (CandidateID),
      ApplicationStatus varchar(50) NOT NULL,
      ApplicationStartDate dateTime2 NOT NULL,
      LastUpdateDate dateTime2 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(50) NOT NULL,
      CONSTRAINT OfferStatusCheck
CHECK(lower(ltrim(rtrim(OfferStatus))) IN
('accepted','declined','negotiating','actionNeeded'))
)
GO

CREATE TABLE InterviewRemarks(
      RemarkID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
      InterviewerRemarks varchar(200) NOT NULL,
      CandidateRemarks varchar(200) NULL)
```

```
GO

CREATE TABLE Interviews(
	InterviewID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
	ApplicationID int NOT NULL REFERENCES
Applications(ApplicationID),
	RemarkID int NOT NULL REFERENCES InterviewRemarks (RemarkID),
	StartTime dateTime2 NOT NULL,
	EndTime dateTime2 NOT NULL,
	Type varchar(10) NOT NULL,
	Result varchar(50) NOT NULL,
	CONSTRAINT TypeCheck CHECK(lower(ltrim(rtrim(Type))) IN
('online','onsite'))
	)
GO

CREATE TABLE Interviewer(
	InterviewerID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
	Name varchar(50) NOT NULL,
	Department varchar(20) NOT NULL,
	Title varchar(20) NOT NULL)
GO

CREATE TABLE InterviewerMapping(
	InterviewID int NOT NULL REFERENCES Interviews(InterviewID),
	InterviewerID int  NOT NULL REFERENCES Interviewer(InterviewerID)
	)
GO

CREATE TABLE Tests(
	TestID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
	InterviewID int NOT NULL REFERENCES Interviews(InterviewID),
	Type varchar(10) NOT NULL,
	StartTime dateTime2 NOT NULL,
	EndTime dateTime2 NOT NULL,
	Answers varchar(500) NOT NULL,
	Grades varchar(20) NOT NULL,
	CONSTRAINT TestCheck CHECK(lower(ltrim(rtrim(Type))) IN
('online','onsite')),
	CONSTRAINT GradesCheck CHECK(lower(ltrim(rtrim(Grades))) IN
('passed','failed'))
	)
GO

CREATE TABLE GraderMapping(
	InterviewerID int NOT NULL REFERENCES Interviewer(InterviewerID),
```

```
      TestID int NOT NULL REFERENCES Tests(TestID)
      )
GO

CREATE TABLE ReimbursementType(
      RequestTypeID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
      RequestName varchar(50) NOT NULL,
      StandardAmt 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
ReimbursementType(RequestTypeID),
      RequestedAmt money DEFAULT 0,
      ProcessedAmt money DEFAULT 0,
      ReceiptLink varchar(200) NOT NULL,
      IsProcessed 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 Onboarding(
      CandidateID int NOT NULL REFERENCES Candidates(CandidateID)
PRIMARY KEY,
      JobID int NOT NULL REFERENCES Job (JobID),
      StartDate dateTime2 NOT NULL
)
GO

CREATE TABLE Complaint(
```

```
      ComplaintID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
      ApplicationID int NOT NULL REFERENCES
Applications(ApplicationID),
      Description varchar(200) NOT NULL,
      Status varchar(50) NOT NULL
)
GO
```



Insert Statements:

```
INSERT INTO JobPosition VALUES
('IT Manager'),
('Software Developer'),
('Project Manager'),
('Team Lead'),
('Senior Software Developer'),
('Associate Developer'),
('System Engineer'),
('Business Analyst'),
('Senior Business Analyst'),
('Database Engineer'),
('Database Architect'),
('QA Analyst'),
('Cybersecurity analyst');
```

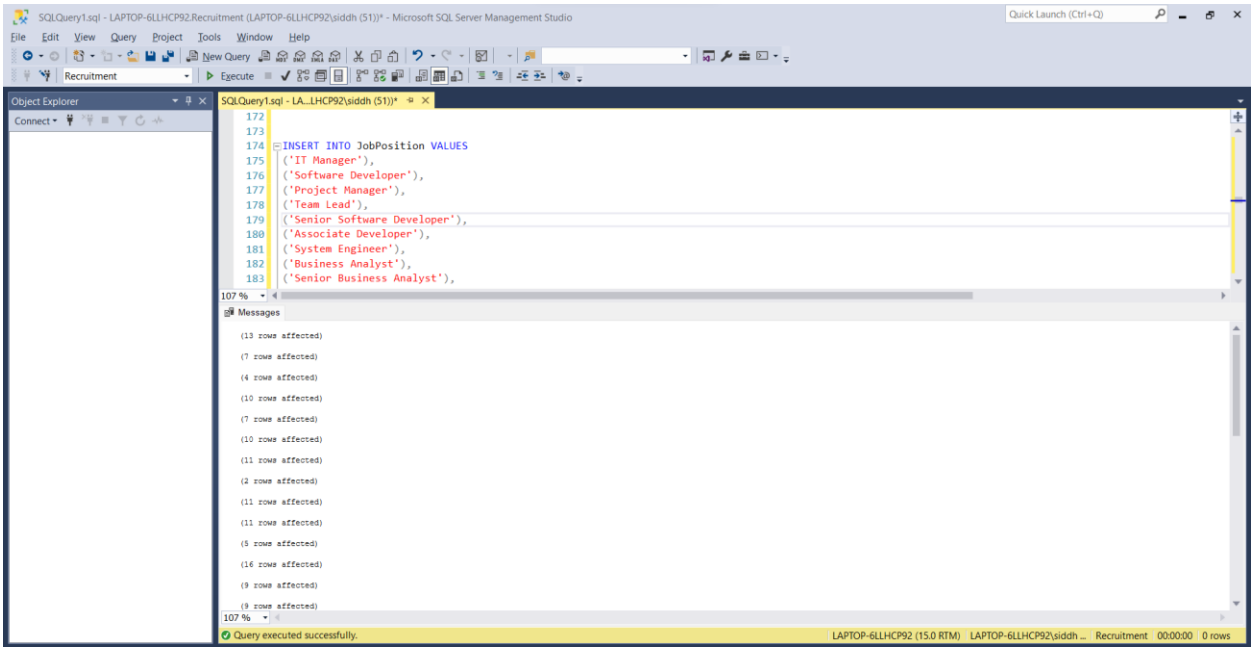
```
INSERT INTO JobCategory VALUES
('Information security'),
('QA'),
('Software development '),
('Analyst'),
('Management'),
('Database Administrator'),
('Cyber Security');

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


INSERT INTO Job VALUES
(9,4,3,'onsite'),
(6,3,1,'online'),
(1,5,2,'onsite'),
(10,6,4,'online'),
(3,5,2,'onsite'),
(2,3,1,'online'),
(13,7,4,'online'),
(8,4,3,'onsite'),
(5,3,2,'onsite'),
(12,2,1,'online');



INSERT INTO JobOpenings VALUES
(1,3),
(2,5),
(3,2),
(4,1),
(5,2),
(6,3),
(7,1);

INSERT INTO Candidates VALUES
('Siddhita Nikam','siddhita123@gmail.com','3154662586','Java Developer
5 Years Experience'),
('Jethalal Gada','jgada12@gmail.com','3157896234','Business Analyst 8
Years Experiene'),
('Aatmaram Bhide','aatmarambhide4@gmail.com','3161129882','Project
manager 15 Years Experience'),
```

```
('Hansraj Hathi','hhathi6@gmail.com','3164363530','Cyber Security
Engineer 2 years Experience'),
('Roshan Singh Sodhi','roshansingh43@gmail.com','3167597178','Java
Developer No experience'),
('Tarak Mehta','tarakmehta_23@gmail.com','3170830826','IT Manager 3
years experience'),
('Popatlal Pande','ppande24@gmail.com','3174064474','Database Engineer
no experience'),
('Babita Iyer','babitaiyer1@gmail.com','3177298122','QA Analyst 6
years experience'),
('Abdul Sheikh','abduls54@gmail.com','3180531770','Software Developer2
years experience'),
('Natwarlal Udhaiwala','natukaka45@gmail.com','3183765418','Senior
Business Analyst 7 years experience');


INSERT INTO Applications VALUES
(1,1,'Applied',GETDATE(),dateadd(w, -3, GETDATE())),
(1,3,'interview 1',GETDATE(),dateadd(w, -5, GETDATE())),
(4,5,'Applied',GETDATE(),dateadd(w, -1, GETDATE())),
(5,8,'Rejected',GETDATE(),dateadd(w, -2, GETDATE())),
(5,1,'offer extended',GETDATE(),dateadd(w, -3, GETDATE())),
(3,3,'interview 3',GETDATE(),dateadd(w, -6, GETDATE())),
(1,6,'Applied',GETDATE(),dateadd(w, -10, GETDATE())),
(2,7,'Waiting',GETDATE(),dateadd(w, -5, GETDATE())),
(4,9,'Applied',GETDATE(),dateadd(w, -1, GETDATE())),
(4,10,'Applied',GETDATE(),dateadd(w, -8, GETDATE())),
(7,5,'Onboarding',GETDATE(),dateadd(w, -3, GETDATE()));

INSERT INTO Offers VALUES
(1, 5, 'actionNeeded'),
(5, 11, 'accepted');


INSERT INTO InterviewRemarks VALUES
('Candidate seems to be good based on the 1st interview. Can proceeed
with 2nd round','Cooperative Interviewer'),
('Candidate can proceed with round 2. Candidate could answer the
questions satisfactorily in the 1st round',NULL),
('Candidate couldnt answer the 50% questions in the 2nd technical
interview. Doesnt fit for this position','Tough Interview'),
('Candidate cleared the 1st technical round.',NULL),
('Candidate cleared the 2nd technical round. Can proceed with 3rd
round of interview for behavioural questions',NULL),
('Candidate cleared the 3rd round of interview',NULL),
```

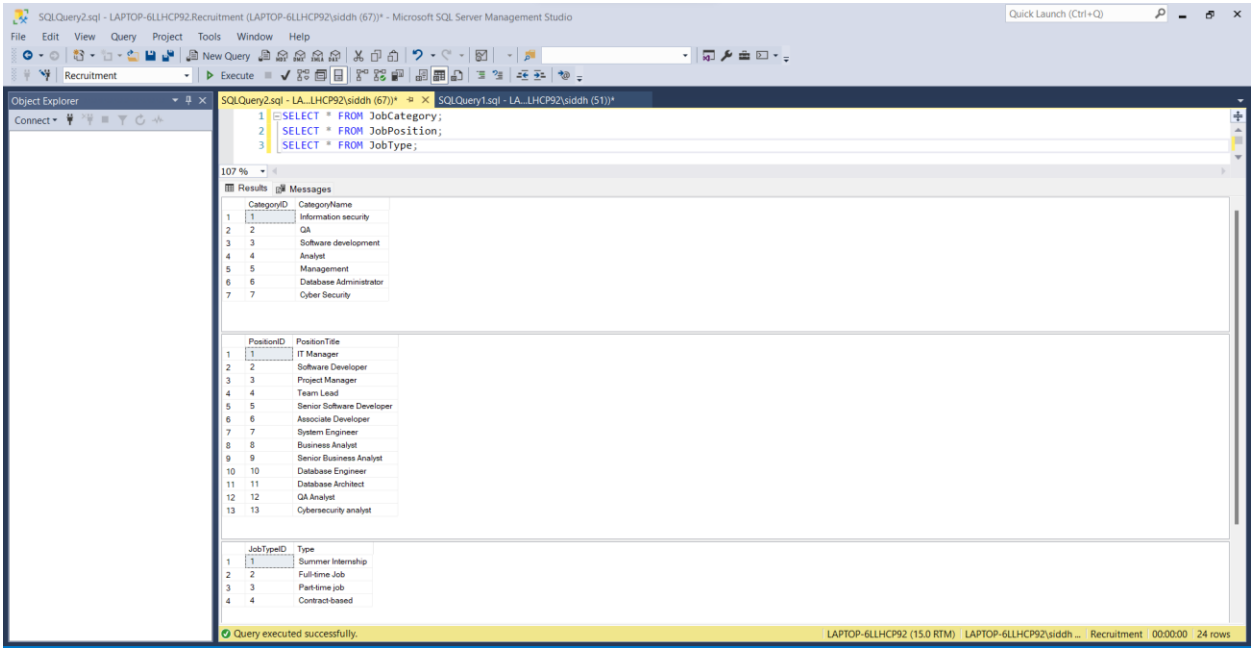
```
('Candidate is a good fit for the given requirements in the 1st
technical round.',NULL),
('Candidate cleared the 2nd round of interview. Candidate is
selected',NULL),
('Candidate is rejected in the 1st interview. Doesnt meet the
requirements', 'Rude interviewer. Asked difficult questions which were
not aligned with job position requirements'),
('Candidate did well the 1st technical round.',NULL),
('Candidate cleared the 2nd round of interview. Candidate is selected
for the given position.',NULL);

INSERT INTO Interviews VALUES
(2, 1, dateadd(hh, -2, dateadd(w, -1, GETDATE())),dateadd(hh, -1,
dateadd(w, -1, GETDATE())), 'online', 'accepted'),
(4, 2, dateadd(hh, -2, dateadd(w, -3, GETDATE())),dateadd(hh, -1,
dateadd(w, -3, GETDATE())), 'onsite', 'accepted'),
(4, 3, dateadd(hh, -2, dateadd(w, -1, GETDATE())),dateadd(hh, -1,
dateadd(w, -1, GETDATE())), 'online', 'rejected'),
(6, 4, dateadd(hh, -2, dateadd(w, -4, GETDATE())),dateadd(hh, -1,
dateadd(w, -4, GETDATE())), 'onsite', 'accepted'),
(6, 5, dateadd(hh, -2, dateadd(w, -6, GETDATE())),dateadd(hh, -1,
dateadd(w, -6, GETDATE())), 'online', 'accepted'),
(6, 6, dateadd(hh, -2, dateadd(w, -4, GETDATE())),dateadd(hh, -1,
dateadd(w, -4, GETDATE())), 'online', 'accepted'),
(5, 7, dateadd(hh, -2, dateadd(w, -3, GETDATE())),dateadd(hh, -1,
dateadd(w, -3, GETDATE())), 'onsite', 'accepted'),
(5, 8, dateadd(hh, -2, dateadd(w, -1, GETDATE())),dateadd(hh, -1,
dateadd(w, -1, GETDATE())), 'onsite', 'accepted'),
(8, 9, dateadd(hh, -2, dateadd(w, -1, GETDATE())),dateadd(hh, -1,
dateadd(w, -1, GETDATE())), 'onsite', 'rejected'),
(11, 10, dateadd(hh, -6, dateadd(w, -1, GETDATE())),dateadd(hh, -4,
dateadd(w, -1, GETDATE())), 'onsite', 'accepted'),
(11, 11, dateadd(hh, -2, dateadd(w, -1, GETDATE())),dateadd(hh, -1,
dateadd(w, -1, GETDATE())), 'onsite', 'accepted');

INSERT INTO Interviewer VALUES
('Sumeet Raghvan', 'IT', 'Project Manager'),
('Siddharth Malhotra', 'Analysts', 'Business Analyst'),
('Dilip Tahil', 'IT', 'Team Lead'),
('Atul Anand', 'Management', 'Product Manager'),
('Sachin Patil', 'IT', 'Cyber Security');

INSERT INTO InterviewerMapping VALUES
(1,1),
(1,2),
(2,5),
```

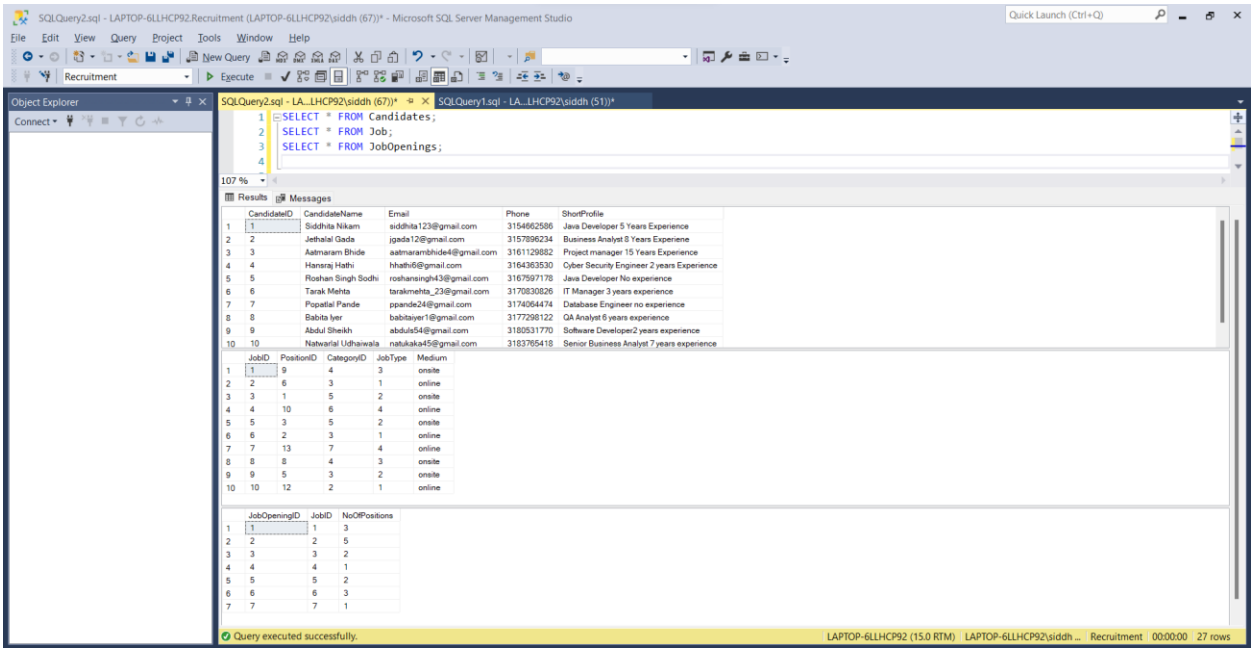
```
(3,4),
(4,3),
(4,4),
(4,1),
(5,2),
(7,4),
(7,3),
(8,5),
(9,1),
(10,1),
(6,4),
(11,3),
(11,5);

INSERT INTO Tests VALUES
(1,'online',dateadd(hh, -2, dateadd(w, -1, GETDATE())),dateadd(hh, -2,
dateadd(w, -1, GETDATE())), '1-a 2-c 3-c 4-d', 'passed'),
(2,'onsite',dateadd(hh, -2, dateadd(w, -1, GETDATE())),dateadd(hh, -1,
dateadd(w, -1, GETDATE())), '1-true 2-false', 'passed'),
(3,'online',dateadd(hh, -2, dateadd(w, -3, GETDATE())),dateadd(hh, -1,
dateadd(w, -2, GETDATE())), '1-false 2-false','failed'),
(4,'onsite',dateadd(hh, -2, dateadd(w, -1, GETDATE())),dateadd(hh, -1,
dateadd(w, -1, GETDATE())), '1-d 2-c','passed'),
(5,'online',dateadd(hh, -2, dateadd(w, -1, GETDATE())),dateadd(hh, -1,
dateadd(w, -1, GETDATE())), '1-a 2-c 3-c 4-d','passed'),
(6,'onsite',dateadd(hh, -2, dateadd(w, -1, GETDATE())),dateadd(hh, -1,
dateadd(w, -1, GETDATE())), '1-false 2-true 3-true 4-false 5-a 6-
b','passed'),
(7,'online',dateadd(hh, -7, dateadd(w, -1, GETDATE())),dateadd(hh, -1,
dateadd(w, -6, GETDATE())), '2-a 3-e','passed'),
(8,'onsite',dateadd(hh, -2, dateadd(w, -1, GETDATE())),dateadd(hh, -1,
dateadd(w, -1, GETDATE())), '1-a 2-c 3-c 4-d','passed'),
(9,'online',dateadd(hh, -2, dateadd(w, -1, GETDATE())),dateadd(hh, -1,
dateadd(w, -1, GETDATE())), '','failed');

INSERT INTO GraderMapping VALUES
(3, 1),
(5, 2),
(3, 3),
(4, 4),
(2, 5),
(1, 6),
(3, 7),
(4, 8),
(2, 9);
```

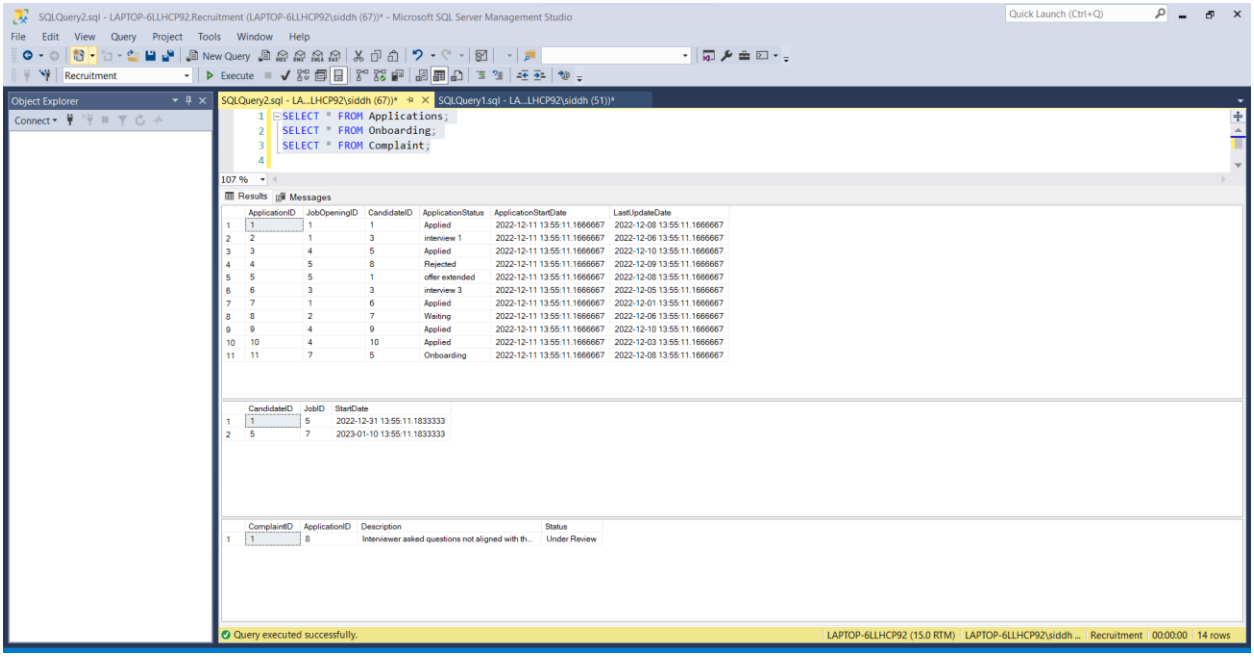
```
INSERT INTO ReimbursementType VALUES
('Airline reservation details', 700.00),
('Hotel reservation details', 300.00),
('Car Rental',150.00),
('Food Expenses',100.00);

INSERT INTO Reimbursement VALUES
(2, 1, 500.00, 500.00,  'emirates.pdf', 1),
(7, 4, 100.00, 100.00, 'miradoor.docx', 1),
(4, 3, 100.00, 100.00, 'zipcar.docx',1),
(8, 1, 700.00, 700.00, 'etihad.pdf', 1),
(2, 2, 350.00, 300.00, 'dragonfly.pdf', 1),
(4, 4, 100.00, 100.00,'sandeep.docx',1),
(2, 4, 100.00, 0,  'restBill.pdf',0),
(9, 1, 600.00, 600.00, 'airindia.docx',1),
(10, 2, 300.00, 300.00, '', 1),
(9, 2, 350.00, 0, 'ranch.docx', 0),
(9, 4, 50.00, 50.00, 'chipotle.pdf', 1),
(8, 2, 200.00, 200.00, 'fivestar.pdf', 1);

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

INSERT INTO Documents VALUES
(1, 1, 'cv.pdf'),
(2, 1, 'cv_developer.pdf'),
(2, 2, 'rl_developer.pdf'),
(3, 1, 'cv3.pdf'),
(4, 1, 'cv_manager.pdf'),
(5, 1, 'cv_sd.pdf'),
(6, 3, 'cover1.pdf'),
(8, 1, 'resume_BA.pdf'),
(8, 2, 'rl_BA.pdf'),
(9, 1, 'resume3.pdf'),
(10, 1, 'resume.pdf'),
(8, 3, 'cover2.pdf');

INSERT INTO Complaint VALUES
(8, 'Interviewer asked questions not aligned with the requirements.
Very Rude behaviour. ', 'Under Review');

INSERT INTO Onboarding VALUES
(1, 5, dateadd(w, 20, GETDATE())),
(5, 7, dateadd(w, 30, GETDATE()));
```

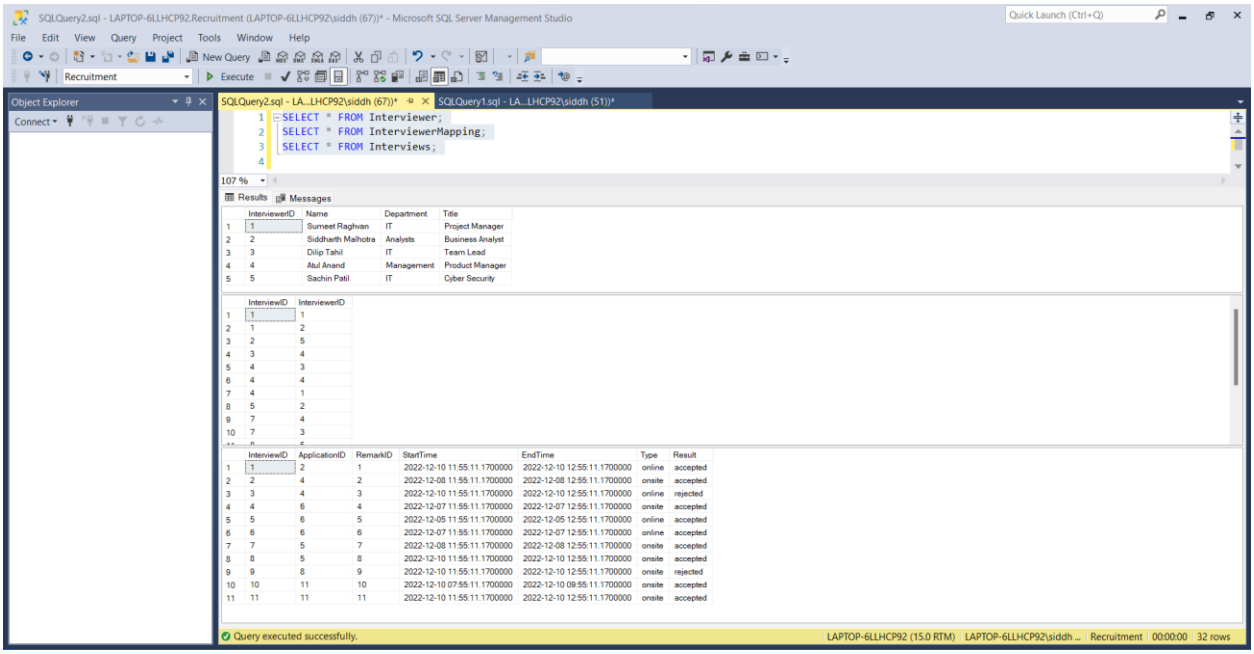


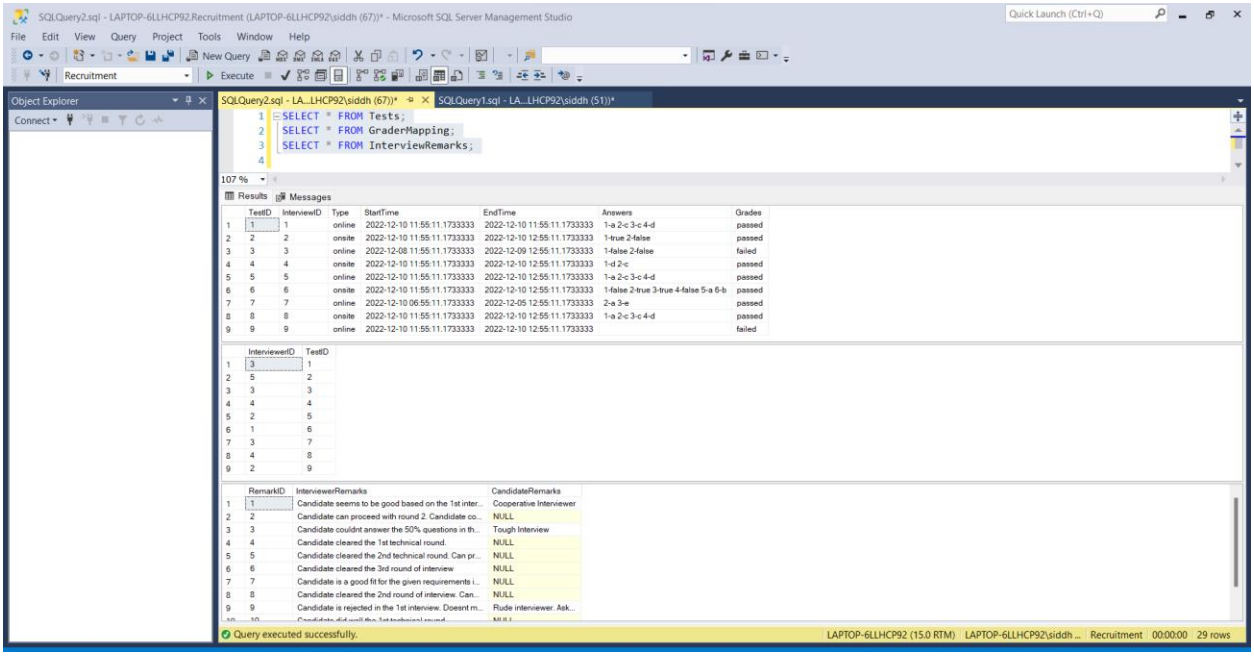
Select Statements

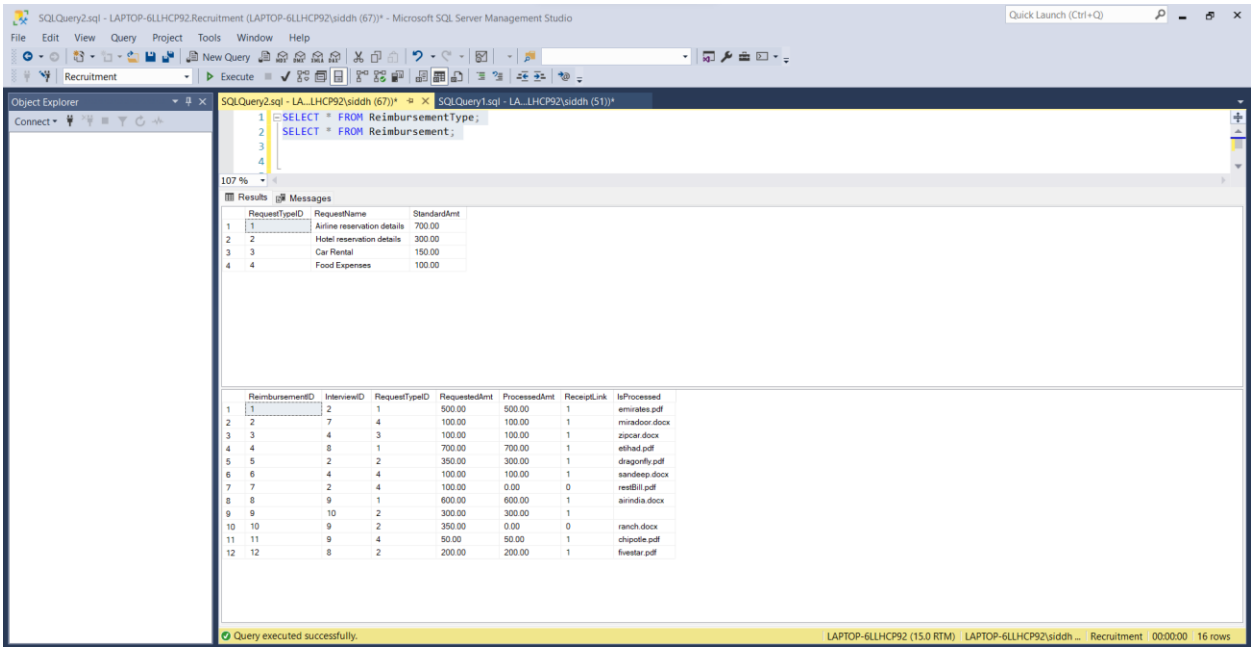


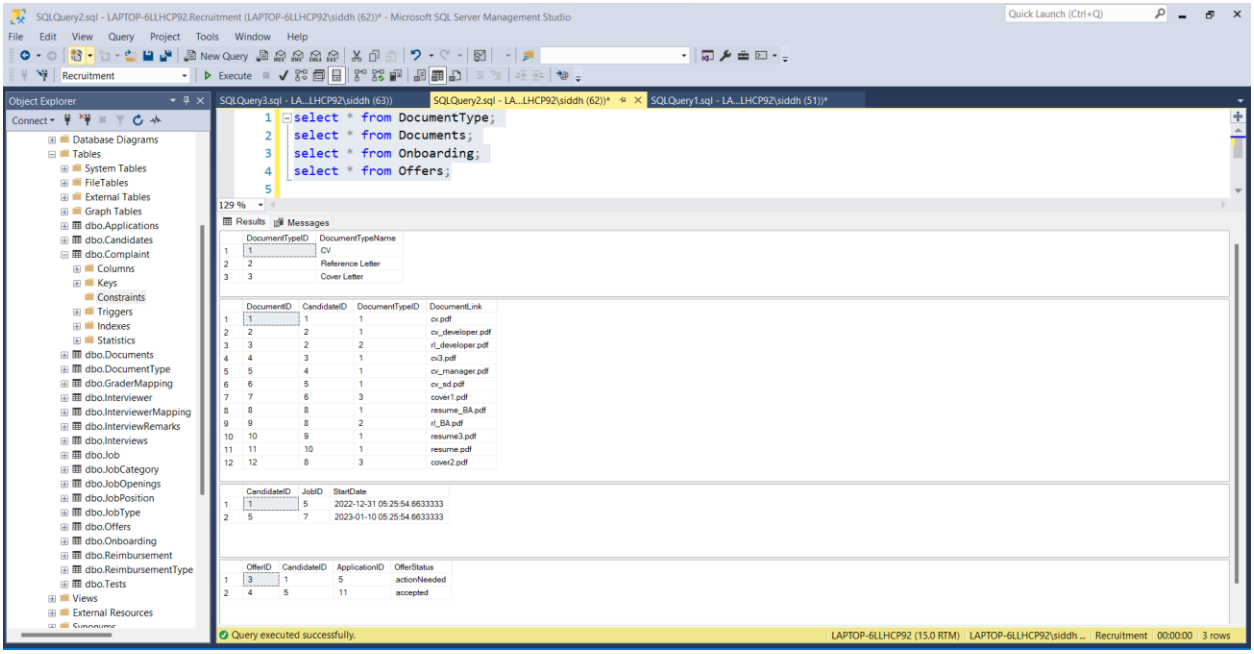

SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\siddh (67))* - Microsoft SQL Server Management Studio
SELECT * FROM Candidates;
SELECT * FROM Job;
SELECT * FROM JobOpenings;
Query executed successfully.


SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\siddh (67))* - Microsoft SQL Server Management Studio
SELECT * FROM Applications;
SELECT * FROM Onboarding;
SELECT * FROM Complaint;
Query executed successfully.


SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\siddh (67))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Recruitment
Execute
Object Explorer
Connect
SQLQuery2.sql - LA...LHCP92\siddh (67))*
SQLQuery1.sql - LA...LHCP92\siddh (51))*
SELECT * FROM Interviewer;
SELECT * FROM InterviewerMapping;
SELECT * FROM Interviews;
107 %
Results
Messages
InterviewerID Name Department Title
1 Sumeet Raghvan IT Project Manager
2 Siddharth Malhotra Analysts Business Analyst
3 Dilip Tahil IT Team Lead
4 Atul Anand Management Product Manager
5 Sachin Patil IT Cyber Security
InterviewID InterviewerID
1 1
1 2
2 5
3 4
4 3
4 4
4 1
5 2
7 4
7 3
InterviewID ApplicationID RemarkID StartTime EndTime Type Result
1 2 1 2022-12-10 11:55:11.1700000 2022-12-10 12:55:11.1700000 online accepted
2 4 2 2022-12-08 11:55:11.1700000 2022-12-08 12:55:11.1700000 onsite accepted
3 4 3 2022-12-10 11:55:11.1700000 2022-12-10 12:55:11.1700000 online rejected
4 6 4 2022-12-07 11:55:11.1700000 2022-12-07 12:55:11.1700000 onsite accepted
5 6 5 2022-12-05 11:55:11.1700000 2022-12-05 12:55:11.1700000 online accepted
6 6 6 2022-12-07 11:55:11.1700000 2022-12-07 12:55:11.1700000 online accepted
7 5 7 2022-12-08 11:55:11.1700000 2022-12-08 12:55:11.1700000 onsite accepted
8 5 8 2022-12-10 11:55:11.1700000 2022-12-10 12:55:11.1700000 onsite accepted
9 8 9 2022-12-10 11:55:11.1700000 2022-12-10 12:55:11.1700000 onsite rejected
10 11 10 2022-12-10 07:55:11.1700000 2022-12-10 09:55:11.1700000 onsite accepted
11 11 11 2022-12-10 11:55:11.1700000 2022-12-10 12:55:11.1700000 onsite accepted
Query executed successfully.
LAPTOP-6LLHCP92 (15.0 RTM) LAPTOP-6LLHCP92\siddh ... Recruitment 00:00:00 32 rows


SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\siddh (67))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Recruitment
Execute
Object Explorer
Connect
SQLQuery2.sql - LA...LHCP92\siddh (67))*
SQLQuery1.sql - LA...LHCP92\siddh (51))*
SELECT * FROM Tests;
SELECT * FROM GraderMapping;
SELECT * FROM InterviewRemarks;
107 %
Results
Messages
TestID InterviewID Type StartTime EndTime Answers Grades
1 1 online 2022-12-10 11:55:11.1733333 2022-12-10 11:55:11.1733333 1-a 2-c 3-c 4-d passed
2 2 onsite 2022-12-10 11:55:11.1733333 2022-12-10 12:55:11.1733333 1-true 2-false passed
3 3 online 2022-12-08 11:55:11.1733333 2022-12-09 12:55:11.1733333 1-false 2-false failed
4 4 onsite 2022-12-10 11:55:11.1733333 2022-12-10 12:55:11.1733333 1-d 2-c passed
5 5 online 2022-12-10 11:55:11.1733333 2022-12-10 12:55:11.1733333 1-a 2-c 3-c 4-d passed
6 6 onsite 2022-12-10 11:55:11.1733333 2022-12-10 12:55:11.1733333 1-false 2-true 3-true 4-false 5-a 6-b passed
7 7 online 2022-12-10 06:55:11.1733333 2022-12-05 12:55:11.1733333 2-a 3-e passed
8 8 onsite 2022-12-10 11:55:11.1733333 2022-12-10 12:55:11.1733333 1-a 2-c 3-c 4-d passed
9 9 online 2022-12-10 11:55:11.1733333 2022-12-10 12:55:11.1733333 failed
InterviewerID TestID
3 1
5 2
3 3
4 4
2 5
1 6
3 7
4 8
2 9
RemarkID InterviewerRemarks CandidateRemarks
1 Candidate seems to be good based on the 1st inter... Cooperative Interviewer
2 Candidate can proceed with round 2. Candidate co... NULL
3 Candidate couldnt answer the 50% questions in th... Tough Interview
4 Candidate cleared the 1st technical round. NULL
5 Candidate cleared the 2nd technical round. Can pr... NULL
6 Candidate cleared the 3rd round of interview NULL
7 Candidate is a good fit for the given requirements i... NULL
8 Candidate cleared the 2nd round of interview. Can... NULL
9 Candidate is rejected in the 1st interview. Doesnt m... Rude interviewer. Ask...
Query executed successfully.
LAPTOP-6LLHCP92 (15.0 RTM) LAPTOP-6LLHCP92\siddh ... Recruitment 00:00:00 29 rows


SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\siddh (67))* - Microsoft SQL Server Management Studio
SELECT * FROM ReimbursementType;
SELECT * FROM Reimbursement;
Query executed successfully.

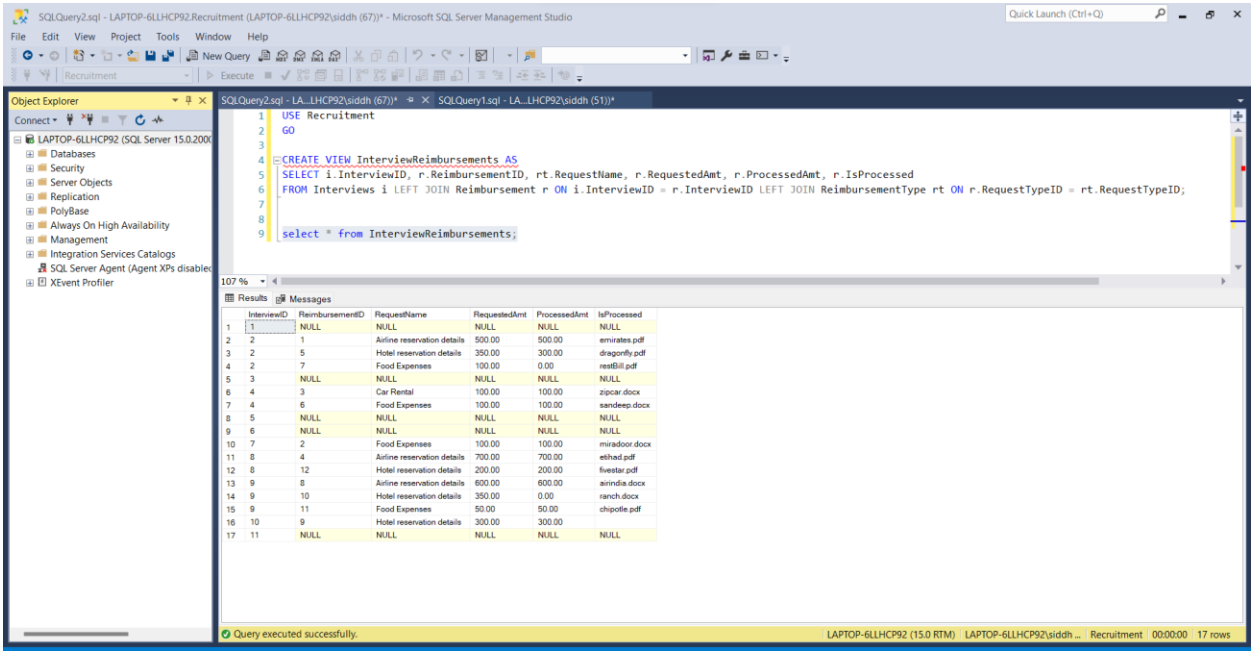

SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\siddh (62))* - Microsoft SQL Server Management Studio
select * from DocumentType;
select * from Documents;
select * from Onboarding;
select * from Offers;
Query executed successfully.

TESTING

VIEWS

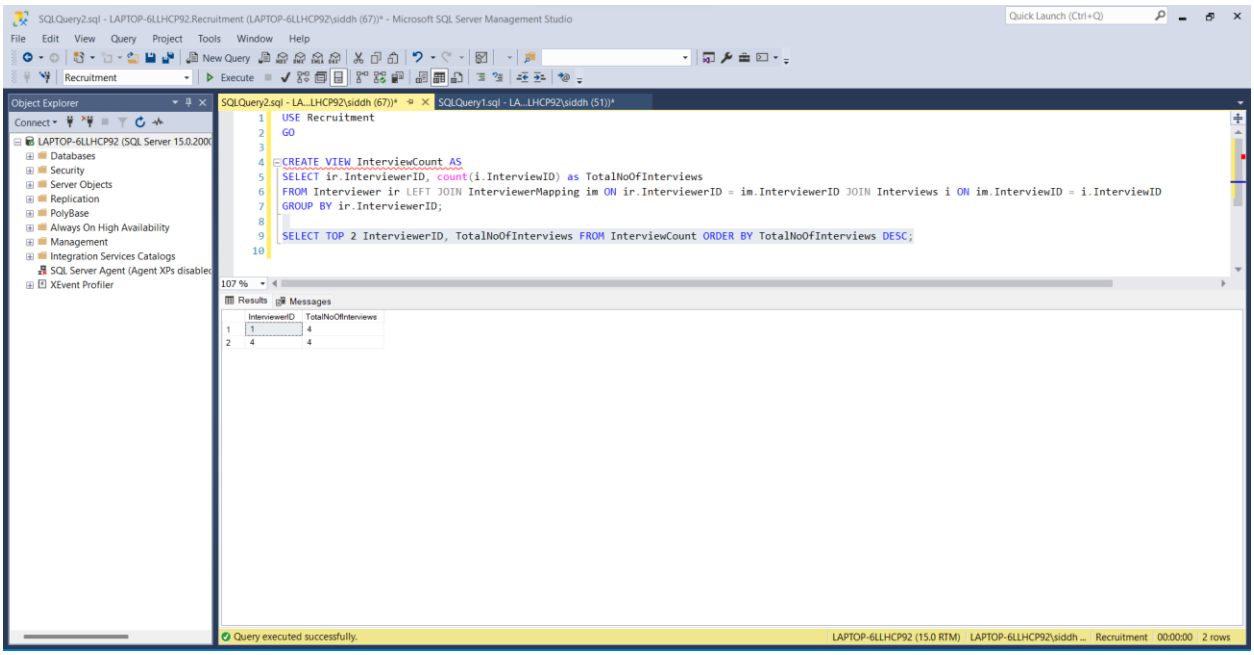
1. Write a view named `InterviewReimbursements` that returns six columns: `InterviewID, ReimbursementID, RequestName, RequestedAmt, ProcessedAmt, IsProcessed`. Then, write a SELECT statement that returns all the columns in the view. The view will return reimbursements done against every interview.

```
USE Recruitment
GO

CREATE VIEW InterviewReimbursements AS
SELECT i.InterviewID, r.ReimbursementID, rt.RequestName,
r.RequestedAmt, r.ProcessedAmt, r.IsProcessed
FROM Interviews i LEFT JOIN Reimbursement r ON i.InterviewID =
r.InterviewID LEFT JOIN ReimbursementType rt ON r.RequestTypeID =
rt.RequestTypeID;


select * from InterviewReimbursements;
```



2. Write a view named `InterviewCount` that returns two columns: `InterviewerID, TotalNoOfInterviews. This will give no of interviews taken by each interviewer` Then, write a SELECT statement that returns all the columns in the view. Show the top 2 interviewers who have taken max number of interviews.

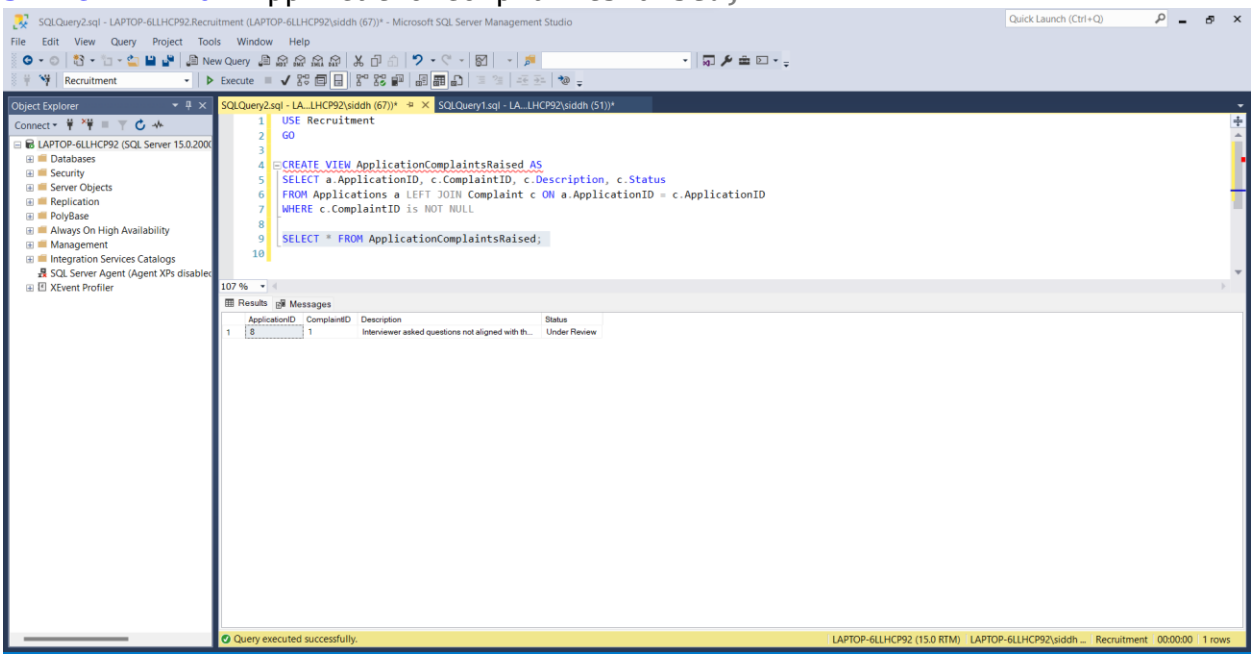
```
USE Recruitment
GO

CREATE VIEW InterviewCount AS
SELECT ir.InterviewerID, count(i.InterviewID) as
TotalNoOfInterviews
FROM Interviewer ir LEFT JOIN InterviewerMapping im ON
ir.InterviewerID = im.InterviewerID JOIN Interviews i ON
im.InterviewID = i.InterviewID
GROUP BY ir.InterviewerID;

SELECT TOP 2 InterviewerID, TotalNoOfInterviews FROM
InterviewCount ORDER BY TotalNoOfInterviews DESC;
```



3. Write a view named ApplicationComplaintsRaised that returns application against which complaints were raised. Show the applications for which the complaints are raised also show the complaint details.

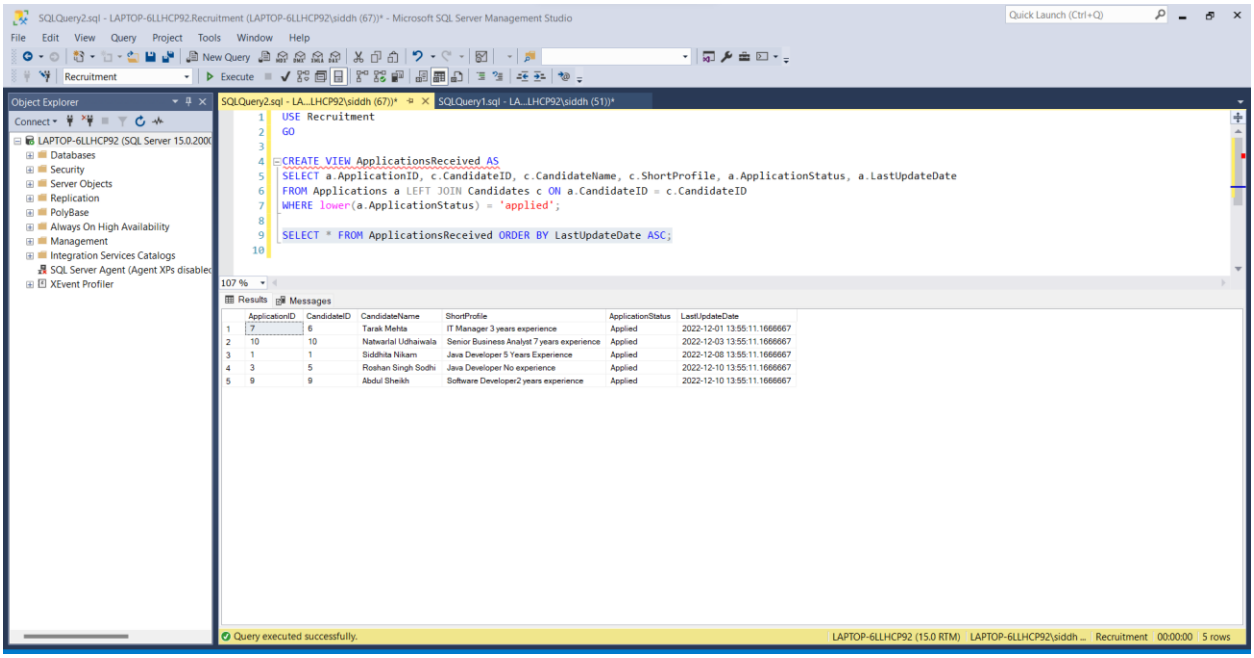
```
USE Recruitment
GO

CREATE VIEW ApplicationComplaintsRaised AS
SELECT a.ApplicationID, c.ComplaintID, c.Description, c.Status
FROM Applications a LEFT JOIN Complaint c ON a.ApplicationID =
c.ApplicationID
WHERE c.ComplaintID is NOT NULL
```

```
SELECT * FROM ApplicationComplaintsRaised;
```



4. Write a view named ApplicationsReceived that returns applications which are received by the company(applicants having applied status). Show the applications and the candidates' details with the order of their applications(First Come First Serve basis).

```
USE Recruitment
GO

CREATE VIEW ApplicationsReceived AS
SELECT a.ApplicationID, c.CandidateID, c.CandidateName,
c.ShortProfile, a.ApplicationStatus, a.LastUpdateDate
FROM Applications a LEFT JOIN Candidates c ON a.CandidateID =
c.CandidateID
WHERE lower(a.ApplicationStatus) = 'applied';

SELECT * FROM ApplicationsReceived ORDER BY LastUpdateDate ASC;
```

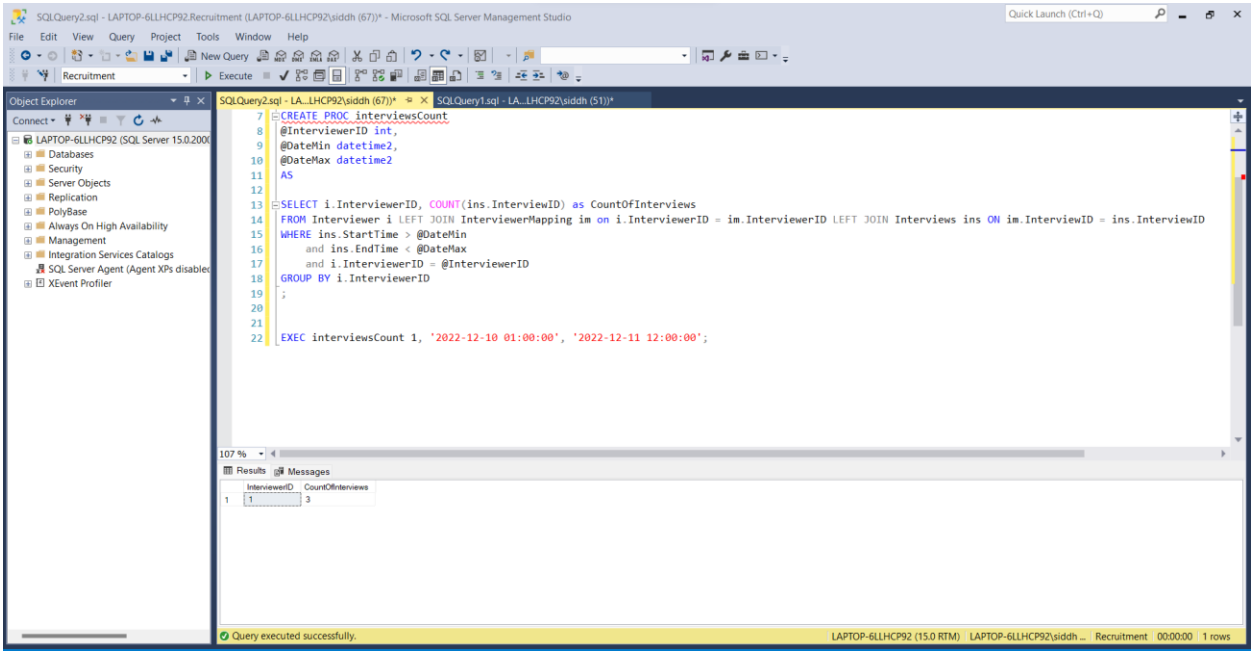


STORED PROCEDURES

1. Create a stored procedure named `interviewsCount` that accepts three parameters. The procedure should return a result set consisting of Number of interviews taken by given interviewer between provided date range.
   Interviewer with ID 1 has taken total 4 interviews but only 3 lie within the given date range.
   Query :

```
CREATE PROC interviewsCount
@InterviewerID int,
@DateMin datetime2,
@DateMax datetime2
AS

SELECT i.InterviewerID, COUNT(ins.InterviewID) as
CountOfInterviews
FROM Interviewer i LEFT JOIN InterviewerMapping im on
i.InterviewerID = im.InterviewerID LEFT JOIN Interviews ins ON
im.InterviewID = ins.InterviewID
WHERE ins.StartTime > @DateMin
      and ins.EndTime < @DateMax
      and i.InterviewerID = @InterviewerID
GROUP BY i.InterviewerID
;
```

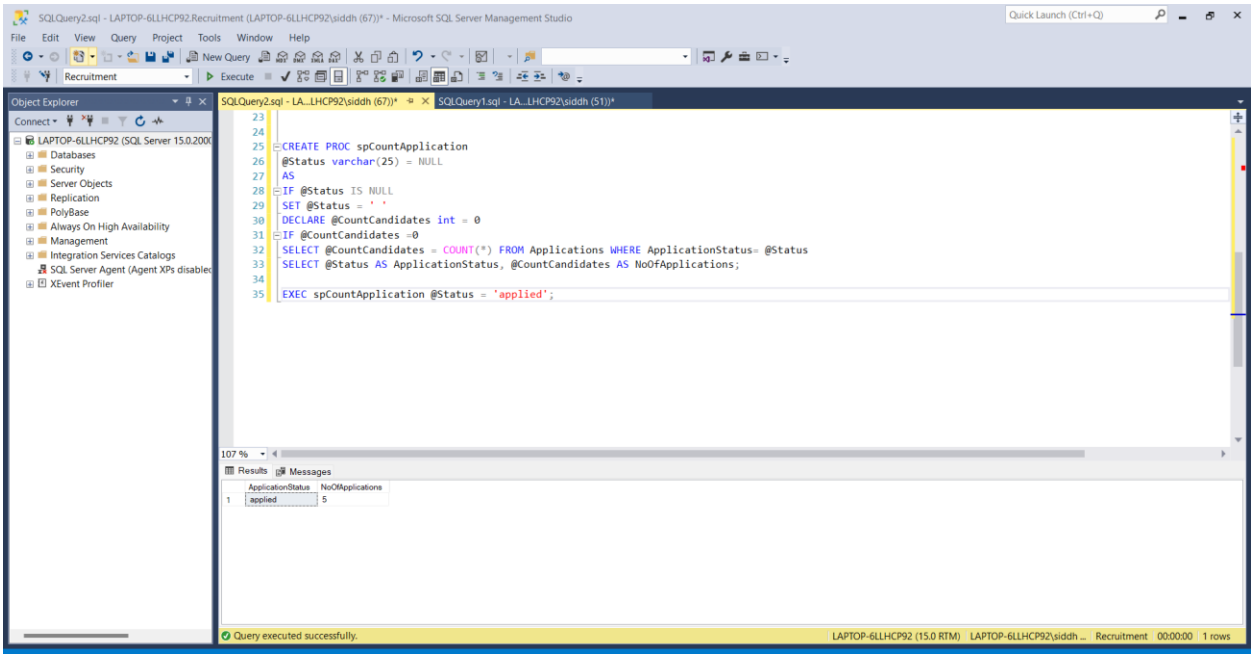
```
EXEC interviewsCount 1, '2022-12-10 01:00:00', '2022-12-11
12:00:00';
```



2. Create a stored procedure named `spCountApplication` that accepts status. The procedure should return a result set consisting of Number of candidates having the given status.
   Database has 5 applicants with applied status.

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

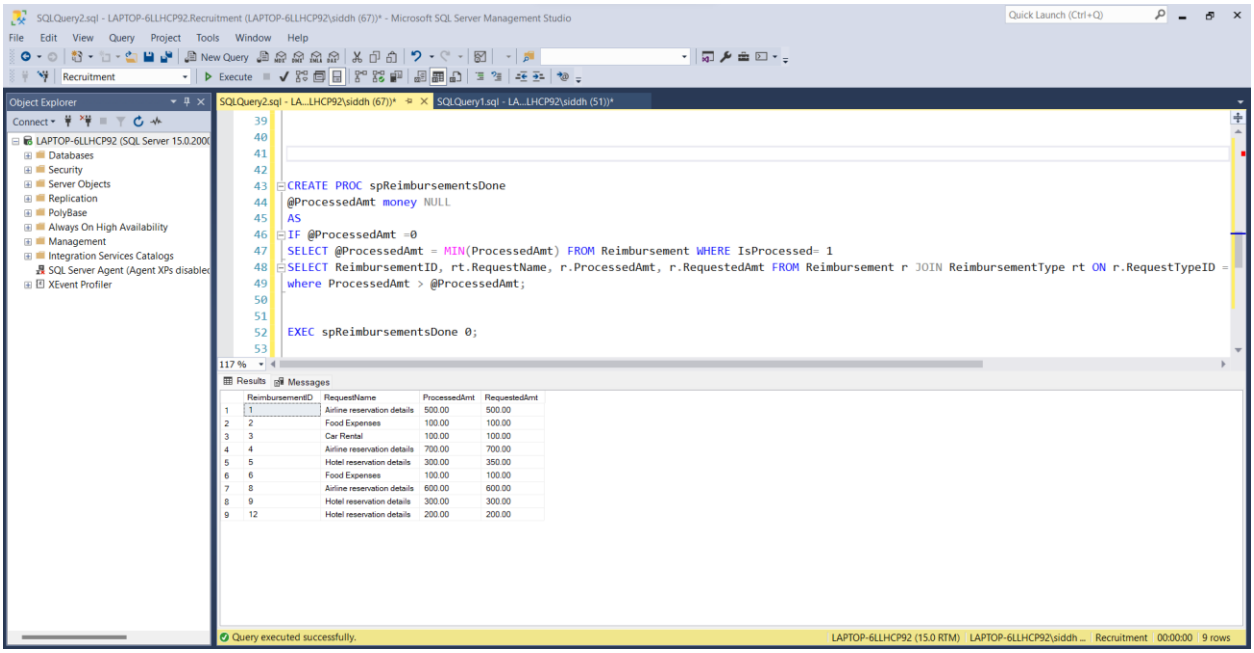
```
EXEC spCountApplication @Status = 'applied';
```



3. Create a stored procedure named spReimbursementsDone that accepts processedAmt. The procedure should return a result set consisting of processed Reimbursements which have greater value than the provided one.

```
CREATE PROC spReimbursementsDone
@ProcessedAmt money NULL
AS
IF @ProcessedAmt =0
SELECT @ProcessedAmt = MIN(ProcessedAmt) FROM Reimbursement WHERE
IsProcessed= 1
SELECT ReimbursementID, rt.RequestName, r.ProcessedAmt,
r.RequestedAmt FROM Reimbursement r JOIN ReimbursementType rt ON
r.RequestTypeID = rt.RequestTypeID
where ProcessedAmt > @ProcessedAmt;
```

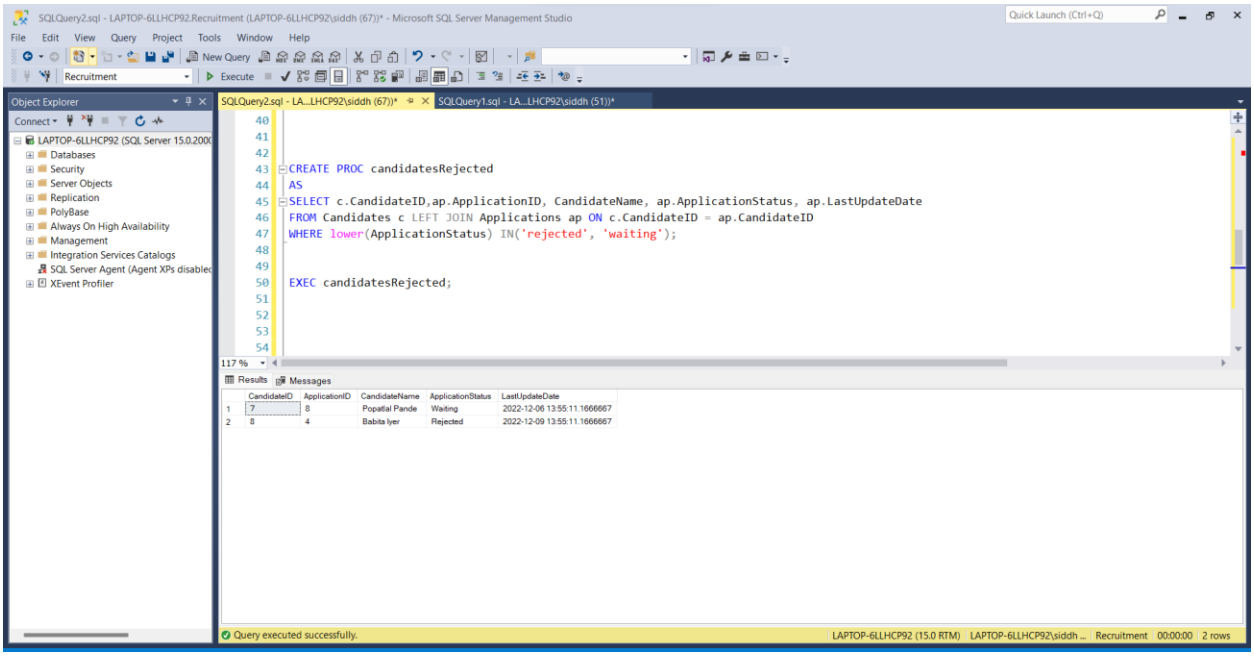
```
EXEC spReimbursementsDone 0;
```



4. Create a stored procedure named `candidatesRejected` which lists down all the candidates who are rejected at any stage.
   Procedure with no parameters passed. Just listing down candidates having rejected/waiting status.

```
CREATE PROC candidatesRejected
AS
SELECT c.CandidateID,ap.ApplicationID, CandidateName,
ap.ApplicationStatus, ap.LastUpdateDate
FROM Candidates c LEFT JOIN Applications ap ON c.CandidateID =
ap.CandidateID
WHERE lower(ApplicationStatus) IN('rejected', 'waiting');
```

```
EXEC candidatesRejected;
```

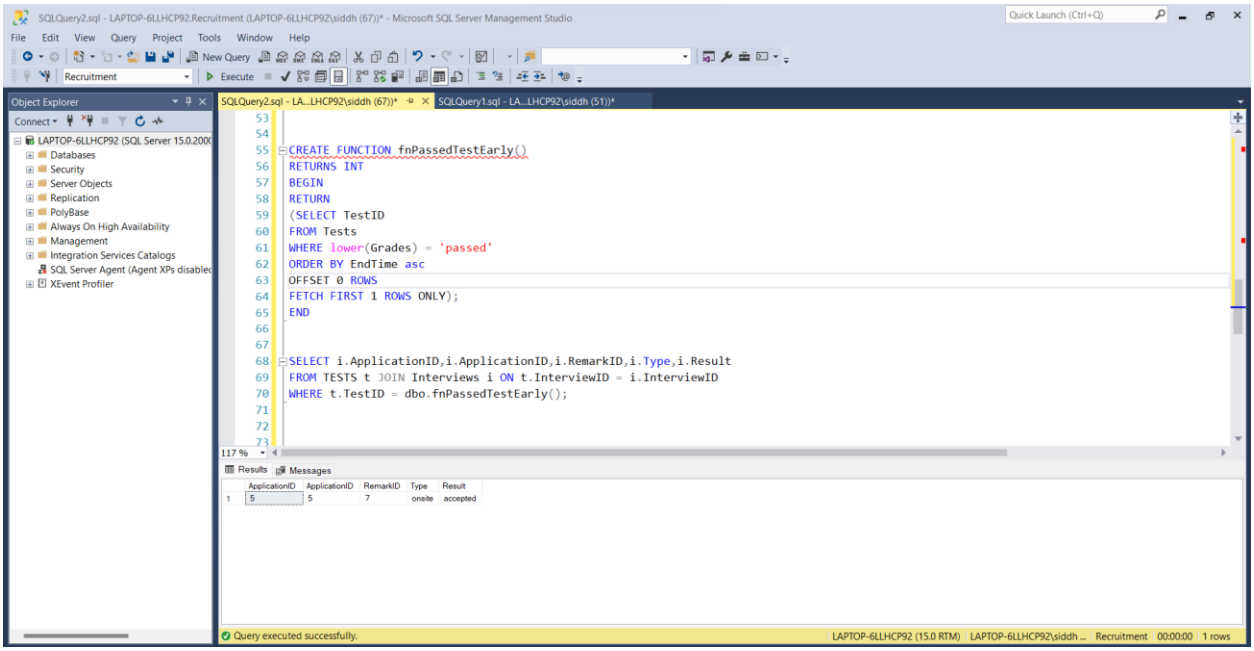


USER DEFINED FUNCTIONS

1. Create a scalar-valued function named `fnPassedTestEarly` that returns the TestID of the test with passed grades and which was finished earliest of all.

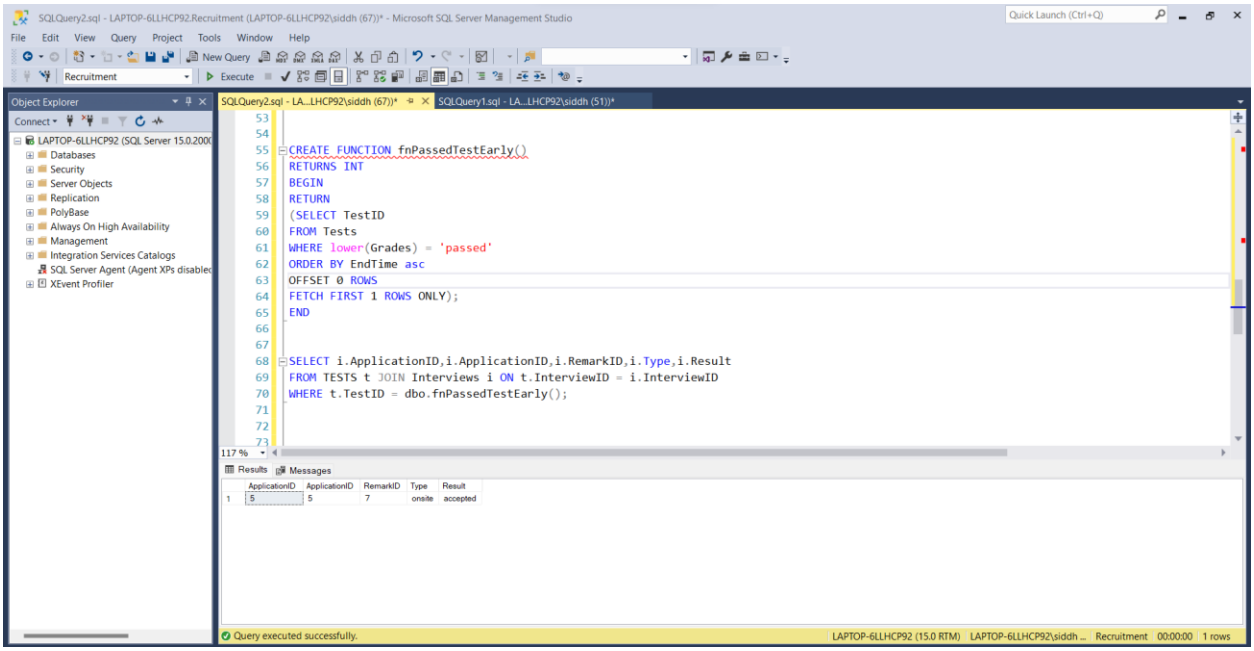
```
CREATE FUNCTION fnPassedTestEarly()
RETURNS INT
BEGIN
RETURN
(SELECT TestID
FROM Tests
WHERE lower(Grades) = 'passed'
ORDER BY EndTime asc
OFFSET 0 ROWS
FETCH FIRST 1 ROWS ONLY);
END
```

```
SELECT i.ApplicationID,i.ApplicationID,i.RemarkID,i.Type,i.Result
FROM TESTS t JOIN Interviews i ON t.InterviewID = i.InterviewID
WHERE t.TestID = dbo.fnPassedTestEarly();
```



2. Create a scalar-valued function named `fnPassedTestEarly` that returns the latest rejected candidate.

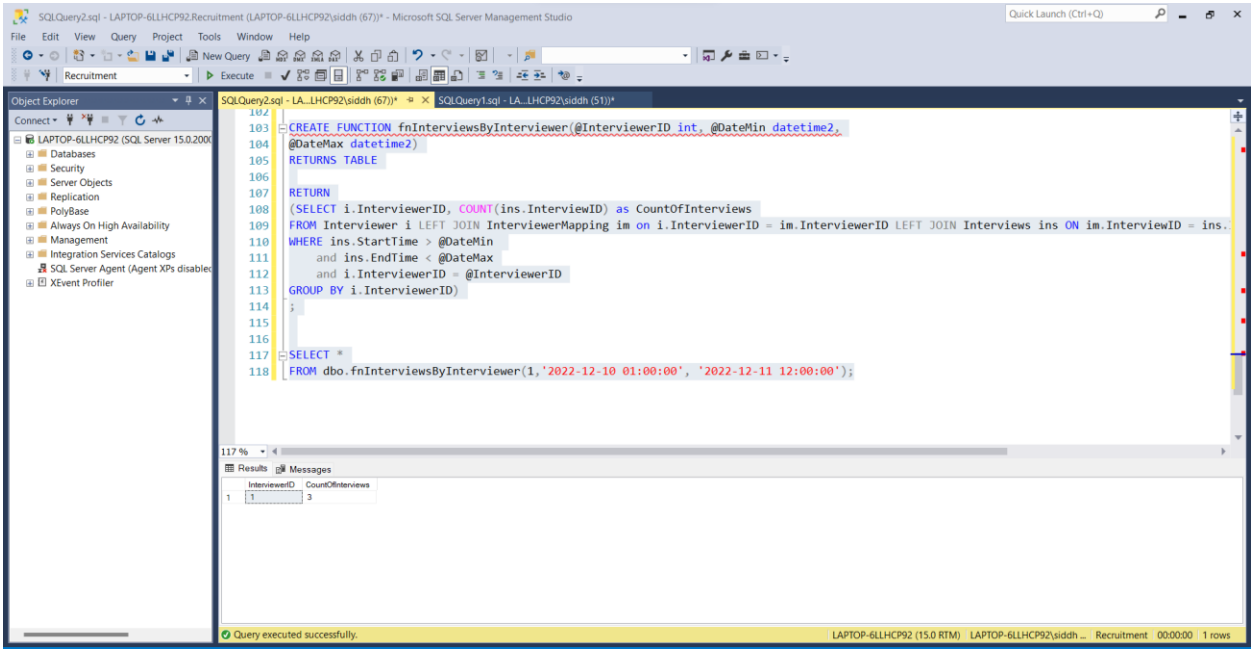
```
CREATE FUNCTION fnRejectedCandidates1()
RETURNS INT
BEGIN
RETURN
(SELECT CandidateID
FROM Applications ap
WHERE lower(ApplicationStatus) IN('rejected', 'waiting')
ORDER BY LastUpdateDate DESC
OFFSET 0 ROWS
FETCH FIRST 1 ROWS ONLY);
END

SELECT * FROM Candidates
WHERE CandidateID = dbo.fnRejectedCandidates1();
```



3. Create a scalar-valued function named `fnInterviewsByInterviewer` similar to stored procedure - `interviewsCount`. The result should display the interviews taken by interviewer between the date range.

```
CREATE FUNCTION fnInterviewsByInterviewer(@InterviewerID int,
@DateMin datetime2,
@DateMax datetime2)
RETURNS TABLE

RETURN
(SELECT i.InterviewerID, COUNT(ins.InterviewID) as
CountOfInterviews
FROM Interviewer i LEFT JOIN InterviewerMapping im on
i.InterviewerID = im.InterviewerID LEFT JOIN Interviews ins ON
im.InterviewID = ins.InterviewID
WHERE ins.StartTime > @DateMin
      and ins.EndTime < @DateMax
      and i.InterviewerID = @InterviewerID
GROUP BY i.InterviewerID)
;


SELECT *
```

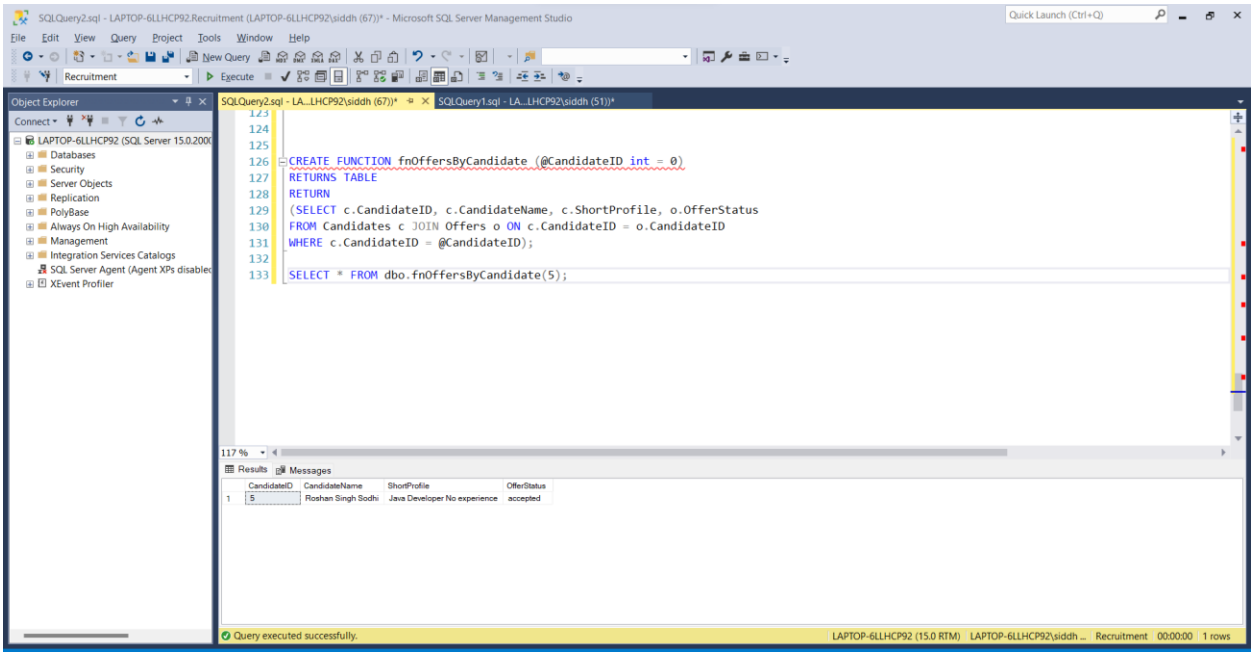
```
FROM dbo.fnInterviewsByInterviewer(1,'2022-12-10 01:00:00',
'2022-12-11 12:00:00');
```



4. Create a scalar-valued function named `fnOffersByCandidate` which will show the candidate's information, who has received the offer

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

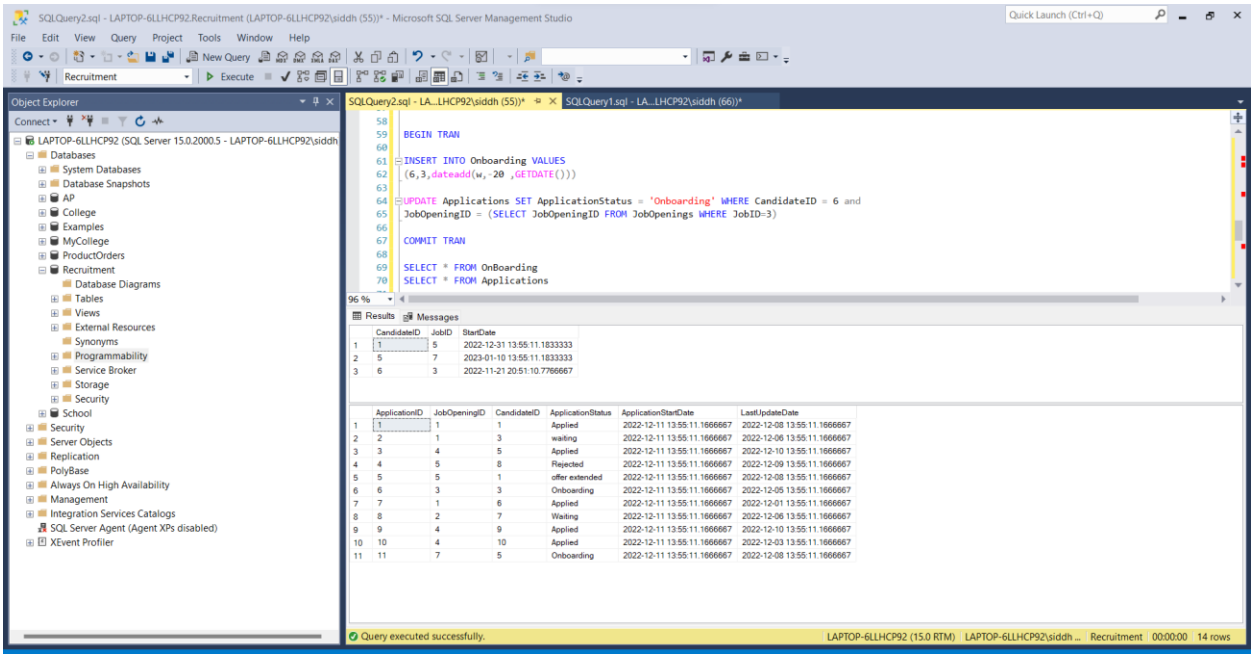
```
SELECT * FROM dbo.fnOffersByCandidate(5);
```



TRANSACTIONS:

1. Write a set of action queries coded as a transaction to change status of an application to onboarding as we get entry in Onboarding table. Use SELECT statement to verify the results.

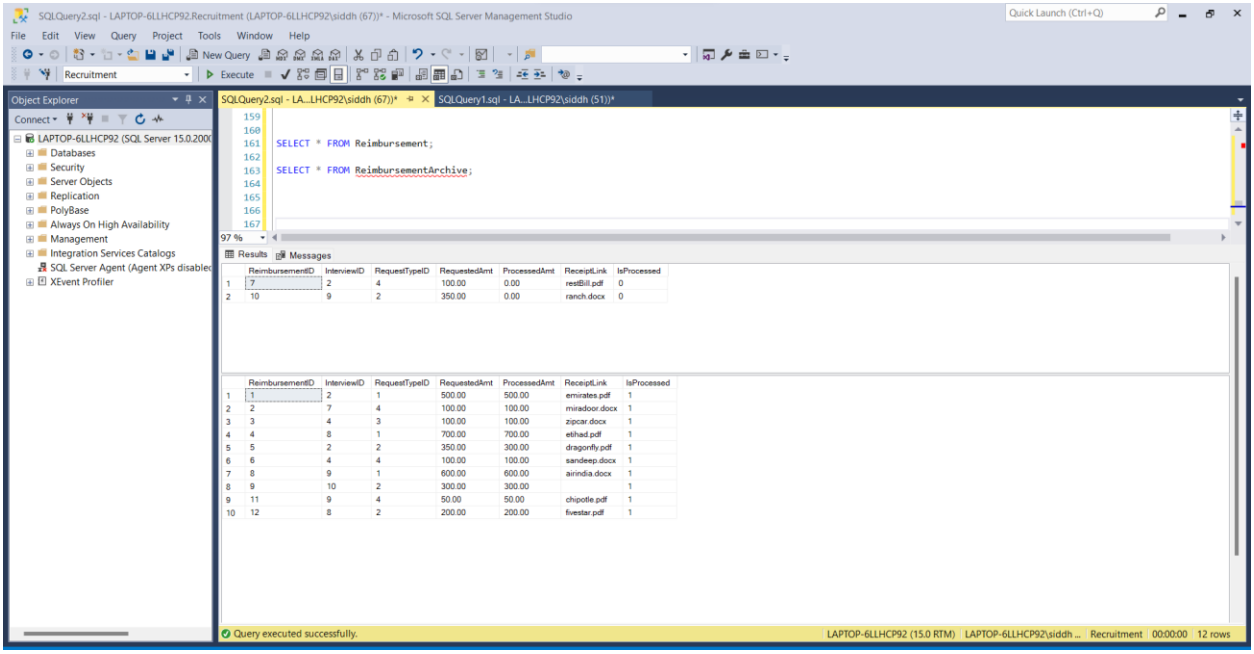
```
BEGIN TRAN

INSERT INTO Onboarding VALUES
(6,3,dateadd(w,-20 ,GETDATE()))

UPDATE Applications SET ApplicationStatus = 'Onboarding' WHERE CandidateID = 6 and
JobOpeningID = (SELECT JobOpeningID FROM JobOpenings WHERE JobID=3)

COMMIT TRAN

SELECT * FROM OnBoarding
SELECT * FROM Applications
```



2. Write a set of action queries coded as a transaction to move rows from the Reimbursement table to the ReimbursementArchive table. Insert all processed reimbursements from Reimbursement into ReimbursementArchive. Then, delete all processed reimbursements from the Reimbursement table, but only if the invoice exists in the ReimbursementArchive table. Use SELECT statement to verify the results.

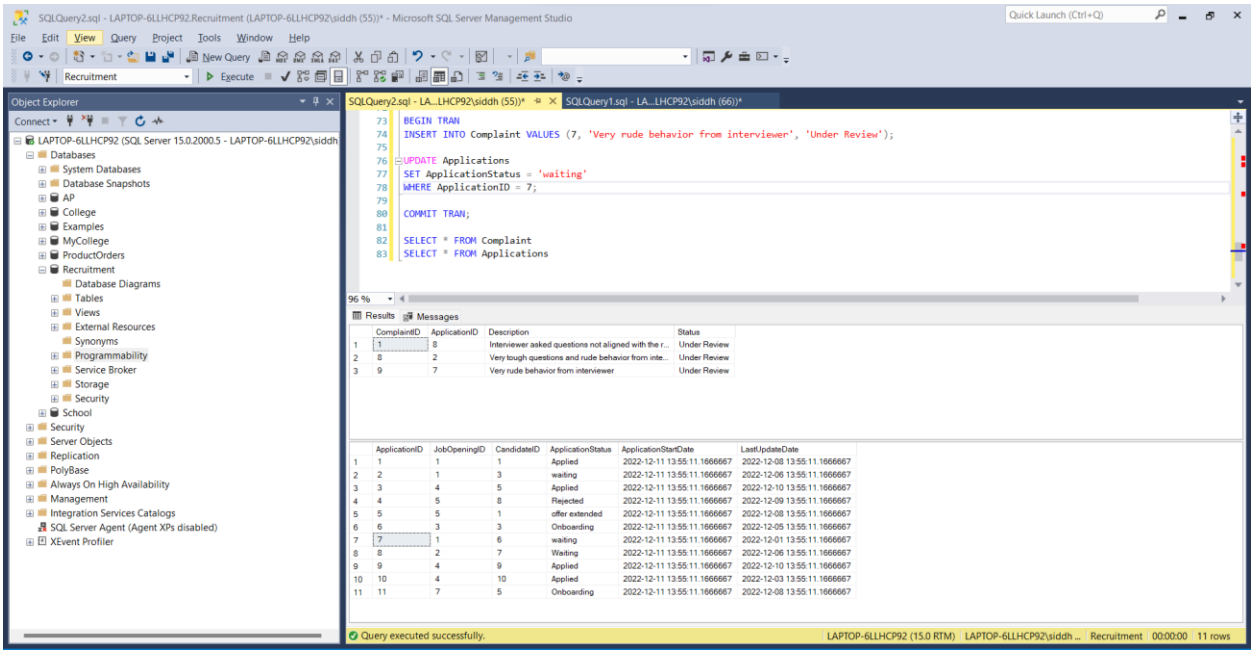
```
CREATE TABLE ReimbursementArchive(
      ReimbursementID int,
      InterviewID int,
      RequestTypeID int,
      RequestedAmt money,
      ProcessedAmt money,
      ReceiptLink varchar(200),
      IsProcessed varchar(200)
      )
GO

BEGIN TRAN

INSERT INTO ReimbursementArchive
SELECT r.*
FROM Reimbursement r
WHERE IsProcessed = 1;

DELETE FROM Reimbursement
```

```
WHERE ReimbursementID IN(SELECT ReimbursementID FROM
ReimbursementArchive);

COMMIT TRAN;

SELECT * FROM Reimbursement;

SELECT * FROM ReimbursementArchive;
```



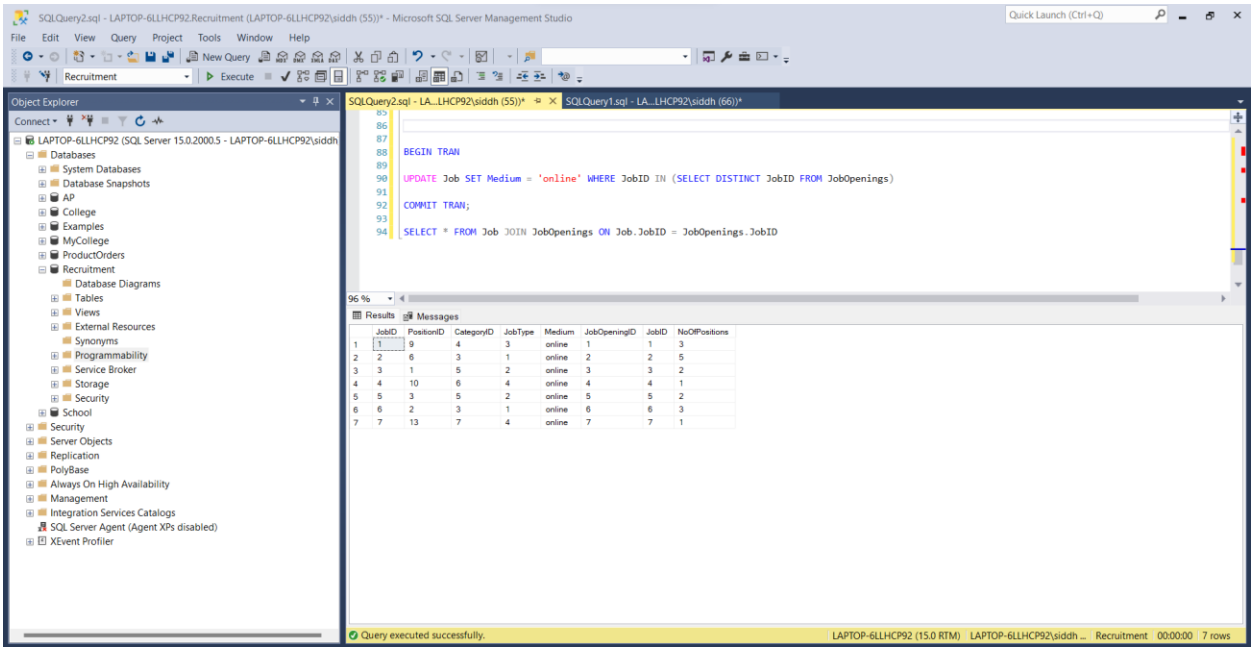
3. Write a set of action queries coded as a transaction to change status of an application to onboarding as we get entry in Onboarding table. Use SELECT statement to verify the results.

```
BEGIN TRAN
INSERT INTO Complaint VALUES (7, 'Very rude behavior from interviewer', 'Under
Review');

UPDATE Applications
SET ApplicationStatus = 'waiting'
WHERE ApplicationID = 7;

COMMIT TRAN;

SELECT * FROM Complaint
SELECT * FROM Applications
```



4. Write a set of action queries coded as a transaction to change medium of an job to online.

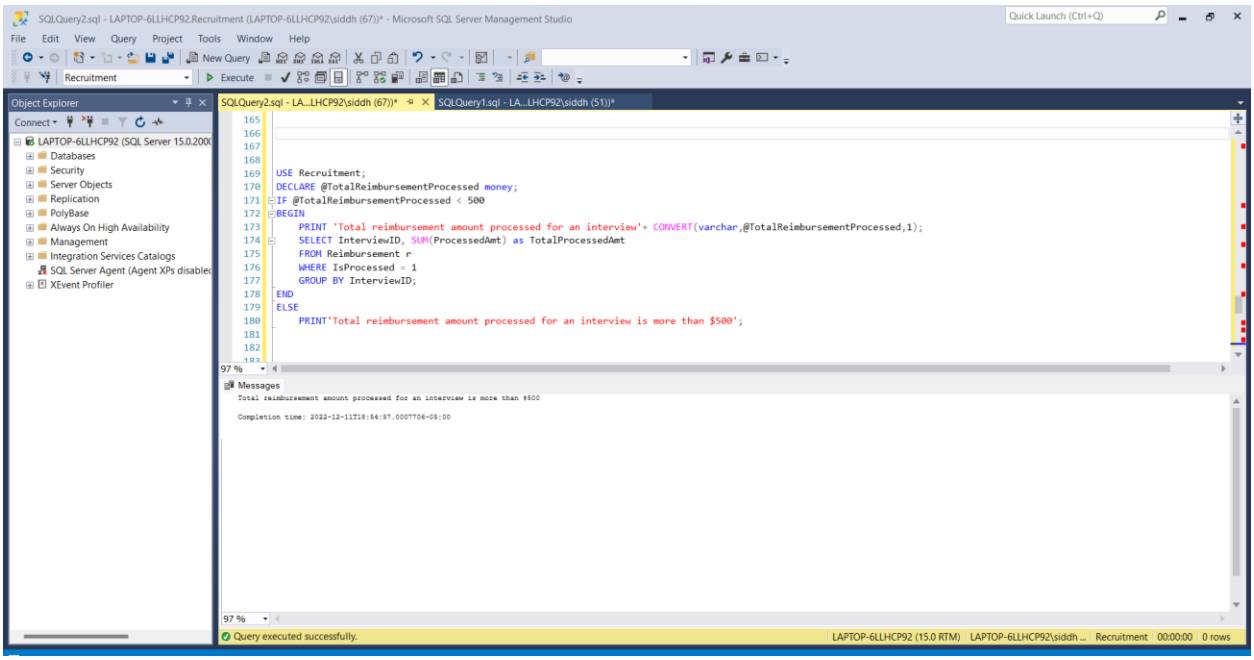
```
BEGIN TRAN

UPDATE Job SET Medium = 'online' WHERE JobID IN (SELECT DISTINCT JobID FROM
JobOpenings)

COMMIT TRAN;

SELECT * FROM Job JOIN JobOpenings ON Job.JobID = JobOpenings.JobID;
```



SCRIPTS

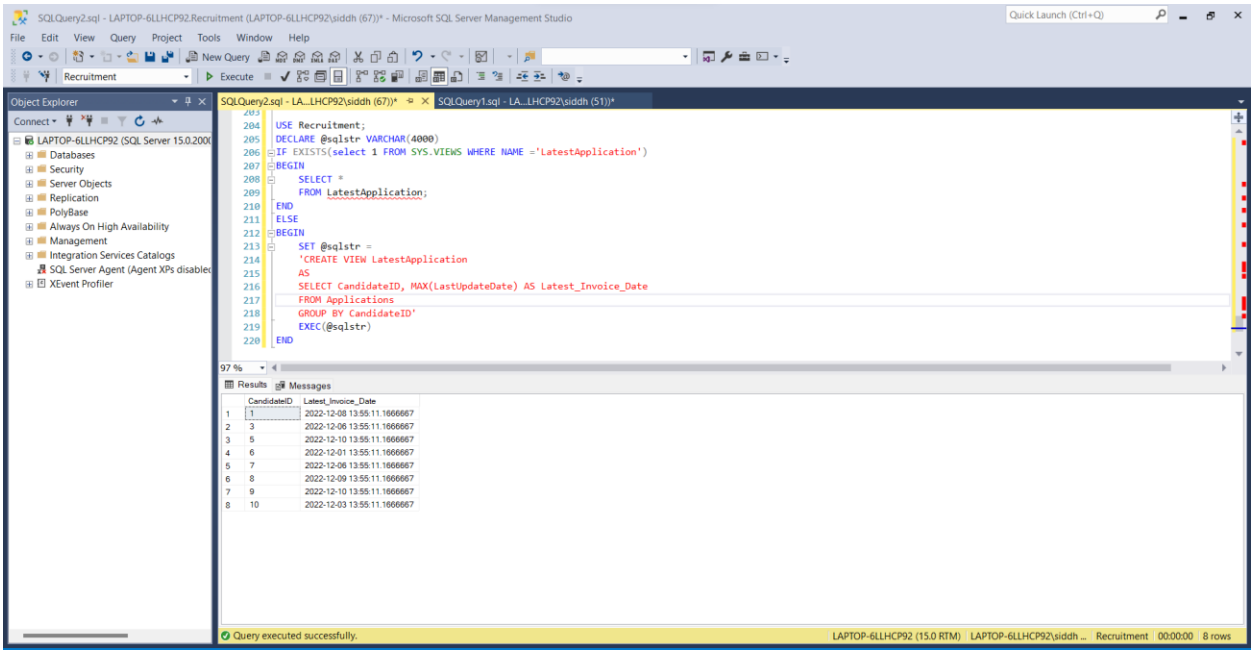
1. Write a script that declares and sets a variable named @TotalReimbursementProcessed, which is equal to the total processed amount. What is the datatype of the variable @ TotalReimbursementProcessed? If that total processed amount is less than $500, the script should return a result set consisting of InterviewID , TotalProcessedAmt for each. Then also return the value of @ TotalReimbursementProcessed in the format of "Total reimbursement amount processed ...". If the total outstanding balance due is more than $500, the script should return the message "'Total reimbursement amount processed for an interview is more than $500".

```
USE Recruitment;
DECLARE @TotalReimbursementProcessed money;
IF @TotalReimbursementProcessed < 500
BEGIN
	PRINT 'Total reimbursement amount processed for an
interview'+ CONVERT(varchar,@TotalReimbursementProcessed,1);
	SELECT InterviewID, SUM(ProcessedAmt) as TotalProcessedAmt
	FROM Reimbursement r
	WHERE IsProcessed = 1
	GROUP BY InterviewID;
END
ELSE
	PRINT 'Total reimbursement amount processed for an interview
is more than $500';
```



2. Write a script that generates the date on which the candidate has given his last ineterview, using a view instead of a derived table. Also write the script that creates the view, then use SELECT statement to show result of the view. Make sure that your script tests for the existence of the view. The view doesn't need to be redefined each time the script is executed.

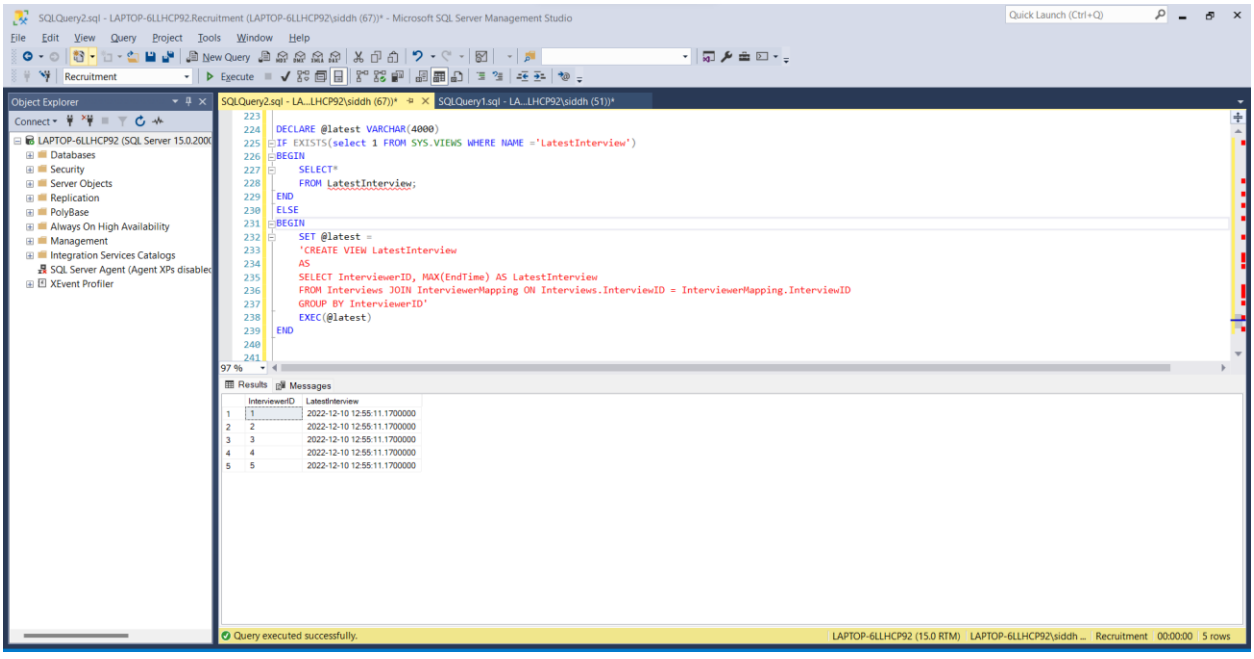
```
USE Recruitment;
DECLARE @sqlstr VARCHAR(4000)
IF EXISTS(select 1 FROM SYS.VIEWS WHERE NAME
='LatestApplication')
BEGIN
	SELECT *
	FROM LatestApplication;
END
ELSE
BEGIN
	SET @sqlstr =
	'CREATE VIEW LatestApplication
	AS
	SELECT CandidateID, MAX(LastUpdateDate) AS
Latest_Invoice_Date
	FROM Applications
	GROUP BY CandidateID'
	EXEC(@sqlstr)
END
```



3. Write a script that generates the date on which the interviewer has taken his last ineterview, using a view instead of a derived table. Also write the script that creates the

view, then use SELECT statement to show result of the view. Make sure that your script tests for the existence of the view. The view doesn't need to be redefined each time the script is executed.

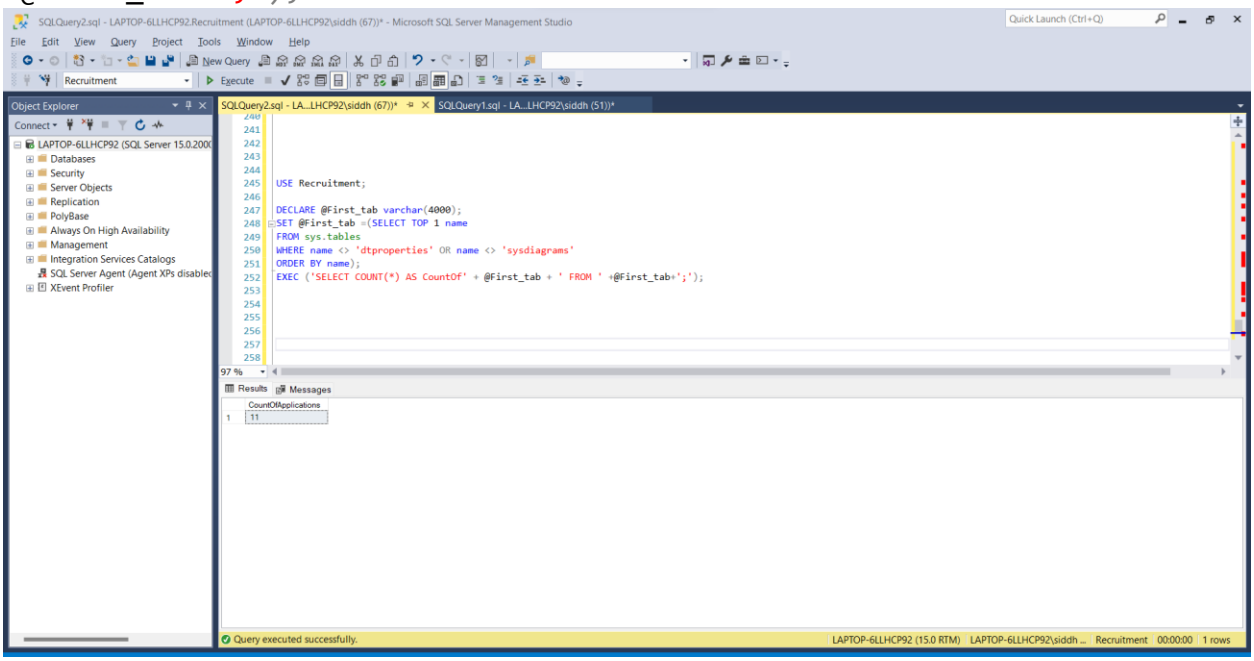
```
DECLARE @latest VARCHAR(4000)
IF EXISTS(select 1 FROM SYS.VIEWS WHERE NAME ='LatestInterview')
BEGIN
	SELECT*
	FROM LatestInterview;
END
ELSE
BEGIN
	SET @latest =
	'CREATE VIEW LatestInterview
	AS
	SELECT InterviewerID, MAX(EndTime) AS LatestInterview
	FROM Interviews JOIN InterviewerMapping ON
Interviews.InterviewID = InterviewerMapping.InterviewID
	GROUP BY InterviewerID'
	EXEC(@latest)
END
```



4. Write a script that uses dynamic SQL to return a single column that represents the number of rows in the first table in the current database. The script should automatically choose the table that appears first alphabetically, and it should exclude

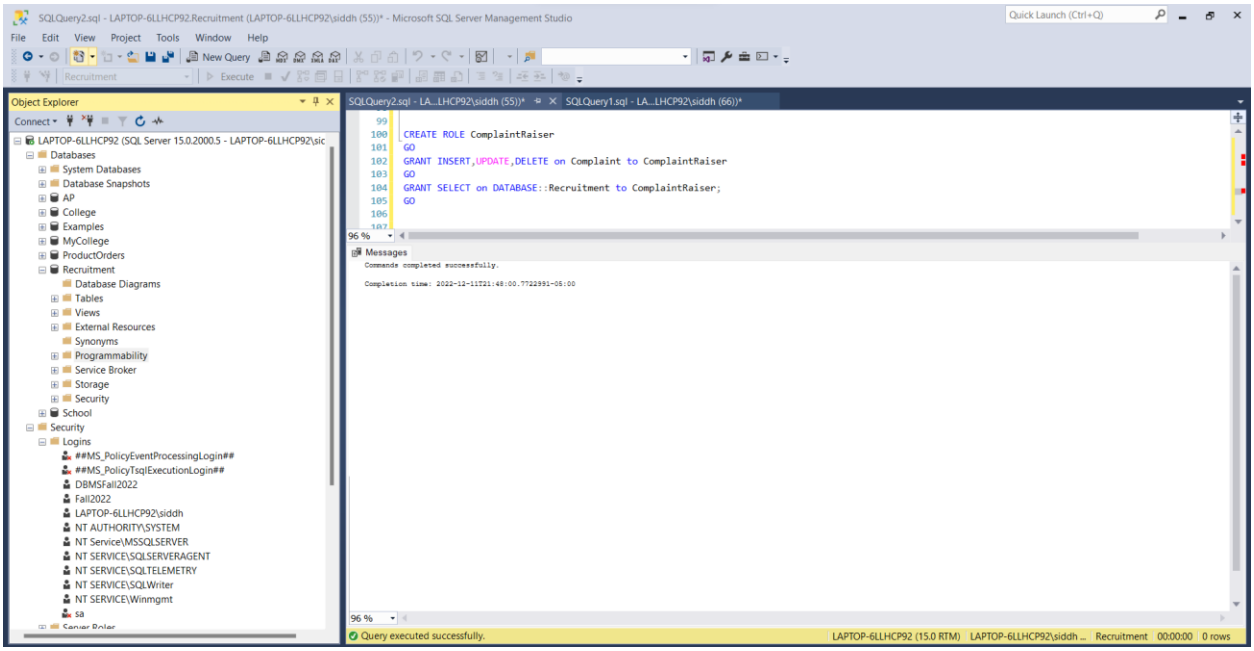
tables named dtproperties and sysdiagrams. Name the column CountOfTable, where Table is the chosen table name. Show results for AP database.

```
USE Recruitment;

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+';');
```

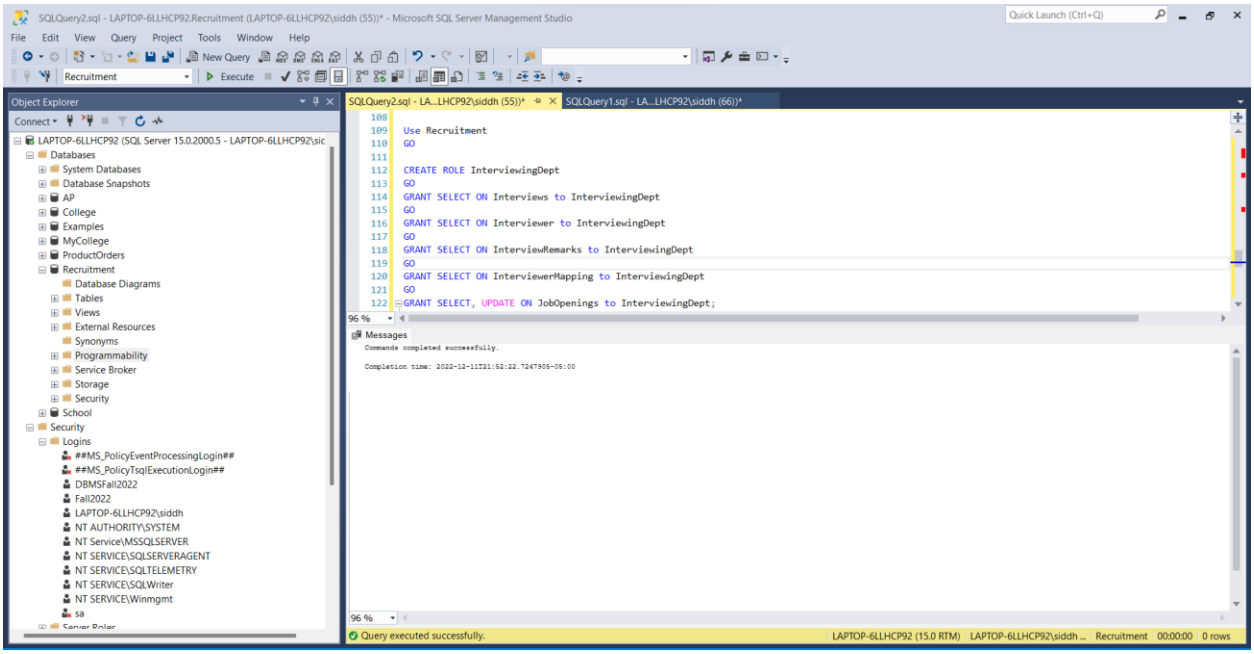


Script Roles

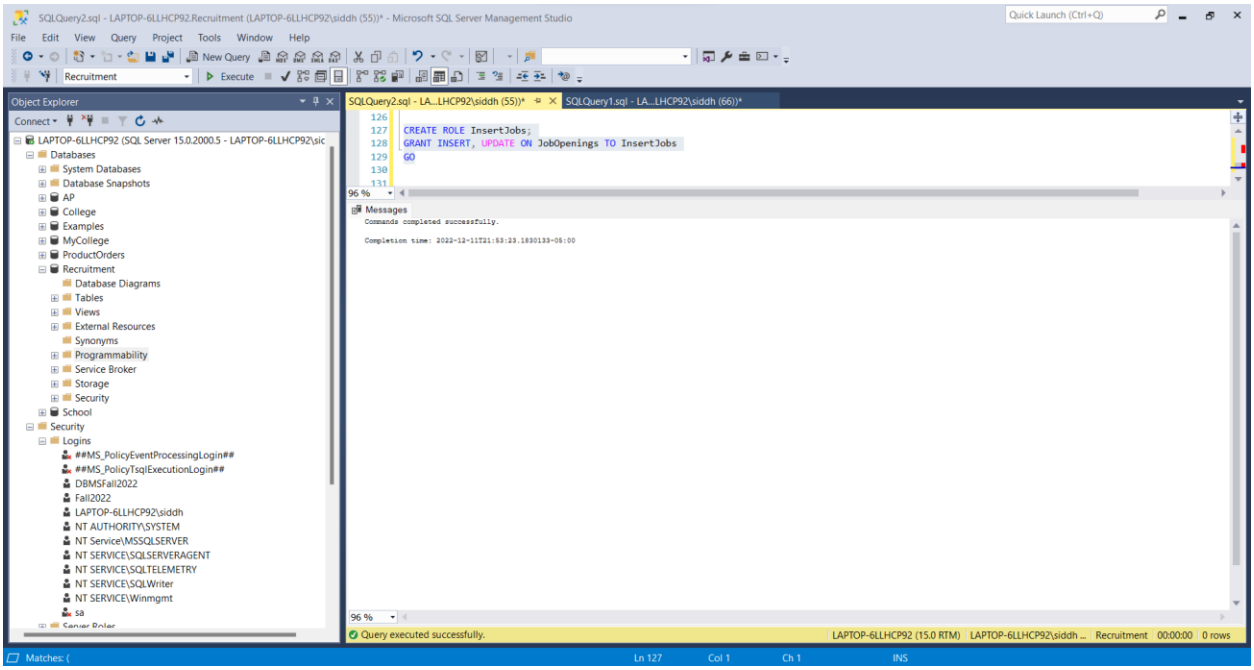
1. ComplaintRaiser role have insert, update, delete access on complaint table. Select access for recruitment database tables.



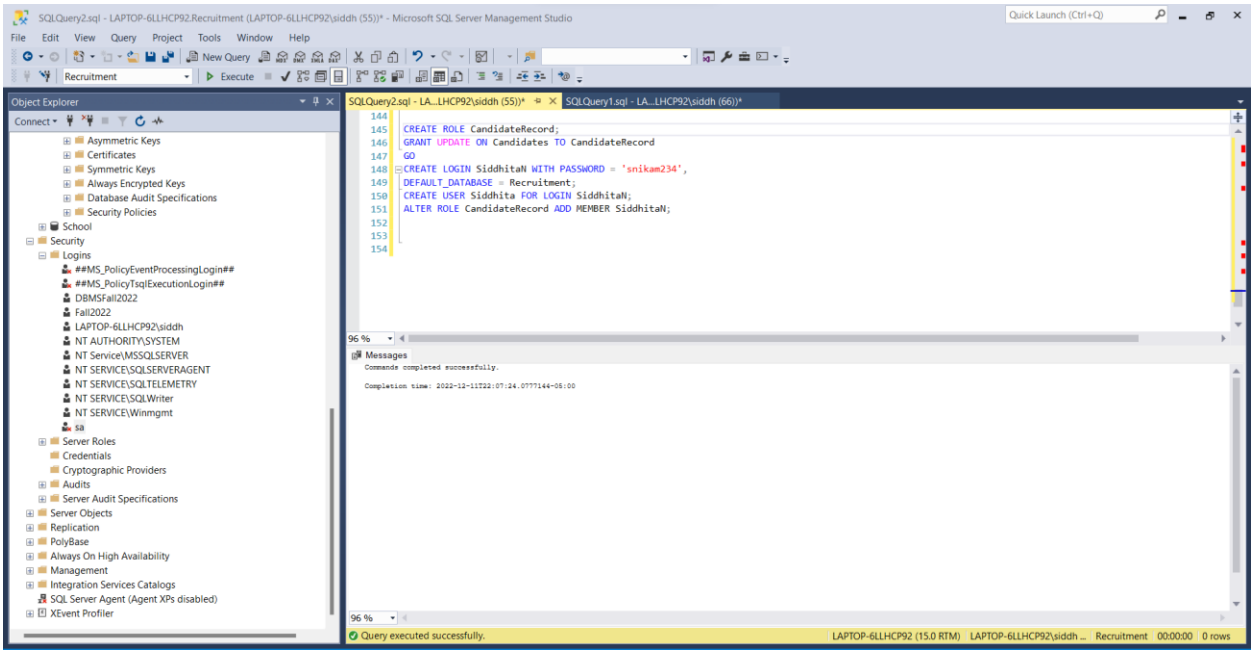
2. Interview Department Role to have access to all the interview related tables.



3. InsertJobs role will have access to add or update job information jobOpenings table.



4. Candidate role to have access to update own information. Created Login and added user for the same. Added this member in the candidate role.

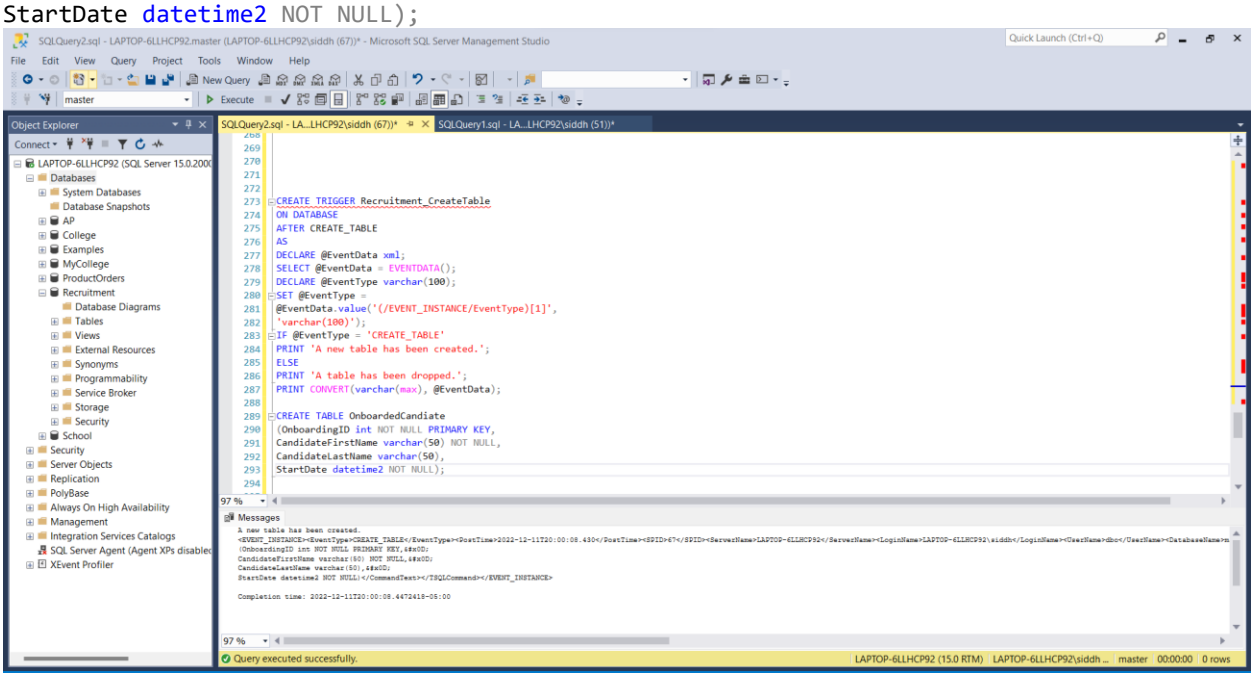


TRIGGERS

1. Write a trigger on database so that whenever a new table is created, the trigger will generate a message indicating the same.

```
CREATE TRIGGER RecruitmentDB_CreateTable
```

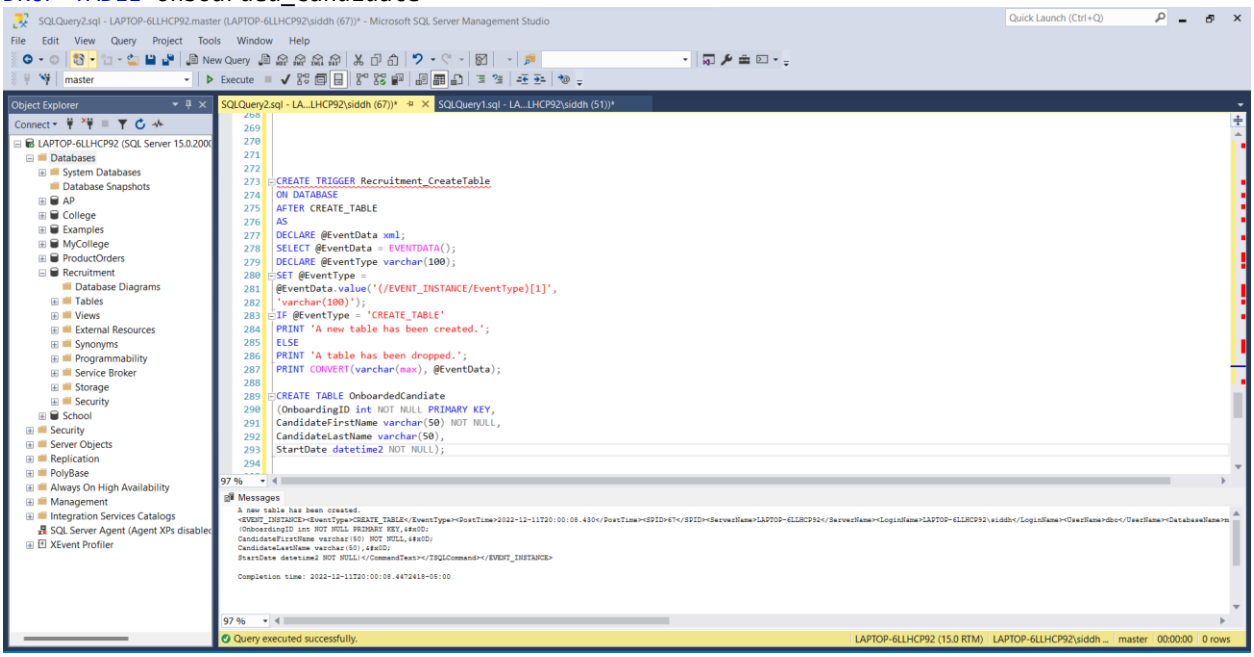
```
ON DATABASE
AFTER CREATE_TABLE
AS
DECLARE @EventData xml;
SELECT @EventData = EVENTDATA();
DECLARE @EventType varchar(100);
SET @EventType =
@EventData.value('(/EVENT_INSTANCE/EventType)[1]',
'varchar(100)');
IF @EventType = 'CREATE_TABLE'
PRINT 'A new table has been created.';

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

USE Recruitment
GO
CREATE TABLE Onboarded_candidate
(OnboardingID int NOT NULL PRIMARY KEY,
CandidateFirstName varchar(50) NOT NULL,
CandidateLastName varchar(50),
StartDate datetime2 NOT NULL);
```



2. Write a trigger on database so that whenever a table is dropped, the trigger will generate a message indicating the same.

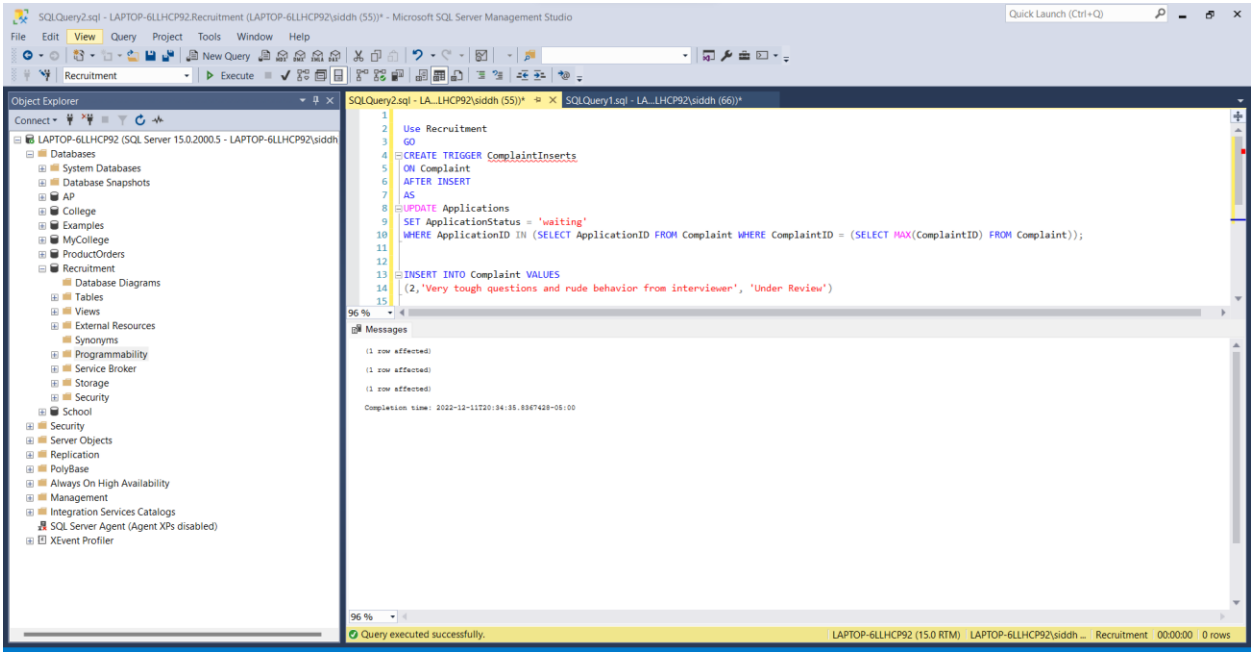
```
CREATE TRIGGER Recruitment_DropTable
ON DATABASE
AFTER DROP_TABLE
AS
DECLARE @EventData xml;
SELECT @EventData = EVENTDATA();
DECLARE @EventType varchar(100);
SET @EventType =
@EventData.value('(/EVENT_INSTANCE/EventType)[1]',
'varchar(100)');
```

```
IF @EventType = 'DROP_TABLE'
PRINT 'A table has been dropped.';
PRINT CONVERT(varchar(max), @EventData);

USE Recruitment
GO
DROP TABLE Onboarded_candidate
```

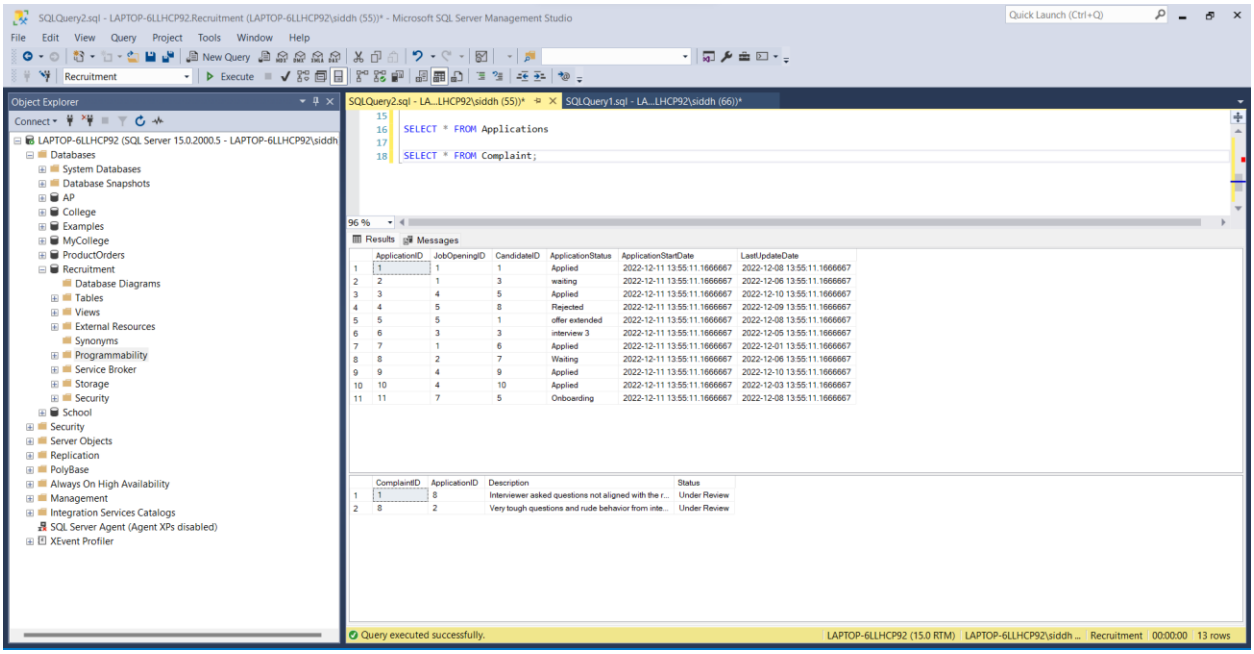


3. Write A Trigger to update the application status of the application for which complaint is raised. On update/insert in complaint table, application status will change in applications table.

```
Use Recruitment
GO
CREATE TRIGGER ComplaintInserts
ON Complaint
AFTER INSERT
AS
UPDATE Applications
SET ApplicationStatus = 'waiting'
WHERE ApplicationID IN (SELECT ApplicationID FROM Complaint WHERE ComplaintID =
(SELECT MAX(ComplaintID) FROM Complaint));


INSERT INTO Complaint VALUES
(2,'Very tough questions and rude behavior from interviewer', 'Under Review')
```



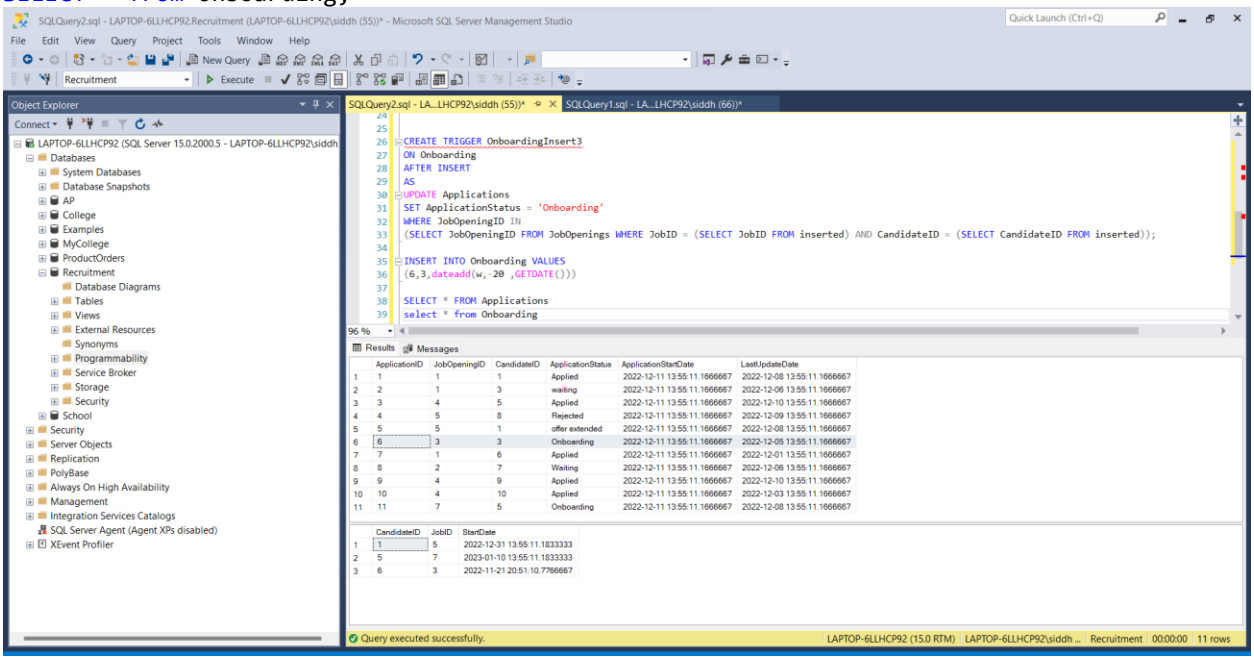
Below rows are affected



4. Write a trigger on Onboarding table so that whenever there is a new entry in onboarding table for a candidate application status for that jobid in the applications table will change to ‘Onboarding’.

```
CREATE TRIGGER OnboardingInsert3
ON Onboarding
AFTER INSERT
AS
```

```
UPDATE Applications
SET ApplicationStatus = 'Onboarding'
WHERE JobOpeningID IN
(SELECT JobOpeningID FROM JobOpenings WHERE JobID = (SELECT JobID FROM inserted)
AND CandidateID = (SELECT CandidateID FROM inserted));

INSERT INTO Onboarding VALUES
(6,3,dateadd(w,-20 ,GETDATE()));

SELECT * FROM Applications;
SELECT * from Onboarding;
```



Conclusion

In this project, we designed, implemented, and tested a database for the Recruitment branch of the HR Department in a company.

The interview process updated a candidate's status from time-to-time.

We have written views, stored procedures, user defined functions, transactions, triggers, scripts as well as security roles for this database.