PROJECT – 2

RECRUITMENT DATABASE

Sanath Krishna Satish

SUID: 226991034

Abstract

A Human Resource Department or an HRD in any company or an organization is one of the most essential, if not crucial, components in an organization regardless of the size of organization. This department is setup for the smooth and efficient running of the organization. HRD is usually charged with finding, screening, recruiting and training job applicants, and administering the employee-benefit programs. As companies reorganize to gain a competitive edge, HR plays a key role in helping companies deal with a fast-changing environment and the greater demand for quality employees. A HR Department helps to maintain a company's core values and culture.

For this project, I have chosen to design a database model for a company. The database is named as XYZRecruitmentDatabase. The implementation of this project includes creating a database for the Human Resources team to store the necessary information about the candidates who have applied to various available in the company. The main goal of this project is to create a real-life environment which depicts the model used to address the problems faced by an HR Department in an organization. To make this project an efficient one, I have used available database concepts like Views, Stored Procedures, Triggers, Functions, Transactions, Security Roles and other User-Defined Functions. Database design of my project is taken care by using Entity Relationship diagrams and by normalizing it to improve efficiency and reducing redundancy. The database tracks the status of the application from Applied till the employee is Onboarded Successfully. Certain issues like a person getting blacklisted for not joining the company, even after he has accepted the offer, have been addressed in this database.

Table of Contents

1. Database Name

The name of the database that I have created is **XYZRecruitmentDatabase**

This total project is mainly divided into three parts:

1. Design Phase
2. Implementation Phase
3. Testing Phase

2. Design Phase

a. Tables and columns used

After going through the design flow and the project requirements, I have created 16 tables. The tables and the columns' details are given below:

1. Candidate
    - CandidateID
    - CandidateName
    - CandidateAge
    - Address
    - Contact
    - InterviewID
    - ComplaintID
2. Applications
    - ApplicationID
    - JobID
    - CandidateID
    - ApplicationStatus
3. JobOpenings
    - JobID
    - JobDescription
    - DepartmentID
4. Departments
    - DepartmentID
    - DepartmentManager
5. Interview
    - InterviewID
    - InterviewType
6. Interviewer
    - InterviewerID
    - InterviewerName
    - InterviewerPhone
7. InterviewLocation

- InterviewLocationID
- StreetName
- City
- State
- ZipCode

8. NextJobOppurtunity
    - CandidateID
    - CandidateDescription
9. CandidateDetails
    - CandidateDetailsID
    - Education
    - WorkExperience
    - PreviousOrganisation
    - SSN
10. ComplaintHandling
    - ComplaintID
    - ComplaintDetails
    - CandidateID
    - InterviewerID
11. CarRental
    - RentalID
    - InterviewID
    - CandidateID
    - CarDetails
12. AirlineReservation
    - ReservationID
    - InterviewID
    - CandidateID
    - InterviewLocationID
    - FlightDetails
13. HotelReservation
    - HotelReservationID
    - InterviewID
    - CandidateID
    - InterviewLocationID
    - HotelDetails
14. Reimbursement
    - ReimbursementID
    - InterviewID
    - CandidateID
    - RentalID
    - ReservationID
    - HotelReservationID

- ReimbursementAmount

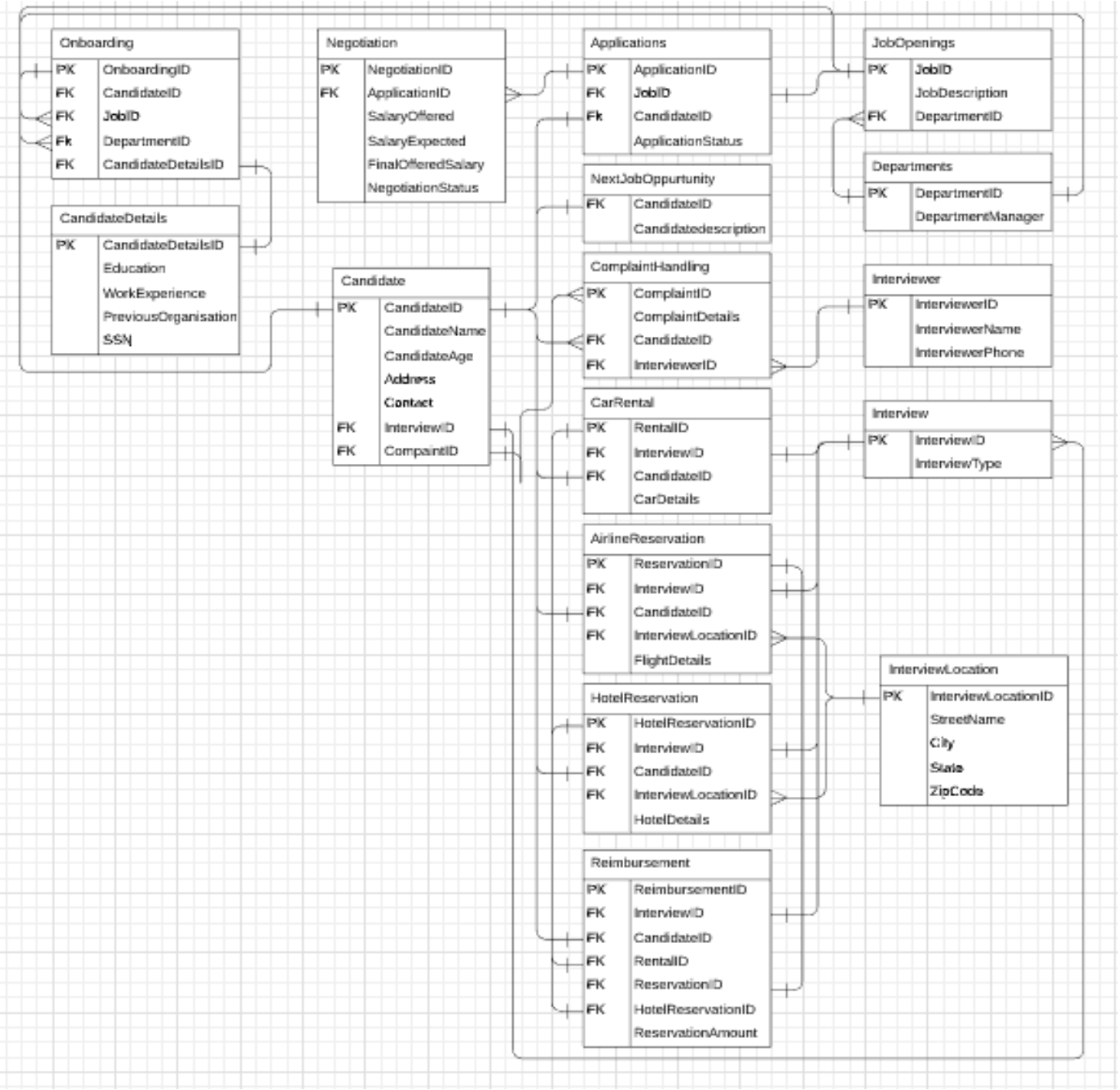
15. Onboarding
    - OnboardingID
    - CandidateID
    - JobID
    - DepartmentID
    - CandidateDetailsID
16. Negotiation
    - NegotiationID
    - ApplicationID
    - SalaryOffered
    - SalaryExpected
    - FinalOfferedSalary
    - NegotiationStatus

These are the tables and columns that I have used for the database.

b. E/R Diagram

The Entity Relationship(E/R) diagram mainly focusses on how entities in the database are related to each other and how to link them.

Here is the E/R diagram given below that is used for my project.



c. Potential Integrity and Security Threats

The potential integrity and security threats can be reduced by transforming the above database into the format mentioned below. This format is obtained by normalizing the tables.

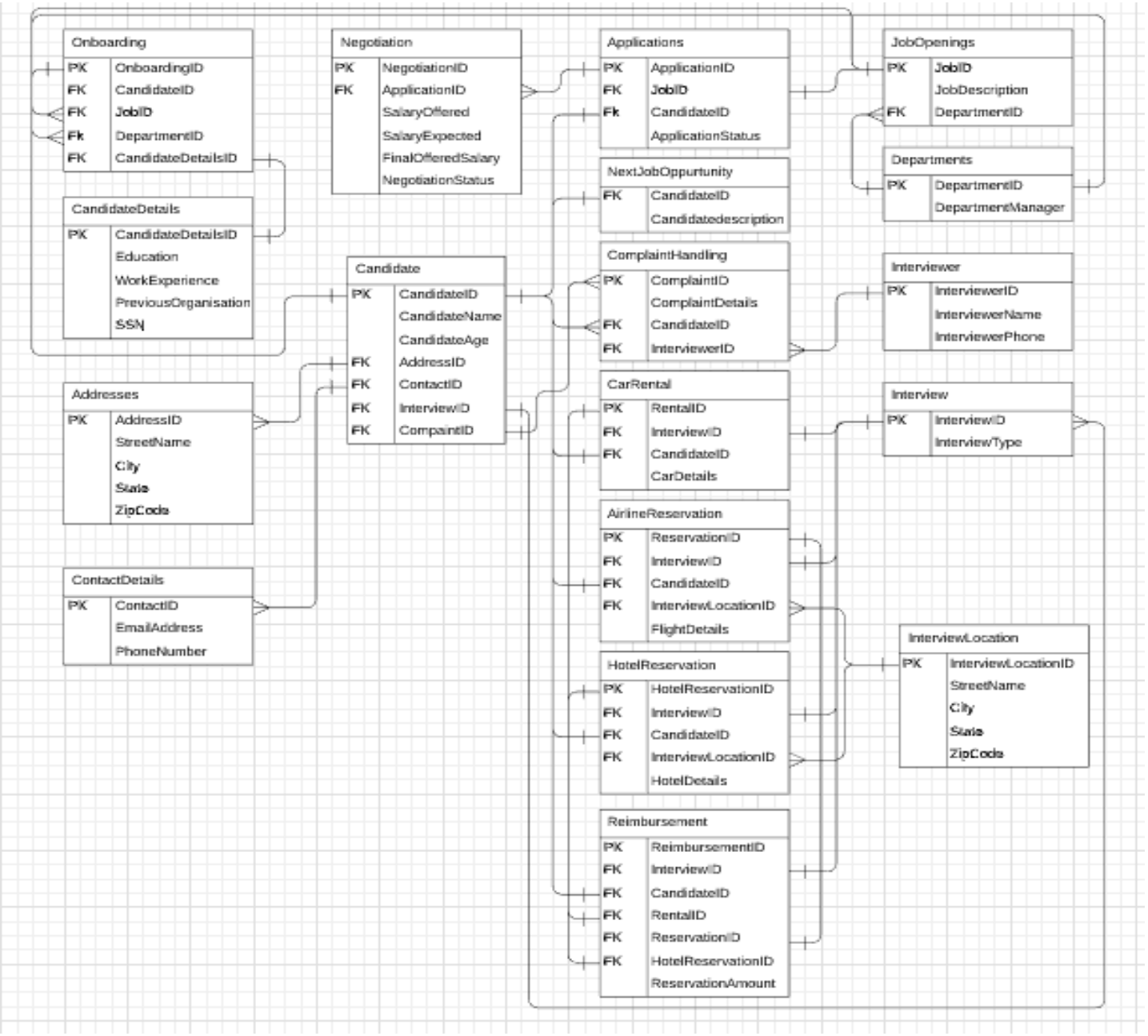
Referential Integrity means that the relationships between tables are maintained correctly. This is used to address the integrity issues in a database design. For referential integrity to hold in a database, any column in a base table that is declared a foreign key can contain either null values or values from a parent table's primary

key or a candidate key. This means that all of the foreign keys in a foreign key table must have matching primary keys in the related table. If referential integrity is not enforced, deleting a record that contains a value referred to by a foreign key in another table would break the referential integrity.

After enforcing the referential integrity, I have modified the above database design to address integrity and security threats. This also reduces redundancy.

d. E/R Diagram in 3NF

The below diagram is in 3rd normal form



In this design, I have created two new tables that are a reference to the Candidate table. Hence in my final design, I have a total of 18 tables.

The details of the new tables are given below:

1. Addresses
    - AddressId
    - StreetName
    - City
    - State
    - ZipCode
2. ContactDetails
    - ContactID
    - EmailAddress
    - PhoneNumber

e. Relationships

The relationships between all the tables is given as follows:

- There is a one-to-one relationship between Candidate and Applications
- There is a one-to-one relationship between Candidate and NextJobOppurtunity
- There is a one-to-many relationship between Candidate and ComplaintHandling
- There is a one-to-one relationship between Candidate and CarRental
- There is a one-to-one relationship between Candidate and AirlineReservation
- There is a one-to-one relationship between Candidate and HotelReservation
- There is a one-to-one relationship between Candidate and Reimbursement
- There is a one-to-one relationship between Candidate and Onboarding
- There is a one-to-many relationship between Candidate and Addresses
- There is a one-to-many relationship between Candidate and ContactDetails
- There is a one-to-many relationship between Candidate and Interview
- There is a one-to-one relationship between CandidateDetails and Onboarding
- There is a one-to-many relationship between Applications and Negotiation
- There is a many-to-one relationship between ComplaintHandling and Candidate
- There is a one-to-one relationship between Reimbursement and CarRental
- There is a one-to-one relationship between Reimbursement and AirlineReservation
- There is a one-to-one relationship between Reimbursement and HotelReservation
- There is a one-to-many relationship between InterviewLocation and AirlineReservation
- There is a one-to-many relationship between InterviewLocation and HotelReservation
- There is a one-to-one relationship between JobOpenings and Applications
- There is a one-to-one relationship between JobOpenings and Onboarding
- There is a one-to-many relationship between Departments and JobOpenings
- There is a one-to-many relationship between Interviewer and ComplaintHandling
- There is a one-to-one relationship between Interview and CarRental
- There is a one-to-one relationship between Interview and AirlineReservation
- There is a one-to-one relationship between Interview and HotelReservation
- There is a one-to-one relationship between Interview and Reimbursement

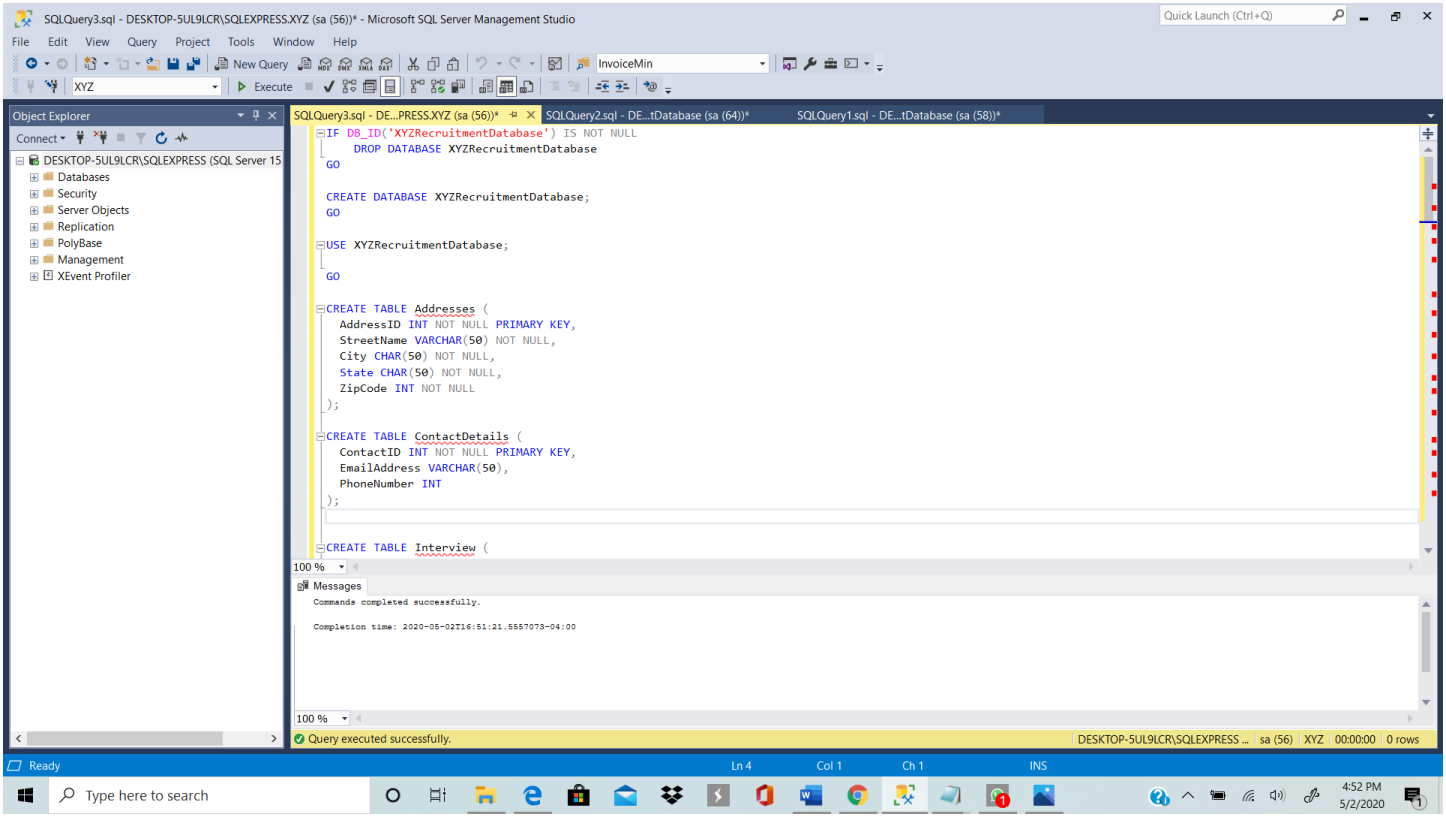
3. Implementation Phase

In the implementation phase, I have created the database and tables using SQL queries in the Microsoft SQL Server Management Studio. I have mentioned the constraints for each column in the database and I have created triggers as per my business logic.

a. Database and Tables Creation

Here initial step to create all the necessary tables that are required. Here I have created all the tables with the all constraints and relations.

Here is the screenshot of the tables and columns that I have created in the database.



b. Constraints

Here are the primary keys, foreign keys, data types, nullable and relationships that I have used in this project:

1. For Candidate table
    - CandidateID INT NOT NULL PRIMARY KEY
    - CandidateName CHAR(50) NOT NULL
    - CandidateAge INT NOT NULL
    - AddressID INT NOT NULL FOREIGN KEY
    - ContactID INT NOT NULL FOREIGN KEY
    - InterviewType INT NOT NULL FOREIGN KEY
    - ComplaintID INT NOT NULL FOREIGN KEY

2. For Applications table
    - ApplicationID INT NOT NULL PRIMARY KEY
    - JobID INT NOT NULL FOREIGN KEY
    - CandidateID INT NOT NULL FOREIGN KEY
    - ApplicationStatus VARCHAR(20) NULL
    - CandidateName CHAR(50) NULL

3. For JobOpenings table
    - JobID INT NOT NULL PRIMARY KEY
    - JobDescription VARCHAR(50) NOT NULL
    - DepartmentID INT NOT NULL FOREIGN KEY

4. For Departments table
    - DepartmentID INT NOT NULL PRIMARY KEY
    - DepartmentManager CHAR(30) NOT NULL

5. For Interview table
    - InterviewID INT NOT NULL PRIMARY KEY
    - InterviewType VARCHAR(20) NOT NULL

6. For Interviewer table
    - InterviewerID INT NOT NULL PRIMARY KEY
    - InterviewerName CHAR(50) NOT NULL
    - InterviewerPhone INT

7. For InterviewLocation table
    - InterviewLocationID INT NOT NULL PRIMARY KEY
    - StreetName VARCHAR(50)
    - City CHAR(50)
    - State CHAR(50)
    - ZipCode INT
8. For NextJobOppurtunity table
    - CandidateID INT NOT NULL FOREIGN KEY
    - CandidateDescription VARCHAR(100) NOT NULL

9. For CandidateDetails table
    - CandidateDetailsID INT NOT NULL PRIMARY KEY
    - Education VARCHAR(50) NOT NULL
    - WorkExperience VARCHAR(50) NOT NULL
    - PreviousOrganisation VARCHAR(50) NOT NULL
    - SSN INT

10. For ComplaintHandling table
    - ComplaintID INT NOT NULL PRIMARY KEY

- ComplaintDetails VARCHAR(100)
- CandidateID INT NOT NULL
- InterviewerID INT NOT NULL

11. For CarRental table
    - RentalID INT NOT NULL PRIMARY KEY
    - InterviewID INT NOT NULL FOREIGN KEY
    - CandidateID INT NOT NULL FOREIGN KEY
    - CarDetails VARCHAR(50) NOT NULL

12. For AirlineReservation table
    - ReservationID INT NOT NULL PRIMARY KEY,
    - InterviewID INT NOT NULL FOREIGN KEY
    - CandidateID INT NOT NULL FOREIGN KEY
    - InterviewLocationID INT NOT NULL FOREIGN KEY
    - FlightDetails VARCHAR(100) NOT NULL

13. For HotelReservation table
    - HotelReservationID INT NOT NULL PRIMARY KEY
    - InterviewID INT NOT NULL FOREIGN KEY
    - CandidateID INT NOT NULL FOREIGN KEY
    - InterviewLocationID INT NOT NULL FOREIGN KEY
    - HotelDetails VARCHAR(100) NOT NULL

14. For Reimbursement table
    - ReimbursementID INT NOT NULL PRIMARY KEY
    - InterviewID INT NOT NULL FOREIGN KEY
    - CandidateID INT NOT NULL FOREIGN KEY
    - RentalID INT NOT NULL FOREIGN KEY
    - ReservationID INT NOT NULL FOREIGN KEY
    - HotelReservationID INT NOT NULL FOREIGN KEY
    - ReimbursementAmount MONEY NOT NULL

15. For Onboarding table
    - OnboardingID INT NOT NULL PRIMARY KEY
    - CandidateID INT NOT NULL FOREIGN KEY
    - JobID INT NOT NULL FOREIGN KEY
    - DepartmentID INT NOT NULL FOREIGN KEY
    - CandidateDetailsID INT NOT NULL FOREIGN KEY

16. For Negotiation table
    - NegotiationID INT NOT NULL PRIMARY KEY
    - ApplicationID INT NOT NULL FOREIGN KEY

- SalaryOffered MONEY NOT NULL
- SalaryExpected MONEY NOT NULL
- FinalOfferedSalary MONEY NULL
- NegotiationStatus VARCHAR(30) NOT NULL

17. For Addresses table
    - AddressID INT NOT NULL PRIMARY KEY
    - StreetName VARCHAR(50) NOT NULL
    - City CHAR(50) NOT NULL
    - State CHAR(50) NOT NULL
    - ZipCode INT NOT NULL

18. For ContactDetails table
    - ContactID INT NOT NULL PRIMARY KEY
    - EmailAddress VARCHAR(50)
    - PhoneNumber INT

c. Triggers

SQL Server triggers are special stored procedures that are executed automatically in response to the database object, database, and server events.

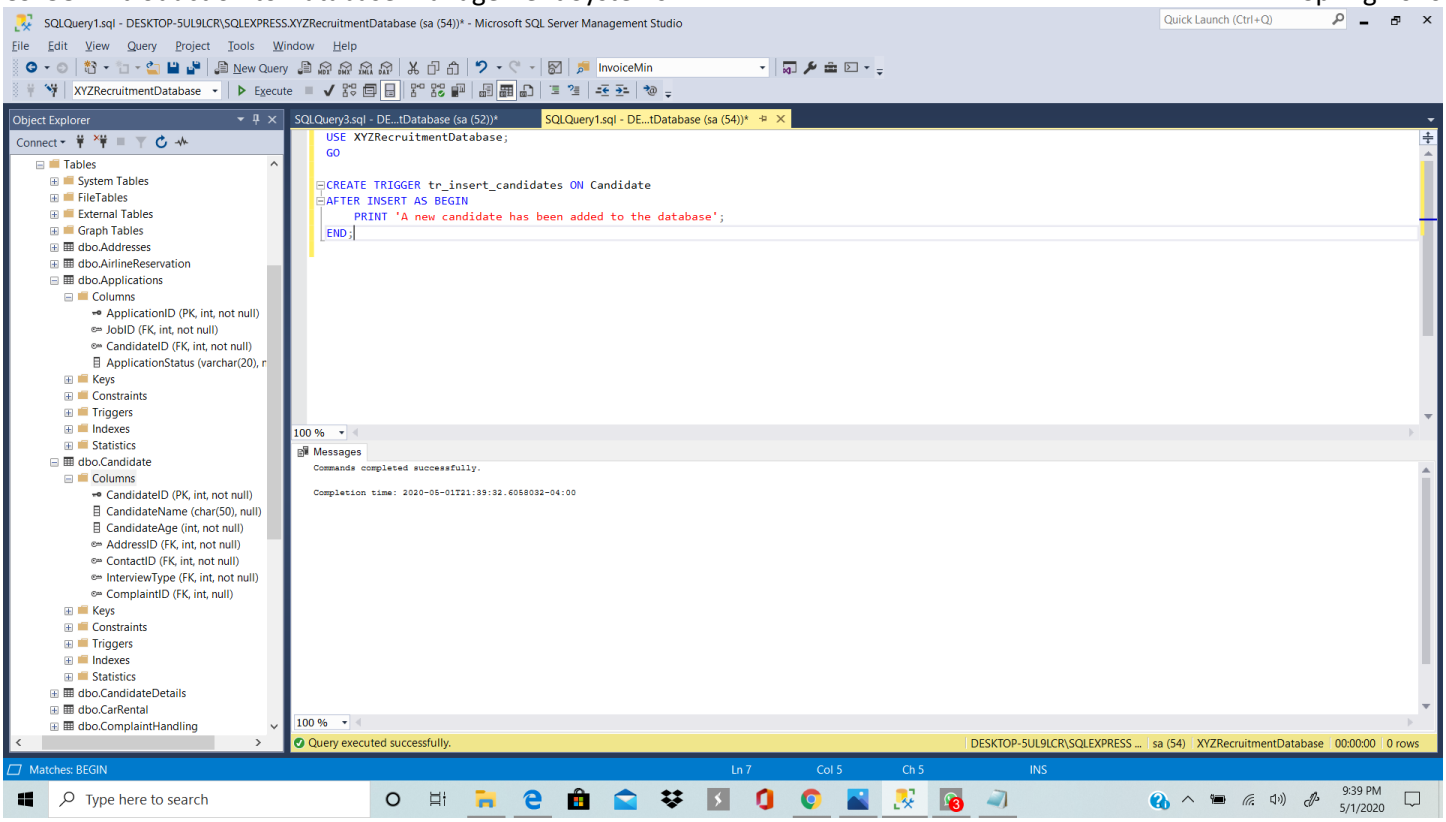
In this project, I have created two triggers. They are as follows:

Trigger-1:

This trigger is created on the Candidate table to print a message whenever an insert operation is performed on the Candidate table.

```
CREATE TRIGGER tr_insert_candidates ON Candidate
AFTER INSERT
AS
BEGIN
        PRINT 'A new candidate gas been added to the database';
END;
```

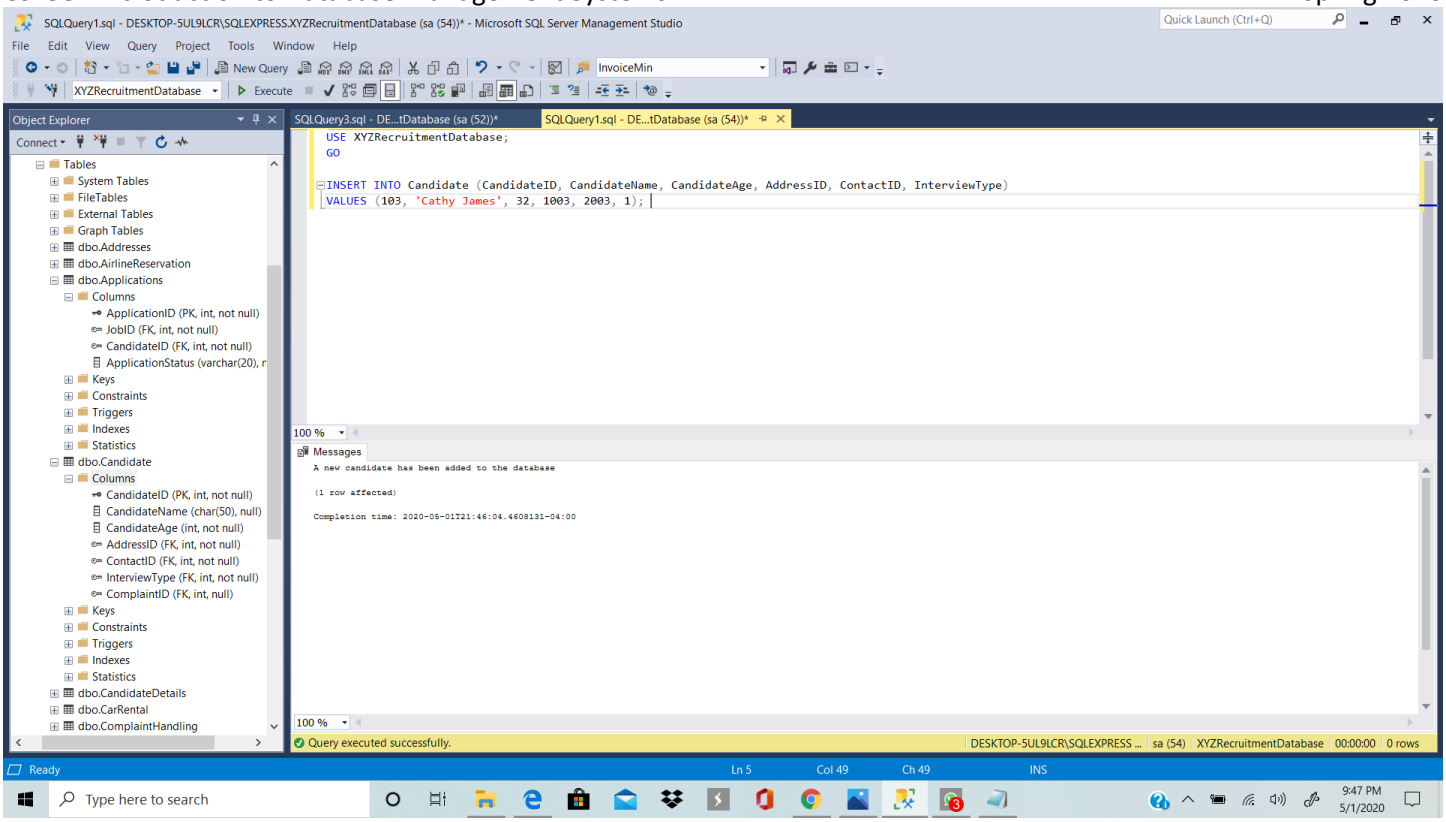
The following script creates a trigger named 'tr_insert_candidates' on Candidate table.



Here an insert is performed to add a row into the Candidate table.

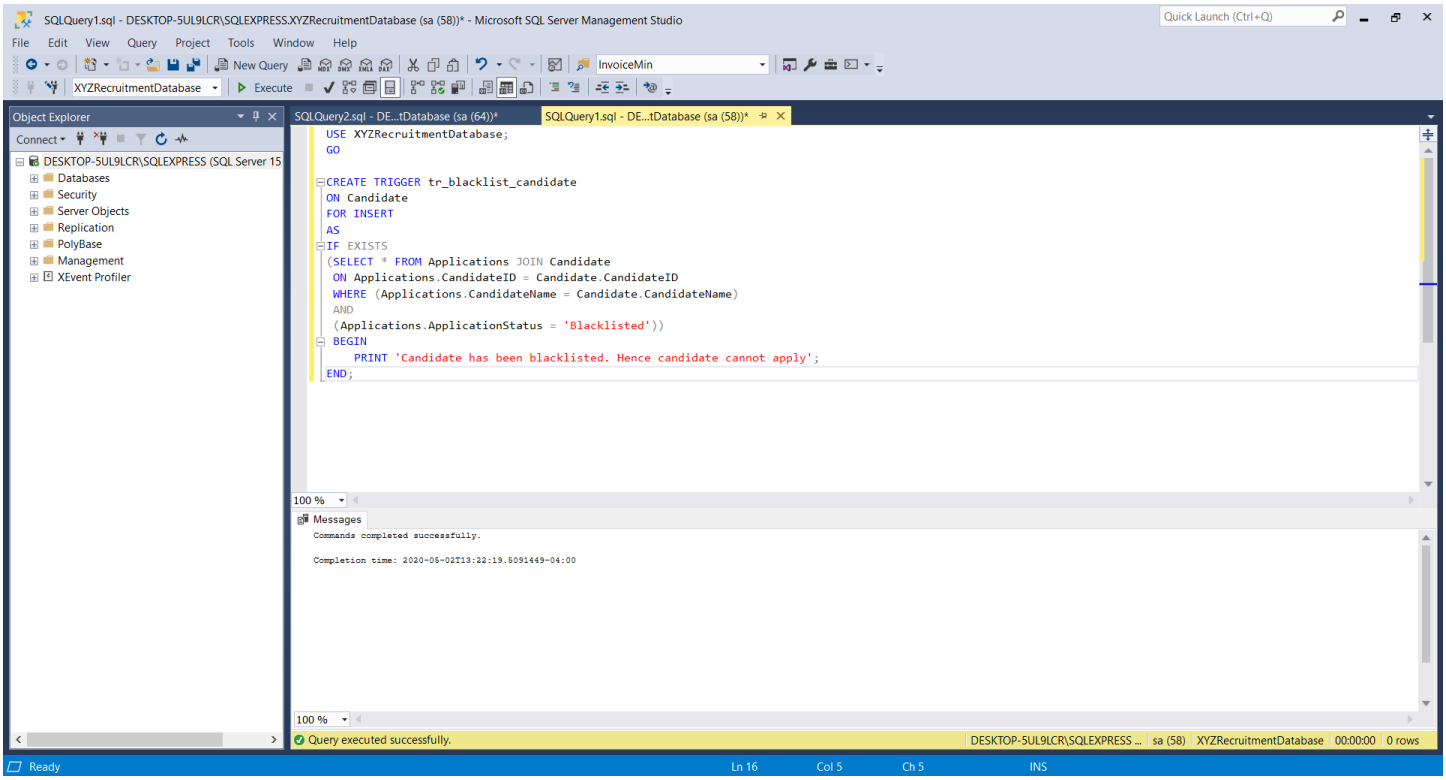
INSERT INTO Candidate (CandidateID, CandidateName, CandidateAge, AddressID, ContactID, InterviewType)
VALUES
(103, 'Cathy James', 32, 1003, 2003, 1);

In the screenshot below, we can see that the message is printed.



Trigger-2:

The following script creates a trigger named ‘tr_blacklist_candidate’ on the Candidate table. The script checks on insertion of an entry into the Candidate table. If the candidate had applied before, and if their previous ApplicationStatus was Blacklisted in the Applications table, then this trigger prints a message.

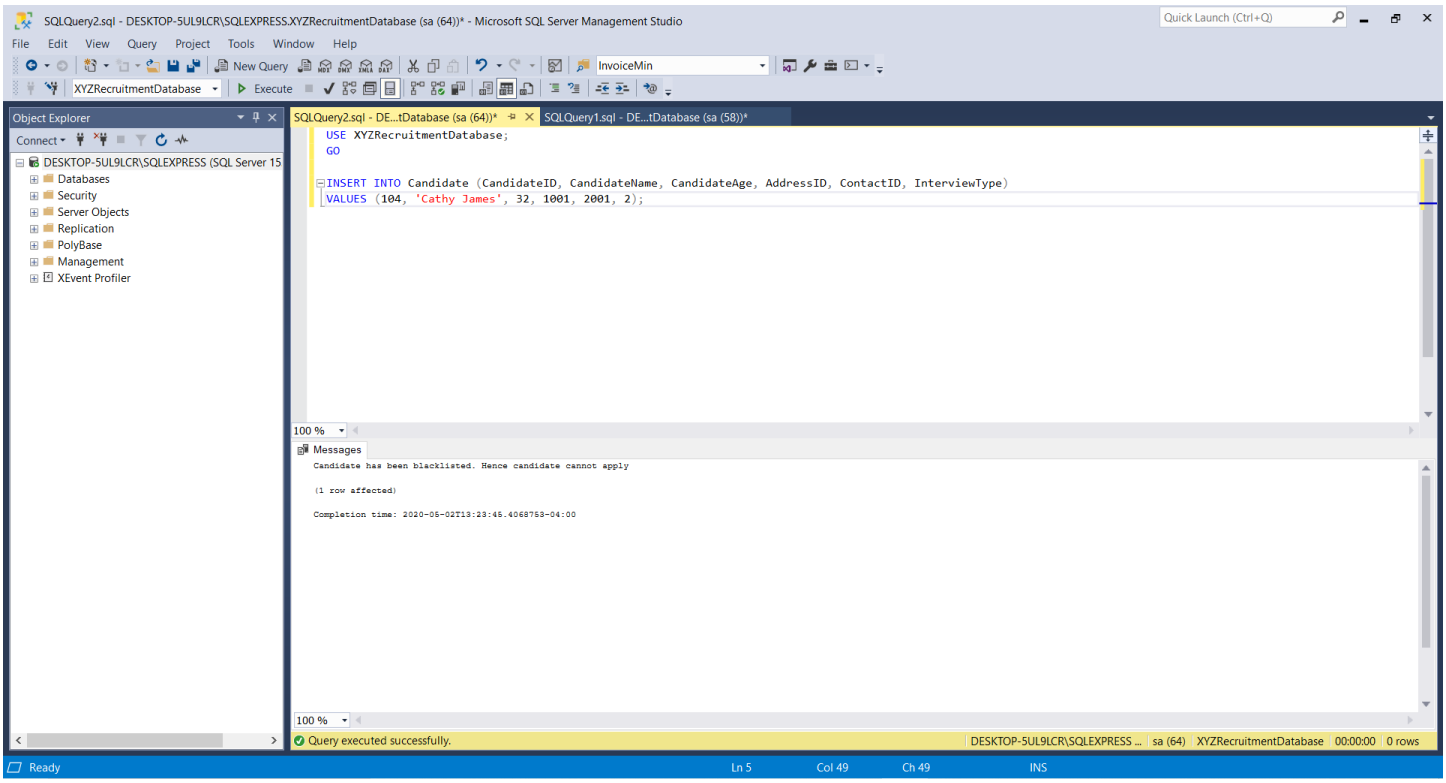


```
CREATE TRIGGER tr_blacklist_candidate
ON Candidate
FOR INSERT
AS

IF EXISTS
(SELECT * FROM Applications JOIN Candidate
ON Applications.CandidateID = Candidate.CandidateID
WHERE (Applications.CandidateName = Candidate.CandidateName)
AND
(Applications.ApplicationStatus = 'Blacklisted'))
BEGIN
        PRINT 'Candidate has been blacklisted. Hence candidate cannot apply';
END;
```

Here an insert is performed to add a row into the Candidate table.

```
INSERT INTO Candidate (CandidateID, CandidateName, CandidateAge, AddressID, ContactID, InterviewType)
VALUES
(104, 'Cathy James', 32, 1001, 2001, 2);
```

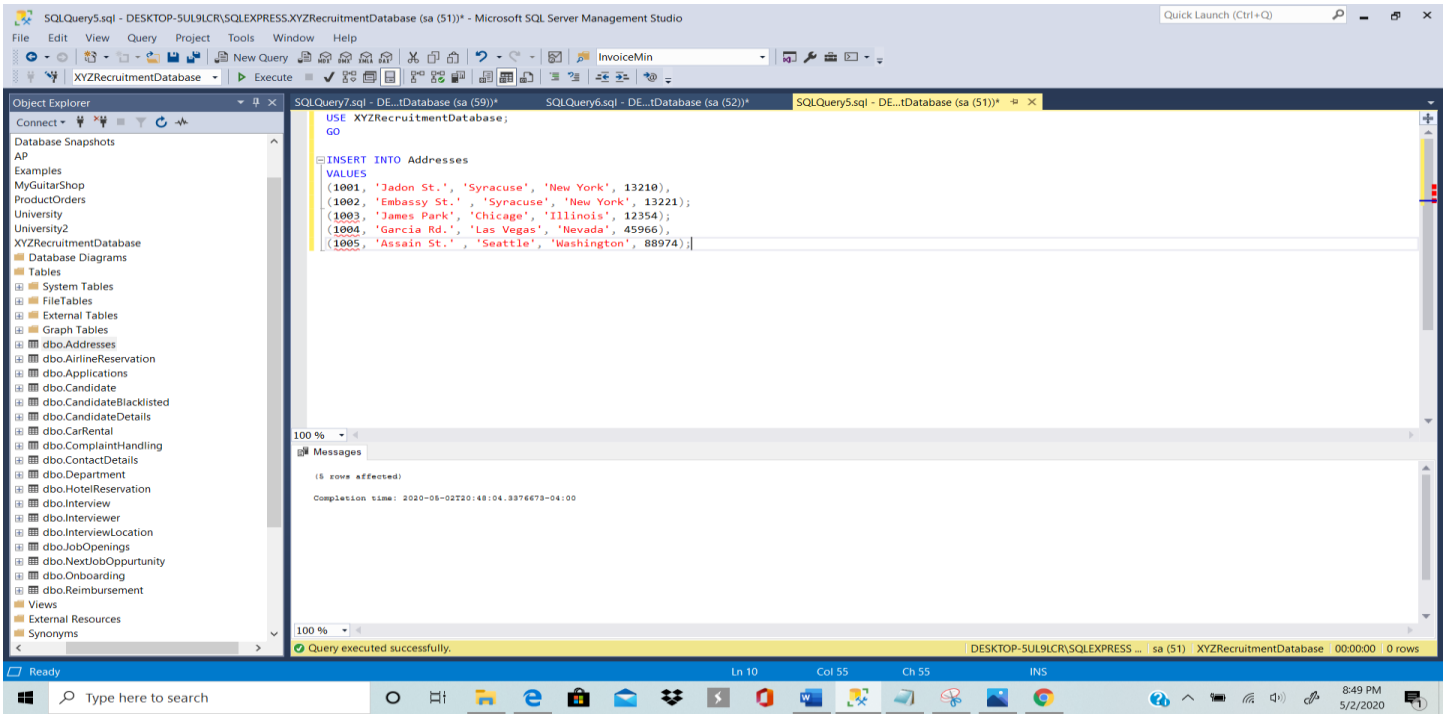
Since Cathy James is blacklisted, we can see that the trigger prints the following message.

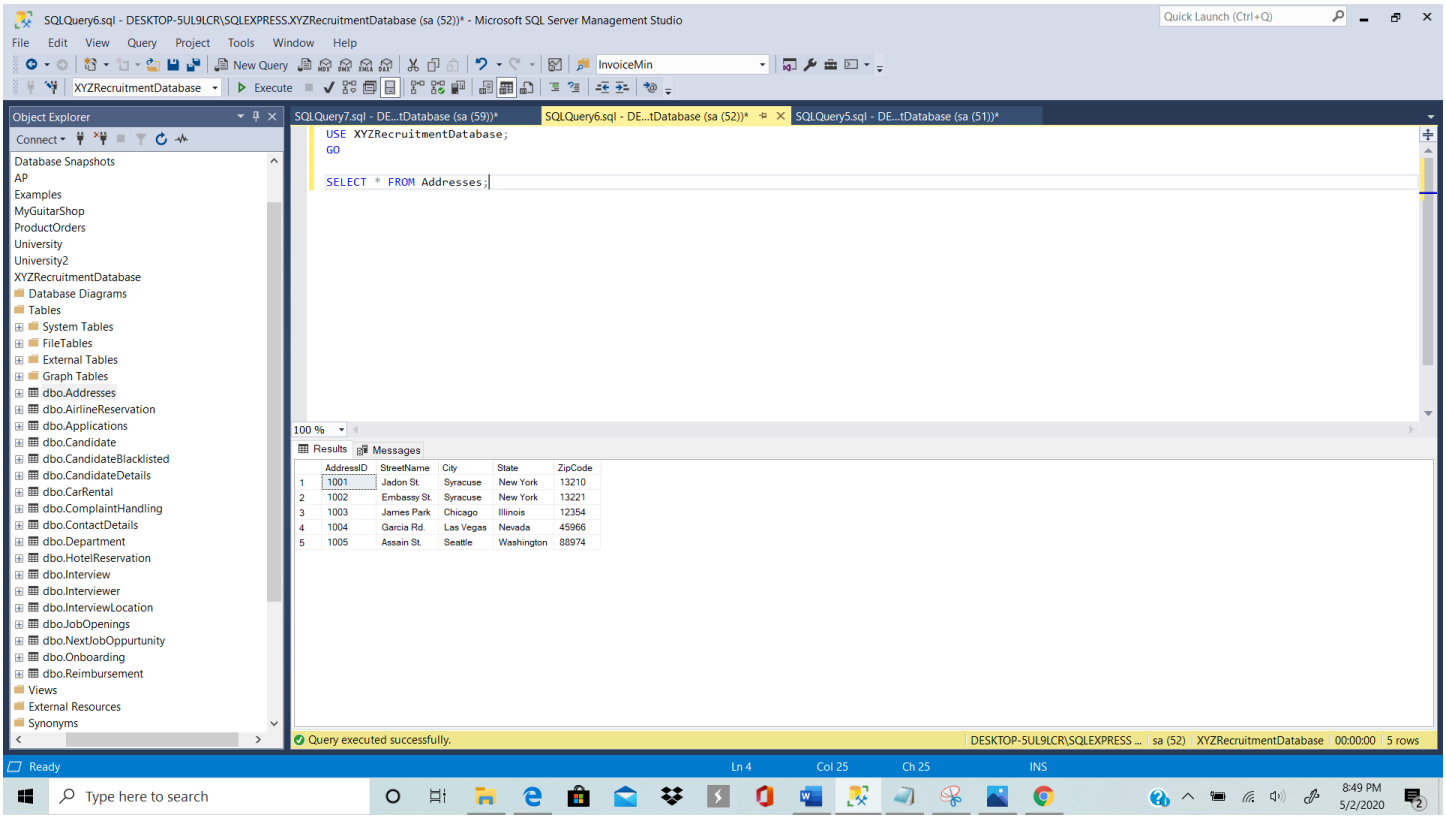


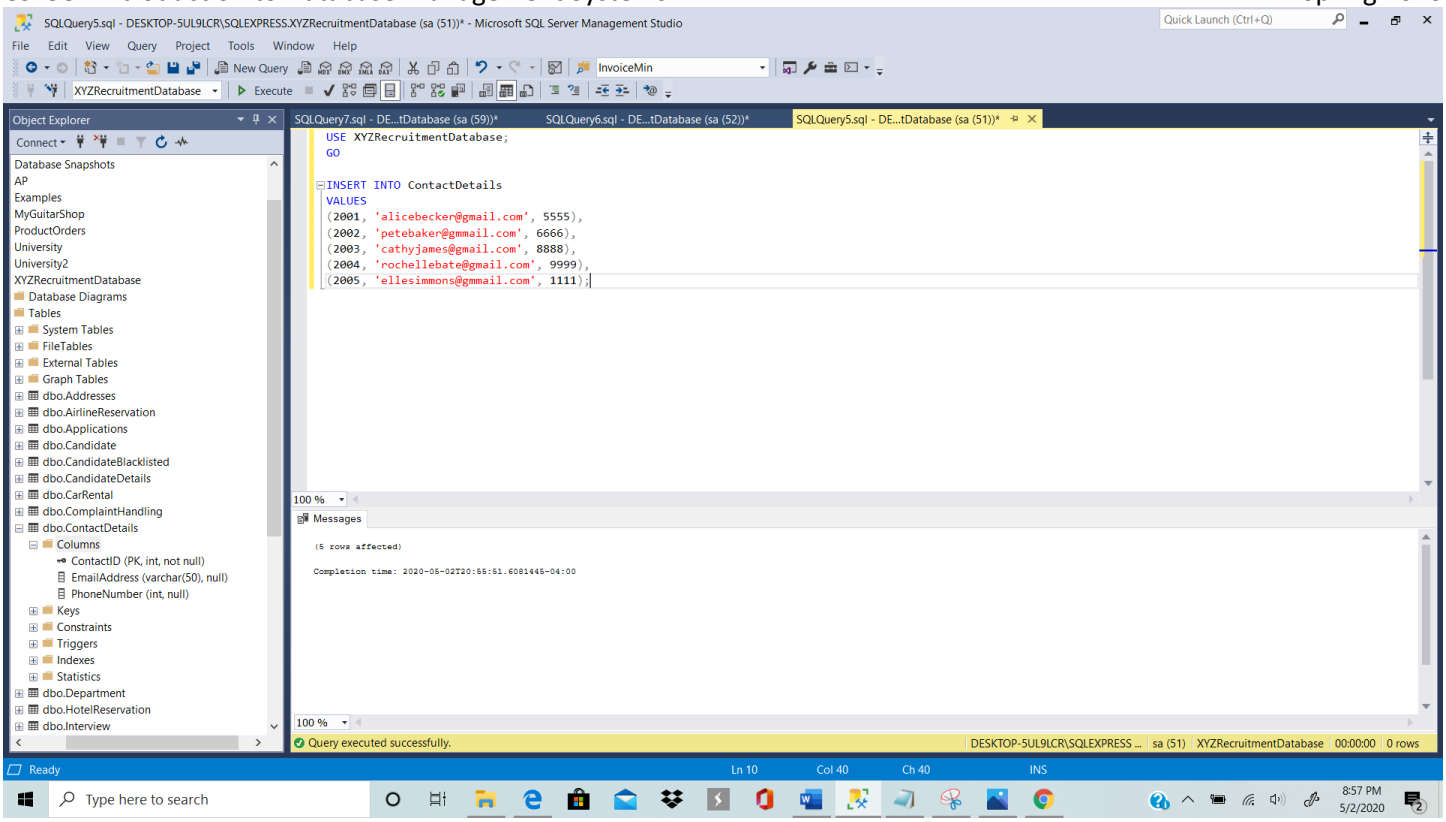
4. Testing Phase

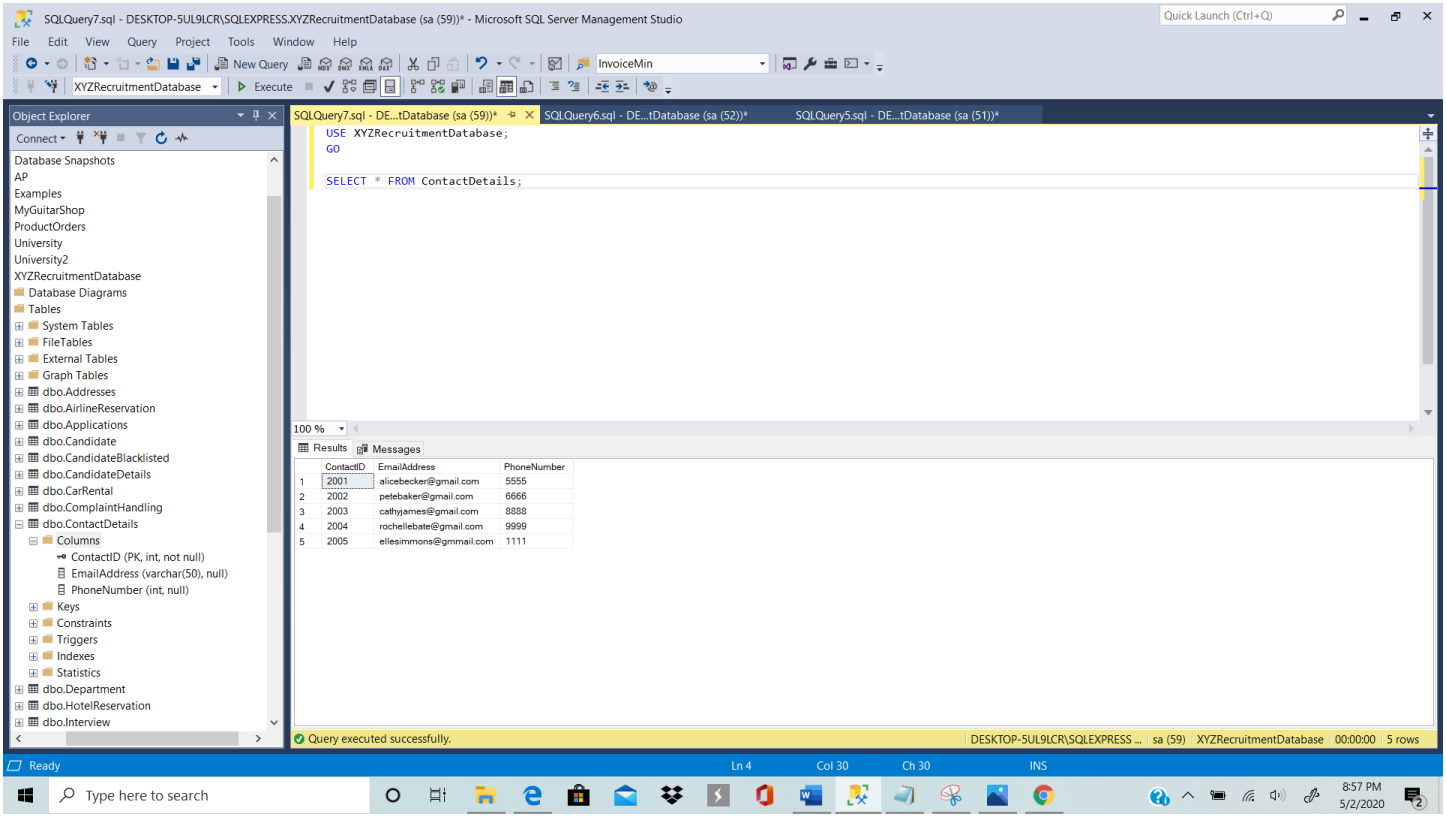
a. Populating the database with test data

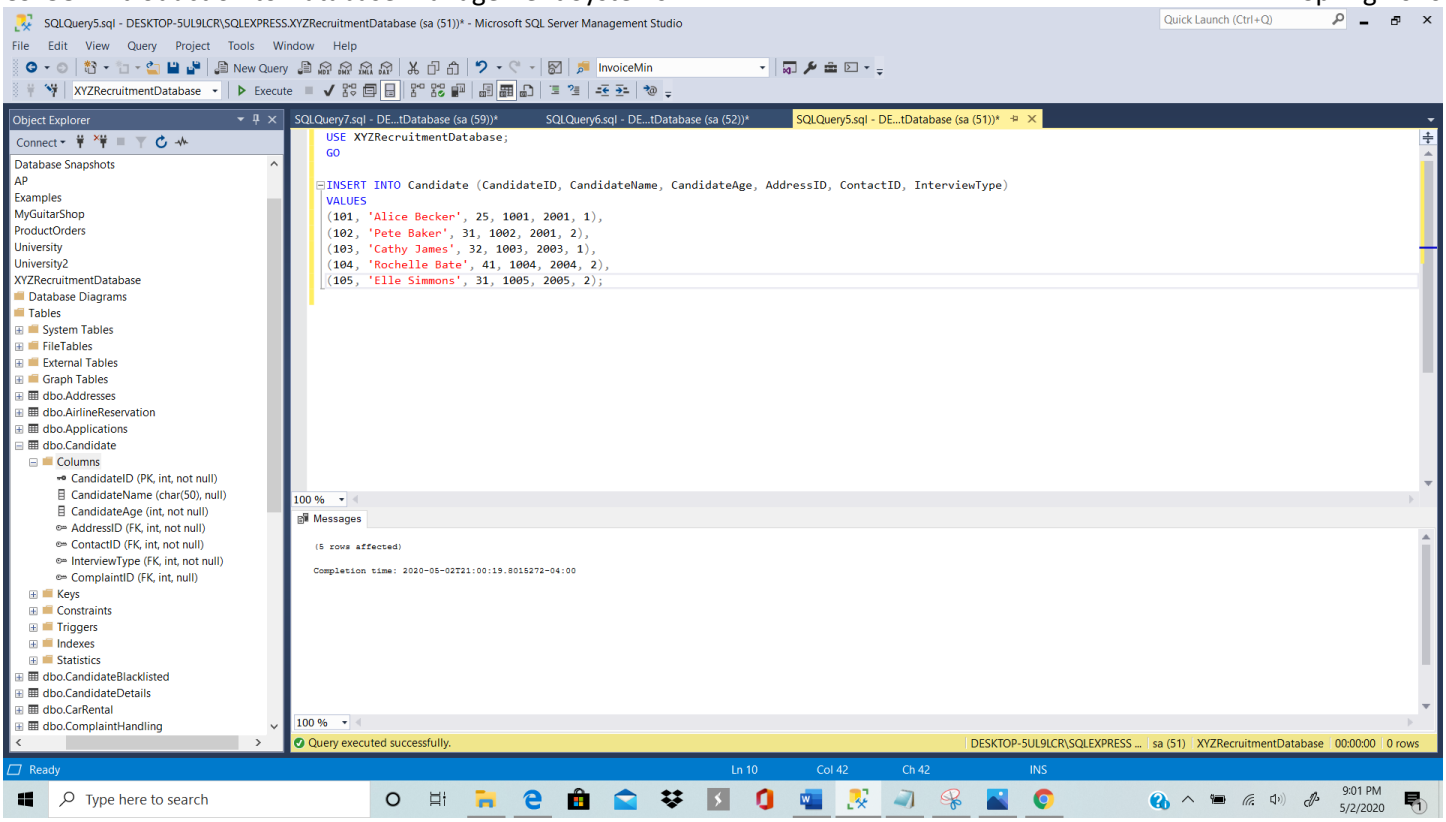
Here the initial step is to create tables, which has been shown earlier in the Implementation Phase. After creating tables, we insert test data or entries into the tables. Here are the screenshots of all the data that have been inserted into the database.

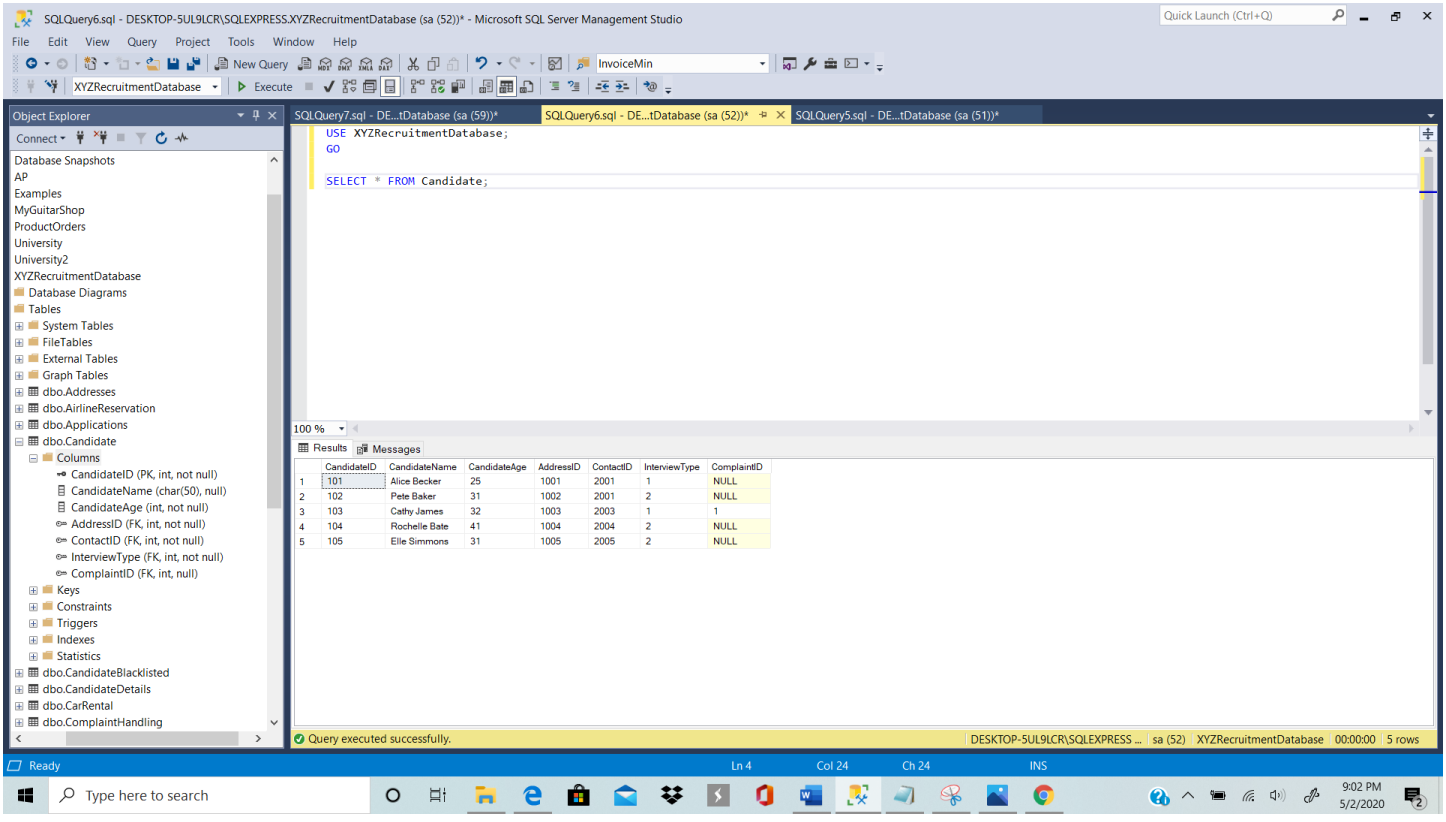


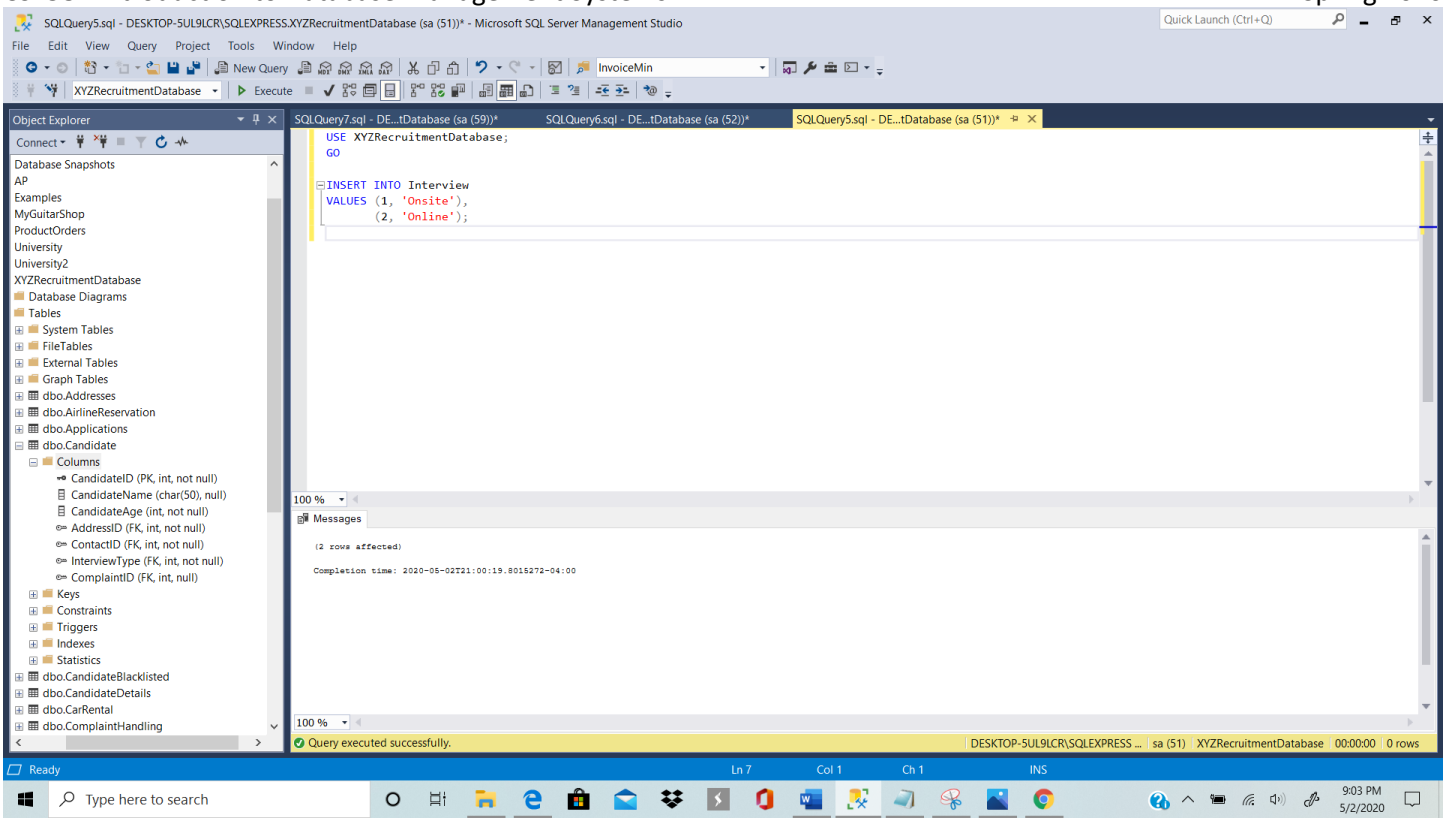


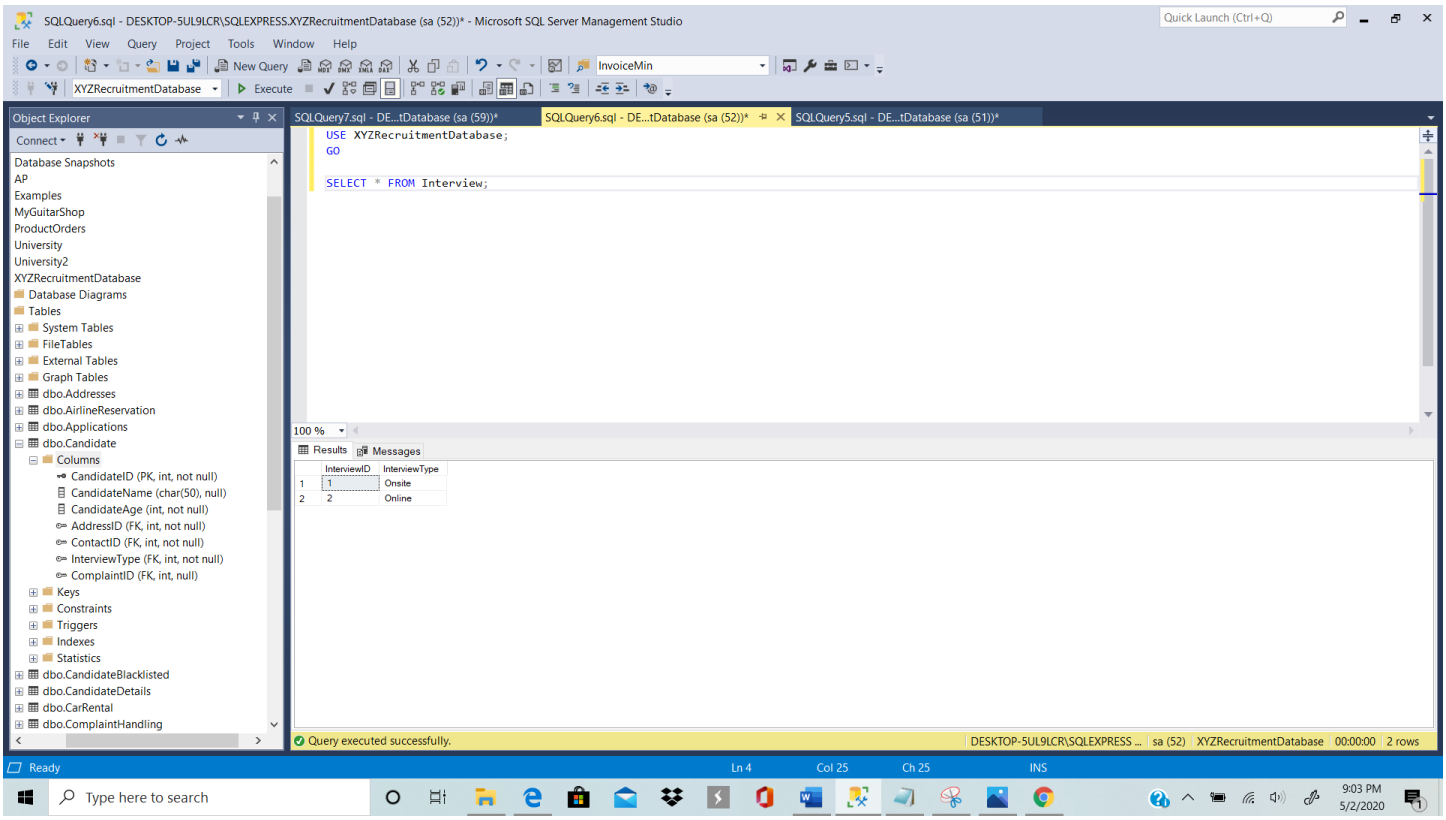

SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
INSERT INTO ContactDetails
VALUES
(2001, 'alicebecker@gmail.com', 5555),
(2002, 'petebaker@gmmail.com', 6666),
(2003, 'cathyjames@gmail.com', 8888),
(2004, 'rochellebate@gmail.com', 9999),
(2005, 'ellesimmons@gmmail.com', 1111);
(5 rows affected)
Completion time: 2020-05-02T20:55:51.6081445-04:00
Query executed successfully.

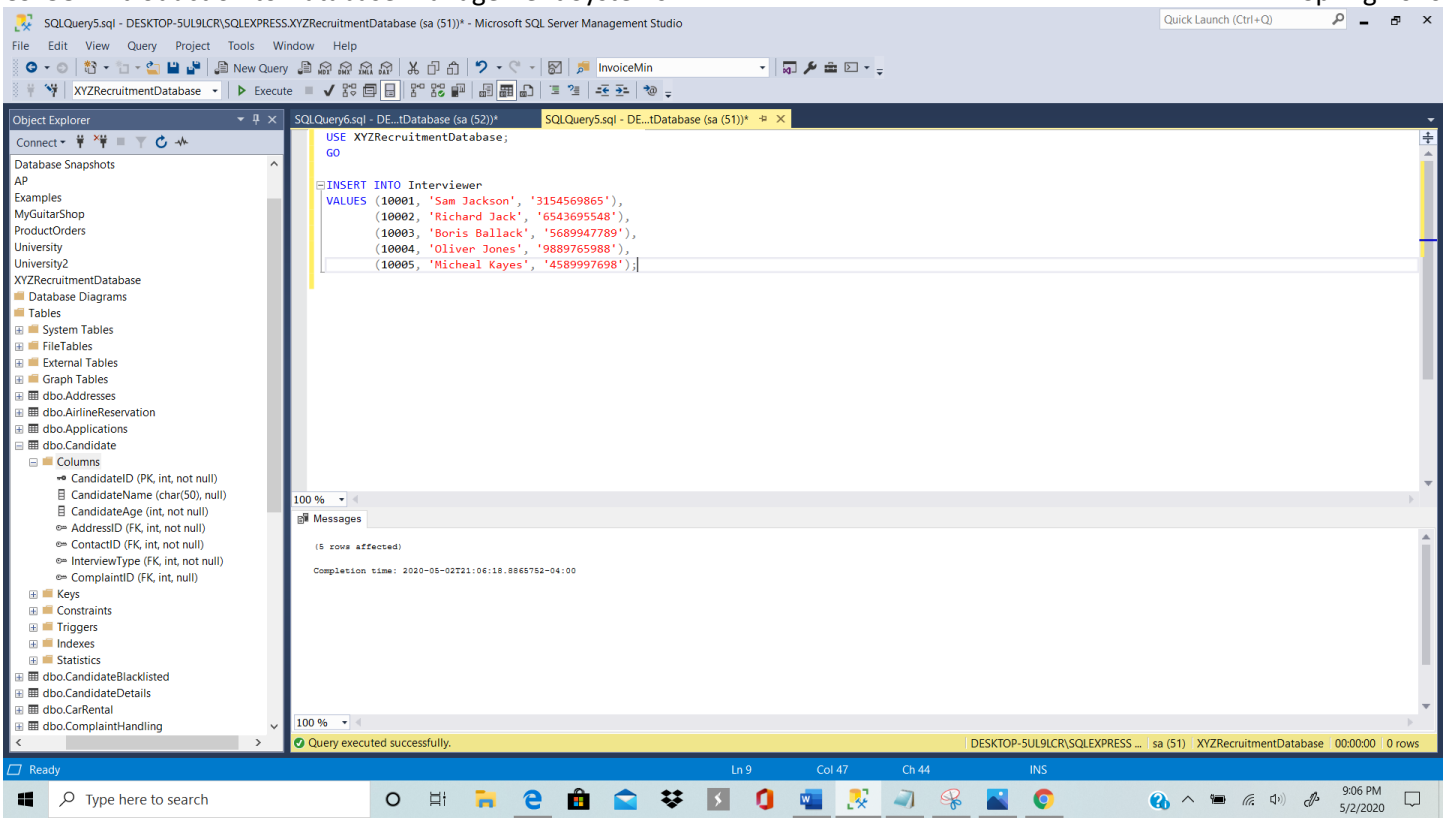

SQLQuery7.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (59))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
SELECT * FROM ContactDetails;
ContactID EmailAddress PhoneNumber
2001 alicebecker@gmail.com 5555
2002 petebaker@gmail.com 6666
2003 cathyjames@gmail.com 8888
2004 rochellebate@gmail.com 9999
2005 ellesimmons@gmmail.com 1111
Query executed successfully.

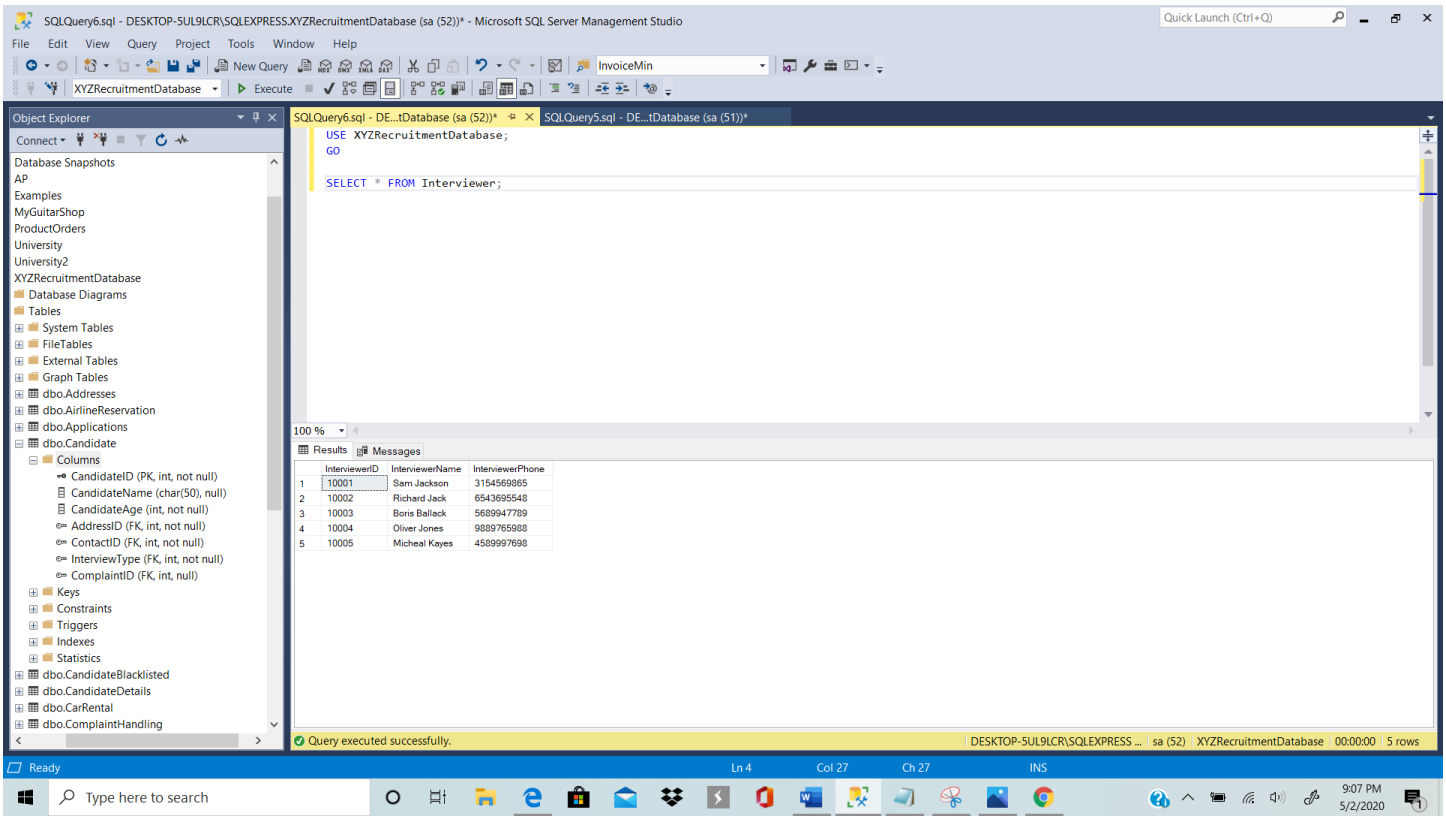

SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
INSERT INTO Candidate (CandidateID, CandidateName, CandidateAge, AddressID, ContactID, InterviewType)
VALUES
(101, 'Alice Becker', 25, 1001, 2001, 1),
(102, 'Pete Baker', 31, 1002, 2001, 2),
(103, 'Cathy James', 32, 1003, 2003, 1),
(104, 'Rochelle Bate', 41, 1004, 2004, 2),
(105, 'Elle Simmons', 31, 1005, 2005, 2);
(5 rows affected)
Completion time: 2020-05-02T21:00:19.8015272-04:00
Query executed successfully.
Ready
Ln 10 Col 42 Ch 42 INS

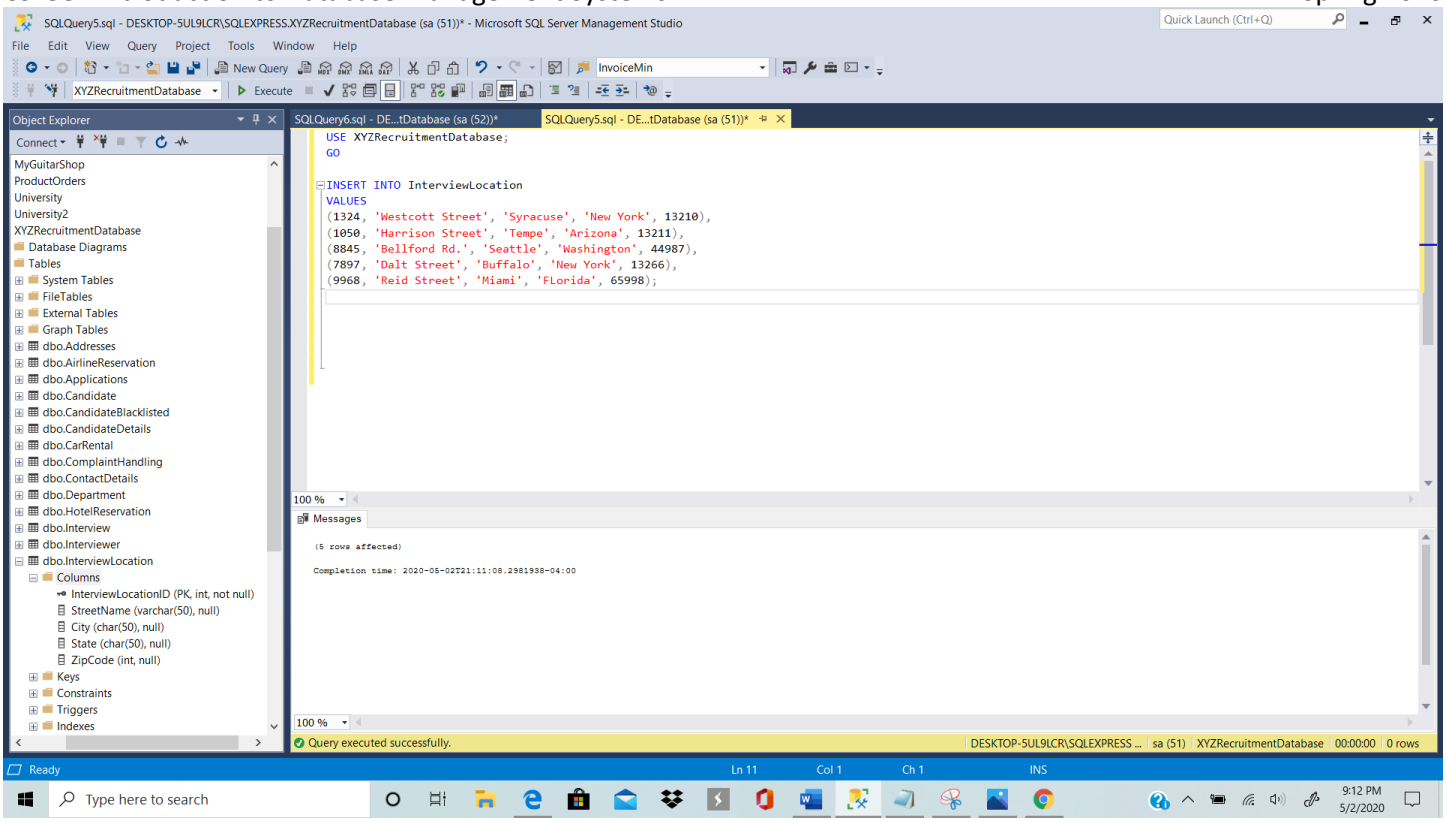

SQLQuery6.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
SELECT * FROM Candidate;
CandidateID CandidateName CandidateAge AddressID ContactID InterviewType ComplaintID
1 101 Alice Becker 25 1001 2001 1 NULL
2 102 Pete Baker 31 1002 2001 2 NULL
3 103 Cathy James 32 1003 2003 1 1
4 104 Rochelle Bate 41 1004 2004 2 NULL
5 105 Elle Simmons 31 1005 2005 2 NULL
Query executed successfully.
Ready
Ln 4 Col 24 Ch 24 INS

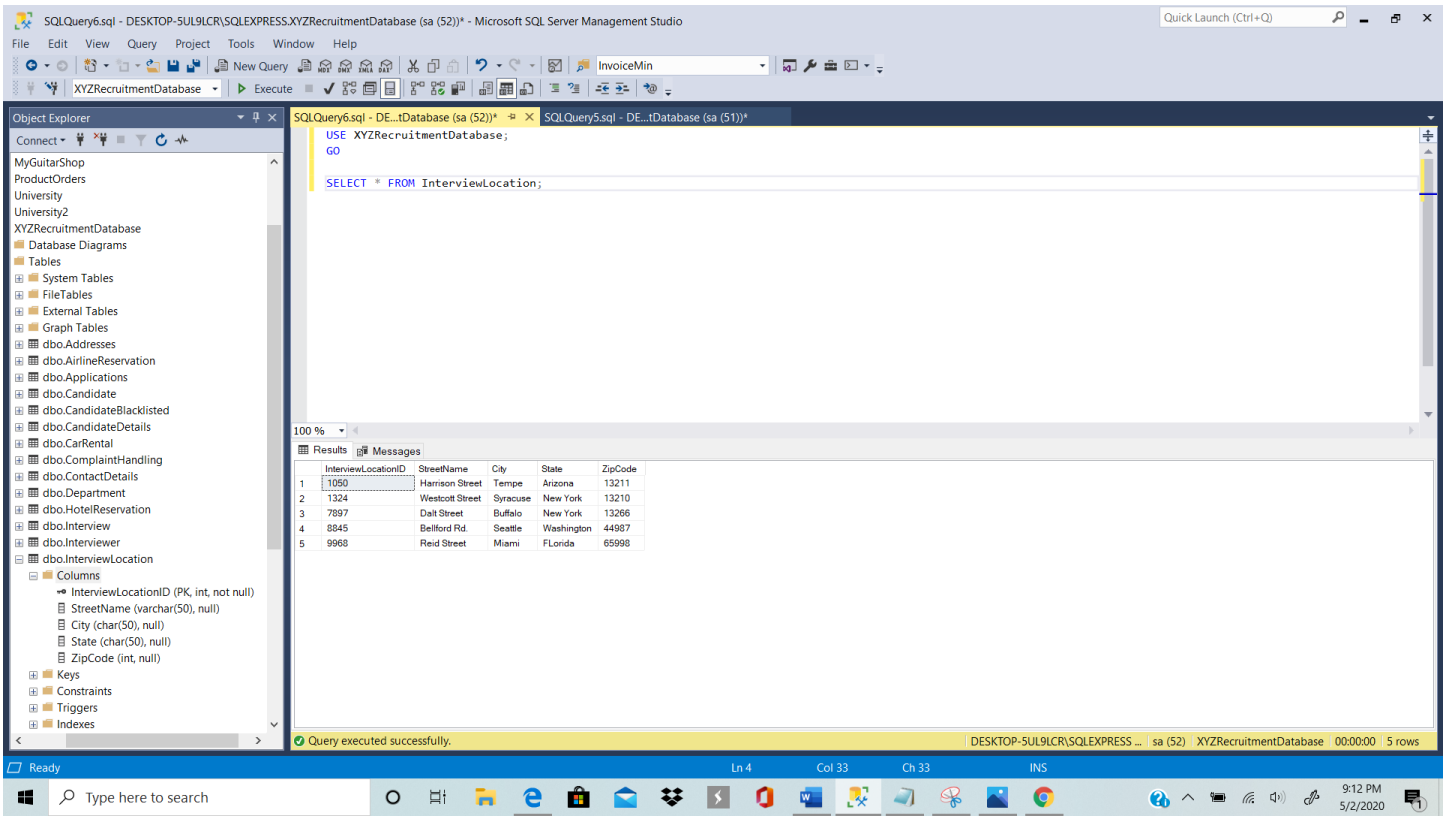

SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
INSERT INTO Interview
VALUES (1, 'Onsite'),
(2, 'Online');
(2 rows affected)
Completion time: 2020-05-02T21:00:19.8015272-04:00
Query executed successfully.

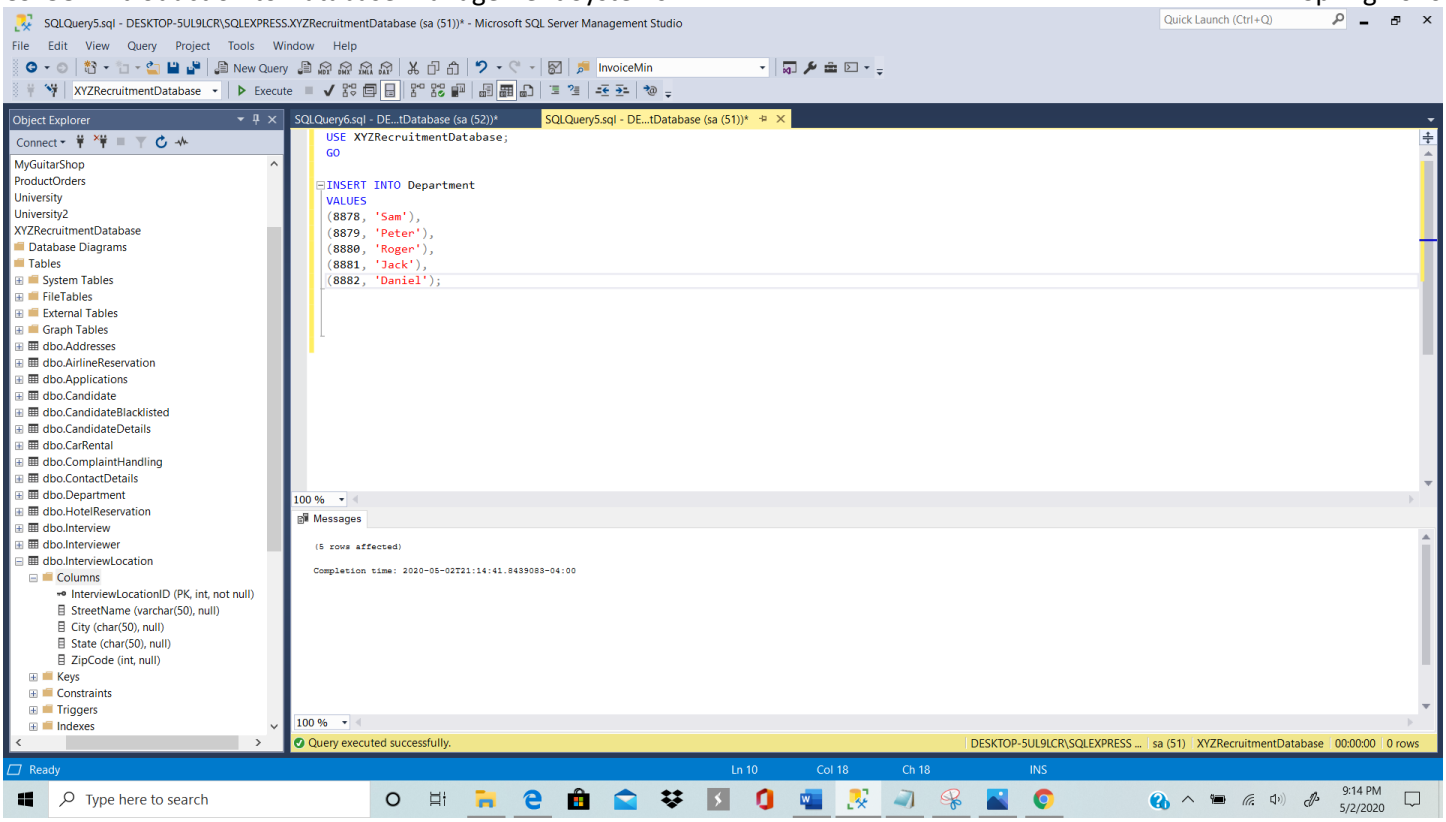

SQLQuery6.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
SELECT * FROM Interview;
InterviewID InterviewType
1 1 Onsite
2 2 Online
Query executed successfully.

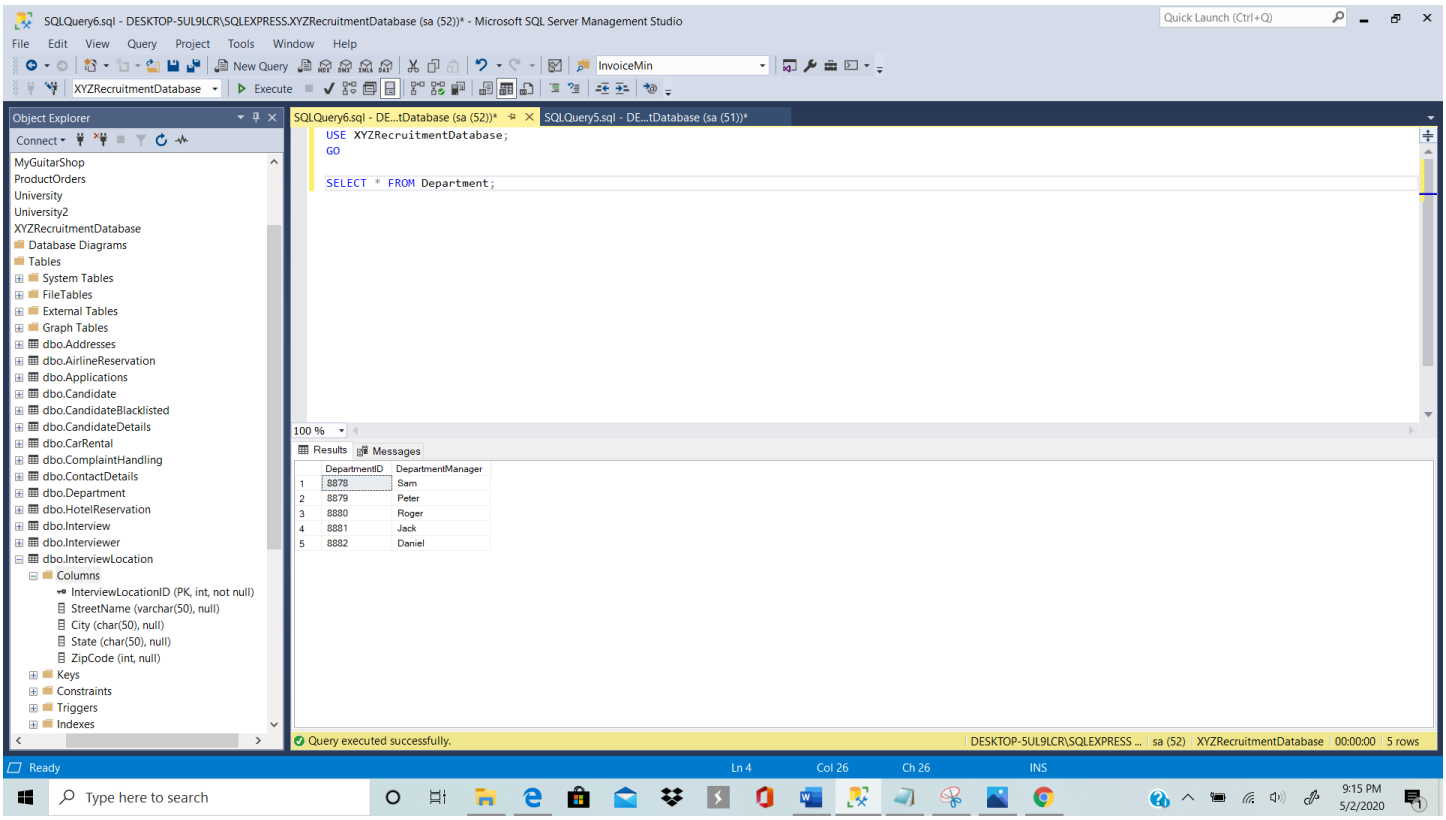

SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
Object Explorer
SQLQuery6.sql - DE...tDatabase (sa (52))*
SQLQuery5.sql - DE...tDatabase (sa (51))*
USE XYZRecruitmentDatabase;
GO
INSERT INTO Interviewer
VALUES (10001, 'Sam Jackson', '3154569865'),
(10002, 'Richard Jack', '6543695548'),
(10003, 'Boris Ballack', '5689947789'),
(10004, 'Oliver Jones', '9889765988'),
(10005, 'Micheal Kayes', '4589997698');
Messages
(5 rows affected)
Completion time: 2020-05-02T21:06:18.8865752-04:00
Query executed successfully.
Ready
Type here to search

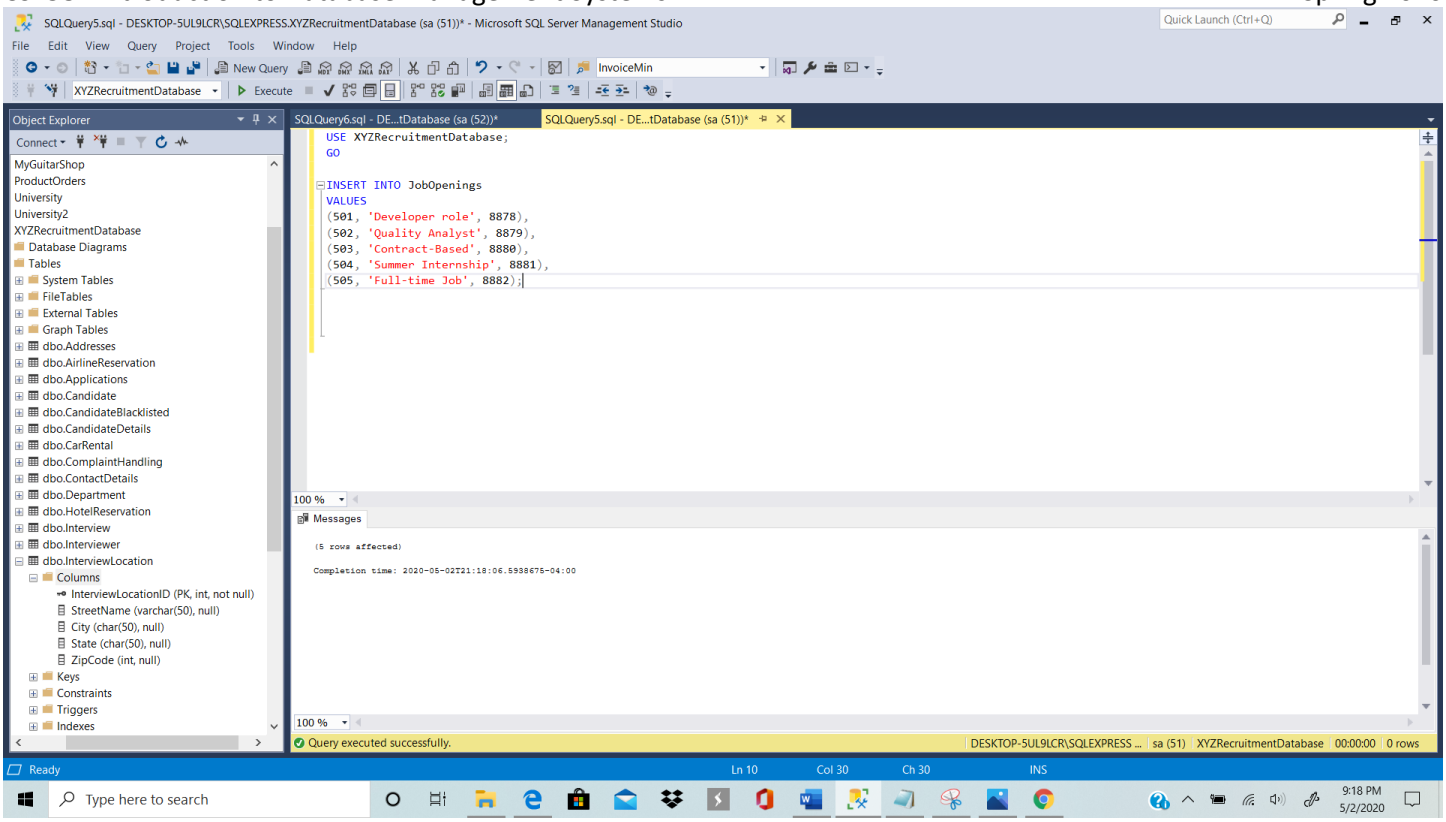

SQLQuery6.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
Object Explorer
SQLQuery6.sql - DE...tDatabase (sa (52))*
SQLQuery5.sql - DE...tDatabase (sa (51))*
USE XYZRecruitmentDatabase;
GO
SELECT * FROM Interviewer;
Results
Messages
InterviewerID InterviewerName InterviewerPhone
1 10001 Sam Jackson 3154569865
2 10002 Richard Jack 6543695548
3 10003 Boris Ballack 5689947789
4 10004 Oliver Jones 9889765988
5 10005 Micheal Kayes 4589997698
Query executed successfully.
Ready
Type here to search

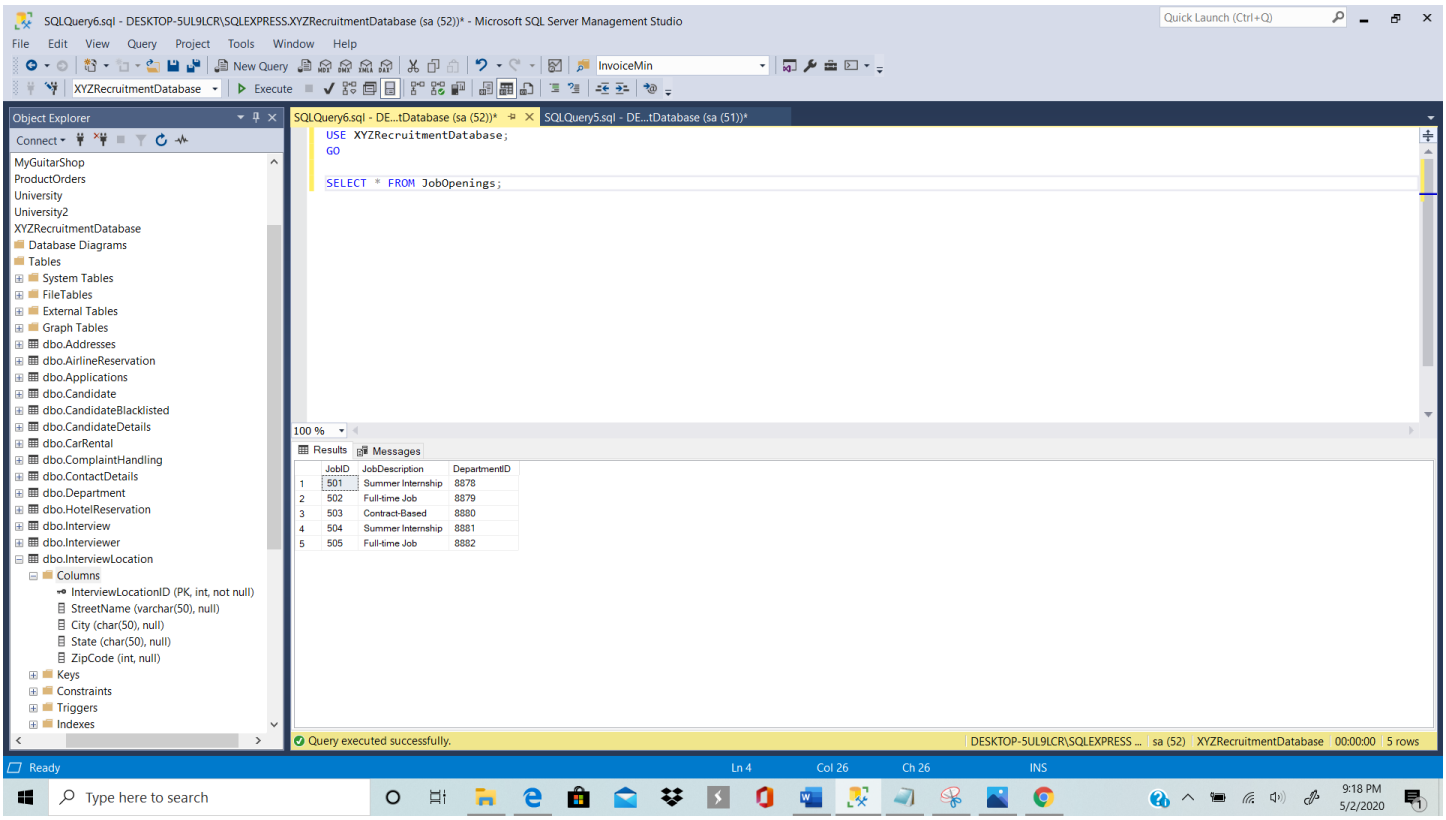

SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
Object Explorer
SQLQuery6.sql - DE...tDatabase (sa (52))*
SQLQuery5.sql - DE...tDatabase (sa (51))*
USE XYZRecruitmentDatabase;
GO
INSERT INTO InterviewLocation
VALUES
(1324, 'Westcott Street', 'Syracuse', 'New York', 13210),
(1050, 'Harrison Street', 'Tempe', 'Arizona', 13211),
(8845, 'Bellford Rd.', 'Seattle', 'Washington', 44987),
(7897, 'Dalt Street', 'Buffalo', 'New York', 13266),
(9968, 'Reid Street', 'Miami', 'FLorida', 65998);
Messages
(5 rows affected)
Completion time: 2020-05-02T21:11:08.2981938-04:00
Query executed successfully.
Ready

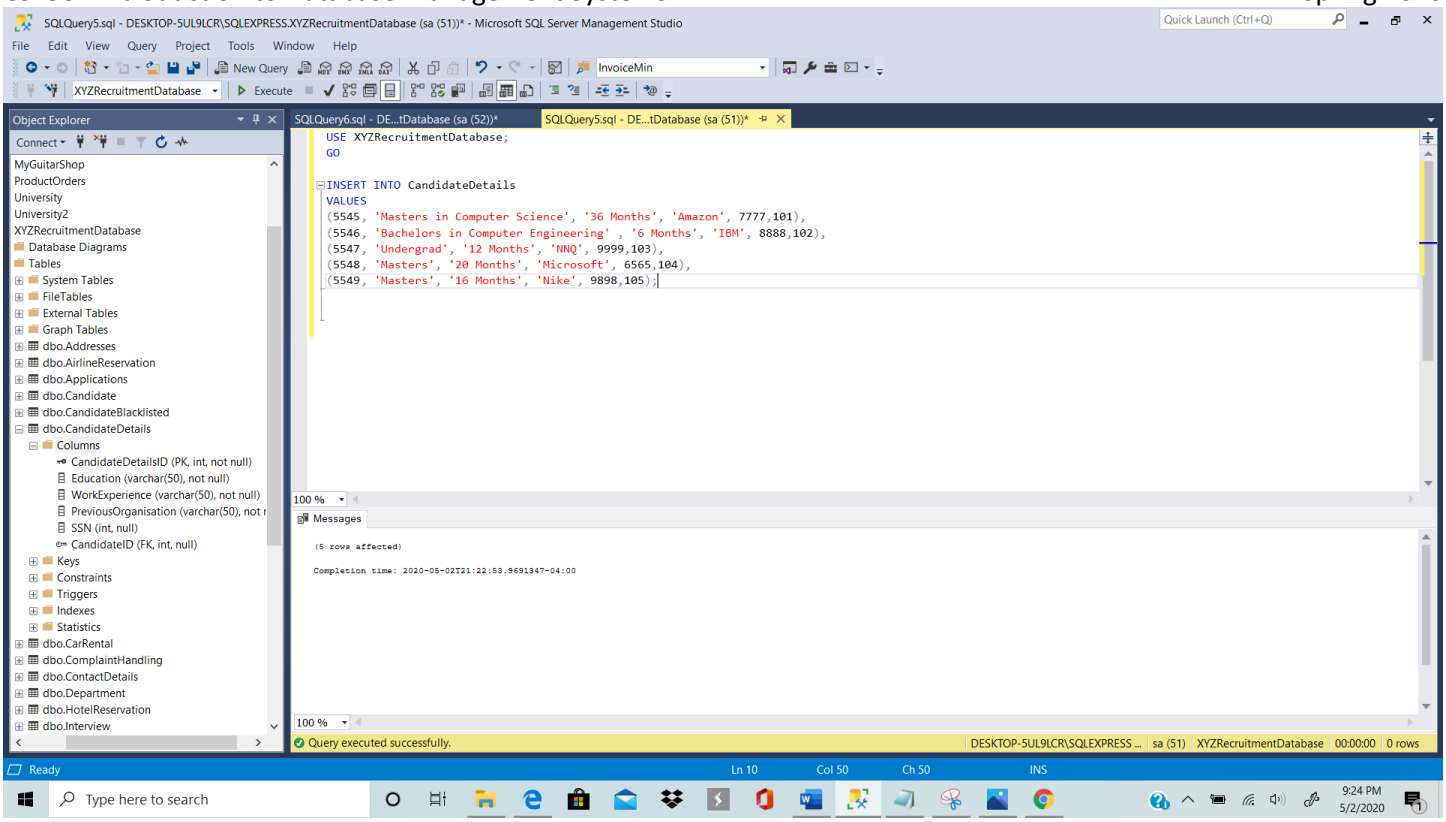

SQLQuery6.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
Object Explorer
SQLQuery6.sql - DE...tDatabase (sa (52))*
SQLQuery5.sql - DE...tDatabase (sa (51))*
USE XYZRecruitmentDatabase;
GO
SELECT * FROM InterviewLocation;
Results
Messages
InterviewLocationID StreetName City State ZipCode
1 1050 Harrison Street Tempe Arizona 13211
2 1324 Westcott Street Syracuse New York 13210
3 7897 Dalt Street Buffalo New York 13266
4 8845 Bellford Rd. Seattle Washington 44987
5 9968 Reid Street Miami FLorida 65998
Query executed successfully.
Ready

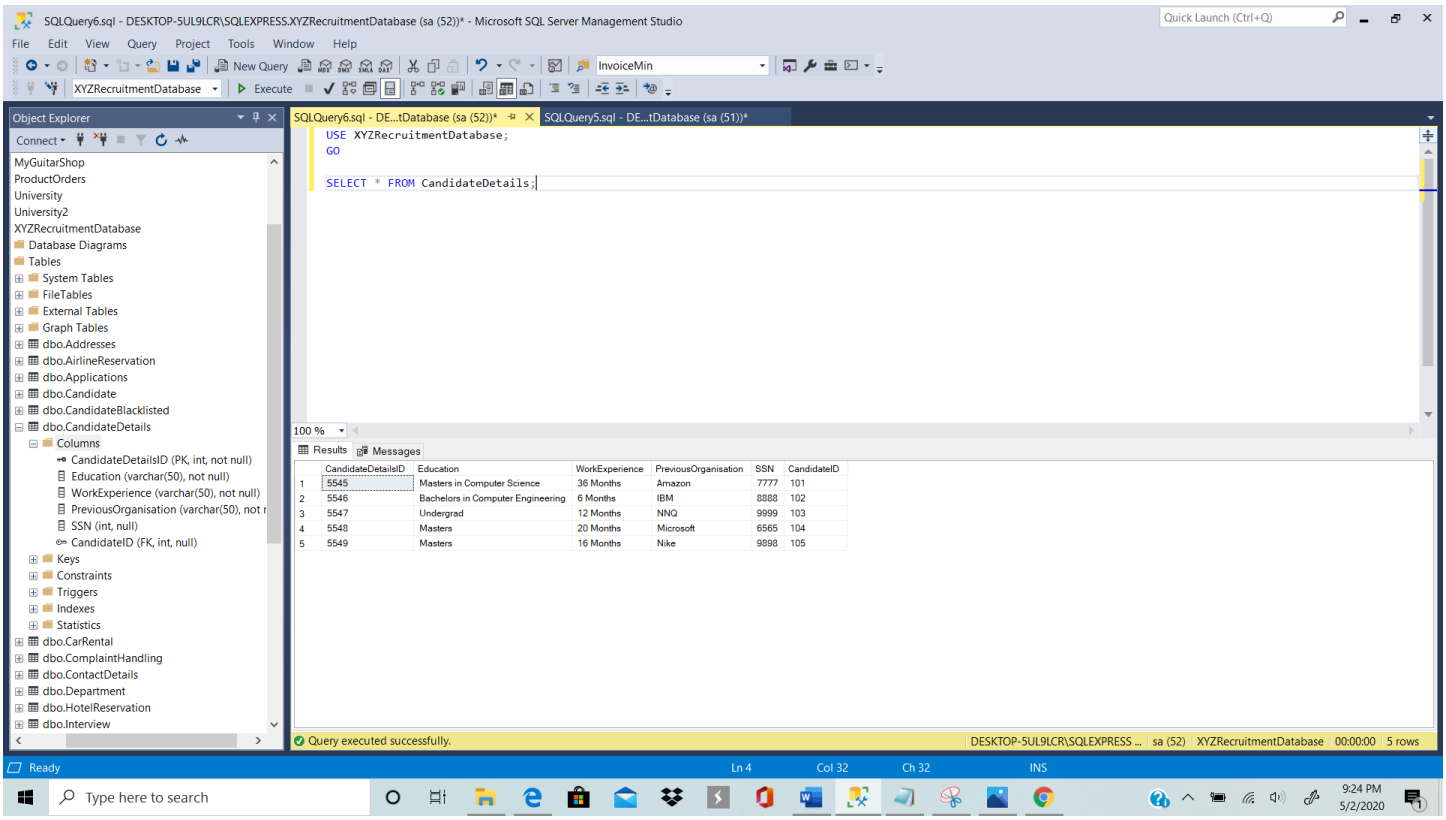

SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
INSERT INTO Department
VALUES
(8878, 'Sam'),
(8879, 'Peter'),
(8880, 'Roger'),
(8881, 'Jack'),
(8882, 'Daniel');
(5 rows affected)
Completion time: 2020-05-02T21:14:41.8439083-04:00
Query executed successfully.

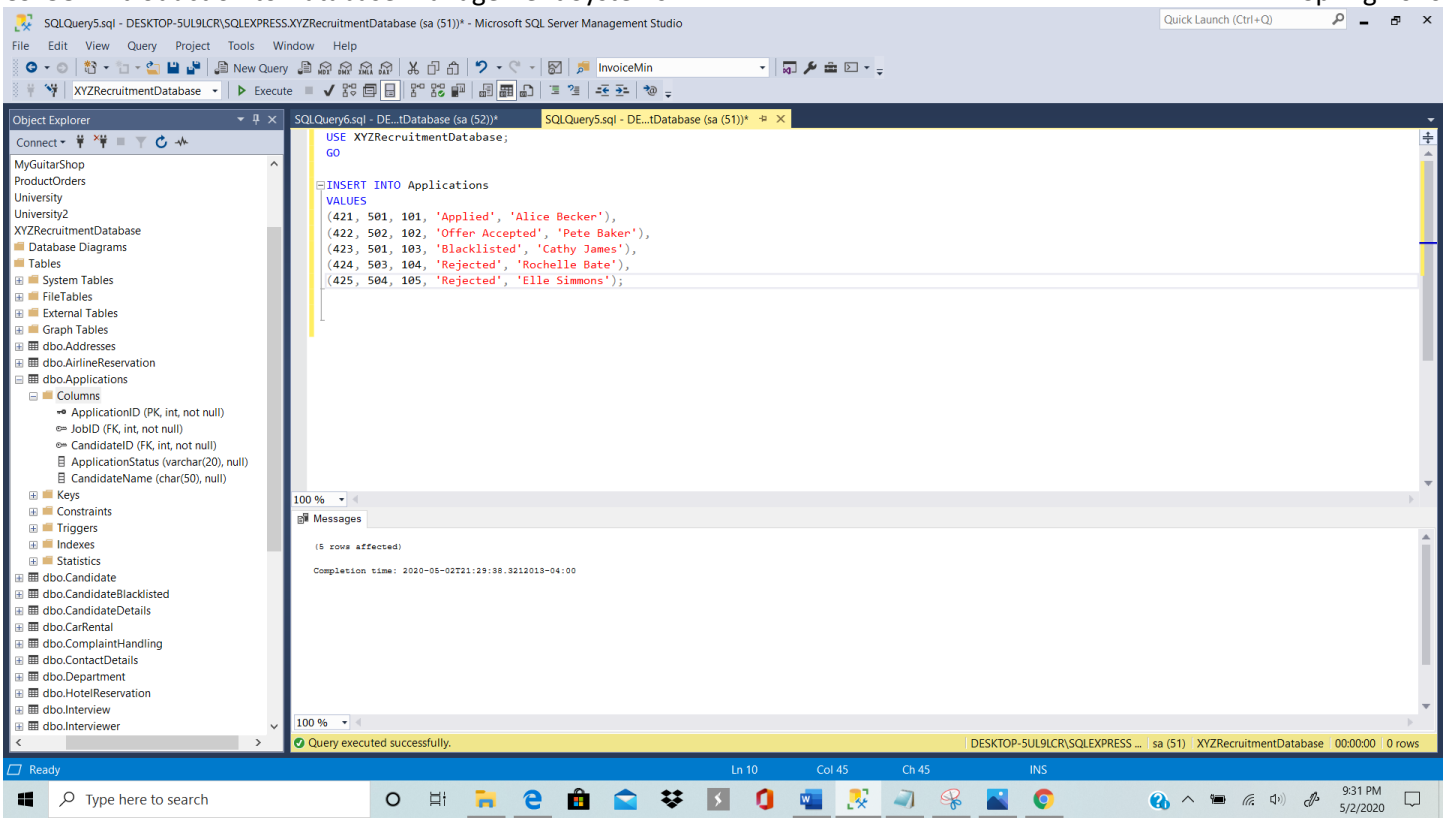

SQLQuery6.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
SELECT * FROM Department;
DepartmentID DepartmentManager
1 8878 Sam
2 8879 Peter
3 8880 Roger
4 8881 Jack
5 8882 Daniel
Query executed successfully.

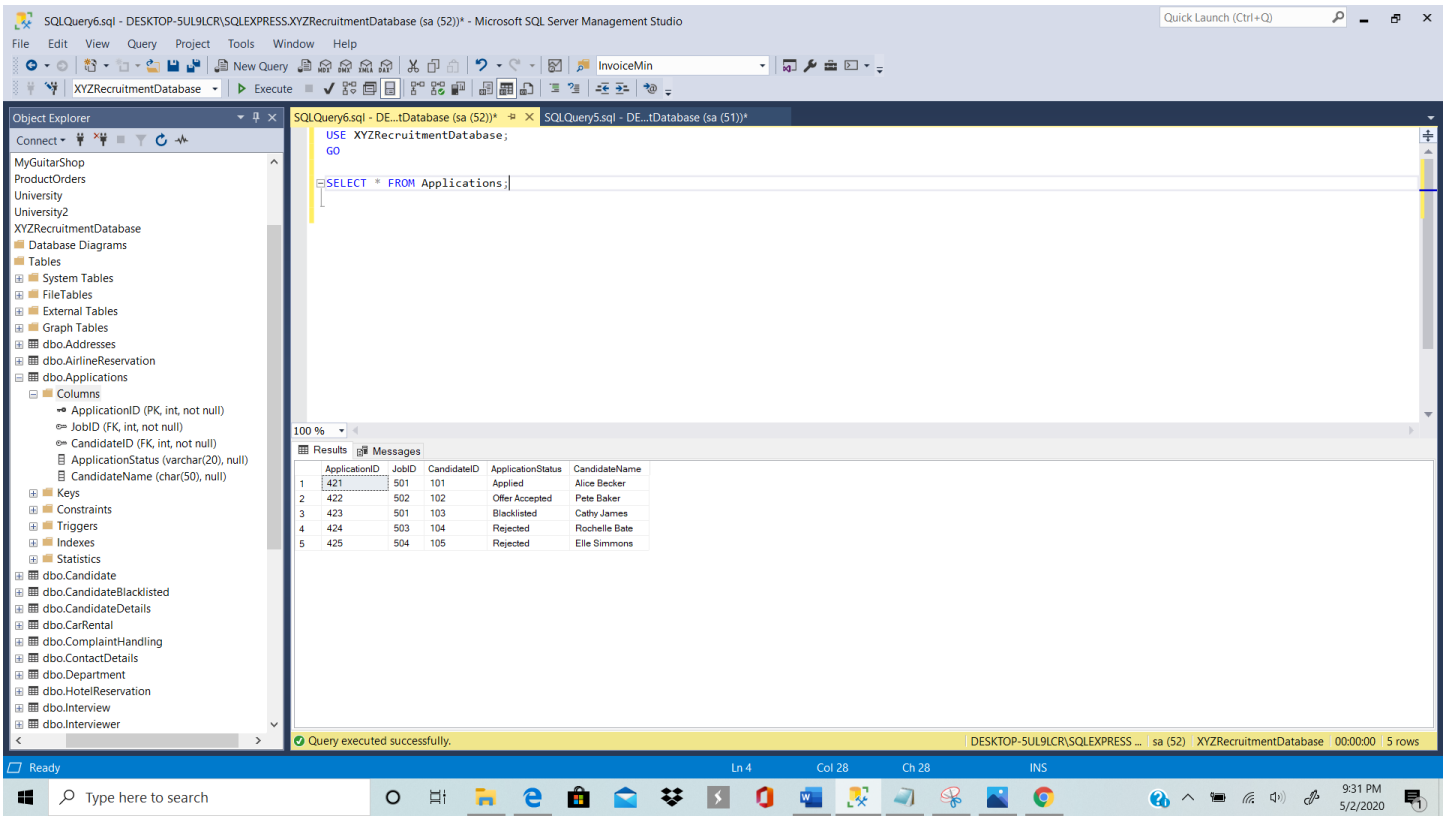

SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
InvoiceMin
XYZRecruitmentDatabase
Execute
Object Explorer
Connect
MyGuitarShop
ProductOrders
University
University2
XYZRecruitmentDatabase
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Addresses
dbo.AirlineReservation
dbo.Applications
dbo.Candidate
dbo.CandidateBlacklisted
dbo.CandidateDetails
dbo.CarRental
dbo.ComplaintHandling
dbo.ContactDetails
dbo.Department
dbo.HotelReservation
dbo.Interview
dbo.Interviewer
dbo.InterviewLocation
Columns
InterviewLocationID (PK, int, not null)
StreetName (varchar(50), null)
City (char(50), null)
State (char(50), null)
ZipCode (int, null)
Keys
Constraints
Triggers
Indexes
SQLQuery6.sql - DE...tDatabase (sa (52))*
SQLQuery5.sql - DE...tDatabase (sa (51))*
USE XYZRecruitmentDatabase;
GO
INSERT INTO JobOpenings
VALUES
(501, 'Developer role', 8878),
(502, 'Quality Analyst', 8879),
(503, 'Contract-Based', 8880),
(504, 'Summer Internship', 8881),
(505, 'Full-time Job', 8882);
100 %
Messages
(5 rows affected)
Completion time: 2020-05-02T21:18:06.5938675-04:00
Query executed successfully.
DESKTOP-5UL9LCR\SQLEXPRESS ... sa (51) XYZRecruitmentDatabase 00:00:00 0 rows
Ready
Ln 10 Col 30 Ch 30 INS
Type here to search
9:18 PM
5/2/2020

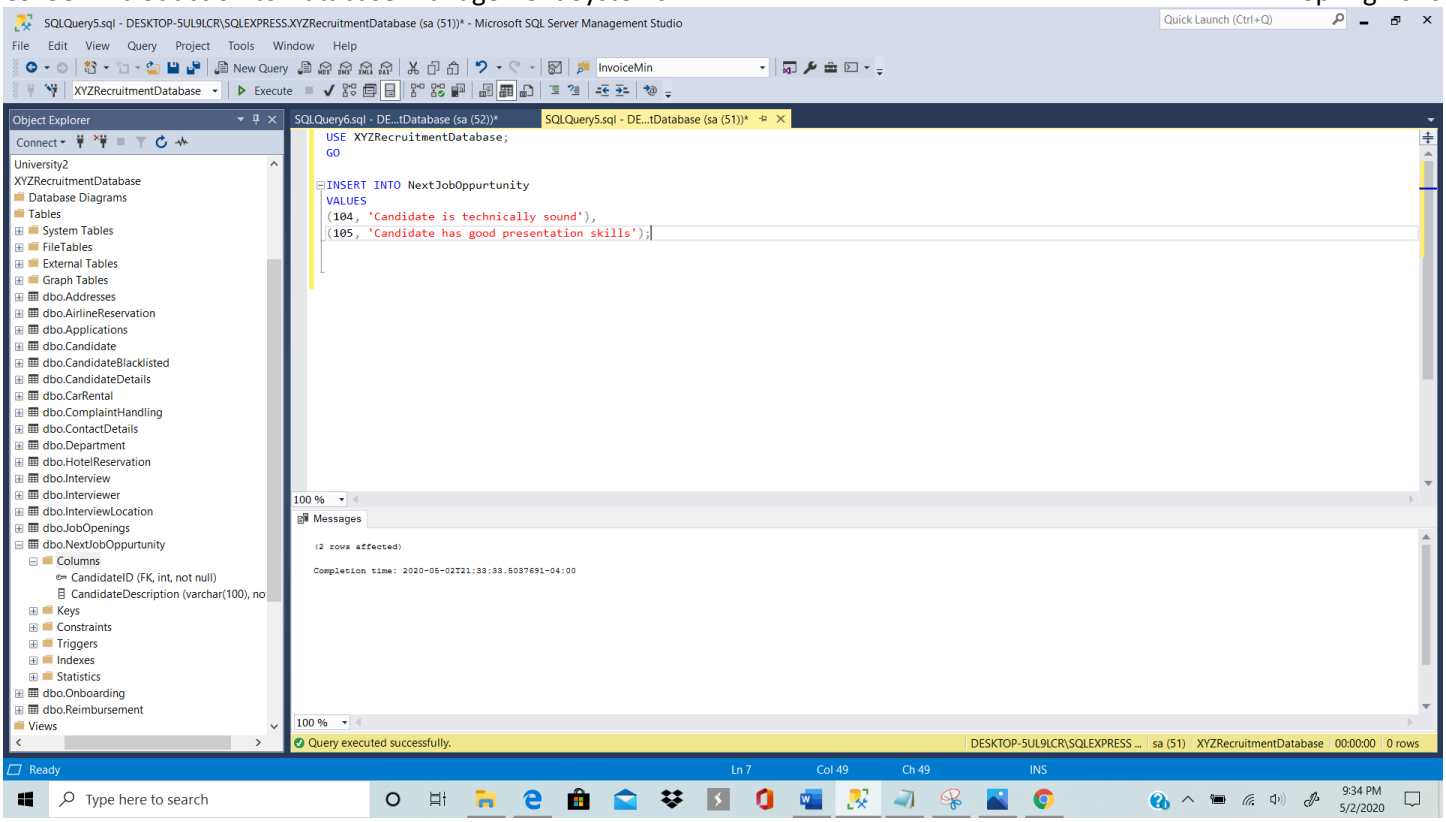

SQLQuery6.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
InvoiceMin
XYZRecruitmentDatabase
Execute
Object Explorer
Connect
MyGuitarShop
ProductOrders
University
University2
XYZRecruitmentDatabase
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Addresses
dbo.AirlineReservation
dbo.Applications
dbo.Candidate
dbo.CandidateBlacklisted
dbo.CandidateDetails
dbo.CarRental
dbo.ComplaintHandling
dbo.ContactDetails
dbo.Department
dbo.HotelReservation
dbo.Interview
dbo.Interviewer
dbo.InterviewLocation
Columns
InterviewLocationID (PK, int, not null)
StreetName (varchar(50), null)
City (char(50), null)
State (char(50), null)
ZipCode (int, null)
Keys
Constraints
Triggers
Indexes
SQLQuery6.sql - DE...tDatabase (sa (52))*
SQLQuery5.sql - DE...tDatabase (sa (51))*
USE XYZRecruitmentDatabase;
GO
SELECT * FROM JobOpenings;
100 %
Results
Messages
JobID JobDescription DepartmentID
1 501 Summer Internship 8878
2 502 Full-time Job 8879
3 503 Contract-Based 8880
4 504 Summer Internship 8881
5 505 Full-time Job 8882
Query executed successfully.
DESKTOP-5UL9LCR\SQLEXPRESS ... sa (52) XYZRecruitmentDatabase 00:00:00 5 rows
Ready
Ln 4 Col 26 Ch 26 INS
Type here to search
9:18 PM
5/2/2020

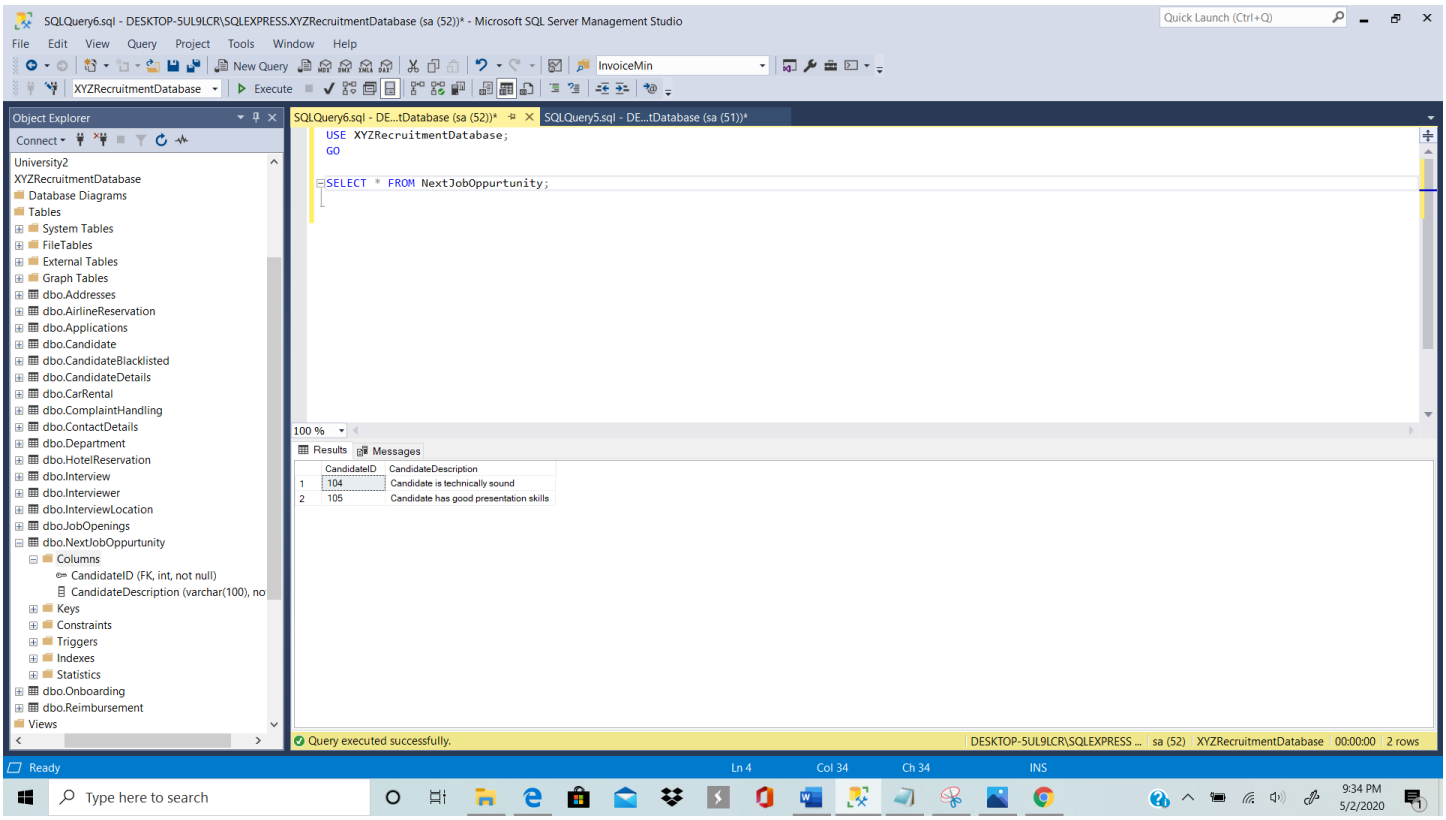

SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
InvoiceMin
XYZRecruitmentDatabase
Execute
Object Explorer
Connect
MyGuitarShop
ProductOrders
University
University2
XYZRecruitmentDatabase
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Addresses
dbo.AirlineReservation
dbo.Applications
dbo.Candidate
dbo.CandidateBlacklisted
dbo.CandidateDetails
Columns
CandidateDetailsID (PK, int, not null)
Education (varchar(50), not null)
WorkExperience (varchar(50), not null)
PreviousOrganisation (varchar(50), not
SSN (int, null)
CandidateID (FK, int, null)
Keys
Constraints
Triggers
Indexes
Statistics
dbo.CarRental
dbo.ComplaintHandling
dbo.ContactDetails
dbo.Department
dbo.HotelReservation
dbo.Interview
SQLQuery6.sql - DE...tDatabase (sa (52))*
SQLQuery5.sql - DE...tDatabase (sa (51))*
USE XYZRecruitmentDatabase;
GO
INSERT INTO CandidateDetails
VALUES
(5545, 'Masters in Computer Science', '36 Months', 'Amazon', 7777,101),
(5546, 'Bachelors in Computer Engineering' , '6 Months', 'IBM', 8888,102),
(5547, 'Undergrad', '12 Months', 'NNQ', 9999,103),
(5548, 'Masters', '20 Months', 'Microsoft', 6565,104),
(5549, 'Masters', '16 Months', 'Nike', 9898,105);
100 %
Messages
(5 rows affected)
Completion time: 2020-05-02T21:22:53.9691347-04:00
100 %
Query executed successfully.
DESKTOP-5UL9LCR\SQLEXPRESS ... sa (51) XYZRecruitmentDatabase 00:00:00 0 rows
Ready
Ln 10 Col 50 Ch 50 INS
Type here to search
9:24 PM
5/2/2020

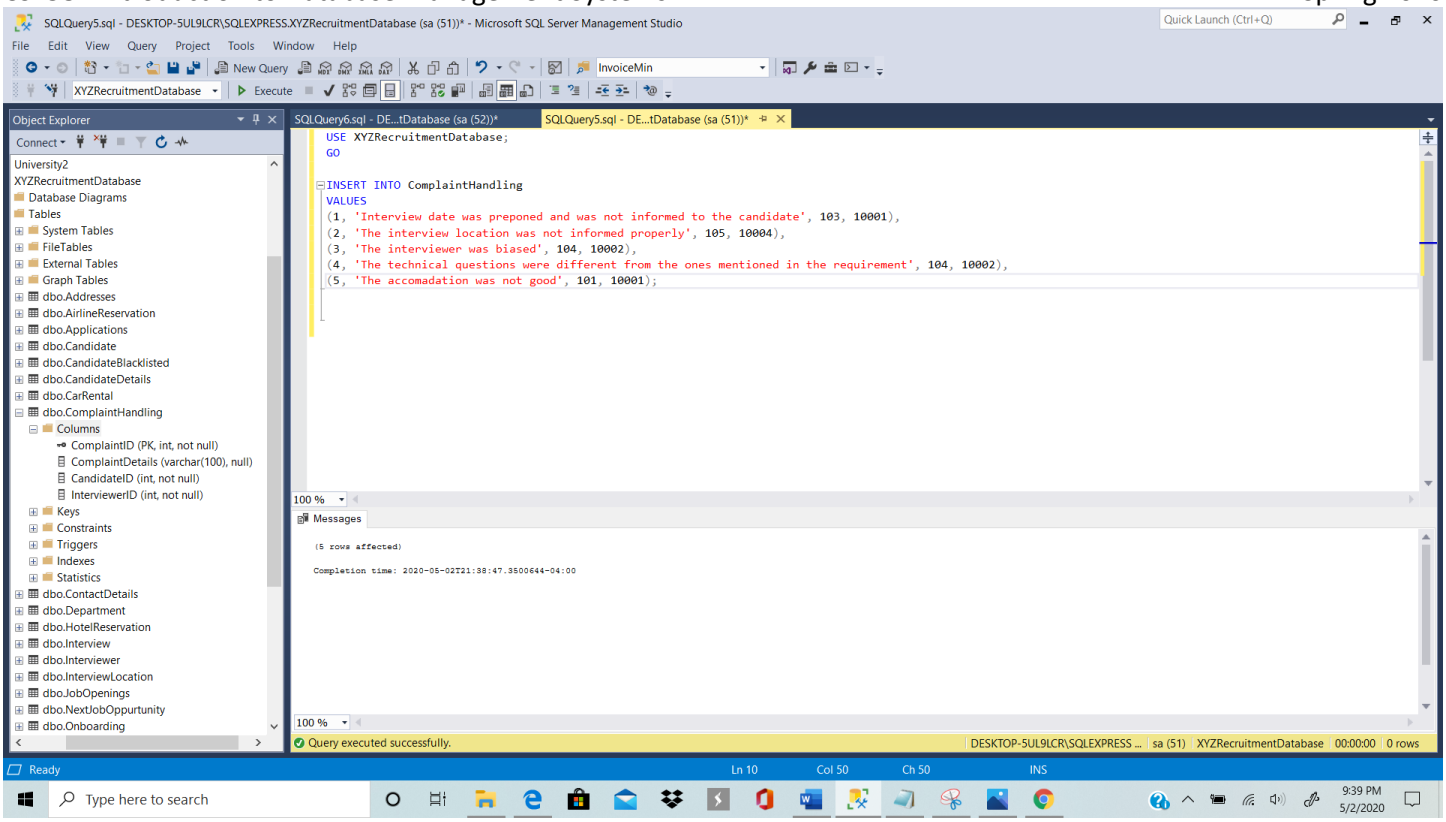

SQLQuery6.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
InvoiceMin
XYZRecruitmentDatabase
Execute
Object Explorer
Connect
MyGuitarShop
ProductOrders
University
University2
XYZRecruitmentDatabase
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Addresses
dbo.AirlineReservation
dbo.Applications
dbo.Candidate
dbo.CandidateBlacklisted
dbo.CandidateDetails
Columns
CandidateDetailsID (PK, int, not null)
Education (varchar(50), not null)
WorkExperience (varchar(50), not null)
PreviousOrganisation (varchar(50), not
SSN (int, null)
CandidateID (FK, int, null)
Keys
Constraints
Triggers
Indexes
Statistics
dbo.CarRental
dbo.ComplaintHandling
dbo.ContactDetails
dbo.Department
dbo.HotelReservation
dbo.Interview
SQLQuery6.sql - DE...tDatabase (sa (52))*
SQLQuery5.sql - DE...tDatabase (sa (51))*
USE XYZRecruitmentDatabase;
GO
SELECT * FROM CandidateDetails;
100 %
Results
Messages
CandidateDetailsID Education WorkExperience PreviousOrganisation SSN CandidateID
1 5545 Masters in Computer Science 36 Months Amazon 7777 101
2 5546 Bachelors in Computer Engineering 6 Months IBM 8888 102
3 5547 Undergrad 12 Months NNQ 9999 103
4 5548 Masters 20 Months Microsoft 6565 104
5 5549 Masters 16 Months Nike 9898 105
Query executed successfully.
DESKTOP-5UL9LCR\SQLEXPRESS ... sa (52) XYZRecruitmentDatabase 00:00:00 5 rows
Ready
Ln 4 Col 32 Ch 32 INS
Type here to search
9:24 PM
5/2/2020

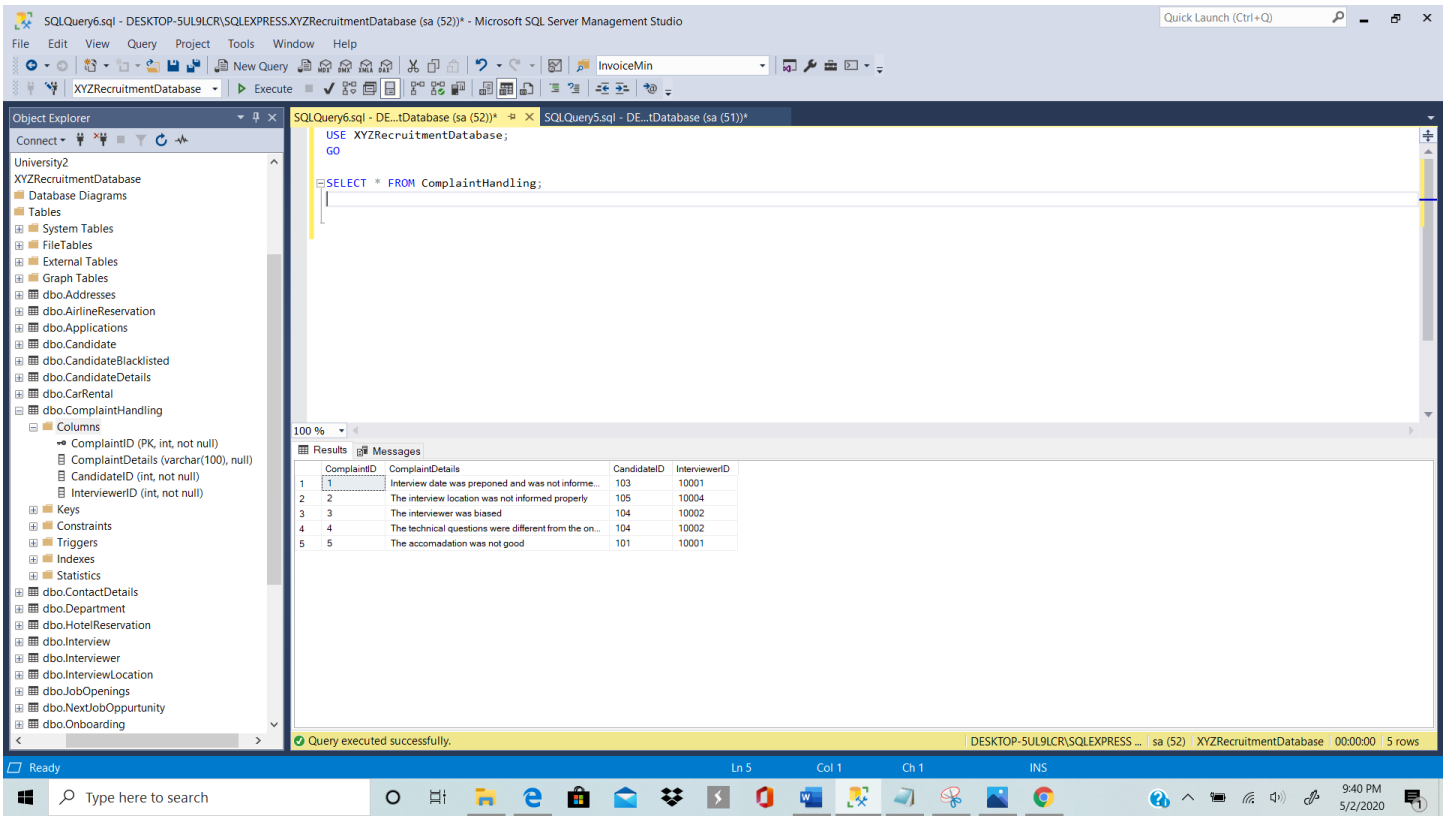

SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
INSERT INTO Applications
VALUES
(421, 501, 101, 'Applied', 'Alice Becker'),
(422, 502, 102, 'Offer Accepted', 'Pete Baker'),
(423, 501, 103, 'Blacklisted', 'Cathy James'),
(424, 503, 104, 'Rejected', 'Rochelle Bate'),
(425, 504, 105, 'Rejected', 'Elle Simmons');
(5 rows affected)
Completion time: 2020-05-02T21:29:38.3212013-04:00
Query executed successfully.

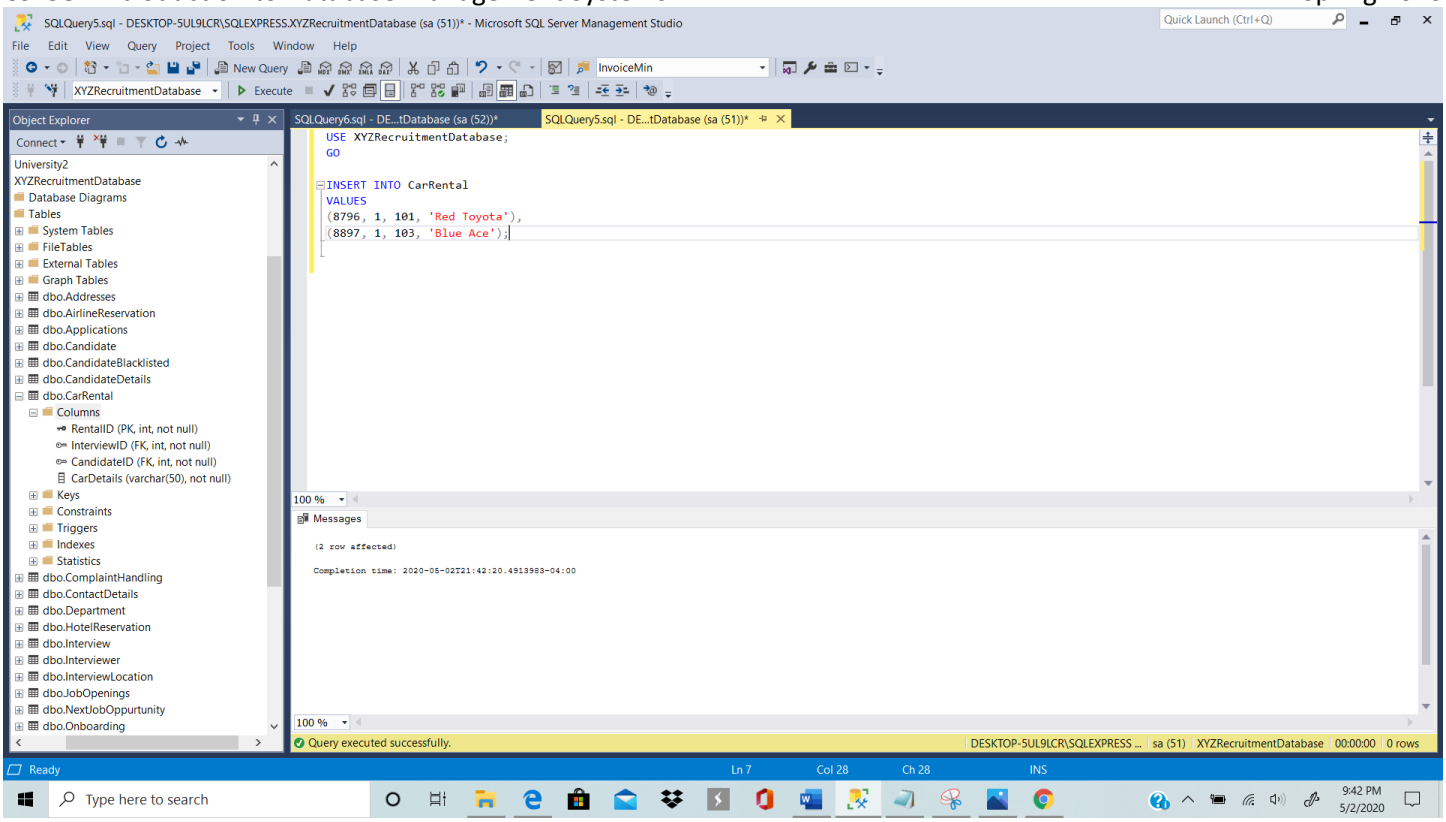

SQLQuery6.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
SELECT * FROM Applications;
ApplicationID JobID CandidateID ApplicationStatus CandidateName
1 421 501 101 Applied Alice Becker
2 422 502 102 Offer Accepted Pete Baker
3 423 501 103 Blacklisted Cathy James
4 424 503 104 Rejected Rochelle Bate
5 425 504 105 Rejected Elle Simmons
Query executed successfully.

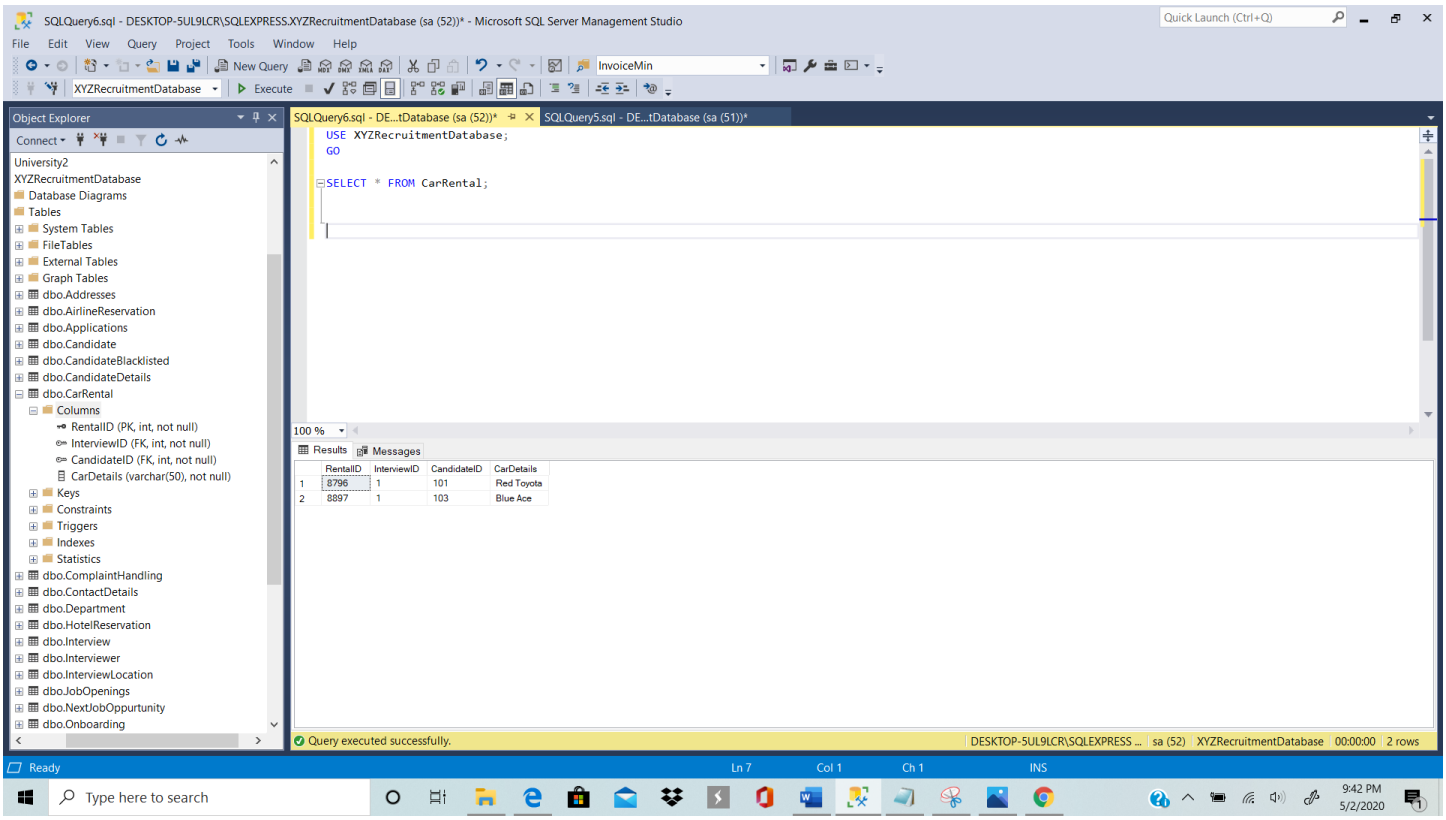

SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
InvoiceMin
XYZRecruitmentDatabase
Execute
Object Explorer
Connect
University2
XYZRecruitmentDatabase
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Addresses
dbo.AirlineReservation
dbo.Applications
dbo.Candidate
dbo.CandidateBlacklisted
dbo.CandidateDetails
dbo.CarRental
dbo.ComplaintHandling
dbo.ContactDetails
dbo.Department
dbo.HotelReservation
dbo.Interview
dbo.Interviewer
dbo.InterviewLocation
dbo.JobOpenings
dbo.NextJobOppurtunity
Columns
CandidateID (FK, int, not null)
CandidateDescription (varchar(100), no
Keys
Constraints
Triggers
Indexes
Statistics
dbo.Onboarding
dbo.Reimbursement
Views
SQLQuery6.sql - DE...tDatabase (sa (52))*
SQLQuery5.sql - DE...tDatabase (sa (51))*
USE XYZRecruitmentDatabase;
GO
INSERT INTO NextJobOppurtunity
VALUES
(104, 'Candidate is technically sound'),
(105, 'Candidate has good presentation skills');
100 %
Messages
(2 rows affected)
Completion time: 2020-05-02T21:33:33.5037691-04:00
100 %
Query executed successfully.
DESKTOP-5UL9LCR\SQLEXPRESS ... sa (51) XYZRecruitmentDatabase 00:00:00 0 rows
Ready
Ln 7 Col 49 Ch 49 INS
Type here to search
9:34 PM
5/2/2020

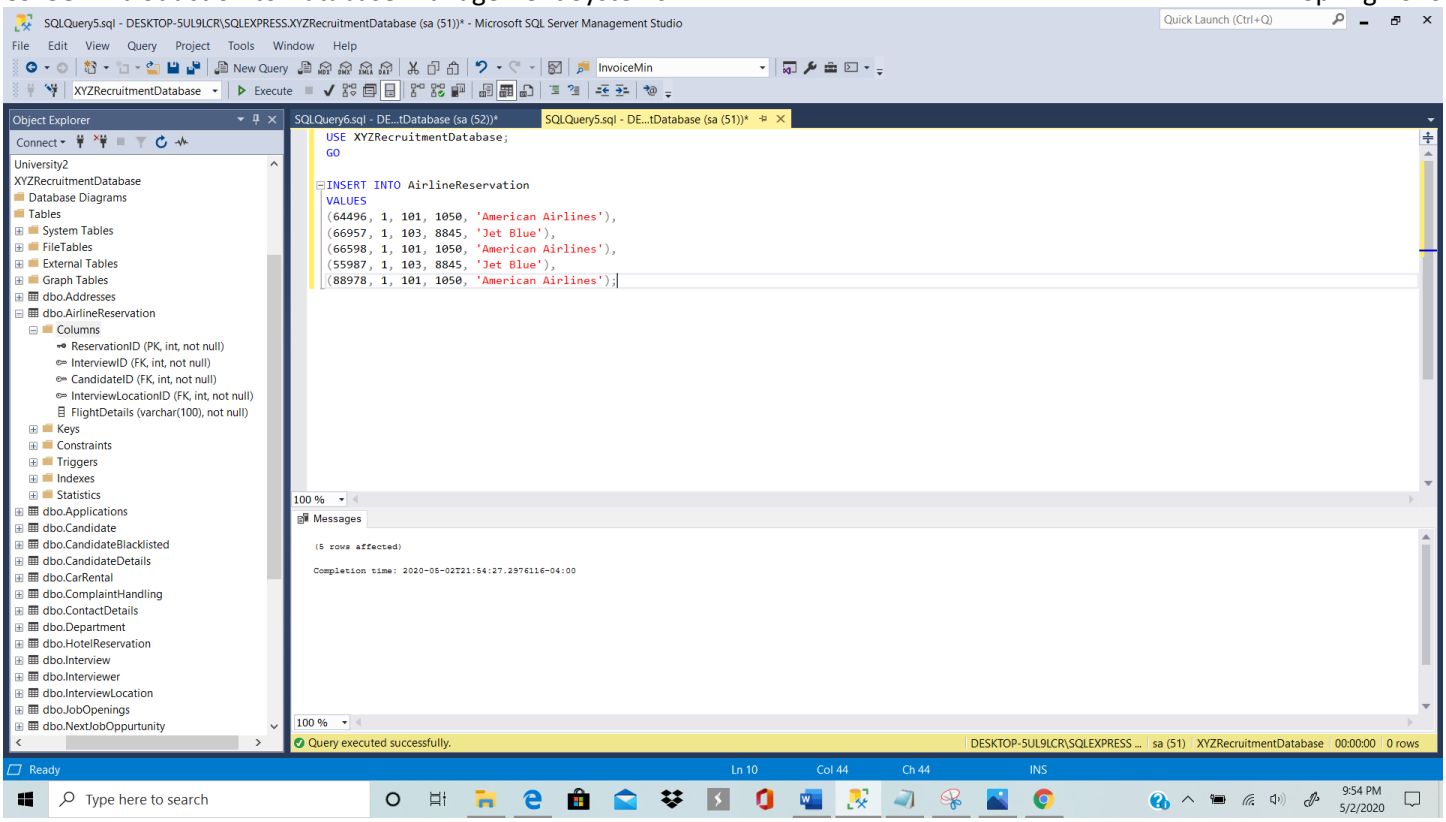

SQLQuery6.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
InvoiceMin
XYZRecruitmentDatabase
Execute
Object Explorer
Connect
University2
XYZRecruitmentDatabase
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Addresses
dbo.AirlineReservation
dbo.Applications
dbo.Candidate
dbo.CandidateBlacklisted
dbo.CandidateDetails
dbo.CarRental
dbo.ComplaintHandling
dbo.ContactDetails
dbo.Department
dbo.HotelReservation
dbo.Interview
dbo.Interviewer
dbo.InterviewLocation
dbo.JobOpenings
dbo.NextJobOppurtunity
Columns
CandidateID (FK, int, not null)
CandidateDescription (varchar(100), no
Keys
Constraints
Triggers
Indexes
Statistics
dbo.Onboarding
dbo.Reimbursement
Views
SQLQuery6.sql - DE...tDatabase (sa (52))*
SQLQuery5.sql - DE...tDatabase (sa (51))*
USE XYZRecruitmentDatabase;
GO
SELECT * FROM NextJobOppurtunity;
100 %
Results
Messages
CandidateID CandidateDescription
1 104 Candidate is technically sound
2 105 Candidate has good presentation skills
Query executed successfully.
DESKTOP-5UL9LCR\SQLEXPRESS ... sa (52) XYZRecruitmentDatabase 00:00:00 2 rows
Ready
Ln 4 Col 34 Ch 34 INS
Type here to search
9:34 PM
5/2/2020

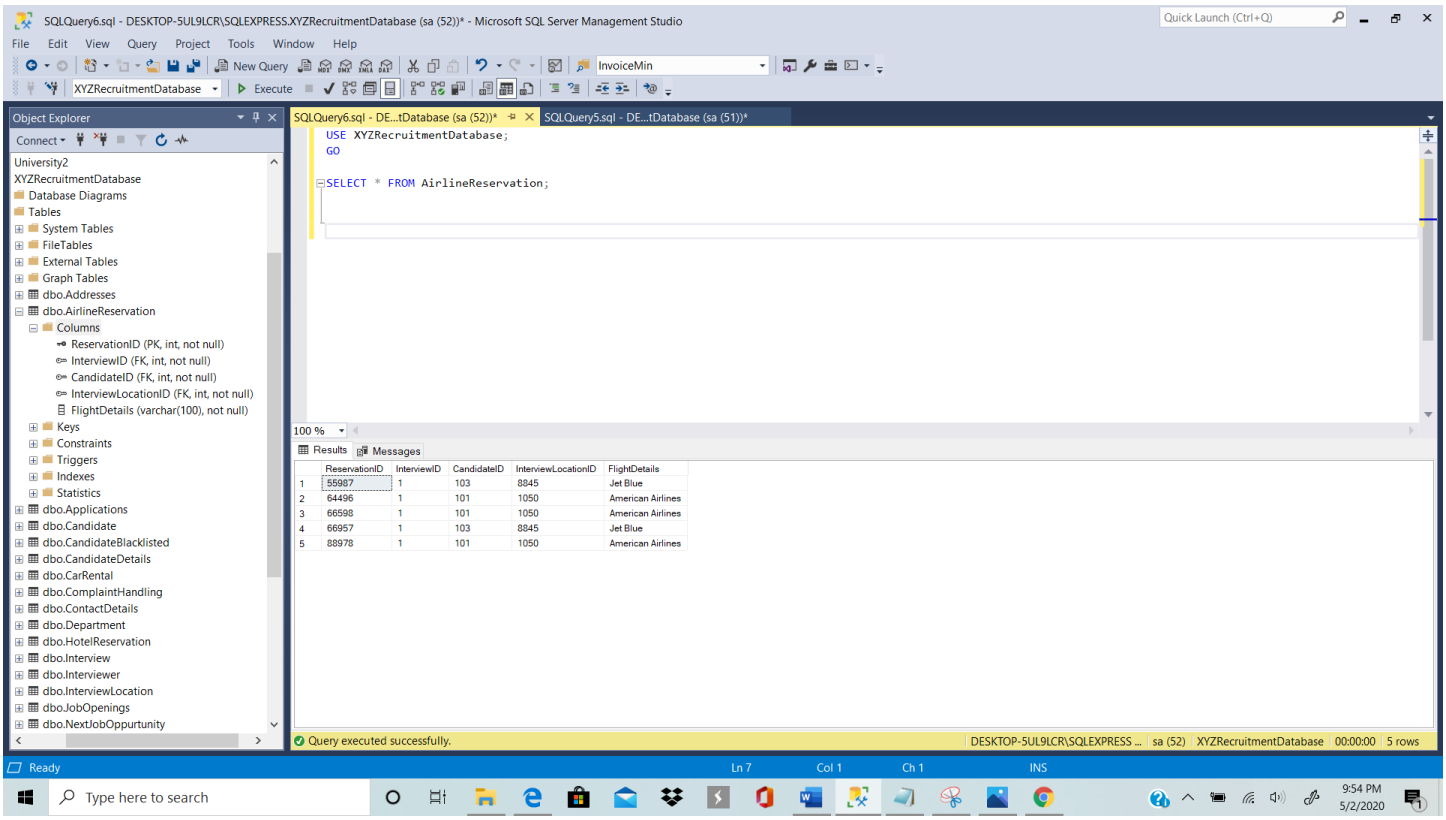

SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
InvoiceMin
XYZRecruitmentDatabase
Execute
Object Explorer
Connect
University2
XYZRecruitmentDatabase
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Addresses
dbo.AirlineReservation
dbo.Applications
dbo.Candidate
dbo.CandidateBlacklisted
dbo.CandidateDetails
dbo.CarRental
dbo.ComplaintHandling
Columns
ComplaintID (PK, int, not null)
ComplaintDetails (varchar(100), null)
CandidateID (int, not null)
InterviewerID (int, not null)
Keys
Constraints
Triggers
Indexes
Statistics
dbo.ContactDetails
dbo.Department
dbo.HotelReservation
dbo.Interview
dbo.Interviewer
dbo.InterviewLocation
dbo.JobOpenings
dbo.NextJobOppurtunity
dbo.Onboarding
SQLQuery6.sql - DE...tDatabase (sa (52))*
SQLQuery5.sql - DE...tDatabase (sa (51))*
USE XYZRecruitmentDatabase;
GO
INSERT INTO ComplaintHandling
VALUES
(1, 'Interview date was preponed and was not informed to the candidate', 103, 10001),
(2, 'The interview location was not informed properly', 105, 10004),
(3, 'The interviewer was biased', 104, 10002),
(4, 'The technical questions were different from the ones mentioned in the requirement', 104, 10002),
(5, 'The accomadation was not good', 101, 10001);
100 %
Messages
(5 rows affected)
Completion time: 2020-05-02T21:38:47.3500644-04:00
100 %
Query executed successfully.
DESKTOP-5UL9LCR\SQLEXPRESS ... sa (51) XYZRecruitmentDatabase 00:00:00 0 rows
Ready
Ln 10 Col 50 Ch 50 INS
Type here to search
9:39 PM
5/2/2020

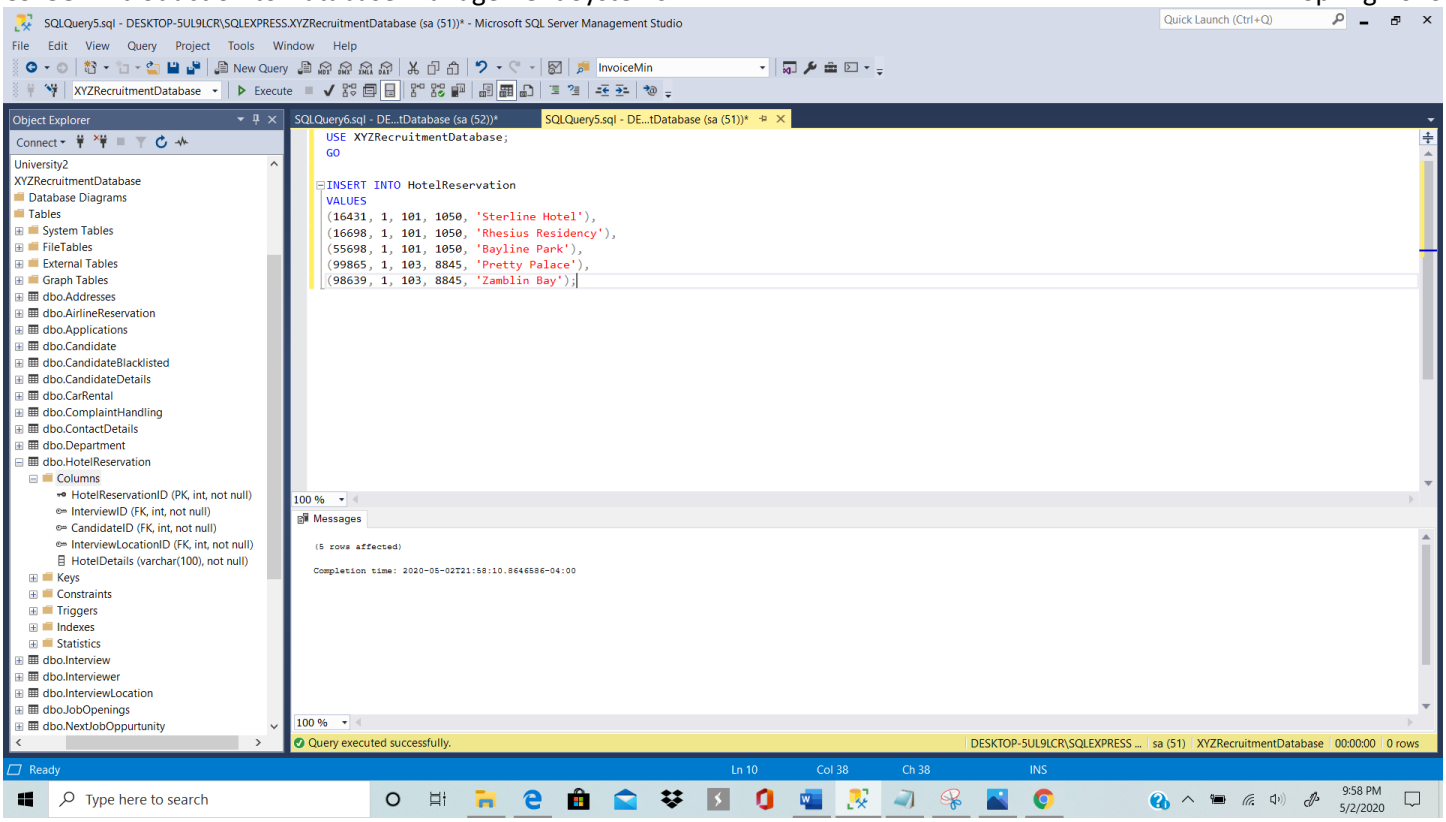

SQLQuery6.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
InvoiceMin
XYZRecruitmentDatabase
Execute
Object Explorer
Connect
University2
XYZRecruitmentDatabase
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Addresses
dbo.AirlineReservation
dbo.Applications
dbo.Candidate
dbo.CandidateBlacklisted
dbo.CandidateDetails
dbo.CarRental
dbo.ComplaintHandling
Columns
ComplaintID (PK, int, not null)
ComplaintDetails (varchar(100), null)
CandidateID (int, not null)
InterviewerID (int, not null)
Keys
Constraints
Triggers
Indexes
Statistics
dbo.ContactDetails
dbo.Department
dbo.HotelReservation
dbo.Interview
dbo.Interviewer
dbo.InterviewLocation
dbo.JobOpenings
dbo.NextJobOppurtunity
dbo.Onboarding
SQLQuery6.sql - DE...tDatabase (sa (52))*
SQLQuery5.sql - DE...tDatabase (sa (51))*
USE XYZRecruitmentDatabase;
GO
SELECT * FROM ComplaintHandling;
100 %
Results
Messages
ComplaintID ComplaintDetails CandidateID InterviewerID
1 1 Interview date was preponed and was not informe... 103 10001
2 2 The interview location was not informed properly 105 10004
3 3 The interviewer was biased 104 10002
4 4 The technical questions were different from the on... 104 10002
5 5 The accomadation was not good 101 10001
Query executed successfully.
DESKTOP-5UL9LCR\SQLEXPRESS ... sa (52) XYZRecruitmentDatabase 00:00:00 5 rows
Ready
Ln 5 Col 1 Ch 1 INS
Type here to search
9:40 PM
5/2/2020

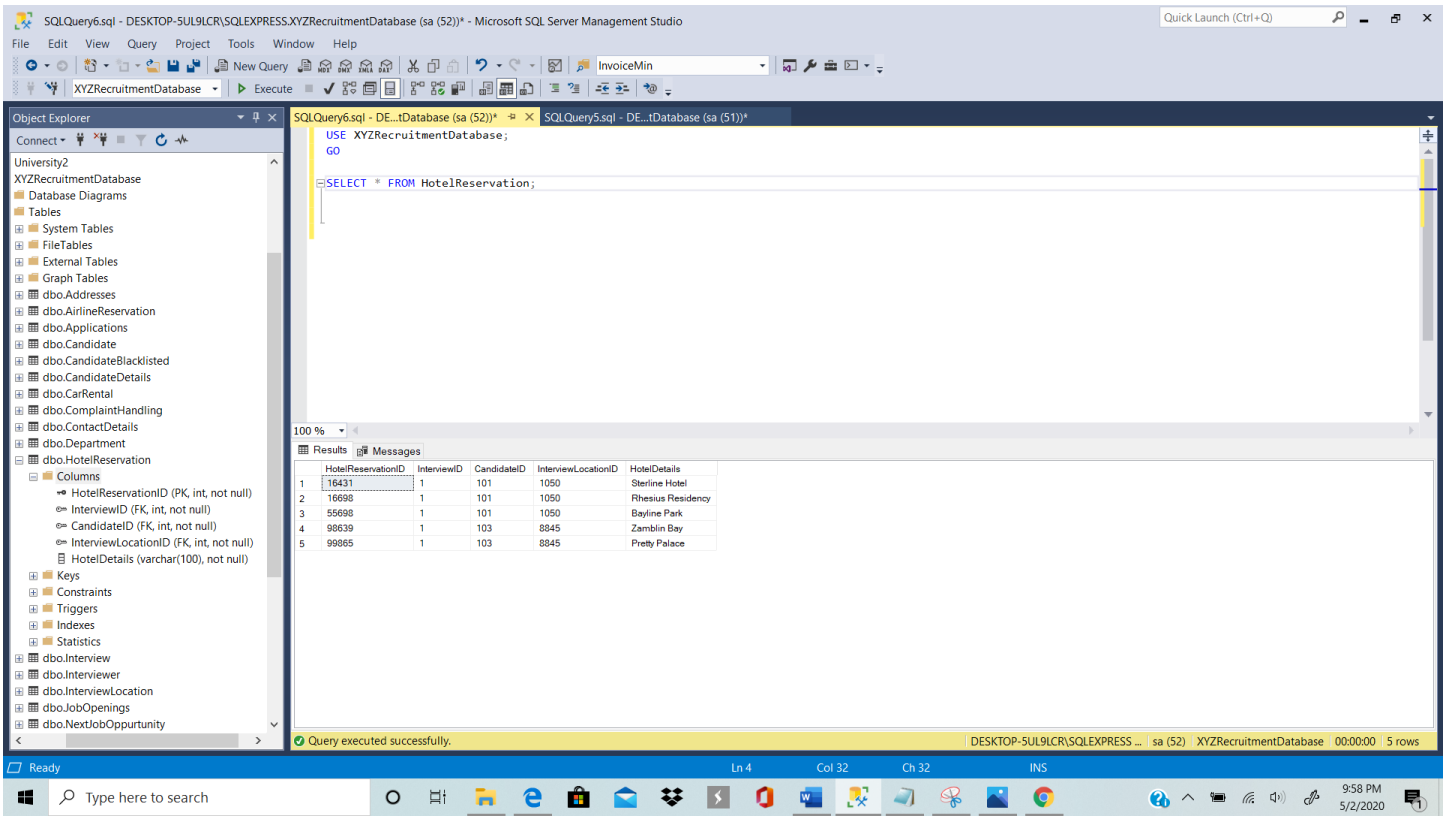

SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
INSERT INTO CarRental
VALUES
(8796, 1, 101, 'Red Toyota'),
(8897, 1, 103, 'Blue Ace');
(2 row affected)
Completion time: 2020-05-02T21:42:20.4913983-04:00
Query executed successfully.

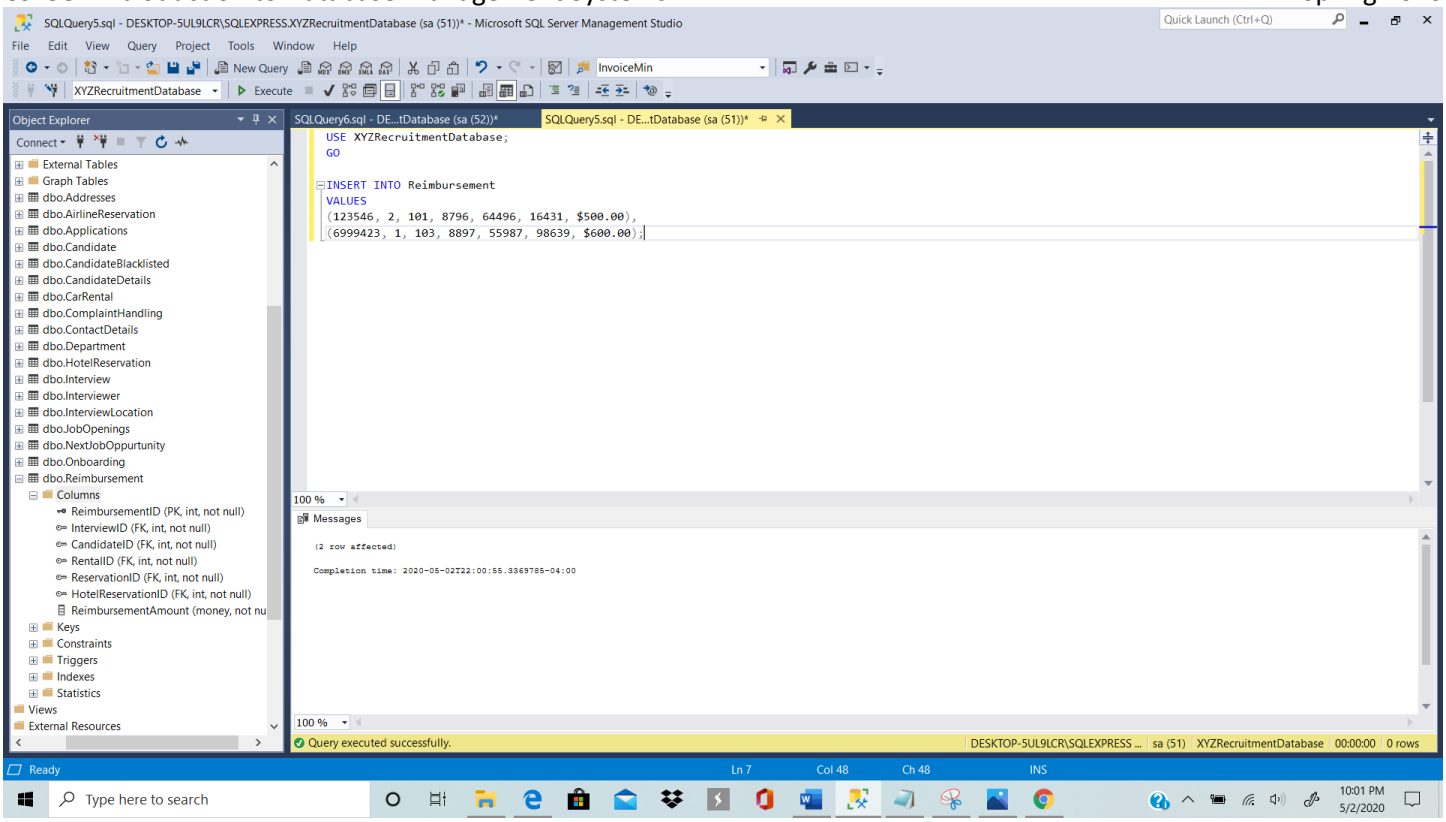

SQLQuery6.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
SELECT * FROM CarRental;
RentalID InterviewID CandidateID CarDetails
1 8796 1 101 Red Toyota
2 8897 1 103 Blue Ace
Query executed successfully.

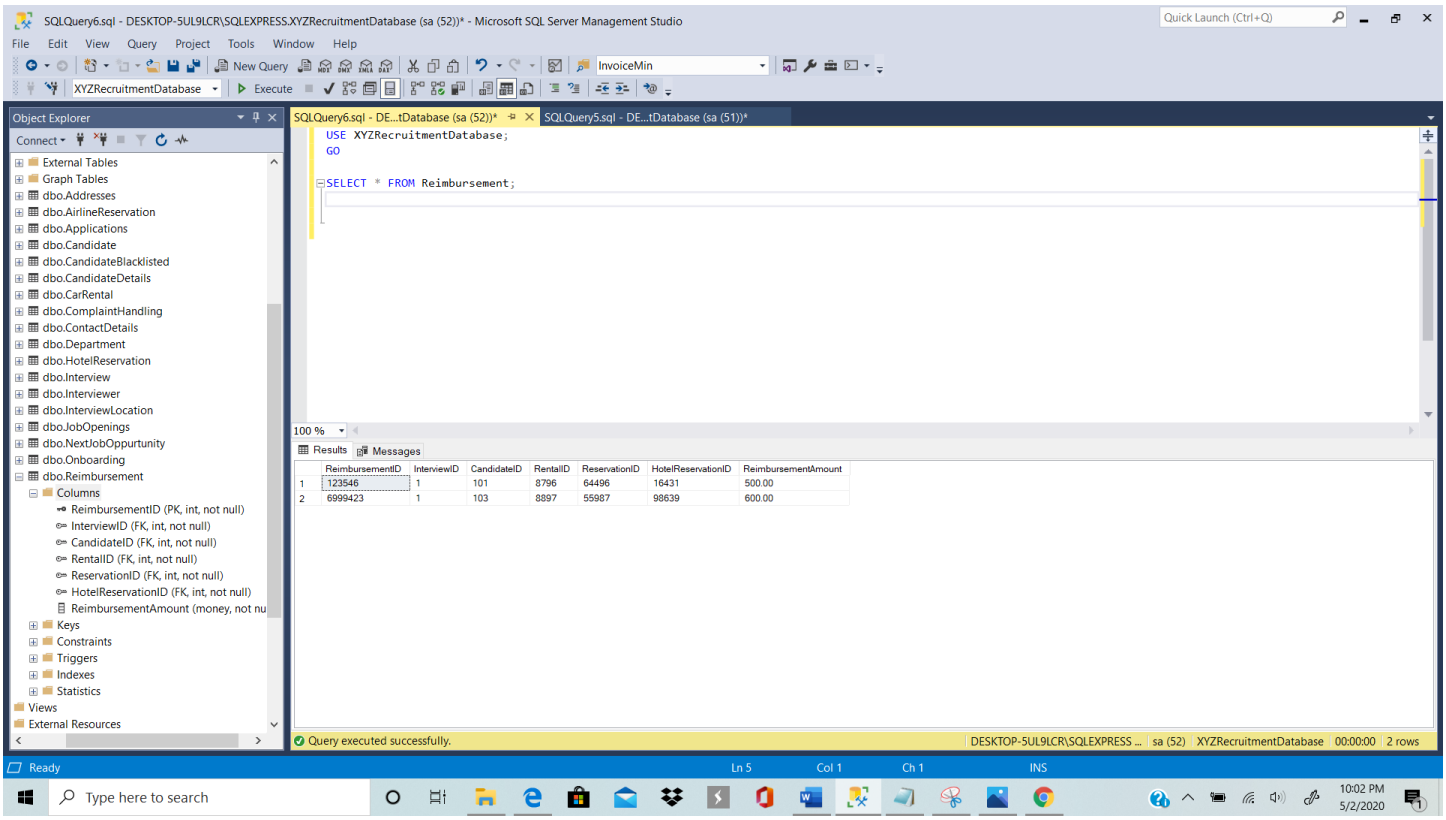

SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
INSERT INTO AirlineReservation
VALUES
(64496, 1, 101, 1050, 'American Airlines'),
(66957, 1, 103, 8845, 'Jet Blue'),
(66598, 1, 101, 1050, 'American Airlines'),
(55987, 1, 103, 8845, 'Jet Blue'),
(88978, 1, 101, 1050, 'American Airlines');
(5 rows affected)
Completion time: 2020-05-02T21:54:27.2976116-04:00
Query executed successfully.

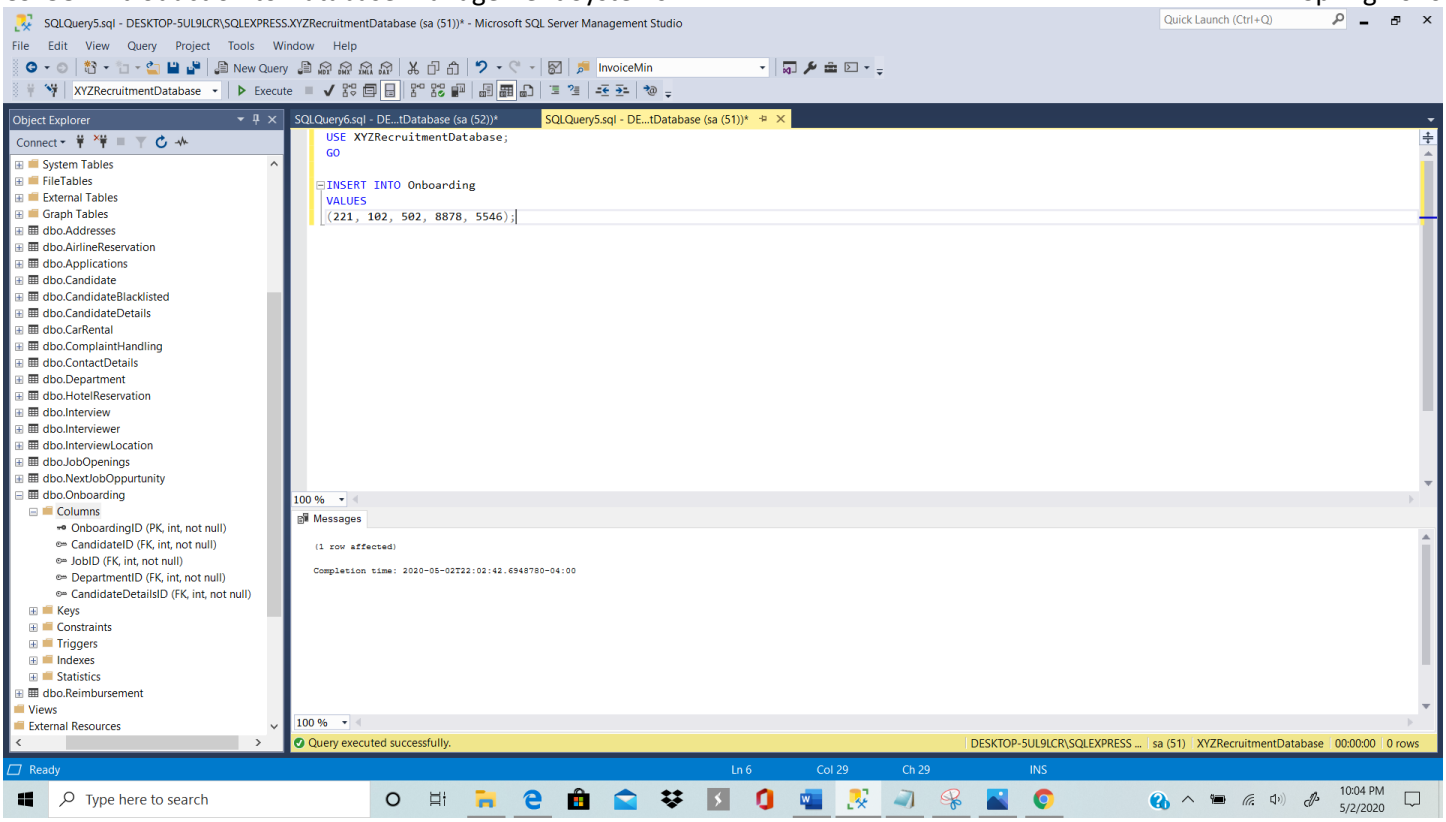

SQLQuery6.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
SELECT * FROM AirlineReservation;
ReservationID InterviewID CandidateID InterviewLocationID FlightDetails
55987 1 103 8845 Jet Blue
64496 1 101 1050 American Airlines
66598 1 101 1050 American Airlines
66957 1 103 8845 Jet Blue
88978 1 101 1050 American Airlines
Query executed successfully.

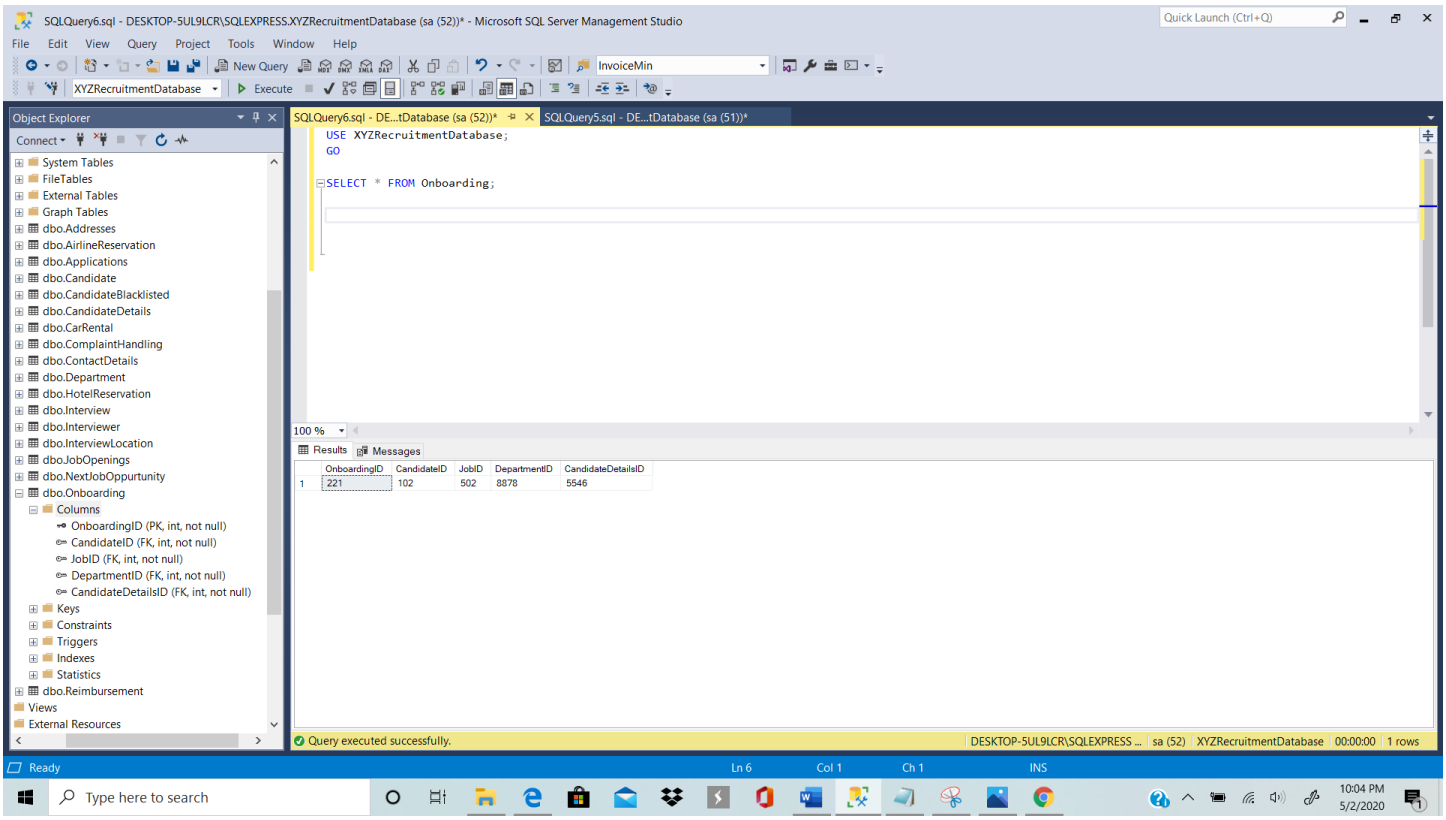

SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
INSERT INTO HotelReservation
VALUES
(16431, 1, 101, 1050, 'Sterline Hotel'),
(16698, 1, 101, 1050, 'Rhesius Residency'),
(55698, 1, 101, 1050, 'Bayline Park'),
(99865, 1, 103, 8845, 'Pretty Palace'),
(98639, 1, 103, 8845, 'Zamblin Bay');
(5 rows affected)
Completion time: 2020-05-02T21:58:10.8646586-04:00
Query executed successfully.

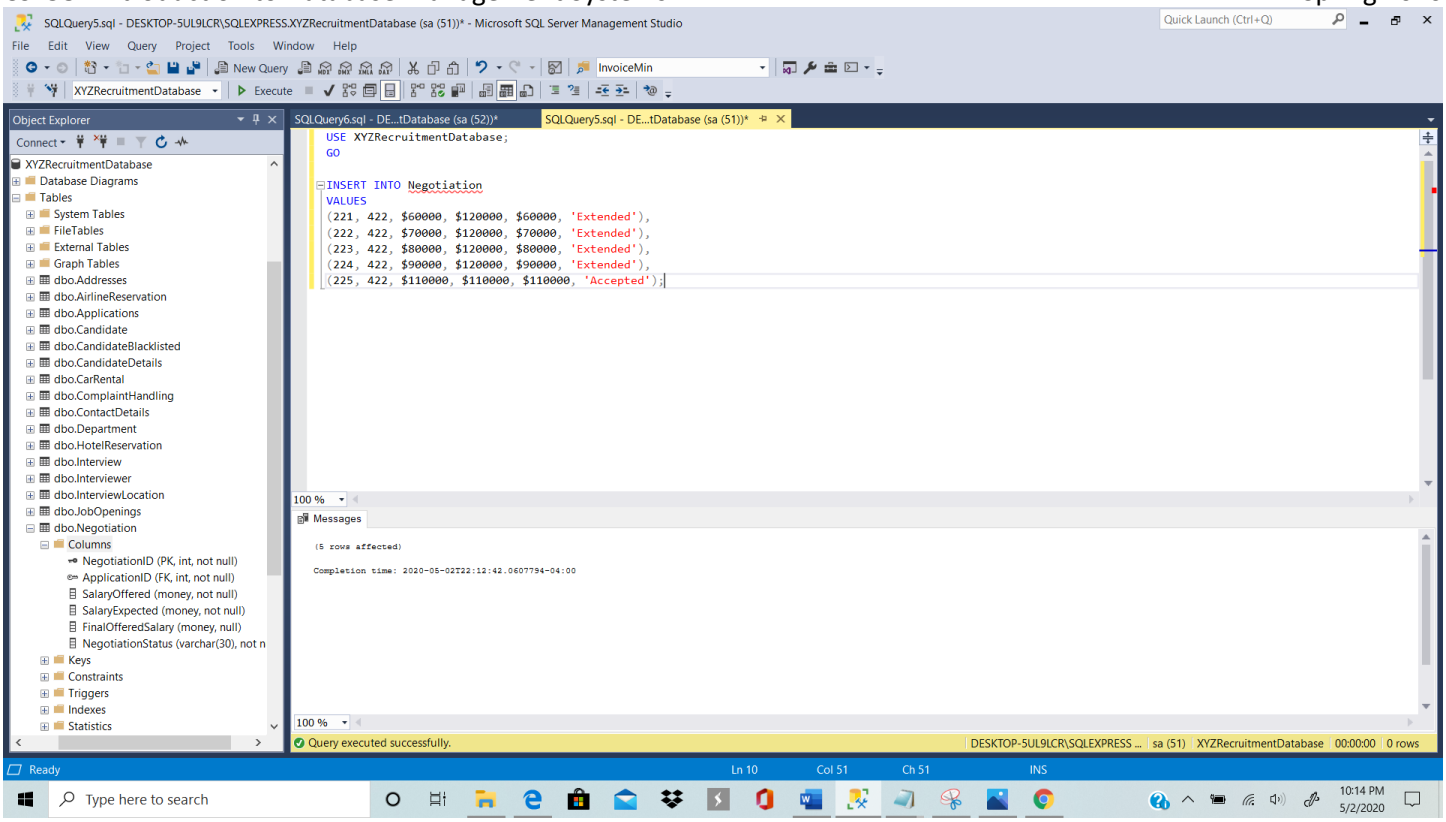

SQLQuery6.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
SELECT * FROM HotelReservation;
HotelReservationID InterviewID CandidateID InterviewLocationID HotelDetails
1 16431 1 101 1050 Sterline Hotel
2 16698 1 101 1050 Rhesius Residency
3 55698 1 101 1050 Bayline Park
4 98639 1 103 8845 Zamblin Bay
5 99865 1 103 8845 Pretty Palace
Query executed successfully.

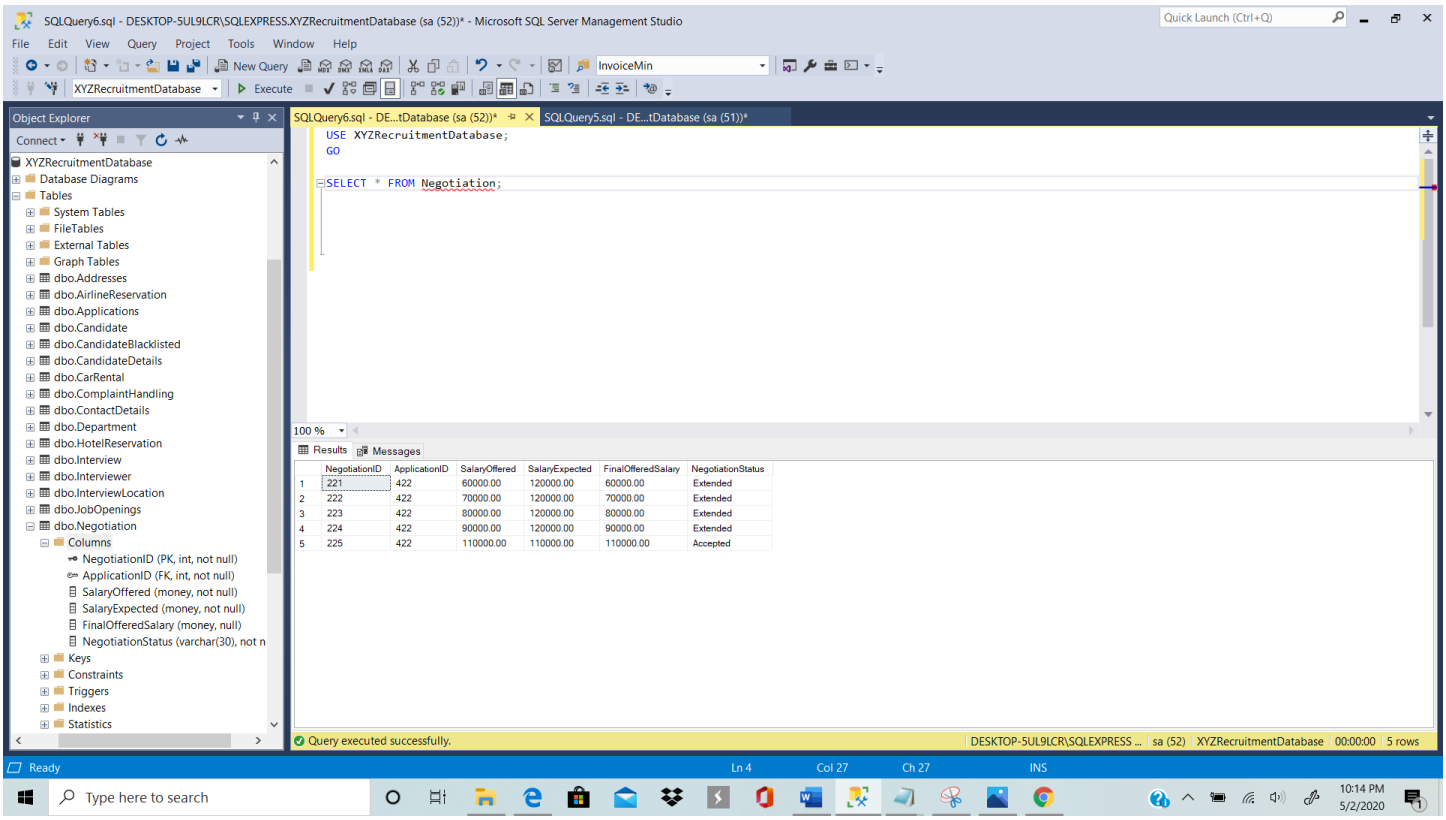

SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
INSERT INTO Reimbursement
VALUES
(123546, 2, 101, 8796, 64496, 16431, $500.00),
(6999423, 1, 103, 8897, 55987, 98639, $600.00);
(2 row affected)
Completion time: 2020-05-02T22:00:55.3369785-04:00
Query executed successfully.


SQLQuery6.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
SELECT * FROM Reimbursement;
ReimbursementID InterviewID CandidateID RentalID ReservationID HotelReservationID ReimbursementAmount
1 123546 1 101 8796 64496 16431 500.00
2 6999423 1 103 8897 55987 98639 600.00
Query executed successfully.


SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
INSERT INTO Onboarding
VALUES
(221, 102, 502, 8878, 5546);
(1 row affected)
Completion time: 2020-05-02T22:02:42.6948780-04:00
Query executed successfully.


SQLQuery6.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
SELECT * FROM Onboarding;
OnboardingID CandidateID JobID DepartmentID CandidateDetailsID
1 221 102 502 8878 5546
Query executed successfully.


SQLQuery5.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (51))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
INSERT INTO Negotiation
VALUES
(221, 422, $60000, $120000, $60000, 'Extended'),
(222, 422, $70000, $120000, $70000, 'Extended'),
(223, 422, $80000, $120000, $80000, 'Extended'),
(224, 422, $90000, $120000, $90000, 'Extended'),
(225, 422, $110000, $110000, $110000, 'Accepted');
(5 rows affected)
Completion time: 2020-05-02T22:12:42.0607794-04:00
Query executed successfully.


SQLQuery6.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
USE XYZRecruitmentDatabase;
GO
SELECT * FROM Negotiation;
NegotiationID ApplicationID SalaryOffered SalaryExpected FinalOfferedSalary NegotiationStatus
1 221 422 60000.00 120000.00 60000.00 Extended
2 222 422 70000.00 120000.00 70000.00 Extended
3 223 422 80000.00 120000.00 80000.00 Extended
4 224 422 90000.00 120000.00 90000.00 Extended
5 225 422 110000.00 110000.00 110000.00 Accepted
Query executed successfully.

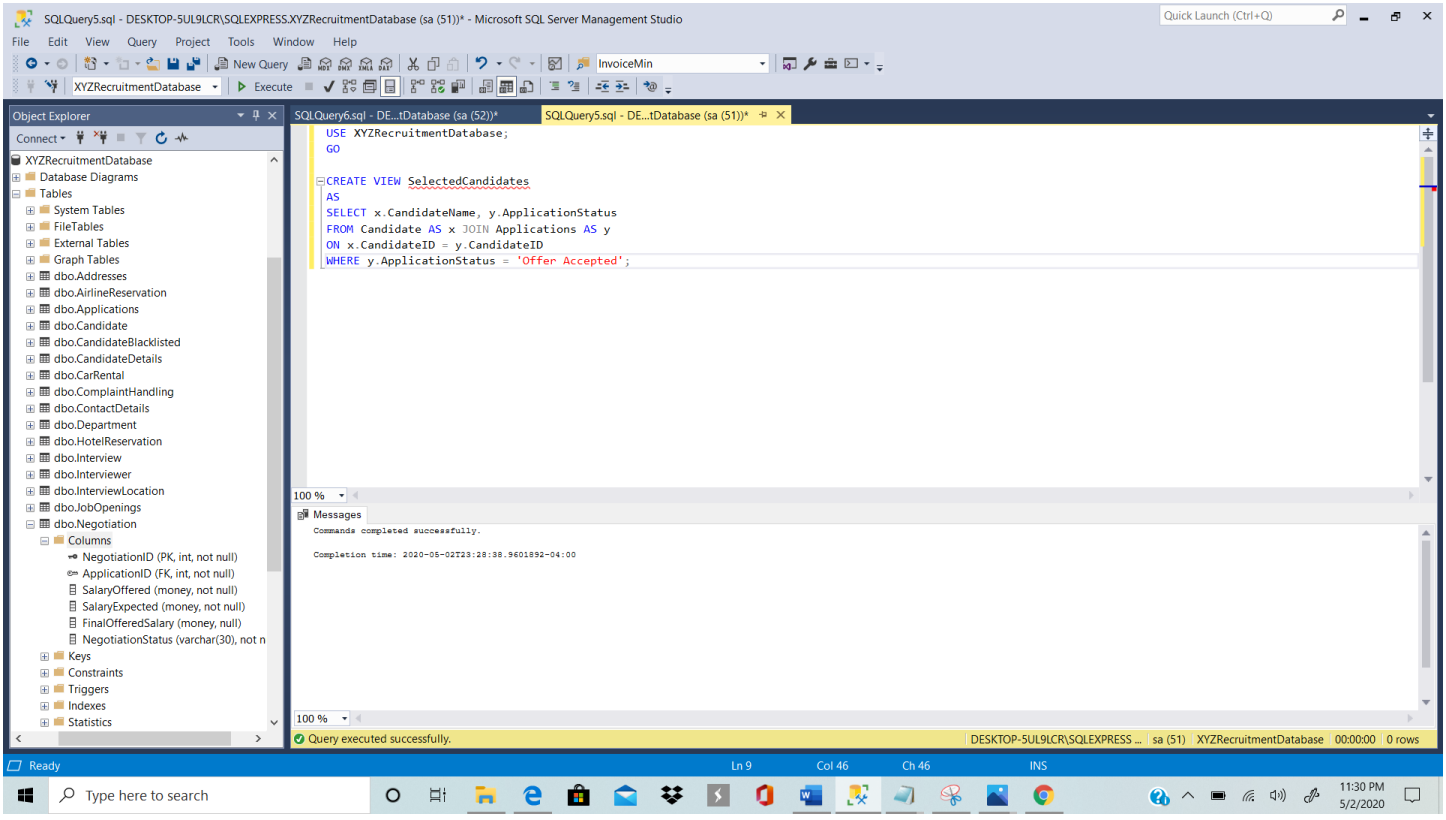
b. Views

In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. Here is a list of views that I have created for this project.

View-1:

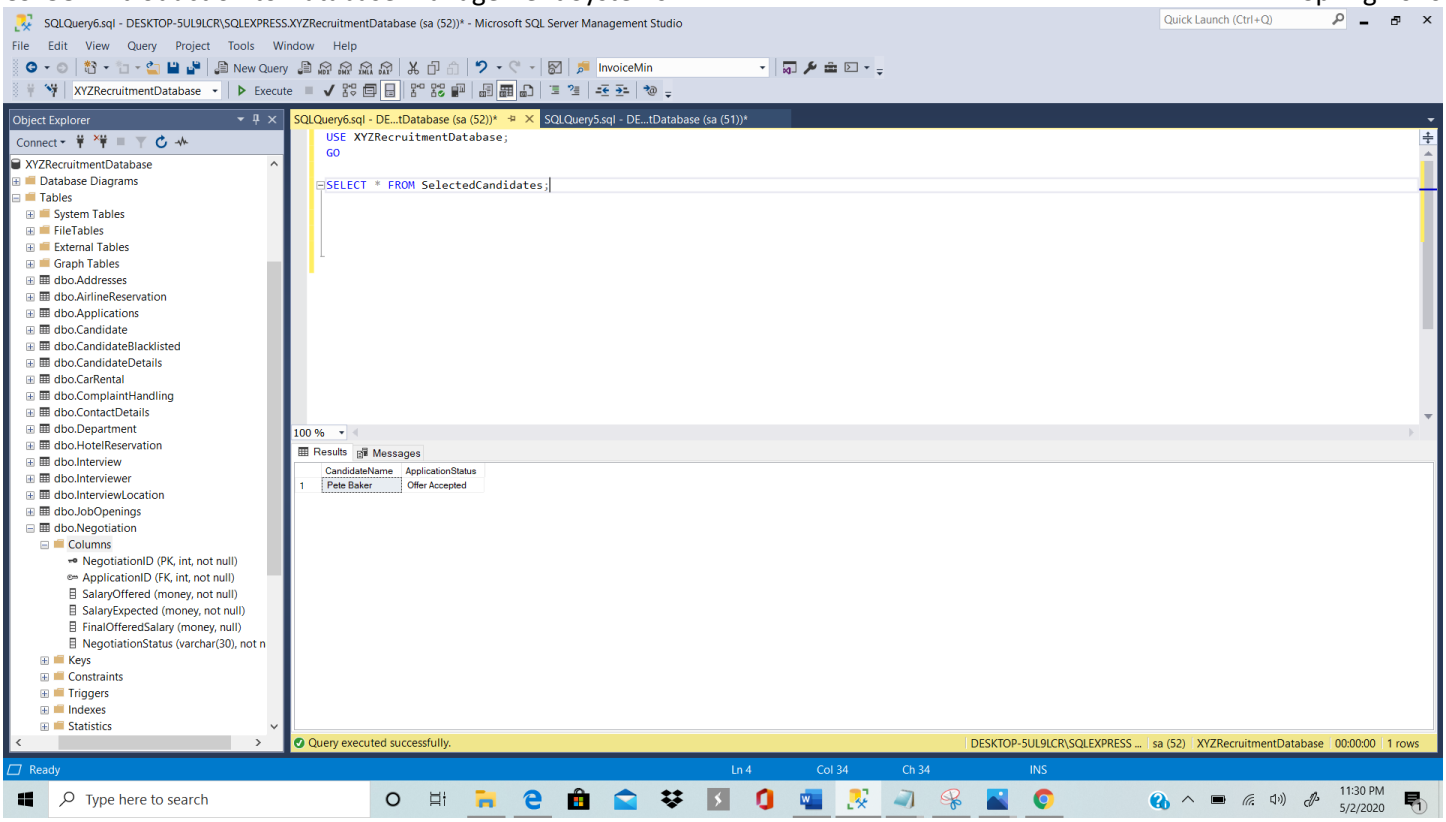
```
CREATE VIEW SelectedCandidates
AS
SELECT x.CandidateName, y.ApplicationStatus
FROM Candidate AS x JOIN Applications AS y
ON x.CandidateID = y.CandidateID
WHERE y.ApplicationStatus = 'Offer Accepted';
```

The following script creates a view named ‘SelectedCandidates’. This view returns the names of all the candidates who have been selected after the interview process and have accepted the offer.



SELECT * FROM SelectedCandidates;

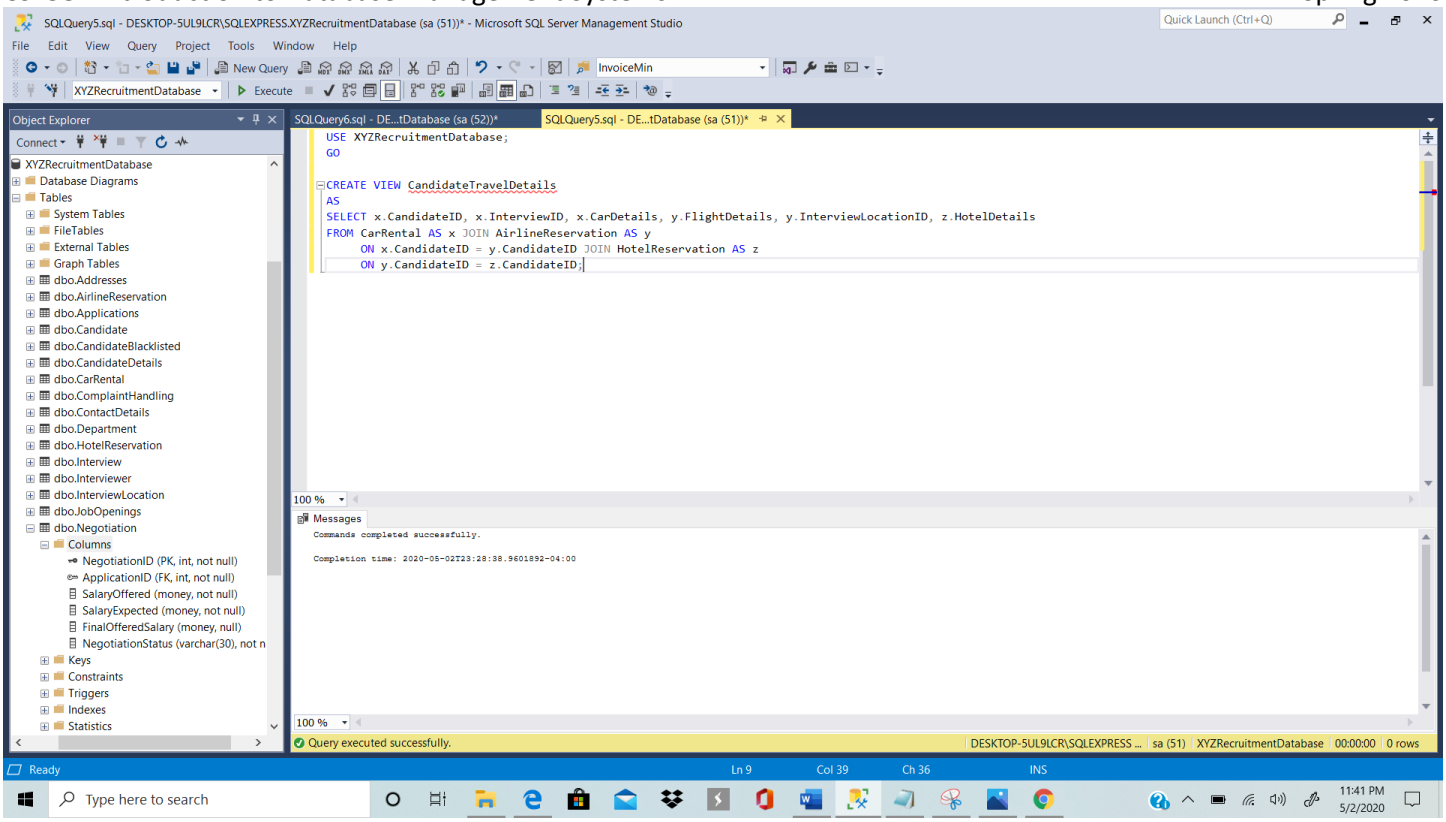
The following query returns all the entries in the view named ‘SelectedCandidates’.



View-2:

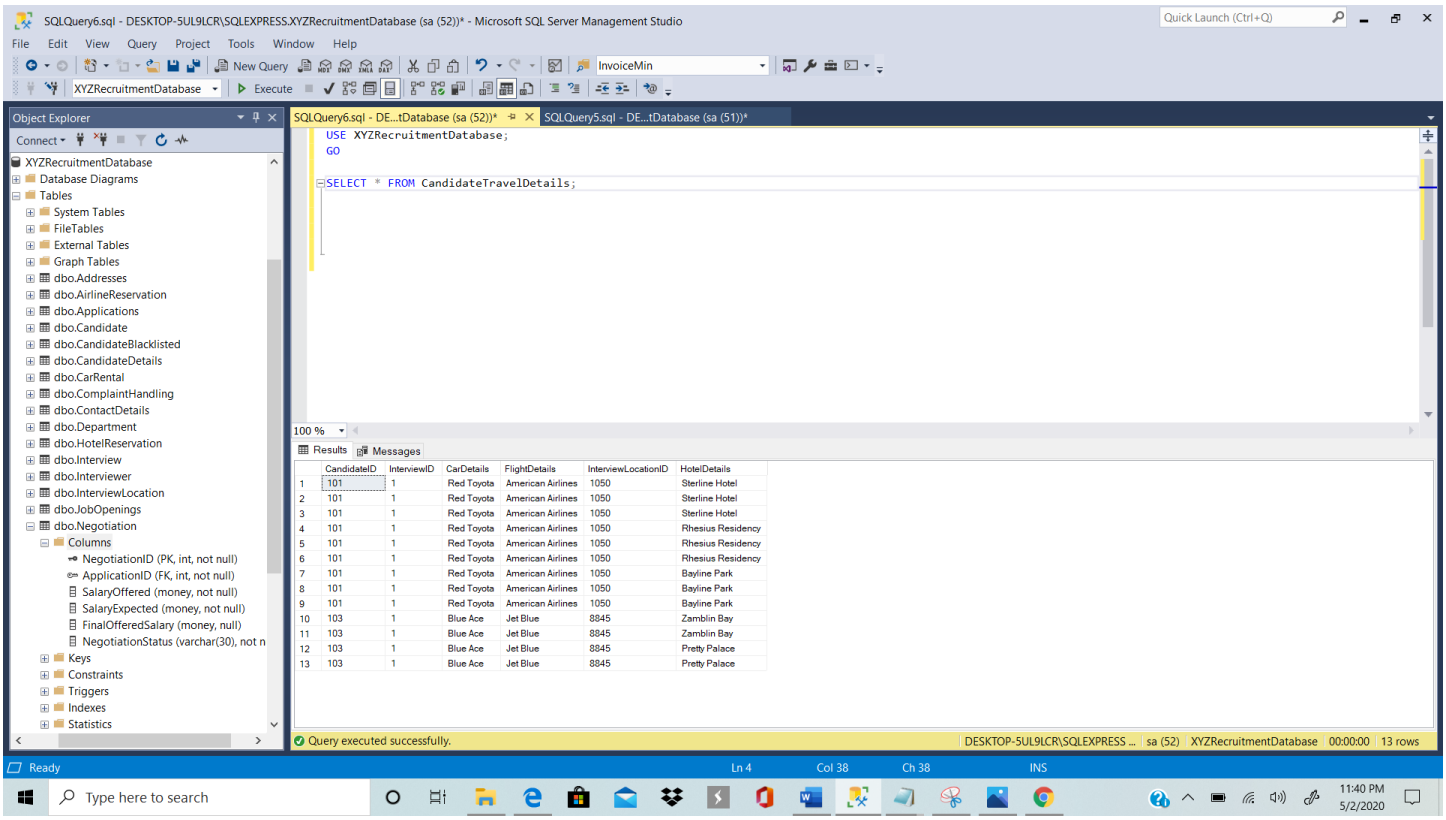
```
CREATE VIEW CandidateTravelDetails
AS
SELECT x.CandidateID, x.InterviewID, x.CarDetails, y.FlightDetails, y.InterviewLocationID,
z.HotelDetails
FROM CarRental AS x JOIN AirlineReservation AS y
        ON x.CandidateID = y.CandidateID JOIN HotelReservation AS z
        ON y.CandidateID = z.CandidateID;
```

The following script creates a view named ‘CandidateTravelDetails’. This view returns the travel details namely, CarDetails, FlightDetails, HotelDetails, InterviewLocationDetails along with the CandidateID for all the candidates who have an interview onsite.



SELECT * FROM CandidateTravelDetails;

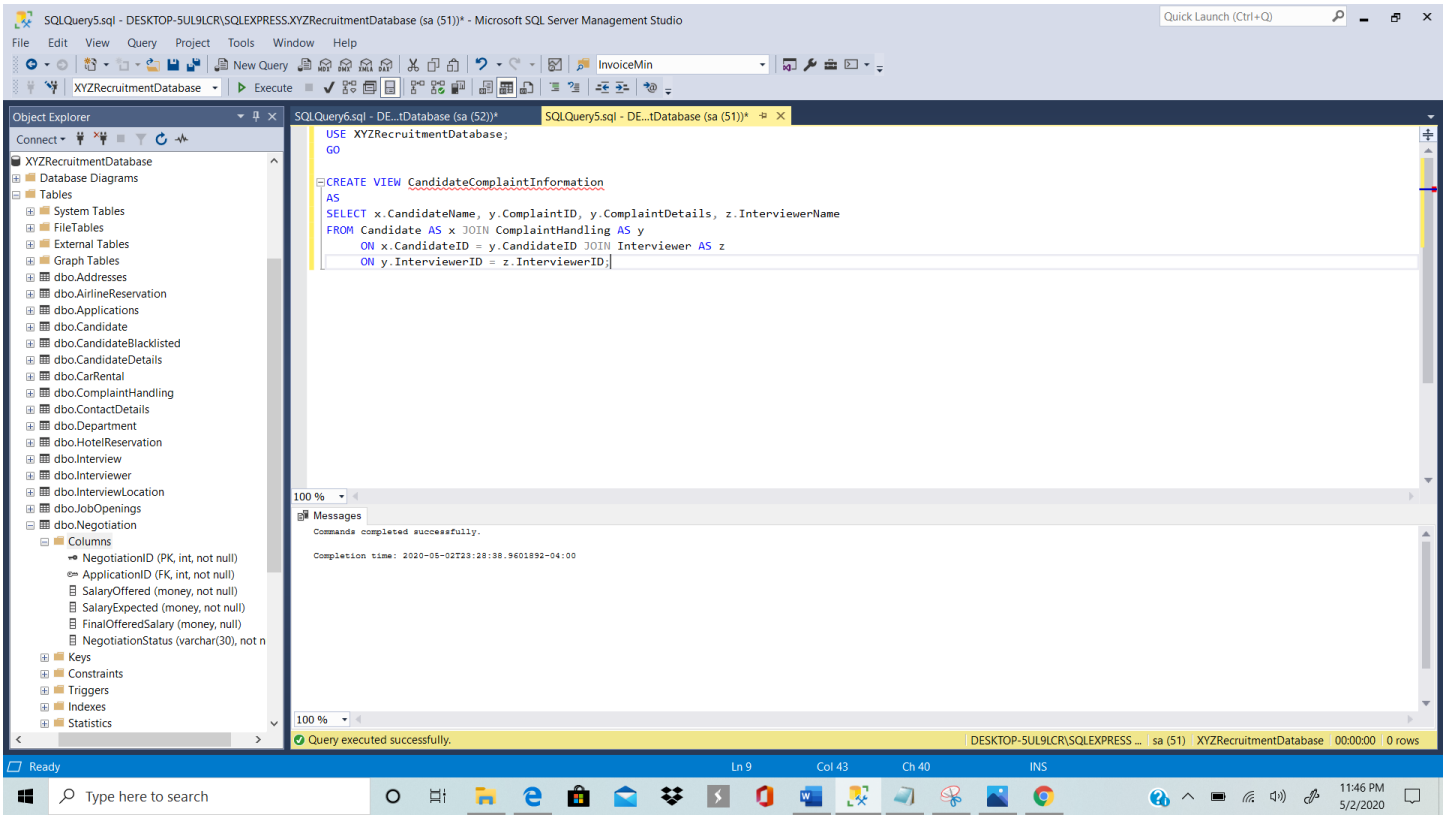
The following query returns all the entries in the CandidateTravelDetails view.



View-3:

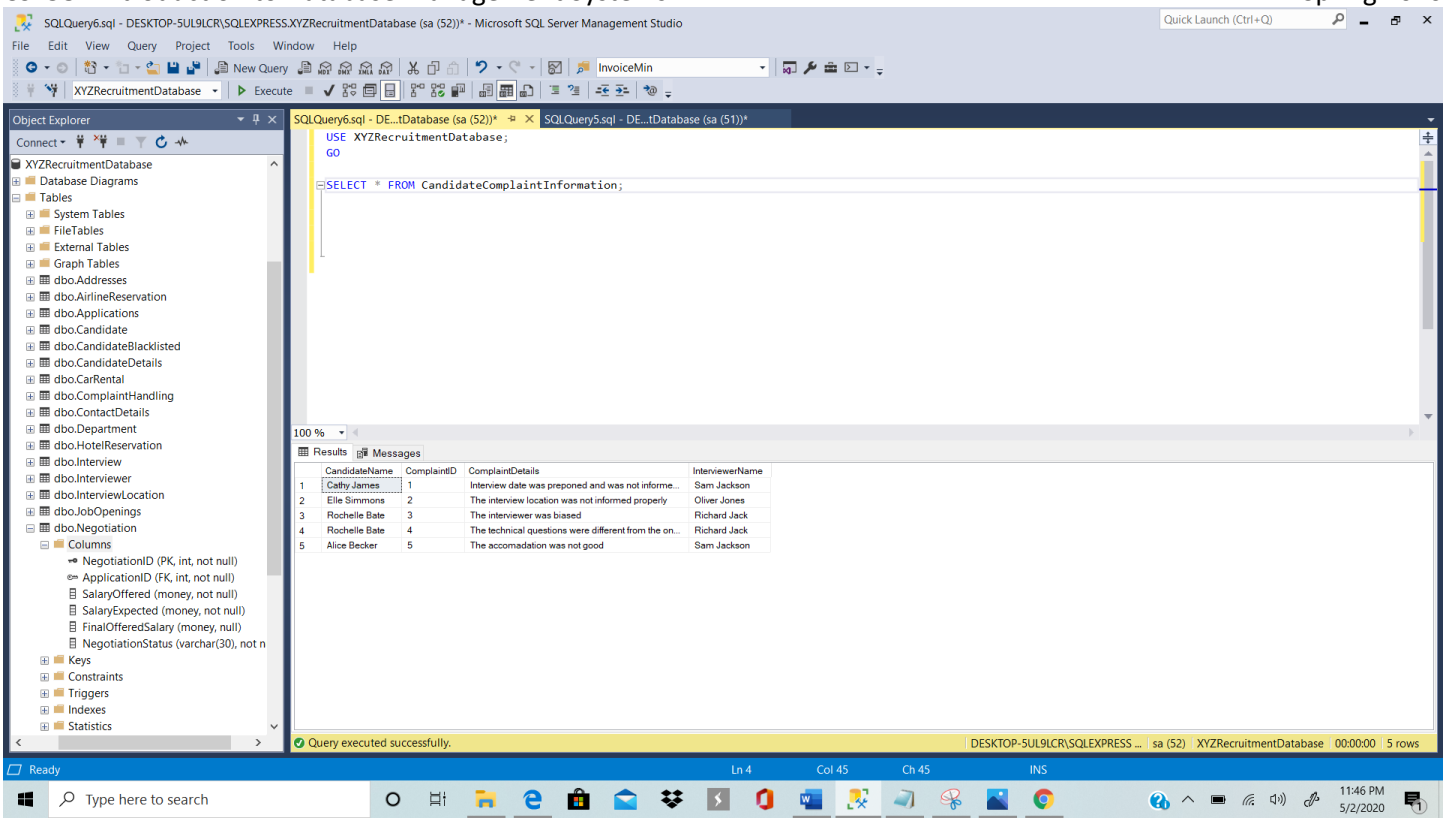
```
CREATE VIEW CandidateComplaintInformation
AS
SELECT x.CandidateName, y.ComplaintID, y.ComplaintDetails, z.InterviewerName
FROM Candidate AS x JOIN ComplaintHandling AS y
        ON x.CandidateID = y.CandidateID JOIN Interviewer AS z
        ON y.InterviewerID = z.InterviewerID;
```

The following script creates a view named 'CandidateComplaintInformation'. This view returns all the details related to any complaints that a candidate has lodged.




```
SELECT * FROM CandidateComplaintInformation;
```

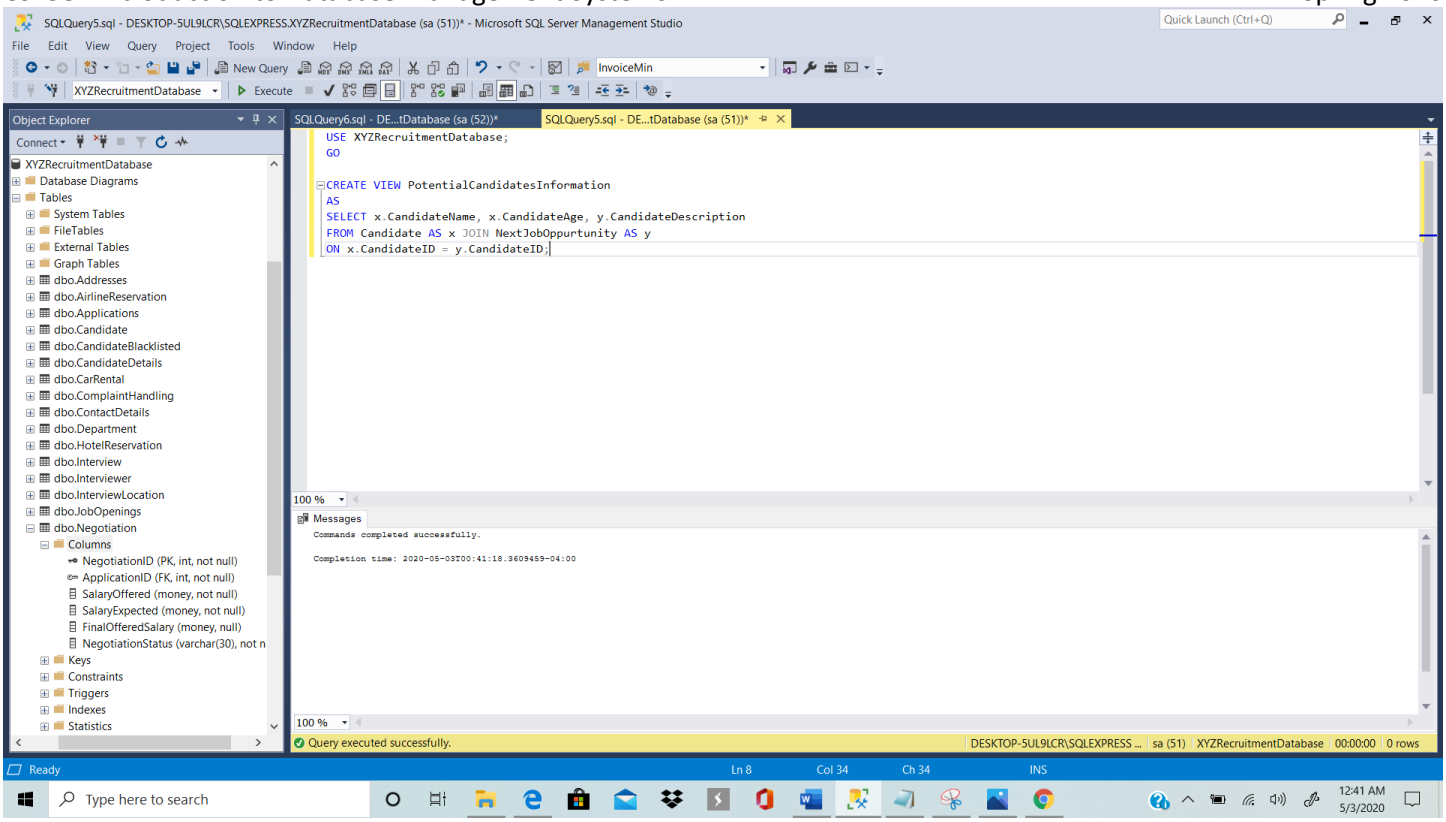
The following query returns all the entries from the view 'CandidateComplaintInformation'.



View – 4:

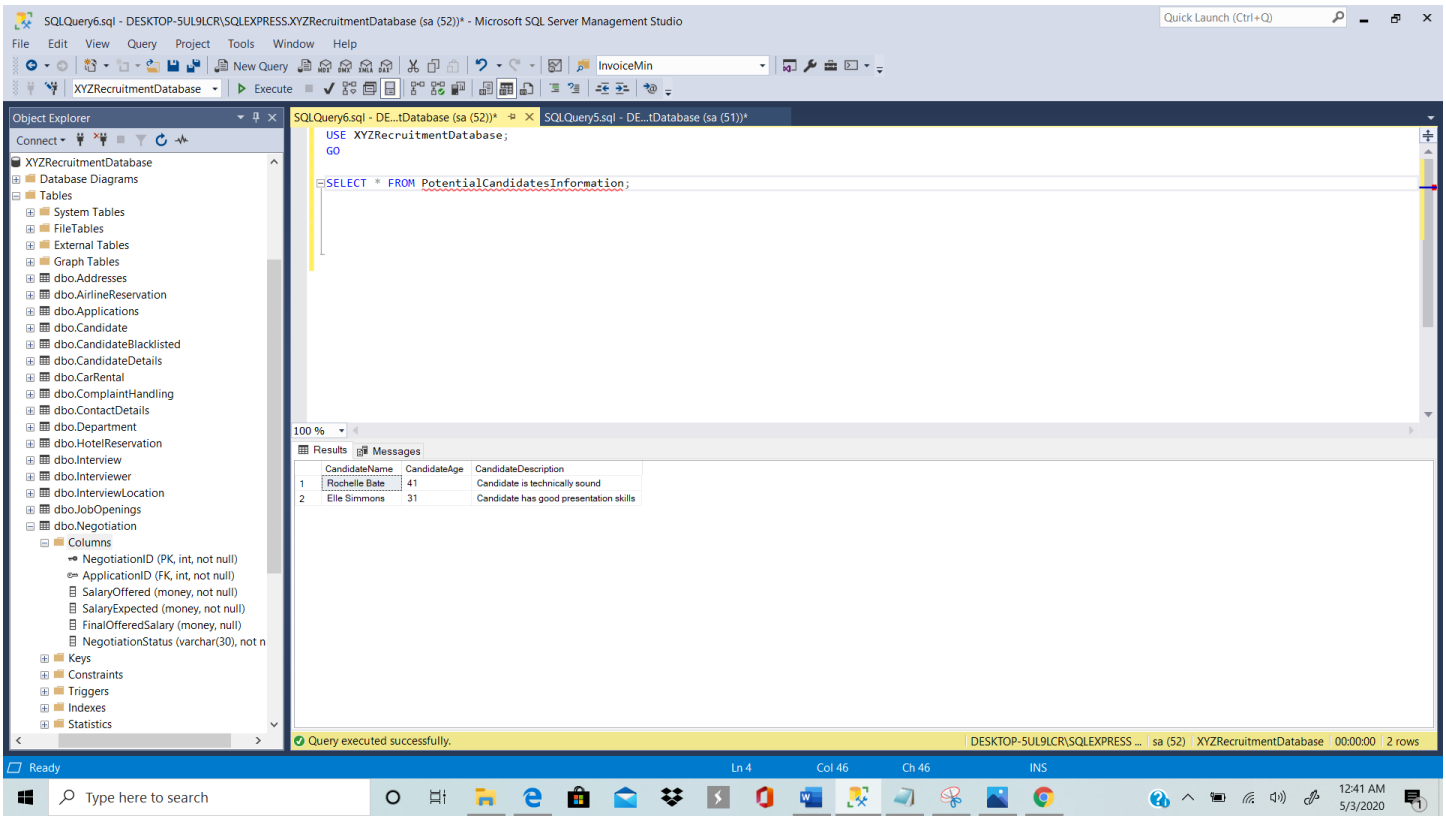
The following script creates a view named 'PotentailCandidateInformation'. This view returns the CandidateName, CandidateAge and CandidateDescription of the candidates who were rejected in the interview, but are viewed by the company as potential candidates for next job opportunity in the organization. These details are returned by joining the Candidate and NextJobOppurtunity tables from the database.

```
CREATE VIEW PotentialCandidatesInformation
AS
SELECT x.CandidateName, x.CandidateAge, y.CandidateDescription
FROM Candidate AS x JOIN NextJobOppurtunity AS y
        ON x.CandidateID = y.CandidateID;
```



SELECT * FROM PotentialCandidatesInformation;

The following query retrieves all the entries from the view 'PotentialCandidatesInformation'.



c. Stored Procedures

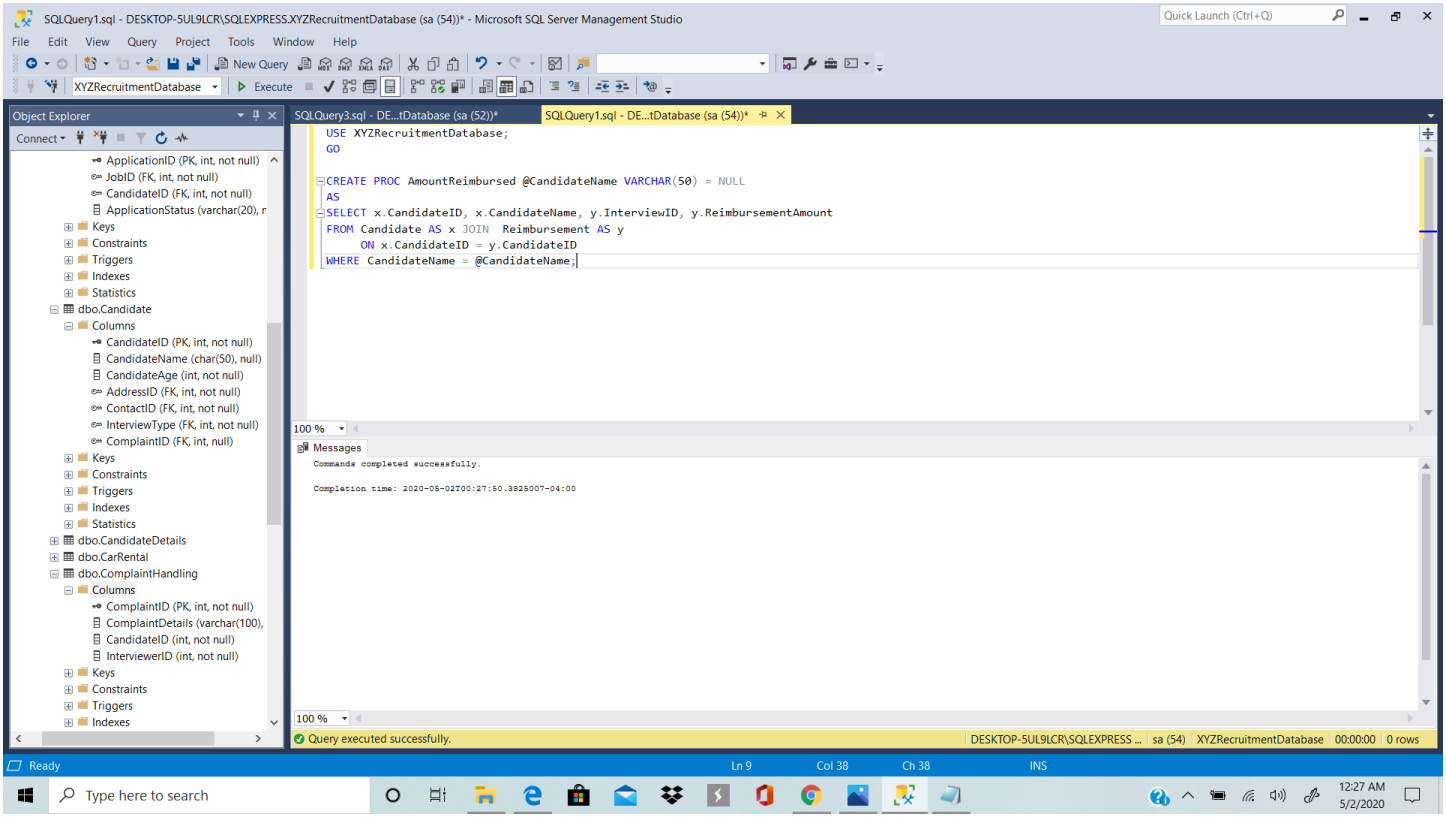
A Stored Procedure is a SQL code that the user can save, so that the code can be reused. So if there is a query that an user has to write over and over again, the user can save it as a stored procedure, and then just call it to execute it by passing related parameters.

Here is a list of stored procedures that I have created for this project.

Stored Procedure – 1:

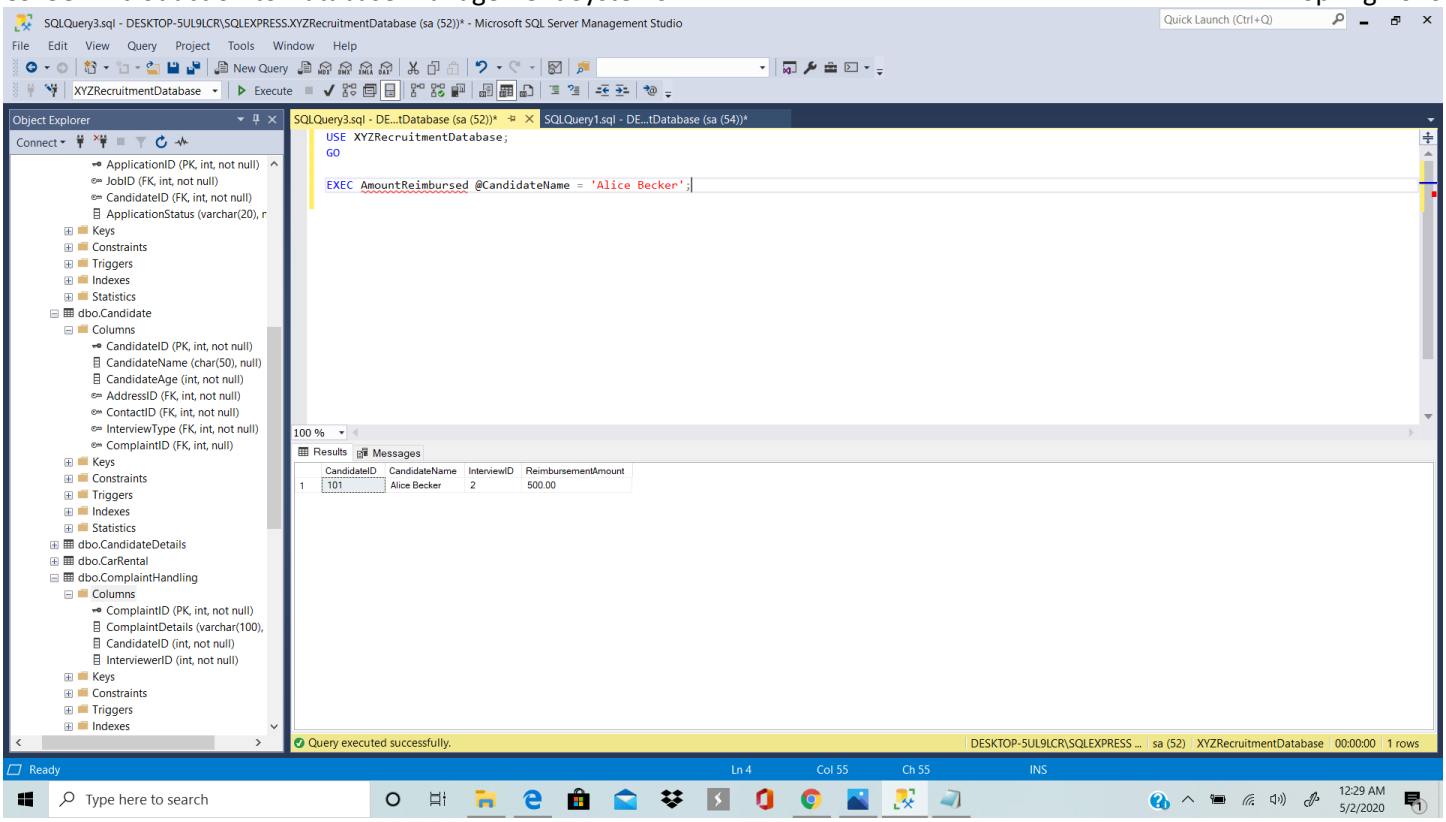
```
CREATE PROC AmountReimbursed @CandidateName VARCHAR(50) = NULL
AS
SELECT x.CandidateID, x.CandidateName, y.InterviewID, y.ReimbursementAmount
FROM Candidate AS x JOIN  Reimbursement AS y
        ON x.CandidateID = y.CandidateID
WHERE CandidateName = @CandidateName;
```

The following script creates a stored procedure named 'AmountReimbursed' that takes a variable named @CandidateName which takes a VARCHAR datatype. The @CandidateName returns the amount that has been reimbursed for the candidate from the organization. The candidate name is passed as a parameter here.



EXEC AmountReimbursed @CandidateName = 'Alice Becker';

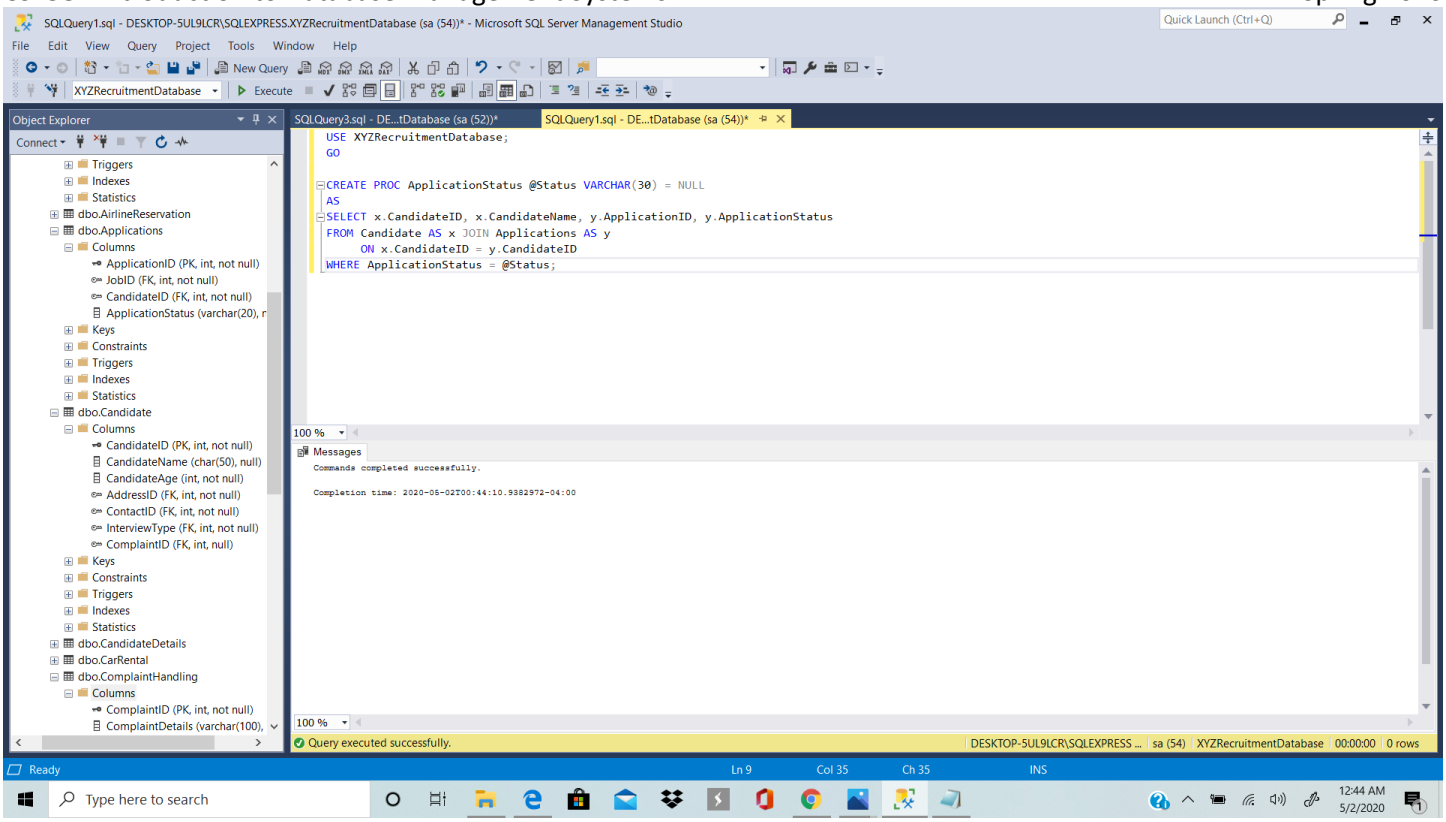
Here Alice Becker is passed as a parameter to the variable in AmountReimbursed.



Stored Procedure – 2:

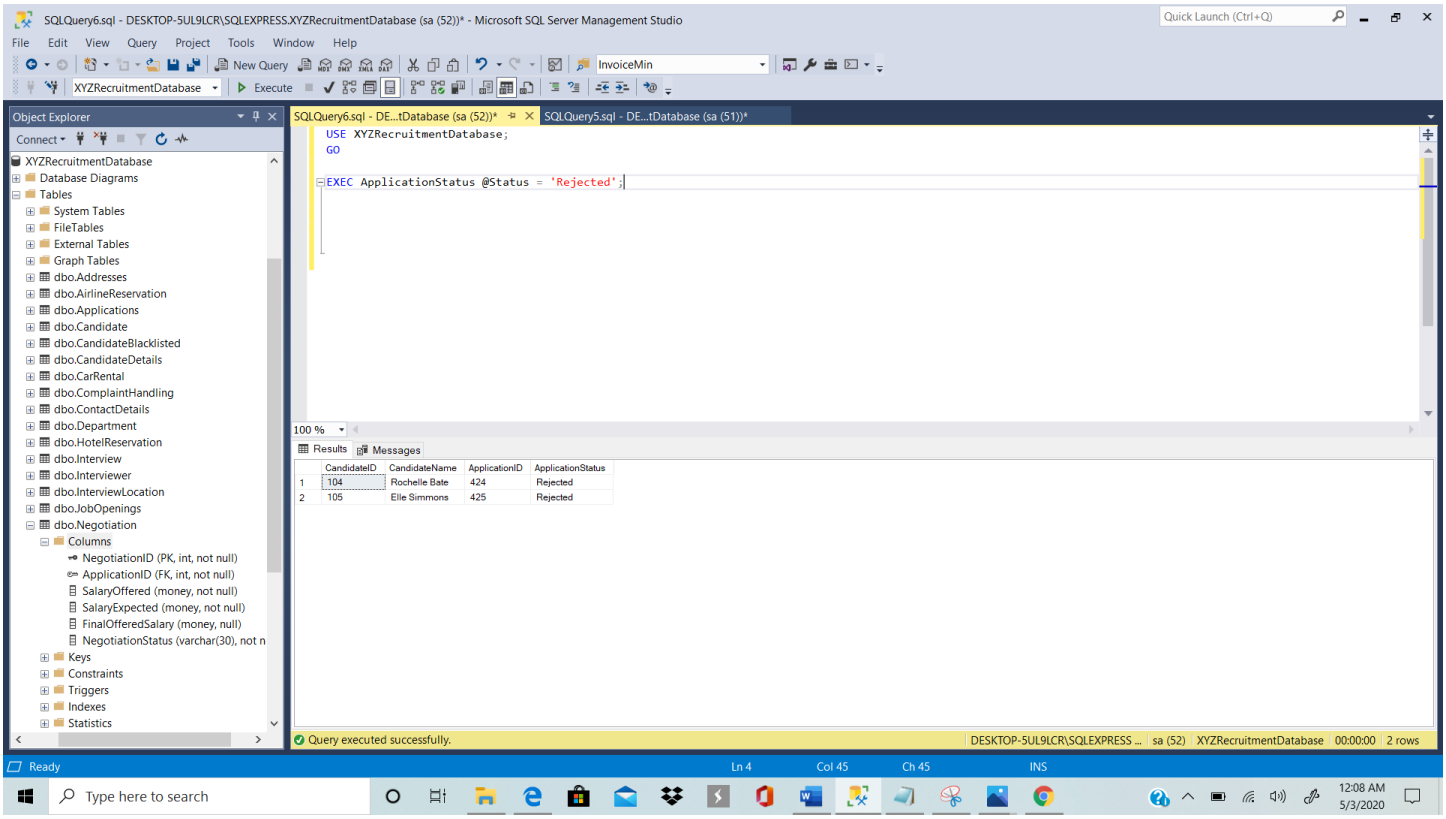
```
CREATE PROC ApplicationStatus @Status VARCHAR(30) = NULL
AS
SELECT x.CandidateID, x.CandidateName, y.ApplicationID, y.ApplicationStatus
FROM Candidate AS x JOIN Applications AS y
        ON x.CandidateID = y.CandidateID
WHERE ApplicationStatus = @Status;
```

The following script creates a procedure named ‘ApplicationStatus’ that returns all the entries from the Applications with the status value passed to the @Status variable used in the procedure.



EXEC ApplicationStatus @Status = 'Rejected';

The following query passes 'Rejected' as the value to the variable @Status. The procedure takes this value and returns the information of all the candidates whose applications have been rejected.



d. Functions

Functions are of two types in SQL. They are: Pre-defined and User-defined functions

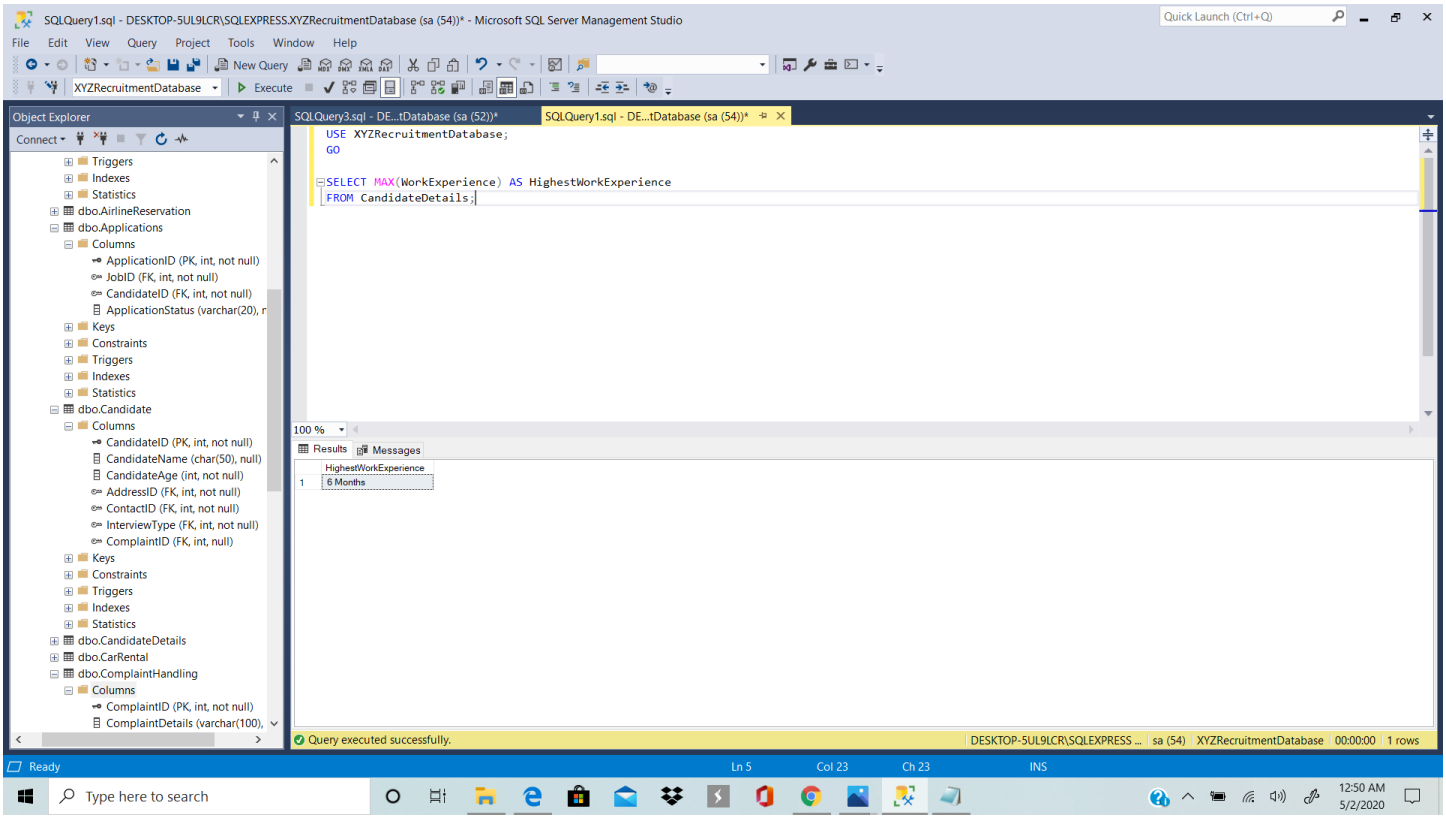
Pre-defined Functions:

In SQL, there are many pre-defined functions aggregate functions like COUNT, MAX, MIN, AVG and so on. These Pre-defined functions are used in SQL SELECT expressions to calculate values and manipulate data. These functions can be used anywhere expressions are allowed. Common uses of functions include to change a name to all upper case.

In this project, I have created one pre-defined function.

```
USE XYZRecruitmentDatabase;
GO

SELECT MAX(WorkExperience) AS HighestWorkExperience
FROM CandidateDetails;
```

The following query returns the maximum work experience that the candidates have.



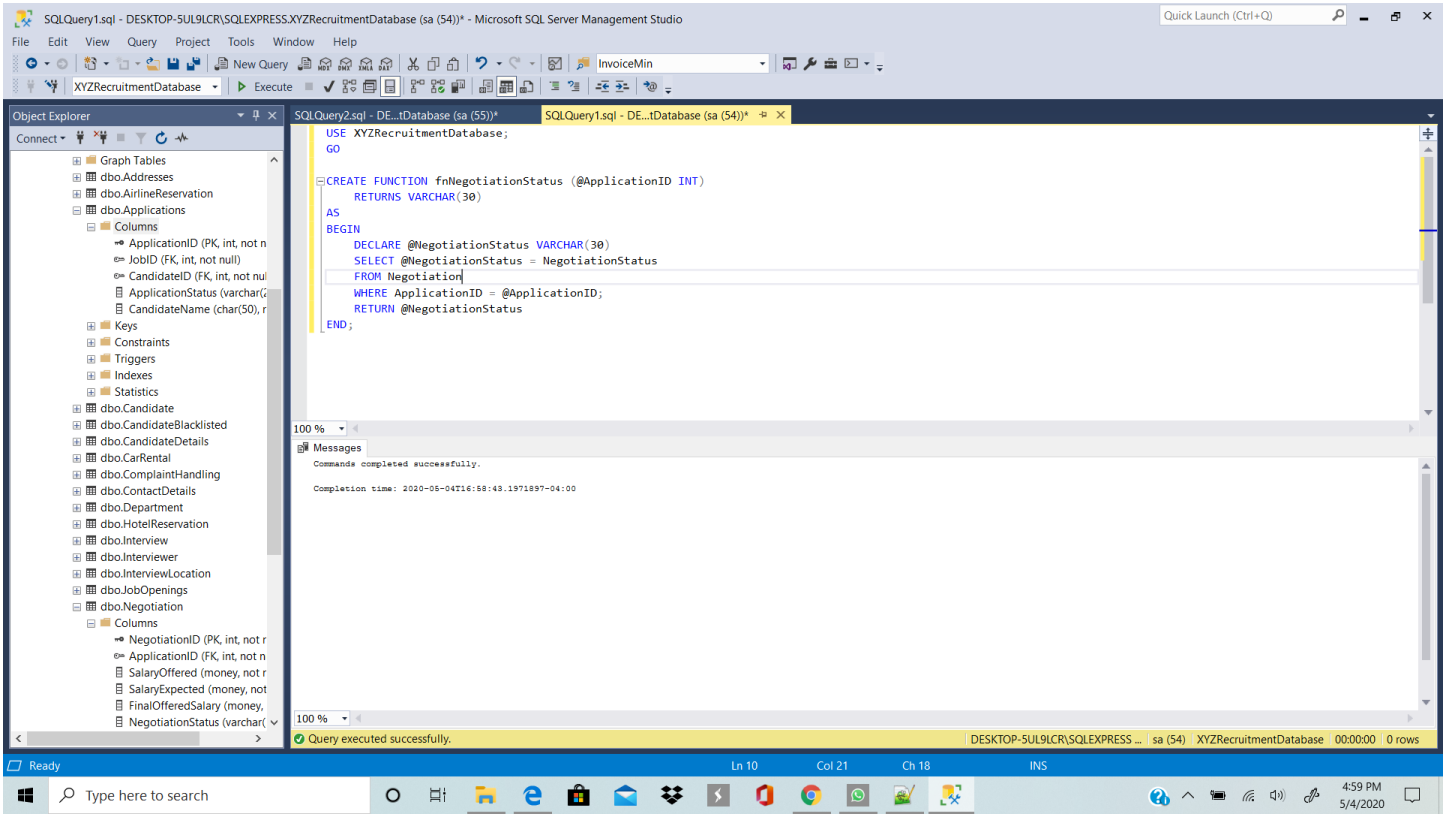
User-Defined Functions:

Like functions in programming languages, SQL Server user-defined functions are routines that accept parameters, perform an action, such as a complex calculation, and return the result of that action as a value. The return value can either be a single scalar value or a result set. There are three types of UDF's. They are Scalar valued functions, Simple table valued functions and Multi-statement table-valued function.

For this project, I have created a Simple table valued function.

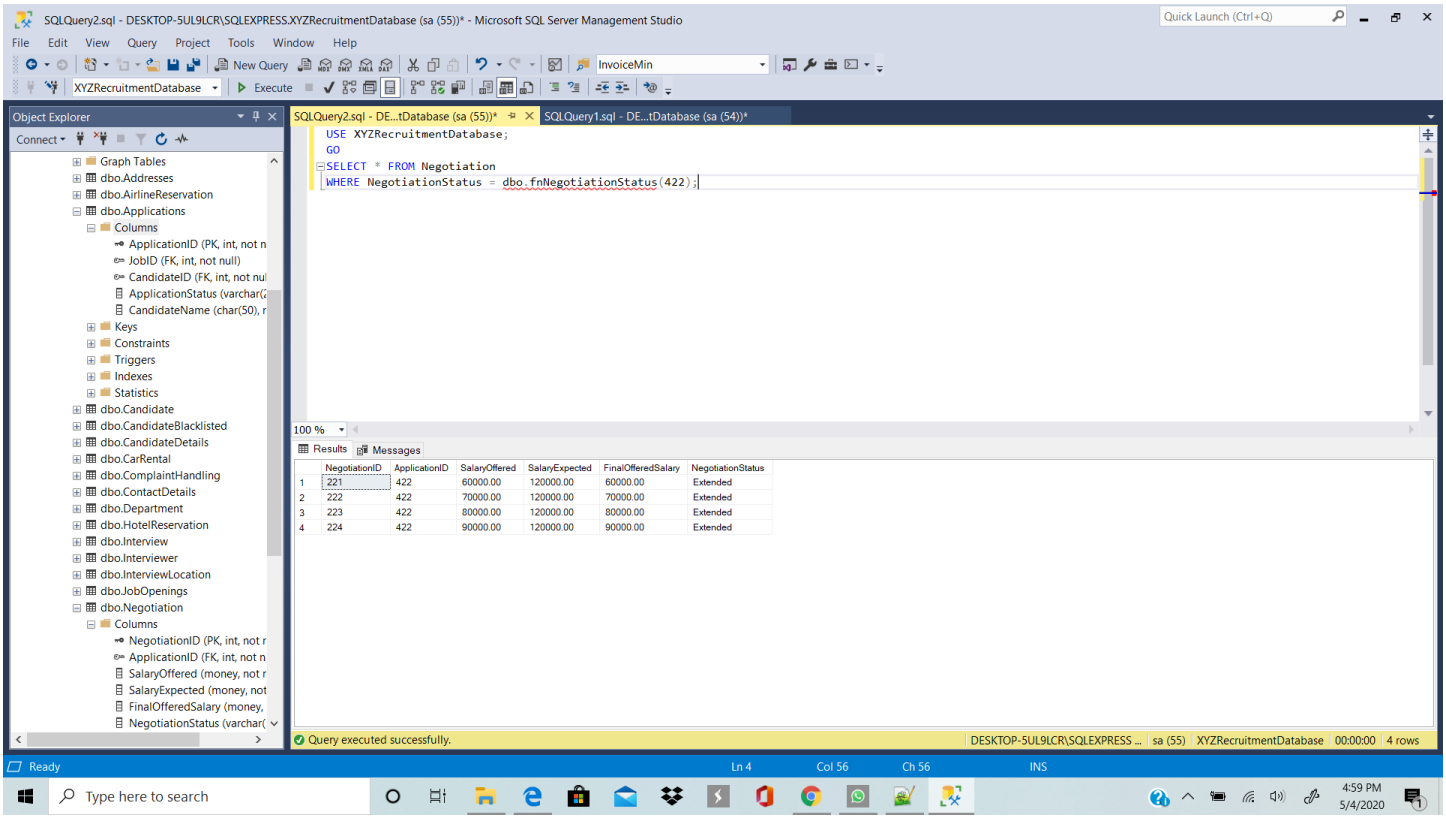
```
CREATE FUNCTION fnNegotiationStatus (@ApplicationID INT)
    RETURNS VARCHAR(30)

AS
BEGIN
    DECLARE @NegotiationStatus VARCHAR(30)
    SELECT @NegotiationStatus = NegotiationStatus
    FROM Negotiation
    WHERE ApplicationID = @ApplicationID;
    RETURN @NegotiationStatus

END;
```

The following query creates a function named 'fnNegotiationStatus'. This function takes an integer value ApplicationID as the input and returns the varchar value, i.e. the negotiation status of that ApplicationID. The actions are performed on the Negotiation table.



SELECT *
FROM Negotiation
WHERE NegotiationStatus = dbo.fnNegotiationStatus(422);

The following query returns the negotiation status for the application ID 422.



e. Scripts

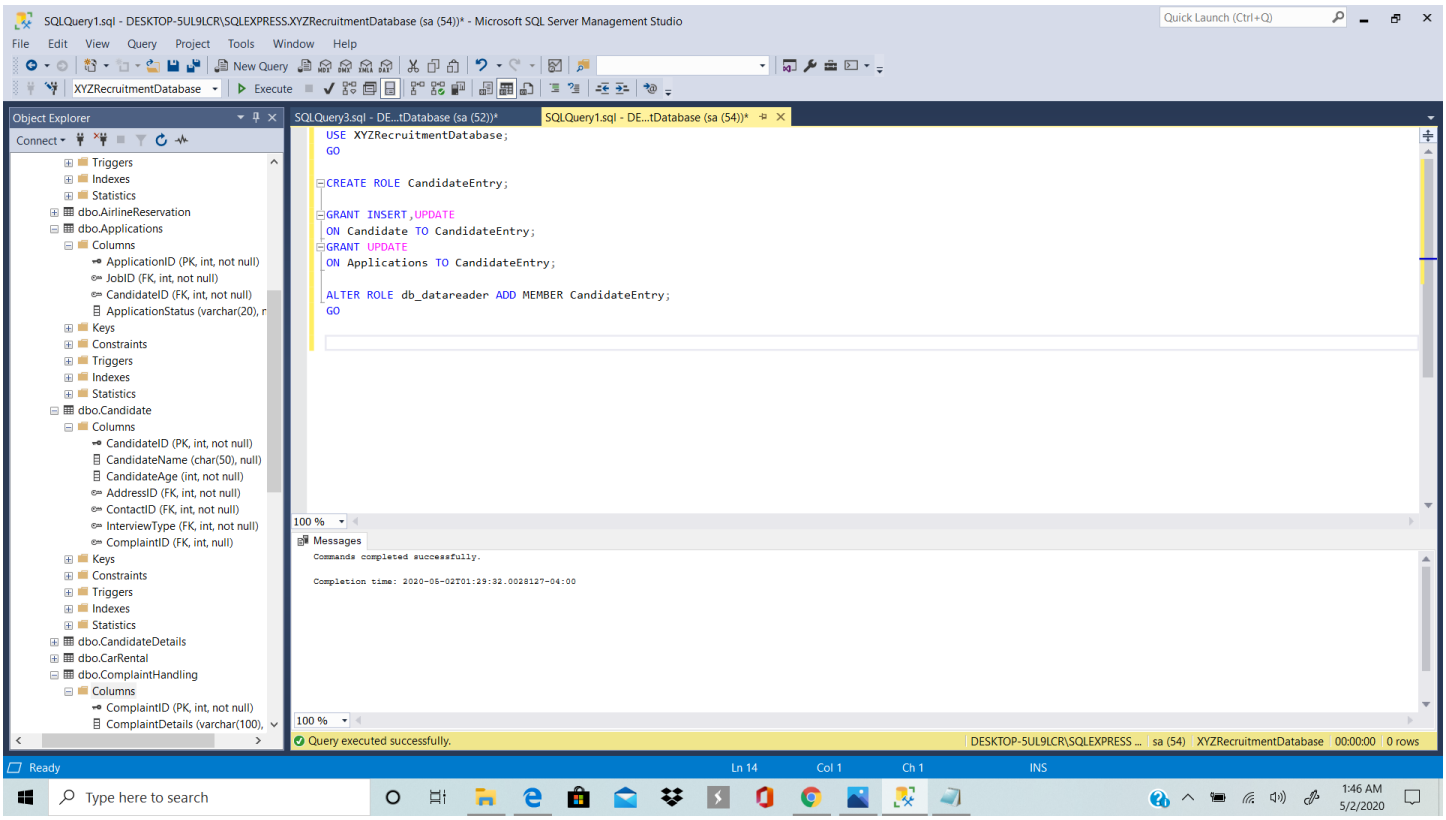
A Script is a series of SQL statements stored in a file. Scripts are one or more batches that are executed as a unit. To signal the end of a batch we use GO command.

For the purpose of this project, I have created two scripts

Script – 1:

```
CREATE ROLE CandidateEntry;

GRANT INSERT,UPDATE
ON Candidate TO CandidateEntry;
GRANT UPDATE
ON Applications TO CandidateEntry;

ALTER ROLE db_datareader ADD MEMBER CandidateEntry;
GO
```

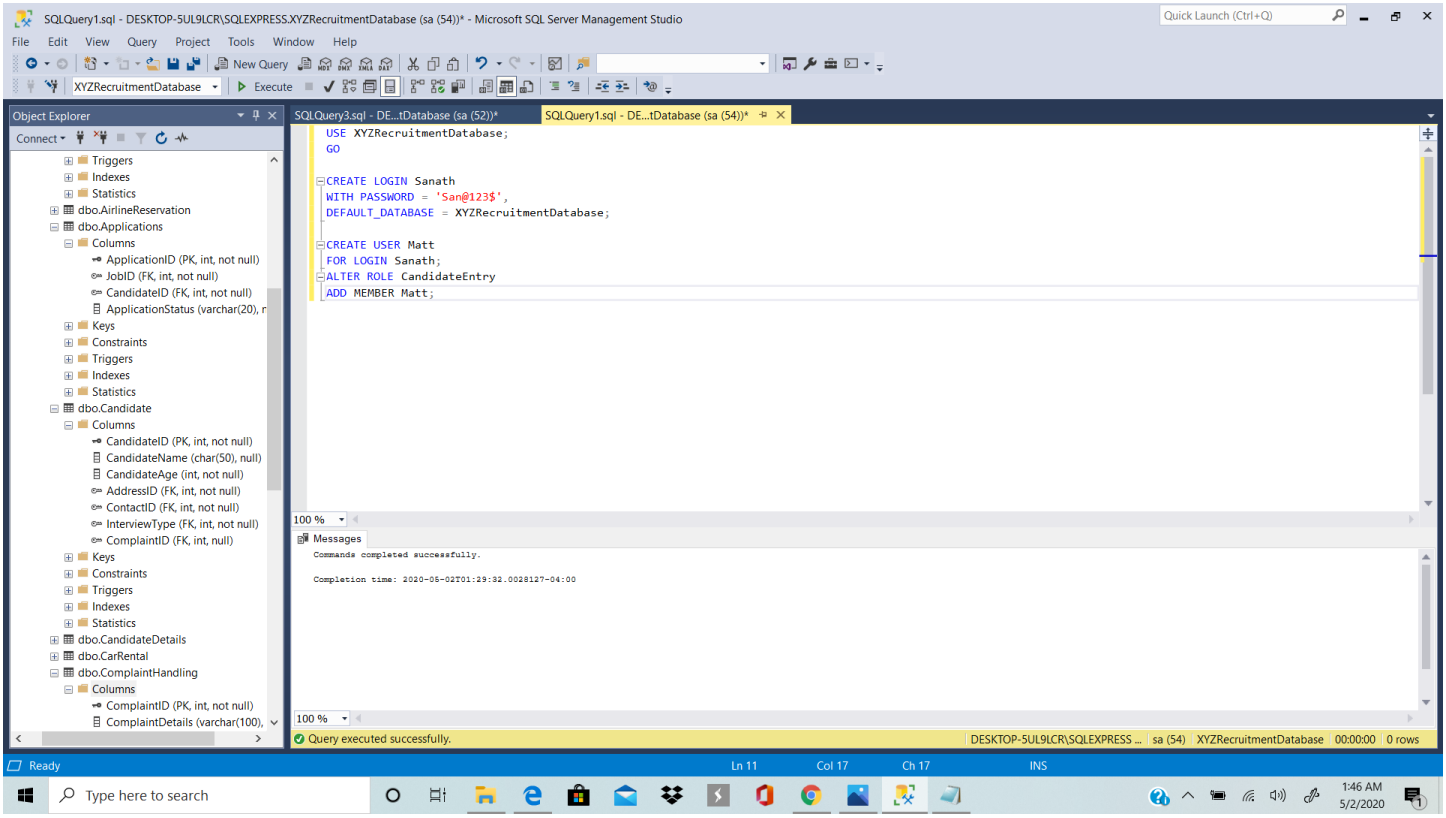
The following script creates a role named 'CandidateEntry'. We assign the insert and update permissions for the CandidateEntry role on the Candidate table. We assign the update permissions on the Applications table for the CandidateEntry role. We assign the select permissions for the CandidateEntry role on all the tables in the database.



Script – 2:

```
CREATE LOGIN Sanath
WITH PASSWORD = 'San@123$',
DEFAULT_DATABASE = XYZRecruitmentDatabase;

CREATE USER Matt
FOR LOGIN Sanath;
ALTER ROLE CandidateEntry
ADD MEMBER Matt;
```

The following script creates a new login ID named ‘Sanath’ with the password ‘San@123$’ on the XYZRecruitmentDatabase. We then create a new user named Matt for this login ID and we add Matt to the CandidateEntry role created in the previous script.



f. Transaction

Transactions are a programming abstraction that enables the database to handle the Concurrency, Recovery and Durability. Using concurrency we can achieve better performance by parallelizing the transactions without creating anomalies. Recovery and Durability can be achieved by keeping the database data consistent and durable in the face of crashes, aborts and system shutdowns.

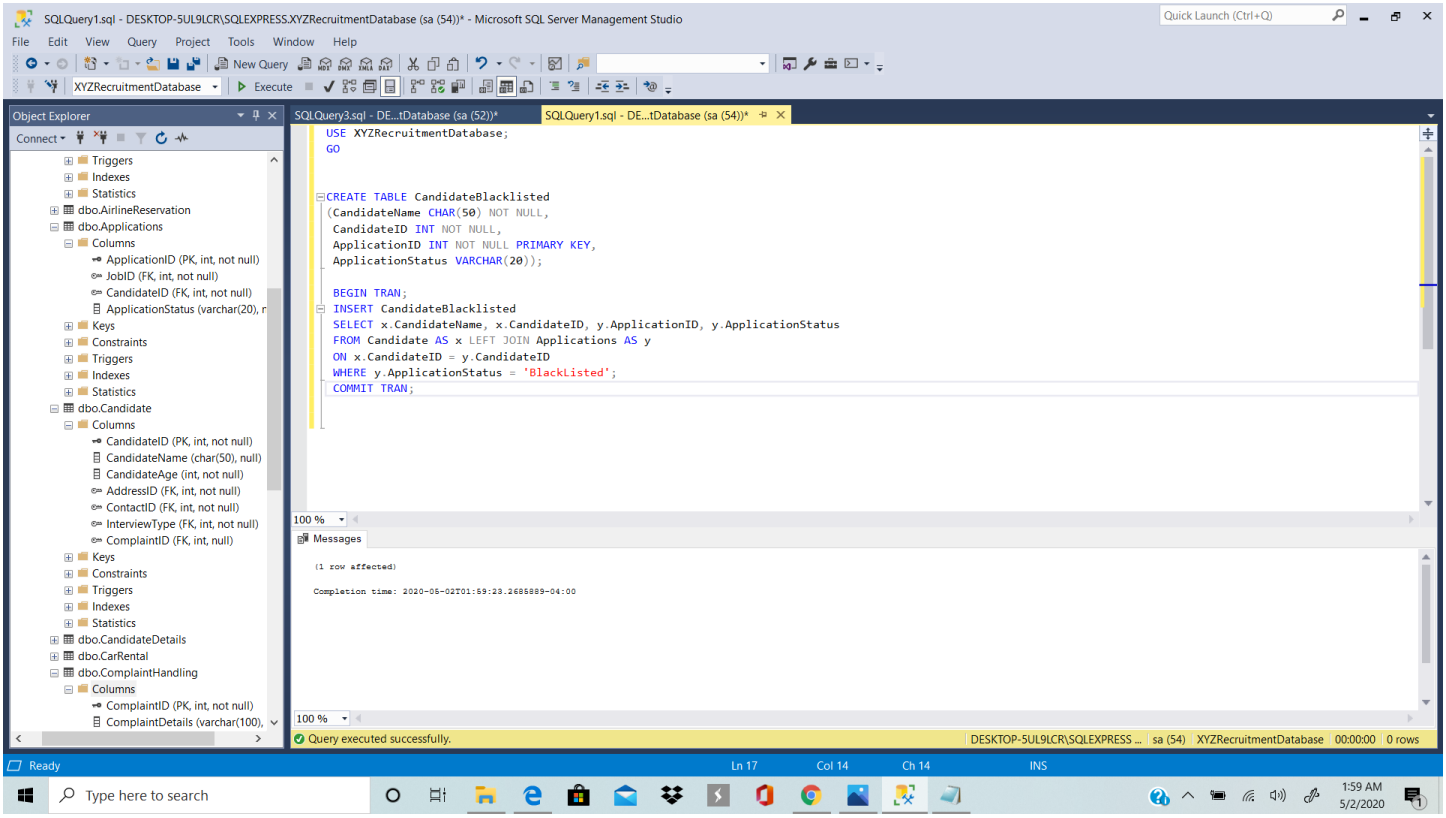
In this project, I have created one transaction script. It is given as follows:

```
CREATE TABLE CandidateBlacklisted
(CandidateName CHAR(50) NOT NULL,
 CandidateID INT NOT NULL,
 ApplicationID INT NOT NULL PRIMARY KEY
 ApplicationStatus VARCHAR(20));
```

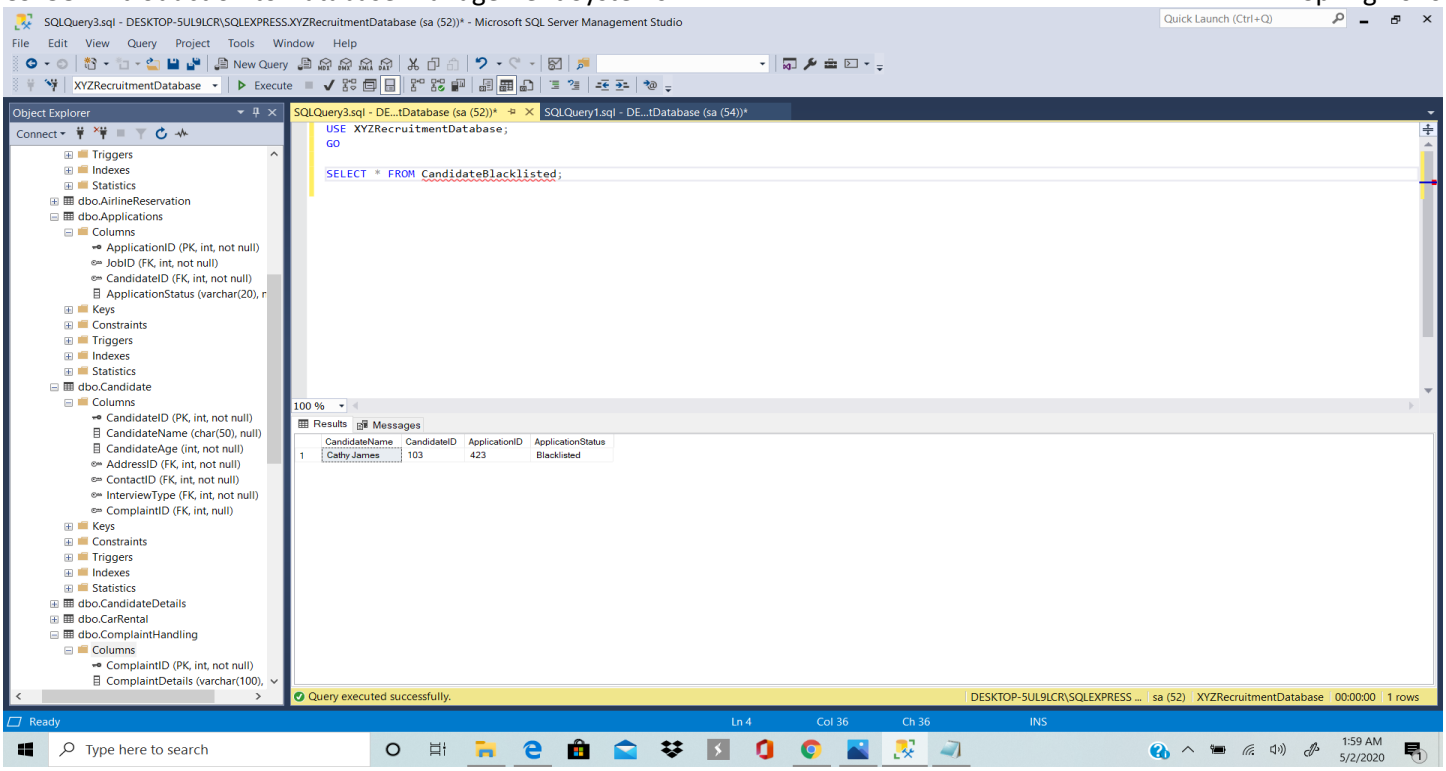
BEGIN TRAN;
INSERT CandidateBlacklisted
SELECT x.CandidateName, x.CandidateID, y.ApplcationID, y.ApplicationStatus
FROM Candidate AS x LEFT JOIN Application AS y
ON x.CandidateID = y.CandidateID
WHERE y.ApplicationStauts = 'Blacklisted';
COMMIT TRAN;

For the purpose of this transaction, I am creating a new table names CandidateBlacklisted that stores the CandidateName, CandidateID, ApplicationID and ApplicationStatus of the blacklisted candidate.

The transaction coded above moves all the rows in the Applications table have ApplicationStatus as Blacklisted to another table named CandidateBlacklisted.



SELECT * FROM CandidateBlacklisted;
The following query retrieves all the rows from the CandidateBlacklisted table.


SQLQuery3.sql - DESKTOP-5UL9LCR\SQLEXPRESS.XYZRecruitmentDatabase (sa (52))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
XYZRecruitmentDatabase
Execute
Object Explorer
Connect
Triggers
Indexes
Statistics
dbo.AirlineReservation
dbo.Applications
Columns
ApplicationID (PK, int, not null)
JobID (FK, int, not null)
CandidateID (FK, int, not null)
ApplicationStatus (varchar(20), r
Keys
Constraints
Triggers
Indexes
Statistics
dbo.Candidate
Columns
CandidateID (PK, int, not null)
CandidateName (char(50), null)
CandidateAge (int, not null)
AddressID (FK, int, not null)
ContactID (FK, int, not null)
InterviewType (FK, int, not null)
ComplaintID (FK, int, not null)
Keys
Constraints
Triggers
Indexes
Statistics
dbo.CandidateDetails
dbo.CarRental
dbo.ComplaintHandling
Columns
ComplaintID (PK, int, not null)
ComplaintDetails (varchar(100),
SQLQuery3.sql - DE...tDatabase (sa (52))*
SQLQuery1.sql - DE...tDatabase (sa (54))*
USE XYZRecruitmentDatabase;
GO
SELECT * FROM CandidateBlacklisted;
100 %
Results
Messages
CandidateName CandidateID ApplicationID ApplicationStatus
1 Cathy James 103 423 Blacklisted
Query executed successfully.
DESKTOP-5UL9LCR\SQLEXPRESS ... sa (52) XYZRecruitmentDatabase 00:00:00 1 rows
Ready
Ln 4 Col 36 Ch 36 INS
Type here to search
1:59 AM
5/2/2020

5. Conclusion

The implementation and design of the XYZRecruitmentDatabase is done in this project. This database is a replica to any organization's recruitment database. This database is created from the scratch and all the data that I have used in my project is considered from the real life values.

In this project, all the basic database concepts are implemented i.e. Starting with the creation of tables, determining its constraints, relationships between the tables used, Entity Relationship diagram, inserting data, usage of standard queries, Normalizing the database, creation of Views, usage of Predefined and User defined functions, Stored Procedures, Triggers and ending with transactions. By doing this project I have got great command over various skills on all the database concepts also got experience on working with the real world project from scratch.

I understood in depth about the aspects of creation of a database, how to design a database, using Entity Relationship diagrams and normalizing those designs. I also understood the implementation of various database concepts that I have mentioned above. Overall, I am now confident to design any database for different purposes.

****