[Human Resources Database]

[CSE581 Project2]
[Wang,Xiao]
[xwang99@syr.edu]

Abstract

The Human Resources (HR) department in any company develops and administers programs designed to increase the organization's effectiveness. It includes the entire spectrum of creating, managing, and cultivating the employer-employee relationship. Human resource management is therefore focused on several major areas, including Recruitment, Compensation and benefits，Training and learning， Labor and employee relations, and Organization Development.
In this project, I will design, implement, and test a database for the Recruitment branch of the HR Department in a company of your choice.

Table of contents

Database Design

Design Introduction

In this project, my design follows the standard recruitment process: 1. Candidates should prepare their resumes, documents, and personal information. 2. The company decided to recruit qualified new employees and provide jobs. 3. The recruitment department needs to publish this requirement and recruit candidates through interviews, tests, and background reviews. 4. When the company finds suitable candidates, then negotiate the treatment. If the applicant accepts the treatment, they become new employees, and the recruitment process ends.
At the same time, according to the project description, some details to improve the fairness of the interview process and improve the applicants' participation in the interview will also be added. Therefore, I follow this core logic and order to build my database.

Database requirements and design analysis

In this project, the requirements document provides 32 requirements and 11 original table templates. The purpose of the database I designed is to effectively and correctly meet all the above requirements.
All designs are based on the requirement documents (analysis of documents are in **Appendix 1.1**) and ordinary business routines. In addition, all scenarios for testing the design are also derived from the description of the requirement document.

Database design

In this part, I will follow the above analysis and recruitment process to design tables with columns, primary keys, foreign keys, proper data types, null-abilities, and necessary relationships and constraints. Each table of the design must at least reach the 3rd Normal Form. At the same time, this design also considers the query overhead when the data size is large enough. For test purposes, I will describe some real scenarios. Based on these scenarios, we will design test data and insert these data into our database. Finally, the E/R diagram of each part of the design will be attached.

Candidates

Candidates have the intention to work but have yet to apply formally. Ten candidates in the scene I designed come from different backgrounds, and all of them have filled required information.

Candidates:

CandidateID(PK)	CandidateFName	CandidateLName
1	Wang	Xiao
2	Wang	YuChen
3	He	XiangYu
4	Krishna	Jash
5	Robbin	Rosan
6	John	Cena
7	Tianlu	Jacoob
8	MengNan	Hong
9	XiaoXiao	Peng
10	MengYing	Wang

CandidatesInfo:

Each candidate can only have one information, and each information can only belong to one candidate , so the relation between them is **One-to-One.**

CandidateInfoID (PK)	CandidateID (FK)	Email	Phone	ShortProfile	Citizenship	Education	LocationInfo
1	1	xwang99@syr.edu	375-88689	C++ Expert & OS design	CHN	Master	NY
2	2	yuchen@syr.edu	375-88690	Java Expert	CHN	Master	NY
3	3	xiangyu@syr.edu	375-88691	C++ Expert	CHN	Master	NY
4	4	Dongl u@bufflo.edu	375-88692	AI& DeepLearning	CHN	Doctor	NY
5	5	Krishna@huawei.edu	375-88693	C Expert	IND	HighSchool	MASS
6	6	John @gmail.com	375-88694	C++ Expert	UnderGraduate	USA	NY
7	7	tianlu @ben	375-89689	C++ Expert	UnderGraduate	USA	NY

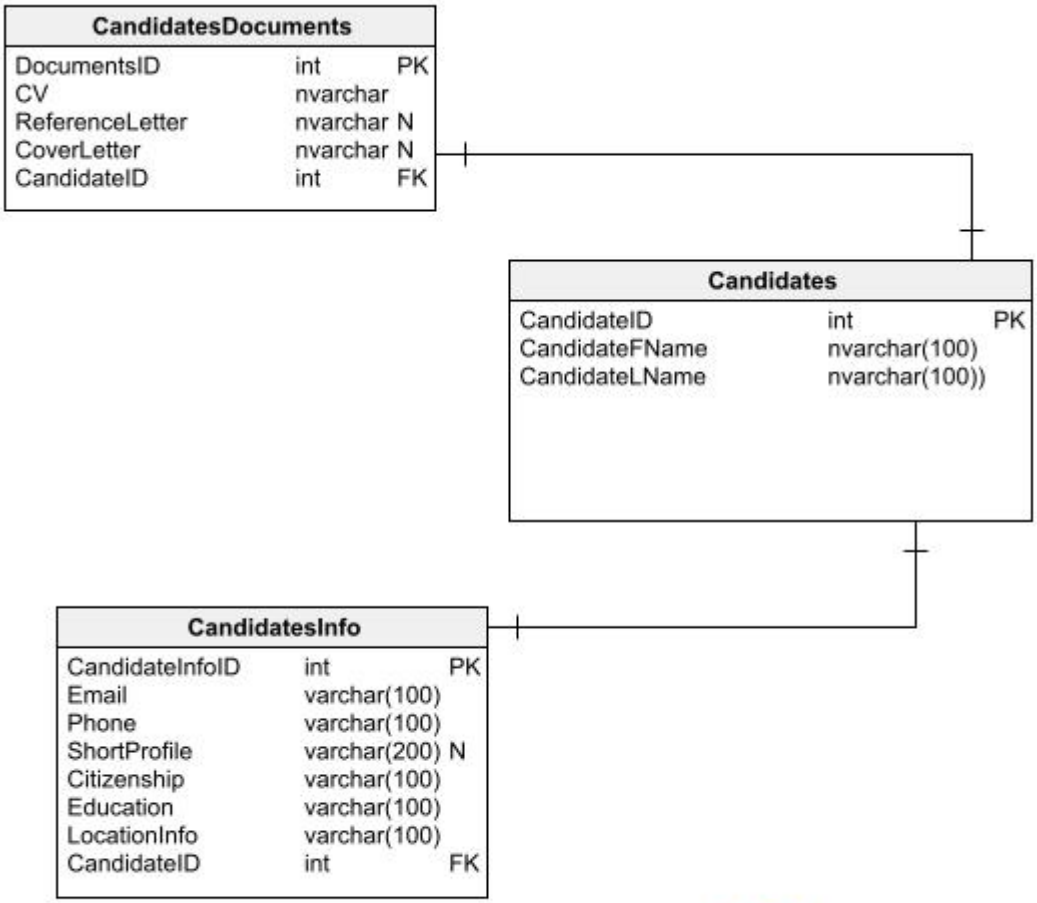
		tley.edu					
8	8	scarlletHong@bentley.edu	375-88989	C++ Expert	UnderGraduate	CHN	NY
9	9	xiaoxiao@syr.edu	375-88289	C++ Expert	Master	CHN	CA
10	10	mengying@syr.edu	375-81689	C++ Expert	Master	CHN	CA

CandidatesDocuments

Each candidate can only have one Document, and each Document can only belong to one candidate , so the relation between them is **One-to-One.**

CandidateDocumentID(PK)	CandidateID(FK)	CV	ReferenceLetter	CoverLetter
1	1	url//resume_abcdefghijklmn123456781	url//RefenceLetter_abcdefghijklmn123456781	url//coverletter_abcdefghijklmn123456781
2	2	url//coverletter_abcdefghijklmn123456782	url//RefenceLetter_abcdefghijklmn123456782	url//coverletter_abcdefghijklmn123456782
3	3	url//coverletter_abcdefghijklmn123456783	url//RefenceLetter_abcdefghijklmn123456783	url//coverletter_abcdefghijklmn123456783
4	4	url//coverletter_abcdefghijklmn123456784	url//RefenceLetter_abcdefghijklmn123456784	url//coverletter_abcdefghijklmn123456784
5	5	url//coverletter_abcdefghijklmn123456785	url//RefenceLetter_abcdefghijklmn123456785	url//coverletter_abcdefghijklmn123456785
6	6	url//coverletter_abcdefghijklmn123456786	url//RefenceLetter_abcdefghijklmn123456786	url//coverletter_abcdefghijklmn123456786
7	7	url//coverletter_abcdefghijklmn123456787	url//RefenceLetter_abcdefghijklmn123456787	url//coverletter_abcdefghijklmn123456787
8	8	url//coverletter_abcdefghijklmn123456788	url//RefenceLetter_abcdefghijklmn123456788	url//coverletter_abcdefghijklmn123456788
9	9	url//coverletter_abcdefghijklmn123456789	url//RefenceLetter_abcdefghijklmn123456789	url//coverletter_abcdefghijklmn123456789
10	10	url//coverletter_abcdefghijklmn1234567810	url//RefenceLetter_abcdefghijklmn1234567810	url//coverletter_abcdefghijklmn1234567810

E/R Diagram for Candidates:



JobOpening

The operation of a company needs employees to create products and make profits. Therefore, when a company needs more employees, it will publish the demand on Jobopening, and the recruitment process will start to help it find qualified employees. The scenario mostly comes from the requirement document, and some descriptions are added to make the design clear.

JobTypes:

JobTypeID(PK)	JobType	JobTypeDescription
1	Summer Internship	Candidate works for 2- 3 months as intern.
2	Full Time	Candidate must work at least 9 hours per day.
3	Part Time	Candidate must work at most 3 hours per day.
4	Contract	Candidate sign with the company and get paid based on workload.

JobCategories:

JobCategoryID(PK)	JobCategory	JobCategoryDescription
1	IT	installing, maintaining and repairing hardware & software components of the organization's computers
2	Software Design	Transfer customers' requirements into services
3	Software Test	responsible for the quality of software development and deployment

JobMediums :

JobMediumID(PK)	JobMedium	JobMediumDescription
1	Online	Work at home
2	Onsite	Work at office

JobPositions :

JobPositionID(PK)	JobPosition	JobMediumDescription
1	Manager	10 years+ working experience
2	Senior	3 years+ working experience
3	Entry level	0-3 years working experience

JobOpeningPlatforms

JobOpeningPlatformID(PK)	PlatformName
1	Company Webpage
2	LinkedIn
3	HandShake

Jobs

Since each JobType can belong to multiple jobs and each job can only have one type, the relation between them is **one-to-many**.
Since each Job category can belong to multiple jobs and each job can only have one Job category, the relation between them is **one-to-many**.
Since each Job Position can belong to multiple jobs and each job can only have one Job Position, the relation between them is **one-to-many**.
Since each Job Medium can belong to multiple jobs and each job can only have one Job medium, the relation between them is one-to-many.

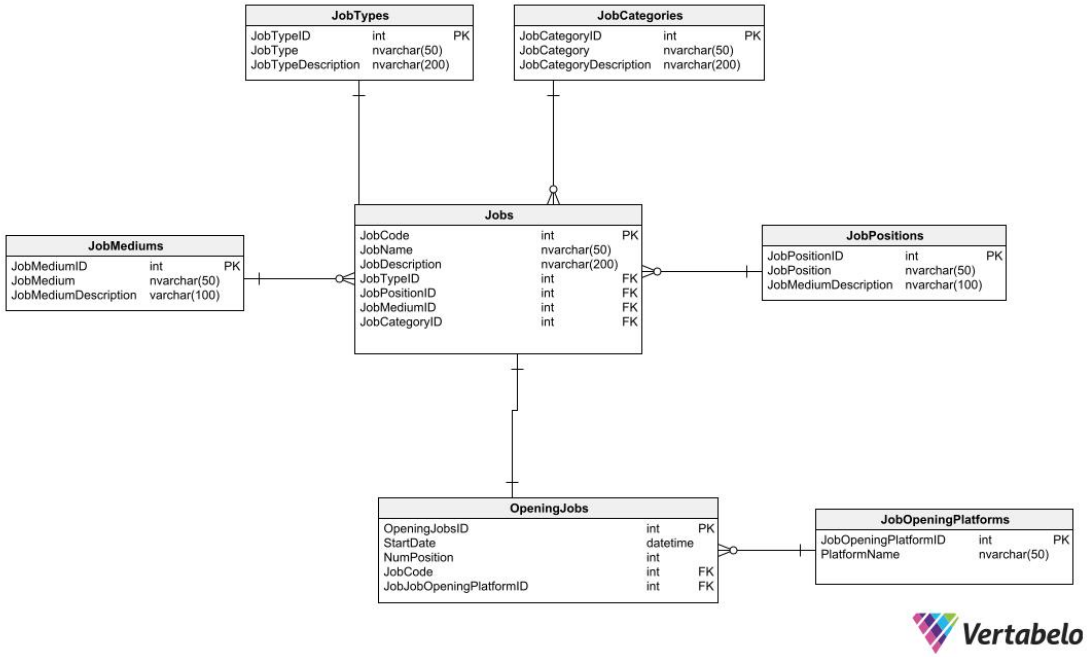
JobCode(PK)	JobName	JobDescription	JobTypeID(FK)	JobCategoryID(FK)	JobMediumID(FK)	JobPositionID(FK)
1	FullStack designer	Use c++ maintain the OS system and design websit	2	2	2	2
2	Backend design	Use x-86 architecture improve performance of the software	2	2	2	2
3	Softer-ware test	Test softerware and design test cases	4	3	1	3
4	IT Guys	Do some trivial works	1	1	2	3

OpeningJobs

Since each JobOpening can belong to one job and each job can only have one JobOpening, the relation between them is one-to-one.
Since each JobOpeningPlatform can belong to multiple JobOpenings and each JobOpenings can only have one Platform, the relation between them is one-to-many.
Check set on `NumPosition ensure it cannot be` **`negtive.`**

JobOpeningID(PK)	JobCode(FK)	StartDate	`NumPosition(>=0)`	JobOpeningPlatformID(FK)
1	1	‘2023-1-1’	2	1
2	2	‘2023-1-1’	2	1
3	3	‘2023-1-1’	1	2
4	4	‘2023-1-1’	1	3

E/R Diagram for Jobs & OpeningJobs:



Interview

The company selects qualified employees through interviews. To accomplish this task, the company will select outstanding employees or technical directors to be interviewers and design test questions in advance. The interviewer will recruit qualified employees based on the test results.

Interview preparation

Interviewers

The recruiting team will arrange for some interviewers to grade the applicants, and we will arrange for three interviewers(In this scenario) to grade all applicants for each interview.

InterviewerID(PK)	InviewerFname	InviewerLname	Titles
1	Wang	PiPi	Chief Technology Officer
2	Ehat	Ercanli	Technical Consulting
3	Lee	Robin	Chief Engineer

TestTypes
In our scenario, each interview will have two tests, one for programming and another for system design. Both tests have a start and end time, both online and onsite.

TestTypeID(PK)	TestType
1	Onsite
2	Online

Tests
Since each Test can belong to multiple TestTypes and each TestType can only have one **Test**, the relation between them is one-to-many.

TestID (PK)	TestStartTime	TestEendTime	TestTittle	TestTypeID
1	2022-11-15 09:00	2022-11-15 11:00	Programming1	1
2	2022-11-15 14:00	2022-11-15 16:00	SystemDesign1	1
3	2022-12-15 09:00	2022-12-15 11:00	Programming2	1
4	2022-12-15 14:00	2022-12-15 16:00	SystemDesign2	1

Interview

We arrange interviews according to the following procedures. We will hold two interviews on 2022-11-15 and 2022-12-15. The interviews can be online or onsite but we decide onsite. There are two tests per interview, and each test is different. At the same time, three interviewers will come to the interview to give each applicant an evaluation.

InterviewTypes

InterviewTypeID(PK)	InterviewType
1	Onsite
2	Online

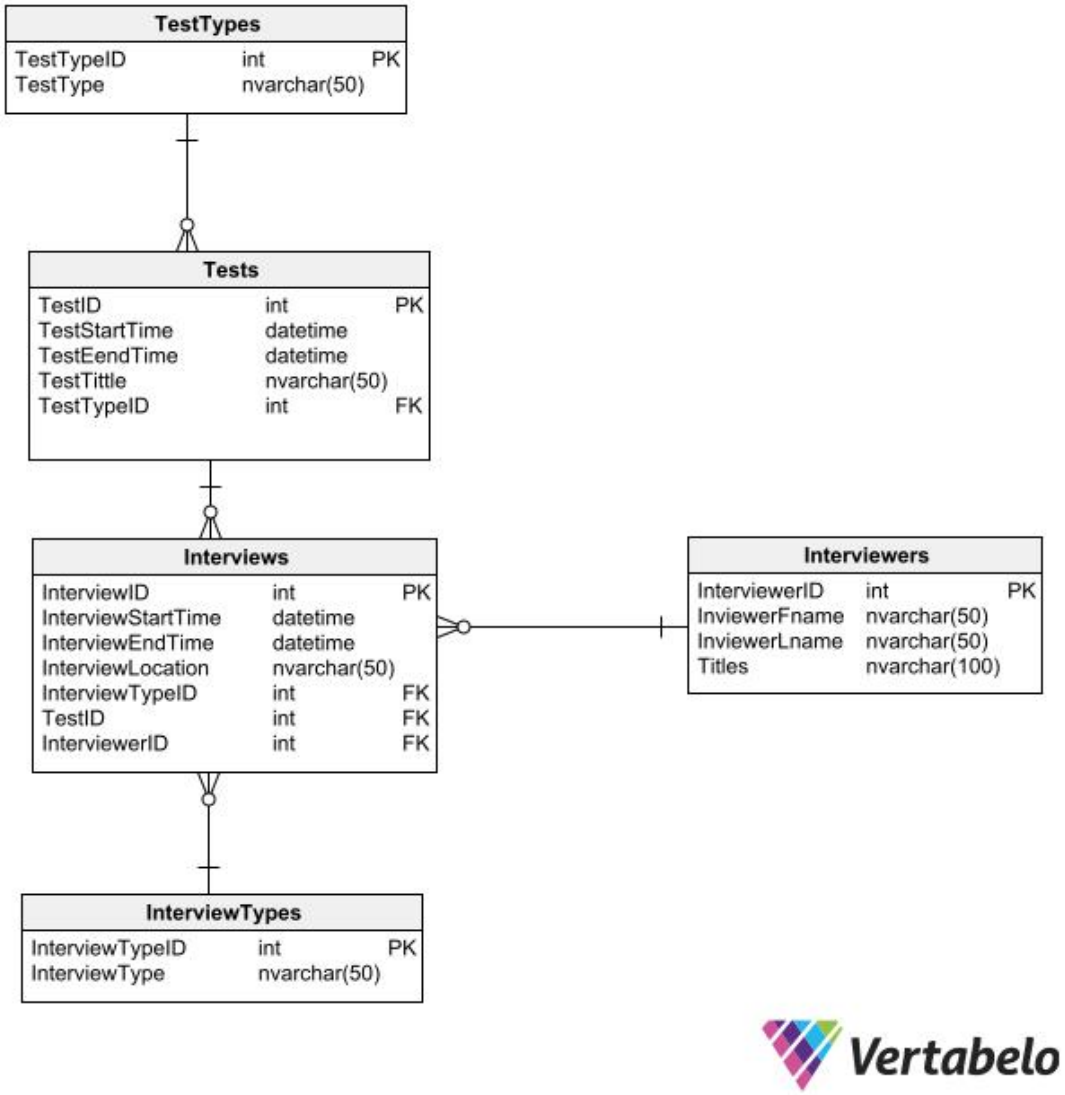
Interviews
One test should belong to one interview, but one interview can have multiple tests.(If the same questions are given to different interviews at different times, it will be unfair to the applicants who participated in the earlier interview) So the relationship is **one to many.**

Each interviewer will involve interviews, and each interviewID can be evaluated by one interviewer. So the relationship is o**ne to many**.

Each interview can have have one type but each type can belong to mutiple interviews. So the relationship is **one to many.**

InterviewID(PK)	InterviewStartTime	InterviewEndTime	InterviewLocation	InterviewTypeID(FK)	TestID(FK)	Interviewer
1	2022-11-15	2022-11-16	NY	1	1	1
2	2022-11-15	2022-11-16	NY	1	1	2
3	2022-11-15	2022-11-16	NY	1	1	3
4	2022-11-15	2022-11-16	NY	1	2	1
5	2022-11-15	2022-11-16	NY	1	2	2
6	2022-11-15	2022-11-16	NY	1	2	3
7	2022-12-15	2022-12-16	NY	1	3	1
8	2022-12-15	2022-12-16	NY	1	3	2
9	2022-12-15	2022-12-16	NY	1	3	3
10	2022-12-15	2022-12-16	NY	1	4	1
11	2022-12-15	2022-12-16	NY	1	4	2
12	2022-12-15	2022-12-16	NY	1	4	3

E/R Diagram for Interviews:



Application & Hiring

Scenarios design

Now I will design ten scenarios, and we will create the final database and insert data through the scenarios.

Scenarios Background: The company decided to recruit staff and announced the job positions through the opening- jobs. All ten candidates got the news and applied for jobs. The company also arranged corresponding interviews and the recruitment process.

Candidate 1: He wants to apply for JobOpenID1 (Fullstack developer). He passed the Application scanning and entered the interview process. In the first round of interviews, he was selected by the judges. The scores given to him by the three judges were test1(100, 100, 100),test2(100,100,100). During the interview process, he and the interviewers have no mutual evaluation. Then, He accepted the offer, passed the onboarding investigation, and became an employee.

Candidate 2: He wants to apply for jobopenID1 (Fullstack developer) passed the Application scanning and entered the interview session. The poor performance in the first round of interviews was scored test1 (70, 70, 80), test2(70,70,70) but the excellent performance in the second round was scored (100, 100, 100),(100,100,100). There was no mutual evaluation between him and the interviewers during the interview. He accepted the offer, passed the onboarding survey, and became an employee.

Candiate3: He wants to apply for jobopenID1 (Fullstack developer). He passed the Application scanning and entered the interview process. The interviewers selected him in the first round of interviews. The scores given to him by the three judges were (100, 100, 100),(90,100,90). During the interview process, he and the judges have no mutual evaluation. However, jobopenID1 has been fully recruited, and he was put on the on-call status.

Candiate4: He wants to apply for jobopenID2 (Backend design). He passed the Application scanning and entered the interview process. He was eliminated in the first round of interviews. The scores given to him by the three judges were (30, 25, 40),(30,40,30). During the interview process, he and the interviewers have no mutual evaluation. **Rejected**.

Candiate5: He wanted to apply for jobopenID2(Backend design) but failed the Application scanning and directly **rejected the list**.

Candidate 6: He wants to apply for jobopenID2(Backend design) and pass Application scanning. He entered the interview process. He was eliminated in the first round of interviews. The scores given to him by the three judges were (45, 55, 40),(45,55,50). During the interview process, there is no mutual evaluation with the

reviewers. The applicant complained, saying that the test was too difficult, but the investigation result was still **rejected**.

Candidate7: He wanted to apply for jobopenID2 (Backend design) and passed Application scanning. He entered the interview process. He was selected in the first round of interviews. The scores given to him by the three judges were (100, 98, 100),(100, 98, 100). During the interview, there was no mutual evaluation with the judges. He accepted the offer but failed the onboarding investigation and was put on the **on-call list**.

Candidate8: He wanted to apply for jobopenID2 (Backend design) and passed Application scanning. He entered the interview process. He was selected in the first round of interviews. The scores given to him by the three judges were (80, 98, 90). During the interview, there was no mutual evaluation with the judges; he did not accept the offer(**declined**) and was put on the **on-call** list.

Candidate9: She came from CA. She bought a plane ticket, stayed in a hotel, rented a car, and generated expenses that needed to be reimbursed. She wanted to apply for jobopenID2(Backend design) and passed the Application scanning. Then, she entered the interview process. She was selected in the first round of interviews. The three judges gave him scores of (100, 98, and 100),(100, 80, and 80). She accepted the offer, passed the Onboarding investigation, and became an **employee**.

Cadidate10: She came all the way from CA, bought a plane ticket, stayed in a hotel, rented a car, and incurred expenses that need to be reimbursed. She wanted to apply for jobopenID2(Backend design) and passed the Application scanning. Then, she entered the interview session. The three judges gave him a score of (80, 70, 80),(80, 70, 80) During the interview, she thought that one (ID1 Wang Pipi) of the interviewers sexually harassed her, so she wrote Comments, she tried to communicate with the company, but no one accepted. She was very annoyed and decided to retaliate against the company. So she participated in the second interview. This time, she bought a first-class air ticket, lived in the best hotel, and rented a Bentley car. In the second interview, her score was (100, 80, and 80),(100, 80, and 80). In order to retaliate against the company, she accepted the offer but did not become an employee, so she is **Blacklisted**.

Reimbursement system

In order to encourage candidates to participate in the on-site interview, the company provides reimbursement for interviews not close to the candidate's residence who participates in the on-site interview. At the same time, candidates need to provide relevant information to prove that their expenses are reasonable.

AirlineReservations

ONE reimbursement can combined by mutiple AirlineReservations, but each AirlineReservation can only belong to one reimbursement . The relation is **many to one**.
Check AirLineCost>=0

AirlineReservationID(PK)	AirlineName	AirlineTrackNumber	AirLineCost	ReimbursementID(FK)
1	JetBlue	TN359462YH	300	9
2	Delta	TN359432YH	330	10
3	UltraLuxury	TN358888YH	1888	10

HotelReservations
ONE reimbursement can combined by mutiple HotelReservations, but each HotelReservation can only belong to one reimbursement . The relation is many to one.
Check HotelCost>=0

HotelReservationID(PK)	HotelName	HotelTrackNumber	HotelCost	ReimbursementID(FK)
1	Hilton	HT359462YH	300	9
2	Hilton	HT359432YH	310	10
3	The Ranch at Rock Creek	HT358888YH	4700	10

CarRentals
ONE reimbursement can combined by mutiple CarRentals, but each CarRental can only belong to one reimbursement . The relation is many to one.
Check CarRentCost>=0

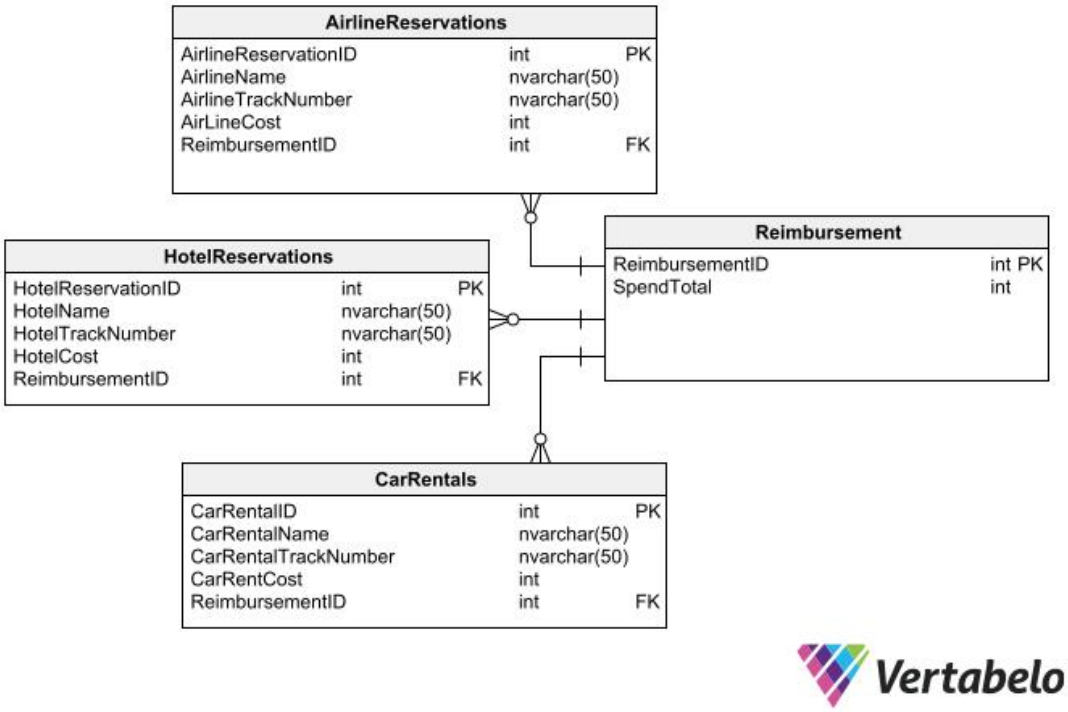
CarRentalID(PK)	CarRentalName	CarRentalTrackNumber	CarRentCost	ReimbursementID(FK)
1	Toyota	CAR359462YH	100	9
2	Toyota	CAR359432YH	120	10
3	Benltley	CAR358888YH	2000	10

Reimbursement

ReimbursemenID	SpendTotal
1	0
2	0
3	0
4	0
5	0
6	0

7	0
8	0
9	700
10	9348

E/R Diagram for `Reimbursement`



Application

Through the project statement, we can group applicants into the following statuses.

Status ID	Statues Name	Statues Description
1	Submitted	Application created successfully
2	Reject	1. Application scanning is not met 2. Fail in the interview
3	Select	Select by the interview
4	Waiting	Complains About the hiring or interviewing process
5	Re-interview	Complain successful
6	on-call	1. Select but no more job openings 2. Onboarding is unsuccessful 3. Decline the offer
7	Negotiating	1. Negotiation Negotiate the offers 2. Decline:Not accept offers (Since the declined candidates are put into the on-call list, I use on call to present the status) 3. Accept offers
8	Onboarding	Background checking
9	BlackListed	Cannot enter the application procedure any more
10	Employee	End of the application-hire procedure

Each applicant will eventually have a status, and I choose to classify and store the applicants under these different statuses tables. First, the selection process consumes a lot of money and time. Therefore, proper storage of this information can maximize the company's revenue. At the same time, when there are many applicants in the future, the system overhead of select_join will be substantial, so we store them in advance to avoid future overhead.

Thus, I divided them by the following tables, and the above scenario inserts data.

STATUES

Status ID(PK)	StatueName
1	Submitted
2	Reject
3	Select
4	Waiting
5	Re-interview
6	OnCall
7	Negotiating
8	Accept
9	Onboarding
10	BlackListed
11	Employee
12	NextRound

OnCallType

OncallTypeID(PK)	OncallReason
1	Select but no more job openings
2	Onboarding is unsuccessful
3	Decline the offer

OnCallList

One applicant can only have one statute, so the relation is **one-to-one.**
Each on-call has one reason, but one reason can belong to many on-calls, so **one to many.**

OncallCode(PK)	OncallTypeID(FK)	ApplicantID(FK)
1	1	3
2	2	7
3	3	8

RejectType

RejectTypeID(PK)	RejectType
1	Application scanning is not met
2	Fail in the interview

RejectList
One applicant can only have one statute, so the relation is **one-to-one.**

Each Reject has one reason, but one reason can belong to many on-calls, so **one to many.**

RejectCode(PK)	ApplicantID(FK)	RejectTypeID(FK)
1	4	2
2	5	1
3	6	2

Onboarding

One applicant can only have one statute, so the relation is **one-to-one.**

OnboardingID(PK)	ApplicationID	EducationCheck	Legitimacy
1	1	pass	pass
2	2	pass	pass
3	9	pass	pass
4	7	fail	fail
5	10	pass	pass

BlackListed

One applicant can only have one statute, so the relation is **one-to-one.**

BlackListedID(PK)	ApplicationID(FK)
1	10

Employee

One applicant can only have one statute, so the relation is **one-to-one.**

EmployeeID(PK)	ApplicationID(FK)
1	1
2	2
3	9

ComplaintHandling

One applicant can only have one statute, so the relation is **one-to-one.**

ComplaintID(PK)	ApplicationID(FK)	ComplainDesc
1	6	This test is so hard that I think the company is making things difficult.

Application

The application table is the core for recruiting, candidates applying, arranging interviews, and admitting/rejecting candidates.

Each candidate can apply multiple applications, but each applicant can only belong to one candidate. One-to- Many
Each application has one chance to reimburse, and one reimbursement can only belong to one applicant. One- to -One.
Each job can be applied by multiple applicant, but each applicant can only apply one job. One-To-Many
Each applicant can have only one statue but each statue can belong to many applicants. One-To -Many

ApplicationID(PK)	OpeningJobsID(FK)	StatueID(FK)	CandidateID (FK)	ReimbursementID(FK)
1	1	11	1	1
2	1	11	2	2
3	1	6	3	3
4	2	2	4	4
5	2	2	5	5
6	2	2	6	6
7	2	6	7	7
8	2	6	8	8
9	2	11	9	9
10	2	10	10	10

Evaluation:
Since each applicant has to participate in multiple interviews, and one interview also interviews multiple applicants. Therefore, we need a link table to present this many-to-many relationship, and the grade will also be given in the evaluation.
Check Grade >=0 &Grade<=100

EvaluationID(PK)	InterviewID(FK)	ApplicationID(FK)	GRADE
1	1	1	100
2	2	1	100
3	3	1	100
4	4	1	100
5	5	1	100
6	6	1	100
7	1	2	70
8	2	2	70
9	3	2	80
10	4	2	70
11	5	2	70
12	6	2	80
13	7	2	100
14	8	2	100
15	9	2	100
16	10	2	100
17	11	2	100
18	12	2	100
19	1	3	100
20	2	3	100
21	3	3	100
22	4	3	100
23	5	3	100
24	6	3	100
25	1	4	30
26	2	4	25
27	3	4	40
28	4	4	0
29	5	4	0
30	6	4	0
31	1	6	45
32	2	6	55
33	3	6	60
34	4	6	45
35	5	6	30
36	6	6	27
37	1	7	100
38	2	7	98
39	3	7	100
40	4	7	100
41	5	7	98
42	6	7	100
43	1	8	80

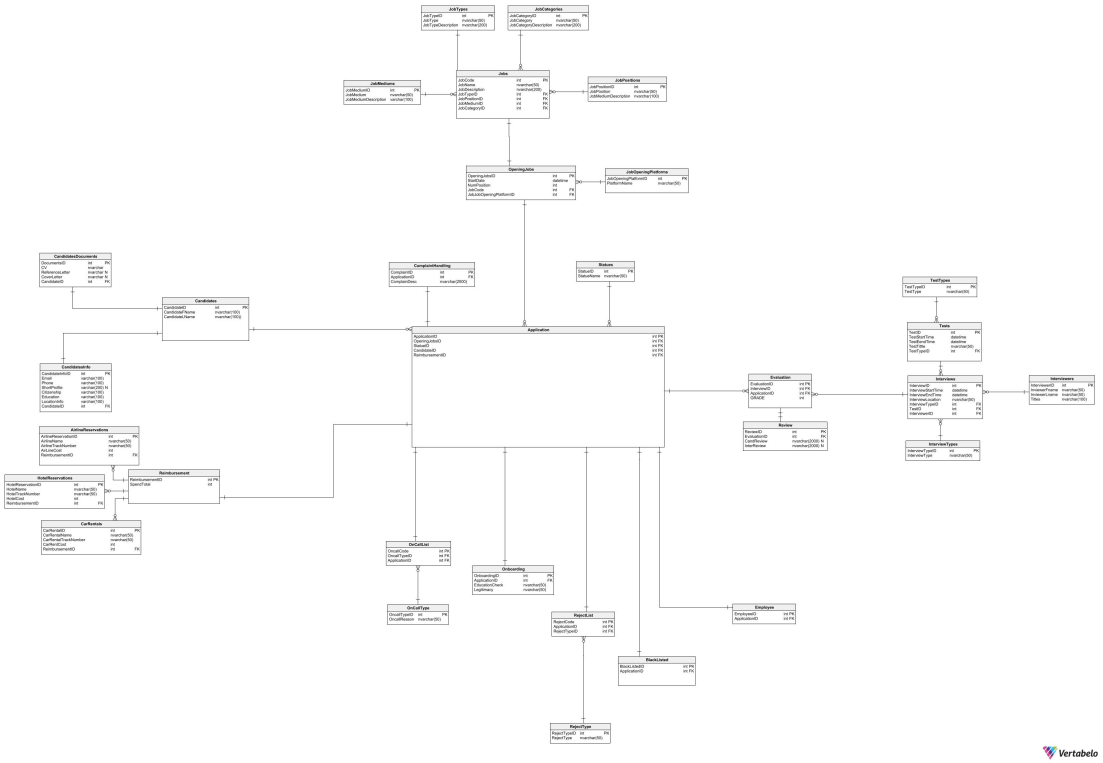
44	2	8	98
45	3	8	90
46	4	8	100
47	5	8	100
48	6	8	100
49	1	9	100
50	2	9	98
51	3	9	100
52	4	9	100
53	5	9	100
54	6	9	100
55	1	10	80
56	2	10	70
57	3	10	80
58	4	10	80
59	5	10	70
60	6	10	80
61	7	10	100
62	8	10	70
63	9	10	80
64	10	10	100
65	11	10	70
66	12	10	80

Review:

ReviewID	EvaluationID	CandReview	InterReview
1	55	I could not believe that the examiner said my breasts stood beautiful and that if I could have a one-night stand with him, he could pass me straight away.	Strong applicant!
2	61	There is no one in your company to deal with this matter, and there has no handling mechanism. Just wait for my fireback!!!	Very Strong applicant!

E/R Diagram for Recruitment

31 tables for the design.



Database Implementation

Create database

In this project, we will design a database for Human Resources (HR) department, so the database is named HR, and we will focus on the Recruitment branch, so the Schema is Recruitment.
For the implementation:
1. I follow the above to design tables with columns, primary keys, foreign keys, proper data types, null abilities, and necessary relationships and constraints.
2. For the integrity proposal, all primary and foreign keys are set constraints.
3. Each table of the design must at least reach the 3rd Normal Form.
4. I cluster their primary key to increase performance for a considerable size table.

SQL for Create database (Screen shots are in the **Appendix 1.2 screenshots):**

```
USE master
GO

IF SCHEMA_ID('Recruitment') IS NOT NULL
DROP SCHEMA Recruitment
GO

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


CREATE DATABASE HR
GO

USE HR
GO

CREATE SCHEMA Recruitment
GO



CREATE TABLE  Recruitment.Candidates
(
	CandidateID int IDENTITY NOT NULL,
	CandidateFName  varchar(100) NOT NULL,
	CandidateLName varchar(100) NOT NULL,
	--constraint and performance
	CONSTRAINT PK_Candidates PRIMARY KEY CLUSTERED(CandidateID)
```

```
)

CREATE TABLE  Recruitment.CandidatesInfo
(
	CandidateInfoID int IDENTITY NOT NULL,
	Email varchar(100) NOT NULL,
	Phone varchar(100) NOT NULL,
	ShortProfile varchar (200) NULL,
	Citizenship varchar(100) NOT NULL,
	Education varchar(100) NOT NULL,
	LocationInfo varchar(100) NOT NULL,
	CandidateID int NOT NULL
	--constraint and performance
	CONSTRAINT PK_CandidatesInfo PRIMARY KEY CLUSTERED(CandidateInfoID)
	--FK Address potential integrity
	CONSTRAINT FK_CandidatesInfo_Candidates FOREIGN KEY(CandidateID)
REFERENCES Recruitment.Candidates(CandidateID)
)

CREATE TABLE  Recruitment.CandidatesDocuments
(
	CandidateDocumentID int IDENTITY NOT NULL,
	CV varchar(2000) NOT NULL,
	ReferenceLetter varchar(2000)NULL,
	CoverLetter varchar (2000) NULL,
	CandidateID int NOT NULL
	--constraint and performance
	CONSTRAINT PK_CandidatesDocuments PRIMARY KEY
CLUSTERED(CandidateDocumentID)
	--FK Address potential integrity
	CONSTRAINT FK_CandidatesDocuments_Candidates FOREIGN KEY(CandidateID)
REFERENCES Recruitment.Candidates(CandidateID)
)


CREATE TABLE Recruitment.JobTypes (
  JobTypeID int IDENTITY NOT NULL,
  JobType nvarchar(50)  NOT NULL,
  JobTypeDescription nvarchar(200)  NOT NULL,
  CONSTRAINT JobTypes_pk PRIMARY KEY  (JobTypeID)
);

CREATE TABLE Recruitment.JobMediums  (
  JobMediumID int IDENTITY  NOT NULL,
  JobMedium nvarchar(50)  NOT NULL,
  JobMediumDescription varchar(100)  NOT NULL,
  CONSTRAINT JobMediums_pk PRIMARY KEY  (JobMediumID)
);

CREATE TABLE Recruitment.JobCategories (
  JobCategoryID int IDENTITY NOT NULL,
  JobCategory nvarchar(50)  NOT NULL,
  JobCategoryDescription nvarchar(200)  NOT NULL,
  CONSTRAINT JobCategories_pk PRIMARY KEY  (JobCategoryID)
);
CREATE TABLE Recruitment.JobPositions  (
  JobPositionID int IDENTITY NOT NULL,
```

```
  JobPosition nvarchar(50)  NOT NULL,
  JobMediumDescription nvarchar(100)  NOT NULL,
  CONSTRAINT JobPositions_pk PRIMARY KEY  (JobPositionID)
);
CREATE TABLE Recruitment.JobOpeningPlatforms (
  JobOpeningPlatformID int IDENTITY NOT NULL,
  PlatformName nvarchar(50)  NOT NULL,
  CONSTRAINT JobOpeningPlatforms_pk PRIMARY KEY  (JobOpeningPlatformID)
);

CREATE TABLE Recruitment.Jobs (
  JobCode int IDENTITY NOT NULL,
  JobName nvarchar(50)  NOT NULL,
  JobDescription nvarchar(200)  NOT NULL,
  JobTypeID int  NOT NULL,
  JobCategoryID int  NOT NULL,
  JobPositionID int  NOT NULL,
  JobMediumID int  NOT NULL,
  CONSTRAINT Jobs_pk PRIMARY KEY CLUSTERED (JobCode),--Increase performance
  CONSTRAINT FK_Jobs_JobTypes FOREIGN KEY(JobTypeID) REFERENCES
Recruitment.JobTypes(JobTypeID),--interity constrains
  CONSTRAINT FK_Jobs_JobCategories FOREIGN KEY(JobCategoryID) REFERENCES
Recruitment.JobCategories(JobCategoryID),
  CONSTRAINT FK_Jobs_JobPositions FOREIGN KEY(JobPositionID) REFERENCES
Recruitment.JobPositions(JobPositionID),
  CONSTRAINT FK_Jobs_JobMediums FOREIGN KEY(JobMediumID) REFERENCES
Recruitment.JobMediums(JobMediumID)
);

CREATE TABLE Recruitment.OpeningJobs (
  OpeningJobsID  int IDENTITY NOT NULL,
  StartDate  datetime  NOT NULL,
  NumPosition  int  NOT NULL CHECK(NumPosition>=0),--Avoid mistake
  JobCode int  NOT NULL,
  JobJobOpeningPlatformID int  NOT NULL,
  CONSTRAINT OpeningJobs_pk PRIMARY KEY CLUSTERED (OpeningJobsID ),
  CONSTRAINT OpeningJobs_Jobs FOREIGN KEY (JobCode) REFERENCES
Recruitment.Jobs (JobCode),
  CONSTRAINT OpeningJobs_JobOpeningPlatforms FOREIGN KEY
(JobJobOpeningPlatformID) REFERENCES Recruitment.JobOpeningPlatforms
(JobOpeningPlatformID)
);


CREATE TABLE Recruitment.Interviewers (
  InterviewerID int  IDENTITY  NOT NULL,
  InviewerFname nvarchar(50)  NOT NULL,
  InviewerLname nvarchar(50)  NOT NULL,
  Titles  nvarchar(100)  NOT NULL,
  CONSTRAINT Interviewers_pk PRIMARY KEY CLUSTERED (InterviewerID)
);

CREATE TABLE Recruitment.TestTypes (
  TestTypeID int IDENTITY NOT NULL,
  TestType nvarchar(50)  NOT NULL,
  CONSTRAINT TestTypes_pk PRIMARY KEY  (TestTypeID)
);
```

```
CREATE TABLE Recruitment.Tests (
  TestID int  IDENTITY  NOT NULL,
  TestStartTime datetime  NOT NULL,
  TestEendTime datetime  NOT NULL,
  TestTittle nvarchar(50)  NOT NULL ,
  TestTypeID int  NOT NULL,
  CONSTRAINT Test_pk PRIMARY KEY  (TestID),
  CONSTRAINT Test_TestTypes FOREIGN KEY (TestTypeID) REFERENCES
Recruitment.TestTypes (TestTypeID)
);

CREATE TABLE Recruitment.InterviewTypes (
  InterviewTypeID int IDENTITY   NOT NULL,
  InterviewType nvarchar(50)  NOT NULL,
  CONSTRAINT InterviewTypes_pk PRIMARY KEY  (InterviewTypeID)
);

CREATE TABLE Recruitment.Interviews (
  InterviewID int IDENTITY  NOT NULL,
  InterviewStartTime datetime  NOT NULL,
  InterviewEndTime datetime  NOT NULL,
  InterviewLocation nvarchar(50)  NOT NULL,
  InterviewTypeID int  NOT NULL,
  TestID int  NOT NULL,
  InterviewerID int  NOT NULL,
  CONSTRAINT Interviews_pk PRIMARY KEY CLUSTERED (InterviewID),
  CONSTRAINT Interviews_Test FOREIGN KEY (TestID) REFERENCES Recruitment.Tests
(TestID),
  CONSTRAINT Interviews_InterviewTypes FOREIGN KEY (InterviewTypeID)REFERENCES
Recruitment.InterviewTypes (InterviewTypeID),
  CONSTRAINT Interviews_Interviewers FOREIGN KEY (InterviewerID) REFERENCES
Recruitment.Interviewers (InterviewerID)
);

CREATE TABLE Recruitment.Reimbursement (
  ReimbursementID int IDENTITY   NOT NULL,
  SpendTotal int  NOT NULL CHECK(SpendTotal >=0),
  CONSTRAINT Reimbursement_pk PRIMARY KEY  (ReimbursementID)
);

CREATE TABLE  Recruitment.AirlineReservations (
  AirlineReservationID int IDENTITY   NOT NULL,
  AirlineName nvarchar(50)  NOT NULL,
  AirlineTrackNumber nvarchar(50)  NOT NULL,
  AirLineCost int  NOT NULL CHECK(AirLineCost >=0),
  ReimbursementID int  NOT NULL,
  CONSTRAINT  AirlineReservations_pk PRIMARY KEY  (AirlineReservationID),
  CONSTRAINT    AirlineReservations_Reimbursement FOREIGN KEY (ReimbursementID)
REFERENCES Recruitment.Reimbursement (ReimbursementID)
);
CREATE TABLE Recruitment.HotelReservations (
  HotelReservationID  int IDENTITY    NOT NULL,
  HotelName nvarchar(50)  NOT NULL,
  HotelTrackNumber nvarchar(50)  NOT NULL,
  HotelCost int  NOT NULL CHECK(HotelCost >=0),
```

```
  ReimbursementID int  NOT NULL,
  CONSTRAINT HotelReservations_pk PRIMARY KEY  (HotelReservationID ),
  CONSTRAINT  HotelReservations_Reimbursement  FOREIGN  KEY  (ReimbursementID)
REFERENCES Recruitment.Reimbursement (ReimbursementID)
);

CREATE TABLE Recruitment.CarRentals (
  CarRentalID int IDENTITY    NOT NULL,
  CarRentalName nvarchar(50)  NOT NULL,
  CarRentalTrackNumber nvarchar(50)  NOT NULL,
  CarRentCost int  NOT NULL CHECK(CarRentCost >=0),
  ReimbursementID int  NOT NULL,
  CONSTRAINT CarRentals_pk PRIMARY KEY  (CarRentalID),
  CONSTRAINT  CarRentals_Reimbursement  FOREIGN  KEY
(ReimbursementID)REFERENCES Recruitment.Reimbursement (ReimbursementID)
);



CREATE TABLE Recruitment.OnCallType (
  OncallTypeID int IDENTITY  NOT NULL,
  OncallReason nvarchar(50)  NOT NULL,
  CONSTRAINT OnCallType_pk PRIMARY KEY  (OncallTypeID)
);

CREATE TABLE Recruitment.RejectType (
  RejectTypeID int IDENTITY  NOT NULL,
  RejectType nvarchar(50)  NOT NULL,
  CONSTRAINT RejectType_pk PRIMARY KEY  (RejectTypeID)
);

CREATE TABLE Recruitment.Statues (
  StatueID int IDENTITY NOT NULL,
  StatueName nvarchar(50)  NOT NULL,
  CONSTRAINT Statues_pk PRIMARY KEY  (StatueID)
);

CREATE TABLE Recruitment.Application (
  ApplicationID int IDENTITY NOT NULL,
  OpeningJobsID  int  NOT NULL,
  StatueID int  NOT NULL,
  CandidateID  int  NOT NULL,
  ReimbursementID int  NOT NULL,
  CONSTRAINT Application_pk PRIMARY KEY CLUSTERED (ApplicationID),
  CONSTRAINT  Application_OpeningJobs  FOREIGN  KEY  (OpeningJobsID  )  REFERENCES
Recruitment.OpeningJobs (OpeningJobsID ),
  CONSTRAINT  Application_Statues  FOREIGN  KEY  (StatueID)  REFERENCES
Recruitment.Statues (StatueID),
  CONSTRAINT  Application_Candidates  FOREIGN  KEY  (CandidateID  )REFERENCES
Recruitment.Candidates (CandidateID ),
  CONSTRAINT  Application_Reimbursement  FOREIGN  KEY  (ReimbursementID)
REFERENCES Recruitment.Reimbursement (ReimbursementID)

);

CREATE TABLE Recruitment.OnCallList (
  OncallCode int IDENTITY  NOT NULL,
```

```
  OncallTypeID int  NOT NULL,
  ApplicationID int  NOT NULL,
  CONSTRAINT OnCallList_pk PRIMARY KEY  (OncallCode),
  CONSTRAINT OnCall_OnCallType FOREIGN KEY (OncallTypeID)REFERENCES
Recruitment.OnCallType (OncallTypeID),
  CONSTRAINT OnCallList_Application FOREIGN KEY (ApplicationID) REFERENCES
Recruitment.Application (ApplicationID)
);

CREATE TABLE Recruitment.Onboarding (
  OnboardingID int IDENTITY  NOT NULL,
  ApplicationID int  NOT NULL,
  EducationCheck nvarchar(50)  NOT NULL,
  Legitimacy nvarchar(50)  NOT NULL,
  CONSTRAINT Onboarding_pk PRIMARY KEY  (OnboardingID),
  CONSTRAINT Onboarding_Application FOREIGN KEY (ApplicationID) REFERENCES
Recruitment.Application (ApplicationID)
);


CREATE TABLE Recruitment.RejectList (
  RejectCode int IDENTITY  NOT NULL,
  ApplicationID int  NOT NULL,
  RejectTypeID int  NOT NULL,
  CONSTRAINT RejectList_pk PRIMARY KEY  (RejectCode),
  CONSTRAINT RejectList_RejectType FOREIGN KEY (RejectTypeID) REFERENCES
Recruitment.RejectType (RejectTypeID),
  CONSTRAINT RejectList_Application FOREIGN KEY (ApplicationID) REFERENCES
Recruitment.Application (ApplicationID)
  )


CREATE TABLE Recruitment.BlackListed (
  BlackListedID int IDENTITY NOT NULL,
  ApplicationID int  NOT NULL,
  CONSTRAINT BlackListed_pk PRIMARY KEY  (BlackListedID),
  CONSTRAINT BlackListed_Application FOREIGN KEY (ApplicationID) REFERENCES
Recruitment.Application (ApplicationID)
);


CREATE TABLE Recruitment.Employee (
  EmployeeID int IDENTITY NOT NULL,
  ApplicationID int  NOT NULL,
  CONSTRAINT Employee_pk PRIMARY KEY  (EmployeeID),
  CONSTRAINT Employee_Application FOREIGN KEY (ApplicationID) REFERENCES
Recruitment.Application (ApplicationID)
);

CREATE TABLE Recruitment.ComplaintHandling (
  ComplaintID int IDENTITY NOT NULL,
  ApplicationID int  NOT NULL,
  ComplainDesc nvarchar(2000)  NOT NULL,
  CONSTRAINT ComplaintHandling_pk PRIMARY KEY  (ComplaintID),
  CONSTRAINT ComplaintHandling_Application FOREIGN KEY (ApplicationID)
REFERENCES Recruitment.Application (ApplicationID)
);
```

```
CREATE TABLE Recruitment.Evaluation (
  EvaluationID int IDENTITY NOT NULL,
  InterviewID int  NOT NULL,
  ApplicationID int  NOT NULL,
  GRADE int  NOT NULL CHECK(GRADE>=0 and GRADE<=100) ,
  CONSTRAINT Evaluation_pk PRIMARY KEY CLUSTERED (EvaluationID),
  CONSTRAINT Evaluation_Application FOREIGN KEY (ApplicationID) REFERENCES
Recruitment.Application (ApplicationID),
  CONSTRAINT Evaluation_Interviews FOREIGN KEY (InterviewID) REFERENCES
Recruitment.Interviews (InterviewID)
);
```

```
CREATE TABLE Recruitment.Review (
  ReviewID int  IDENTITY NOT NULL,
  EvaluationID int  NOT NULL,
  CandReview nvarchar(2000)   NULL,
  InterReview nvarchar(2000)   NULL,
  CONSTRAINT Review_pk PRIMARY KEY  (ReviewID),
  CONSTRAINT Review_Evaluation FOREIGN KEY (EvaluationID) REFERENCES
Recruitment.Evaluation (EvaluationID)
);
```

Insert Test data

All test data is based on the scenario described above, and to insert the primary key, the DENTITY_INSERT method is used.

Screen shots are in the **Appendix 1.2 screenshots**

SQL for Insert Test data:

```
USE HR;

SET IDENTITY_INSERT  Recruitment.Candidates ON
INSERT INTO Recruitment.Candidates(CandidateID,CandidateFName,CandidateLName)
VALUES
(1,'Wang','Xiao'),
(2,'Wang','YuChen'),
(3,'He','XiangYu'),
(4,'Krishna','Jash'),
(5,'Robbin','Rosan'),
(6,'John','Cena'),
(7,'Tianlu','Jacoob'),
(8,'MengNan','Hong'),
(9,'XiaoXiao','Peng'),
(10,'MengYing','Wang')

SET IDENTITY_INSERT Recruitment.Candidates OFF

SET IDENTITY_INSERT  Recruitment.CandidatesInfo ON
INSERT INTO
Recruitment.CandidatesInfo(CandidateInfoID,Email,Phone,ShortProfile,Citizenship,Education,Lo
cationInfo,CandidateID) VALUES
(1,'xwang99@syr.edu','375-88689','C++ Expert & OS design','CHN','Master','NY',1),
(2,'yuchen@syr.edu','375-88690','Java Expert','CHN','Master','NY',2),
(3,'xiangyu@syr.edu','375-88691','C++ Expert','CHN','Master','NY',3),
(4,'Donglu@bufflo.edu','375-88692','AI& DeepLearning','CHN','Doctor','NY',4),
(5,'Krishna@huawei.edu','375-88693','C Expert','IND','HighSchool','MASS',5),
(6,'John@gmail.com','375-88694','C++ Expert','USA','UnderGraduate','NY',6),
(7,'tianlu@bentley.edu','375-89689','C++ Expert','USA','UnderGraduate','NY',7),
(8,'scarlletHong@bentley.edu','375-88989','C++ Expert','CHN','UnderGraduate','NY',8),
(9,'xiaoxiao@syr.edu','375-88289','C++ Expert','CHN','Master','CA',9),
(10,'mengying@syr.edu','375-81689','C++ Expert','CHN','Master','CA',10)
SET IDENTITY_INSERT Recruitment.CandidatesInfo OFF

SET IDENTITY_INSERT  Recruitment.CandidatesDocuments ON
INSERT INTO
Recruitment.CandidatesDocuments(CandidateDocumentID,CV,ReferenceLetter,CoverLetter,Cand
idateID) VALUES
(1,'url//resume_abcdefghijklmn123456781','url//RefenceLetter_abcdefghijklmn123456780','url//co
verletter_abcdefghijklmn123456780',1),
(2,'url//resume_abcdefghijklmn123456782','url//RefenceLetter_abcdefghijklmn123456781','url//co
verletter_abcdefghijklmn123456781',2),
```

```
(3,'url//resume_abcdefghijklmn123456783','url//RefenceLetter_abcdefghijklmn123456782','url//co
verletter_abcdefghijklmn123456782',3),
(4,'url//resume_abcdefghijklmn123456784','url//RefenceLetter_abcdefghijklmn123456783','url//co
verletter_abcdefghijklmn123456783',4),
(5,'url//resume_abcdefghijklmn123456785','url//RefenceLetter_abcdefghijklmn123456784','url//co
verletter_abcdefghijklmn123456784',5),
(6,'url//resume_abcdefghijklmn123456786','url//RefenceLetter_abcdefghijklmn123456785','url//co
verletter_abcdefghijklmn123456785',6),
(7,'url//resume_abcdefghijklmn123456787','url//RefenceLetter_abcdefghijklmn123456786','url//co
verletter_abcdefghijklmn123456786',7),
(8,'url//resume_abcdefghijklmn123456788','url//RefenceLetter_abcdefghijklmn123456787','url//co
verletter_abcdefghijklmn123456787',8),
(9,'url//resume_abcdefghijklmn123456789','url//RefenceLetter_abcdefghijklmn123456788','url//co
verletter_abcdefghijklmn123456788',9),
(10,'url//resume_abcdefghijklmn123456780','url//RefenceLetter_abcdefghijklmn12345679','url//co
verletter_abcdefghijklmn123456789',10)
SET IDENTITY_INSERT Recruitment.CandidatesDocuments OFF

SET IDENTITY_INSERT  Recruitment.JobTypes ON
INSERT INTO Recruitment.JobTypes(JobTypeID,JobType,JobTypeDescription) VALUES
(1,'Summer Internship','Candidate works for 2- 3 months as intern'),
(2,'Full Time ','Candidate must work at least 9 hours per day'),
(3,'Part Time','Candidate must work at most 3 hours per day'),
(4,'Contract','Candidate sign with the company and get paid based on workload')
SET IDENTITY_INSERT Recruitment.JobTypes OFF

SET IDENTITY_INSERT  Recruitment.JobCategories ON
INSERT  INTO  Recruitment.JobCategories(JobCategoryID,JobCategory,JobCategoryDescription)
VALUES
(1,'IT','installing, maintaining and repairing hardware & software components of the organization
computers'),
(2,'Software Design ','Transfer customer requirements into services'),
(3,'Software Test','responsible for the quality of software development and deployment')
SET IDENTITY_INSERT Recruitment.JobCategories OFF

SET IDENTITY_INSERT  Recruitment.JobMediums  ON
INSERT  INTO  Recruitment.JobMediums  (JobMediumID,JobMedium,JobMediumDescription)
VALUES
(1,'Online','Work at home'),
(2,'Onsite','Work at office')
SET IDENTITY_INSERT Recruitment.JobMediums OFF

SET IDENTITY_INSERT  Recruitment.JobPositions ON
INSERT  INTO  Recruitment.JobPositions  (JobPositionID,JobPosition,JobMediumDescription)
VALUES
(1,'Manager','10 years+ working experience'),
(2,'Senior','3 years+ working experience'),
(3,'Entry-level ','0-3 years working experience')
SET IDENTITY_INSERT Recruitment.JobPositions OFF

SET IDENTITY_INSERT  Recruitment.JobOpeningPlatforms ON
INSERT  INTO  Recruitment.JobOpeningPlatforms  (JobOpeningPlatformID,PlatformName)
VALUES
(1,'Company Webpage'),
(2,'LinkedIn'),
(3,'HandShake')
SET IDENTITY_INSERT Recruitment.JobOpeningPlatforms OFF
```

```
SET IDENTITY_INSERT  Recruitment.Jobs ON
INSERT INTO Recruitment.Jobs
(JobCode,JobName,JobDescription,JobTypeID,JobCategoryID,JobMediumID,JobPositionID)
VALUES
(1,'FullStack designer','Use c++ maintain the OS system and design websit',2,2,2,2),
(2,'Backend designer','Use x-86 architecture improve performance of the software',2,2,2,2),
(3,'Softer-ware test','Test softerware and design test cases',4,3,1,3),
(4,'IT Guys','Do some trivial works',1,1,2,3)
SET IDENTITY_INSERT Recruitment.Jobs OFF

SET IDENTITY_INSERT  Recruitment.OpeningJobs ON
INSERT INTO Recruitment.OpeningJobs
(OpeningJobsID,StartDate,NumPosition,JobCode,JobJobOpeningPlatformID) VALUES
(1,'2023-1-1',2,1,1),
(2,'2023-1-1',2,2,1),
(3,'2023-1-1',1,3,2),
(4,'2023-1-1',1,4,3)

SET IDENTITY_INSERT Recruitment.OpeningJobs OFF

SET IDENTITY_INSERT  Recruitment.Interviewers ON
INSERT INTO Recruitment.Interviewers (InterviewerID,InviewerFname,InviewerLname,Titles)
VALUES
(1,'Wang','Pipi','Chief Technology Officer'),
(2,'Ehat ','Ercanli','Technical Consulting'),
(3,'Lee','Robin','Chief Engineer')

SET IDENTITY_INSERT Recruitment.Interviewers  OFF

SET IDENTITY_INSERT  Recruitment.TestTypes ON
INSERT INTO Recruitment.TestTypes (TestTypeID ,TestType) VALUES
(1,'Onsite'),
(2,'Online')
SET IDENTITY_INSERT Recruitment.TestTypes  OFF

SET IDENTITY_INSERT  Recruitment.Tests ON
INSERT INTO Recruitment.Tests (TestID ,TestStartTime,TestEendTime,TestTittle,TestTypeID)
VALUES
(1,'2022-11-15 09:00:00','2022-11-15 11:00:00','Programming1',1),
(2,'2022-11-15 14:00:00','2022-11-15 16:00:00','SystemDesign1',1),
(3,'2022-12-15 09:00:00','2022-12-15 11:00:00','Programming2',1),
(4,'2022-12-15 14:00:00','2022-12-15 16:00:00','SystemDesign2',1)
SET IDENTITY_INSERT Recruitment.Tests  OFF

SET IDENTITY_INSERT  Recruitment.InterviewTypes  ON
INSERT INTO Recruitment.InterviewTypes  (InterviewTypeID ,InterviewType) VALUES
(1,'Onsite'),
(2,'Online')
SET IDENTITY_INSERT  Recruitment.InterviewTypes   OFF

SET IDENTITY_INSERT  Recruitment.Interviews  ON
INSERT INTO Recruitment.Interviews
(InterviewID ,InterviewStartTime,InterviewEndTime,InterviewLocation,InterviewTypeID,TestID,
InterviewerID) VALUES
(1,'2022-11-15','2022-11-16','NY',1,1,1),
(2,'2022-11-15','2022-11-16','NY',1,1,2),
```

```
(3,'2022-11-15','2022-11-16','NY',1,1,3),
(4,'2022-11-15','2022-11-16','NY',1,2,1),
(5,'2022-11-15','2022-11-16','NY',1,2,2),
(6,'2022-11-15','2022-11-16','NY',1,2,3),
(7,'2022-12-15','2022-12-16','NY',1,3,1),
(8,'2022-12-15','2022-12-16','NY',1,3,2),
(9,'2022-12-15','2022-12-16','NY',1,3,3),
(10,'2022-12-15','2022-12-16','NY',1,4,1),
(11,'2022-12-15','2022-12-16','NY',1,4,2),
(12,'2022-12-15','2022-12-16','NY',1,4,3)
SET IDENTITY_INSERT  Recruitment.Interviews   OFF

SET IDENTITY_INSERT  Recruitment.Reimbursement ON
INSERT INTO Recruitment.Reimbursement (ReimbursementID,SpendTotal) VALUES
(1,0),
(2,0),
(3,0),
(4,0),
(5,0),
(6,0),
(7,0),
(8,0),
(9,700),
(10,9348)

SET IDENTITY_INSERT Recruitment.Reimbursement  OFF



SET IDENTITY_INSERT  Recruitment.AirlineReservations ON
INSERT INTO Recruitment.AirlineReservations
(AirlineReservationID,AirlineName,AirlineTrackNumber,AirLineCost,ReimbursementID)
VALUES
(1,'JetBlue','TN359462YH',300,9),
(2,'Delta','TN359432YH',330,10),
(3,'UltraLuxury','TN358888YH',1888,10)
SET IDENTITY_INSERT Recruitment.AirlineReservations  OFF

SET IDENTITY_INSERT  Recruitment.HotelReservations ON
INSERT INTO Recruitment.HotelReservations
(HotelReservationID,HotelName,HotelTrackNumber,HotelCost,ReimbursementID) VALUES
(1,'Hilton','HT359462YH',300,9),
(2,'Hilton','HT359432YH',310,10),
(3,'The Ranch at Rock Creek','HT358888YH',4700,10)
SET IDENTITY_INSERT Recruitment.HotelReservations  OFF

SET IDENTITY_INSERT  Recruitment.CarRentals ON
INSERT INTO Recruitment.CarRentals
(CarRentalID,CarRentalName,CarRentalTrackNumber,CarRentCost,ReimbursementID) VALUES
(1,'Toyota','CAR359462YH',100,9),
(2,'Toyota','CAR359432YH',120,10),
(3,'Benltley','CAR358888YH',2000,10)
SET IDENTITY_INSERT Recruitment.CarRentals  OFF

SET IDENTITY_INSERT  Recruitment.OnCallType ON
INSERT INTO Recruitment.OnCallType  (OncallTypeID,OncallReason) VALUES
(1,'Select but no more job openings '),
```

```
(2,'Onboarding  is unsuccessful'),
(3,'Onboarding  is unsuccessful')
SET IDENTITY_INSERT Recruitment.OnCallType  OFF

SET IDENTITY_INSERT  Recruitment.RejectType ON
INSERT INTO Recruitment.RejectType  (RejectTypeID,RejectType) VALUES
(1,'SApplication scanning is not met '),
(2,'Fail in the interview')
SET IDENTITY_INSERT Recruitment.RejectType  OFF

SET IDENTITY_INSERT  Recruitment.Statues ON
INSERT INTO Recruitment.Statues  (StatueID,StatueName) VALUES
(1,'Submitted'),
(2,'Reject'),
(3,'Select'),
(4,'Waiting'),
(5,'Re-interview'),
(6,'OnCall'),
(7,'Negotiating'),
(8,'Accept'),
(9,'Onboarding'),
(10,'BlackListed'),
(11,'Employee'),
(12,'NextRound')
SET IDENTITY_INSERT Recruitment.Statues  OFF


SET IDENTITY_INSERT  Recruitment.Application ON
INSERT INTO Recruitment.Application
(ApplicationID,OpeningJobsID,StatueID,CandidateID,ReimbursementID) VALUES
(1,1,11,1,1),
(2,1,11,2,2),
(3,1,6,3,3),
(4,2,2,4,4),
(5,2,2,5,5),
(6,2,2,6,6),
(7,2,6,7,7),
(8,2,6,8,8),
(9,2,11,9,9),
(10,2,10,10,10)

SET IDENTITY_INSERT Recruitment.Application  OFF

SET IDENTITY_INSERT  Recruitment.BlackListed ON
INSERT INTO Recruitment.BlackListed  (BlackListedID,ApplicationID) VALUES
(1,1)
SET IDENTITY_INSERT Recruitment.BlackListed  OFF

SET IDENTITY_INSERT  Recruitment.OnCallList ON
INSERT INTO Recruitment.OnCallList  (OncallCode,OncallTypeID,ApplicationID) VALUES
(1,1,3),
(2,2,7),
(3,3,8)
SET IDENTITY_INSERT Recruitment.OnCallList  OFF

SET IDENTITY_INSERT  Recruitment.Onboarding ON
INSERT INTO Recruitment.Onboarding
```

```
(OnboardingID,ApplicationID,EducationCheck,Legitimacy) VALUES
(1,1,'PASS','PASS'),
(2,2,'PASS','PASS'),
(3,9,'PASS','PASS')
SET IDENTITY_INSERT Recruitment.Onboarding  OFF


SET IDENTITY_INSERT  Recruitment.RejectList ON
INSERT INTO Recruitment.RejectList  (RejectCode,ApplicationID,RejectTypeID) VALUES
(1,4,2),
(2,5,1),
(3,6,2)
SET IDENTITY_INSERT Recruitment.RejectList  OFF

SET IDENTITY_INSERT  Recruitment.Employee ON
INSERT INTO Recruitment.Employee  (EmployeeID,ApplicationID) VALUES
(1,1),
(2,2),
(3,9)
SET IDENTITY_INSERT Recruitment.Employee  OFF


SET IDENTITY_INSERT  Recruitment.ComplaintHandling ON
INSERT  INTO  Recruitment.ComplaintHandling  (ComplaintID,ApplicationID,ComplainDesc)
VALUES
(1,6,'This test is so hard that I think the company is making things difficult.')
SET IDENTITY_INSERT Recruitment.ComplaintHandling  OFF

SET IDENTITY_INSERT  Recruitment.Evaluation ON
INSERT  INTO  Recruitment.Evaluation  (EvaluationID,InterviewID,ApplicationID,GRADE)
VALUES
(1,1,1,100),
(2,2,1,100),
(3,3,1,100),
(4,4,1,100),
(5,5,1,100),
(6,6,1,100),
(7,1,2,70),
(8,2,2,70),
(9,3,2,80),
(10,4,2,70),
(11,5,2,70),
(12,6,2,80),
(13,7,2,100),
(14,8,2,100),
(15,9,2,100),
(16,10,2,100),
(17,11,2,100),
(18,12,2,100),
(19,1,3,100),
(20,2,3,100),
(21,3,3,100),
(22,4,3,100),
(23,5,3,100),
(24,6,3,100),
(25,1,4,30),
(26,2,4,25),
```

```
(27,3,4,40),
(28,4,4,0),
(29,5,4,0),
(30,6,4,0),
(31,1,6,45),
(32,2,6,55),
(33,3,6,60),
(34,4,6,45),
(35,5,6,30),
(36,6,6,27),
(37,1,7,100),
(38,2,7,98),
(39,3,7,100),
(40,4,7,100),
(41,5,7,98),
(42,6,7,100),
(43,1,8,80),
(44,2,8,98),
(45,3,8,90),
(46,4,8,100),
(47,5,8,100),
(48,6,8,100),
(49,1,9,100),
(50,2,9,98),
(51,3,9,100),
(52,4,9,100),
(53,5,9,100),
(54,6,9,100),
(55,1,10,80),
(56,2,10,70),
(57,3,10,80),
(58,4,10,80),
(59,5,10,70),
(60,6,10,80),
(61,7,10,100),
(62,8,10,70),
(63,9,10,80),
(64,10,10,100),
(65,11,10,70),
(66,12,10,80)
SET IDENTITY_INSERT Recruitment.Evaluation  OFF

SET IDENTITY_INSERT  Recruitment.Review ON
INSERT INTO Recruitment.Review (ReviewID,EvaluationID,CandReview,InterReview)
VALUES
(1,55,'I could not believe that the examiner said my breasts stood beautiful and that if I could have a one-night stand with him, he could pass me straight away.','Strong applicant!'),
(2,61,'There is no one in your company to deal with this matter, and there has no handling mechanism. Just wait for my fireback!!!','Very Strong applicant!')
SET IDENTITY_INSERT Recruitment.Review  OFF
```

Database Test & Performance Improvement

Introduction

Next, I will use views, stored procedures, user-defined functions, triggers, transactions and scripts to create users with various security levels, passwords, and roles. To improve the performance and user-friendly of the database, make the database more convenient and more secure.
These will also test the correctness of the database to see if the output results are as expected.

Views

Just as a function (in programming) can provide abstraction, so can a database view. Views can represent a subset of the data contained in a table. There are some merits of views:
1.Views can join and simplify multiple tables into a single virtual table.
2.Views can act as aggregated tables.
3.Views can hide the complexity of data.
4.Views take very little space to store.

Views1

Propose:

During the recruitment process, we need to get the results of each round of interviews to determine the applicant's state: selected, selected for the next interview, or rejected. These decisions are all based on interview results. Therefore, I design a view **`FinalGradeInterview1`** to summarize each applicant's test result, supporting further decisions.

`Query:`

```
USE HR;
GO
--If EXIST DROP
IF OBJECT_ID('FinalGradeInterview1') IS NOT NULL
DROP VIEW FinalGradeInterview1;
GO

CREATE VIEW FinalGradeInterview1 AS
SELECT
Recruitment.Evaluation.ApplicationID,Recruitment.Tests.TestTittle,AVG(Recruitme
nt.Evaluation.GRADE)AS [FinalGradeInterview1]
--Use average grade of three grader as the final grade
FROM  Recruitment.Evaluation JOIN Recruitment.Interviews ON
Recruitment.Evaluation.InterviewID=Recruitment.Interviews.InterviewID
             JOIN Recruitment.Tests ON
Recruitment.Interviews.TestID=Recruitment.Tests.TestID
WHERE Recruitment.Tests.TestTittle='Programming1' or
Recruitment.Tests.TestTittle='SystemDesign1'--2 TESTS in interview1
GROUP BY  Recruitment.Evaluation.ApplicationID, Recruitment.Tests.TestTittle
GO

SELECT*
FROM FinalGradeInterview1;
```

Output:

The screenshot is in **appendix 2.1.1**, and the output is correct, which shows the applicant selects by the first interview has a high score.

Views2

Propose:

During the recruitment process, we need to get the results of 2- round of interview to determine the applicant's .
These decisions are all based on interview results. Therefore, I design a view **`FinalGradeInterview2`** to summarize each applicant's test result, supporting further decisions.

`Query`:

```
USE HR;
GO
IF OBJECT_ID('FinalGradeInterview2') IS NOT NULL
DROP VIEW FinalGradeInterview2;
GO

CREATE VIEW FinalGradeInterview2 AS
SELECT
Recruitment.Evaluation.ApplicationID,Recruitment.Tests.TestTittle,AVG(Recruitme
nt.Evaluation.GRADE)AS [FinalGradeInterview2]
--Use average grade of three grader as the final grade
FROM  Recruitment.Evaluation JOIN Recruitment.Interviews ON
Recruitment.Evaluation.InterviewID=Recruitment.Interviews.InterviewID
             JOIN Recruitment.Tests ON
Recruitment.Interviews.TestID=Recruitment.Tests.TestID
WHERE Recruitment.Tests.TestTittle='Programming2' or
Recruitment.Tests.TestTittle='SystemDesign2'--2 TESTS in interview1
GROUP BY  Recruitment.Evaluation.ApplicationID, Recruitment.Tests.TestTittle
GO
--TEST
SELECT*
FROM FinalGradeInterview2;
```

Output:

The screenshot is in **appendix 2.1.2**, and the output is correct, which shows the applicant selects by the second interview has a high score.

Views3

Propose:

For different jobs, we have different requirements for applicants. For example, some jobs require strong programming skills, and some have high system design requirements. So we create a view to linking each applicant's corresponding job application with his interview grade for future decisions.

Query:

```
USE HR;
GO
--If EXIST DROP
IF OBJECT_ID('JobTestScore') IS NOT NULL
DROP VIEW JobTestScore ;
GO
-- query logic:
-- read my E/R diag, from the diag, link the job-openingjobs to the evaluation
from table to table linked by pk-fk
--
Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate,TestTittle
are can be grouped together present the relationship we want to explore.
-- use avg present the final grade for each test.
CREATE VIEW JobTestScore AS
SELECT Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate
AS[JobStartDate],TestTittle, AVG(GRADE) AS [TestResult]
FROM  Recruitment.Application JOIN Recruitment.OpeningJobs ON
Recruitment.Application.OpeningJobsID=Recruitment.OpeningJobs.OpeningJobsID
      JOIN Recruitment.Jobs ON  Recruitment.Jobs.JobCode=
Recruitment.OpeningJobs.JobCode
         JOIN  Recruitment.Evaluation ON
Recruitment.Evaluation.ApplicationID=Recruitment.Application.ApplicationID
         JOIN Recruitment.Interviews ON
Recruitment.Evaluation.InterviewID=Recruitment.Interviews.InterviewID
         JOIN Recruitment.Tests ON
Recruitment.Interviews.TestID=Recruitment.Tests.TestID
GROUP BY
Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate,TestTittle
GO

SELECT *
FROM JobTestScore
```

Output:

The screenshot is in **appendix 2.1.3**, and the output is correct (meet our scenario design).

Views4

Propose:

Reimbursement for job fairs is a considerable expense. In our design, we can reimburse that 1. the candidate’ s location differs from the interview location, and 2. the interview is onsite.
This view **EligbleReimbursement** is to find applicants who meet these qualifications for future reimbursement.

Query:

```
USE HR;
GO

IF OBJECT_ID('EligbleReimbursement') IS NOT NULL
DROP VIEW EligbleReimbursement;
GO
-- query logic:
--1. location of candidate cannot near to the interview
--2  the interview must be onesite interview
-- read my E/R diag, from the diag, link the candidateInfo to the interview
Type from table to table linked by pk-fk

CREATE VIEW EligbleReimbursement AS
SELECT distinct Recruitment.Application.ApplicationID
FROM
Recruitment.Application JOIN Recruitment.Candidates ON
Recruitment.Candidates.CandidateID=Recruitment.Application.CandidateID
JOIN Recruitment.CandidatesInfo ON
Recruitment.Candidates.CandidateID=Recruitment.CandidatesInfo.CandidateID
JOIN Recruitment.Evaluation ON
Recruitment.Evaluation.ApplicationID=Recruitment.Application.ApplicationID
JOIN Recruitment.Interviews ON
Recruitment.Evaluation.InterviewID=Recruitment.Interviews.InterviewID
JOIN Recruitment.InterviewTypes ON
Recruitment.Interviews.InterviewTypeID=Recruitment.InterviewTypes.InterviewType
ID
WHERE
Recruitment.CandidatesInfo.LocationInfo<>Recruitment.Interviews.InterviewLocati
on AND InterviewType='Onsite'
GO

SELECT*
FROM EligbleReimbursement;
```

Output:

The screenshot is in **appendix 2.1.4**, and the output is correct, which is 2 candidates comes from ‘CA’ for the interview hold in ‘NY’.

User Defined Functions

A user-defined function (UDF) lets you create a function by using a SQL expression. A UDF accepts input columns, performs actions on the input, and returns the result as a value. You can define a UDFs as either persistent or temporary.

User-defined functions help decompose an extensive program into small segments, making the program easy to understand, maintain and debug. Suppose repeated code occurs in a program. The function can include those codes and execute them when needed by calling that function.

Function1

Propose:
In our interview, we need to determine what grades we can select, rejects, and the second round of interviews. In this process, I want to reduce human intervention. Therefore, when we set the bar, the function automatically gives the applicant status. This can not only reduce the workload but also avoid human intervention.

Query:

```
USE HR
GO

IF OBJECT_ID('fnInterviewsEvaluation') IS NOT NULL--if eixit drop
DROP FUNCTION fnInterviewsEvaluation
GO


CREATE FUNCTION fnInterviewsEvaluation
(@HighBar INT,@LowBar INT) -- two inputs , if below the lowbar 'Reject'
-- higher than the highbar 'select'
-- otherwise next round of interview
RETURNS TABLE --return a table

RETURN
(
	SELECT ApplicationID, Decision=
	CASE
	WHEN AVG(FinalGradeInterview1)>@HighBar THEN 'Select' -- aggerate
functions to evluate the solution
	WHEN AVG(FinalGradeInterview1)<@LowBar THEN 'Reject'
	ELSE 'NextRound'
	END
	From dbo.FinalGradeInterview1
	GROUP BY ApplicationID -- group by each applicant
)

GO

SELECT*
FROM fnInterviewsEvaluation(80,60)
```

Output: In appendix 2.2.1, the output match our designed scenario, so it is correct.

Function2

Propose:
In our final interview, we need to only determine what grades we can select and rejects. In this process, I want to reduce human intervention. Therefore, when we set the bar, the function automatically gives the applicant status. This can not only reduce the workload but also avoid human intervention.

Query:

```
IF OBJECT_ID('fnFinalInterviewsEvaluation') IS NOT NULL--if eixit drop
DROP FUNCTION fnFinalInterviewsEvaluation
GO


CREATE FUNCTION fnFinalInterviewsEvaluation
(@Bar INT) -- one inputs , if below the bar 'Reject'  higher than the bar
'select'
RETURNS TABLE --return a table

RETURN
(
	SELECT ApplicationID, Decision=
	CASE
	WHEN AVG(FinalGradeInterview2)>@Bar THEN 'Select' -- aggerate functions
to evluate the solution
	ELSE 'Reject'
	END
	From dbo.FinalGradeInterview2
	GROUP BY ApplicationID -- group by each applicant
)

GO
--TEST
SELECT*
FROM fnFinalInterviewsEvaluation(80)
```

Output:

In appendix 2.2.2, the output match our designed scenario, so it is correct.

Function3

Propose:
In many cases, we need to study applicants' grades for different jobs, so we can set different bars to admitting different job applicants.

Query:

```
USE HR
GO

IF OBJECT_ID('JobTest') IS NOT NULL--if eixit drop
DROP FUNCTION JobTest
GO


CREATE FUNCTION JobTest
(@jobcode int)
RETURNS TABLE

RETURN
(
 --query logic:
-- read my E/R diag, from the diag, link the job-openingjobs to the evaluation
from table to table linked by pk-fk
--
Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate,TestTittle
are can be grouped together present the relationship we want to explore.
-- use avg present the final grade for each test.
       SELECT
Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate
AS[JobStartDate],TestTittle, AVG(GRADE) AS [TestResult]
       FROM  Recruitment.Application JOIN Recruitment.OpeningJobs ON
Recruitment.Application.OpeningJobsID=Recruitment.OpeningJobs.OpeningJobsID
      JOIN Recruitment.Jobs ON  Recruitment.Jobs.JobCode=
Recruitment.OpeningJobs.JobCode
         JOIN  Recruitment.Evaluation ON
Recruitment.Evaluation.ApplicationID=Recruitment.Application.ApplicationID
         JOIN Recruitment.Interviews ON
Recruitment.Evaluation.InterviewID=Recruitment.Interviews.InterviewID
         JOIN Recruitment.Tests ON
Recruitment.Interviews.TestID=Recruitment.Tests.TestID
       WHERE Recruitment.Jobs.JobCode=@jobcode --only select the input jobcode
       GROUP BY
Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate,TestTittle

)

GO

SELECT*
FROM JobTest (1)
```

Output: In appendix 2.2.3, the output match our designed scenario, so it is correct.

Function4

Propose:
Find the applicant with the highest score under different test.

Query:

```
USE HR
GO

IF OBJECT_ID('fnBestApplicant') IS NOT NULL--if eixit drop
DROP FUNCTION fnBestApplicant
GO


CREATE FUNCTION fnBestApplicant
(@TESTNAME nvarchar(50))
RETURNS TABLE
-- query logic:
-- read my E/R diag, from the diag, link the job-openingjobs to the evaluation
from table to table linked by pk-fk
--
Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate,TestTittle
are can be grouped together present the relationship we want to explore.
-- use avg present the final grade for each test.
RETURN
(
SELECT TOP 1 Application.ApplicationID,TestTittle, AVG(GRADE) AS [TestResult]--
find the best
FROM  Recruitment.Application JOIN Recruitment.OpeningJobs ON
Recruitment.Application.OpeningJobsID=Recruitment.OpeningJobs.OpeningJobsID
      JOIN Recruitment.Jobs ON  Recruitment.Jobs.JobCode=
Recruitment.OpeningJobs.JobCode
         JOIN  Recruitment.Evaluation ON
Recruitment.Evaluation.ApplicationID=Recruitment.Application.ApplicationID
         JOIN Recruitment.Interviews ON
Recruitment.Evaluation.InterviewID=Recruitment.Interviews.InterviewID
         JOIN Recruitment.Tests ON
Recruitment.Interviews.TestID=Recruitment.Tests.TestID
WHERE TestTittle=@TESTNAME --name mactch
GROUP BY
Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate,TestTittle
ORDER BY TestResult DESC --order score from high to low
)
GO

select*
from fnBestApplicant('Programming1')
```

Output: In appendix 2.2.4, the output match our designed scenario, so it is correct.

Stored procedures

One way of saving time when writing code is with stored procedures; prepare SQL code that you can save so the code can be reused over and over again. Stored procedures are a powerful vehicle for speeding up your code and giving you gains in performance.
Stored procedures have many benefits—they can be easily modified, reusable, and can automate a task requiring multiple SQL statements. Another advantage of stored procedures is accessing or modifying data in a database.

Procedure 1

Propose:

At work, we have to review the reimbursement materials listed to determine whether they can be reimbursed. This process helps us integrate relevant reimbursement materials for all eligible reimbursements.

Query:

```
USE HR;
GO
IF OBJECT_ID('spReimbursementInformation') IS NOT NULL
DROP PROC spReimbursementInformation
GO

CREATE PROC spReimbursementInformation
AS
-- JOIN ALL TABLES CAN PROVIDE Reimbursement information (E/R)
-- ONLY SELECT THOSE APPLICANT WHO CAN Reimbursement their cost by the VIEW
EligbleReimbursement
select*
from Recruitment.Reimbursement JOIN Recruitment.HotelReservations ON
Recruitment.Reimbursement.ReimbursementID=Recruitment.HotelReservations.Reimbur
sementID
JOIN Recruitment.CarRentals ON
Recruitment.Reimbursement.ReimbursementID=Recruitment.CarRentals.ReimbursementI
D
JOIN Recruitment.AirlineReservations ON
Recruitment.AirlineReservations.ReimbursementID=Recruitment.Reimbursement.Reimb
ursementID
WHERE Recruitment.Reimbursement.ReimbursementID IN
(
       select Recruitment.Application.ReimbursementID
       from Recruitment.Application JOIN EligbleReimbursement ON
Recruitment.Application.ApplicationID=EligbleReimbursement.ApplicationID
)
GO

EXEC spReimbursementInformation
```

Output: In appendix 2.3.1, the output match our designed scenario, so it is correct.

Procedure 2
Propose:

After we have reviewed the materials, we will formulate a range, and if it is within the reimbursement range, we will approve the reimbursement.

Query:

```
USE HR;
GO
IF OBJECT_ID('spReimbursementTest') IS NOT NULL
DROP PROC spReimbursementTest
GO

CREATE PROC spReimbursementTest
	@Range int =0
AS
	IF @Range <0 --- avoid possible logical mistake
		THROW 50001,'Range must be postive interger',1;
-- When the spReimbursement is eligibe and the difference between
-- real spend (get from database)and asked Reimbursement less then the rage
-- Accept else Reject
	Select Recruitment.Reimbursement.ReimbursementID,Decision=
CASE
WHEN SpendTotal-TotalSpend<=@Range THEN 'Accept'
ELSE 'Reject'
END
from Recruitment.Reimbursement JOIN
(
	select ReimbursementID,SUM(Cost) as TotalSpend
	from
	(
		-- USE UNION function aggregate all information relate to
Reimbursement
		SELECT Recruitment.AirlineReservations.ReimbursementID,
		Recruitment.AirlineReservations.AirlineName AS[NAME],
Recruitment.AirlineReservations.AirlineTrackNumber AS[TrackNumber],
		Recruitment.AirlineReservations.AirLineCost[Cost],
Recruitment.AirlineReservations.AirlineReservationID AS[ServiceID]
		FROM Recruitment.AirlineReservations
		UNION ALL
		SELECT Recruitment.CarRentals.ReimbursementID,
		Recruitment.CarRentals.[CarRentalName] AS[NAME],
Recruitment.CarRentals.[CarRentalTrackNumber] AS[TrackNumber],
		Recruitment.CarRentals.[CarRentCost][Cost],
Recruitment.CarRentals.CarRentalID AS[ServiceID]
		FROM Recruitment.CarRentals
		UNION ALL
		SELECT  Recruitment.HotelReservations.ReimbursementID,
		Recruitment.HotelReservations.[HotelName] AS[NAME],
Recruitment.HotelReservations.[HotelTrackNumber] AS[TrackNumber],
		Recruitment.HotelReservations.[HotelCost][Cost],
Recruitment.HotelReservations.[HotelReservationID] AS[ServiceID]
		FROM Recruitment.HotelReservations
		) AS TEMP--save the result in the TEMP table
		WHERE TEMP.ReimbursementID IN--use the EligbleReimbursement view
check the Reimbursement is Eligble or not
```

```
            (
            select Recruitment.Application.ReimbursementID
            from Recruitment.Application JOIN EligbleReimbursement ON
Recruitment.Application.ApplicationID=EligbleReimbursement.ApplicationID
            )
       group by ReimbursementID
) AS TEMP2 ON TEMP2.ReimbursementID=Recruitment.Reimbursement.ReimbursementID

GO
----test

EXEC spReimbursementTest @Range =200
GO
```

Output: In appendix 2.3.2, the output match our designed scenario, so it is correct.

Procedure 3
Propose:
The applicants on the on-call list are qualified candidates who have been selected through coatings of selection. When we have job opportunities, we will notify them first. The order of our notifications is based on their test scores.

Query:

```
USE HR;
GO
IF OBJECT_ID('spOnCallOreder') IS NOT NULL
DROP PROC spOnCallOreder
GO

CREATE PROC spOnCallOreder
AS
SELECT Recruitment.OnCallList.ApplicationID,AVG(TestResult) AS [Performance] --
USE Test performance as evidence of the order
FROM Recruitment.OnCallList JOIN HR.dbo.JobTestScore --USE view JobTestScore
get peformance of onlist candidate
ON Recruitment.OnCallList.ApplicationID=HR.dbo.JobTestScore.ApplicationID
group by Recruitment.OnCallList.ApplicationID
GO

--test
EXEC spOnCallOreder
```

Output:
In appendix 2.3.3, the output match our designed scenario, so it is correct.

Procedure 4

Propose:
The applicants on the on-call list are all qualified candidates who have been selected through layers of selection. When we have job opportunities, we will notify them first. The order of our notifications is based on their test scores. At the same time, when there are vacancies in different jobs, we can find out the sequence of results when applying for this job according to different jobs.

Query:

```
USE HR;
GO
IF OBJECT_ID('spOnCallOrederJob') IS NOT NULL
DROP PROC spOnCallOrederJob
GO

CREATE PROC spOnCallOrederJob
 @JobID int =0
AS
IF(@JobID=0)
       THROW 50002,'invalid jobID',1;
SELECT
Recruitment.OnCallList.ApplicationID,HR.dbo.JobTestScore.JobCode,AVG(TestResult)
AS[Performance]
FROM Recruitment.OnCallList JOIN HR.dbo.JobTestScore --USE view JobTestScore
get peformance of onlist candidate
ON Recruitment.OnCallList.ApplicationID=HR.dbo.JobTestScore.ApplicationID
WHERE HR.dbo.JobTestScore.JobCode=@JobID --use the input id
group by Recruitment.OnCallList.ApplicationID,HR.dbo.JobTestScore.JobCode
GO

--test
EXEC spOnCallOrederJob @JobID=2
```

Output:

In appendix 2.3.4, the output match our designed scenario, so it is correct.

Triggers & Transactions

A trigger is a special type of stored procedure that automatically runs when an event occurs in the database server. DML triggers run when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view.

On the other hand, A transaction is a unit of work that is performed against a database. Transactions are units or sequences of work accomplished in a logical order, whether in a manual fashion by a user or automatically by some database program.
Transactions have the following four standard properties, usually referred to by the acronym **ACID**.

Atomicity − ensures that all operations within the work unit are completed successfully. Otherwise, the transaction is aborted at the point of failure, and all the previous operations are rolled back to their former state.
Consistency − ensures that the database properly changes states upon a successfully committed transaction.
Isolation − enables transactions to operate independently of and transparent to each other.
Durability − ensures that the result or effect of a committed transaction persists in case of a system failure.

One purpose of my design and implementation is to make the status changes due to the ongoing interview process per candidate as automatic as possible, as the design of the database can allow.

Another purpose is an integrity issue. In my design, when an applicant's status changes(e.g., an applicant is added to the table of blackList, the applicant status in the application table should also be changed, and the number of jobs in the Openjobs table should also be changed) tables that relate to this change (logical level) must be changed to the corresponding statue/value to ensure the integrity.
To solve the problem, I decide to use Trigger to monitor the DML events of some tables that can update statutes.

Then I use Transaction with the Trigger to ensure the Atomicity of these changes.
Besides, the rollback mechanism is another reason I use Transaction.
We can use ROLLBACK TRANSACTION to erase all data modifications made from the start of the Transaction. So, when one query fails (such as the number of jobs<0 due to my CHECK in the OpenJobs table), everything will hold and not change, which can protect our database integrity.

Triggers & Transactions 1

Propose:

When the applicant is added to BlackedList, the applicant status in application table change to BlackedList 10 and the number of jobs of OpeningJob Table +1.

Query:

```
USE HR;
GO

DROP TRIGGER IF EXISTS Recruitment.BlackListInsert;
GO

CREATE TRIGGER BlackListInsert
       ON Recruitment.BlackListed
       AFTER INSERT--when insert into Recruitment.BlackListed trigger run
AS
       DECLARE @ApID int; --save application ID
       SELECT @ApID=inserted.ApplicationID FROM inserted
       DECLARE @OpJobID int; ---Save OpenjobID need to update
       SELECT  @OpJobID=HR.Recruitment.Application.OpeningJobsID FROM
HR.Recruitment.Application
BEGIN TRY
       BEGIN TRAN --Tran make database consistent
       UPDATE HR.Recruitment.Application
       SET HR.Recruitment.Application.StatueID=10 --change the application
statues to 10,which means blacklisted
       WHERE HR.Recruitment.Application.ApplicationID=@ApID

       UPDATE HR.Recruitment.OpeningJobs
       SET
HR.Recruitment.OpeningJobs.NumPosition=HR.Recruitment.OpeningJobs.NumPosition+1
--change number of openjobs+1
       WHERE HR.Recruitment.OpeningJobs.OpeningJobsID=@OpJobID

       COMMIT TRAN
END TRY
BEGIN CATCH--if any one of the tran cannot work, rollback and throw the error
message
       ROLLBACK TRAN
END CATCH
GO
```

Test&Result: Correct

```
SET IDENTITY_INSERT  Recruitment.BlackListed ON
INSERT INTO Recruitment.BlackListed  (BlackListedID,ApplicationID) VALUES
(1,1)
SET IDENTITY_INSERT Recruitment.BlackListed  OFF
```

	Changes	Output
Before	Applicant.StatueID=11 OpeningJobs.NumPosition =2	Appendix 2.4.1.1
After	Applicant.StatueID=10 OpeningJobs.NumPosition =3	Appendix 2.4.1.2

Triggers & Transactions 2

Propose:

When the applicant declined the offer,(in my design declined statue is in the oncallList and type=3) the applicant status in application table change to OnCall (6) and the number of jobs of OpeningJob Table +1.

Query:

```
USE HR;
GO

DROP TRIGGER IF EXISTS Recruitment.Declined;
GO

CREATE TRIGGER Declined
       ON Recruitment.OnCallList
       AFTER INSERT--when insert into Recruitment.BlackListed trigger run
AS
       DECLARE @ApID int; --save application ID
       SELECT @ApID=inserted.ApplicationID FROM inserted
       DECLARE @OpJobID int; ---Save OpenjobID need to update
       SELECT  @OpJobID=HR.Recruitment.Application.OpeningJobsID FROM
HR.Recruitment.Application
       DECLARE @DeclinedID int
       SET @DeclinedID=3

    IF @DeclinedID= (select inserted.OncallTypeID From inserted)
       BEGIN;
              BEGIN TRY
                     BEGIN TRAN --Tran make database consistent
                            UPDATE HR.Recruitment.Application
                            SET HR.Recruitment.Application.StatueID=6 --change the
application statues to 6,which means Oncall
                            WHERE HR.Recruitment.Application.ApplicationID=@ApID

                            UPDATE HR.Recruitment.OpeningJobs
                            SET
HR.Recruitment.OpeningJobs.NumPosition=HR.Recruitment.OpeningJobs.NumPosition+1 --
change number of openjobs+1
                            WHERE
HR.Recruitment.OpeningJobs.OpeningJobsID=@OpJobID

                     COMMIT TRAN
              END TRY
              BEGIN CATCH--if any one of the tran cannot work, rollback and throw
the error message
                     ROLLBACK TRAN
              END CATCH
       END;

GO
```

Test&Result: Correct

```
SET IDENTITY_INSERT  Recruitment.OnCallList ON
INSERT INTO Recruitment.OnCallList  (OncallCode,OncallTypeID,ApplicationID)
VALUES
(4,3,1)
SET IDENTITY_INSERT Recruitment.OnCallList  OFF
```

	Changes	Output
Before	Applicant.StatueID=10 OpeningJobs.NumPosition =3	Appendix 2.4.2.1
After	Applicant.StatueID=6 OpeningJobs.NumPosition =4	Appendix 2.4.2.2

Triggers & Transactions 3

Propose:

When the applicant is select but not more job opening,(in my design declined statue is in the oncallList and type=1) the applicant status in application table change to OnCall (6) .

Query:

```
USE HR;
GO

DROP TRIGGER IF EXISTS Recruitment.Oncalled;
GO

CREATE TRIGGER Oncalled
       ON Recruitment.OnCallList
       AFTER INSERT--when insert into Recruitment.Oncalled.trigger run
AS
       DECLARE @ApID int; --save application ID
       SELECT @ApID=inserted.ApplicationID FROM inserted
       DECLARE @OpJobID int; ---Save OpenjobID need to update
       SELECT  @OpJobID=HR.Recruitment.Application.OpeningJobsID FROM
HR.Recruitment.Application
       DECLARE @DeclinedID int
       SET @DeclinedID=3

    IF @DeclinedID<> (select inserted.OncallTypeID From inserted)
       BEGIN;
              BEGIN TRY
                     BEGIN TRAN --Tran make database consistent
                            UPDATE HR.Recruitment.Application
                            SET HR.Recruitment.Application.StatueID=6 --change
the application statues to 6,which means Oncall
                            WHERE
HR.Recruitment.Application.ApplicationID=@ApID

                     COMMIT TRAN
              END TRY
              BEGIN CATCH--if any one of the tran cannot work, rollback and
throw the error message
                     ROLLBACK TRAN
              END CATCH
       END;

GO
```

Test&Result: Correct

```
SET IDENTITY_INSERT  Recruitment.OnCallList ON
INSERT INTO Recruitment.OnCallList  (OncallCode,OncallTypeID,ApplicationID)
VALUES
(4,3,1)
SET IDENTITY_INSERT Recruitment.OnCallList  OFF
```

	Changes	Output
Before	Applicant.StatueID=11 OpeningJobs.NumPosition =4	Appendix 2.4.3.1
After	Applicant.StatueID=6 OpeningJobs.NumPosition =4	Appendix 2.4.3.2

Triggers & Transactions 4

Propose:

When the applicant accept the offer then startig the boarding , the applicant status in application table change to Onboarding(9) and the number of jobs of OpeningJob Table -1.

Query:

```
USE HR;
GO

DROP TRIGGER IF EXISTS Recruitment.Oncalled;
GO

CREATE TRIGGER Oncalled
       ON Recruitment.OnCallList
       AFTER INSERT--when insert into Recruitment.Oncalled.trigger run
AS
       DECLARE @ApID int; --save application ID
       SELECT @ApID=inserted.ApplicationID FROM inserted
       DECLARE @OpJobID int; ---Save OpenjobID need to update
       SELECT  @OpJobID=HR.Recruitment.Application.OpeningJobsID FROM
HR.Recruitment.Application
       DECLARE @DeclinedID int
       SET @DeclinedID=3

    IF @DeclinedID<> (select inserted.OncallTypeID From inserted)
       BEGIN;
              BEGIN TRY
                     BEGIN TRAN --Tran make database consistent
                            UPDATE HR.Recruitment.Application
                            SET HR.Recruitment.Application.StatueID=6 --change
the application statues to 6,which means Oncall
                            WHERE
HR.Recruitment.Application.ApplicationID=@ApID

                     COMMIT TRAN
              END TRY
              BEGIN CATCH--if any one of the tran cannot work, rollback and
throw the error message
                     ROLLBACK TRAN
              END CATCH
       END;

GO
```

Test&Result: Correct

```
SET IDENTITY_INSERT  Recruitment.Onboarding ON
INSERT INTO Recruitment.Onboarding
(OnboardingID,ApplicationID,EducationCheck,Legitimacy) VALUES
(4,2,'PASS','PASS')
SET IDENTITY_INSERT Recruitment.Onboarding  OFF
```

	Changes	Output
Before	Applicant.StatueID=6 OpeningJobs.NumPosition =4	Appendix 2.4.4.1
After	Applicant.StatueID=9 OpeningJobs.NumPosition =3	Appendix 2.4.4.2

Triggers & Transactions 5

Propose:

When the applicant fail in the onboarding process,(in my design fail in the onboarding statue is in the oncallList and type=2) the applicant status in application table change to OnCall (6) and the number of jobs of OpeningJob Table +1.

Query:

```
USE HR;
GO

DROP TRIGGER IF EXISTS Recruitment.OnboardingFail;
GO

CREATE TRIGGER OnboardingFail
       ON Recruitment.OnCallList
       AFTER INSERT--when insert into Recruitment.BlackListed trigger run
AS
       DECLARE @ApID int; --save application ID
       SELECT @ApID=inserted.ApplicationID FROM inserted
       DECLARE @OpJobID int; ---Save OpenjobID need to update
       SELECT  @OpJobID=HR.Recruitment.Application.OpeningJobsID FROM
HR.Recruitment.Application
       DECLARE @DeclinedID int
       SET @DeclinedID=2---type=2

    IF @DeclinedID= (select inserted.OncallTypeID From inserted)
       BEGIN;
              BEGIN TRY
                     BEGIN TRAN --Tran make database consistent
                            UPDATE HR.Recruitment.Application
                            SET HR.Recruitment.Application.StatueID=6 --change
the application statues to 6,which means Oncall
                            WHERE
HR.Recruitment.Application.ApplicationID=@ApID

                            UPDATE HR.Recruitment.OpeningJobs
                            SET
HR.Recruitment.OpeningJobs.NumPosition=HR.Recruitment.OpeningJobs.NumPosition+1
--change number of openjobs+1
                            WHERE
HR.Recruitment.OpeningJobs.OpeningJobsID=@OpJobID

                     COMMIT TRAN
              END TRY
              BEGIN CATCH--if any one of the tran cannot work, rollback and
throw the error message
                     ROLLBACK TRAN
              END CATCH
       END;

GO
```

Test&Result: Correct

```
SET IDENTITY_INSERT  Recruitment.OnCallList ON
INSERT INTO Recruitment.OnCallList  (OncallCode,OncallTypeID,ApplicationID)
VALUES
(10,2,3)
SET IDENTITY_INSERT Recruitment.OnCallList  OFF
```

	Changes	Output
Before	Applicant.StatueID=6 OpeningJobs.NumPosition =4	Appendix 2.4.5.1
After	Applicant.StatueID=9 OpeningJobs.NumPosition =5	Appendix 2.4.5.2

Triggers & Transactions 6

Propose:

When the applicant reject by the interview (added in the RejectList), the application statue of the the applicant change to 2(reject).

Query:

```
USE HR;
GO

DROP TRIGGER IF EXISTS Recruitment.RejectAdd;
GO

CREATE TRIGGER RejectAdd
       ON Recruitment.RejectList
       AFTER INSERT--when insert into Recruitment.Oncalled.trigger run
AS
       DECLARE @ApID int; --save application ID
       SELECT @ApID=inserted.ApplicationID FROM inserted
       BEGIN;
              BEGIN TRY
                     BEGIN TRAN --Tran make database consistent
                            UPDATE HR.Recruitment.Application
                            SET HR.Recruitment.Application.StatueID=2 --change
the application statues to 2,which means reject
                            WHERE
HR.Recruitment.Application.ApplicationID=@ApID

                     COMMIT TRAN
              END TRY
              BEGIN CATCH--if any one of the tran cannot work, rollback and
throw the error message
                     ROLLBACK TRAN
              END CATCH
       END;

GO
```

Test&Result: Correct

```
SET IDENTITY_INSERT  Recruitment.RejectList ON
INSERT INTO Recruitment.RejectList  (RejectCode,ApplicationID,RejectTypeID)
VALUES
(5,7,1)
SET IDENTITY_INSERT Recruitment.RejectList OFF
```

	Changes	Output
Before	Applicant.StatueID=6	Appendix 2.4.6.1
After	Applicant.StatueID=2	Appendix 2.4.6.2

Triggers & Transactions 7

Propose:

When the applicant became an employee (added in the Employee), the application statue of the the applicant change to 11(Employee).

Query:

```
USE HR;
GO

DROP TRIGGER IF EXISTS Recruitment.EmployeeAdd;
GO

CREATE TRIGGER EmployeeAdd
       ON Recruitment.Employee
       AFTER INSERT--when insert into Recruitment.Oncalled.trigger run
AS
       DECLARE @ApID int; --save application ID
       SELECT @ApID=inserted.ApplicationID FROM inserted
       BEGIN;
              BEGIN TRY
                     BEGIN TRAN --Tran make database consistent
                            UPDATE HR.Recruitment.Application
                            SET HR.Recruitment.Application.StatueID=11 --change
the application statues to 11,which means employee
                            WHERE
HR.Recruitment.Application.ApplicationID=@ApID

                     COMMIT TRAN
              END TRY
              BEGIN CATCH--if any one of the tran cannot work, rollback and
throw the error message
                     ROLLBACK TRAN
              END CATCH
       END;

GO
```

Test&Result: Correct

```
SET IDENTITY_INSERT  Recruitment.Employee ON
INSERT INTO Recruitment.Employee (EmployeeID,ApplicationID) VALUES
(4,1)
SET IDENTITY_INSERT Recruitment.Employee  OFF
```

	Changes	Output
Before	Applicant.StatueID=2	Appendix 2.4.7.1
After	Applicant.StatueID=11	Appendix 2.4.7.2

Triggers & Transactions 8

Propose:

When the applicant make a complain (added in the ComplaintHandling), the application statue of the the applicant change to 4(Waiting).

Query:

```
USE HR;
GO

DROP TRIGGER IF EXISTS Recruitment.ComplainAdd;
GO

CREATE TRIGGER ComplainAdd
       ON Recruitment.ComplaintHandling
       AFTER INSERT--when insert into Recruitment.Oncalled.trigger run
AS
       DECLARE @ApID int; --save application ID
       SELECT @ApID=inserted.ApplicationID FROM inserted
       BEGIN;
              BEGIN TRY
                     BEGIN TRAN --Tran make database consistent
                            UPDATE HR.Recruitment.Application
                            SET HR.Recruitment.Application.StatueID=4 --change
the application statues to 4,which means waiting
                            WHERE
HR.Recruitment.Application.ApplicationID=@ApID

                     COMMIT TRAN
              END TRY
              BEGIN CATCH--if any one of the tran cannot work, rollback and
throw the error message
                     ROLLBACK TRAN
              END CATCH
       END;

GO
```

Test&Result: Correct

```
SET IDENTITY_INSERT  Recruitment.ComplaintHandling ON
INSERT INTO Recruitment.ComplaintHandling
(ComplaintID,ApplicationID,ComplainDesc) VALUES
(2,1,'Project kill me')
SET IDENTITY_INSERT Recruitment.ComplaintHandling  OFF
```

	Changes	Output
Before	Applicant.StatueID=11	Appendix 2.4.8.1
After	Applicant.StatueID=4	Appendix 2.4.8.2

Server Security

In real life, different people are responsible for different sections of a department, and they cannot interfere with each other. For example, here I have designed different Roles for my database, and each role can only handle fixed tables and DML operations. At the same time, employees in the HR department can access the database by them through each role, ensuring our database's isolation and security.

There is my design :

LOGIN	ROLE	TABLES	GRAND	MEMBER(USER)
CandidateGroup	CandidateManager	Recruitment.Candidates Recruitment.CandidatesInfo Recruitment.CandidatesDocuments	Update, Insert	WangXiao, DongLu
JobGroup	JobManager	Recruitment.JobTypes Recruitment.JobMediums Recruitment.JobCategories Recruitment.JobPositions Recruitment.JobOpeningPlatforms Recruitment.Jobs Recruitment.OpeningJobs	Update, Insert	WangXiao, WangPiPi
InterviewGroup	InterviewManager	Recruitment.Interviewers Recruitment.TestTypes Recruitment.Tests Recruitment.InterviewTypes Recruitment.Interviews	Update, Insert	WangXiao, Shawn
ReimbursementGroup	ReimbursementManager	Recruitment.Reimbursement Recruitment.AirlineReservations Recruitment.HotelReservations Recruitment.CarRentals	Update, Insert	WangXiao, Robin
ApplicationGroup	ApplicationManager	Recruitment.OnCallType Recruitment.RejectType Recruitment.Statues Recruitment.Application Recruitment.OnCallList Recruitment.Onboarding Recruitment.RejectList Recruitment.BlackListed Recruitment.Employee Recruitment.ComplaintHandling Recruitment.Evaluation Recruitment.Review	Update, Insert	WangXiao, Jack

I will implement this design step by step in the following part.

1. Create 5 roles

Query:

```
-- role for CandidateManger
--DROP ROLE CandidateManager;
CREATE ROLE CandidateManager;
GRANT UPDATE, INSERT
ON Recruitment.Candidates TO CandidateManager;
GRANT UPDATE, INSERT
ON Recruitment.CandidatesInfo TO CandidateManager;
GRANT UPDATE, INSERT
ON Recruitment.CandidatesDocuments TO CandidateManager;
ALTER ROLE db_datareader ADD MEMBER CandidateManager

--Role for JobManager
CREATE ROLE JobManager;
GRANT UPDATE, INSERT
ON  Recruitment.JobTypes TO JobManager;
GRANT UPDATE, INSERT
ON  Recruitment.JobMediums TO JobManager;
GRANT UPDATE, INSERT
ON  Recruitment.JobCategories TO JobManager;
GRANT UPDATE, INSERT
ON  Recruitment.JobPositions TO JobManager;
GRANT UPDATE, INSERT
ON  Recruitment.JobOpeningPlatforms TO JobManager;
GRANT UPDATE, INSERT
ON  Recruitment.Jobs TO JobManager;
GRANT UPDATE, INSERT
ON  Recruitment.OpeningJobs TO JobManager;
ALTER ROLE db_datareader ADD MEMBER JobManager
--Role for InterviewManager

CREATE ROLE InterviewManager;
GRANT UPDATE, INSERT
ON  Recruitment.Interviewers TO InterviewManager;
GRANT UPDATE, INSERT
ON  Recruitment.TestTypes TO InterviewManager;
GRANT UPDATE, INSERT
ON  Recruitment.Tests  TO InterviewManager;
GRANT UPDATE, INSERT
ON  Recruitment.InterviewTypes TO InterviewManager;
GRANT UPDATE, INSERT
ON  Recruitment.Interviews TO InterviewManager;
ALTER ROLE db_datareader ADD MEMBER InterviewManager

--Role ReimbursementManager
CREATE ROLE ReimbursementManager;
GRANT UPDATE, INSERT
ON  Recruitment.Reimbursement TO ReimbursementManager;
GRANT UPDATE, INSERT
ON  Recruitment.AirlineReservations TO ReimbursementManager;
GRANT UPDATE, INSERT
ON  Recruitment.HotelReservations TO ReimbursementManager;
GRANT UPDATE, INSERT
ON  Recruitment.CarRentals TO ReimbursementManager;
ALTER ROLE db_datareader ADD MEMBER ReimbursementManager

--Role ApplicationManager
CREATE ROLE  ApplicationManager;
```

```
GRANT UPDATE, INSERT
ON  Recruitment.OnCallType TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.RejectType TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.Statues TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.Application TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.OnCallList TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.Onboarding TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.RejectList TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.BlackListed TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.Employee TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.ComplaintHandling TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.Evaluation TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.Review TO ApplicationManager ;
ALTER ROLE db_datareader ADD MEMBER  ApplicationManager
```

Successful screenshot: Appendix 2.5.1

2. Create Lord login WANGXIAO

Query:

```
CREATE LOGIN Lord WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO

CREATE USER WANGXIAO FOR LOGIN [Lord];
ALTER ROLE CandidateManager ADD MEMBER WANGXIAO
ALTER ROLE InterviewManager ADD MEMBER WANGXIAO
ALTER ROLE ReimbursementManager ADD MEMBER WANGXIAO
ALTER ROLE ApplicationManager ADD MEMBER WANGXIAO
ALTER ROLE JobManager ADD MEMBER WANGXIAO
```

Successful screenshot: Appendix 2.5.2

3. Create CandidateGroup login DongLu

Query:

```
CREATE LOGIN CandidateGroup WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO

CREATE USER DongLu FOR LOGIN [CandidateGroup];
ALTER ROLE CandidateManager ADD MEMBER DongLu
```

Successful screenshot: Appendix 2.5.3

4. Create JobGroup login WangPiPi

Query:

```
CREATE LOGIN JobGroup WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO

CREATE USER WangPiPi FOR LOGIN [JobGroup];
ALTER ROLE JobManager ADD MEMBER WangPiPi
```

Successful screenshot: Appendix 2.5.4

5. Create InterviewGroup login Shawn

Query:

```
CREATE LOGIN InterviewGroup WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO

CREATE USER Shawn FOR LOGIN [InterviewGroup];
ALTER ROLE InterviewManager ADD MEMBER Shawn
```

Successful screenshot: Appendix 2.5.5

6. Create `ReimbursementGroup` login `Robin`

Query:

```
CREATE LOGIN ReimbursementGroup WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO

CREATE USER Robin FOR LOGIN [ReimbursementGroup];
ALTER ROLE ReimbursementManager ADD MEMBER Robin
```

Successful screenshot: Appendix 2.5.6

7. Create `ApplicationGroup` login `Jack`

Query:

```
CREATE LOGIN ApplicationGroup WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO

CREATE USER Jack FOR LOGIN [ApplicationGroup];
ALTER ROLE ApplicationManager ADD MEMBER Jack
```

Successful screenshot: Appendix 2.5.7

Business Report

Next, we ask some questions (questions that might interest management） we try to answer with our database and give possible business insights.

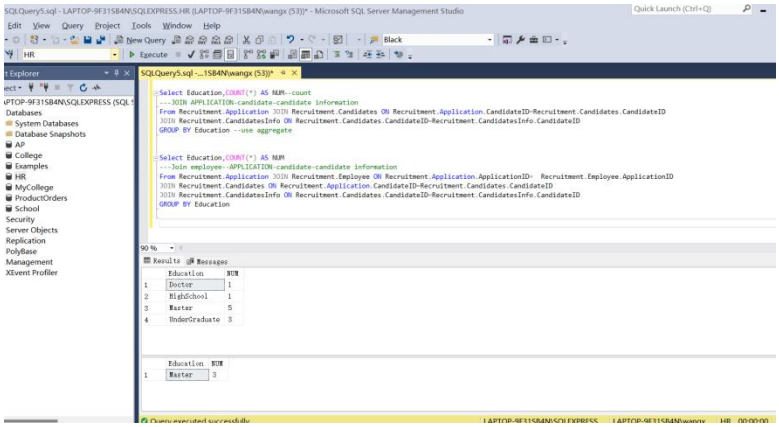
1.What is the educational background of applicants, and what are new employees? Is there any connection between the two?

Query:

```
Select Education,COUNT(*) AS NUM--count
---JOIN APPLICATION-candidate-candidate information
From Recruitment.Application JOIN Recruitment.Candidates ON
Recruitment.Application.CandidateID=Recruitment.Candidates.CandidateID
JOIN Recruitment.CandidatesInfo ON
Recruitment.Candidates.CandidateID=Recruitment.CandidatesInfo.CandidateID
GROUP BY Education --use aggregate
```

```
Select Education,COUNT(*) AS NUM
---Join employee--APPLICATION-candidate-candidate information
From Recruitment.Application JOIN Recruitment.Employee ON
Recruitment.Application.ApplicationID=  Recruitment.Employee.ApplicationID
JOIN Recruitment.Candidates ON
Recruitment.Application.CandidateID=Recruitment.Candidates.CandidateID
JOIN Recruitment.CandidatesInfo ON
Recruitment.Candidates.CandidateID=Recruitment.CandidatesInfo.CandidateID
GROUP BY Education
```

Screenshot:



Business insight:

The academic background of our applicants shows a normal distribution, one doctorate degree, one high school, and most of them are undergraduate and master's degrees.

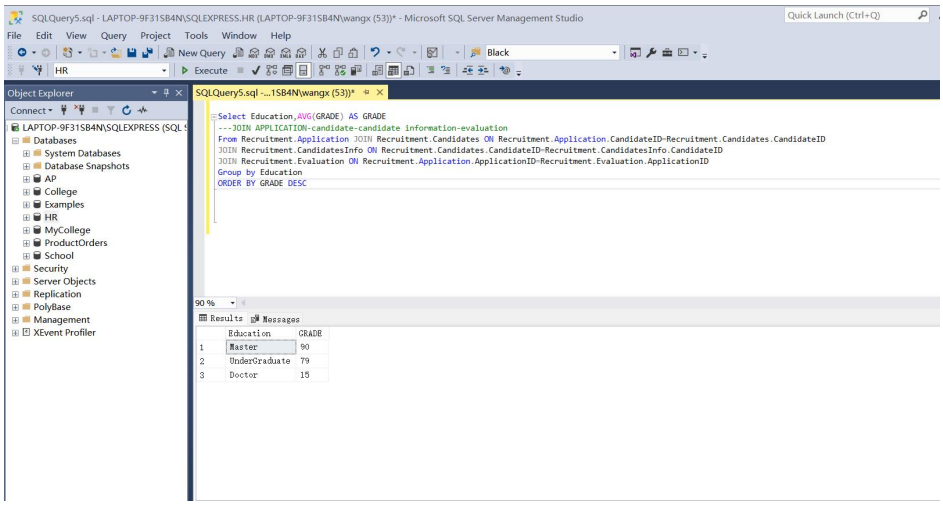
Our newly recruited employees all have master's degrees, which shows that our work requires (master+ level) employees to be competent.

2.How do people with different academic backgrounds perform on tests?

Query:

```
Select Education,AVG(GRADE) AS GRADE
---JOIN APPLICATION-candidate-candidate information-evaluation
From Recruitment.Application JOIN Recruitment.Candidates ON
Recruitment.Application.CandidateID=Recruitment.Candidates.CandidateID
JOIN Recruitment.CandidatesInfo ON
Recruitment.Candidates.CandidateID=Recruitment.CandidatesInfo.CandidateID
JOIN Recruitment.Evaluation ON
Recruitment.Application.ApplicationID=Recruitment.Evaluation.ApplicationID
Group by Education
ORDER BY GRADE DESC
```

Screenshot:



Business insight:

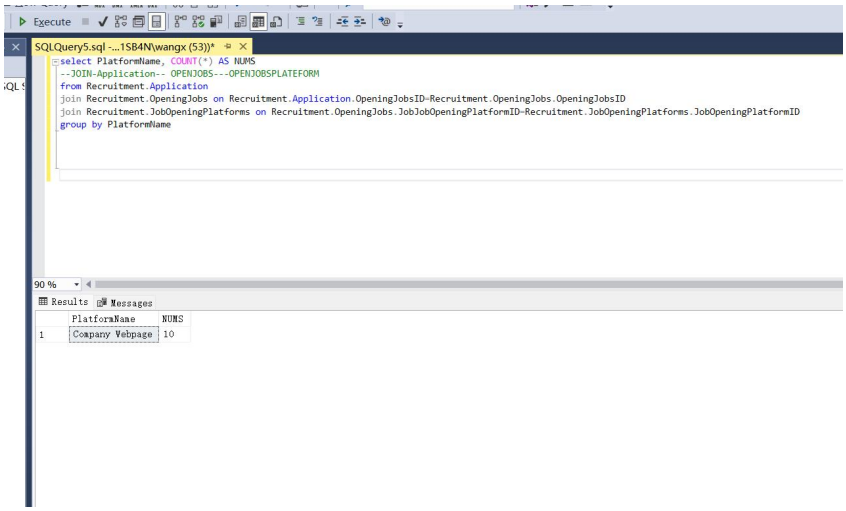
We can find that applicants with high school degrees fail to pass the resume test, so there is no relevant record. Unexpectedly, the person with a doctoral degree has an unexpectedly low grade. He was likely in a hurry at the time, so he Dropped out of the test halfway through.
We can find that applicants with a master's degree have significantly higher grades than a bachelor's degree, which is also in line with our common sense.

3.What kind of recruitment platform can attract more applicants?

Query：

```
select PlatformName, COUNT(*) AS NUMS
--JOIN-Application-- OPENJOBS---OPENJOBSPLATEFORM
from Recruitment.Application
join Recruitment.OpeningJobs on
Recruitment.Application.OpeningJobsID=Recruitment.OpeningJobs.OpeningJobsID
join Recruitment.JobOpeningPlatforms on
Recruitment.OpeningJobs.JobJobOpeningPlatformID=Recruitment.JobOpeningPlatforms.
JobOpeningPlatformID
group by PlatformName
```

Screenshot：



Business insight：

This indicates a huge problem in the company's recruitment process. The company only puts its recruitment information on the company's website. In this era full of information and developed social media, it is hard to imagine that the applicant will find the company website rather than the recruitment information. To attract more applicants, I suggest the company recruit on multiple platforms, such as Linkin, Handshake.

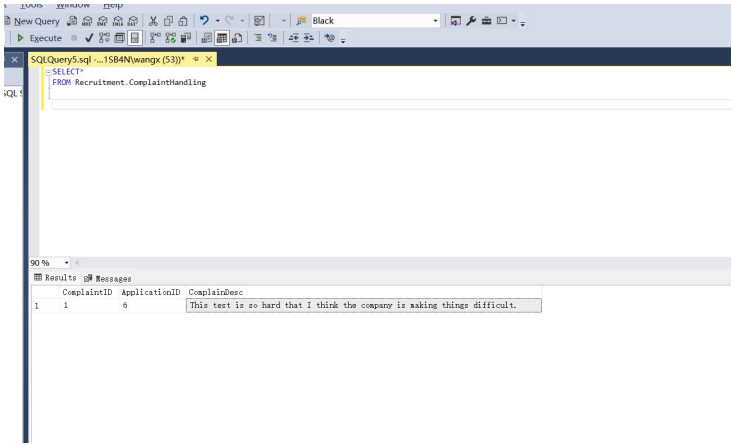
4. A case that how we handling the complains.

We find a complaint exists and try to solve it.

Step1:We can find the applicant complaining that "this test is so hard that I think the company is making things difficult."
Query:

```
SELECT*
FROM Recruitment.ComplaintHandling
```

ScreenShot:

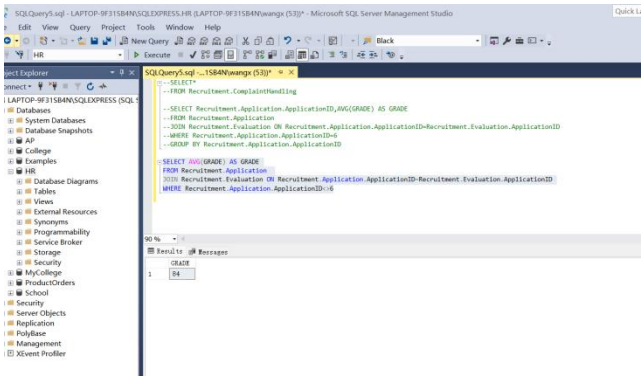


2.We check the performance of others except him.

Query:

```
SELECT AVG(GRADE) AS GRADE
FROM Recruitment.Application
JOIN Recruitment.Evaluation ON
Recruitment.Application.ApplicationID=Recruitment.Evaluation.ApplicationID
WHERE Recruitment.Application.ApplicationID<>6
```

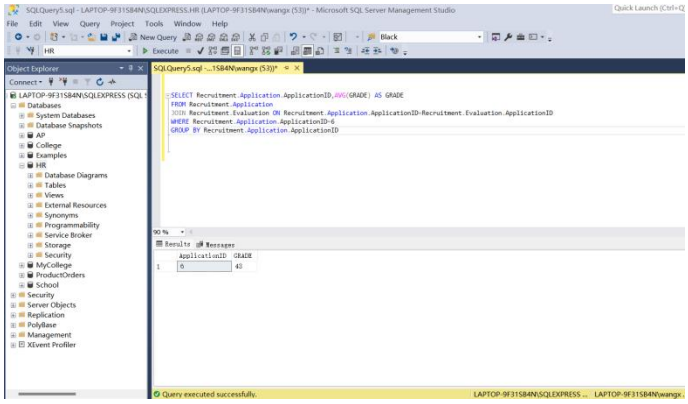
Screenshot:



3.His performance:
Query:

```
SELECT Recruitment.Application.ApplicationID,AVG(GRADE) AS GRADE
FROM Recruitment.Application
JOIN Recruitment.Evaluation ON
Recruitment.Application.ApplicationID=Recruitment.Evaluation.ApplicationID
WHERE Recruitment.Application.ApplicationID=6
GROUP BY Recruitment.Application.ApplicationID
```

Screenshot:



Decision:

We can find that his grades are much lower than the average grade. So if, as he complained, everyone's scores should be similar to his, he is making trouble for no reason. Reject!

Conclusion

In this project, we build our database step by step. First, we analyze the requirements, the relationship between each entity, and the characteristics of each column storage data, then draw the E/R diagram. At the same time, to ensure integrity, constraints are designed between each connected table. Then use SQL to implement our design.
Second, I design some scenarios to fill and test our database and designing the scene took me a lot of time.
Third, for the convenience of operation, I created views, UDF, and stored procedures to improve work efficiency. At the same time, the combined use of triggers&transaction is designed further to enhance the integrity and automation of the database and avoid potential errors.
Finally, I designed different roles to give them different permissions, further improving the database's security.

Future improvements:
Since this is a one-person project, I may have some mindsets in the design. For example, can the scenarios cover all application scenarios? The status of the status is whether it can cover all applicants. Will there be a status that we should have expected? Can the design idea of triggers&transaction be implemented for all tables? I believe that various problems will be exposed in practical applications, and we need to update our database according to the problems interactively.

Remarks:
For your convenience in testing, I have attached all the codes in the appendix ALL CODE part. Please run them in order.
Since we finally require the pdf version to be submitted, the code may have indentation and transcoding problems during the process of covering into pdf. So if your test fails, please email me (xwang99@syr.edu) at. I will send you my project code as soon as I get the email. Thanks.

Appendix

1. Database Design
1.1 Database requirements and design analysis

Requirements	Database design
C-3: Candidates: name, email, phone, short profile, etc. C-4: Documents: candidate, CVs, reference letters, cover letter, etc. Please note that documents table have links to actual documents.	I will design at least three tables to meet these requirements. All tables will meet at least third-level normalization. Some new columns, such as Locations and Education should be added for background evaluation
B-1: Candidate applies for job openings at Your Company.	Describe the fundamental logic.
B-2 Company rejects or selects the candidate for 1st interview, which can be online or onsite. B-3 : Company rejects or selects the candidate for the following interviews till the final 'nth' interview. B-7: The candidate can be rejected after any of the 'n' interviews. B-5: Interviews have start time and end time. One interview may involve multiple interviewers.	For each candidate/applicant, there may have many interviews for each candidate, and for each interview, there are many candidates. So the relationship between the **interviews and the applicants** is **many to many**. For each interview, the evaluation team can decide on each applicant and **change the statue** of the applicant.
B-5: Interviews have start time and end time. One interview may involve multiple interviewers. C-7: Interviews: application, type, start time, end times, interviewer, etc.	Two columns for **Interview start time** and **interview end time** should be added
B-6:Interviews may include multiple tests, which can again be online or onsite. Tests would have start and end time. A test may need to be graded by multiple interviewers with a simple result, passed or failed. C-6: Interviewers: name, department, title, etc. C-7:Interviews: application, type, start time, end times, interviewer, etc.	For each interview, there may have multiple tests. Moreover, One test should belong to one interview. (If the same questions are given to different interviews at different times, it will be unfair to the applicants who participated in the earlier interview) So the relationship is **one to many.** Each test should have a type (TestType Table (Online or In-person)) and time for start and end. For each interviewer, they should grade multiple applicants; and for each applicant, they should be graded by multiple interviewers. So the

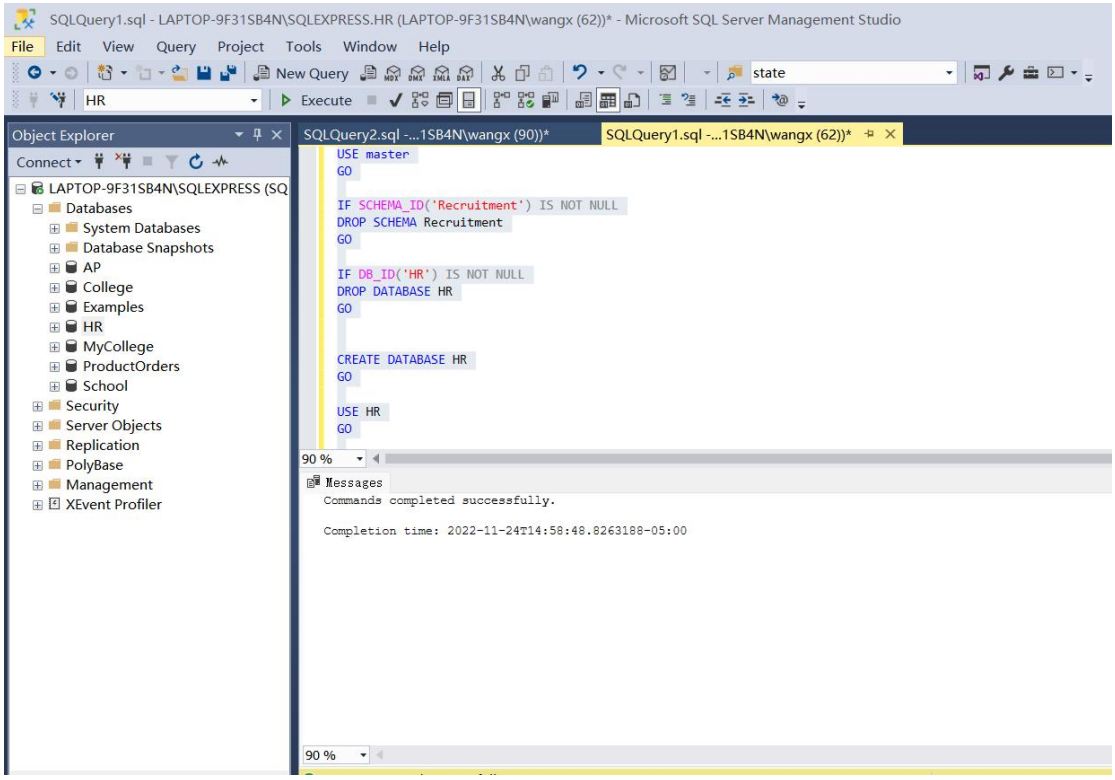
	relationship between interviewers and applicants is many to many. So, a the table of interviews should present the many to many relationship.
B-19:If any interview is onsite, your database needs to keep track of the following things per candidate: Airline reservation details, Hotel reservation details, Car Rental details. B-20:The candidate will be asked to submit their expense receipts - all expenses that the company says it will cover for food, etc. during the period of the onsite recruitment process. The reimbursement team will update whether the reimbursement that the candidate is asking for is valid and how much reimbursement will be given. B-21:If any interview is online or at a location close to the candidate ' s residence, one or more of the above given details would not be needed. B-31:Per interview, the candidate can give a review for the interviewer, and the interviewer can give a review for the candidate. C-10:Reimbursement: application, request, processed, amount, etc. Please note that this includes airfare, car rental, hotel reservation, food, etc.	The airline, Hotel, and Car Rental should be tacked for each interview. Three tables for each will be added to the design. For each interview, each applicant should have one order package for them. The relationship between **these three** and the **interviews table** are **one-to-one or one- to-many with applicant table .** Each applicant should have a **one-to-one relationship to reimbursement table.** Since the applicants and interviewers' relationship are many to many, so review can be add in the interview table.
B-8: If a rejected candidate re-applies for any job-opening, a brand-new status flow starts for that candidate. There is no limit on the number of times a candidate can re-apply or the time after which a candidate can re-apply.	Rules for add into Application table.
B-9: If a rejected candidate complains about the hiring or interviewing process, the company puts the candidate on a 'waiting' status. The 'Complaint Handling' branch of the HR department investigates the complaint (with whose working we are not concerned). If the Complaint department finds that the complaint is correct, a re-interview takes place and status is changed accordingly. However, if the Complaint Department finds that the complaint is invalid, status is again changed to 'rejected'.	1. The candidate have a **waiting statue.** 2. The candidate have a **reject statue.** 3. A **table for the information of complains** should be added. 4. A **trigger** can used here for change the state.
B-10: If after the final 'nth' interview the company likes the candidate but there are no more job openings available for the position	The candidate have a **on-call for next job opportunity statue.**

interviewed for, the candidate is put on a "on-call for next job opportunity" status. This means that the company will themselves contact this candidate for interviews during the next round of recruitment.	
B-11: If after the final 'nth' interview the company likes the candidate and there is a job opening available, an offer is extended to the candidate. B-13: The candidate can either decline or accept this offer or can negotiate 'n' number of times and then decline or accept the negotiated offer. Status is changed accordingly (Declined, Accepted, Negotiating). B-15:If the candidate declines the given or negotiated offer, s/he is again put on the "on-call for next job opportunity" status.	The candidate have **Negotiating ,Declined, Accepted statues.**
B-14: If the given or negotiated offer is accepted, the on-boarding procedure starts. The on-boarding procedure involves candidate's background check, and collection and checking of all documents required for employment. B-16:If on-boarding is successfully completed, the candidate becomes an employee. For this project, this is where our recruitment procedure will end. B-17: If the onboarding is unsuccessful (some joining requirements were not met) the candidate is again put on the "on-call for next job opportunity" status. B-18:If after accepting the offer and/or after successful completion of onboarding, the candidate does not join the company, s/he is blacklisted. This means that the candidate cannot again apply for any job-opening in the company. C-11:Onboarding: candidate, **job, start date, etc.**	**A table for on-boarding check** which contain background check and documents check. The candidate have **the employee statue.** The candidate have **the blacklisted statue.**
C-1:Job: position, title, type, medium, numbers of positions, etc. C-2: Job Openings: job, numbers of positions, etc. B-22: The job openings can be for any department's any position with the following details: actual job description including code,	Design for Job, the relation is clear and more column will be add in the design.

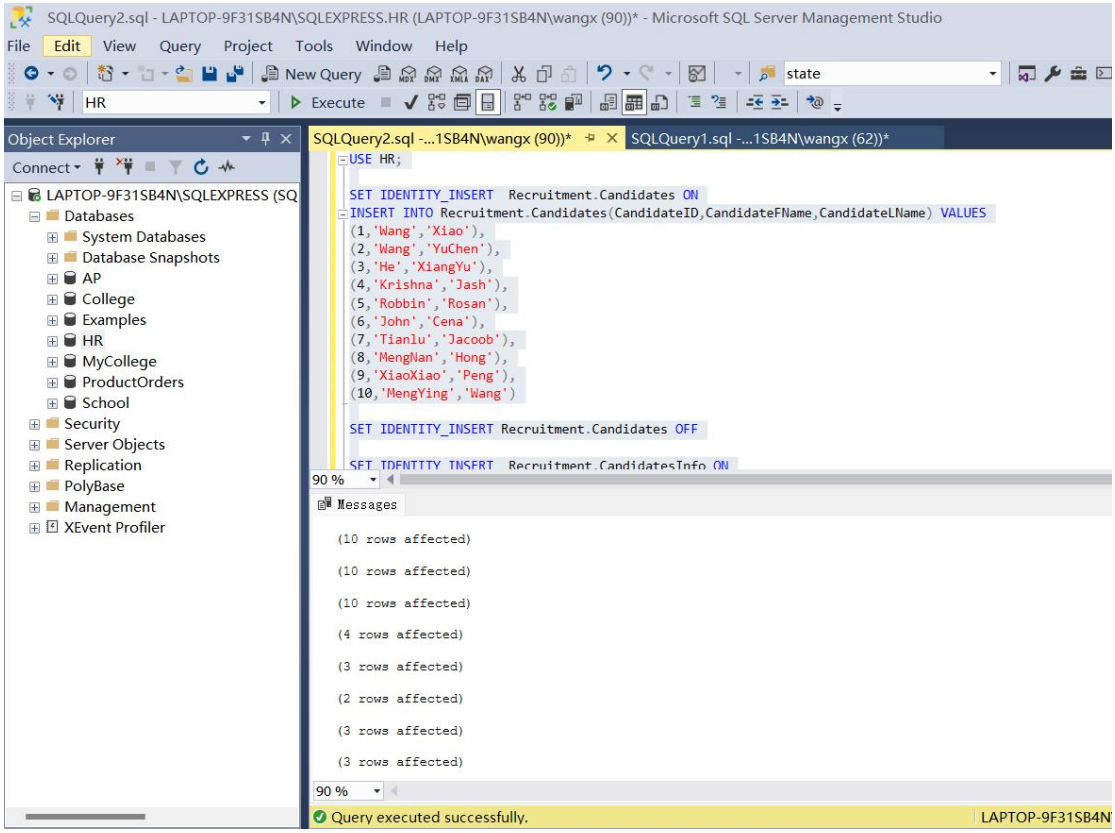
name, description, job start date, job type, job, type, job medium, job category, job platform, job start date, number of positions, etc. B-23: Job Type can be Summer Internship, Full-time Job, Part-time job, or Contract-based. B-24:Job Medium can be Online or Onsite. B-25:Job Categories can be such as “IT”, “software design”, “testing”, “finance”, etc. Job Position can be anything like “IT Manager”, “Software Developer”, etc. B-26: Job Platform is the medium used to post the job opening such as an online job board, company webpage, etc.	
B-28:The number of job openings per position are fixed and need to be tracked. Every time a position is filled by a candidate (that is when the status becomes ‘offer extended’), the number of job openings for that position should be reduced by 1).	Trigger for this situation will enforce on the job-opening table.
B-29: And every time an offer is declined, or somebody is blacklisted, the corresponding job opening should be increased by 1.	Trigger for this situation will enforce on the job-opening table
B-30:Although here we have only one database admin, the database has other roles of interviewers, onboarding team specialist, etc., with the power to change a few things in the database. Thus, when the reducing or increasing of the number of job-openings by 1 is taking place, make sure that nobody else can do that at the same time, else there would be a synchronization issue.	Use transactions.
Make the status changes due to the on-going interview process per candidate as automatic as possible as the design of your database can allow.	

1.2 Screen Shots

1.2.1 build database



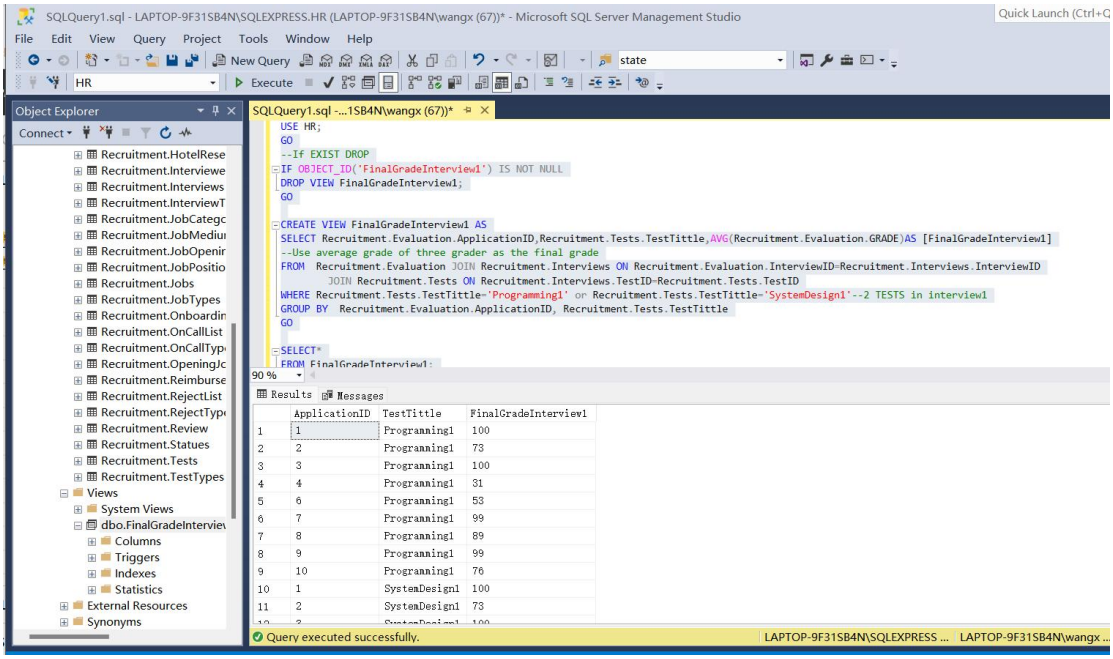
1.2.2 insert data



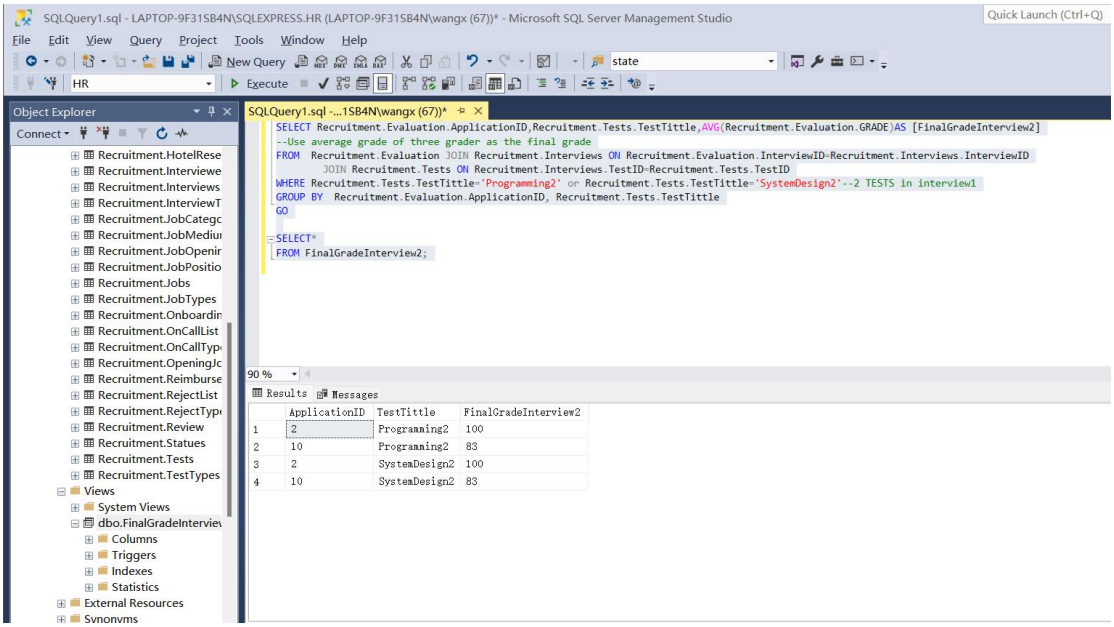
2.Database Test & Performance Improvement

2.1 views

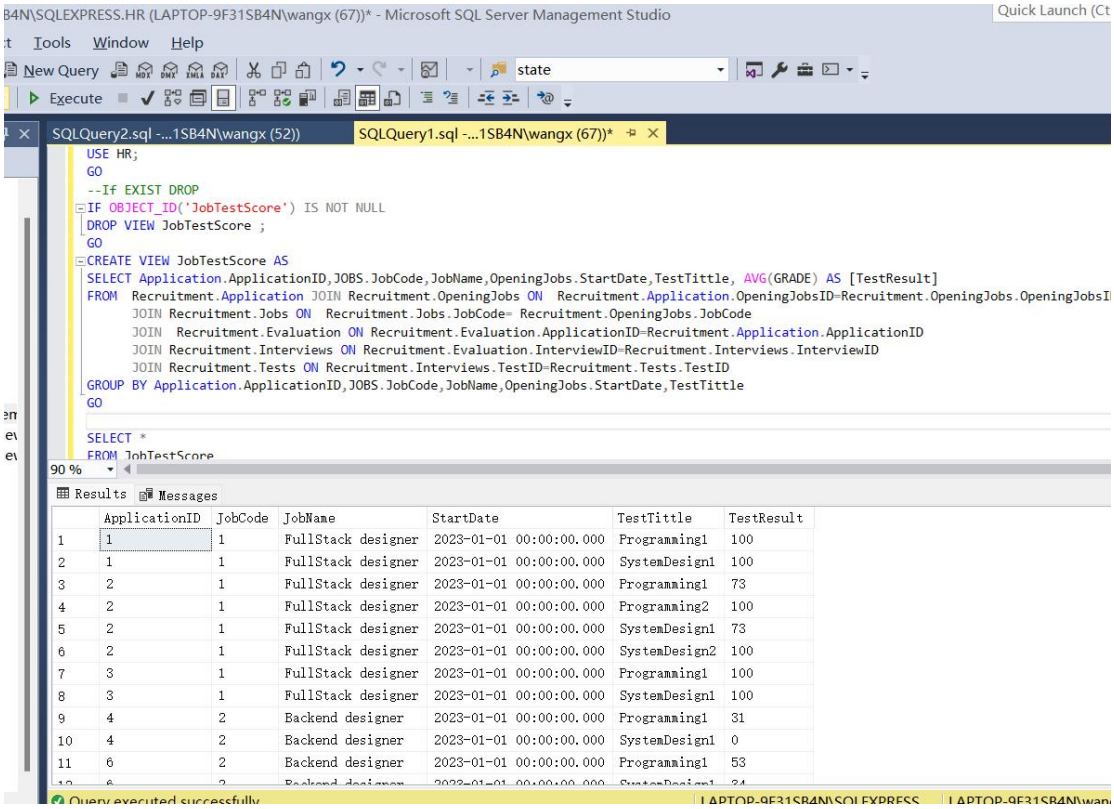
2.1.1



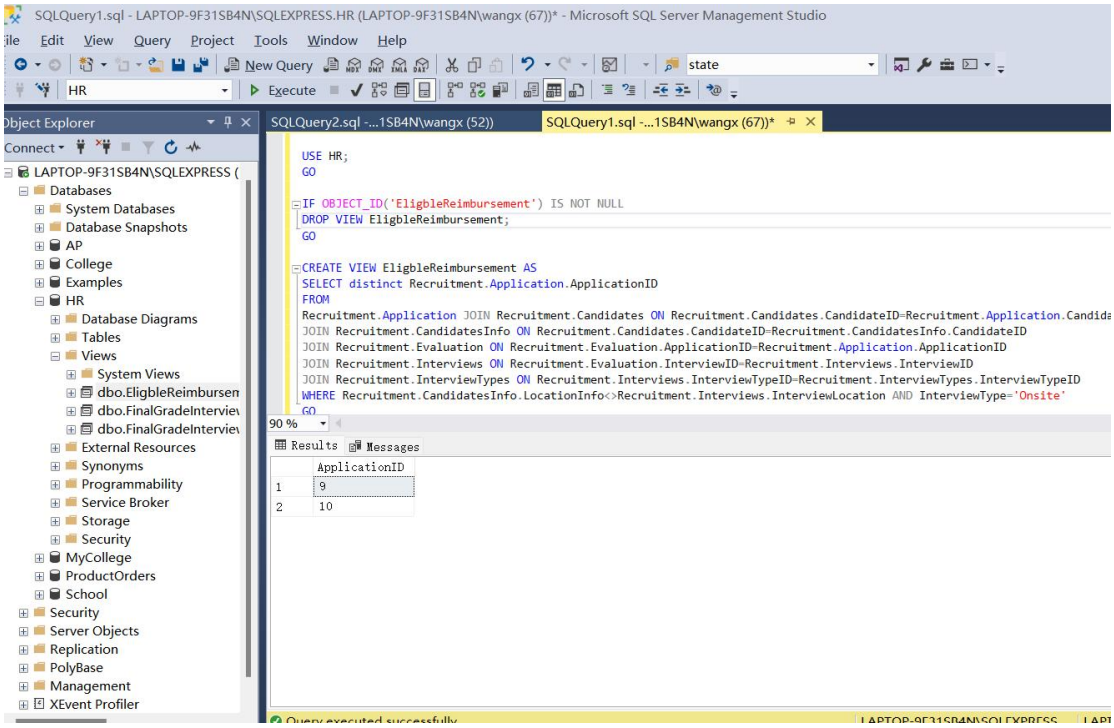
2.1.2



2.1.3

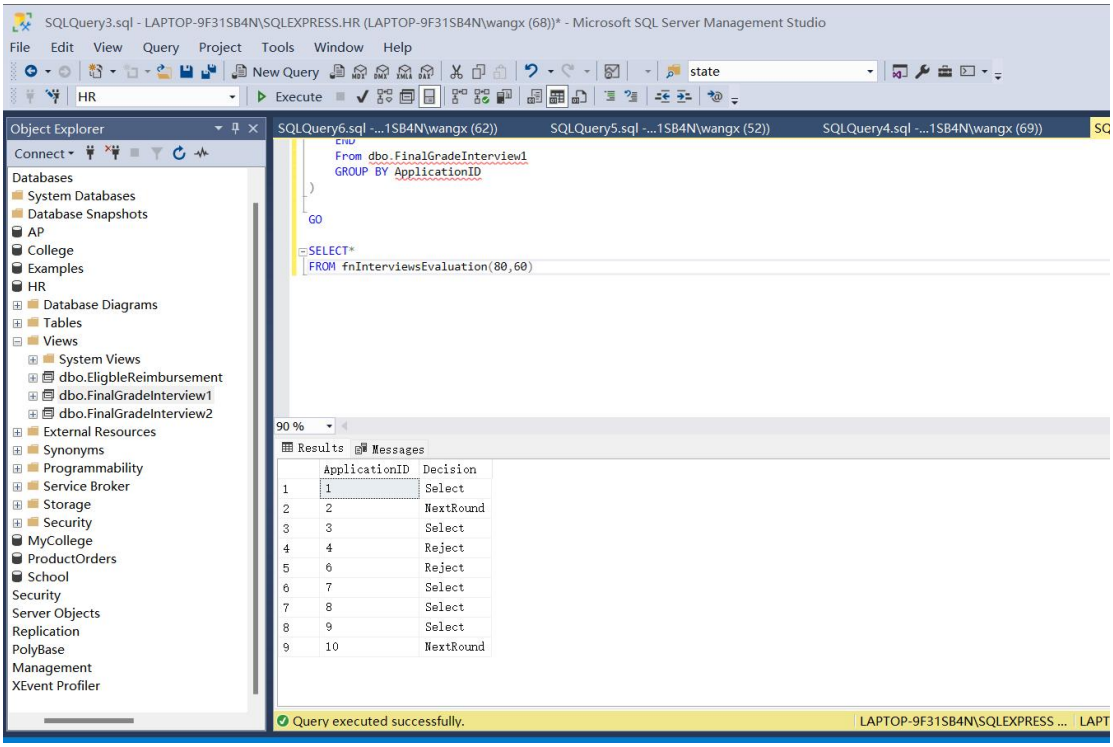

B4N\SQLEXPRESS.HR (LAPTOP-9F31SB4N\wangx (67))* - Microsoft SQL Server Management Studio
Quick Launch (Ct
Tools Window Help
New Query
state
Execute
SQLQuery2.sql -...1SB4N\wangx (52))
SQLQuery1.sql -...1SB4N\wangx (67))*
USE HR;
GO
--If EXIST DROP
IF OBJECT_ID('JobTestScore') IS NOT NULL
DROP VIEW JobTestScore ;
GO
CREATE VIEW JobTestScore AS
SELECT Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate,TestTittle, AVG(GRADE) AS [TestResult]
FROM Recruitment.Application JOIN Recruitment.OpeningJobs ON Recruitment.Application.OpeningJobsID=Recruitment.OpeningJobs.OpeningJobsI
JOIN Recruitment.Jobs ON Recruitment.Jobs.JobCode= Recruitment.OpeningJobs.JobCode
JOIN Recruitment.Evaluation ON Recruitment.Evaluation.ApplicationID=Recruitment.Application.ApplicationID
JOIN Recruitment.Interviews ON Recruitment.Evaluation.InterviewID=Recruitment.Interviews.InterviewID
JOIN Recruitment.Tests ON Recruitment.Interviews.TestID=Recruitment.Tests.TestID
GROUP BY Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate,TestTittle
GO
SELECT *
FROM JobTestScore
90 %
Results
Messages
ApplicationID JobCode JobName StartDate TestTittle TestResult
1 1 1 FullStack designer 2023-01-01 00:00:00.000 Programming1 100
2 1 1 FullStack designer 2023-01-01 00:00:00.000 SystemDesign1 100
3 2 1 FullStack designer 2023-01-01 00:00:00.000 Programming1 73
4 2 1 FullStack designer 2023-01-01 00:00:00.000 Programming2 100
5 2 1 FullStack designer 2023-01-01 00:00:00.000 SystemDesign1 73
6 2 1 FullStack designer 2023-01-01 00:00:00.000 SystemDesign2 100
7 3 1 FullStack designer 2023-01-01 00:00:00.000 Programming1 100
8 3 1 FullStack designer 2023-01-01 00:00:00.000 SystemDesign1 100
9 4 2 Backend designer 2023-01-01 00:00:00.000 Programming1 31
10 4 2 Backend designer 2023-01-01 00:00:00.000 SystemDesign1 0
11 6 2 Backend designer 2023-01-01 00:00:00.000 Programming1 53

2.1.4

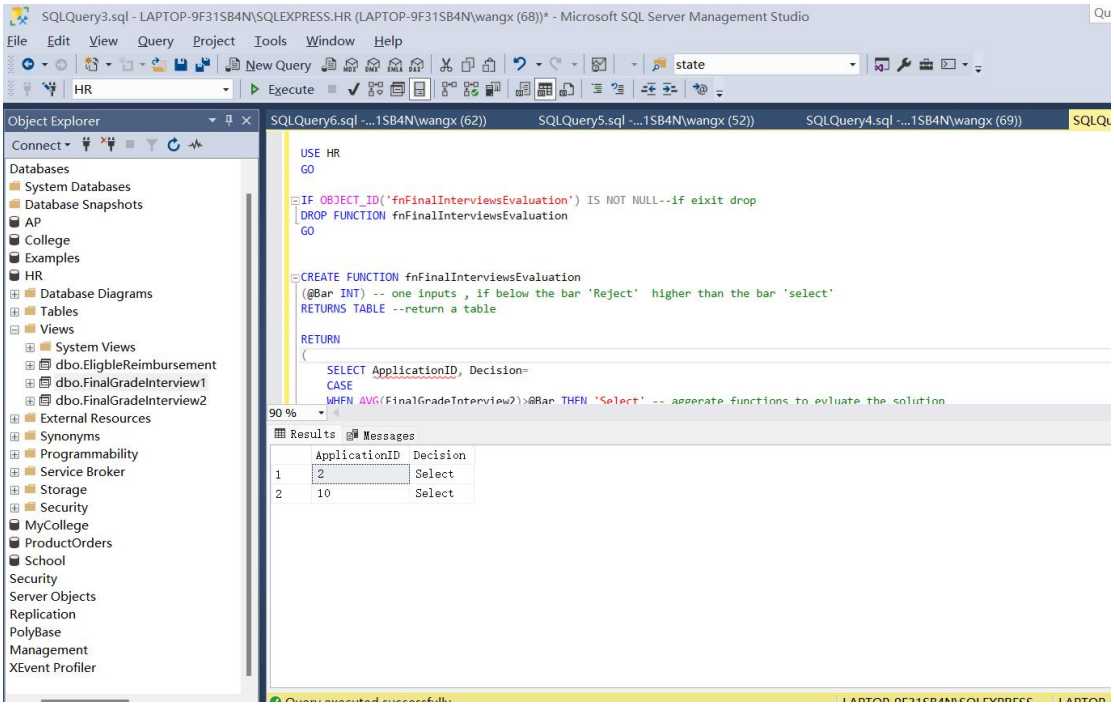

SQLQuery1.sql - LAPTOP-9F31SB4N\SQLEXPRESS.HR (LAPTOP-9F31SB4N\wangx (67))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
state
HR
Execute
Object Explorer
Connect
LAPTOP-9F31SB4N\SQLEXPRESS (
Databases
System Databases
Database Snapshots
AP
College
Examples
HR
Database Diagrams
Tables
Views
System Views
dbo.EligbleReimbursen
dbo.FinalGradeInterviev
dbo.FinalGradeInterviev
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
MyCollege
ProductOrders
School
Security
Server Objects
Replication
PolyBase
Management
XEvent Profiler
SQLQuery2.sql -...1SB4N\wangx (52))
SQLQuery1.sql -...1SB4N\wangx (67))*
USE HR;
GO
IF OBJECT_ID('EligbleReimbursement') IS NOT NULL
DROP VIEW EligbleReimbursement;
GO
CREATE VIEW EligbleReimbursement AS
SELECT distinct Recruitment.Application.ApplicationID
FROM
Recruitment.Application JOIN Recruitment.Candidates ON Recruitment.Candidates.CandidateID=Recruitment.Application.Candida
JOIN Recruitment.CandidatesInfo ON Recruitment.Candidates.CandidateID=Recruitment.CandidatesInfo.CandidateID
JOIN Recruitment.Evaluation ON Recruitment.Evaluation.ApplicationID=Recruitment.Application.ApplicationID
JOIN Recruitment.Interviews ON Recruitment.Evaluation.InterviewID=Recruitment.Interviews.InterviewID
JOIN Recruitment.InterviewTypes ON Recruitment.Interviews.InterviewTypeID=Recruitment.InterviewTypes.InterviewTypeID
WHERE Recruitment.CandidatesInfo.LocationInfo<>Recruitment.Interviews.InterviewLocation AND InterviewType='Onsite'
GO
90 %
Results
Messages
ApplicationID
1 9
2 10

2.2 Function

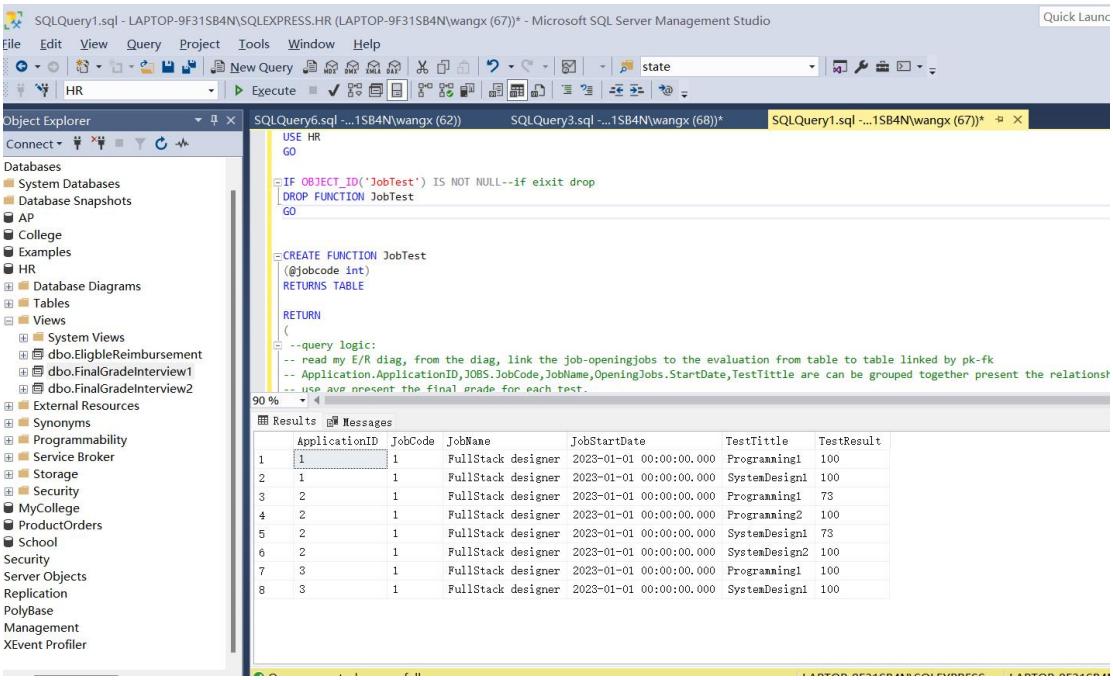
2.2.1



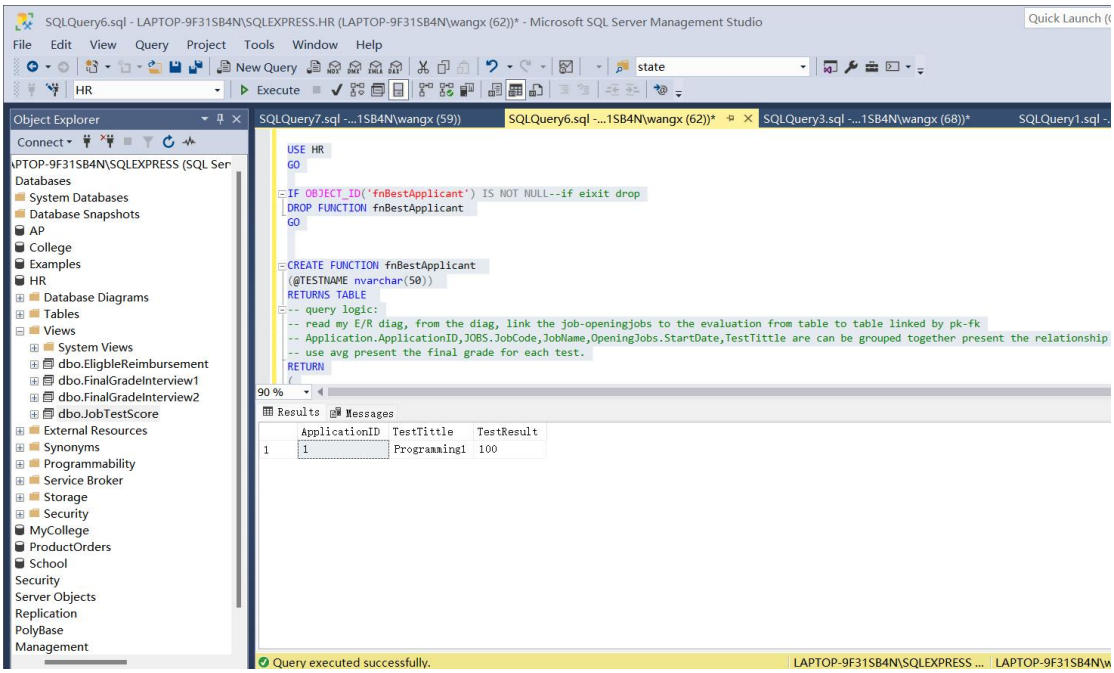
2.2.2



2.2.3

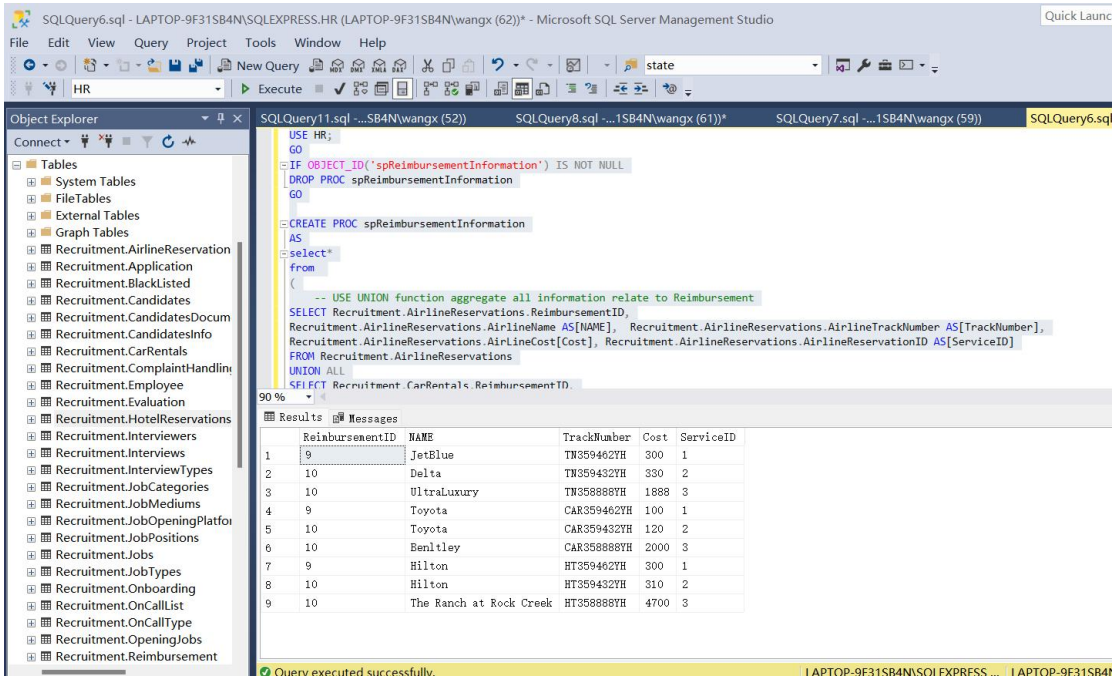

SQLQuery1.sql - LAPTOP-9F31SB4N\SQLEXPRESS.HR (LAPTOP-9F31SB4N\wangx (67))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Execute
HR
state
Object Explorer
Connect
Databases
System Databases
Database Snapshots
AP
College
Examples
HR
Database Diagrams
Tables
Views
System Views
dbo.EligbleReimbursement
dbo.FinalGradeInterview1
dbo.FinalGradeInterview2
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
MyCollege
ProductOrders
School
Security
Server Objects
Replication
PolyBase
Management
XEvent Profiler
SQLQuery6.sql -...1SB4N\wangx (62))
SQLQuery3.sql -...1SB4N\wangx (68))*
SQLQuery1.sql -...1SB4N\wangx (67))*
USE HR
GO
IF OBJECT_ID('JobTest') IS NOT NULL--if eixit drop
DROP FUNCTION JobTest
GO
CREATE FUNCTION JobTest
(@jobcode int)
RETURNS TABLE
RETURN
(
--query logic:
-- read my E/R diag, from the diag, link the job-openingjobs to the evaluation from table to table linked by pk-fk
-- Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate,TestTittle are can be grouped together present the relationsh
90 %
Results
Messages
ApplicationID JobCode JobName JobStartDate TestTittle TestResult
1 1 1 FullStack designer 2023-01-01 00:00:00.000 Programming1 100
2 1 1 FullStack designer 2023-01-01 00:00:00.000 SystemDesign1 100
3 2 1 FullStack designer 2023-01-01 00:00:00.000 Programming1 73
4 2 1 FullStack designer 2023-01-01 00:00:00.000 Programming2 100
5 2 1 FullStack designer 2023-01-01 00:00:00.000 SystemDesign1 73
6 2 1 FullStack designer 2023-01-01 00:00:00.000 SystemDesign2 100
7 3 1 FullStack designer 2023-01-01 00:00:00.000 Programming1 100
8 3 1 FullStack designer 2023-01-01 00:00:00.000 SystemDesign1 100

2.2.4:

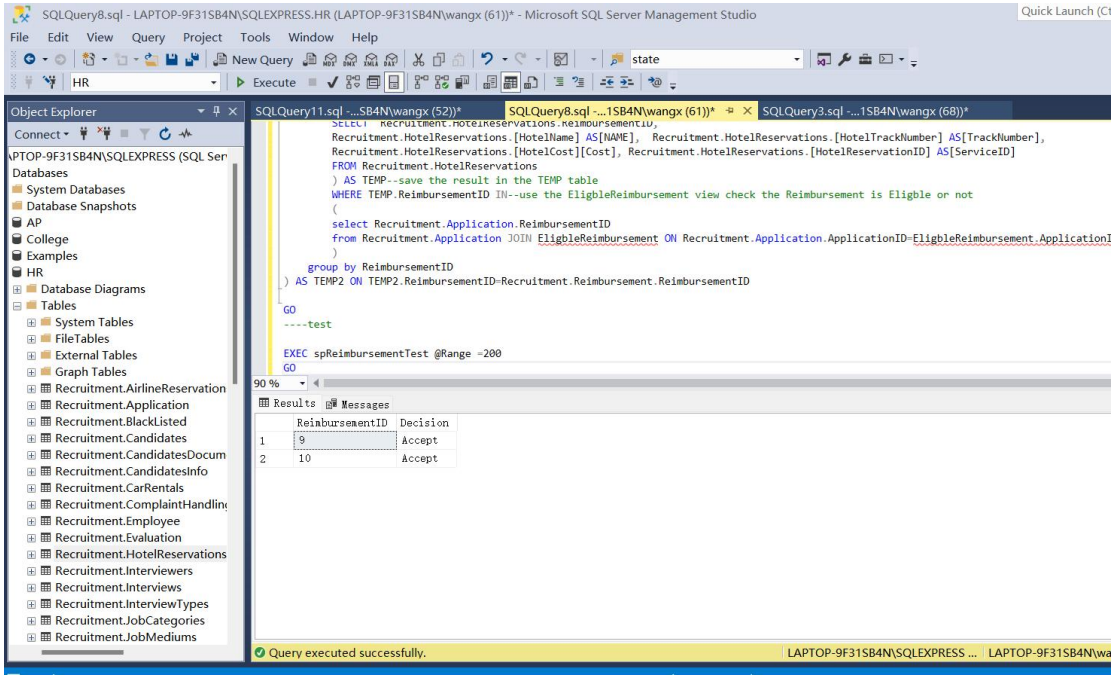

SQLQuery6.sql - LAPTOP-9F31SB4N\SQLEXPRESS.HR (LAPTOP-9F31SB4N\wangx (62))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Execute
HR
state
Object Explorer
Connect
\PTOP-9F31SB4N\SQLEXPRESS (SQL Ser
Databases
System Databases
Database Snapshots
AP
College
Examples
HR
Database Diagrams
Tables
Views
System Views
dbo.EligbleReimbursement
dbo.FinalGradeInterview1
dbo.FinalGradeInterview2
dbo.JobTestScore
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
MyCollege
ProductOrders
School
Security
Server Objects
Replication
PolyBase
Management
SQLQuery7.sql -...1SB4N\wangx (59))
SQLQuery6.sql -...1SB4N\wangx (62))*
SQLQuery3.sql -...1SB4N\wangx (68))*
USE HR
GO
IF OBJECT_ID('fnBestApplicant') IS NOT NULL--if eixit drop
DROP FUNCTION fnBestApplicant
GO
CREATE FUNCTION fnBestApplicant
(@TESTNAME nvarchar(50))
RETURNS TABLE
-- query logic:
-- read my E/R diag, from the diag, link the job-openingjobs to the evaluation from table to table linked by pk-fk
-- Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate,TestTittle are can be grouped together present the relationship
-- use avg present the final grade for each test.
RETURN
(
90 %
Results
Messages
ApplicationID TestTittle TestResult
1 1 Programming1 100
Query executed successfully.
LAPTOP-9F31SB4N\SQLEXPRESS ...
LAPTOP-9F31SB4N\w

2.3 Procedures

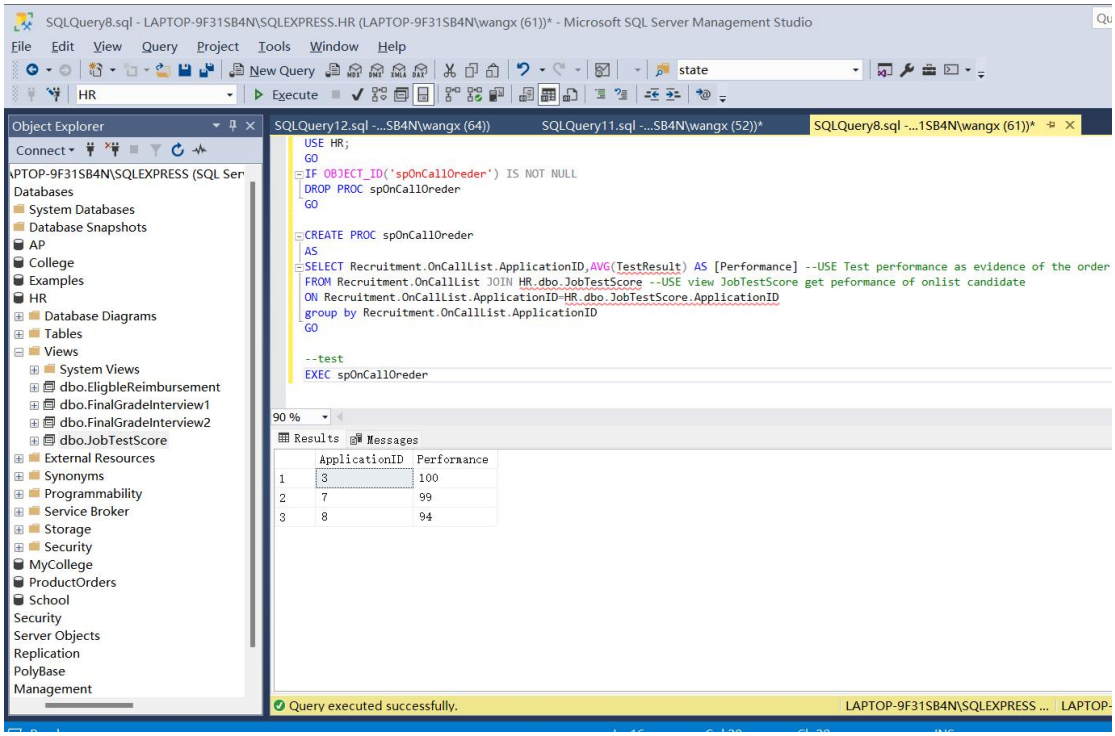
2.3.1:



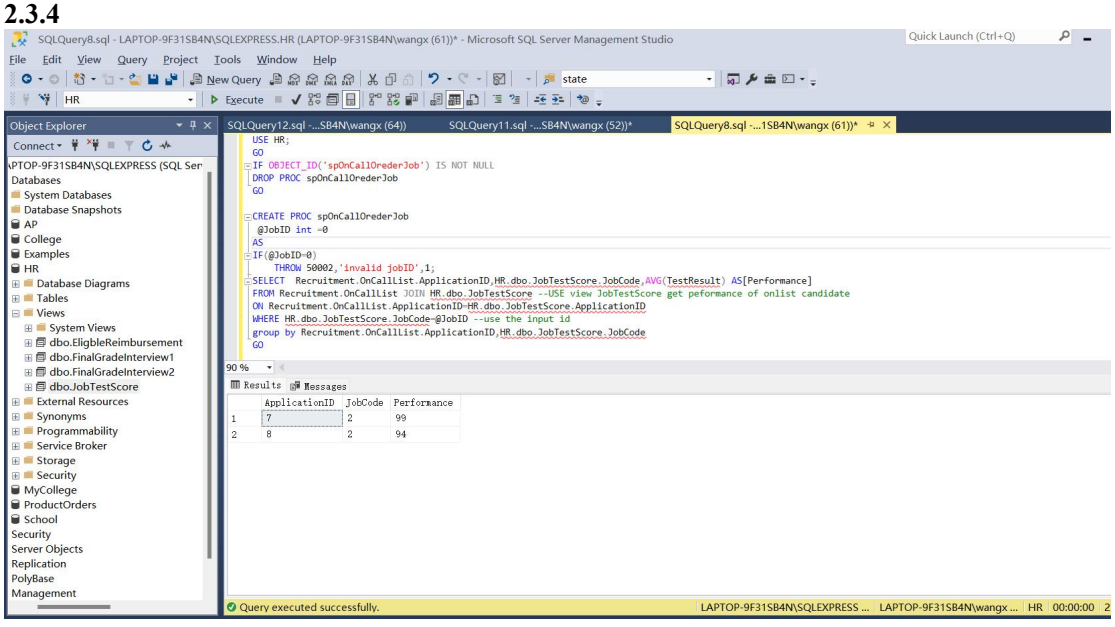
2.3.2



2.3.3


SQLQuery8.sql - LAPTOP-9F31SB4N\SQLEXPRESS.HR (LAPTOP-9F31SB4N\wangx (61))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
state
HR
Execute
Object Explorer
Connect
\PTOP-9F31SB4N\SQLEXPRESS (SQL Ser
Databases
System Databases
Database Snapshots
AP
College
Examples
HR
Database Diagrams
Tables
Views
System Views
dbo.EligbleReimbursement
dbo.FinalGradeInterview1
dbo.FinalGradeInterview2
dbo.JobTestScore
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
MyCollege
ProductOrders
School
Security
Server Objects
Replication
PolyBase
Management
SQLQuery12.sql -...SB4N\wangx (64))
SQLQuery11.sql -...SB4N\wangx (52))*
SQLQuery8.sql -...1SB4N\wangx (61))*
USE HR;
GO
IF OBJECT_ID('spOnCallOreder') IS NOT NULL
DROP PROC spOnCallOreder
GO
CREATE PROC spOnCallOreder
AS
SELECT Recruitment.OnCallList.ApplicationID,AVG(TestResult) AS [Performance] --USE Test performance as evidence of the order
FROM Recruitment.OnCallList JOIN HR.dbo.JobTestScore --USE view JobTestScore get peformance of onlist candidate
ON Recruitment.OnCallList.ApplicationID=HR.dbo.JobTestScore.ApplicationID
group by Recruitment.OnCallList.ApplicationID
GO
--test
EXEC spOnCallOreder
90 %
Results
Messages
ApplicationID Performance
1 3 100
2 7 99
3 8 94
Query executed successfully.
LAPTOP-9F31SB4N\SQLEXPRESS ...

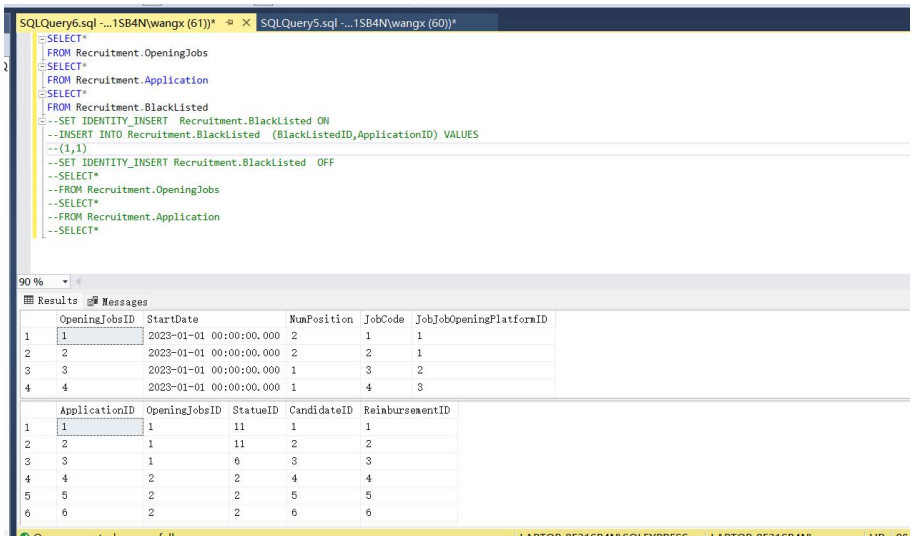
2.3.4


SQLQuery8.sql - LAPTOP-9F31SB4N\SQLEXPRESS.HR (LAPTOP-9F31SB4N\wangx (61))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
state
HR
Execute
Object Explorer
Connect
\PTOP-9F31SB4N\SQLEXPRESS (SQL Ser
Databases
System Databases
Database Snapshots
AP
College
Examples
HR
Database Diagrams
Tables
Views
System Views
dbo.EligbleReimbursement
dbo.FinalGradeInterview1
dbo.FinalGradeInterview2
dbo.JobTestScore
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
MyCollege
ProductOrders
School
Security
Server Objects
Replication
PolyBase
Management
SQLQuery12.sql -...SB4N\wangx (64))
SQLQuery11.sql -...SB4N\wangx (52))*
SQLQuery8.sql -...1SB4N\wangx (61))*
USE HR;
GO
IF OBJECT_ID('spOnCallOrederJob') IS NOT NULL
DROP PROC spOnCallOrederJob
GO
CREATE PROC spOnCallOrederJob
@JobID int =0
AS
IF(@JobID=0)
THROW 50002,'invalid jobID',1;
SELECT Recruitment.OnCallList.ApplicationID,HR.dbo.JobTestScore.JobCode,AVG(TestResult) AS[Performance]
FROM Recruitment.OnCallList JOIN HR.dbo.JobTestScore --USE view JobTestScore get peformance of onlist candidate
ON Recruitment.OnCallList.ApplicationID=HR.dbo.JobTestScore.ApplicationID
WHERE HR.dbo.JobTestScore.JobCode=@JobID --use the input id
group by Recruitment.OnCallList.ApplicationID,HR.dbo.JobTestScore.JobCode
GO
90 %
Results
Messages
ApplicationID JobCode Performance
1 7 2 99
2 8 2 94
Query executed successfully.
LAPTOP-9F31SB4N\SQLEXPRESS ...
LAPTOP-9F31SB4N\wangx ...
HR
00:00:00

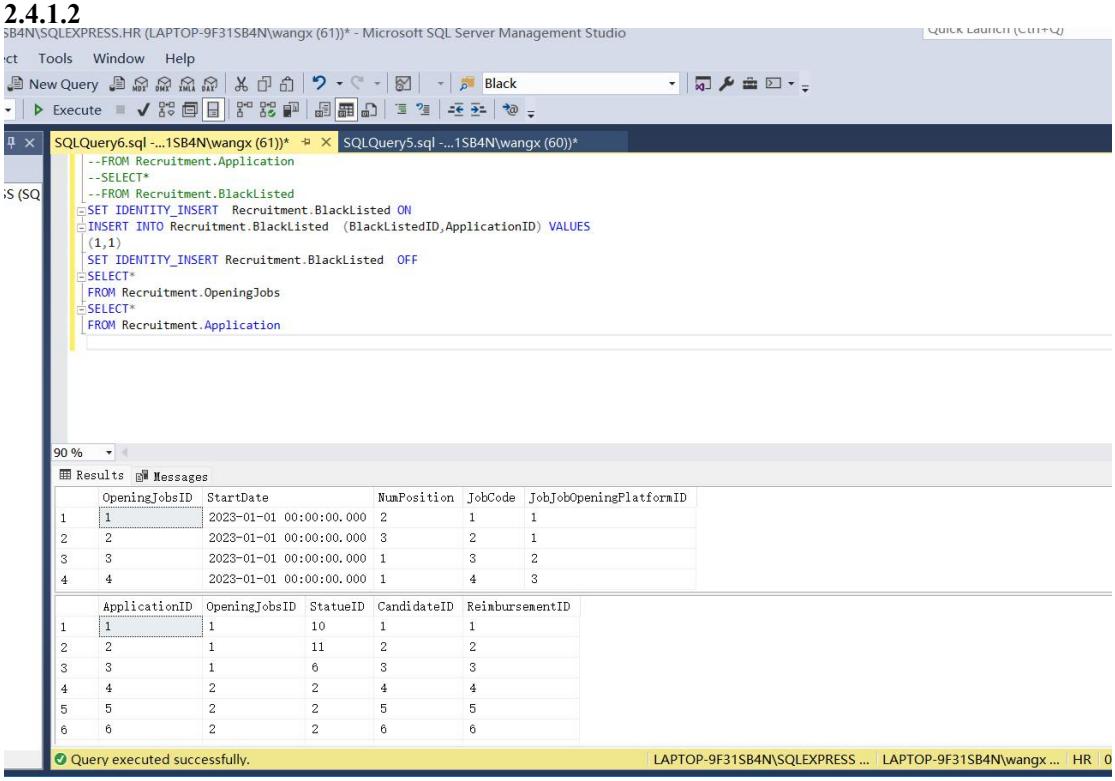
2.4Triggers & Transactions

2.4.1

2.4.1.1

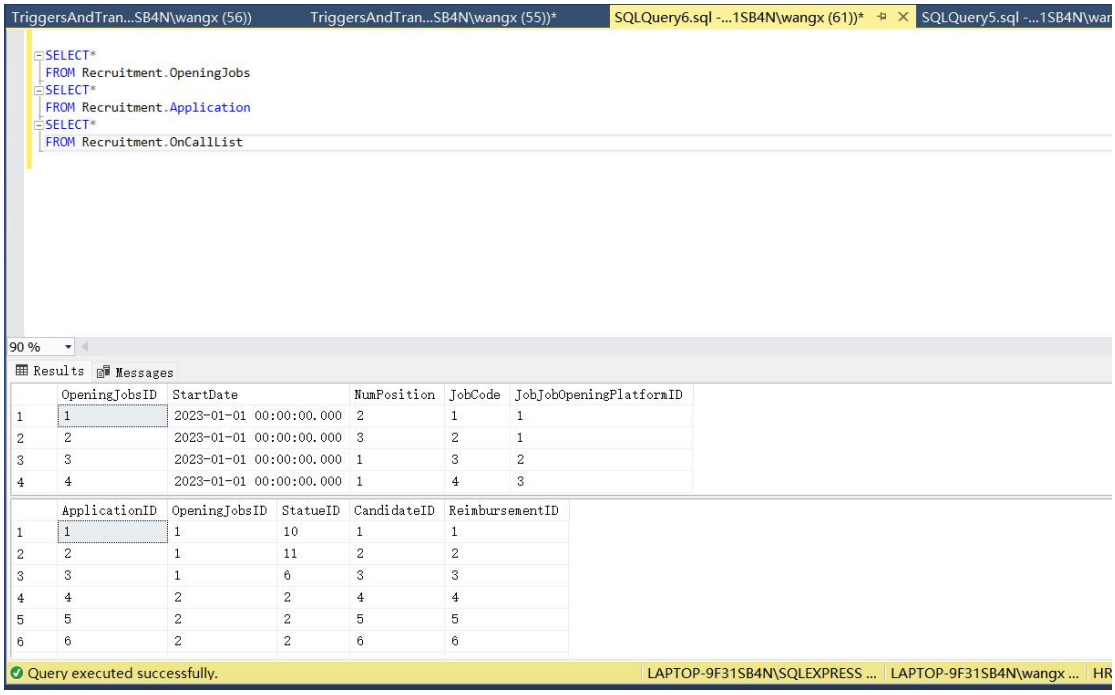


2.4.1.2

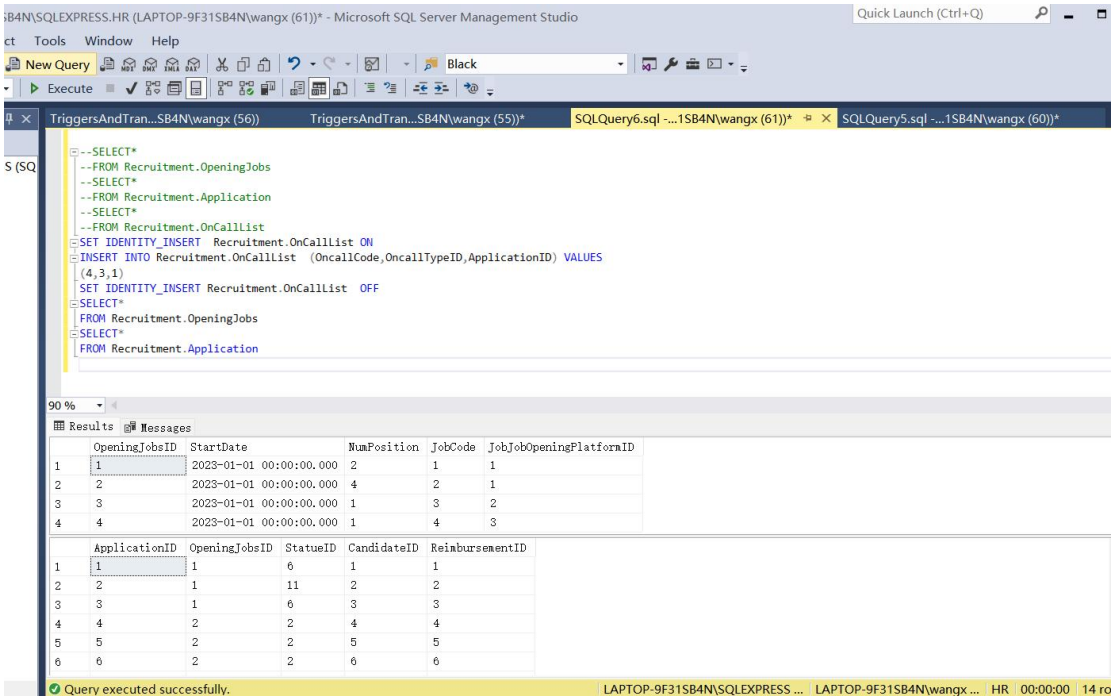


2.4.2

2.4.2.1

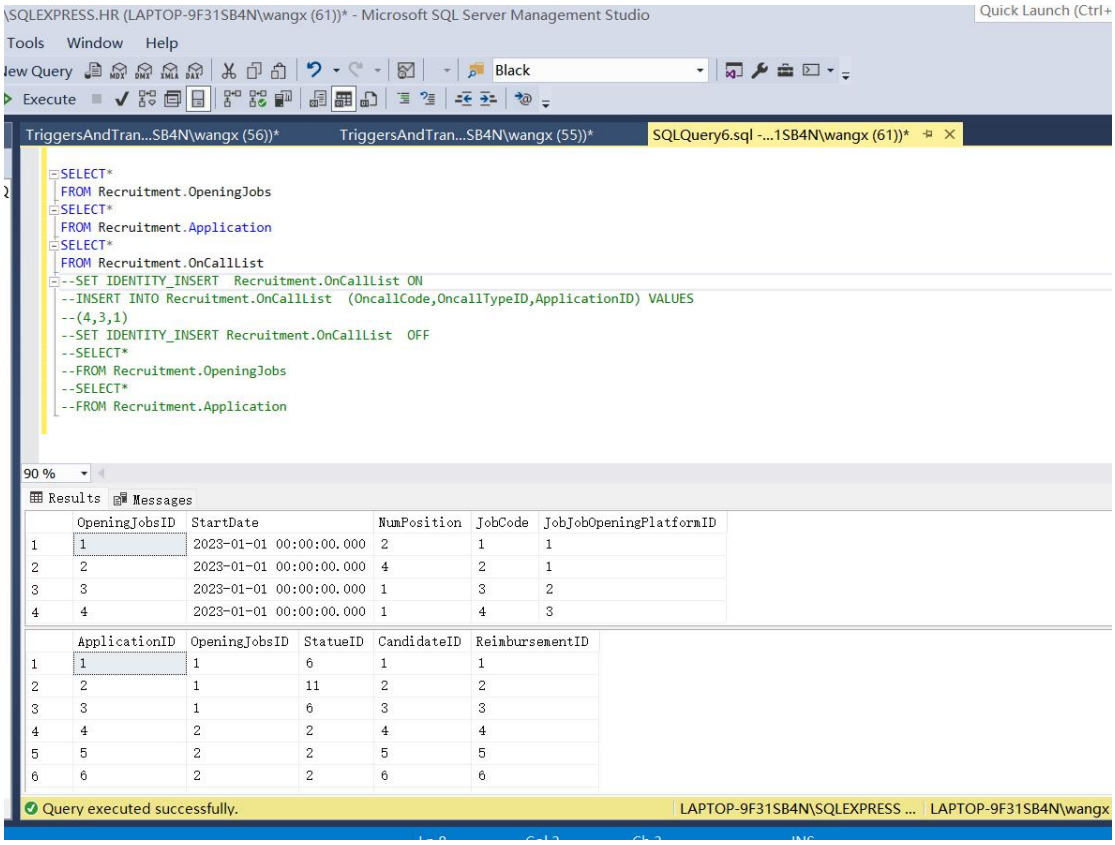

TriggersAndTran...SB4N\wangx (56))
TriggersAndTran...SB4N\wangx (55))*
SQLQuery6.sql -...1SB4N\wangx (61))*
SQLQuery5.sql -...1SB4N\wan
SELECT*
FROM Recruitment.OpeningJobs
SELECT*
FROM Recruitment.Application
SELECT*
FROM Recruitment.OnCallList
90 %
Results
Messages
OpeningJobsID StartDate NumPosition JobCode JobJobOpeningPlatformID
1 1 2023-01-01 00:00:00.000 2 1 1
2 2 2023-01-01 00:00:00.000 3 2 1
3 3 2023-01-01 00:00:00.000 1 3 2
4 4 2023-01-01 00:00:00.000 1 4 3
ApplicationID OpeningJobsID StatueID CandidateID ReimbursementID
1 1 1 10 1 1
2 2 1 11 2 2
3 3 1 6 3 3
4 4 2 2 4 4
5 5 2 2 5 5
6 6 2 2 6 6
Query executed successfully.
LAPTOP-9F31SB4N\SQLEXPRESS ...
LAPTOP-9F31SB4N\wangx ...
HR

2.4.2.2

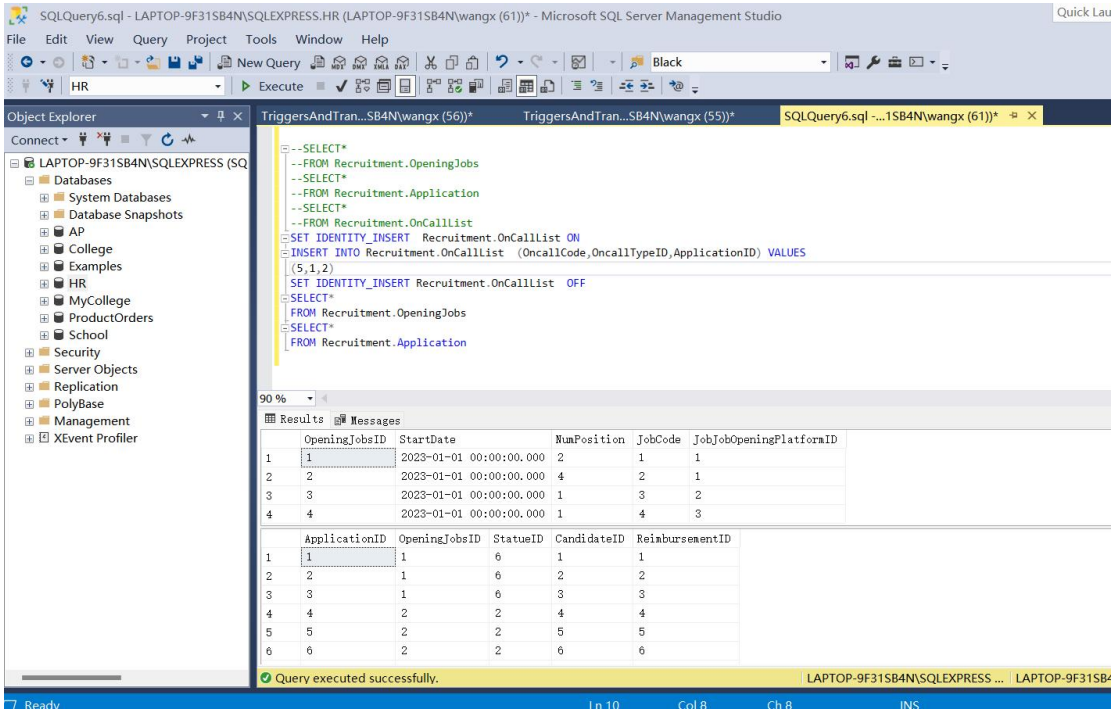

SB4N\SQLEXPRESS.HR (LAPTOP-9F31SB4N\wangx (61))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
ct Tools Window Help
New Query
Black
Execute
TriggersAndTran...SB4N\wangx (56))
TriggersAndTran...SB4N\wangx (55))*
SQLQuery6.sql -...1SB4N\wangx (61))*
SQLQuery5.sql -...1SB4N\wangx (60))*
--SELECT*
--FROM Recruitment.OpeningJobs
--SELECT*
--FROM Recruitment.Application
--SELECT*
--FROM Recruitment.OnCallList
SET IDENTITY_INSERT Recruitment.OnCallList ON
INSERT INTO Recruitment.OnCallList (OncallCode,OncallTypeID,ApplicationID) VALUES
(4,3,1)
SET IDENTITY_INSERT Recruitment.OnCallList OFF
SELECT*
FROM Recruitment.OpeningJobs
SELECT*
FROM Recruitment.Application
90 %
Results
Messages
OpeningJobsID StartDate NumPosition JobCode JobJobOpeningPlatformID
1 1 2023-01-01 00:00:00.000 2 1 1
2 2 2023-01-01 00:00:00.000 4 2 1
3 3 2023-01-01 00:00:00.000 1 3 2
4 4 2023-01-01 00:00:00.000 1 4 3
ApplicationID OpeningJobsID StatueID CandidateID ReimbursementID
1 1 1 6 1 1
2 2 1 11 2 2
3 3 1 6 3 3
4 4 2 2 4 4
5 5 2 2 5 5
6 6 2 2 6 6
Query executed successfully.
LAPTOP-9F31SB4N\SQLEXPRESS ...
LAPTOP-9F31SB4N\wangx ...
HR
00:00:00
14 ro

2.4.3:

2.4.3.1

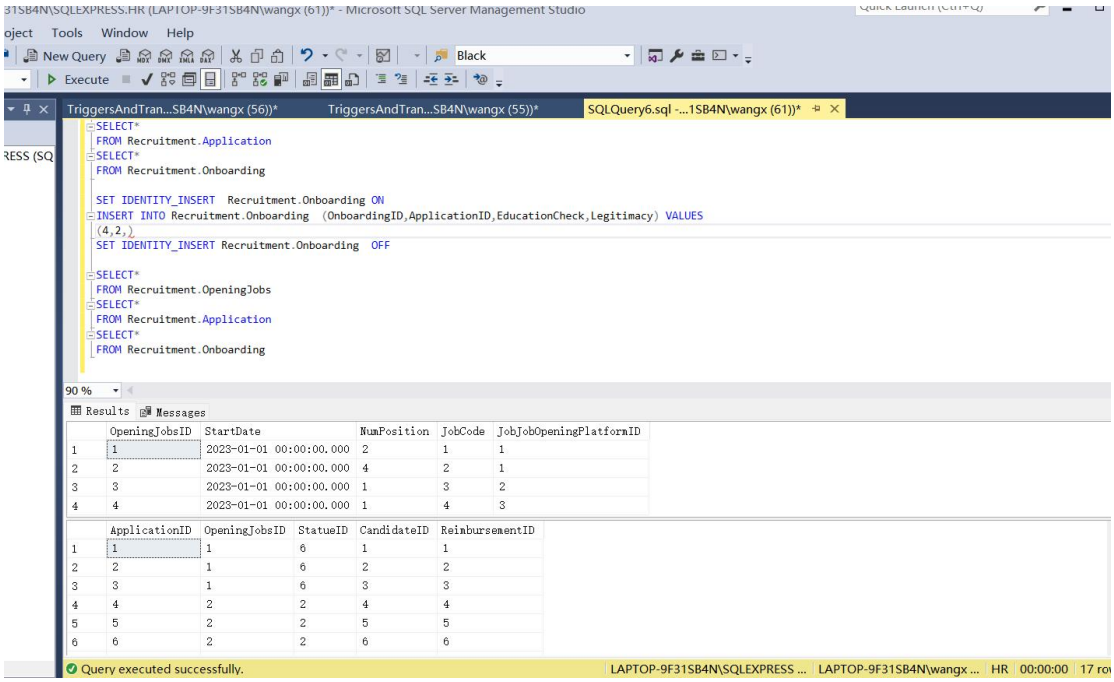


2.4.3.2

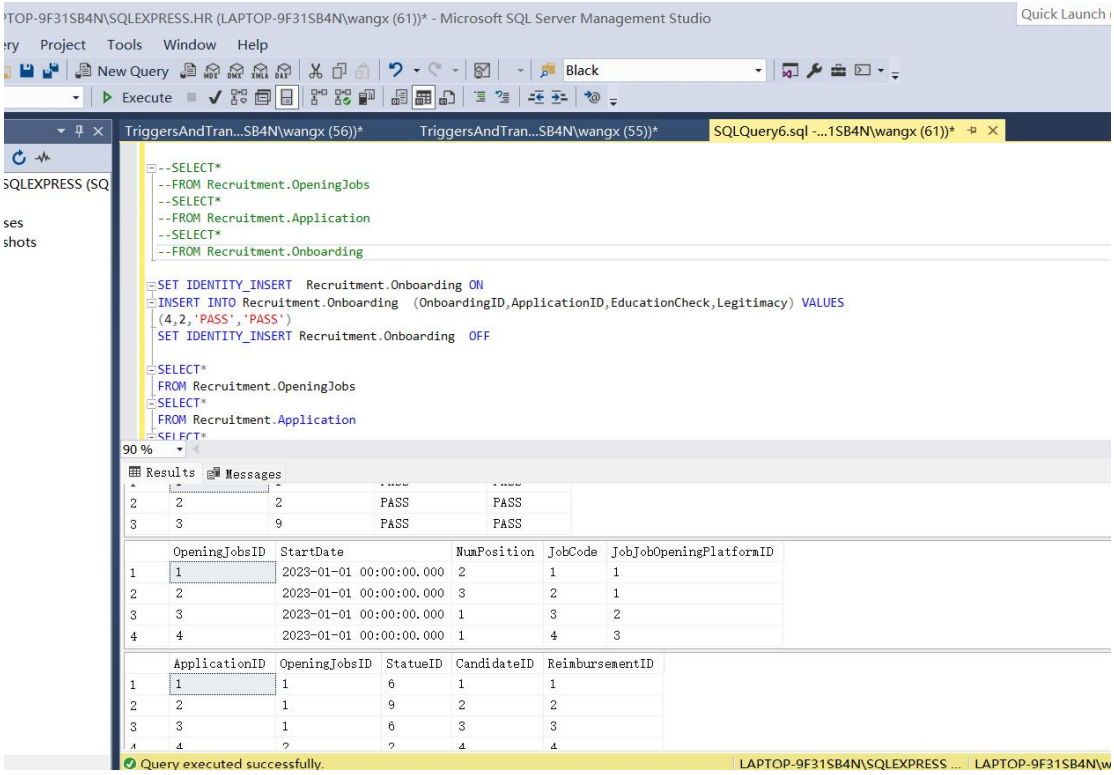


2.4.4:

2.4.4.1

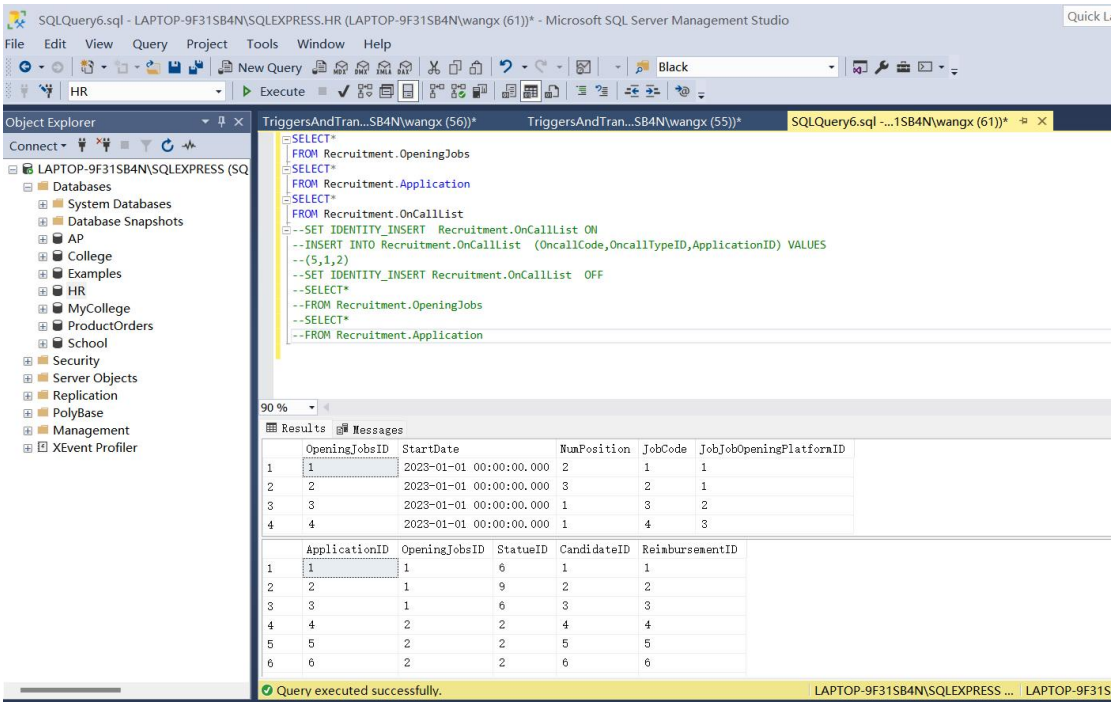

SELECT*
FROM Recruitment.Application
SELECT*
FROM Recruitment.Onboarding
SET IDENTITY_INSERT Recruitment.Onboarding ON
INSERT INTO Recruitment.Onboarding (OnboardingID,ApplicationID,EducationCheck,Legitimacy) VALUES
(4,2,)
SET IDENTITY_INSERT Recruitment.Onboarding OFF
SELECT*
FROM Recruitment.OpeningJobs
SELECT*
FROM Recruitment.Application
SELECT*
FROM Recruitment.Onboarding
Results
Messages
OpeningJobsID StartDate NumPosition JobCode JobJobOpeningPlatformID
1 2023-01-01 00:00:00.000 2 1 1
2 2023-01-01 00:00:00.000 4 2 1
3 2023-01-01 00:00:00.000 1 3 2
4 2023-01-01 00:00:00.000 1 4 3
ApplicationID OpeningJobsID StatueID CandidateID ReimbursementID
Query executed successfully.

2.4.4.2

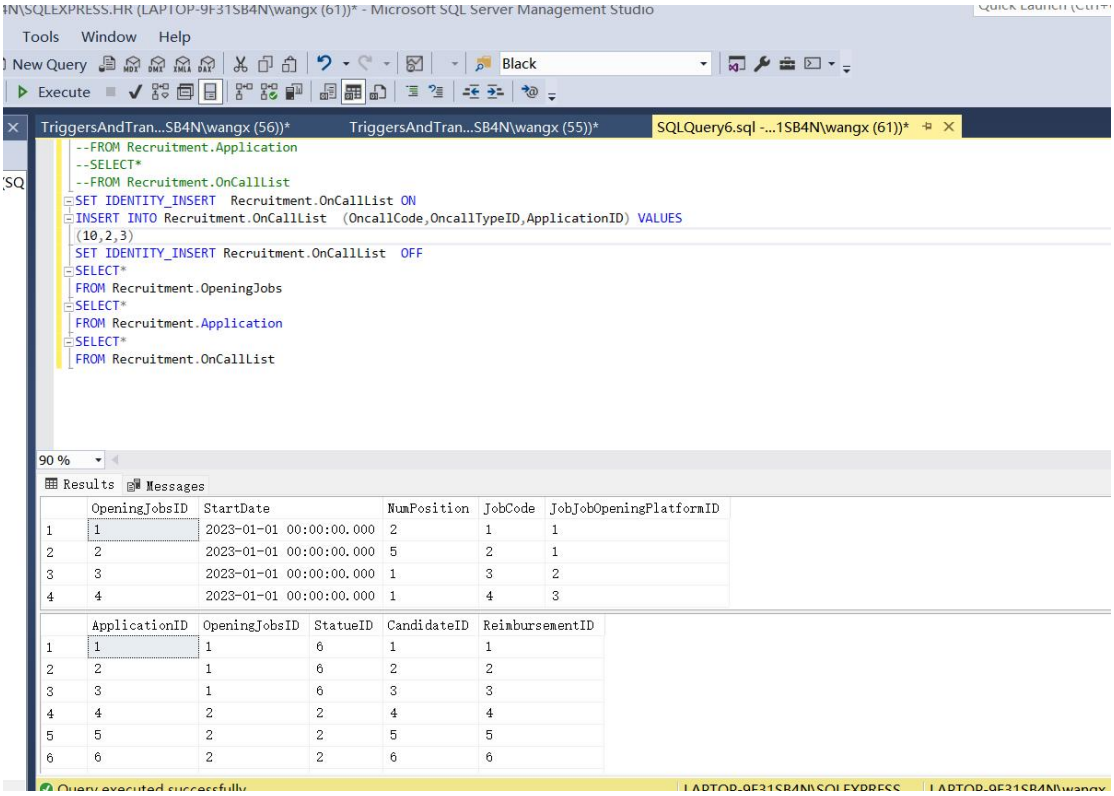

--SELECT*
--FROM Recruitment.OpeningJobs
--SELECT*
--FROM Recruitment.Application
--SELECT*
--FROM Recruitment.Onboarding
SET IDENTITY_INSERT Recruitment.Onboarding ON
INSERT INTO Recruitment.Onboarding (OnboardingID,ApplicationID,EducationCheck,Legitimacy) VALUES
(4,2,'PASS','PASS')
SET IDENTITY_INSERT Recruitment.Onboarding OFF
SELECT*
FROM Recruitment.OpeningJobs
SELECT*
FROM Recruitment.Application
Results
Messages
2 2 2 PASS PASS
3 3 9 PASS PASS
OpeningJobsID StartDate NumPosition JobCode JobJobOpeningPlatformID
1 2023-01-01 00:00:00.000 2 1 1
2 2023-01-01 00:00:00.000 3 2 1
3 2023-01-01 00:00:00.000 1 3 2
4 2023-01-01 00:00:00.000 1 4 3
ApplicationID OpeningJobsID StatueID CandidateID ReimbursementID
Query executed successfully.

2.4.5:

2.4.5.1

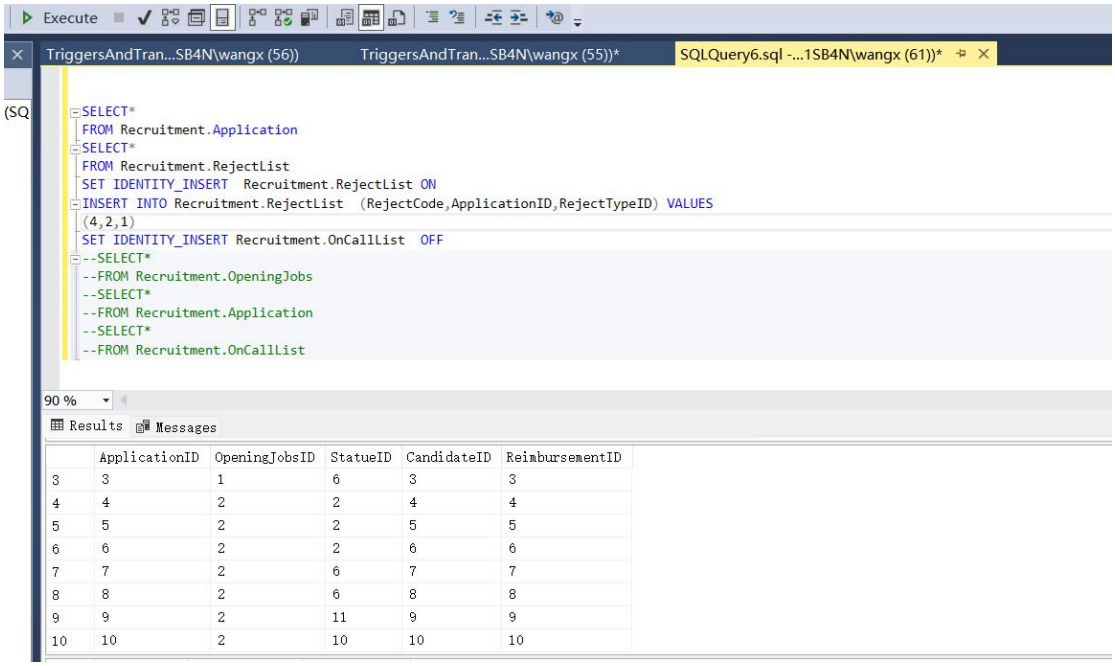


2.4.5.2



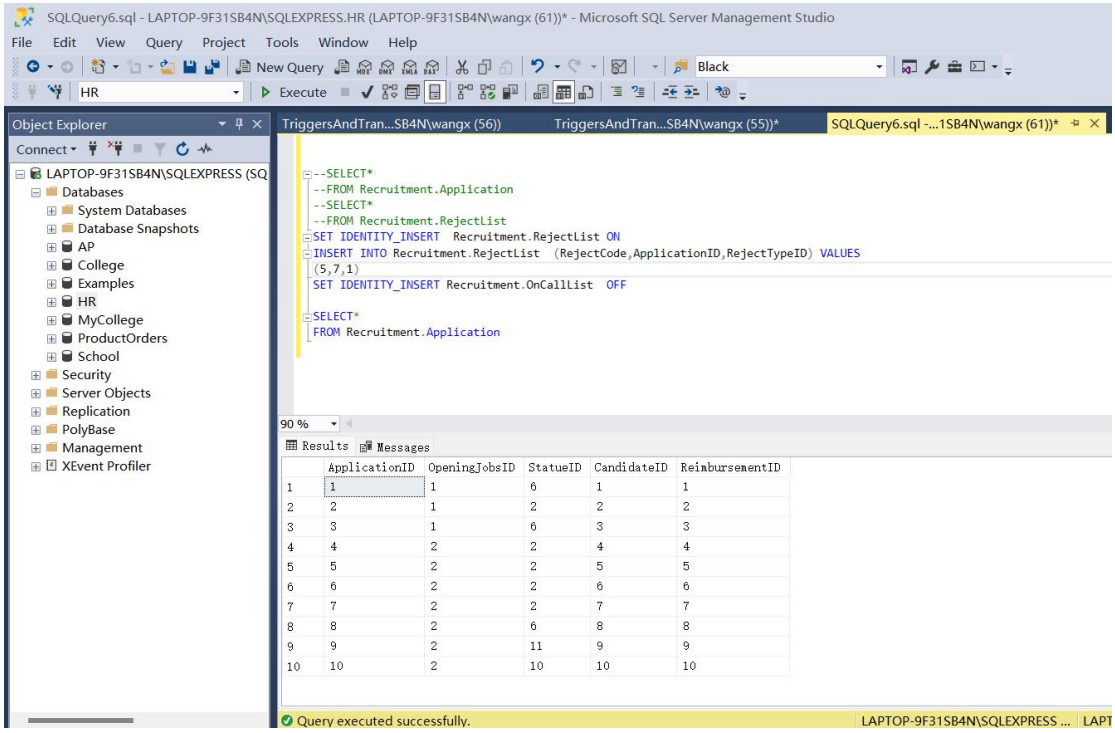
2.4.6:

2.4.6.1



	ApplicationID	OpeningJobsID	StatueID	CandidateID	ReimbursementID
3	3	1	6	3	3
4	4	2	2	4	4
5	5	2	2	5	5
6	6	2	2	6	6
7	7	2	6	7	7
8	8	2	6	8	8
9	9	2	11	9	9
10	10	2	10	10	10

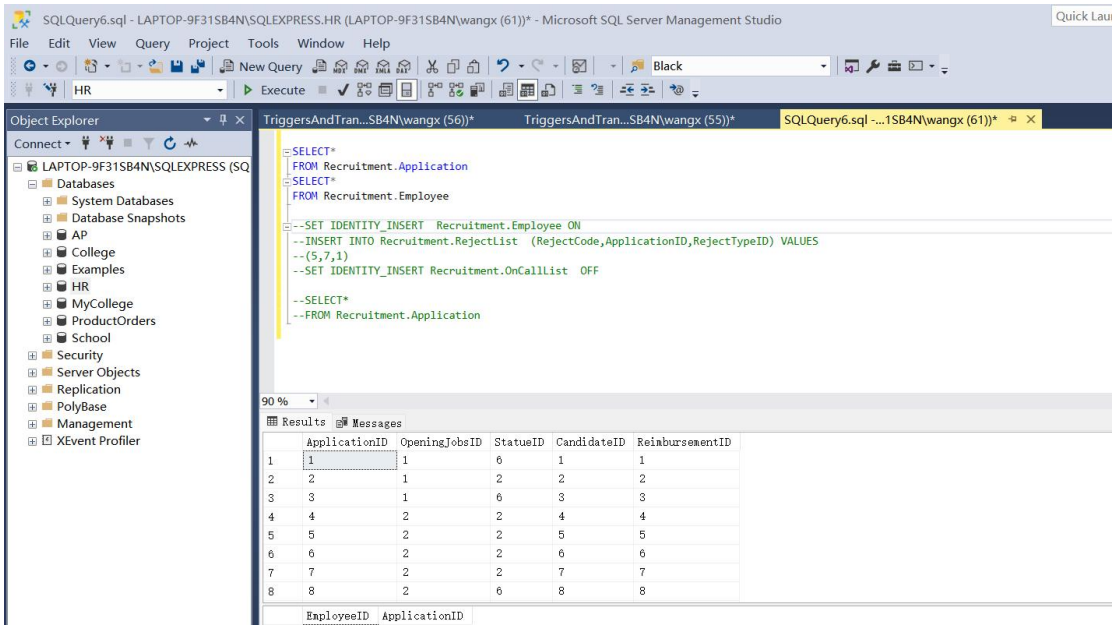
2.4.6.2



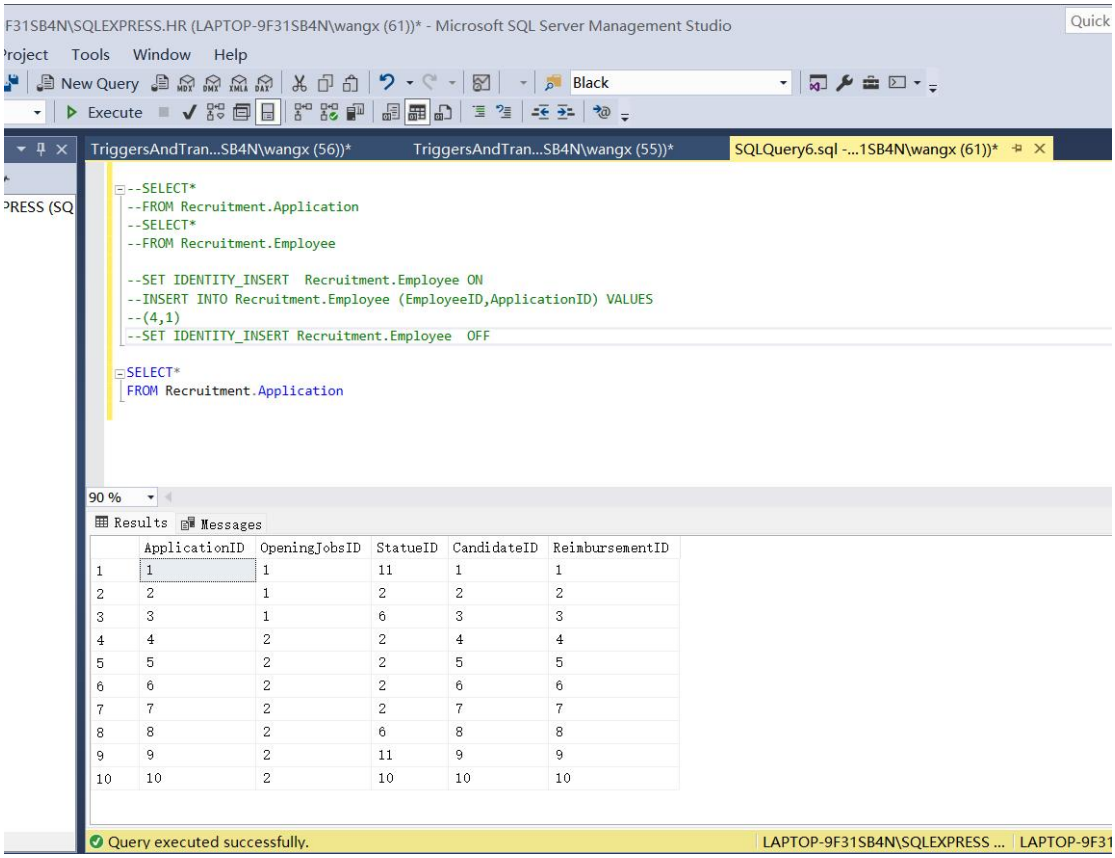
	ApplicationID	OpeningJobsID	StatueID	CandidateID	ReimbursementID
1	1	1	6	1	1
2	2	1	2	2	2
3	3	1	6	3	3
4	4	2	2	4	4
5	5	2	2	5	5
6	6	2	2	6	6
7	7	2	2	7	7
8	8	2	6	8	8
9	9	2	11	9	9
10	10	2	10	10	10

2.4.7:

2.4.7.1

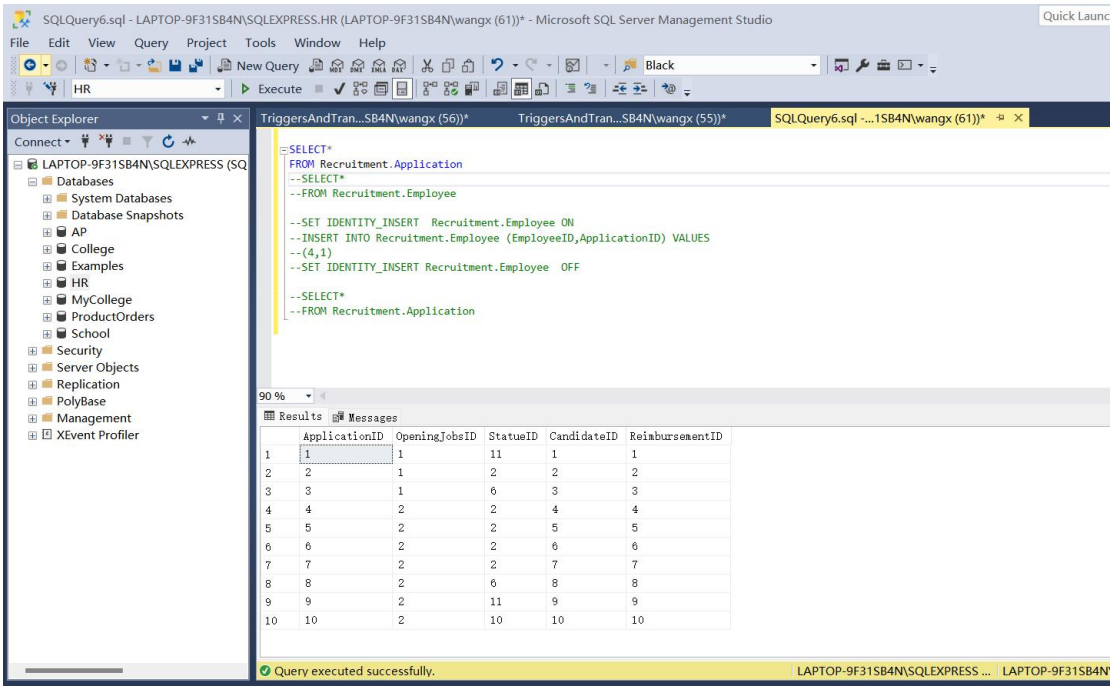


2.4.7.2



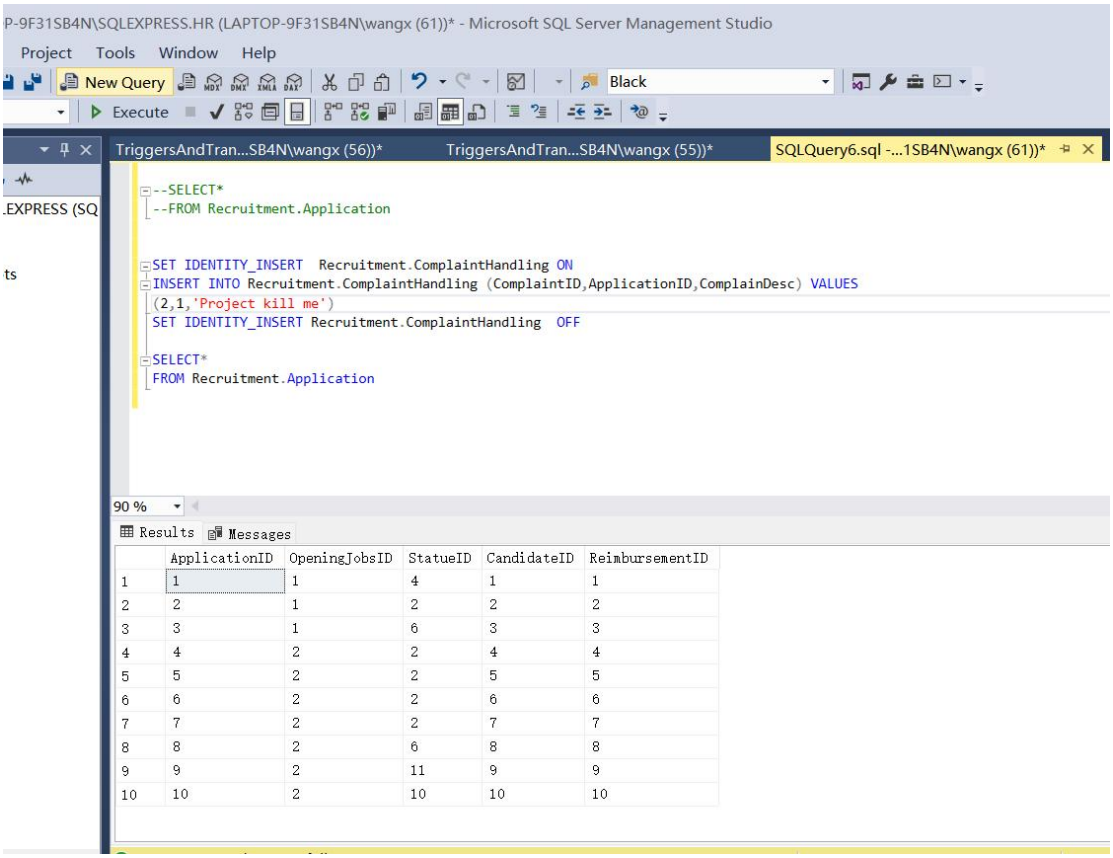
2.4.8:

2.4.8.1



	ApplicationID	OpeningJobsID	StatueID	CandidateID	ReimbursementID
1	1	1	11	1	1
2	2	1	2	2	2
3	3	1	6	3	3
4	4	2	2	4	4
5	5	2	2	5	5
6	6	2	2	6	6
7	7	2	2	7	7
8	8	2	6	8	8
9	9	2	11	9	9
10	10	2	10	10	10

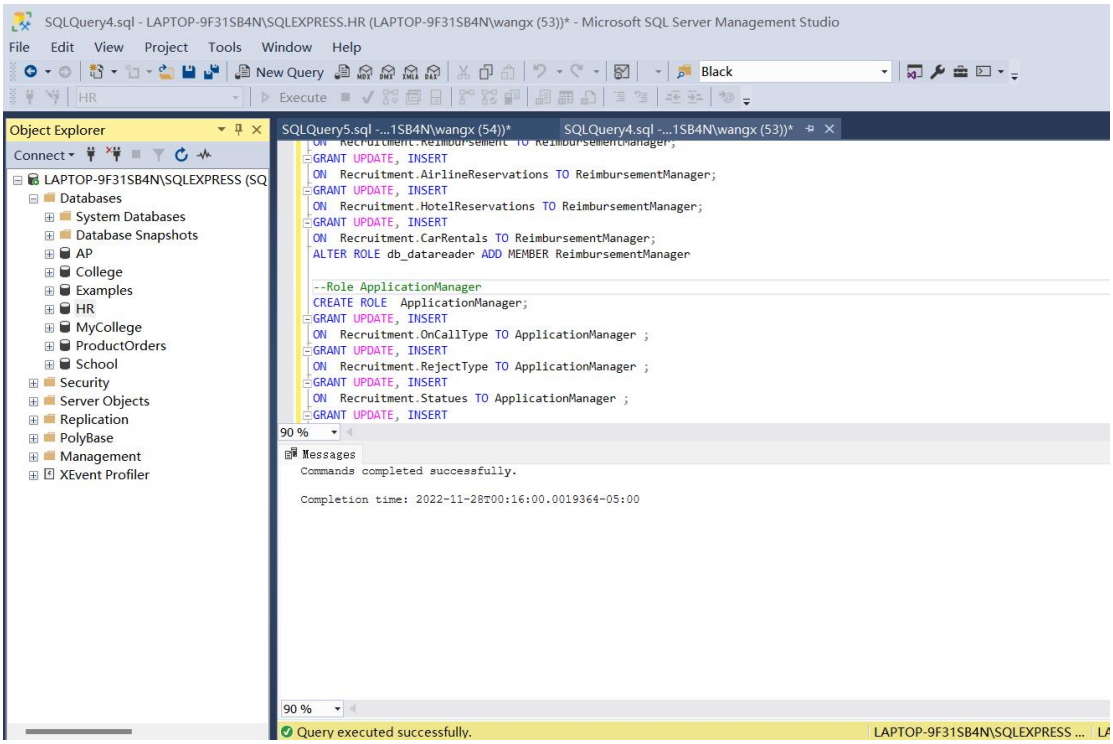
2.4.8.2



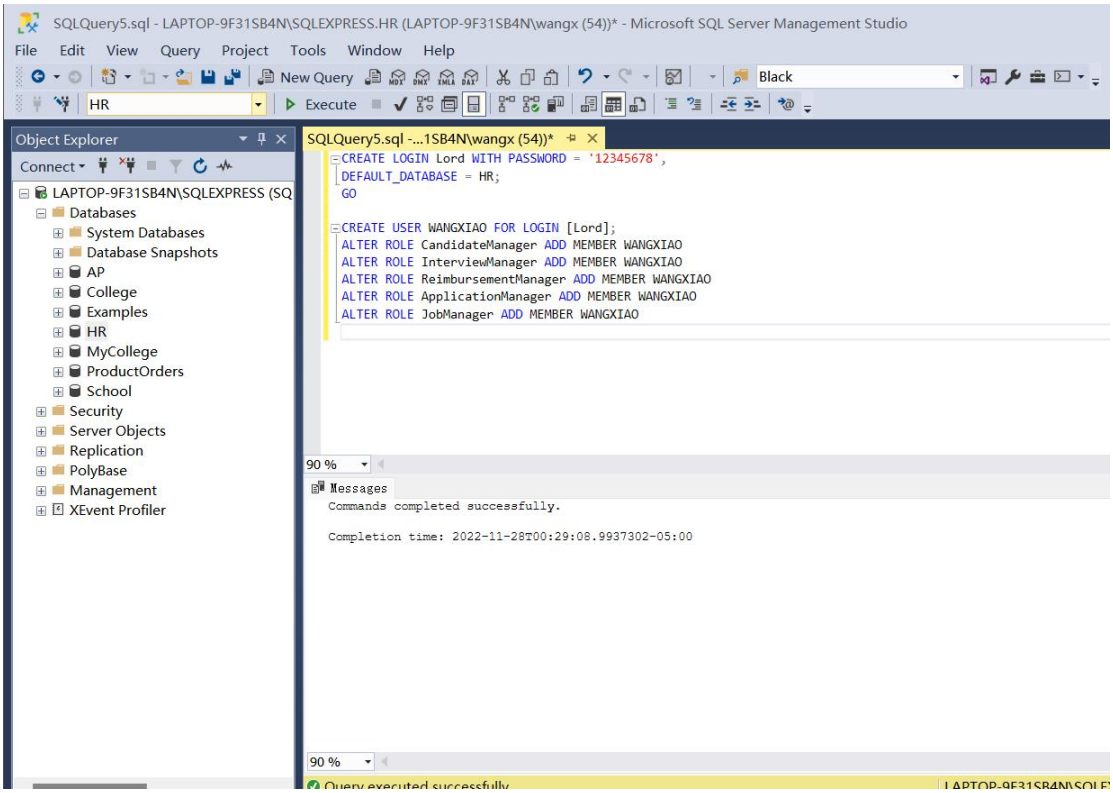
	ApplicationID	OpeningJobsID	StatueID	CandidateID	ReimbursementID
1	1	1	4	1	1
2	2	1	2	2	2
3	3	1	6	3	3
4	4	2	2	4	4
5	5	2	2	5	5
6	6	2	2	6	6
7	7	2	2	7	7
8	8	2	6	8	8
9	9	2	11	9	9
10	10	2	10	10	10

2.5 Server Security

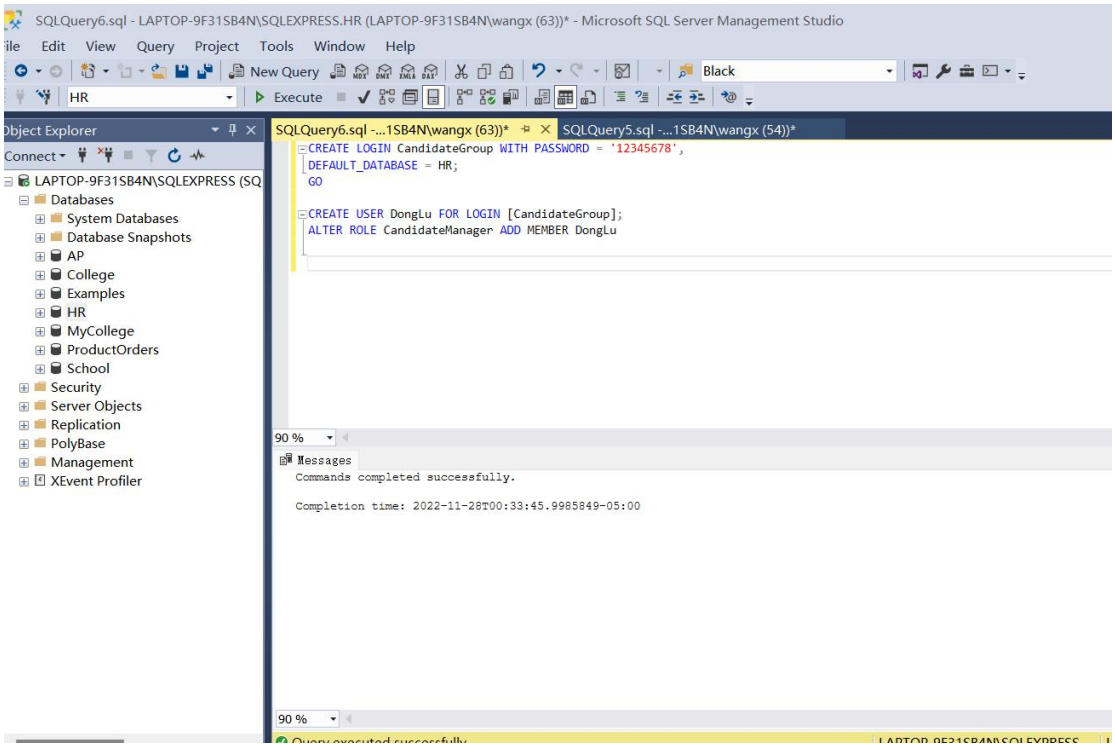
2.5.1



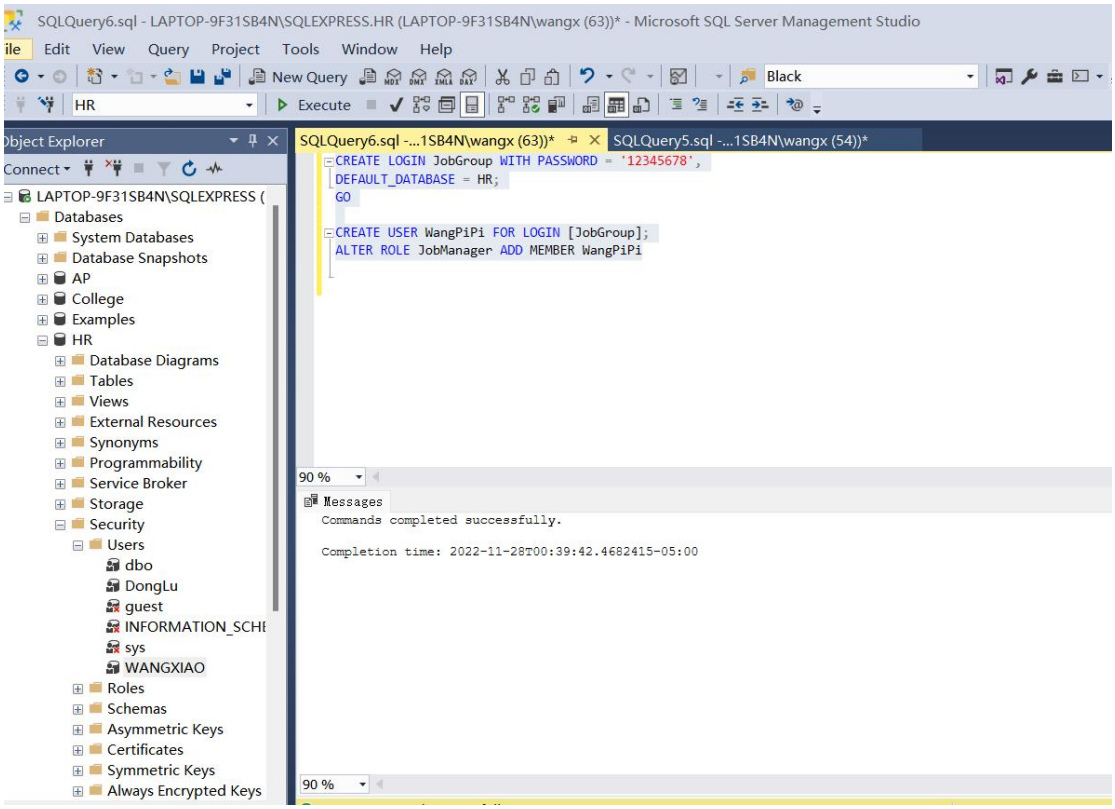
2.5.2



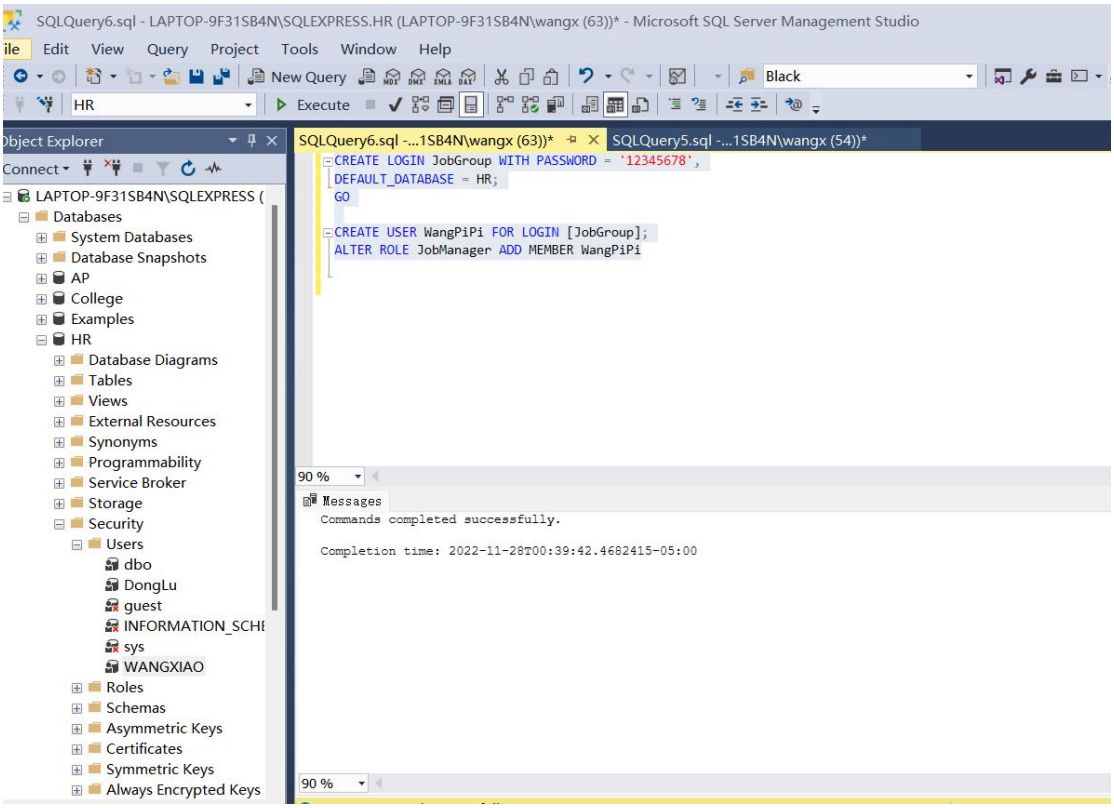
2.5.3


SQLQuery6.sql - LAPTOP-9F31SB4N\SQLEXPRESS.HR (LAPTOP-9F31SB4N\wangx (63))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Execute
HR
Black
Object Explorer
Connect
LAPTOP-9F31SB4N\SQLEXPRESS (SQ
Databases
System Databases
Database Snapshots
AP
College
Examples
HR
MyCollege
ProductOrders
School
Security
Server Objects
Replication
PolyBase
Management
XEvent Profiler
SQLQuery6.sql -...1SB4N\wangx (63))*
SQLQuery5.sql -...1SB4N\wangx (54))*
CREATE LOGIN CandidateGroup WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO
CREATE USER DongLu FOR LOGIN [CandidateGroup];
ALTER ROLE CandidateManager ADD MEMBER DongLu
90 %
Messages
Commands completed successfully.
Completion time: 2022-11-28T00:33:45.9985849-05:00
90 %

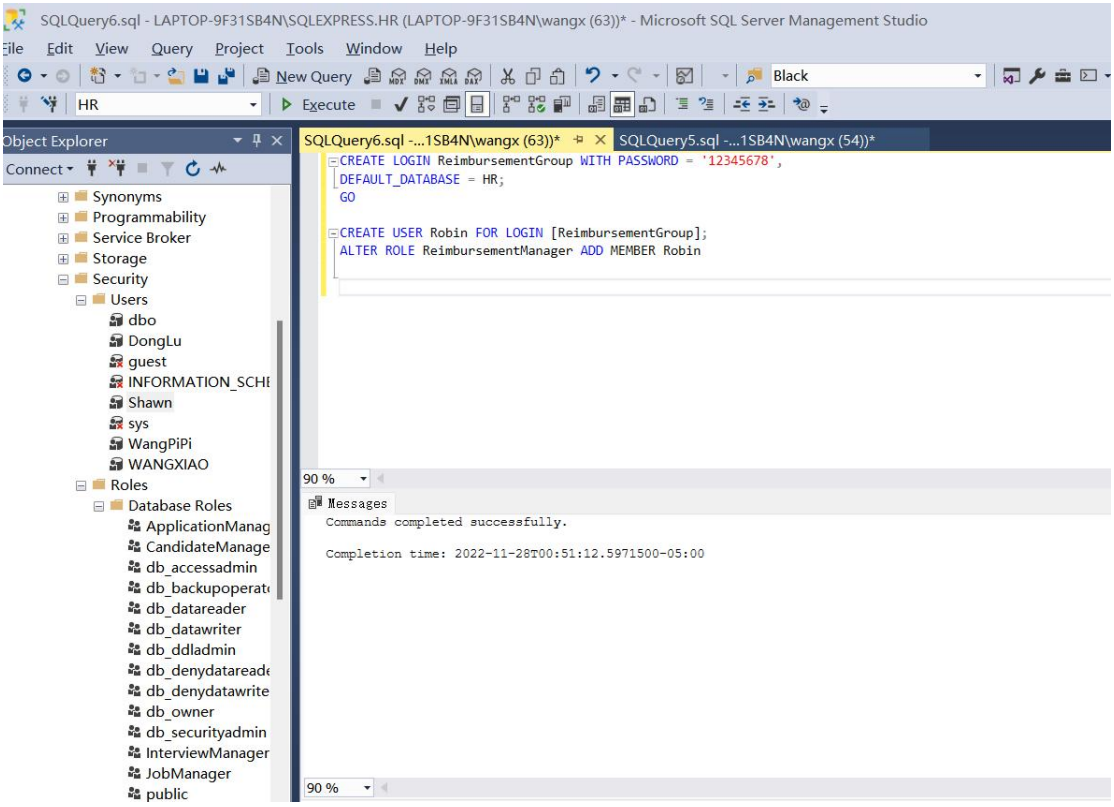
2.5.4


SQLQuery6.sql - LAPTOP-9F31SB4N\SQLEXPRESS.HR (LAPTOP-9F31SB4N\wangx (63))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Execute
HR
Black
Object Explorer
Connect
LAPTOP-9F31SB4N\SQLEXPRESS (
Databases
System Databases
Database Snapshots
AP
College
Examples
HR
Database Diagrams
Tables
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
Users
dbo
DongLu
guest
INFORMATION_SCHE
sys
WANGXIAO
Roles
Schemas
Asymmetric Keys
Certificates
Symmetric Keys
Always Encrypted Keys
SQLQuery6.sql -...1SB4N\wangx (63))*
SQLQuery5.sql -...1SB4N\wangx (54))*
CREATE LOGIN JobGroup WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO
CREATE USER WangPiPi FOR LOGIN [JobGroup];
ALTER ROLE JobManager ADD MEMBER WangPiPi
90 %
Messages
Commands completed successfully.
Completion time: 2022-11-28T00:39:42.4682415-05:00
90 %

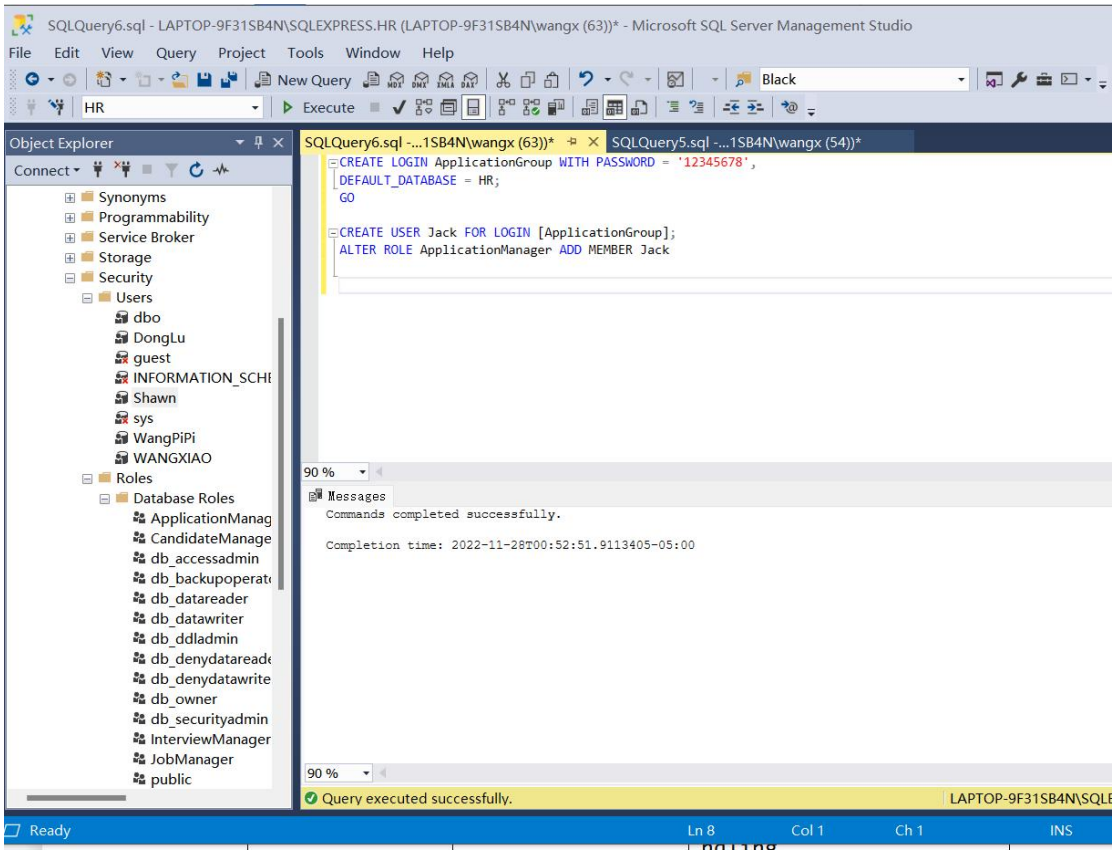
2.5.5


SQLQuery6.sql - LAPTOP-9F31SB4N\SQLEXPRESS.HR (LAPTOP-9F31SB4N\wangx (63))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Black
HR
Execute
Object Explorer
Connect
LAPTOP-9F31SB4N\SQLEXPRESS (
Databases
System Databases
Database Snapshots
AP
College
Examples
HR
Database Diagrams
Tables
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
Users
dbo
DongLu
guest
INFORMATION_SCHE
sys
WANGXIAO
Roles
Schemas
Asymmetric Keys
Certificates
Symmetric Keys
Always Encrypted Keys
SQLQuery6.sql -...1SB4N\wangx (63))*
SQLQuery5.sql -...1SB4N\wangx (54))*
CREATE LOGIN JobGroup WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO
CREATE USER WangPiPi FOR LOGIN [JobGroup];
ALTER ROLE JobManager ADD MEMBER WangPiPi
90 %
Messages
Commands completed successfully.
Completion time: 2022-11-28T00:39:42.4682415-05:00
90 %

2.5.6


SQLQuery6.sql - LAPTOP-9F31SB4N\SQLEXPRESS.HR (LAPTOP-9F31SB4N\wangx (63))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Black
HR
Execute
Object Explorer
Connect
Synonyms
Programmability
Service Broker
Storage
Security
Users
dbo
DongLu
guest
INFORMATION_SCHE
Shawn
sys
WangPiPi
WANGXIAO
Roles
Database Roles
ApplicationManag
CandidateManage
db_accessadmin
db_backupoperat
db_datareader
db_datawriter
db_ddladmin
db_denydatareade
db_denydatawrite
db_owner
db_securityadmin
InterviewManager
JobManager
public
SQLQuery6.sql -...1SB4N\wangx (63))*
SQLQuery5.sql -...1SB4N\wangx (54))*
CREATE LOGIN ReimbursementGroup WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO
CREATE USER Robin FOR LOGIN [ReimbursementGroup];
ALTER ROLE ReimbursementManager ADD MEMBER Robin
90 %
Messages
Commands completed successfully.
Completion time: 2022-11-28T00:51:12.5971500-05:00
90 %

2.5.7


SQLQuery6.sql - LAPTOP-9F31SB4N\SQLEXPRESS.HR (LAPTOP-9F31SB4N\wangx (63))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Black
HR
Execute
Object Explorer
Connect
Synonyms
Programmability
Service Broker
Storage
Security
Users
dbo
DongLu
guest
INFORMATION_SCHE
Shawn
sys
WangPiPi
WANGXIAO
Roles
Database Roles
ApplicationManag
CandidateManage
db_accessadmin
db_backupoperat
db_datareader
db_datawriter
db_ddladmin
db_denydataread
db_denydatawrite
db_owner
db_securityadmin
InterviewManager
JobManager
public
SQLQuery6.sql -...1SB4N\wangx (63))*
SQLQuery5.sql -...1SB4N\wangx (54))*
CREATE LOGIN ApplicationGroup WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO
CREATE USER Jack FOR LOGIN [ApplicationGroup];
ALTER ROLE ApplicationManager ADD MEMBER Jack
90 %
Messages
Commands completed successfully.
Completion time: 2022-11-28T00:52:51.9113405-05:00
90 %
Query executed successfully.
LAPTOP-9F31SB4N\SQLE
Ready
Ln 8
Col 1
Ch 1
INS

All CODES

Create Database

```
USE master
GO

IF SCHEMA_ID('Recruitment') IS NOT NULL
DROP SCHEMA Recruitment
GO

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


CREATE DATABASE HR
GO

USE HR
GO

CREATE SCHEMA Recruitment
GO



CREATE TABLE  Recruitment.Candidates
(
	CandidateID int IDENTITY NOT NULL,
	CandidateFName  varchar(100) NOT NULL,
	CandidateLName varchar(100) NOT NULL,
	--constraint and performance
	CONSTRAINT PK_Candidates PRIMARY KEY CLUSTERED(CandidateID)
)

CREATE TABLE  Recruitment.CandidatesInfo
(
	CandidateInfoID int IDENTITY NOT NULL,
	Email varchar(100) NOT NULL,
	Phone varchar(100) NOT NULL,
	ShortProfile varchar (200) NULL,
	Citizenship varchar(100) NOT NULL,
	Education varchar(100) NOT NULL,
	LocationInfo varchar(100) NOT NULL,
	CandidateID int NOT NULL
	--constraint and performance
	CONSTRAINT PK_CandidatesInfo PRIMARY KEY CLUSTERED(CandidateInfoID)
	--FK Address potential integrity
	CONSTRAINT FK_CandidatesInfo_Candidates FOREIGN KEY(CandidateID)
REFERENCES Recruitment.Candidates(CandidateID)
)

CREATE TABLE  Recruitment.CandidatesDocuments
```

```
(
	CandidateDocumentID int IDENTITY NOT NULL,
	CV varchar(2000) NOT NULL,
	ReferenceLetter varchar(2000)NULL,
	CoverLetter varchar (2000) NULL,
	CandidateID int NOT NULL
	--constraint and performance
	CONSTRAINT PK_CandidatesDocuments PRIMARY KEY CLUSTERED(CandidateDocumentID)
	--FK Address potential integrity
	CONSTRAINT FK_CandidatesDocuments_Candidates FOREIGN KEY(CandidateID) REFERENCES Recruitment.Candidates(CandidateID)
)


CREATE TABLE Recruitment.JobTypes (
  JobTypeID int IDENTITY NOT NULL,
  JobType nvarchar(50)  NOT NULL,
  JobTypeDescription nvarchar(200)  NOT NULL,
  CONSTRAINT JobTypes_pk PRIMARY KEY  (JobTypeID)
);

CREATE TABLE Recruitment.JobMediums  (
  JobMediumID int IDENTITY  NOT NULL,
  JobMedium nvarchar(50)  NOT NULL,
  JobMediumDescription varchar(100)  NOT NULL,
  CONSTRAINT JobMediums_pk PRIMARY KEY  (JobMediumID)
);

CREATE TABLE Recruitment.JobCategories (
  JobCategoryID int IDENTITY NOT NULL,
  JobCategory nvarchar(50)  NOT NULL,
  JobCategoryDescription nvarchar(200)  NOT NULL,
  CONSTRAINT JobCategories_pk PRIMARY KEY  (JobCategoryID)
);
CREATE TABLE Recruitment.JobPositions  (
  JobPositionID int IDENTITY NOT NULL,
  JobPosition nvarchar(50)  NOT NULL,
  JobMediumDescription nvarchar(100)  NOT NULL,
  CONSTRAINT JobPositions_pk PRIMARY KEY  (JobPositionID)
);
CREATE TABLE Recruitment.JobOpeningPlatforms (
  JobOpeningPlatformID int IDENTITY NOT NULL,
  PlatformName nvarchar(50)  NOT NULL,
  CONSTRAINT JobOpeningPlatforms_pk PRIMARY KEY  (JobOpeningPlatformID)
);

CREATE TABLE Recruitment.Jobs (
  JobCode int IDENTITY NOT NULL,
  JobName nvarchar(50)  NOT NULL,
  JobDescription nvarchar(200)  NOT NULL,
  JobTypeID int  NOT NULL,
  JobCategoryID int  NOT NULL,
  JobPositionID int  NOT NULL,
  JobMediumID int  NOT NULL,
  CONSTRAINT Jobs_pk PRIMARY KEY CLUSTERED (JobCode),--Increase performance
  CONSTRAINT FK_Jobs_JobTypes FOREIGN KEY(JobTypeID) REFERENCES
```

```
Recruitment.JobTypes(JobTypeID),--interity constrains
  CONSTRAINT FK_Jobs_JobCategories FOREIGN KEY(JobCategoryID) REFERENCES
Recruitment.JobCategories(JobCategoryID),
  CONSTRAINT FK_Jobs_JobPositions FOREIGN KEY(JobPositionID) REFERENCES
Recruitment.JobPositions(JobPositionID),
  CONSTRAINT FK_Jobs_JobMediums FOREIGN KEY(JobMediumID) REFERENCES
Recruitment.JobMediums(JobMediumID)
);

CREATE TABLE Recruitment.OpeningJobs (
  OpeningJobsID  int IDENTITY NOT NULL,
  StartDate  datetime  NOT NULL,
  NumPosition  int  NOT NULL CHECK(NumPosition>=0),--Avoid mistake
  JobCode int  NOT NULL,
  JobJobOpeningPlatformID int  NOT NULL,
  CONSTRAINT OpeningJobs_pk PRIMARY KEY CLUSTERED (OpeningJobsID ),
  CONSTRAINT OpeningJobs_Jobs FOREIGN KEY (JobCode) REFERENCES
Recruitment.Jobs (JobCode),
  CONSTRAINT OpeningJobs_JobOpeningPlatforms FOREIGN KEY
(JobJobOpeningPlatformID) REFERENCES Recruitment.JobOpeningPlatforms
(JobOpeningPlatformID)
);


CREATE TABLE Recruitment.Interviewers (
  InterviewerID int  IDENTITY  NOT NULL,
  InviewerFname nvarchar(50)  NOT NULL,
  InviewerLname nvarchar(50)  NOT NULL,
  Titles  nvarchar(100)  NOT NULL,
  CONSTRAINT Interviewers_pk PRIMARY KEY CLUSTERED (InterviewerID)
);

CREATE TABLE Recruitment.TestTypes (
  TestTypeID int IDENTITY NOT NULL,
  TestType nvarchar(50)  NOT NULL,
  CONSTRAINT TestTypes_pk PRIMARY KEY  (TestTypeID)
);


CREATE TABLE Recruitment.Tests (
  TestID int  IDENTITY  NOT NULL,
  TestStartTime datetime  NOT NULL,
  TestEendTime datetime  NOT NULL,
  TestTittle nvarchar(50)  NOT NULL ,
  TestTypeID int  NOT NULL,
  CONSTRAINT Test_pk PRIMARY KEY  (TestID),
  CONSTRAINT Test_TestTypes FOREIGN KEY (TestTypeID) REFERENCES
Recruitment.TestTypes (TestTypeID)
);

CREATE TABLE Recruitment.InterviewTypes (
  InterviewTypeID int IDENTITY    NOT NULL,
  InterviewType nvarchar(50)  NOT NULL,
  CONSTRAINT InterviewTypes_pk PRIMARY KEY  (InterviewTypeID)
);

CREATE TABLE Recruitment.Interviews (
```

```
  InterviewID int IDENTITY  NOT NULL,
  InterviewStartTime datetime  NOT NULL,
  InterviewEndTime datetime  NOT NULL,
  InterviewLocation nvarchar(50)  NOT NULL,
  InterviewTypeID int  NOT NULL,
  TestID int  NOT NULL,
  InterviewerID int  NOT NULL,
  CONSTRAINT Interviews_pk PRIMARY KEY CLUSTERED (InterviewID),
  CONSTRAINT  Interviews_Test  FOREIGN  KEY  (TestID)  REFERENCES  Recruitment.Tests
(TestID),
  CONSTRAINT Interviews_InterviewTypes FOREIGN KEY (InterviewTypeID)REFERENCES
Recruitment.InterviewTypes (InterviewTypeID),
  CONSTRAINT  Interviews_Interviewers  FOREIGN  KEY  (InterviewerID)  REFERENCES
Recruitment.Interviewers (InterviewerID)
);

CREATE TABLE Recruitment.Reimbursement (
  ReimbursementID int IDENTITY   NOT NULL,
  SpendTotal int  NOT NULL CHECK(SpendTotal >=0),
  CONSTRAINT Reimbursement_pk PRIMARY KEY  (ReimbursementID)
);

CREATE TABLE  Recruitment.AirlineReservations (
  AirlineReservationID int IDENTITY   NOT NULL,
  AirlineName nvarchar(50)  NOT NULL,
  AirlineTrackNumber nvarchar(50)  NOT NULL,
  AirLineCost int  NOT NULL CHECK(AirLineCost >=0),
  ReimbursementID int  NOT NULL,
  CONSTRAINT  AirlineReservations_pk PRIMARY KEY  (AirlineReservationID),
  CONSTRAINT    AirlineReservations_Reimbursement  FOREIGN  KEY  (ReimbursementID)
REFERENCES Recruitment.Reimbursement (ReimbursementID)
);
CREATE TABLE Recruitment.HotelReservations (
  HotelReservationID  int IDENTITY    NOT NULL,
  HotelName nvarchar(50)  NOT NULL,
  HotelTrackNumber nvarchar(50)  NOT NULL,
  HotelCost int  NOT NULL CHECK(HotelCost >=0),
  ReimbursementID int  NOT NULL,
  CONSTRAINT HotelReservations_pk PRIMARY KEY  (HotelReservationID ),
  CONSTRAINT  HotelReservations_Reimbursement  FOREIGN  KEY  (ReimbursementID)
REFERENCES Recruitment.Reimbursement (ReimbursementID)
);

CREATE TABLE Recruitment.CarRentals (
  CarRentalID int IDENTITY    NOT NULL,
  CarRentalName nvarchar(50)  NOT NULL,
  CarRentalTrackNumber nvarchar(50)  NOT NULL,
  CarRentCost int  NOT NULL CHECK(CarRentCost >=0),
  ReimbursementID int  NOT NULL,
  CONSTRAINT CarRentals_pk PRIMARY KEY  (CarRentalID),
  CONSTRAINT  CarRentals_Reimbursement  FOREIGN  KEY
(ReimbursementID)REFERENCES Recruitment.Reimbursement (ReimbursementID)
);



CREATE TABLE Recruitment.OnCallType (
```

```
  OncallTypeID int IDENTITY  NOT NULL,
  OncallReason nvarchar(50)  NOT NULL,
  CONSTRAINT OnCallType_pk PRIMARY KEY  (OncallTypeID)
);

CREATE TABLE Recruitment.RejectType (
  RejectTypeID int IDENTITY  NOT NULL,
  RejectType nvarchar(50)  NOT NULL,
  CONSTRAINT RejectType_pk PRIMARY KEY  (RejectTypeID)
);

CREATE TABLE Recruitment.Statues (
  StatueID int IDENTITY NOT NULL,
  StatueName nvarchar(50)  NOT NULL,
  CONSTRAINT Statues_pk PRIMARY KEY  (StatueID)
);

CREATE TABLE Recruitment.Application (
  ApplicationID int IDENTITY NOT NULL,
  OpeningJobsID  int  NOT NULL,
  StatueID int  NOT NULL,
  CandidateID  int  NOT NULL,
  ReimbursementID int  NOT NULL,
  CONSTRAINT Application_pk PRIMARY KEY CLUSTERED (ApplicationID),
  CONSTRAINT Application_OpeningJobs FOREIGN KEY (OpeningJobsID ) REFERENCES
Recruitment.OpeningJobs (OpeningJobsID ),
  CONSTRAINT Application_Statues FOREIGN KEY (StatueID) REFERENCES
Recruitment.Statues (StatueID),
  CONSTRAINT Application_Candidates FOREIGN KEY (CandidateID )REFERENCES
Recruitment.Candidates (CandidateID ),
  CONSTRAINT Application_Reimbursement FOREIGN KEY (ReimbursementID)
REFERENCES Recruitment.Reimbursement (ReimbursementID)

);

CREATE TABLE Recruitment.OnCallList (
  OncallCode int IDENTITY  NOT NULL,
  OncallTypeID int  NOT NULL,
  ApplicationID int  NOT NULL,
  CONSTRAINT OnCallList_pk PRIMARY KEY  (OncallCode),
  CONSTRAINT OnCall_OnCallType FOREIGN KEY (OncallTypeID)REFERENCES
Recruitment.OnCallType (OncallTypeID),
  CONSTRAINT OnCallList_Application FOREIGN KEY (ApplicationID) REFERENCES
Recruitment.Application (ApplicationID)
);

CREATE TABLE Recruitment.Onboarding (
  OnboardingID int IDENTITY  NOT NULL,
  ApplicationID int  NOT NULL,
  EducationCheck nvarchar(50)  NOT NULL,
  Legitimacy nvarchar(50)  NOT NULL,
  CONSTRAINT Onboarding_pk PRIMARY KEY  (OnboardingID),
  CONSTRAINT Onboarding_Application FOREIGN KEY (ApplicationID) REFERENCES
Recruitment.Application (ApplicationID)
);
```

```
CREATE TABLE Recruitment.RejectList (
  RejectCode int IDENTITY  NOT NULL,
  ApplicationID int  NOT NULL,
  RejectTypeID int  NOT NULL,
  CONSTRAINT RejectList_pk PRIMARY KEY  (RejectCode),
  CONSTRAINT  RejectList_RejectType  FOREIGN  KEY  (RejectTypeID)  REFERENCES
Recruitment.RejectType (RejectTypeID),
  CONSTRAINT  RejectList_Application  FOREIGN  KEY  (ApplicationID)  REFERENCES
Recruitment.Application (ApplicationID)
  )


CREATE TABLE Recruitment.BlackListed (
  BlackListedID int IDENTITY NOT NULL,
  ApplicationID int  NOT NULL,
  CONSTRAINT BlackListed_pk PRIMARY KEY  (BlackListedID),
  CONSTRAINT  BlackListed_Application  FOREIGN  KEY  (ApplicationID)  REFERENCES
Recruitment.Application (ApplicationID)
);


CREATE TABLE Recruitment.Employee (
  EmployeeID int IDENTITY NOT NULL,
  ApplicationID int  NOT NULL,
  CONSTRAINT Employee_pk PRIMARY KEY  (EmployeeID),
  CONSTRAINT  Employee_Application  FOREIGN  KEY  (ApplicationID)  REFERENCES
Recruitment.Application (ApplicationID)
);

CREATE TABLE Recruitment.ComplaintHandling (
  ComplaintID int IDENTITY NOT NULL,
  ApplicationID int  NOT NULL,
  ComplainDesc nvarchar(2000)  NOT NULL,
  CONSTRAINT ComplaintHandling_pk PRIMARY KEY  (ComplaintID),
  CONSTRAINT  ComplaintHandling_Application  FOREIGN  KEY  (ApplicationID)
REFERENCES Recruitment.Application (ApplicationID)
);

CREATE TABLE Recruitment.Evaluation (
  EvaluationID int IDENTITY NOT NULL,
  InterviewID int  NOT NULL,
  ApplicationID int  NOT NULL,
  GRADE int  NOT NULL CHECK(GRADE>=0 and GRADE<=100) ,
  CONSTRAINT Evaluation_pk PRIMARY KEY CLUSTERED (EvaluationID),
  CONSTRAINT  Evaluation_Application  FOREIGN  KEY  (ApplicationID)  REFERENCES
Recruitment.Application (ApplicationID),
  CONSTRAINT  Evaluation_Interviews  FOREIGN  KEY  (InterviewID)  REFERENCES
Recruitment.Interviews (InterviewID)
);


CREATE TABLE Recruitment.Review (
  ReviewID int  IDENTITY NOT NULL,
  EvaluationID int  NOT NULL,
  CandReview nvarchar(2000)   NULL,
  InterReview nvarchar(2000)   NULL,
```

```
  CONSTRAINT Review_pk PRIMARY KEY  (ReviewID),
  CONSTRAINT Review_Evaluation FOREIGN KEY (EvaluationID) REFERENCES Recruitment.Evaluation (EvaluationID)
);
```

InsertData

```
USE HR;

SET IDENTITY_INSERT  Recruitment.Candidates ON
INSERT INTO Recruitment.Candidates(CandidateID,CandidateFName,CandidateLName) VALUES
(1,'Wang','Xiao'),
(2,'Wang','YuChen'),
(3,'He','XiangYu'),
(4,'Krishna','Jash'),
(5,'Robbin','Rosan'),
(6,'John','Cena'),
(7,'Tianlu','Jacoob'),
(8,'MengNan','Hong'),
(9,'XiaoXiao','Peng'),
(10,'MengYing','Wang')

SET IDENTITY_INSERT Recruitment.Candidates OFF

SET IDENTITY_INSERT  Recruitment.CandidatesInfo ON
INSERT INTO Recruitment.CandidatesInfo(CandidateInfoID,Email,Phone,ShortProfile,Citizenship,Education,LocationInfo,CandidateID) VALUES
(1,'xwang99@syr.edu','375-88689','C++ Expert & OS design','CHN','Master','NY',1),
(2,'yuchen@syr.edu','375-88690','Java Expert','CHN','Master','NY',2),
(3,'xiangyu@syr.edu','375-88691','C++ Expert','CHN','Master','NY',3),
(4,'Donglu@bufflo.edu','375-88692','AI& DeepLearning','CHN','Doctor','NY',4),
(5,'Krishna@huawei.edu','375-88693','C Expert','IND','HighSchool','MASS',5),
(6,'John@gmail.com','375-88694','C++ Expert','USA','UnderGraduate','NY',6),
(7,'tianlu@bentley.edu','375-89689','C++ Expert','USA','UnderGraduate','NY',7),
(8,'scarlletHong@bentley.edu','375-88989','C++ Expert','CHN','UnderGraduate','NY',8),
(9,'xiaoxiao@syr.edu','375-88289','C++ Expert','CHN','Master','CA',9),
(10,'mengying@syr.edu','375-81689','C++ Expert','CHN','Master','CA',10)
SET IDENTITY_INSERT Recruitment.CandidatesInfo OFF

SET IDENTITY_INSERT  Recruitment.CandidatesDocuments ON
INSERT INTO Recruitment.CandidatesDocuments(CandidateDocumentID,CV,ReferenceLetter,CoverLetter,CandidateID) VALUES
(1,'url//resume_abcdefghijklmn123456781','url//RefenceLetter_abcdefghijklmn123456780','url//coverletter_abcdefghijklmn123456780',1),
(2,'url//resume_abcdefghijklmn123456782','url//RefenceLetter_abcdefghijklmn123456781','url//coverletter_abcdefghijklmn123456781',2),
(3,'url//resume_abcdefghijklmn123456783','url//RefenceLetter_abcdefghijklmn123456782','url//coverletter_abcdefghijklmn123456782',3),
(4,'url//resume_abcdefghijklmn123456784','url//RefenceLetter_abcdefghijklmn123456783','url//coverletter_abcdefghijklmn123456783',4),
(5,'url//resume_abcdefghijklmn123456785','url//RefenceLetter_abcdefghijklmn123456784','url//coverletter_abcdefghijklmn123456784',5),
```

```
(6,'url//resume_abcdefghijklmn123456786','url//RefenceLetter_abcdefghijklmn123456785','url//co
verletter_abcdefghijklmn123456785',6),
(7,'url//resume_abcdefghijklmn123456787','url//RefenceLetter_abcdefghijklmn123456786','url//co
verletter_abcdefghijklmn123456786',7),
(8,'url//resume_abcdefghijklmn123456788','url//RefenceLetter_abcdefghijklmn123456787','url//co
verletter_abcdefghijklmn123456787',8),
(9,'url//resume_abcdefghijklmn123456789','url//RefenceLetter_abcdefghijklmn123456788','url//co
verletter_abcdefghijklmn123456788',9),
(10,'url//resume_abcdefghijklmn123456780','url//RefenceLetter_abcdefghijklmn12345679','url//co
verletter_abcdefghijklmn123456789',10)
SET IDENTITY_INSERT Recruitment.CandidatesDocuments OFF

SET IDENTITY_INSERT  Recruitment.JobTypes ON
INSERT INTO Recruitment.JobTypes(JobTypeID,JobType,JobTypeDescription) VALUES
(1,'Summer Internship','Candidate works for 2- 3 months as intern'),
(2,'Full Time ','Candidate must work at least 9 hours per day'),
(3,'Part Time','Candidate must work at most 3 hours per day'),
(4,'Contract','Candidate sign with the company and get paid based on workload')
SET IDENTITY_INSERT Recruitment.JobTypes OFF

SET IDENTITY_INSERT  Recruitment.JobCategories ON
INSERT  INTO  Recruitment.JobCategories(JobCategoryID,JobCategory,JobCategoryDescription)
VALUES
(1,'IT','installing, maintaining and repairing hardware & software components of the organization
computers'),
(2,'Software Design ','Transfer customer requirements into services'),
(3,'Software Test','responsible for the quality of software development and deployment')
SET IDENTITY_INSERT Recruitment.JobCategories OFF

SET IDENTITY_INSERT  Recruitment.JobMediums  ON
INSERT  INTO  Recruitment.JobMediums  (JobMediumID,JobMedium,JobMediumDescription)
VALUES
(1,'Online','Work at home'),
(2,'Onsite','Work at office')
SET IDENTITY_INSERT Recruitment.JobMediums OFF

SET IDENTITY_INSERT  Recruitment.JobPositions ON
INSERT  INTO  Recruitment.JobPositions  (JobPositionID,JobPosition,JobMediumDescription)
VALUES
(1,'Manager','10 years+ working experience'),
(2,'Senior','3 years+ working experience'),
(3,'Entry-level ','0-3 years working experience')
SET IDENTITY_INSERT Recruitment.JobPositions OFF

SET IDENTITY_INSERT  Recruitment.JobOpeningPlatforms ON
INSERT  INTO  Recruitment.JobOpeningPlatforms  (JobOpeningPlatformID,PlatformName)
VALUES
(1,'Company Webpage'),
(2,'LinkedIn'),
(3,'HandShake')
SET IDENTITY_INSERT Recruitment.JobOpeningPlatforms OFF

SET IDENTITY_INSERT  Recruitment.Jobs ON
INSERT                    INTO                    Recruitment.Jobs
(JobCode,JobName,JobDescription,JobTypeID,JobCategoryID,JobMediumID,JobPositionID)
VALUES
(1,'FullStack designer','Use c++ maintain the OS system and design websit',2,2,2,2),
```

```
(2,'Backend designer','Use x-86 architecture improve performance of the software',2,2,2,2),
(3,'Softer-ware test','Test softerware and design test cases',4,3,1,3),
(4,'IT Guys','Do some trivial works',1,1,2,3)
SET IDENTITY_INSERT Recruitment.Jobs OFF

SET IDENTITY_INSERT  Recruitment.OpeningJobs ON
INSERT INTO Recruitment.OpeningJobs (OpeningJobsID,StartDate,NumPosition,JobCode,JobJobOpeningPlatformID) VALUES
(1,'2023-1-1',2,1,1),
(2,'2023-1-1',2,2,1),
(3,'2023-1-1',1,3,2),
(4,'2023-1-1',1,4,3)

SET IDENTITY_INSERT Recruitment.OpeningJobs OFF

SET IDENTITY_INSERT  Recruitment.Interviewers ON
INSERT INTO Recruitment.Interviewers (InterviewerID,InviewerFname,InviewerLname,Titles) VALUES
(1,'Wang','Pipi','Chief Technology Officer'),
(2,'Ehat ','Ercanli','Technical Consulting'),
(3,'Lee','Robin','Chief Engineer')

SET IDENTITY_INSERT Recruitment.Interviewers  OFF

SET IDENTITY_INSERT  Recruitment.TestTypes ON
INSERT INTO Recruitment.TestTypes (TestTypeID ,TestType) VALUES
(1,'Onsite'),
(2,'Online')
SET IDENTITY_INSERT Recruitment.TestTypes  OFF

SET IDENTITY_INSERT  Recruitment.Tests ON
INSERT INTO Recruitment.Tests (TestID ,TestStartTime,TestEendTime,TestTittle,TestTypeID) VALUES
(1,'2022-11-15 09:00:00','2022-11-15 11:00:00','Programming1',1),
(2,'2022-11-15 14:00:00','2022-11-15 16:00:00','SystemDesign1',1),
(3,'2022-12-15 09:00:00','2022-12-15 11:00:00','Programming2',1),
(4,'2022-12-15 14:00:00','2022-12-15 16:00:00','SystemDesign2',1)
SET IDENTITY_INSERT Recruitment.Tests  OFF

SET IDENTITY_INSERT  Recruitment.InterviewTypes  ON
INSERT INTO Recruitment.InterviewTypes  (InterviewTypeID ,InterviewType) VALUES
(1,'Onsite'),
(2,'Online')
SET IDENTITY_INSERT  Recruitment.InterviewTypes   OFF

SET IDENTITY_INSERT  Recruitment.Interviews  ON
INSERT INTO Recruitment.Interviews (InterviewID  ,InterviewStartTime,InterviewEndTime,InterviewLocation,InterviewTypeID,TestID,
InterviewerID) VALUES
(1,'2022-11-15','2022-11-16','NY',1,1,1),
(2,'2022-11-15','2022-11-16','NY',1,1,2),
(3,'2022-11-15','2022-11-16','NY',1,1,3),
(4,'2022-11-15','2022-11-16','NY',1,2,1),
(5,'2022-11-15','2022-11-16','NY',1,2,2),
(6,'2022-11-15','2022-11-16','NY',1,2,3),
(7,'2022-12-15','2022-12-16','NY',1,3,1),
(8,'2022-12-15','2022-12-16','NY',1,3,2),
```

```
(9,'2022-12-15','2022-12-16','NY',1,3,3),
(10,'2022-12-15','2022-12-16','NY',1,4,1),
(11,'2022-12-15','2022-12-16','NY',1,4,2),
(12,'2022-12-15','2022-12-16','NY',1,4,3)
SET IDENTITY_INSERT  Recruitment.Interviews   OFF

SET IDENTITY_INSERT  Recruitment.Reimbursement ON
INSERT INTO Recruitment.Reimbursement (ReimbursementID,SpendTotal) VALUES
(1,0),
(2,0),
(3,0),
(4,0),
(5,0),
(6,0),
(7,0),
(8,0),
(9,700),
(10,9348)

SET IDENTITY_INSERT Recruitment.Reimbursement  OFF



SET IDENTITY_INSERT  Recruitment.AirlineReservations ON
INSERT INTO Recruitment.AirlineReservations
(AirlineReservationID,AirlineName,AirlineTrackNumber,AirLineCost,ReimbursementID)
VALUES
(1,'JetBlue','TN359462YH',300,9),
(2,'Delta','TN359432YH',330,10),
(3,'UltraLuxury','TN358888YH',1888,10)
SET IDENTITY_INSERT Recruitment.AirlineReservations  OFF

SET IDENTITY_INSERT  Recruitment.HotelReservations ON
INSERT INTO Recruitment.HotelReservations
(HotelReservationID,HotelName,HotelTrackNumber,HotelCost,ReimbursementID) VALUES
(1,'Hilton','HT359462YH',300,9),
(2,'Hilton','HT359432YH',310,10),
(3,'The Ranch at Rock Creek','HT358888YH',4700,10)
SET IDENTITY_INSERT Recruitment.HotelReservations  OFF

SET IDENTITY_INSERT  Recruitment.CarRentals ON
INSERT INTO Recruitment.CarRentals
(CarRentalID,CarRentalName,CarRentalTrackNumber,CarRentCost,ReimbursementID) VALUES
(1,'Toyota','CAR359462YH',100,9),
(2,'Toyota','CAR359432YH',120,10),
(3,'Benltley','CAR358888YH',2000,10)
SET IDENTITY_INSERT Recruitment.CarRentals  OFF

SET IDENTITY_INSERT  Recruitment.OnCallType ON
INSERT INTO Recruitment.OnCallType  (OncallTypeID,OncallReason) VALUES
(1,'Select but no more job openings '),
(2,'Onboarding  is unsuccessful'),
(3,'Onboarding  is unsuccessful')
SET IDENTITY_INSERT Recruitment.OnCallType  OFF

SET IDENTITY_INSERT  Recruitment.RejectType ON
INSERT INTO Recruitment.RejectType  (RejectTypeID,RejectType) VALUES
```

```
(1,'SApplication scanning is not met '),
(2,'Fail in the interview')
SET IDENTITY_INSERT Recruitment.RejectType  OFF

SET IDENTITY_INSERT  Recruitment.Statues ON
INSERT INTO Recruitment.Statues  (StatueID,StatueName) VALUES
(1,'Submitted'),
(2,'Reject'),
(3,'Select'),
(4,'Waiting'),
(5,'Re-interview'),
(6,'OnCall'),
(7,'Negotiating'),
(8,'Accept'),
(9,'Onboarding'),
(10,'BlackListed'),
(11,'Employee'),
(12,'NextRound')
SET IDENTITY_INSERT Recruitment.Statues  OFF


SET IDENTITY_INSERT  Recruitment.Application ON
INSERT INTO Recruitment.Application
(ApplicationID,OpeningJobsID,StatueID,CandidateID,ReimbursementID) VALUES
(1,1,11,1,1),
(2,1,11,2,2),
(3,1,6,3,3),
(4,2,2,4,4),
(5,2,2,5,5),
(6,2,2,6,6),
(7,2,6,7,7),
(8,2,6,8,8),
(9,2,11,9,9),
(10,2,10,10,10)

SET IDENTITY_INSERT Recruitment.Application  OFF

SET IDENTITY_INSERT  Recruitment.BlackListed ON
INSERT INTO Recruitment.BlackListed  (BlackListedID,ApplicationID) VALUES
(1,1)
SET IDENTITY_INSERT Recruitment.BlackListed  OFF

SET IDENTITY_INSERT  Recruitment.OnCallList ON
INSERT INTO Recruitment.OnCallList  (OncallCode,OncallTypeID,ApplicationID) VALUES
(1,1,3),
(2,2,7),
(3,3,8)
SET IDENTITY_INSERT Recruitment.OnCallList  OFF

SET IDENTITY_INSERT  Recruitment.Onboarding ON
INSERT INTO Recruitment.Onboarding
(OnboardingID,ApplicationID,EducationCheck,Legitimacy) VALUES
(1,1,'PASS','PASS'),
(2,2,'PASS','PASS'),
(3,9,'PASS','PASS')
SET IDENTITY_INSERT Recruitment.Onboarding  OFF
```

```
SET IDENTITY_INSERT  Recruitment.RejectList ON
INSERT INTO Recruitment.RejectList  (RejectCode,ApplicationID,RejectTypeID) VALUES
(1,4,2),
(2,5,1),
(3,6,2)
SET IDENTITY_INSERT Recruitment.RejectList  OFF

SET IDENTITY_INSERT  Recruitment.Employee ON
INSERT INTO Recruitment.Employee  (EmployeeID,ApplicationID) VALUES
(1,1),
(2,2),
(3,9)
SET IDENTITY_INSERT Recruitment.Employee  OFF


SET IDENTITY_INSERT  Recruitment.ComplaintHandling ON
INSERT  INTO  Recruitment.ComplaintHandling  (ComplaintID,ApplicationID,ComplainDesc)
VALUES
(1,6,'This test is so hard that I think the company is making things difficult.')
SET IDENTITY_INSERT Recruitment.ComplaintHandling  OFF

SET IDENTITY_INSERT  Recruitment.Evaluation ON
INSERT  INTO  Recruitment.Evaluation  (EvaluationID,InterviewID,ApplicationID,GRADE)
VALUES
(1,1,1,100),
(2,2,1,100),
(3,3,1,100),
(4,4,1,100),
(5,5,1,100),
(6,6,1,100),
(7,1,2,70),
(8,2,2,70),
(9,3,2,80),
(10,4,2,70),
(11,5,2,70),
(12,6,2,80),
(13,7,2,100),
(14,8,2,100),
(15,9,2,100),
(16,10,2,100),
(17,11,2,100),
(18,12,2,100),
(19,1,3,100),
(20,2,3,100),
(21,3,3,100),
(22,4,3,100),
(23,5,3,100),
(24,6,3,100),
(25,1,4,30),
(26,2,4,25),
(27,3,4,40),
(28,4,4,0),
(29,5,4,0),
(30,6,4,0),
(31,1,6,45),
(32,2,6,55),
```

(33,3,6,60),
(34,4,6,45),
(35,5,6,30),
(36,6,6,27),
(37,1,7,100),
(38,2,7,98),
(39,3,7,100),
(40,4,7,100),
(41,5,7,98),
(42,6,7,100),
(43,1,8,80),
(44,2,8,98),
(45,3,8,90),
(46,4,8,100),
(47,5,8,100),
(48,6,8,100),
(49,1,9,100),
(50,2,9,98),
(51,3,9,100),
(52,4,9,100),
(53,5,9,100),
(54,6,9,100),
(55,1,10,80),
(56,2,10,70),
(57,3,10,80),
(58,4,10,80),
(59,5,10,70),
(60,6,10,80),
(61,7,10,100),
(62,8,10,70),
(63,9,10,80),
(64,10,10,100),
(65,11,10,70),
(66,12,10,80)
SET IDENTITY_INSERT Recruitment.Evaluation OFF

SET IDENTITY_INSERT Recruitment.Review ON
INSERT INTO Recruitment.Review (ReviewID,EvaluationID,CandReview,InterReview) VALUES
(1,55,'I could not believe that the examiner said my breasts stood beautiful and that if I could have a one-night stand with him, he could pass me straight away.','Strong applicant!'),
(2,61,'There is no one in your company to deal with this matter, and there has no handling mechanism. Just wait for my fireback!!!','Very Strong applicant!')
SET IDENTITY_INSERT Recruitment.Review OFF

Views

```
--VIEWS
---VIEW1
USE HR;
GO
--If EXIST DROP
IF OBJECT_ID('FinalGradeInterview1') IS NOT NULL
DROP VIEW FinalGradeInterview1;
GO
```

```
CREATE VIEW FinalGradeInterview1 AS
SELECT
Recruitment.Evaluation.ApplicationID,Recruitment.Tests.TestTittle,AVG(Recruitme
nt.Evaluation.GRADE)AS [FinalGradeInterview1]
--Use average grade of three grader as the final grade
FROM  Recruitment.Evaluation JOIN Recruitment.Interviews ON
Recruitment.Evaluation.InterviewID=Recruitment.Interviews.InterviewID
             JOIN Recruitment.Tests ON
Recruitment.Interviews.TestID=Recruitment.Tests.TestID
WHERE Recruitment.Tests.TestTittle='Programming1' or
Recruitment.Tests.TestTittle='SystemDesign1'--2 TESTS in interview1
GROUP BY  Recruitment.Evaluation.ApplicationID, Recruitment.Tests.TestTittle
GO
--TEST
SELECT*
FROM FinalGradeInterview1;

--VIEW2
--If EXIST DROP
USE HR;
GO
IF OBJECT_ID('FinalGradeInterview2') IS NOT NULL
DROP VIEW FinalGradeInterview2;
GO

CREATE VIEW FinalGradeInterview2 AS
SELECT
Recruitment.Evaluation.ApplicationID,Recruitment.Tests.TestTittle,AVG(Recruitme
nt.Evaluation.GRADE)AS [FinalGradeInterview2]
--Use average grade of three grader as the final grade
FROM  Recruitment.Evaluation JOIN Recruitment.Interviews ON
Recruitment.Evaluation.InterviewID=Recruitment.Interviews.InterviewID
             JOIN Recruitment.Tests ON
Recruitment.Interviews.TestID=Recruitment.Tests.TestID
WHERE Recruitment.Tests.TestTittle='Programming2' or
Recruitment.Tests.TestTittle='SystemDesign2'--2 TESTS in interview1
GROUP BY  Recruitment.Evaluation.ApplicationID, Recruitment.Tests.TestTittle
GO
--TEST
SELECT*
FROM FinalGradeInterview2;

--VIEW3
USE HR;
GO
--If EXIST DROP
IF OBJECT_ID('JobTestScore') IS NOT NULL
DROP VIEW JobTestScore ;
GO
-- query logic:
-- read my E/R diag, from the diag, link the job-openingjobs to the evaluation
from table to table linked by pk-fk
--
Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate,TestTittle
are can be grouped together present the relationship we want to explore.
-- use avg present the final grade for each test.
CREATE VIEW JobTestScore AS
SELECT Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate
AS[JobStartDate],TestTittle, AVG(GRADE) AS [TestResult]
FROM  Recruitment.Application JOIN Recruitment.OpeningJobs ON
Recruitment.Application.OpeningJobsID=Recruitment.OpeningJobs.OpeningJobsID
```

```
      JOIN Recruitment.Jobs ON  Recruitment.Jobs.JobCode=
Recruitment.OpeningJobs.JobCode
         JOIN  Recruitment.Evaluation ON
Recruitment.Evaluation.ApplicationID=Recruitment.Application.ApplicationID
         JOIN Recruitment.Interviews ON
Recruitment.Evaluation.InterviewID=Recruitment.Interviews.InterviewID
         JOIN Recruitment.Tests ON
Recruitment.Interviews.TestID=Recruitment.Tests.TestID
GROUP BY
Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate,TestTittle
GO
--TEST3
SELECT *
FROM JobTestScore

--VIEW4
USE HR;
GO

IF OBJECT_ID('EligbleReimbursement') IS NOT NULL
DROP VIEW EligbleReimbursement;
GO
-- query logic:
--1. location of candidate cannot near to the interview
--2  the interview must be onesite interview
-- read my E/R diag, from the diag, link the candidateInfo to the interview
Type from table to table linked by pk-fk

CREATE VIEW EligbleReimbursement AS
SELECT distinct Recruitment.Application.ApplicationID
FROM
Recruitment.Application JOIN Recruitment.Candidates ON
Recruitment.Candidates.CandidateID=Recruitment.Application.CandidateID
JOIN Recruitment.CandidatesInfo ON
Recruitment.Candidates.CandidateID=Recruitment.CandidatesInfo.CandidateID
JOIN Recruitment.Evaluation ON
Recruitment.Evaluation.ApplicationID=Recruitment.Application.ApplicationID
JOIN Recruitment.Interviews ON
Recruitment.Evaluation.InterviewID=Recruitment.Interviews.InterviewID
JOIN Recruitment.InterviewTypes ON
Recruitment.Interviews.InterviewTypeID=Recruitment.InterviewTypes.InterviewType
ID
WHERE
Recruitment.CandidatesInfo.LocationInfo<>Recruitment.Interviews.InterviewLocati
on AND InterviewType='Onsite'
GO
--TEST
SELECT*
FROM EligbleReimbursement;
```

UDF

```
--User Defined Functions
--UDF1
USE HR
GO

IF OBJECT_ID('fnInterviewsEvaluation') IS NOT NULL--if eixit drop
DROP FUNCTION fnInterviewsEvaluation
```

```
GO


CREATE FUNCTION fnInterviewsEvaluation
(@HighBar INT,@LowBar INT) -- two inputs , if below the lowbar 'Reject'
-- higher than the highbar 'select'
-- otherwise next round of interview
RETURNS TABLE --return a table

RETURN
(
	SELECT ApplicationID, Decision=
	CASE
	WHEN AVG(FinalGradeInterview1)>@HighBar THEN 'Select' -- aggerate
functions to evluate the solution
	WHEN AVG(FinalGradeInterview1)<@LowBar THEN 'Reject'
	ELSE 'NextRound'
	END
	From dbo.FinalGradeInterview1
	GROUP BY ApplicationID -- group by each applicant
)

GO
--TEST
SELECT*
FROM fnInterviewsEvaluation(80,60)

Query:USE HR
GO

--UDF2
IF OBJECT_ID('fnFinalInterviewsEvaluation') IS NOT NULL--if eixit drop
DROP FUNCTION fnFinalInterviewsEvaluation
GO


CREATE FUNCTION fnFinalInterviewsEvaluation
(@Bar INT) -- one inputs , if below the bar 'Reject'  higher than the bar
'select'
RETURNS TABLE --return a table

RETURN
(
	SELECT ApplicationID, Decision=
	CASE
	WHEN AVG(FinalGradeInterview2)>@Bar THEN 'Select' -- aggerate functions
to evluate the solution
	ELSE 'Reject'
	END
	From dbo.FinalGradeInterview2
	GROUP BY ApplicationID -- group by each applicant
)

GO
--TEST
SELECT*
FROM fnFinalInterviewsEvaluation(80)
--UDF3
USE HR
GO

IF OBJECT_ID('JobTest') IS NOT NULL--if eixit drop
```

```
DROP FUNCTION JobTest
GO


CREATE FUNCTION JobTest
(@jobcode int)
RETURNS TABLE

RETURN
(
 --query logic:
-- read my E/R diag, from the diag, link the job-openingjobs to the evaluation
from table to table linked by pk-fk
--
Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate,TestTittle
are can be grouped together present the relationship we want to explore.
-- use avg present the final grade for each test.
       SELECT
Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate
AS[JobStartDate],TestTittle, AVG(GRADE) AS [TestResult]
       FROM  Recruitment.Application JOIN Recruitment.OpeningJobs ON
Recruitment.Application.OpeningJobsID=Recruitment.OpeningJobs.OpeningJobsID
      JOIN Recruitment.Jobs ON  Recruitment.Jobs.JobCode=
Recruitment.OpeningJobs.JobCode
         JOIN  Recruitment.Evaluation ON
Recruitment.Evaluation.ApplicationID=Recruitment.Application.ApplicationID
         JOIN Recruitment.Interviews ON
Recruitment.Evaluation.InterviewID=Recruitment.Interviews.InterviewID
         JOIN Recruitment.Tests ON
Recruitment.Interviews.TestID=Recruitment.Tests.TestID
       WHERE Recruitment.Jobs.JobCode=@jobcode --only select the input jobcode
       GROUP BY
Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate,TestTittle

)

GO
--TEST
SELECT*
FROM JobTest (1)

--UDF4
USE HR
GO

IF OBJECT_ID('fnBestApplicant') IS NOT NULL--if eixit drop
DROP FUNCTION fnBestApplicant
GO


CREATE FUNCTION fnBestApplicant
(@TESTNAME nvarchar(50))
RETURNS TABLE
-- query logic:
-- read my E/R diag, from the diag, link the job-openingjobs to the evaluation
from table to table linked by pk-fk
--
Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate,TestTittle
are can be grouped together present the relationship we want to explore.
-- use avg present the final grade for each test.
RETURN
(
```

```
SELECT TOP 1 Application.ApplicationID,TestTittle, AVG(GRADE) AS [TestResult]--
find the best
FROM  Recruitment.Application JOIN Recruitment.OpeningJobs ON
Recruitment.Application.OpeningJobsID=Recruitment.OpeningJobs.OpeningJobsID
      JOIN Recruitment.Jobs ON  Recruitment.Jobs.JobCode=
Recruitment.OpeningJobs.JobCode
         JOIN  Recruitment.Evaluation ON
Recruitment.Evaluation.ApplicationID=Recruitment.Application.ApplicationID
         JOIN Recruitment.Interviews ON
Recruitment.Evaluation.InterviewID=Recruitment.Interviews.InterviewID
         JOIN Recruitment.Tests ON
Recruitment.Interviews.TestID=Recruitment.Tests.TestID
WHERE TestTittle=@TESTNAME --name mactch
GROUP BY
Application.ApplicationID,JOBS.JobCode,JobName,OpeningJobs.StartDate,TestTittle
ORDER BY TestResult DESC --order score from high to low
)
GO
--TEST
select*
from fnBestApplicant('Programming1')
```

Stored procedures

```
--Stored procedures
--SP1
USE HR;
GO
IF OBJECT_ID('spReimbursementInformation') IS NOT NULL
DROP PROC spReimbursementInformation
GO

CREATE PROC spReimbursementInformation
AS
-- JOIN ALL TABLES CAN PROVIDE Reimbursement information (E/R)
-- ONLY SELECT THOSE APPLICANT WHO CAN Reimbursement their cost by the VIEW
EligbleReimbursement
select*
from Recruitment.Reimbursement JOIN Recruitment.HotelReservations ON
Recruitment.Reimbursement.ReimbursementID=Recruitment.HotelReservations.Reimbur
sementID
JOIN Recruitment.CarRentals ON
Recruitment.Reimbursement.ReimbursementID=Recruitment.CarRentals.ReimbursementI
D
JOIN Recruitment.AirlineReservations ON
Recruitment.AirlineReservations.ReimbursementID=Recruitment.Reimbursement.Reimb
ursementID
WHERE Recruitment.Reimbursement.ReimbursementID IN
(
       select Recruitment.Application.ReimbursementID
       from Recruitment.Application JOIN EligbleReimbursement ON
Recruitment.Application.ApplicationID=EligbleReimbursement.ApplicationID
)
GO
--TEST1
EXEC spReimbursementInformation

--SP2
USE HR;
```

```
GO
IF OBJECT_ID('spReimbursementTest') IS NOT NULL
DROP PROC spReimbursementTest
GO

CREATE PROC spReimbursementTest
	@Range int =0
AS
	IF @Range <0 --- avoid possible logical mistake
		THROW 50001,'Range must be postive interger',1;
-- When the spReimbursement is eligibe and the difference between
-- real spend (get from database)and asked Reimbursement less then the rage
-- Accept else Reject
	Select Recruitment.Reimbursement.ReimbursementID,Decision=
CASE
WHEN SpendTotal-TotalSpend<=@Range THEN 'Accept'
ELSE 'Reject'
END
from Recruitment.Reimbursement JOIN
(
	select ReimbursementID,SUM(Cost) as TotalSpend
	from
	(
		-- USE UNION function aggregate all information relate to
Reimbursement
		SELECT Recruitment.AirlineReservations.ReimbursementID,
		Recruitment.AirlineReservations.AirlineName AS[NAME],
Recruitment.AirlineReservations.AirlineTrackNumber AS[TrackNumber],
		Recruitment.AirlineReservations.AirLineCost[Cost],
Recruitment.AirlineReservations.AirlineReservationID AS[ServiceID]
		FROM Recruitment.AirlineReservations
		UNION ALL
		SELECT Recruitment.CarRentals.ReimbursementID,
		Recruitment.CarRentals.[CarRentalName] AS[NAME],
Recruitment.CarRentals.[CarRentalTrackNumber] AS[TrackNumber],
		Recruitment.CarRentals.[CarRentCost][Cost],
Recruitment.CarRentals.CarRentalID AS[ServiceID]
		FROM Recruitment.CarRentals
		UNION ALL
		SELECT  Recruitment.HotelReservations.ReimbursementID,
		Recruitment.HotelReservations.[HotelName] AS[NAME],
Recruitment.HotelReservations.[HotelTrackNumber] AS[TrackNumber],
		Recruitment.HotelReservations.[HotelCost][Cost],
Recruitment.HotelReservations.[HotelReservationID] AS[ServiceID]
		FROM Recruitment.HotelReservations
		) AS TEMP--save the result in the TEMP table
		WHERE TEMP.ReimbursementID IN--use the EligbleReimbursement view
check the Reimbursement is Eligble or not
		(
		select Recruitment.Application.ReimbursementID
		from Recruitment.Application JOIN EligbleReimbursement ON
Recruitment.Application.ApplicationID=EligbleReimbursement.ApplicationID
		)
	group by ReimbursementID
) AS TEMP2 ON TEMP2.ReimbursementID=Recruitment.Reimbursement.ReimbursementID

GO
----test2

EXEC spReimbursementTest @Range =200
GO
```

```
--SP3
USE HR;
GO
IF OBJECT_ID('spOnCallOreder') IS NOT NULL
DROP PROC spOnCallOreder
GO

CREATE PROC spOnCallOreder
AS
SELECT Recruitment.OnCallList.ApplicationID,AVG(TestResult) AS [Performance] --
USE Test performance as evidence of the order
FROM Recruitment.OnCallList JOIN HR.dbo.JobTestScore --USE view JobTestScore
get peformance of onlist candidate
ON Recruitment.OnCallList.ApplicationID=HR.dbo.JobTestScore.ApplicationID
group by Recruitment.OnCallList.ApplicationID
GO

--test3
EXEC spOnCallOreder

--SP4
USE HR;
GO
IF OBJECT_ID('spOnCallOrederJob') IS NOT NULL
DROP PROC spOnCallOrederJob
GO

CREATE PROC spOnCallOrederJob
 @JobID int =0
AS
IF(@JobID=0)
       THROW 50002,'invalid jobID',1;
SELECT
Recruitment.OnCallList.ApplicationID,HR.dbo.JobTestScore.JobCode,AVG(TestResult)
AS[Performance]
FROM Recruitment.OnCallList JOIN HR.dbo.JobTestScore --USE view JobTestScore
get peformance of onlist candidate
ON Recruitment.OnCallList.ApplicationID=HR.dbo.JobTestScore.ApplicationID
WHERE HR.dbo.JobTestScore.JobCode=@JobID --use the input id
group by Recruitment.OnCallList.ApplicationID,HR.dbo.JobTestScore.JobCode
GO

--test4
EXEC spOnCallOrederJob @JobID=2
```

Triggers & Transactions

```
--------TT1
USE HR;
GO

DROP TRIGGER IF EXISTS Recruitment.BlackListInsert;
GO

CREATE TRIGGER BlackListInsert
       ON Recruitment.BlackListed
       AFTER INSERT--when insert into Recruitment.BlackListed trigger run
AS
```

```
	DECLARE @ApID int; --save application ID
	SELECT @ApID=inserted.ApplicationID FROM inserted
	DECLARE @OpJobID int; ---Save OpenjobID need to update
	SELECT  @OpJobID=HR.Recruitment.Application.OpeningJobsID FROM
HR.Recruitment.Application
BEGIN TRY
	BEGIN TRAN --Tran make database consistent
	UPDATE HR.Recruitment.Application
	SET HR.Recruitment.Application.StatueID=10 --change the application
statues to 10,which means blacklisted
	WHERE HR.Recruitment.Application.ApplicationID=@ApID

	UPDATE HR.Recruitment.OpeningJobs
	SET
HR.Recruitment.OpeningJobs.NumPosition=HR.Recruitment.OpeningJobs.NumPosition+1
--change number of openjobs+1
	WHERE HR.Recruitment.OpeningJobs.OpeningJobsID=@OpJobID

	COMMIT TRAN
END TRY
BEGIN CATCH--if any one of the tran cannot work, rollback and throw the error
message
	ROLLBACK TRAN
END CATCH
GO
-----------TT2
USE HR;
GO

DROP TRIGGER IF EXISTS Recruitment.Declined;
GO

CREATE TRIGGER Declined
	ON Recruitment.OnCallList
	AFTER INSERT--when insert into Recruitment.BlackListed trigger run
AS
	DECLARE @ApID int; --save application ID
	SELECT @ApID=inserted.ApplicationID FROM inserted
	DECLARE @OpJobID int; ---Save OpenjobID need to update
	SELECT  @OpJobID=HR.Recruitment.Application.OpeningJobsID FROM
HR.Recruitment.Application
	DECLARE @DeclinedID int
	SET @DeclinedID=3

    IF @DeclinedID= (select inserted.OncallTypeID From inserted)
	BEGIN;
		BEGIN TRY
			BEGIN TRAN --Tran make database consistent
				UPDATE HR.Recruitment.Application
				SET HR.Recruitment.Application.StatueID=6 --change
the application statues to 6,which means Oncall
				WHERE
HR.Recruitment.Application.ApplicationID=@ApID

				UPDATE HR.Recruitment.OpeningJobs
				SET
HR.Recruitment.OpeningJobs.NumPosition=HR.Recruitment.OpeningJobs.NumPosition+1
--change number of openjobs+1
				WHERE
HR.Recruitment.OpeningJobs.OpeningJobsID=@OpJobID

			COMMIT TRAN
```

```
              END TRY
              BEGIN CATCH--if any one of the tran cannot work, rollback and
throw the error message
                     ROLLBACK TRAN
              END CATCH
       END;

GO
------TT3
USE HR;
GO

DROP TRIGGER IF EXISTS Recruitment.Oncalled;
GO

CREATE TRIGGER Oncalled
       ON Recruitment.OnCallList
       AFTER INSERT--when insert into Recruitment.Oncalled.trigger run
AS
       DECLARE @ApID int; --save application ID
       SELECT @ApID=inserted.ApplicationID FROM inserted
       DECLARE @OpJobID int; ---Save OpenjobID need to update
       SELECT  @OpJobID=HR.Recruitment.Application.OpeningJobsID FROM
HR.Recruitment.Application
       DECLARE @DeclinedID int
       SET @DeclinedID=3

    IF @DeclinedID<> (select inserted.OncallTypeID From inserted)
       BEGIN;
              BEGIN TRY
                     BEGIN TRAN --Tran make database consistent
                            UPDATE HR.Recruitment.Application
                            SET HR.Recruitment.Application.StatueID=6 --change
the application statues to 6,which means Oncall
                            WHERE
HR.Recruitment.Application.ApplicationID=@ApID

                     COMMIT TRAN
              END TRY
              BEGIN CATCH--if any one of the tran cannot work, rollback and
throw the error message
                     ROLLBACK TRAN
              END CATCH
       END;

GO
-----TT4
USE HR;
GO

DROP TRIGGER IF EXISTS Recruitment.Oncalled;
GO

CREATE TRIGGER Oncalled
       ON Recruitment.OnCallList
       AFTER INSERT--when insert into Recruitment.Oncalled.trigger run
AS
       DECLARE @ApID int; --save application ID
       SELECT @ApID=inserted.ApplicationID FROM inserted
       DECLARE @OpJobID int; ---Save OpenjobID need to update
       SELECT  @OpJobID=HR.Recruitment.Application.OpeningJobsID FROM
HR.Recruitment.Application
```

```
       DECLARE @DeclinedID int
       SET @DeclinedID=3

    IF @DeclinedID<> (select inserted.OncallTypeID From inserted)
       BEGIN;
              BEGIN TRY
                     BEGIN TRAN --Tran make database consistent
                            UPDATE HR.Recruitment.Application
                            SET HR.Recruitment.Application.StatueID=6 --change
the application statues to 6,which means Oncall
                            WHERE
HR.Recruitment.Application.ApplicationID=@ApID

                     COMMIT TRAN
              END TRY
              BEGIN CATCH--if any one of the tran cannot work, rollback and
throw the error message
                     ROLLBACK TRAN
              END CATCH
       END;

GO
------------------TT5
USE HR;
GO

DROP TRIGGER IF EXISTS Recruitment.OnboardingFail;
GO

CREATE TRIGGER OnboardingFail
       ON Recruitment.OnCallList
       AFTER INSERT--when insert into Recruitment.BlackListed trigger run
AS
       DECLARE @ApID int; --save application ID
       SELECT @ApID=inserted.ApplicationID FROM inserted
       DECLARE @OpJobID int; ---Save OpenjobID need to update
       SELECT  @OpJobID=HR.Recruitment.Application.OpeningJobsID FROM
HR.Recruitment.Application
       DECLARE @DeclinedID int
       SET @DeclinedID=2---type=2

    IF @DeclinedID= (select inserted.OncallTypeID From inserted)
       BEGIN;
              BEGIN TRY
                     BEGIN TRAN --Tran make database consistent
                            UPDATE HR.Recruitment.Application
                            SET HR.Recruitment.Application.StatueID=6 --change
the application statues to 6,which means Oncall
                            WHERE
HR.Recruitment.Application.ApplicationID=@ApID

                            UPDATE HR.Recruitment.OpeningJobs
                            SET
HR.Recruitment.OpeningJobs.NumPosition=HR.Recruitment.OpeningJobs.NumPosition+1
--change number of openjobs+1
                            WHERE
HR.Recruitment.OpeningJobs.OpeningJobsID=@OpJobID

                     COMMIT TRAN
              END TRY
              BEGIN CATCH--if any one of the tran cannot work, rollback and
throw the error message
```

```
                ROLLBACK TRAN
            END CATCH
        END;

GO
-------TT6
USE HR;
GO

DROP TRIGGER IF EXISTS Recruitment.RejectAdd;
GO

CREATE TRIGGER RejectAdd
        ON Recruitment.RejectList
        AFTER INSERT--when insert into Recruitment.Oncalled.trigger run
AS
        DECLARE @ApID int; --save application ID
        SELECT @ApID=inserted.ApplicationID FROM inserted
        BEGIN;
            BEGIN TRY
                BEGIN TRAN --Tran make database consistent
                    UPDATE HR.Recruitment.Application
                    SET HR.Recruitment.Application.StatueID=2 --change
the application statues to 2,which means reject
                    WHERE
HR.Recruitment.Application.ApplicationID=@ApID

                COMMIT TRAN
            END TRY
            BEGIN CATCH--if any one of the tran cannot work, rollback and
throw the error message
                ROLLBACK TRAN
            END CATCH
        END;

GO
-----TT7
USE HR;
GO

DROP TRIGGER IF EXISTS Recruitment.EmployeeAdd;
GO

CREATE TRIGGER EmployeeAdd
        ON Recruitment.Employee
        AFTER INSERT--when insert into Recruitment.Oncalled.trigger run
AS
        DECLARE @ApID int; --save application ID
        SELECT @ApID=inserted.ApplicationID FROM inserted
        BEGIN;
            BEGIN TRY
                BEGIN TRAN --Tran make database consistent
                    UPDATE HR.Recruitment.Application
                    SET HR.Recruitment.Application.StatueID=11 --change
the application statues to 11,which means employee
                    WHERE
HR.Recruitment.Application.ApplicationID=@ApID

                COMMIT TRAN
            END TRY
            BEGIN CATCH--if any one of the tran cannot work, rollback and
throw the error message
```

```
                    ROLLBACK TRAN
                END CATCH
        END;

GO
---------TT8
USE HR;
GO

DROP TRIGGER IF EXISTS Recruitment.ComplainAdd;
GO

CREATE TRIGGER ComplainAdd
        ON Recruitment.ComplaintHandling
        AFTER INSERT--when insert into Recruitment.Oncalled.trigger run
AS
        DECLARE @ApID int; --save application ID
        SELECT @ApID=inserted.ApplicationID FROM inserted
        BEGIN;
                BEGIN TRY
                        BEGIN TRAN --Tran make database consistent
                                UPDATE HR.Recruitment.Application
                                SET HR.Recruitment.Application.StatueID=4 --change
the application statues to 4,which means waiting
                                WHERE
HR.Recruitment.Application.ApplicationID=@ApID

                        COMMIT TRAN
                END TRY
                BEGIN CATCH--if any one of the tran cannot work, rollback and
throw the error message
                        ROLLBACK TRAN
                END CATCH
        END;

GO
```

ROLES

```
CREATE ROLE CandidateManager;
GRANT UPDATE, INSERT
ON Recruitment.Candidates TO CandidateManager;
GRANT UPDATE, INSERT
ON Recruitment.CandidatesInfo TO CandidateManager;
GRANT UPDATE, INSERT
ON Recruitment.CandidatesDocuments TO CandidateManager;
ALTER ROLE db_datareader ADD MEMBER CandidateManager

--Role for JobManager
CREATE ROLE JobManager;
GRANT UPDATE, INSERT
ON  Recruitment.JobTypes TO JobManager;
GRANT UPDATE, INSERT
ON  Recruitment.JobMediums TO JobManager;
GRANT UPDATE, INSERT
ON  Recruitment.JobCategories TO JobManager;
GRANT UPDATE, INSERT
ON  Recruitment.JobPositions TO JobManager;
GRANT UPDATE, INSERT
```

```
ON  Recruitment.JobOpeningPlatforms TO JobManager;
GRANT UPDATE, INSERT
ON  Recruitment.Jobs TO JobManager;
GRANT UPDATE, INSERT
ON  Recruitment.OpeningJobs TO JobManager;
ALTER ROLE db_datareader ADD MEMBER JobManager
--Role for InterviewManager

CREATE ROLE InterviewManager;
GRANT UPDATE, INSERT
ON  Recruitment.Interviewers TO InterviewManager;
GRANT UPDATE, INSERT
ON  Recruitment.TestTypes TO InterviewManager;
GRANT UPDATE, INSERT
ON  Recruitment.Tests  TO InterviewManager;
GRANT UPDATE, INSERT
ON  Recruitment.InterviewTypes TO InterviewManager;
GRANT UPDATE, INSERT
ON  Recruitment.Interviews TO InterviewManager;
ALTER ROLE db_datareader ADD MEMBER InterviewManager

--Role ReimbursementManager
CREATE ROLE ReimbursementManager;
GRANT UPDATE, INSERT
ON  Recruitment.Reimbursement TO ReimbursementManager;
GRANT UPDATE, INSERT
ON  Recruitment.AirlineReservations TO ReimbursementManager;
GRANT UPDATE, INSERT
ON  Recruitment.HotelReservations TO ReimbursementManager;
GRANT UPDATE, INSERT
ON  Recruitment.CarRentals TO ReimbursementManager;
ALTER ROLE db_datareader ADD MEMBER ReimbursementManager

--Role ApplicationManager
CREATE ROLE  ApplicationManager;
GRANT UPDATE, INSERT
ON  Recruitment.OnCallType TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.RejectType TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.Statues TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.Application TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.OnCallList TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.Onboarding TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.RejectList TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.BlackListed TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.Employee TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.ComplaintHandling TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.Evaluation TO ApplicationManager ;
GRANT UPDATE, INSERT
ON  Recruitment.Review TO ApplicationManager ;
ALTER ROLE db_datareader ADD MEMBER  ApplicationManager
```

LOGIN AND USER

```
CREATE LOGIN Lord WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO

CREATE USER WANGXIAO FOR LOGIN [Lord];
ALTER ROLE CandidateManager ADD MEMBER WANGXIAO
ALTER ROLE InterviewManager ADD MEMBER WANGXIAO
ALTER ROLE ReimbursementManager ADD MEMBER WANGXIAO
ALTER ROLE ApplicationManager ADD MEMBER WANGXIAO
ALTER ROLE JobManager ADD MEMBER WANGXIAO

------------------
CREATE LOGIN CandidateGroup WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO

CREATE USER DongLu FOR LOGIN [CandidateGroup];
ALTER ROLE CandidateManager ADD MEMBER DongLu
------------------------
CREATE LOGIN JobGroup WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO

CREATE USER WangPiPi FOR LOGIN [JobGroup];
ALTER ROLE JobManager ADD MEMBER WangPiPi
-------------------------
CREATE LOGIN InterviewGroup WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO

CREATE USER Shawn FOR LOGIN [InterviewGroup];
ALTER ROLE InterviewManager ADD MEMBER Shawn
-----------------------------
CREATE LOGIN ReimbursementGroup WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO

CREATE USER Robin FOR LOGIN [ReimbursementGroup];
ALTER ROLE ReimbursementManager ADD MEMBER Robin
--------------------------------
CREATE LOGIN ApplicationGroup WITH PASSWORD = '12345678',
DEFAULT_DATABASE = HR;
GO

CREATE USER Jack FOR LOGIN [ApplicationGroup];
ALTER ROLE ApplicationManager ADD MEMBER Jack
----------------------------------
```

Business

```
USE HR
GO

Select Education,COUNT(*) AS NUM--count
---JOIN APPLICATION-candidate-candidate information
From Recruitment.Application JOIN Recruitment.Candidates ON
Recruitment.Application.CandidateID=Recruitment.Candidates.CandidateID
JOIN Recruitment.CandidatesInfo ON
Recruitment.Candidates.CandidateID=Recruitment.CandidatesInfo.CandidateID
GROUP BY Education --use aggregate


Select Education,COUNT(*) AS NUM
---Join employee--APPLICATION-candidate-candidate information
From Recruitment.Application JOIN Recruitment.Employee ON
Recruitment.Application.ApplicationID=  Recruitment.Employee.ApplicationID
JOIN Recruitment.Candidates ON
Recruitment.Application.CandidateID=Recruitment.Candidates.CandidateID
JOIN Recruitment.CandidatesInfo ON
Recruitment.Candidates.CandidateID=Recruitment.CandidatesInfo.CandidateID
GROUP BY Education
--------------
Select Education,AVG(GRADE) AS GRADE
---JOIN APPLICATION-candidate-candidate information-evaluation
From Recruitment.Application JOIN Recruitment.Candidates ON
Recruitment.Application.CandidateID=Recruitment.Candidates.CandidateID
JOIN Recruitment.CandidatesInfo ON
Recruitment.Candidates.CandidateID=Recruitment.CandidatesInfo.CandidateID
JOIN Recruitment.Evaluation ON
Recruitment.Application.ApplicationID=Recruitment.Evaluation.ApplicationID
Group by Education
ORDER BY GRADE DESC
-------------------
select PlatformName, COUNT(*) AS NUMS
--JOIN-Application-- OPENJOBS---OPENJOBSPLATEFORM
from Recruitment.Application
join Recruitment.OpeningJobs on
Recruitment.Application.OpeningJobsID=Recruitment.OpeningJobs.OpeningJobsID
join Recruitment.JobOpeningPlatforms on
Recruitment.OpeningJobs.JobJobOpeningPlatformID=Recruitment.JobOpeningPlatforms.
JobOpeningPlatformID
group by PlatformName
-------------------------
SELECT*
FROM Recruitment.ComplaintHandling

SELECT AVG(GRADE) AS GRADE
FROM Recruitment.Application
JOIN Recruitment.Evaluation ON
Recruitment.Application.ApplicationID=Recruitment.Evaluation.ApplicationID
WHERE Recruitment.Application.ApplicationID<>6

SELECT Recruitment.Application.ApplicationID,AVG(GRADE) AS GRADE
FROM Recruitment.Application
JOIN Recruitment.Evaluation ON
Recruitment.Application.ApplicationID=Recruitment.Evaluation.ApplicationID
WHERE Recruitment.Application.ApplicationID=6
GROUP BY Recruitment.Application.ApplicationID
```