2022

CSE 581 Fall 2022
Project 2

HUMAN RESOURCES DATABASE

JACK WILLIS

SYRACUSE UNIVERSITY | College of Engineering & Computer Science | Dr. Ercanli

Contents

Abstract

This project is the final project in Syracuse University's Introduction to Database Management Systems course CSE 581. Generally, it encompasses all of the topics covered in the class including: DML statements, DDL statements, database implementation, referential integrity, table relationships, database design, user defined functions, triggers, scripts, stored procedures, aggregate functions, transactions, server & database security, and generating business reports. The intent of the project was to design and implement a database that could be used by the human resources department of a company, more specifically the recruitment aspects of human resources. It allows candidates to apply to listed jobs with the potential of landing open positions after being interviewed by an employee and passing assigned tests. The database also tracks hiring information on each candidate and necessary accommodations for onsite interviews. This report is divided into three sections, design, implementation, and testing. A disclaimer to the reader and grader of my project, I was unable to complete the second task in the testing section. I did run basic test cases on inserted data to ensure everything worked but they are not documented.

Section A: Design

Introduction

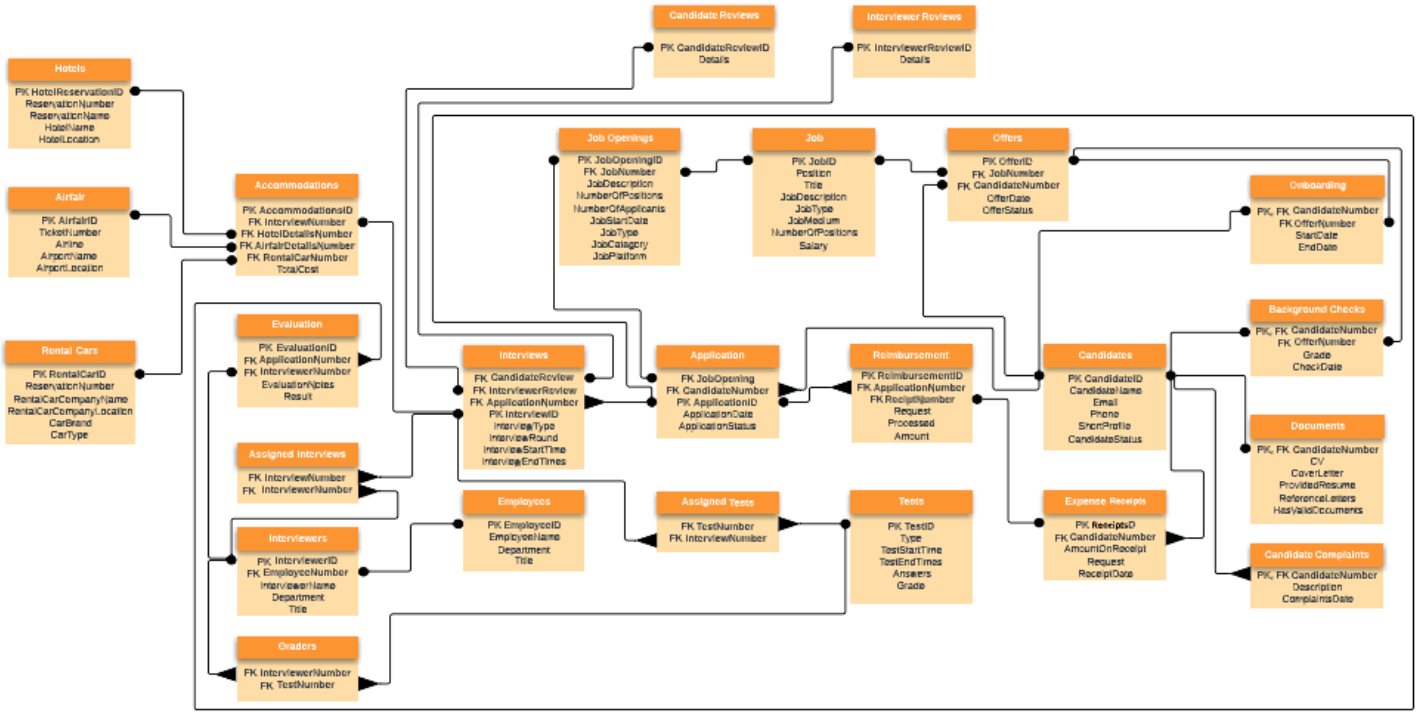
My design includes twenty-five different tables supporting the desired workflow of accepting applications, interviewing candidates, and extending offers if deemed hirable. Please see the database objects section for in depth descriptions of each table. With normalization in mind, my design includes various foreign key primary key relationships and three linking tables. I approached modeling my design with three main sections in mind, the job section, the interview section, and the candidate sections. In the diagram, the job section are the tables located towards the top of the figure, the interview section is towards the left and the candidate section is to the right. I wanted the diagram to read like the process it's attempting to represent. We start as the candidate, to the right, apply for a job, the middle top, and end with interviews to left. Obviously, details like job offers and accommodations don't strictly adhere to this pattern, but for the most part this is the main organizational method I used when creating my design and E/R diagram.

Design Considerations

To begin designing, I implemented the eleven provided tables and took inventory on what requirements they satisfied. I then looked for requirements not covered by these tables and began adding new ones, considering data needs that had yet to be fulfilled. After I felt I had enough tables to store the necessary information for future queries, I began the process of relating the tables to each other, this phase took the longest. To accomplish normalization in the third normal form, I added three linking tables and ensured each data table depended entirely on its own primary key and only held data related to a tables purpose. While designing I kept in mind procedures necessary to interact with the data, my main focus was allowing candidates information to be accessed with few primary keys and placing a small number of relations between the candidate, their interview and the job they were applying to. I then worried about extra detail that was desired, but not necessarily must haves when it came to the core functionality of my design. In my diagram this can be seen with most detail-

oriented tables be located on the outer edges of my diagram. Please see the diagram section below to see a visual representation of my design.

E/R Diagram



E/R Diagram for Human Resources Database

Database Objects

Hotels, Airfair & Rental Cars Tables

Description:

These three tables each hold the travel information necessary for a candidate to interview onsite. Each table has a primary key that has a one-to-one relationship with the Accommodations table which connects each of the three foreign keys to a given interview. An interview can only accommodate one candidate and thus, presumably, only needs one airfare entry, one hotel entry and a single rental car which is why I used a one-to-one relationship for each of these tables.

Keys:

Primary Keys: HotelReservationID, AirfairID & RentalCarID

Related Tables:

Accommodations

Candidate & Interviewer Review Tables

Description:

These two tables hold optional reviews that can be given by a candidate or interviewer after an interview is conducted. The tables are very simple, only including a primary key and a details attribute. They each have a one-to-one relationship with the Interviews table as a single interview can only have one candidate review and one interviewer review.

Keys:

Primary Keys: CandidateReviewID & InterviewerReviewID

Related Tables:

Interviews

Onboarding Table

Description:

This table holds information on onboarding for candidates that were extend an offer and only allows candidates who have a valid background check and documents to be inserted into it. This table has a primary key, which also serves as a foreign key, relating the Onboarding table to a candidate in the Candidates table with a one-to-one relationship since one onboarding entry corresponds with a single candidate. The table also has a foreign key that connects an onboarding entry to a single offer, this relation was described before. Since a candidate is onboarded for a particular role, a one-to-one relationship was used here.

Keys:

Primary Keys: CandidateNumber

Foreign Keys: CandidateNumber & OfferNumber

Related Tables:

Candidates, Offers

Background Checks Table

Description:

This table is very similar to the previous one, it has a primary key used as a foreign key with a one-to-one relationship with Candidates, since each candidate has a background check, and another foreign key with a one-to-one relationship to Offers since each offer requires a background check be passed to proceed to onboarding. This table includes an attribute Grade, which holds a passed or failed score for a given background check.

Keys:

Primary Keys: CandidateNumber

Foreign Keys: CandidateNumber & OfferNumber

Related Tables:

Candidates & Offers

Documents & Candidate Complaints Tables

Description:

These tables both have a primary key that's also a foreign key relating each documents entry and candidate complaint to a candidate. Each candidate can file multiple complaints so a one-to-many relationship exists between a candidate and their complaints. The documents table has a one-to-one relationship per candidate since a single candidate can only have one copy of documents. The candidate complaints table has a trigger that places a candidate in a waiting status if they file a complaint.

Keys:

Primary Keys: CandidateNumber

Foreign Keys: CandidateNumber

Related Tables:

Candidates

Triggers:

processComplaint

Accommodations Table

Description:

This table is used to record the total costs of accommodations per interview and relate specific accommodations to each interview. Because of this, there are many foreign keys each with a one-to-one relationship. Each accommodation has one of three in depth accommodations, hotel, airfair and rental car. An accommodation is unique to each onsite interview so there is a one-to-one relationship between the Accommodations and Interviews tables.

Keys:

Primary Keys: AccommodationsID

Foreign Keys: InterviewNumber, HotelDetailsNumber, AirfairDetailsNumber & RentalCarNumber

Related Tables:

Hotels, Airfair, Rental Cars & Interviews

Evaluation Table

Description:

This table is used to store interviewer evaluations for a given interview. Since an evaluation is completed by a single interviewer, there's a one-to-one relationship between each interviewer entry and a given evaluation. An application could potentially have many interviews so each application can have multiple evaluations, thus a one-to-many relationship is needed between Evaluations and Application.

Keys:
Primary Keys: EvaluationID

Foreign Keys: ApplicationsNumber & InterviewerNumber

Related Tables:
Interviewers & Application

Assigned Interviews & Graders Tables

Description:
These tables are linking tables supporting many-to-many relations between Interviews/Interviewers and Graders/Tests respectively. Interviews can have multiple interviewers and each interviewer can be assigned multiple interviews which means a mapping between each interviewer and interview is necessary. Likewise, since a test can be graded by multiple interviewers and each interviewer can be assigned many tests to grade a relationship between the interviewer and the many tests, they could potentially grade is needed.

Keys:
Foreign Keys: InterviewNumber, InterviewerNumber & TestNumber

Related Tables:
Interviews, Interviewers, Graders, & Tests

Interviewers Table

Description:
This table tracks information on each interviewer in the human resources department. It has a one-to-many relationship with the linking table Assigned Interviews which maps each interviewer to interviews they're assigned to. This table maps each interviewer to an employee number since not all employees are interviewers, this is achieved with a one-to-one relationship since an interviewer is a single employee. Finally, Interviewers has a one-to-many relationship with the Graders linking table since an Interviewer can be assigned to grade various tests. An interviewer can also give a single evaluation so there is a one-to-one relation between the Interviewers table and Evaluation table.

Keys:
Primary Keys: InterviewerID

Foreign Keys: EmployeeNumber

Related Tables:
Assigned Interviews, Graders, Employees & Evaluation

Job Openings Table

Description:

This table consists of job openings available at the company and contains all information related to one. It has a one-to-one relationship to the Job table since each job opening corresponds to a single job. It also has a one-to-one relationship with the Application table since each application pertains to a single job opening.

Keys:

Primary Keys: JobOpeningID

Foreign Keys: JobNumber

Related Tables:

Job & Application

Job Table

Description:

This table tracks ever job at the company and information relating to a specific job. Since a job could be open for employment a one-to-one relationship exists between the Job and Job Openings table. A single job relates to a single job opening. A job offer also pertains to a single job, so the Job table relates to the Offers table with a one-to-one relationship. The Job table has a primary key used to identify each unique job and no foreign keys.

Keys:

Primary Keys: JobID

Related Tables:

Job Openings & Offers

Offers Table

Description:

This table holds information for each offer extended to a specific candidate and the position being included in the offer. Since a offer relates to a single Job, a single candidate, has one onboarding procedure and a single background check there are four one-to-one relationships between Offers and the Job, Onboarding, Candidate and Background Check tables. These relationships are achieved via two foreign keys and each entry has its own primary key. This table has a trigger that checks if a added candidate has the proper documentation and passed a background check. It also has two triggers that update the number of job openings based on if an offer is accepted or declined.

Keys:

Primary Keys: OfferID

Foreign Keys: JobNumber & CandidateNumber

Related Tables:

Job, Onboarding, Candidates & Background Checks

Triggers:
verifyCandidate, updateNumberOfJobOpenings & updateNumberOfJobOpeningsOnDecline

Interviews Table

Description:
This table has information on each interview conducted by the department, it has three foreign keys and a single primary. Since an application can have many interviews there is a many-to-one relationship between the Interviews and Application table. Each Interview has a single candidate review, interviewer review and accommodations related to it if the interview is in person. Thus, there is a one-to-one relationship between Interviews, Candidate Reviews, Interviewer Reviews and Accommodations. An interview can be assigned multiple interviews and multiple Tests, so it has a one-to-many relation between itself, Assigned Interviews and Assigned Tests.

Keys:
Primary Keys: InterviewID

Foreign Keys: InterviewerReview, CandidateReview & Applications Number

Related Tables:
Applications, Assigned Interviews, Assigned Tests, Candidate Reviews, Accomodations & Interview Reviews

Application Table

Description:
This table has each candidates application for a specific role recorded in it. An application relates to a single job opening so there is a one-to-one relationship between each job opening and a given application. Since a candidate can submit multiple applications and an application can be subjected to many interviews a many-to-one relationship exists between Application and Candidates. A one-to-many relationship exists between Application and Interviews. An application can have many reimbursements related to it if onsite interviews relates to it so a one-to-many relation exists between the tables Application and Reimbursement. An application can also have many evaluations related to it so a one-to-many relation between the Application table and Evaluation table is also needed.

Keys:
Primary Keys: ApplicationID

Foreign Keys: JobOpening & CandidateNumber

Related Tables:
Interviews, Reimbursement, Candidates, Evaluations & Job Openings

Reimbursement Table

Description:

This table contains information on each reimbursement that is needed per application. There is a one-to-one relationship between the Reimbursement and Expense Receipts table since each reimbursement corresponds to a single receipt. There is also a many-to-one relationship between Reimbursement and Applications since each Application can have many reimbursements.

Keys:

Primary Keys: ReimbursementID

Foreign Keys: ApplicationNumber & ReciptNumber

Related Tables:

Expense Receipts & Application

Candidates Table

Description:

This table holds information on each candidate that applies for a job at the company. Each candidate has a background check, can be onboarded, can be given an offer and has documents so there is a one-to-one relationship between Candidates and Offers, Onboarding, Background Checks, and Documents tables. Since a candidate can submit multiple applications, expense receipts and complaints. A one-to-many relationship exists between Candidates and the Candidate Complaints, Application and Expense Receipts tables. This table has a trigger that increments a job opening when a candidate is blacklisted.

Keys:

Primary Keys: CandidateID

Related Tables:

Offers, Onboarding, Background Checks, Documents, Candidate Complaints, Application & Expense Receipts

Triggers:

updateJobOpeningBlacklist

Employees Table

Description:

This table tracks each employee in the department, has a single primary key and a single one-to-one relationship with Interviewers since each Interviewer is an employee.

Keys:

Primary Keys: EmployeeID

Related Tables:

Interviewers

Assigned Tests Table

Description:

This table tracks which test is assigned to which interview and since many tests can be used many different interviews this linking table is necessary. Because of this, there is a many-to-one relationship between Assigned Tests and Tests and Interviews.

Keys:

Foreign Keys: TestNumber & InterviewNumber

Related Tables:

Interviews & Tests

Tests Table

Description:

This table tracks all the tests that exists within the department and has a one-to-many relationship between itself and Assigned a test can be assigned to multiple interviews. It has the same relation with Graders since a test can have many graders that grade it.

Keys:

Primary Keys: TestID

Related Tables:

Graders & Assigned Tests

Expense Receipts Table

Description:

This table tracks all the receipts submitted for reimbursement by a candidate. A candidate can have many receipts so a many-to-one relation exists between the Expense Receipts Table and the Candidates Table. There is also a one-to-one relationship between the Expense Receipts Table and the Reimbursement table since a processing reimbursement corresponds to a specific receipt

Keys:

Primary Keys: ReceiptID

Foreign Keys: CandidateNumber

Related Tables:

Reimbursement & Candidates

Section B: Implementation

Table Implementation SQL Code

Hotels Table

Description:

This table holds information on each candidate's hotel accommodations for in person interviews.

Code:

```
CREATE TABLE hrdb.Hotels
(HotelReservationID                INT NOT NULL PRIMARY KEY IDENTITY,
ReservationNumber                  INT NOT NULL,
ReservationName                    VARCHAR(50) NOT NULL,
HotelName                          VARCHAR(50) NOT NULL,
HotelLocation                      VARCHAR(50)   NOT NULL)
```

Airfair Table

Description:

This table holds information on each candidate's airfare accommodations for in person interviews.

Code:

```
CREATE TABLE hrdb.Airfair
(AirfairID                  INT NOT NULL PRIMARY KEY IDENTITY,
TicketNumber                INT NOT NULL,
Airline                     VARCHAR(50)   NOT NULL,
AirportName                 VARCHAR(50)   NOT NULL,
AirportLocation             VARCHAR(50)   NOT NULL)
```

Rental Cars Table

Description:

This table holds information on each candidate's rental car accommodations for in person interviews.

Code:

```
CREATE TABLE hrdb.[Rental Cars]
(RentalCarID                              INT    NOT NULL PRIMARY KEY IDENTITY,
ReservationNumber                         INT    NOT NULL,
RentalCarCompanyName                      VARCHAR(50)   NOT NULL,
RentalCarCompanyLocation                  VARCHAR(50)   NOT NULL,
CarBrand                                  VARCHAR(50)   NULL DEFAULT 'Car',
CarType                                   VARCHAR(50)   NULL DEFAULT 'Car')
```

Accommodations Table

Description:

This table holds information all the accommodation information for a given onsite interview.

Code:

```
CREATE TABLE hrdb.Accomodations
(AccomodationsID            INT NOT NULL PRIMARY KEY IDENTITY,
InterviewNumber             INT NOT NULL REFERENCES hrdb.Interviews (InterviewID),
HotelDetailsNumber          INT NOT NULL REFERENCES hrdb.Hotels (HotelReservationID),
AirfairDetailsNumber        INT NOT NULL REFERENCES hrdb.Airfair (AirfairID),
RentalCarNumber             INT NOT NULL REFERENCES hrdb.[Rental Cars] (RentalCarID),
TotalCost                   MONEY NOT NULL CHECK (TotalCost >= 0))
```

Employees Table

Description:

This table holds information on each employee in the human resources department.

Code:

```
CREATE TABLE hrdb.Employees
(EmployeeID                 INT NOT NULL PRIMARY KEY IDENTITY,
EmployeeName                VARCHAR(50) NOT NULL,
Department                  VARCHAR(50) NULL DEFAULT 'No Department Specified',
Title                       VARCHAR(50) NOT NULL)
```

Interviewers Table

Description:

This table holds information on each interviewer in the human resources department.

Code:

```
CREATE TABLE hrdb.Interviewers
(InterviewerID              INT NOT NULL PRIMARY KEY IDENTITY,
EmployeeNumber              INT NOT NULL REFERENCES hrdb.Employees (EmployeeID), --FK
InterviewerName             VARCHAR(50) NOT NULL,
Department                  VARCHAR(50) NOT NULL,
Title                       VARCHAR(50) NOT NULL)
```

Candidate Reviews Table

Description:

This table holds a candidate's optional review they can provide after an interview takes place.

Code:

```
CREATE TABLE hrdb.[Candidate Reviews]
(CandidateReviewID                INT NOT NULL PRIMARY KEY IDENTITY,
Details                           VARCHAR(250) NOT NULL)
```

Interviewers Reviews Table

Description:

This table holds a interviewer's optional review they can provide after an interview takes place.

Code:

```
CREATE TABLE hrdb.[Interviewer Reviews]
(InterviewerReviewID       INT NOT NULL PRIMARY KEY IDENTITY,
Details                                 VARCHAR(250) NOT NULL)
```

Candidates Table

Description:

This table holds data on each candidate the apples for a job opening at the company.

Code:

```
CREATE TABLE hrdb.Candidates
(CandidateID                INT NOT NULL PRIMARY KEY IDENTITY,
CandidateName               VARCHAR(50) NOT NULL,
Email                       VARCHAR(50) NOT NULL,
Phone                       CHAR(12) NOT NULL,
ShortProfile                VARCHAR(500) NOT NULL,
CandidateStatus             VARCHAR(35)   NOT NULL CHECK (CandidateStatus = 'waiting' OR
                     CandidateStatus = 'rejected'
                     OR CandidateStatus = 'on-call for next job opportunity' OR
                     CandidateStatus = 'blacklisted'))
```

Job Table

Description:

This table holds data on each job position at the company.

Code:

```
CREATE TABLE hrdb.Job
(JobID                            INT NOT NULL PRIMARY KEY IDENTITY,
Position                          VARCHAR(50)   NOT NULL,
Title                             VARCHAR(50) NOT NULL,
JobDescription                    VARCHAR(50)   NOT NULL,
JobType                           CHAR(20) NOT NULL CHECK (JobType = 'Summer Internship'
              OR JobType = 'Full-time Job'
              OR JobType = 'Part-time Job' OR JobType = 'Contract-based'),
JobMedium                         CHAR(6) NOT NULL CHECK (JobMedium = 'Online' OR
              JobMedium = 'Onsite'),
NumberOfPositions                 INT NOT NULL CHECK (NumberOfPositions >= 0),
Salary                            INT NOT NULL CHECK (Salary >= 0))
```

Job Openings Table

Description:

This table holds information on each open job position at the company.

Code:

```
CREATE TABLE hrdb.[Job Openings]
(JobOpeningID                      INT NOT NULL PRIMARY KEY IDENTITY,
JobNumber                          INT NOT NULL REFERENCES hrdb.Job (JobID), --FK
JobDescription                     VARCHAR(250) NOT NULL,
NumberOfPositions                  INT NOT NULL CHECK (NumberOfPositions >= 0),
NumberOfApplicants                 INT NOT NULL CHECK (NumberOfApplicants >= 0),
JobStartDate                       DATE NOT NULL,
JobType                            CHAR(20) NOT NULL CHECK (JobType = 'Summer Internship'
              OR JobType = 'Full-time Job'
              OR JobType = 'Part-time Job' OR JobType = 'Contract-based'),
JobCatagory                        CHAR(50) NOT NULL CHECK (JobCatagory = 'IT Manager'
              OR JobCatagory = 'software design'
              OR JobCatagory = 'testing' OR JobCatagory = 'finance'
              OR JobCatagory = 'administrative' OR JobCatagory = 'hr'),
JobPlatform                        CHAR(20) NULL DEFAULT 'Other' CHECK (JobPlatform =
              'Online Job Board' OR JobPlatform = 'Company Webpage'
              OR JobPlatform = 'Other'))
```

Application Table

Description:

This table holds information on each application submitted by a specific candidate, a single candidate can submit multiple applications.

Code:

```
CREATE TABLE hrdb.[Application]
(ApplicationID              INT NOT NULL PRIMARY KEY IDENTITY,
JobOpening                  INT NOT NULL REFERENCES hrdb.[Job Openings] (JobOpeningID),
CandidateNumber             INT NOT NULL REFERENCES hrdb.Candidates (CandidateID),
ApplicationDate             DATE NOT NULL,
ApplicationStatus           CHAR(8) NOT NULL CHECK (ApplicationStatus = 'Pending'
                            OR ApplicationStatus = 'Declined'
                            OR ApplicationStatus = 'Accepted'));
```

Interviews Table

Description:

This table holds information on each interview taking place for a given application and candidate.

Code:

```
CREATE TABLE hrdb.Interviews
(InterviewID         INT NOT NULL PRIMARY KEY IDENTITY,
CandidateReview      INT NULL REFERENCES hrdb.[Candidate Reviews] (CandidateReviewID),
InterviewerReview    INT NULL REFERENCES hrdb.[Interviewer Reviews]
                     (InterviewerReviewID),
ApplicationNumber    INT NOT NULL REFERENCES hrdb.[Application] (ApplicationID),
InterviewType        CHAR(6) NOT NULL CHECK (InterviewType = 'Onsite' OR InterviewType =
               'Online'),
InterviewRound       INT NOT NULL,
InterviewStartTime   TIME(5) NOT NULL,
InterviewEndTime     TIME(5) NULL)
```

Evaluation Table

Description:

This table holds evaluations an interviewer submits after an interview regarding the candidate being interviewed.

Code:

```
CREATE TABLE hrdb.Evaluation
(EvaluationID               INT    NOT NULL PRIMARY KEY IDENTITY,
ApplicationNumber           INT    NOT NULL REFERENCES hrdb.[Application] (ApplicationID),
InterviewerNumber           INT NOT NULL REFERENCES hrdb.Interviewers (InterviewerID),
EvaluationNotes             VARCHAR(500),
Result                      CHAR(4) NOT NULL CHECK (Result = 'Pass' OR Result = 'Fail'))
```

Tests Table

Description:

This table contains each test the company uses to evaluate candidates, a given interview can have many tests and test can have many graders.

Code:

```
CREATE TABLE hrdb.Tests
(TestID              INT NOT NULL PRIMARY KEY IDENTITY,
TestType             CHAR(6) NOT NULL CHECK (TestType = 'Online' OR TestType = 'Onsite'),
TestStartTime        TIME(5) NOT NULL,
TestEndTimes         TIME(5) NOT NULL,
Grade                CHAR(6) NOT NULL CHECK (Grade = 'passed' OR Grade = 'failed'),
Answers              VARCHAR(150) NOT NULL)
```

Assigned Interviews Table

Description:

This table is a linking table that keeps track of which interviewer is assigned to which interview.

Code:

```
CREATE TABLE hrdb.[Assigned Interviews]
(InterviewNumber     INT NOT NULL REFERENCES hrdb.Interviews (InterviewID),
InterviewerNumber    INT NOT NULL REFERENCES hrdb.Interviewers (InterviewerID))
```

Graders Table

Description:

This table is a linking table that keeps track of which interviewer is assigned to grade which test, a single test can have multiple graders.

Code:

```
CREATE TABLE hrdb.Graders
(InterviewerNumber   INT NOT NULL REFERENCES hrdb.Interviewers (InterviewerID),
TestNumber                  INT NOT NULL REFERENCES hrdb.Tests (TestID))
```

Assigned Tests Table

Description:

This table is a linking table that keeps track of which tests are given during a specific interview.

Code:

```
CREATE TABLE hrdb.[Assigned Tests]
(TestNumber          INT NOT NULL REFERENCES hrdb.Tests (TestID),
InterviewNumber      INT NOT NULL REFERENCES hrdb.Interviews (InterviewID))
```

Expense Receipts Table

Description:

This table keeps track of all the receipts submitted for reimbursement by candidates.

Code:

```
CREATE TABLE hrdb.[Expense Recipts]
(ReciptID                   INT NOT NULL PRIMARY KEY IDENTITY,
CandidateNumber             INT NOT NULL REFERENCES hrdb.Candidates (CandidateID),
AmountOnReceipt             MONEY NOT NULL,
Request                     VARCHAR(250) NOT NULL)
```

Reimbursements Table

Description:

This table keeps track of all the reimbursements needed for a given application.

Code:

```
CREATE TABLE hrdb.Reimbursement
(ReimbursementID            INT NOT NULL PRIMARY KEY IDENTITY,
ApplicationNumber           INT NOT NULL REFERENCES hrdb.[Application] (ApplicationID) ,
ReciptNumber                INT NOT NULL REFERENCES hrdb.[Expense Recipts] (ReciptID),
Request                     VARCHAR(250) NOT NULL,
Processed                   BIT NULL DEFAULT 0,
Amount                      MONEY NOT NULL CHECK (Amount >= 0))
```

Offers Table

Description:

This table keeps track of all the offers extended to candidates whose applications made it through the interview process and are desired hires by the company.

Code:

```
CREATE TABLE hrdb.Offers
(OfferID                          INT NOT NULL PRIMARY KEY IDENTITY,
JobNumber                         INT    NOT NULL REFERENCES hrdb.Job (JobID),
CandidateNumber                   INT NOT NULL REFERENCES hrdb.Candidates (CandidateID),
OfferDate                         DATE NOT NULL,
OfferStatus                       CHAR(15) NOT NULL CHECK (OfferStatus = 'Declined' OR
              OfferStatus = 'Accepted'
              OR OfferStatus = 'Negotiating' OR OfferStatus = 'Offer Extended'))
```

Onboarding Table

Description:

This table holds information on each candidate the accepted an offer to join the company.

Code:

```
CREATE TABLE hrdb.Onboarding
(CandidateNumber     INT NOT NULL PRIMARY KEY REFERENCES hrdb.Candidates (CandidateID),
OfferNumber          INT NOT NULL REFERENCES hrdb.Offers (OfferID),
StartDate            DATE NOT NULL,
EndDate              DATE NOT NULL)
```

Background Checks Table

Description:

This table holds information on background checks ran on each candidate during the onboarding.

Code:

```
CREATE TABLE hrdb.[Background Checks]
(CandidateNumber     INT NOT NULL PRIMARY KEY REFERENCES hrdb.Candidates (CandidateID),
OfferNumber          INT NOT NULL REFERENCES hrdb.Offers (OfferID),
Grade                CHAR(6) NOT NULL CHECK (Grade = 'passed' OR Grade = 'failed'),
CheckDate            DATE NOT NULL)
```

Documents Table

Description:

This table holds information on each candidates' documents needed for employment.

Code:

```
CREATE TABLE hrdb.Documents
(CandidateNumber     INT NOT NULL PRIMARY KEY REFERENCES hrdb.Candidates (CandidateID),
CV                   VARCHAR(500) NULL DEFAULT 'Not Provided',
CoverLetter          VARCHAR(500) NULL DEFAULT 'Not Provided',
ProvidedResume       VARCHAR(500) NOT NULL,
ReferenceLetters     VARCHAR(500) NOT NULL,
HasValidDocuments    BIT NULL DEFAULT  0)
```

Candidate Complaints Table

Description:

This table holds information on potential complaints submitted by candidates about the hiring process.

Code:

```
CREATE TABLE hrdb.[Candidate Complaints]
(CandidateNumber            INT NOT NULL PRIMARY KEY REFERENCES hrdb.Candidates
                     (CandidateID),
ComplaintDescription        VARCHAR(500) NOT NULL,
ComplaintDate               DATE NOT NULL)
```

Stored Procedures Implementation SQL Code

Add Candidate Stored Procedure

Description:

This stored procedure can be used to add a candidate into the system, more specifically, into the candidates table and allows basic information to be provided

Code:

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

CREATE PROC spAddCandidate
	@Name VARCHAR(50),
	@Email VARCHAR(50),
	@Phone CHAR(12),
	@Profile VARCHAR(500)
AS
	BEGIN
		INSERT INTO hrdb.Candidates
		VALUES (@Name, @Email, @Phone, @Profile, 'waiting')
	END
```

Offer Decision Stored Procedure

Description:

This stored procedure allows a candidate to accept or decline a job offer, if declined the candidate's status is changed to on-call for next job opportunity.

Code:

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

CREATE PROC spOfferDecision
	@CandidateID INT,
	@Decision BIT
AS
	IF @Decision = 1
		UPDATE hrdb.Offers
		SET OfferStatus = 'Accepted'
		WHERE CandidateNumber = @CandidateID;
	IF @Decision = 0
		UPDATE hrdb.Offers
		SET OfferStatus = 'Declined'
		WHERE CandidateNumber = @CandidateID;
		UPDATE hrdb.Candidates
		SET CandidateStatus = 'on-call for next job opportunity'
		WHERE CandidateID = @CandidateID;
```

Reject Candidate Stored Procedure

Description:

This stored procedure allows a candidate to be rejected from a job opening they applied for.

Code:

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

CREATE PROC spRejectCandidate
	@CandidatesName VARCHAR(50),
	@CandidatesNumber INT
AS
	UPDATE hrdb.Candidates
	SET CandidateStatus = 'rejected'
	WHERE CandidateID = @CandidatesNumber AND CandidateName = @CandidatesName;
```

Select Candidate For Interview Stored Procedure

Description:

This stored procedure allows a candidate to be selected for an interview

Code:

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

CREATE PROC spSelectCandidateForInterview
	@ApplicationNumber INT,
	@Type CHAR(6),
	@Round INT,
	@Time TIME(5)
AS
	INSERT INTO hrdb.Interviews
	VALUES (NULL, NULL, @ApplicationNumber, @Type, @Round, @Time, NULL);
```

User Defined Functions Implementation SQL Code

Number of Open Positions User Defined Function

Description:

This user defined function can be used to determine how many open positions for a provided job number there are.

Code:

```
CREATE FUNCTION hrdb.fnNumberOfOpenPositions
       (@JobIdentification INT)
       RETURNS INT
BEGIN
       RETURN(SELECT hrdb.[Job Openings].NumberOfPositions
                 FROM hrdb.[Job Openings] JOIN hrdb.Job
                            ON Job.JobID = [Job Openings].JobNumber
                 WHERE JobNumber = @JobIdentification);
END;
```

Passed Background Check User Defined Function

Description:

This user defined function can be used to determine if a provided candidate has passed a background check.

Code:

```
CREATE FUNCTION hrdb.fnPassedBackgroundCheck
       (@CandidateNumber INT)
       RETURNS CHAR(6)
BEGIN
       RETURN(SELECT hrdb.[Background Checks].Grade
                 FROM hrdb.[Background Checks]
                 WHERE CandidateNumber = @CandidateNumber);
END;
```

Get Candidate Complaints User Defined Function

Description:

This user defined function can be used to find all the complaints filed by a provided candidate.

Code:

```
CREATE FUNCTION hrdb.fnGetCandidateComplaints
       (@CandidateNumber INT)
       RETURNS TABLE
RETURN (SELECT ComplaintDescription AS [Complaint], ComplaintDate AS [Date Filed]
                FROM hrdb.[Candidate Complaints]
                WHERE CandidateNumber = @CandidateNumber);
```

Get Interview Reviews User Defined Function

Description:

This user defined function can be used to find the candidate and interviewer reviews for a given interview number.

Code:

```
CREATE FUNCTION hrdb.fnGetInterviewReviews
       (@InterviewNumber INT)
       RETURNS TABLE
RETURN (SELECT [Candidate Reviews].Details AS [Candidate Feedback], [Interviewer
Reviews].Details AS [Interviewer Feedback]
                FROM hrdb.[Candidate Reviews]
                     JOIN hrdb.Interviews ON Interviews.CandidateReview = [Candidate
Reviews].CandidateReviewID
                     JOIN hrdb.[Interviewer Reviews] ON Interviews.InterviewerReview =
[Interviewer Reviews].InterviewerReviewID
                WHERE Interviews.InterviewID = @InterviewNumber);
```

Transactions & Trigger Implementation SQL Code

Verify Candidate Trigger

Description:

This trigger verifies a candidate has the correct documents and passes a background check before being added to the onboarding table.

Code:

```
CREATE TRIGGER verifyCandidate
       ON hrdb.Offers
       AFTER INSERT
AS
       IF (SELECT Grade FROM hrdb.[Background Checks] WHERE CandidateNumber = (SELECT
       CandidateNumber FROM Inserted)) = 'failed'
              OR (SELECT HasValidDocuments FROM hrdb.Documents WHERE CandidateNumber =
              (SELECT CandidateNumber FROM Inserted)) = 0
              BEGIN
                     UPDATE hrdb.Candidates
                     SET CandidateStatus = 'on-call for next job opportunity'
                     WHERE CandidateID = (SELECT CandidateNumber FROM Inserted)
                     PRINT 'This candidate has either failed a background check or didnt
submit proper documents.'
                     PRINT 'Candidate status updated to on-call for next job
opportunity.'
              END;
GO
```

Process Complaint Trigger with a Transaction

Description:

This trigger puts a rejected candidate into waiting if they file a complaint, the update part of this trigger is in a transaction, so a candidate is moved without interruption.

Code:

```
CREATE TRIGGER processComplaint
       ON hrdb.[Candidate Complaints]
       AFTER INSERT
AS
       IF (SELECT CandidateStatus FROM hrdb.Candidates WHERE CandidateID = (SELECT
       CandidateNumber FROM Inserted)) = 'rejected'
              BEGIN
                     BEGIN TRAN
                            UPDATE hrdb.Candidates
                            SET CandidateStatus = 'waiting'
                            WHERE CandidateID = (SELECT CandidateNumber FROM Inserted)
                            PRINT 'Candidate complaint recieved. Your profile has been
placed in waiting.'
                     COMMIT TRAN
              END;
GO
```

Update Number of Job Openings Trigger with a Transaction

Description:

This trigger decrements the job opening number of open positions when an offer is accepted, a trigger surrounds the updating of the number of open positions to ensure it remains accurate.

Code:

```
CREATE TRIGGER updateNumberOfJobOpenings
       ON hrdb.Offers
       AFTER UPDATE
AS
       IF (SELECT OfferStatus FROM hrdb.Offers WHERE CandidateNumber = (SELECT
       CandidateNumber FROM Inserted)) = 'Offer Extended'
              BEGIN
                     BEGIN TRAN
                            UPDATE hrdb.[Job Openings]
                            SET NumberOfPositions = NumberOfPositions - 1
                            WHERE JobNumber = (SELECT JobNumber FROM Inserted)
                            PRINT 'Offer extended. Number of openings reduced
accordingly.'
                     COMMIT TRAN
              END;
GO
```

Update Number of Job Openings On Decline Trigger with a Transaction

Description:

This trigger increments the job opening number of open positions when an offer is declined, a trigger surrounds the updating of the number of open positions to ensure it remains accurate.

Code:

```
CREATE TRIGGER updateNumberOfJobOpeningsOnDecline
       ON hrdb.Offers
       AFTER UPDATE
AS
       IF (SELECT OfferStatus FROM hrdb.Offers WHERE CandidateNumber = (SELECT
       CandidateNumber FROM Inserted)) = 'Declined'
              BEGIN
                     BEGIN TRAN
                            UPDATE hrdb.[Job Openings]
                            SET NumberOfPositions = NumberOfPositions + 1
                            WHERE JobNumber = (SELECT JobNumber FROM Inserted)
                            PRINT 'Offer declined. Number of openings increased
accordingly.'
                     COMMIT TRAN
              END;
GO
```

Update Job Opening On Blacklist with a Transaction

Description:

This trigger increments the job opening number of open positions when a candidate is blacklisted, a trigger surrounds the updating of the number of open positions to ensure it remains accurate.

Code:

```
CREATE TRIGGER updateJobOpeningOnBlacklist
       ON hrdb.Candidates
       AFTER UPDATE
AS
       IF (SELECT OfferStatus FROM hrdb.Offers WHERE CandidateNumber = (SELECT
CandidateNumber FROM Inserted)) = 'Declined'
              BEGIN
                     BEGIN TRAN
                            UPDATE hrdb.[Job Openings]
                            SET NumberOfPositions = NumberOfPositions + 1
                            WHERE JobNumber = (SELECT JobOpening FROM hrdb.Application
WHERE CandidateNumber = (SELECT CandidateID FROM Inserted))
                            PRINT 'Candidate blacklisted. Number of openings increased
accordingly.'
                     COMMIT TRAN
              END;
GO
```

Scripts Implementation SQL Code

Script #1

Description:

This script creates a new interviewer login with a user named John Doe that allows him to execute stored procedures on tables that have to do with interviewing and the ability to read all the data in the database.

Code:

```
CREATE LOGIN HumanResourcesInterviewer WITH PASSWORD = '12345678',
       DEFAULT_DATABASE = HumanResources;

CREATE USER JohnDoe
       FOR LOGIN HumanResourcesInterviewer
       WITH DEFAULT_SCHEMA = hrdb

GRANT EXECUTE
ON hrdb.Interviews
TO JohnDoe

GRANT EXECUTE
ON hrdb.Evaluation
TO JohnDoe

GRANT EXECUTE
ON hrdb.[Interviewer Reviews]
TO JohnDoe

ALTER ROLE db_datareader ADD MEMBER JohnDoe
```

Script #2

Description:

This script creates an employee login, new user, and a new role. The user gains the role of AccomodationAccess allowing Megan Smith the ability to make changes to tables related to accommodations. It also allows her to read all information on the database using db_datareader.

Code:

```
CREATE LOGIN HumanResourcesAccomodationSpecialist WITH PASSWORD = 'coolpassword',
       DEFAULT_DATABASE = HumanResources

CREATE USER MeganSmith
       FOR LOGIN HumanResourcesAccomodationSpecialist
       WITH DEFAULT_SCHEMA = hrdb

CREATE ROLE AccomodationAccess;

GRANT INSERT, UPDATE, DELETE
ON hrdb.Accomodations
TO AccomodationAccess

GRANT INSERT, UPDATE, DELETE
ON hrdb.Hotels
TO AccomodationAccess
```

```
GRANT INSERT, UPDATE, DELETE
ON hrdb.[Rental Cars]
TO AccomodationAccess

GRANT INSERT, UPDATE, DELETE
ON hrdb.Airfair
TO AccomodationAccess

ALTER ROLE AccomodationAccess ADD MEMBER MeganSmith
ALTER ROLE db_datareader ADD MEMBER JohnDoe
```

Script #3

Description:

This script creates an employee login, new user, and a new role. The user gains the role of Job Access allowing Richard Welling to perform various actions on job related tables. He can also back up the database.

Code:

```
CREATE LOGIN HumanResourcesJobSpecialist WITH PASSWORD = 'jobpassword',
       DEFAULT_DATABASE = HumanResources

CREATE USER RichardWelling
       FOR LOGIN HumanResourcesJobSpecialist
       WITH DEFAULT_SCHEMA = hrdb

CREATE ROLE JobAccess;

GRANT INSERT, UPDATE, DELETE, REFERENCES
ON hrdb.Job
TO JobAccess

GRANT INSERT, UPDATE, DELETE, REFERENCES
ON hrdb.[Job Openings]
TO JobAccess

ALTER ROLE JobAccess ADD MEMBER RichardWelling
ALTER ROLE db_backupoperator ADD MEMBER RichardWelling
```

Script #4

Description:

This script creates an employee login and new user. The user has many admin roles, the ability to back up the database and read everything on it.

Code:

```
CREATE LOGIN HumanResourcesDatabaseAdmin WITH PASSWORD = 'dbpassword',
       DEFAULT_DATABASE = HumanResources

CREATE USER JackWillis
       FOR LOGIN HumanResourcesDatabaseAdmin
       WITH DEFAULT_SCHEMA = hrdb

ALTER ROLE db_accessadmin ADD MEMBER JackWillis
ALTER ROLE db_securityadmin ADD MEMBER JackWillis
ALTER ROLE db_ddladmin ADD MEMBER JackWillis
ALTER ROLE db_datareader ADD MEMBER JackWillis
ALTER ROLE db_backupoperator ADD MEMBER JackWillis
```

Business Reports Implementation SQL Code

Number of Applications In a Time Period Business Report

Description:

This business report returns the number of applications received in the current year.

Code:

```
CREATE PROC spNumberOfApplicationsInTimePeriod
       @StartDate DATE,
       @EndDate DATE
AS
SELECT COUNT(*) AS [Number of Interviews This Year]
FROM hrdb.Application
WHERE ApplicationDate >= @StartDate AND ApplicationDate <= @EndDate
GO
```

Average Recruitment Time Business Report

Description:

This business report that returns the average time for the recruitment process.

Code:

```
CREATE PROC spAverageRecruitmentTime
AS
WITH Times AS
(
       SELECT ApplicationDate, OfferDate
       FROM hrdb.[Application] JOIN hrdb.Offers
              ON Application.CandidateNumber = Offers.CandidateNumber
)
SELECT AVG(DATEDIFF(day, Times.ApplicationDate, Times.OfferDate)) AS [Average Recruitment
Time (Days)]
FROM Times
GO
```

Offer Rate Business Report

Description:

This business that returns the percent of candidates hired that applied.

Code:

```
CREATE PROC spOfferRate
AS
DECLARE @PercentAns DECIMAL(5, 1);
DECLARE @NumApps DECIMAL(5, 1);
DECLARE @NumOffs DECIMAL(5, 1);
SET @NumApps = CONVERT(DECIMAL(5, 1), (SELECT COUNT(*) FROM hrdb.[Application]), 1)
SET @NumOffs = CONVERT(DECIMAL(5, 1), (SELECT COUNT(*) FROM hrdb.Offers), 1)
SET @PercentAns =  (@NumOffs / @NumApps) * 100
PRINT 'Application Acceptance Rate: ' + CONVERT(varchar, @PercentAns, 1) + '%'
GO
```

Average Cost of Onsite Interview Business Report

Description:

This business that returns the average cost of onsite interviews.

Code:

```
CREATE PROC spAverageCostOfOnsiteInterviews
AS
WITH Costs AS
(
	SELECT TotalCost AS [Accomodations Cost], SUM(Amount) AS [Total Reimbursement
Cost]
	FROM hrdb.Accomodations
		JOIN hrdb.Interviews ON Accomodations.InterviewNumber =
Interviews.InterviewID
		JOIN hrdb.[Application] ON Interviews.ApplicationNumber =
[Application].ApplicationID
		JOIN hrdb.Candidates ON [Application].CandidateNumber =
Candidates.CandidateID
		JOIN hrdb.Reimbursement ON [Application].ApplicationID =
Reimbursement.ApplicationNumber
	GROUP BY InterviewID, CandidateName, TotalCost
),
TotalCosts AS
(
	SELECT [Accomodations Cost] + [Total Reimbursement Cost] AS [Total Onsite Cost]
	FROM Costs
)
SELECT AVG([Total Onsite Cost]) AS [Average Onsite Interview Cost]
FROM TotalCosts
GO
```

Section D: Testing

Database Population SQL Code

Hotels Table

Code:

```
INSERT INTO hrdb.Hotels
VALUES (7777777, 'Cayde Six', 'Tower Suites', 'Last City Street'),
          (5930335, 'Joe Garcia', 'Marriot', '31st Province Street');
```

Airfair Table

Code:

```
INSERT INTO hrdb.Airfair
VALUES (7777777, 'Vanguard Airlines', 'Hancock International', 'Syracuse'),
       (7327987, 'Jetblue', 'Chicago International', 'Chicago'),
          (1490531, 'Southwest', 'Hancock International', 'Syracuse'),
          (5903211, 'Delta', 'LAX', 'Los Angeles');
```

Rental Cars Table

Code:

```
INSERT INTO hrdb.[Rental Cars]
VALUES (7777777, 'Hertz', 'Elm Street', 'Ford', 'SUV'),
          (3245412, 'AVIS', '42nd Street', 'Volvo', 'Sudan');
       VARCHAR(50)   NULL DEFAULT 'Car')
```

Accommodations Table

Code:

```
INSERT INTO hrdb.Accomodations
VALUES (4, 1, 1, 1, 629),
          (5, 2, 3, 2, 831);
```

Employees Table

Code:

```
INSERT INTO hrdb.Employees
VALUES ('Dean Rodriguez', 'Human Resources', 'Human Resources Director'),
              ('Lisa Hicks', 'Leadership', 'Cheif Executive Officer'),
              ('Brian Jefferson', 'Engineering', 'Senior Software Engineer'),
              ('Victoria Brown', 'Engineering', 'Web Developer'),
              ('Jessica Cardenas', 'Engineering', 'Web Developer'),
              ('Sharon Smith', 'Engineering', 'Web Developer'),
              ('Julie Johnson', 'Engineering', 'Web Developer'),
              ('Devin Reynolds', 'Engineering', 'Web Developer'),
              ('Richard Williams', 'Engineering', 'Web Developer'),
              ('Terry Gomez', 'Engineering', 'Web Developer'),
              ('Paul Navarro', 'Engineering', 'Web Developer'),
              ('Megan Brandt', 'Engineering', 'Web Developer'),
              ('Jennifer Dunn', 'Engineering', 'Backend Engineer'),
              ('Julie Phillips', 'Engineering', 'Backend Engineer'),
```

```
            ('Betty Callahan', 'Engineering', 'Backend Engineer'),
            ('Ronald Robinson', 'Engineering', 'Software Engineer Intern'),
            ('April Lindsey', 'Engineering', 'Software Engineer Intern'),
            ('Robert Parker', 'Customer Relations', 'Customer Liaison'),
            ('Regina Cook', 'Customer Relations', 'Customer Liaison'),
            ('Timothy Mason', 'Customer Relations', 'Customer Liaison'),
            ('Christina Randall', 'Legal', 'Legal Advisor');
```

Interviewers Table

Code:

```
INSERT INTO hrdb.Interviewers
VALUES (3, 'Brian Jefferson', 'Engineering', 'Senior Software Engineer'),
         (7, 'Julie Johnson', 'Engineering', 'Web Developer'),
         (4, 'Victoria Brown', 'Engineering', 'Web Developer'),
         (15, 'Betty Callahan', 'Engineering', 'Backend Engineer'),
         (16, 'Ronald Robinson', 'Engineering', 'Software Engineer Intern'),
         (21, 'Christina Randall', 'Legal', 'Legal Advisor'),
         (18, 'Robert Parker', 'Customer Relations', 'Customer Liaison');
```

Candidate Reviews Table

Code:

```
INSERT INTO hrdb.[Candidate Reviews]
VALUES ('Interview went well, I really thought it was fair and I liked the interviewer'),
         ('The interview was tought but I enjoyed the interviewers feedback'),
         ('I dont think the interview was very fair but I did well'),
         ('Your interview system is flawed and broken, FIX IT!'),
         ('Overall everything went well, I hope I will get an offer.'),
         ('I dont think I did very well, but the interview was fair'),
         ('I really liked the interview questions but the interviewer was mean :('),
         ('I love your interview system, well done!'),
         ('Nothing to report everything was as expected'),
         ('I think the second question is flawed but the others are fine'),
         ('The questions did not match the job description');
```

Interviewers Reviews Table

Code:

```
INSERT INTO hrdb.[Interviewer Reviews]
VALUES ('Interviewee did well, knows there fundamentals, would make a good hiring'),
         ('Fantastic candidate and will bring value to our company'),
         ('Didnt do well, lacked some basic knowledge, wouldnt hire'),
         ('knew the answers to the questions but didnt communicate well'),
         ('Strong candidate, would hire'),
         ('Candidate was late but did well otherwise'),
         ('Did okay would hire in the future, but not what the team needs now'),
         ('Great fit for our culture and did well enough on the questions, hire'),
         ('Lacked communication and was too concerned about the pay, avoid'),
         ('Knew there stuff please hire'),
         ('Not good at all, very combative about wrong answers and doesnt fit out
culture');
```

Candidates Table

Code:

```
INSERT INTO hrdb.Candidates
VALUES ('Timothy Zimmerman', 'tzimmerman@gmail.com', '315-223-2923', 'Timothys Profile',
'waiting'),
             ('Jessica Davis', 'jdavis@gmail.com', '315-387-5011', 'Jessicas Profile',
'waiting'),
             ('Michael Mcclure', 'mmcclure@gmail.com', '412-791-9222', 'Michaels
Profile', 'waiting'),
             ('Jessica Mata', 'jmata@gmail.com', '315-387-3214', 'Jessicas Profile',
'waiting'),
             ('Larry Reyes', 'lreyes@gmail.com', '315-347-4071', 'Larrys Profile',
'waiting'),
             ('Cheryl Flores', 'cflores@gmail.com', '315-738-1150', 'Cheryls Profile',
'waiting'),
             ('Lorraine Ray', 'lray@gmail.com', '412-391-3282', 'Lorraines Profile',
'waiting'),
             ('Wendy Allen', 'wallen@gmail.com', '412-231-4123', 'Wendys Profile',
'waiting'),
             ('Misty Watson', 'mwatson@gmail.com', '412-799-3277', 'Mistys Profile',
'waiting'),
             ('Dennis Hanna', 'dhanna@gmail.com', '315-493-2943', 'Dennises Profile',
'waiting'),
        ('Jennifer Garner', 'jgarner@gmail.com', '315-332-5012', 'Jennifers Profile',
'waiting'),
              ('Stephen Delacruz', 'sdelacruz@gmail.com', '412-796-9320', 'Stephens
Profile', 'on-call for next job opportunity'),
            ('Richard Long', 'rlong@gmail.com', '532-490-3285', 'Richards Profile',
'waiting'),
            ('Amanda Walter', 'awalter@gmail.com', '412-543-1121', 'Amandas Profile',
'waiting'),
            ('Gabrielle Barber', 'gbarber@gmail.com', '315-231-2112', 'Gabrielles
Profile', 'waiting'),
            ('Cayde Six', 'hunterVanguard@bungie.net', '777-777-7777', 'Aces Profile',
'waiting'),
            ('Joe Garcia', 'jgarcia@gmail.com', '532-054-3424', 'Joes Profile',
'waiting'),
            ('Andrew Gonzalez', 'agonzalez@gmail.com', '314-409-2254', 'Andrews Profile',
'waiting'),
            ('Joann Kelly', 'jkelly@gmail.com', '315-868-9281', 'Joanns Profile',
'waiting'),
            ('Jason Montgomery', 'jmontgomery@gmail.com', '532-541-1551', 'Jasons
Profile', 'rejected'),
                                ('Jack Willis', 'jackwillis517@gmail.com', '315-663-
                    1705', 'Jacks Profile', 'blacklisted');
```

Job Table

Code:

```
INSERT INTO hrdb.Job
VALUES ('Senior Supervisor', 'Human Resources Directory', 'hr director description',
'Full-time Job', 'Onsite', 1, 150000),
       ('Founder and Leader', 'Chief Executive Officer', 'ceo description', 'Full-time
Job', 'Onsite', 1, 250000),
          ('Engineer Leader', 'Senior Software Engineer', 'senior swe description',
'Full-time Job', 'Onsite', 3, 200000),
          ('Web Developer', 'Software Engineer', 'swe description', 'Full-time Job',
'Online', 12, 90000),
          ('Cloud Infrastructure Developer', 'Backend Engineer', 'backend engineer
description', 'Full-time Job', 'Online', 4, 125000),
          ('Database Supervisor', 'Database Administrator', 'dba description', 'Contract-
based', 'Online', 1, 120000),
          ('SWE Assistant', 'Software Engineer Intern', 'swe intern description', 'Summer
Internship', 'Online', 3, 30000),
          ('Customer Communicator', 'Customer Liaison', 'customer liaison description',
'Part-time Job', 'Onsite', 3, 75000),
          ('Office Assistant', 'Office Intern', 'office intern description', 'Part-time
Job', 'Onsite', 2, 30000),
          ('Legal Advisor', 'Company Counselor', 'company counselor description', 'Full-
time Job', 'Onsite', 2, 175000),
          ('Legal Assistant', 'Law Intern', 'law intern description', 'Summer
Internship', 'Onsite', 1, 30000);
```

Job Openings Table

Code:

```
INSERT INTO hrdb.[Job Openings]
VALUES (3, 'senior swe stuff', 2, 2, '05/15/2023', 'Full-time Job', 'IT Manager',
'Company Webpage'),
       (7, 'software intern stuff', 1, 2, '04/25/2022', 'Summer Internship', 'software
design', 'Online Job Board'),
          (11, 'legal intern stuff', 1, 3, '03/01/2023', 'Summer Internship', 'finance',
'Other'),
          (4, 'web dev stuff', 3, 4, '12/25/2022', 'Full-time Job', 'software design',
'Online Job Board'),
          (6, 'dba stuff', 1, 1, '12/25/2022', 'Contract-based', 'IT Manager', 'Company
Webpage'),
          (5, 'cloud stuff', 1, 1, '01/14/2023', 'Full-time Job', 'software design',
'Online Job Board'),
          (9, 'office intern stuff', 2, 3, '10/09/2023', 'Part-time Job', 'hr', 'Other'),
          (10, 'legal stuff', 1, 2, '02/14/2023', 'Full-time Job', 'administrative',
'Company Webpage');
```

Application Table

Code:

```
INSERT INTO hrdb.Application
VALUES (1, 1, '12/03/2022', 'Accepted'),
(1, 2, '10/01/2022', 'Pending'),
(2, 4, '12/01/2021', 'Accepted'),
(2, 3, '12/25/2021', 'Accepted'),
(3, 5, '11/15/2022', 'Pending'),
(3, 6, '09/05/2022', 'Pending'),
(3, 7, '07/07/2022', 'Pending'),
(4, 8, '10/21/2022', 'Pending'),
(4, 9, '09/24/2022', 'Pending'),
(4, 10, '08/30/2022', 'Pending'),
(4, 11, '09/29/2022', 'Declined'),
(5, 12, '10/24/2022', 'Accepted'),
(6, 13, '12/13/2022', 'Pending'),
(7, 14, '04/04/2023', 'Pending'),
(7, 15, '12/06/2022', 'Pending'),
(7, 16, '11/03/2022', 'Pending'),
(8, 17, '01/15/2023', 'Pending'),
(8, 18, '12/14/2022', 'Pending'),
(4, 19, '09/11/2022', 'Declined'),
(2, 20, '10/11/2021', 'Declined'),
(5, 21, '12/05/2022', 'Declined');
```

Interviews Table

Code:

```
INSERT INTO hrdb.Interviews
VALUES (1, 1, 24, 'Online', 1,'09:45:00', '11:00:00'),
              (2, 2, 27, 'Online', 2,'09:00:00', '10:30:00'),
              (3, 3, 28, 'Online', 1,'09:00:00', '10:30:00'),
              (4, 4, 29, 'Onsite', 1,'08:00:00', '12:45:00'),
              (5, 5, 30, 'Onsite', 2,'08:00:00', '12:45:00'),
              (6, 6, 31, 'Online', 1,'09:30:00', '11:45:00'),
              (7, 7, 32, 'Online', 1,'09:30:00', '11:50:00'),
              (8, 8, 35, 'Online', 1,'09:30:00', '11:45:00'),
              (9, 9, 36, 'Online', 2,'10:00:00', '11:00:00'),
              (10, 10, 37, 'Online', 1,'10:00:00', '11:00:00'),
              (11, 11, 38, 'Online', 1,'10:00:00', '11:00:00');
```

Evaluation Table

Code:

```
INSERT INTO hrdb.Evaluation
VALUES (24, 1, 'Example Notes', 'Pass'),
          (27, 6, 'Example Notes', 'Pass'),
          (28, 6, 'Example Notes', 'Fail'),
          (29, 6, 'Example Notes', 'Pass'),
          (30, 2, 'Example Notes', 'Fail'),
          (31, 3, 'Example Notes', 'Pass'),
          (32, 3, 'Example Notes', 'Pass'),
          (35, 4, 'Example Notes', 'Pass'),
          (36, 7, 'Example Notes', 'Pass'),
          (37, 7, 'Example Notes', 'Fail'),
          (38, 7, 'Example Notes', 'Pass');
```

Tests Table

Code:

```
INSERT INTO hrdb.Tests
VALUES ('Online', '10:00:00', '10:34:11', 'passed', 'Web Dev Answer Examples'),
          ('Online', '10:00:00', '10:54:43', 'failed', 'Web Dev Answer Examples'),
          ('Online', '09:15:00', '10:12:32', 'passed', 'Web Dev Answer Examples'),
          ('Online', '09:15:00', '10:02:12', 'passed', 'Web Dev Answer Examples'),
          ('Onsite', '08:30:00', '12:30:13', 'passed', 'Senior SWE Answer Examples'),
          ('Onsite', '08:30:00', '12:35:42', 'passed', 'Senior SWE Answer Examples'),
          ('Online', '10:00:00', '11:31:11', 'passed', 'Legal Answer Examples'),
          ('Online', '10:00:00', '11:47:24', 'passed', 'Legal Answer Examples'),
          ('Online', '10:00:00', '11:11:56', 'passed', 'Legal Answer Examples'),
          ('Online', '10:15:00', '10:43:25', 'passed', 'Office Answer Examples'),
          ('Online', '10:15:00', '10:29:27', 'passed', 'Office Answer Examples'),
          ('Online', '10:15:00', '10:57:05', 'passed', 'Office Answer Examples'),
          ('Online', '10:00:00', '11:11:56', 'passed', 'Clould Engineering Hard Answer
Examples'),
          ('Onsite', '10:00:00', '11:11:56', 'passed', 'Database Hard Answer Examples'),
          ('Onsite', '09:30:00', '13:22:16', 'passed', 'Legal Hard Answer Examples'),
          ('Onsite', '09:30:00', '13:41:37', 'passed', 'Legal Hard Answer Examples'),
          ('Online', '10:15:00', '10:39:36', 'passed', 'Software Engineering Easy Answer
Examples'),
          ('Online', '10:00:00', '10:43:22', 'passed', 'Software Engineering Easy Answer
Examples');
```

Assigned Interviews Table

Code:

```
INSERT INTO hrdb.[Assigned Interviews]
VALUES (1, 1),
             (2, 6),
             (3, 6),
             (4, 6),
             (5, 2),
             (6, 3),
             (7, 3),
             (8, 4),
             (9, 7),
             (10, 7),
             (11, 7);
```

Graders Table

Code:

```
INSERT INTO hrdb.Graders
VALUES (1, 1),
             (3, 2),
             (2, 3),
             (3, 4),
             (1, 5),
             (1, 6),
             (6, 7),
             (6, 8),
             (6, 9),
             (7, 10),
             (7, 11),
             (7, 12),
             (4, 13),
             (4, 14),
             (6, 15),
             (6, 16),
             (5, 17),
             (5, 18);
```

Assigned Tests Table

Code:

```
INSERT INTO hrdb.[Assigned Tests]
VALUES (5, 1),
             (7, 2),
             (8, 3),
             (9, 4),
             (3, 5),
             (4, 6),
             (2, 7),
             (13, 8),
             (10, 9),
             (11, 10),
             (12, 11);
```

Expense Receipts Table

Code:

```
INSERT INTO hrdb.[Expense Recipts]
VALUES (7, 12.34, 'Ramen'),
         (7, 24.24, 'Gas'),
         (7, 33.21, 'Louises BBQ'),
         (7, 56.67, 'Halls Steakhouse'),
         (7, 6.27, 'Parking'),
         (8, 21.16, 'Chicken Riggies'),
         (8, 5.24, 'Parking'),
         (8, 12.27, 'Waffle House'),
         (8, 35.21, 'Delmonicos');
```

Reimbursements Table

Code:

```
INSERT INTO hrdb.Reimbursement
VALUES   (29, 1,'Ramen', 1, 12.34),
              (29, 2, 'Gas', 0, 24.24),
           (29, 3, 'Louises BBQ', 1, 33.21),
           (29, 4, 'Halls Steakhouse', 1, 56.67),
           (29, 5, 'Parking', 1, 6.27),
           (30, 6, 'Chicken Riggies', 0, 21.16),
           (30, 7, 'Parking', 0, 5.24),
           (30, 8, 'Waffle House', 1, 12.27),
           (30, 9, 'Delmonicos', 1, 35.21);
```

Offers Table

Code:

```
INSERT INTO hrdb.Offers
VALUES (3, 2, '2023-05-08', 'Negotiating'),
            (11, 5, '2023-02-24', 'Accepted'),
            (4, 9, '2022-12-17', 'Offer Extended'),
            (4, 10, '2022-12-18', 'Offer Extended'),
            (5, 13, '2023-01-07', 'Negotiating'),
            (9, 14, '2023-10-01', 'Declined'),
            (9, 16, '2023-10-01', 'Accepted');
```

Onboarding Table

Code:

```
INSERT INTO hrdb.Onboarding
VALUES (5, 2, '2023-03-01', '2023-03-08'),
          (16, 7, '2023-10-08', '2023-10-15');
```

Background Checks Table

Code:

```
INSERT INTO hrdb.[Background Checks]
VALUES  (5, 2, 'passed', '2023-03-01'),
           (16, 7, 'passed', '2023-10-08');
```

Documents Table

Code:

```
INSERT INTO hrdb.Documents
VALUES  (1, NULL, 'Dennises Cover Letter', 'Dennises Resume', 'Dennises Reference
Letters', 1),
              (2, 'Jennifers CV',  'Jennifers Cover Letter', 'Jennifers Resume',
'Jennifers Reference Letters', 1),
              (3, NULL, 'Stephens Cover Letter','Stephens Resume', 'Stephens Reference
Letters', 1),
              (4, NULL, 'Richards Cover Letter','Richards Resume', 'Richards Reference
Letters', 1),
              (5, NULL, 'Amandas Cover Letter','Amandas Resume', 'Amandas Reference
Letters', 1),
              (6, NULL, 'Gabrielles Cover Letter','Gabrielles Resume', 'Gabrielles
Reference Letters', 1),
              (7, 'Caydes CV', 'Caydes Cover Letter', 'Caydes Resume', 'Caydes Reference
Letters', 1),
              (8,'Joes CV', 'Joes Cover Letter', 'Joes Resume', 'Joes Reference Letters',
1),
              (9, NULL, 'Andrews Cover Letter', 'Andrews Resume', 'Andrews Reference
Letters', 1),
              (10,'Joanns CV', 'Joanns Cover Letter', 'Joanns Resume', 'Joanns Reference
Letters', 1),
              (11, 'Jasons CV', 'Jasons Cover Letter', 'Jasons Resume', 'Jasons Reference
Letters', 0),
              (12, NULL, 'Jacks Cover Letter', 'Jacks Resume', 'Jacks Reference Letters',
1),
```

```
              (13, NULL, 'Timothys Cover Letter',  'Timothys Resume', 'Timothys Reference
Letters', 1),
              (14, 'Jessicas CV', 'Jessicas Cover Letter', 'Jessicas Resume', 'Jessicas
Reference Letters', 1),
              (15, NULL, 'Michaels Cover Letter', 'Michaels Resume', 'Michaels Reference
Letters', 1),
              (16, 'Jessica Matas CV', 'Jessica Matas Cover Letter', 'Jessica Matas
Resume', 'Jessica Matas Reference Letters', 1),
              (17, NULL, 'Larrys Cover Letter',  'Larrys Resume', 'Larrys Reference
Letters', 1),
              (18, 'Cheryl Flores CV', 'Cheryl Cover Letter', 'Cheryl Flores Resume',
'Cheryl Flores Reference Letters', 1),
              (19, 'Lorraines CV', 'Lorraines  Cover Letter', 'Lorraines Resume',
'Lorraines Reference Letters', 1),
              (20, 'Wendys CV', 'Wendys Cover Letter', 'Wendys Resume', 'Wendys Reference
Letters', 1),
              (21, NULL, 'Mistys  Cover Letter', 'Mistys Resume', 'Mistys  Reference
Letters', 1);
```

Candidate Complaints Table

Code:

```
INSERT INTO hrdb.[Candidate Complaints]
VALUES  (11, 'I got rejected for invalid documents when they are all valid!', '2022-11-
12'),
              (21, 'I was rejected and I deserve another interview', '2022-12-18'),
              (12, 'My interviewer was late and it cost me time', '2022-09-04'),
              (9, 'I got a low score on a question and I know its correct', '2022-10-
14'),
              (2, 'The interview call was static and kept dropping out', '2022-10-01');
```

Section E: Conclusion

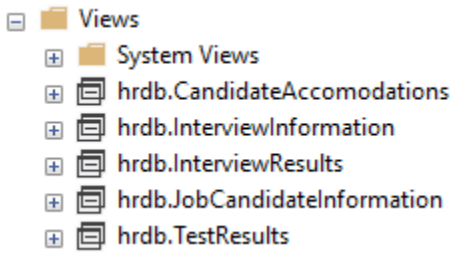
Final Analysis & Remarks

To begin, what I liked about this project. I really enjoyed the design aspect of this project, creating the E/R diagram and using it to create a fully functional database given a set of parameters encompasses everything I wanted to learn from this class. This project did a great job outlining all the things learned throughout the semester. It also reinforced recently learned topics like views, stored procedures, user defined functions, triggers and transactions. The business reports also provided for the use of more complex SQL structures that interrupted the database I created. My main critique of this project is the size, the minimum of twenty tables meant a majority of my time was spent on manually inserting hundreds of lines of data, which from a learning prospective taught me next to nothing. I believe the objectives of this project could be reached with five to ten tables. The second criticism has to do with the topic of this database. I know it would make grading difficult, but having the database be about something I'm passionate about would make the project more enticing and personal. I'd much rather talk to an employer about a database that tells them something about me, not just that I like human resource departments.

Section F: Appendix

Views Screenshots

Object Explorer Showing Views



Job Candidate Information View

Project 2 Source C...80KU7U\unuse (64))

```
--      (21, 'I was rejected and I deserve another interview', '2022-12-18'),
--      (12, 'My interviewer was late and it cost me time', '2022-09-04'),
--      (9, 'I got a low score on a question and I know its correct', '2022-10-14'),
--      (2, 'The interview call was static and kept dropping out', '2022-10-01');
-------------------------------------------------------------------------------



--------------------------------------VIEWS-------------------------------------
--||VIEW that returns application information, contact information and the status for each candidate||--
--CREATE VIEW hrdb.JobCandidateInformation AS
--SELECT CandidateName AS [Candidate Applying], Email AS [Candidate Email], ApplicationID AS [Application Number], CandidateStatus
--FROM hrdb.Candidates JOIN hrdb.[Application]
--  ON Candidates.CandidateID = [Application].CandidateNumber
--WHERE CandidateStatus <> 'blacklisted'

SELECT * FROM hrdb.JobCandidateInformation


--||VIEW that returns important information for each interview||--
```

120 %

Results | Messages

	Candidate Applying	Candidate Email	Application Number	Candidate Status
1	Dennis Hanna	dhanna@gmail.com	23	waiting
2	Jennifer Garner	jgarner@gmail.com	24	waiting
3	Richard Long	rlong@gmail.com	25	rejected
4	Stephen Delacruz	sdelacruz@gmail.com	26	on-call for next job opportunity
5	Amanda Walter	awalter@gmail.com	27	waiting
6	Gabrielle Barber	gbarber@gmail.com	28	waiting
7	Cayde Six	hunterVanguard@bungie.net	29	waiting
8	Joe Garcia	jgarcia@gmail.com	30	waiting
9	Andrew Gonzalez	agonzalez@gmail.com	31	waiting
10	Joann Kelly	jkelly@gmail.com	32	waiting
11	Jason Montgomery	jmontgomery@gmail.com	33	rejected
12	Timothy Zimmerman	tzimmerman@gmail.com	35	waiting
13	Jessica Davis	jdavis@gmail.com	36	waiting
14	Michael Mcclure	mmcclure@gmail.com	37	waiting
15	Jessica Mata	jmata@gmail.com	38	waiting
16	Larry Reyes	lreyes@gmail.com	39	waiting
17	Cheryl Flores	cflores@gmail.com	40	waiting
18	Lorraine Ray	lray@gmail.com	41	waiting
19	Wendy Allen	wallen@gmail.com	42	waiting

Query executed successfully. | DESKTOP-880KU7U (15.0 RTM) | DESKTOP-880KU7U\unuse ... | HumanResources | 00:00:00 | 20 rows

Interview Information View

Project 2 Source C...80KU7U\unuse (64))

```
--FROM hrdb.Candidates JOIN hrdb.[Application]
--  ON Candidates.CandidateID = [Application].CandidateNumber
--WHERE CandidateStatus <> 'blacklisted'

--SELECT * FROM hrdb.JobCandidateInformation


--||VIEW that returns important information for each interview||--
--CREATE VIEW hrdb.InterviewInformation AS
--SELECT InterviewerName AS Interviewer, CandidateName AS Candidate, InterviewID AS [Interview Number], InterviewType AS [Type of I
--FROM hrdb.Interviews
--  JOIN hrdb.[Assigned Interviews] ON Interviews.InterviewID = [Assigned Interviews].InterviewNumber
--  JOIN hrdb.Interviewers ON [Assigned Interviews].InterviewerNumber = Interviewers.InterviewerID
--  JOIN hrdb.[Application] ON Interviews.ApplicationNumber = [Application].ApplicationID
--  JOIN hrdb.Candidates ON [Application].CandidateNumber = Candidates.CandidateID

SELECT * FROM hrdb.InterviewInformation


--||VIEW that returns all the accomodations for each candidate||--
```

120 %

Results | Messages

	Interviewer	Candidate	Interview Number	Type of Interview	Interview Time	Candidate Number	Application Number
1	Brian Jefferson	Jennifer Garner	1	Online	09:45:00.00000	2	24
2	Christina Randall	Amanda Walter	2	Online	09:00:00.00000	5	27
3	Christina Randall	Gabrielle Barber	3	Online	09:00:00.00000	6	28
4	Christina Randall	Cayde Six	4	Onsite	08:00:00.00000	7	29
5	Julie Johnson	Joe Garcia	5	Onsite	08:00:00.00000	8	30
6	Victoria Brown	Andrew Gonzalez	6	Online	09:30:00.00000	9	31
7	Victoria Brown	Joann Kelly	7	Online	09:30:00.00000	10	32
8	Betty Callahan	Timothy Zimmerman	8	Online	09:30:00.00000	13	35
9	Robert Parker	Jessica Davis	9	Online	10:00:00.00000	14	36
10	Robert Parker	Michael Mcclure	10	Online	10:00:00.00000	15	37
11	Robert Parker	Jessica Mata	11	Online	10:00:00.00000	16	38

Candidate Accommodations View

Project 2 Source C...80KU7U\unuse (64))

```
--  JOIN hrdb.Interviewers ON [Assigned Interviews].InterviewerNumber = Interviewers.InterviewerID
--  JOIN hrdb.[Application] ON Interviews.ApplicationNumber = [Application].ApplicationID
--  JOIN hrdb.Candidates ON [Application].CandidateNumber = Candidates.CandidateID

--SELECT * FROM hrdb.InterviewInformation


--||VIEW that returns all the accomodations for each candidate||--
--CREATE VIEW hrdb.CandidateAccomodations AS
--SELECT CandidateName AS Candidate, HotelName AS Hotel, Airline, RentalCarCompanyName AS [Rental Car Provider]
--FROM hrdb.Interviews
--  JOIN hrdb.Accomodations ON Interviews.InterviewID = Accomodations.InterviewNumber
--  JOIN hrdb.Hotels ON Accomodations.HotelDetailsNumber = Hotels.HotelReservationID
--  JOIN hrdb.Airfair ON Accomodations.AirfairDetailsNumber = Airfair.AirfairID
--  JOIN hrdb.[Rental Cars] ON Accomodations.RentalCarNumber = [Rental Cars].RentalCarID
--  JOIN hrdb.[Application] ON Interviews.ApplicationNumber = [Application].ApplicationID
--  JOIN hrdb.Candidates ON [Application].CandidateNumber = Candidates.CandidateID

SELECT * FROM hrdb.CandidateAccomodations
```

120 %

Results | Messages

	Candidate	Hotel	Airline	Rental Car Provider
1	Cayde Six	Tower Suites	Vanguard Airlines	Hertz
2	Joe Garcia	Marriot	Southwest	AVIS

Query executed successfully. | DESKTOP-880KU7U (15.0 RTM) | DESKTOP-880KU7U\unuse ... | HumanResources | 00:00:00 | 2 rows

Test Results View

Project 2 Source C...80KU7U\unuse (64))

```
--  JOIN hrdb.Hotels ON Accomodations.HotelDetailsNumber = Hotels.HotelReservationID
--  JOIN hrdb.Airfair ON Accomodations.AirfairDetailsNumber = Airfair.AirfairID
--  JOIN hrdb.[Rental Cars] ON Accomodations.RentalCarNumber = [Rental Cars].RentalCarID
--  JOIN hrdb.[Application] ON Interviews.ApplicationNumber = [Application].ApplicationID
--  JOIN hrdb.Candidates ON [Application].CandidateNumber = Candidates.CandidateID

--SELECT * FROM hrdb.CandidateAccomodations


--||VIEW that returns the test results and interview number per candidate||--
--CREATE VIEW hrdb.TestResults AS
--SELECT CandidateName AS Candidate, InterviewID AS [Interview Number], Grade AS [Test Result]
--FROM hrdb.Interviews
--  JOIN hrdb.[Assigned Tests] ON Interviews.InterviewID = [Assigned Tests].InterviewNumber
--  JOIN hrdb.Tests ON [Assigned Tests].TestNumber = Tests.TestID
--  JOIN hrdb.[Application] ON Interviews.ApplicationNumber = [Application].ApplicationID
--  JOIN hrdb.Candidates ON [Application].CandidateNumber = Candidates.CandidateID

SELECT * FROM hrdb.TestResults
--------------------------------------------------------------------------------
```

120 %

Results | Messages

	Candidate	Interview Number	Test Result
1	Jennifer Garner	1	passed
2	Amanda Walter	2	passed
3	Gabrielle Barber	3	passed
4	Cayde Six	4	passed
5	Joe Garcia	5	passed
6	Andrew Gonzalez	6	passed
7	Joann Kelly	7	failed
8	Timothy Zimmerman	8	passed
9	Jessica Davis	9	passed
10	Michael Mcclure	10	passed
11	Jessica Mata	11	passed

Stored Procedures Screenshots

Object Explorer Showing Views

- Stored Procedures
  - System Stored Procedures
  - dbo.spAddCandidate
  - dbo.spAverageCostOfOnsiteInterviews
  - dbo.spAverageRecruitmentTime
  - dbo.spNumberOfApplicationsInTimePeriod
  - dbo.spOfferDecision
  - dbo.spOfferRate
  - dbo.spRejectCandidate
  - dbo.spSelectCandidateForInterview
  - hrdb.spAddCandidate

Add Candidate Stored Procedure

Project 2 Source C...80KU7U\unuse (64))

```
---------------------------------STORED PROCEDURES---------------------------------
--||STORED PROCEDURE that lets a candidate be added to the database||--
--IF OBJECT_ID('spAddCandidate') IS NOT NULL
--  DROP PROC spAddCandidate;
--GO

--CREATE PROC spAddCandidate
--  @Name VARCHAR(50),
--  @Email VARCHAR(50),
--  @Phone CHAR(12),
--  @Profile VARCHAR(500)
--AS
--  BEGIN
--      INSERT INTO hrdb.Candidates
--      VALUES (@Name, @Email, @Phone, @Profile, 'waiting')
--  END

EXEC spAddCandidate 'Joe Doe', 'jdoe@gmail.com', '315-543-3321', 'John Doe Profile';
```

120 %

Messages

```
(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)
```

120 %

Query executed successfully. | DESKTOP-880KU7U (15.0 RTM) | DESKTOP-880KU7U\unuse ... | HumanResources | 00:00:00 | 0 rows

Offer Decision Stored Procedure

Project 2 Source C...80KU7U\unuse (64))

```
--||STORED PROCEDURE that allows a candidate to accept or decline a job offer, if declined candidate status is changed to on-call||--
--IF OBJECT_ID('spOfferDecision') IS NOT NULL
--  DROP PROC spOfferDecision;
--GO

--CREATE PROC spOfferDecision
--  @CandidateID INT,
--  @Decision BIT
--AS
--  IF @Decision = 1
--      UPDATE hrdb.Offers
--      SET OfferStatus = 'Accepted'
--      WHERE CandidateNumber = @CandidateID;
--  IF @Decision = 0
--      UPDATE hrdb.Offers
--      SET OfferStatus = 'Declined'
--      WHERE CandidateNumber = @CandidateID;
--      UPDATE hrdb.Candidates
--      SET CandidateStatus = 'on-call for next job opportunity'
--      WHERE CandidateID = @CandidateID;

EXEC spOfferDecision 2, 1;
```

120 %

Messages

```
(1 row affected)

(1 row affected)
Offer declined. Number of openings increased accordingly.
```

Select Candidate For Interview Stored Procedure

DESKTOP-880KU7U.H...- hrdb.Interviews | DESKTOP-880KU7U.H...hrdb.Application | DESKTOP-880KU7U....- hrdb.Candidates | Project 2 Source C...80KU7U\unuse (64))

```
--AS
--  UPDATE hrdb.Candidates
--  SET CandidateStatus = 'rejected'
--  WHERE CandidateID = @CandidatesNumber AND CandidateName = @CandidatesName;

--EXEC spRejectCandidate 'Amanda Walter', 5;


--||STORED PROCEDURE that selects a provided candidates application for an interview||--
--IF OBJECT_ID('spSelectCandidateForInterview') IS NOT NULL
--  DROP PROC spSelectCandidateForInterview;
--GO

--CREATE PROC spSelectCandidateForInterview
--  @ApplicationNumber INT,
--  @Type CHAR(6),
--  @Round INT,
--  @Time TIME(5)
--AS
--  INSERT INTO hrdb.Interviews
--  VALUES (NULL, NULL, @ApplicationNumber, @Type, @Round, @Time, NULL);

EXEC spSelectCandidateForInterview 30, 'Online', 5, '10:35:00';
--------------------------------------------------------------------------------
```

120 %

Messages

```
(1 row affected)

Completion time: 2022-12-08T17:51:24.8426353-05:00
```

120 %

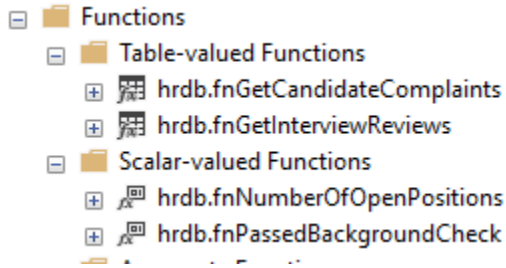
Query executed successfully. | DESKTOP-880KU7U (15.0 RTM) | DESKTOP-880KU7U\unuse ... | HumanResources | 00:00:00 | 0 rows

DESKTOP-880KU7U.H...- hrdb.Interviews | DESKTOP-880KU7U.H...hrdb.Application | DESKTOP-880KU7U....- hrdb.Candidates | Project 2 Source C...80KU7U\unuse (64))

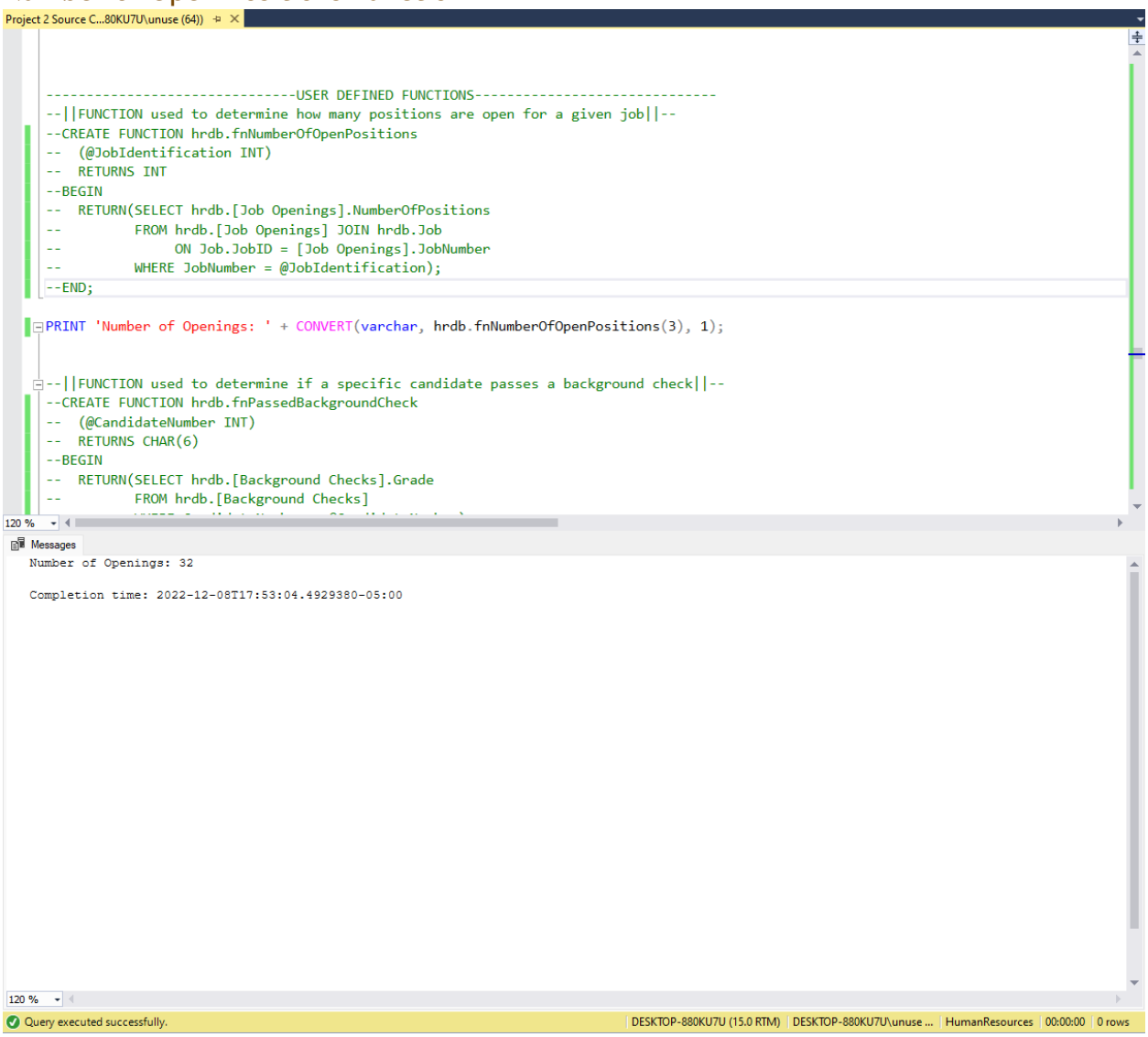
InterviewID	CandidateRevi...	InterviewerRev...	ApplicationNu...	InterviewType	InterviewRound	InterviewStartT...	InterviewEndTi...
1	1	1	24	Online	1	09:45:00	11:00:00
2	2	2	27	Online	2	09:00:00	10:30:00
3	3	3	28	Online	1	09:00:00	10:30:00
4	4	4	29	Onsite	1	08:00:00	12:45:00
5	5	5	30	Onsite	2	08:00:00	12:45:00
6	6	6	31	Online	1	09:30:00	11:45:00
7	7	7	32	Online	1	09:30:00	11:50:00
8	8	8	35	Online	1	09:30:00	11:45:00
9	9	9	36	Online	2	10:00:00	11:00:00
10	10	10	37	Online	1	10:00:00	11:00:00
11	11	11	38	Online	1	10:00:00	11:00:00
1002	NULL	NULL	40	Online	2	10:30:00	NULL
1003	NULL	NULL	30	Online	5	10:35:00	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

User Defined Functions Screenshots

Object Explorer Showing Functions



Number of Open Positions Function



Passed Background Check Function

Project 2 Source C...80KU7U\unuse (64))

```
--  (@JobIdentification INT)
--  RETURNS INT
--BEGIN
--  RETURN(SELECT hrdb.[Job Openings].NumberOfPositions
--         FROM hrdb.[Job Openings] JOIN hrdb.Job
--              ON Job.JobID = [Job Openings].JobNumber
--         WHERE JobNumber = @JobIdentification);
--END;

--PRINT 'Number of Openings: ' + CONVERT(varchar, hrdb.fnNumberOfOpenPositions(3), 1);


--||FUNCTION used to determine if a specific candidate passes a background check||--
--CREATE FUNCTION hrdb.fnPassedBackgroundCheck
--  (@CandidateNumber INT)
--  RETURNS CHAR(6)
--BEGIN
--  RETURN(SELECT hrdb.[Background Checks].Grade
--         FROM hrdb.[Background Checks]
--         WHERE CandidateNumber = @CandidateNumber);
--END;

PRINT 'Grade on Background Check: ' + CONVERT(varchar, hrdb.fnPassedBackgroundCheck(5), 1);
```

120 %

Messages

```
Grade on Background Check: passed

Completion time: 2022-12-08T17:53:42.0786533-05:00
```

120 %

Query executed successfully. | DESKTOP-880KU7U (15.0 RTM) | DESKTOP-880KU7U\unuse ... | HumanResources | 00:00:00 | 0 rows

Get Candidate Complaint Function

Project 2 Source C...80KU7U\unuse (64))

```
--          FROM hrdb.[Background Checks]
--          WHERE CandidateNumber = @CandidateNumber);
--END;

--PRINT 'Grade on Background Check: ' + CONVERT(varchar, hrdb.fnPassedBackgroundCheck(5), 1);


--||FUNCTION that returns the table entry for a candidates complaint||--
--CREATE FUNCTION hrdb.fnGetCandidateComplaints
--  (@CandidateNumber INT)
--  RETURNS TABLE
--RETURN (SELECT ComplaintDescription AS [Complaint], ComplaintDate AS [Date Filed]
--        FROM hrdb.[Candidate Complaints]
          --WHERE CandidateNumber = @CandidateNumber);

SELECT * FROM hrdb.fnGetCandidateComplaints(12)


--||FUNCTION that returns the reviews from both the candidate and the interviewer for a particular interview||--
--CREATE FUNCTION hrdb.fnGetInterviewReviews
--  (@InterviewNumber INT)
--  RETURNS TABLE
--RETURN (SELECT [Candidate Reviews].Details AS [Candidate Feedback], [Interviewer Reviews].Details AS [Interviewer Feedback]
--        FROM hrdb.[Candidate Reviews]
--          JOIN hrdb.Interviews ON Interviews.CandidateReview = [Candidate Reviews].CandidateReviewID
```

120 %

Results | Messages

	Complaint	Date Filed
1	My interviewer was late and it cost me time	2022-09-04

Query executed successfully. | DESKTOP-880KU7U (15.0 RTM) | DESKTOP-880KU7U\unuse ... | HumanResources | 00:00:00 | 1 rows

Get Interview Reviews Function

Project 2 Source C...80KU7U\unuse (64))

```
--SELECT * FROM hrdb.fnGetCandidateComplaints(12)


--||FUNCTION that returns the reviews from both the candidate and the interviewer for a particular interview||--
--CREATE FUNCTION hrdb.fnGetInterviewReviews
--  (@InterviewNumber INT)
--  RETURNS TABLE
--RETURN (SELECT [Candidate Reviews].Details AS [Candidate Feedback], [Interviewer Reviews].Details AS [Interviewer Feedback]
--        FROM hrdb.[Candidate Reviews]
--          JOIN hrdb.Interviews ON Interviews.CandidateReview = [Candidate Reviews].CandidateReviewID
--          JOIN hrdb.[Interviewer Reviews] ON Interviews.InterviewerReview = [Interviewer Reviews].InterviewerReviewID
--        WHERE Interviews.InterviewID = @InterviewNumber);

SELECT * FROM hrdb.fnGetInterviewReviews(1)
--------------------------------------------------------------------------------



------------------------------TRIGGERS & TRANSACTIONS----------------------------
--||TRIGGER that verifys a candidate has submitted the correct documents and passes a background check before being added to onboardin
--CREATE TRIGGER verifyCandidate
--  ON hrdb.Offers
--  AFTER INSERT
--AS
--  IF (SELECT Grade FROM hrdb.[Background Checks] WHERE CandidateNumber = (SELECT CandidateNumber FROM Inserted)) = 'failed'
```

120 %

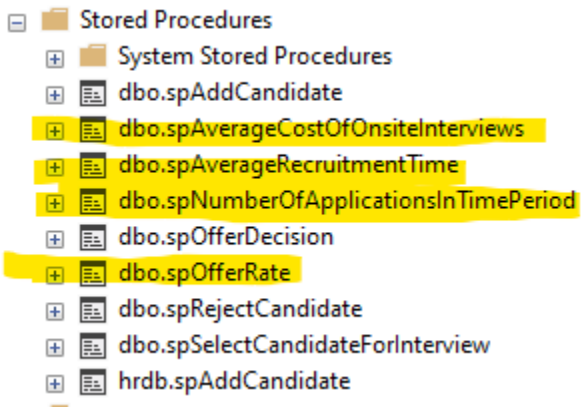
Results | Messages

	Candidate Feedback	Interviewer Feedback
1	Interview went well, I really thought it was f...	Interviewee did well, knows there fundamentals, ...

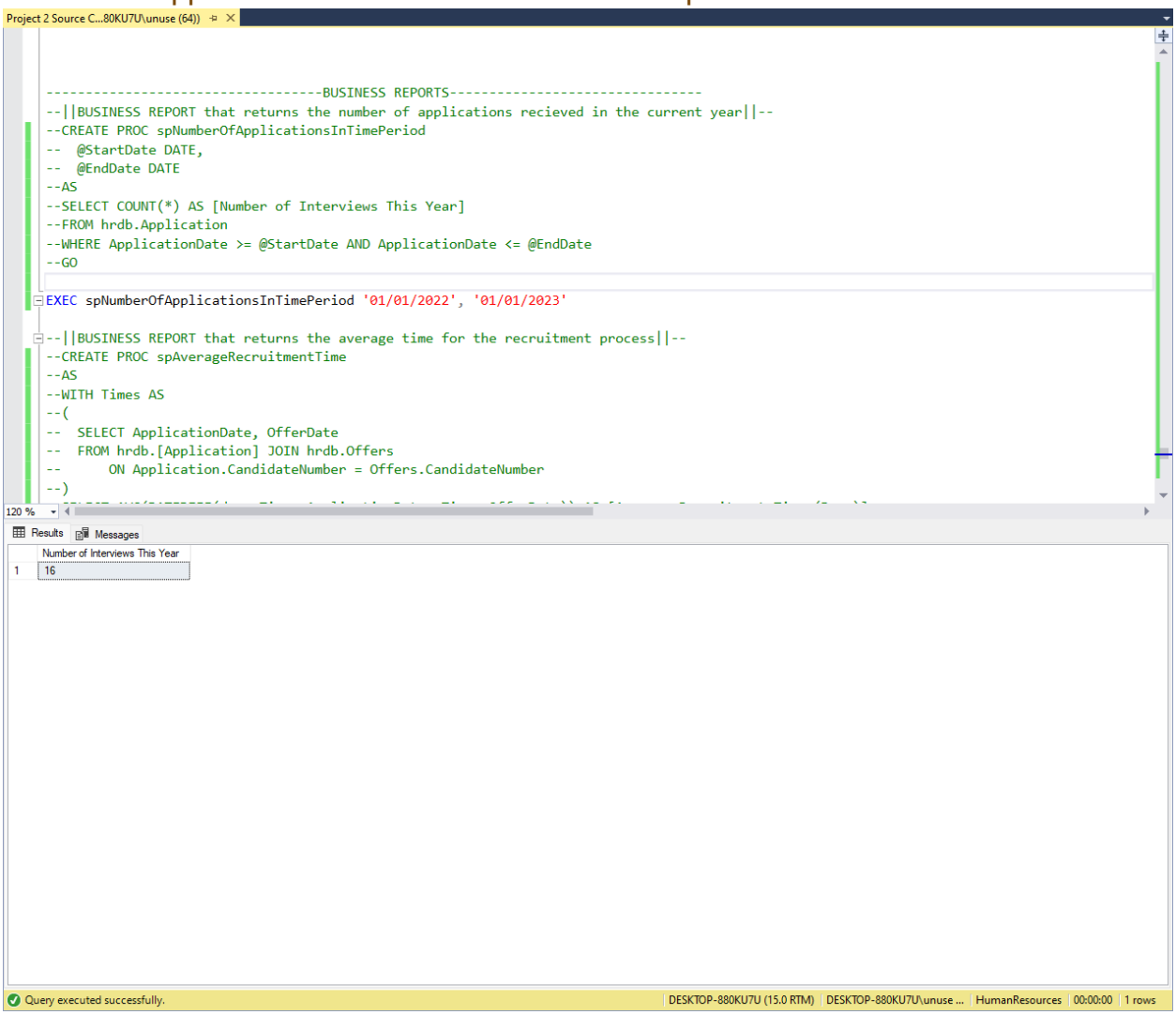
Query executed successfully. | DESKTOP-880KU7U (15.0 RTM) | DESKTOP-880KU7U\unuse ... | HumanResources | 00:00:00 | 1 rows

Business Reports Screenshots

Object Explorer Business Reports



Number of Applications in Time Period Business Report



Average Recruitment Time Business Report

Project 2 Source C...80KU7U\unuse (64))

```
--SELECT COUNT(*) AS [Number of Interviews This Year]
--FROM hrdb.Application
--WHERE ApplicationDate >= @StartDate AND ApplicationDate <= @EndDate
--GO

--EXEC spNumberOfApplicationsInTimePeriod '01/01/2022', '01/01/2023'


--||BUSINESS REPORT that returns the average time for the recruitment process||--
--CREATE PROC spAverageRecruitmentTime
--AS
--WITH Times AS
--(
--  SELECT ApplicationDate, OfferDate
--  FROM hrdb.[Application] JOIN hrdb.Offers
--      ON Application.CandidateNumber = Offers.CandidateNumber
--)
--SELECT AVG(DATEDIFF(day, Times.ApplicationDate, Times.OfferDate)) AS [Average Recruitment Time (Days)]
--FROM Times
--GO

EXEC spAverageRecruitmentTime


--||BUSINESS REPORT that returns the percent of candidates hired that applied||--
```

120 %

Results | Messages

	Average Recruitment Time (Days)
1	150

Query executed successfully. | DESKTOP-880KU7U (15.0 RTM) | DESKTOP-880KU7U\unuse ... | HumanResources | 00:00:00 | 1 rows

Offer Rate Business Report

Project 2 Source C...80KU7U\unuse (64))

```
--EXEC spAverageRecruitmentTime


--||BUSINESS REPORT that returns the percent of candidates hired that applied||--
--CREATE PROC spOfferRate
--AS
--DECLARE @PercentAns DECIMAL(5, 1);
--DECLARE @NumApps DECIMAL(5, 1);
--DECLARE @NumOffs DECIMAL(5, 1);
--SET @NumApps = CONVERT(DECIMAL(5, 1), (SELECT COUNT(*) FROM hrdb.[Application]), 1)
--SET @NumOffs = CONVERT(DECIMAL(5, 1), (SELECT COUNT(*) FROM hrdb.Offers), 1)
--SET @PercentAns =  (@NumOffs / @NumApps) * 100
--PRINT 'Application Acceptance Rate: ' + CONVERT(varchar, @PercentAns, 1) + '%'
--GO

EXEC spOfferRate


--||BUSINESS REPORT that returns the average cost of onsite interviews||--
--CREATE PROC spAverageCostOfOnsiteInterviews
--AS
--WITH Costs AS
--(
--  SELECT TotalCost AS [Accomodations Cost], SUM(Amount) AS [Total Reimbursement Cost]
--  FROM hrdb.Accomodations
```

120 %

Messages

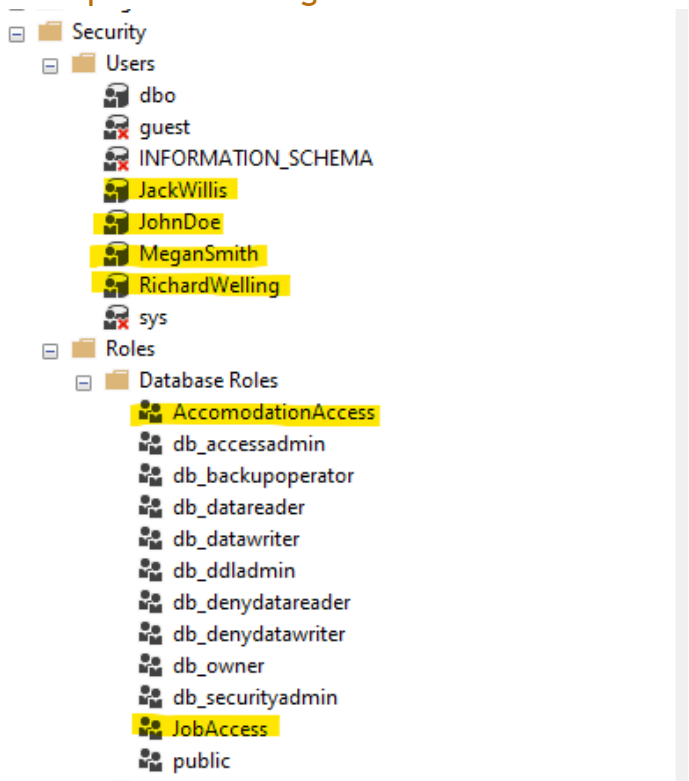
```
Application Acceptance Rate: 33.3%

Completion time: 2022-12-08T18:06:23.7283579-05:00
```

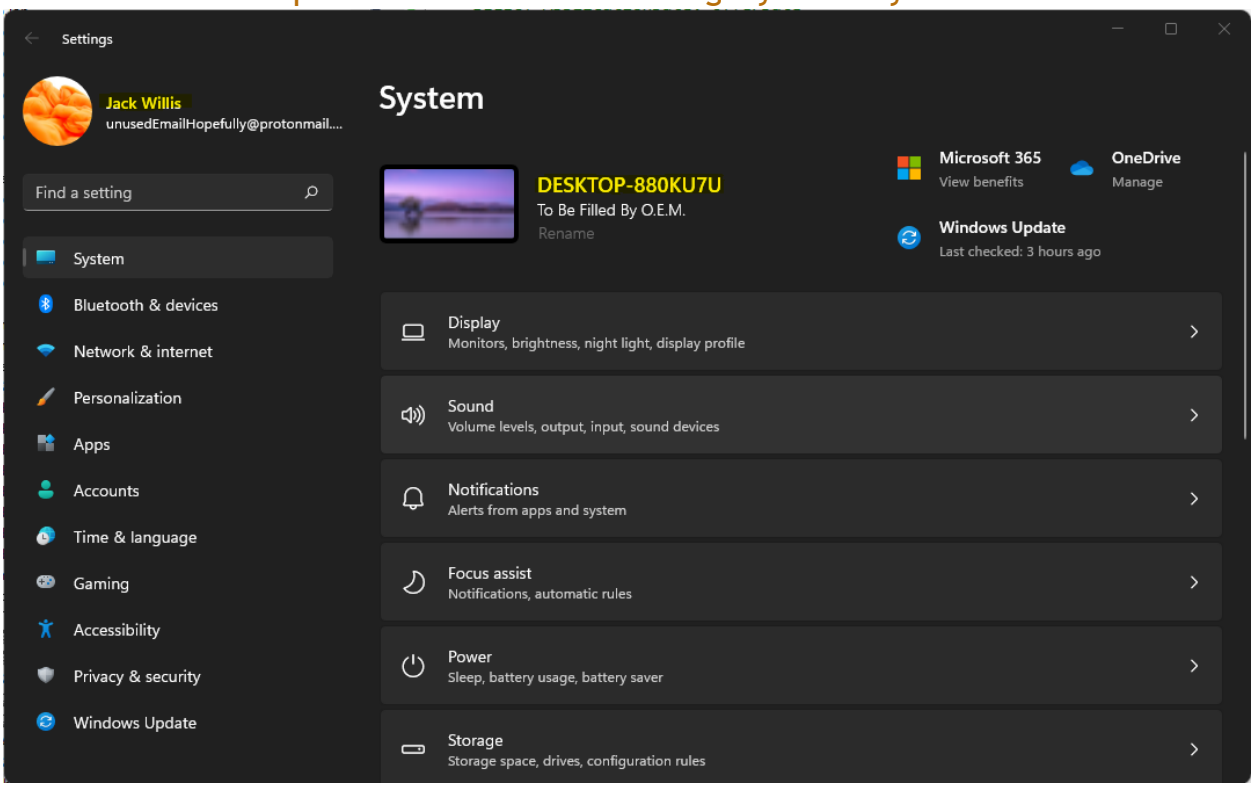
120 %

Query executed successfully. | DESKTOP-880KU7U (15.0 RTM) | DESKTOP-880KU7U\unuse ... | HumanResources | 00:00:00 | 0 rows

Screenshot of Object Explorer Showing Created Accounts and Security Roles



Screenshot of Computer Identification Proving My Identity



At the bottom right of most of my screenshots you can see the computer identification in the yellow tab. My PC has the DESKTOP-880KU7U as a name which matches this image from my computers settings.

Project 2 Source C...80KU7U\unuse (64))

```
--||BUSINESS REPORT that returns the average cost of onsite interviews||--
--CREATE PROC spAverageCostOfOnsiteInterviews
--AS
--WITH Costs AS
--(
--  SELECT TotalCost AS [Accomodations Cost], SUM(Amount) AS [Total Reimbursement Cost]
--  FROM hrdb.Accomodations
--      JOIN hrdb.Interviews ON Accomodations.InterviewNumber = Interviews.InterviewID
--      JOIN hrdb.[Application] ON Interviews.ApplicationNumber = [Application].ApplicationID
--      JOIN hrdb.Candidates ON [Application].CandidateNumber = Candidates.CandidateID
--      JOIN hrdb.Reimbursement ON [Application].ApplicationID = Reimbursement.ApplicationNumber
--  GROUP BY InterviewID, CandidateName, TotalCost
--),
--TotalCosts AS
--(
--  SELECT [Accomodations Cost] + [Total Reimbursement Cost] AS [Total Onsite Cost]
--  FROM Costs
--)
--SELECT AVG([Total Onsite Cost]) AS [Average Onsite Interview Cost]
--FROM TotalCosts
--GO

EXEC spAverageCostOfOnsiteInterviews
--------------------------------------------------------------------------------
```

120 %

Results | Messages

	Average Onsite Interview Cost
1	833.305

Query executed successfully. | DESKTOP-880KU7U (15.0 RTM) | DESKTOP-880KU7U\unuse ... | HumanResources | 00:00:00 | 1 rows