



CSE 581 INTRO TO DATABASE MANAGEMENT SYSTEMS

Project 2



DECEMBER 11, 2022

SIDDHITA NIKAM
SUID - 471330539

ABSTRACT

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

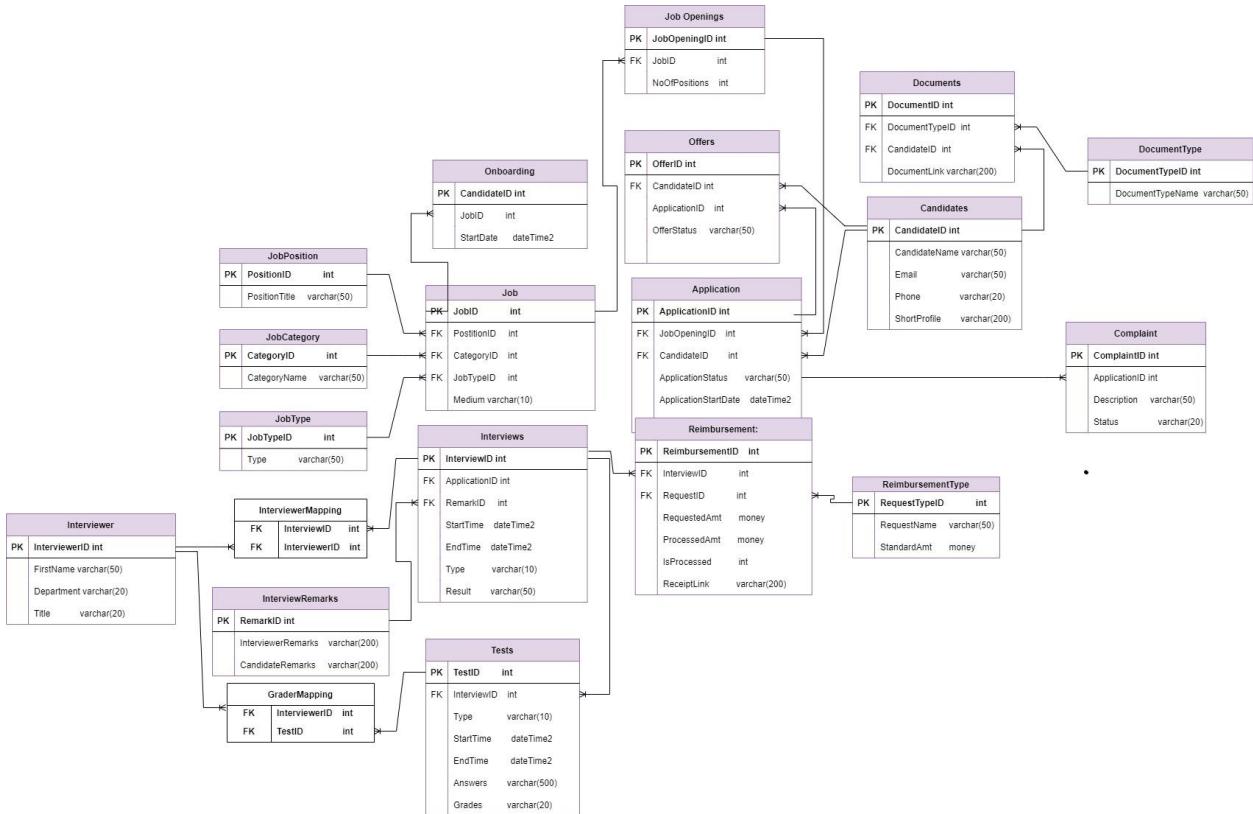
The interview process will update a candidate's status from time-to-time. The main focus is on tracking the status of candidate's application.

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

Contents

DESIGN	3
IMPLEMENTATION	3
TESTING	18
VIEWS.....	18
STORED PROCEDURES.....	21
USER DEFINED FUNCTIONS.....	25
TRANSACTIONS:	29
SCRIPTS	33
Script Roles.....	36
TRIGGERS	38

DESIGN



IMPLEMENTATION

```
USE master
```

```
GO
```

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

```
GO
```

```
CREATE DATABASE Recruitment
GO
```

```

1 USE master
2 GO
3
4
5 IF DB_ID('Recruitment') IS NOT NULL
6     DROP DATABASE Recruitment
7 GO
8
9 CREATE DATABASE Recruitment
10 GO
11
12

```

Messages

Commands completed successfully.

Completion time: 2022-12-11T13:32:06.9149437-05:00

Query executed successfully.

Create Table Statements:

```
USE Recruitment
GO
```

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

```
CREATE TABLE JobCategory(
    CategoryID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    CategoryName varchar(50) NOT NULL)
GO
```

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

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

```

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

CREATE TABLE JobOpenings(
    JobOpeningID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    JobID int NOT NULL REFERENCES Job (JobID),
    NoOfPositions int NOT NULL)
GO

CREATE TABLE Candidates(
    CandidateID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    CandidateName varchar(50) NOT NULL,
    Email varchar(50) NOT NULL,
    Phone varchar(20) NOT NULL,
    ShortProfile varchar(200) NULL
)
GO

CREATE TABLE Applications(
    ApplicationID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    JobOpeningID int NOT NULL REFERENCES JobOpenings (JobOpeningID),
    CandidateID int NOT NULL REFERENCES Candidates (CandidateID),
    ApplicationStatus varchar(50) NOT NULL,
    ApplicationStartDate dateTime2 NOT NULL,
    LastUpdateDate dateTime2 NOT NULL)
GO

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

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

```

```
GO
```

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

```
GO
```

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

```
GO
```

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

```
GO
```

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

```
GO
```

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

```

        TestID int NOT NULL REFERENCES Tests(TestID)
    )
GO

CREATE TABLE ReimbursementType(
    RequestTypeID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    RequestName varchar(50) NOT NULL,
    StandardAmt money NOT NULL
)
GO

CREATE TABLE Reimbursement(
    ReimbursementID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    InterviewID int NOT NULL REFERENCES Interviews(InterviewID),
    RequestTypeID int NOT NULL REFERENCES
ReimbursementType(RequestTypeID),
    RequestedAmt money DEFAULT 0,
    ProcessedAmt money DEFAULT 0,
    ReceiptLink varchar(200) NOT NULL,
    IsProcessed varchar(200) NOT NULL
)
GO

CREATE TABLE DocumentType(
    DocumentTypeID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    DocumentTypeName varchar(50) NOT NULL
)
GO

CREATE TABLE Documents(
    DocumentID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    CandidateID int NOT NULL REFERENCES Candidates (CandidateID),
    DocumentTypeID int NOT NULL REFERENCES DocumentType
(DocumentTypeID),
    DocumentLink varchar(200) NOT NULL
)
GO

CREATE TABLE Onboarding(
    CandidateID int NOT NULL REFERENCES Candidates(CandidateID)
PRIMARY KEY,
    JobID int NOT NULL REFERENCES Job (JobID),
    StartDate dateTime2 NOT NULL
)
GO

CREATE TABLE Complaint(

```

```

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

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure. The central pane contains the following T-SQL code:

```

USE Recruitment
GO

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

CREATE TABLE JobCategory(
    CategoryID int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    CategoryName varchar(50) NOT NULL
)
GO

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

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

```

The status bar at the bottom indicates "Query executed successfully." and "Completion time: 2022-12-11T13:33:08.4320094-05:00".

Insert Statements:

```

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

```

```

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

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

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

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

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

```

```

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

```

INSERT INTO Applications VALUES

```

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

```

INSERT INTO Offers VALUES

```

(1, 5, 'actionNeeded'),
(5, 11, 'accepted');

```

INSERT INTO InterviewRemarks VALUES

```

('Candidate seems to be good based on the 1st interview. Can proceed
with 2nd round','Cooperative Interviewer'),
('Candidate can proceed with round 2. Candidate could answer the
questions satisfactorily in the 1st round.',NULL),
('Candidate couldn't answer the 50% questions in the 2nd technical
interview. Doesn't fit for this position','Tough Interview'),
('Candidate cleared the 1st technical round.',NULL),
('Candidate cleared the 2nd technical round. Can proceed with 3rd
round of interview for behavioural questions',NULL),
('Candidate cleared the 3rd round of interview',NULL),

```

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

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

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

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

```

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

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

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

```

```

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

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

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

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

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

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

```

SQLQuery1.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\sidhh (51)) - Microsoft SQL Server Management Studio

```

172
173
174 INSERT INTO JobPosition VALUES
175     ('IT Manager'),
176     ('Software Developer'),
177     ('Project Manager'),
178     ('Team Lead'),
179     ('Senior Software Developer'),
180     ('Associate Developer'),
181     ('System Engineer'),
182     ('Business Analyst'),
183     ('Senior Business Analyst'),
184
107 % 4

```

Messages

```

(13 rows affected)
(7 rows affected)
(4 rows affected)
(10 rows affected)
(7 rows affected)
(10 rows affected)
(11 rows affected)
(2 rows affected)
(11 rows affected)
(11 rows affected)
(5 rows affected)
(16 rows affected)
(9 rows affected)
(9 rows affected)
(9 rows affected)
107 %

```

Query executed successfully.

Select Statements

SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\sidhh (67)) - Microsoft SQL Server Management Studio

```

1 1>SELECT * FROM JobCategory;
2 2>SELECT * FROM JobPosition;
3 3>SELECT * FROM JobType;

```

Results

CategoryID	CategoryName
1	Information security
2	Software development
3	Analyst
4	Management
5	Database Administrator
6	Cyber Security

PositionID	PositionTitle
1	IT Manager
2	Software Developer
3	Project Manager
4	Team Lead
5	Senior Software Developer
6	Associate Developer
7	System Engineer
8	Business Analyst
9	Senior Business Analyst
10	Database Engineer
11	Database Architect
12	QA Analyst
13	Cybersecurity analyst

JobTypeID	Type
1	Summer Internship
2	Full-time Job
3	Part-time job
4	Contact-based

Query executed successfully.

SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\sidhh (67)) - Microsoft SQL Server Management Studio

```

File Edit View Query Project Tools Window Help
New Query Execute
Object Explorer Connect
Recruitment
SQLQuery2.sql - LA...LHCP92\sidhh (67)* SQLQuery1.sql - LA...LHCP92\sidhh (51)*
1: SELECT * FROM Candidates;
2: SELECT * FROM Job;
3: SELECT * FROM JobOpenings;
4:

```

107 %

Results Messages

CandidateID	CandidateName	Email	Phone	ShortProfile
1	Siddhi Nikam	siddhi123@gmail.com	3154602586	Java Developer 5 Years Experience
2	Jethala Gada	jgada12@gmail.com	315796534	Business Analyst 8 Years Experience
3	Aasthami Bhide	aasthamshide4@gmail.com	311206200	Project manager 15 Years Experience
4	Himanshu Patel	himanshu12345@gmail.com	315454530	Software developer 10 years experience
5	Roshan Singh Bodhi	roshanasingh43@gmail.com	316799178	Java Developer No experience
6	Tarak Mehta	tarakmehta_23@gmail.com	317030526	IT Manager 3 years experience
7	Popalpat Pande	pande24@gmail.com	317404474	Database Engineer no experience
8	Babita Iyer	babita1er@gmail.com	3177296122	QA Analyst 6 years experience
9	Abdul Sheikh	abduls54@gmail.com	3180531770	Software Developer 2 years experience
10	Natwari Udhawala	natwariu45@gmail.com	3183765418	Senior Business Analyst 7 years experience

JobID	PositionID	CategoryID	JobType	Medium
1	9	4	3	onsite
2	2	6	3	1
3	3	1	5	2
4	4	10	6	4
5	5	3	5	2
6	6	2	3	1
7	7	13	7	4
8	8	8	4	3
9	9	5	3	2
10	10	12	2	1

JobOpeningID	JobID	NoOfPositions
1	1	3
2	2	5
3	3	3
4	4	1
5	5	5
6	6	3
7	7	1

Query executed successfully.

LAPTOP-6LLHCP92 (15.0 RTM) | LAPTOP-6LLHCP92\sidhh ... | Recruitment | 00:00:00 | 27 rows

SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\sidhh (67)) - Microsoft SQL Server Management Studio

```

File Edit View Query Project Tools Window Help
New Query Execute
Object Explorer Connect
Recruitment
SQLQuery2.sql - LA...LHCP92\sidhh (67)* SQLQuery1.sql - LA...LHCP92\sidhh (51)*
1: SELECT * FROM Applications;
2: SELECT * FROM Onboarding;
3: SELECT * FROM Complaint;
4:

```

107 %

Results Messages

ApplicationID	JobOpeningID	CandidateID	ApplicationStatus	ApplicationStartDate	LastUpdateDate
1	1	1	Applied	2022-12-11 13:55:11.1666667	2022-12-08 13:55:11.1666667
2	1	3	interview 1	2022-12-11 13:55:11.1666667	2022-12-06 13:55:11.1666667
3	3	4	Applied	2022-12-11 13:55:11.1666667	2022-12-10 13:55:11.1666667
4	4	5	Rejected	2022-12-11 13:55:11.1666667	2022-12-09 13:55:11.1666667
5	5	1	offer extended	2022-12-11 13:55:11.1666667	2022-12-08 13:55:11.1666667
6	6	3	interview 3	2022-12-11 13:55:11.1666667	2022-12-05 13:55:11.1666667
7	7	1	Applied	2022-12-11 13:55:11.1666667	2022-12-01 13:55:11.1666667
8	8	2	Waiting	2022-12-11 13:55:11.1666667	2022-12-06 13:55:11.1666667
9	9	4	Applied	2022-12-11 13:55:11.1666667	2022-12-10 13:55:11.1666667
10	10	4	Applied	2022-12-11 13:55:11.1666667	2022-12-03 13:55:11.1666667
11	11	7	Onboarding	2022-12-11 13:55:11.1666667	2022-12-08 13:55:11.1666667

CandidateID	JobID	StartDate
1	5	2022-12-31 13:55:11.1833333
2	5	2023-01-10 13:55:11.1833333

ComplaintID	ApplicationID	Description	Status
1	8	Interviewer asked questions not aligned with th...	Under Review

Query executed successfully.

LAPTOP-6LLHCP92 (15.0 RTM) | LAPTOP-6LLHCP92\sidhh ... | Recruitment | 00:00:00 | 14 rows

SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\sidhh (67)) - Microsoft SQL Server Management Studio

```

File Edit View Query Project Tools Window Help
New Query Execute
SQLQuery2.sql - LA...LHCP92\sidhh (67)* SQLQuery1.sql - LA...LHCP92\sidhh (51)*
1: SELECT * FROM Interviewer;
2: SELECT * FROM InterviewerMapping;
3: SELECT * FROM Interviews;
4:

```

107 % Results Messages

InterviewerID	Name	Department	Title
1	Sumeet Raghav	IT	Project Manager
2	Siddharth Malhotra	Analysts	Business Analyst
3	Dilip Tahl	IT	Team Lead
4	Abul Anand	Management	Product Manager
5	Sachin Patil	IT	Cyber Security

InterviewID	InterviewerID
1	1
2	2
3	5
4	4
5	3
6	4
7	1
8	2
9	4
10	3
n	e

InterviewID	ApplicationID	RemarkID	StartTime	EndTime	Type	Result
1	1	1	2022-12-10 11:55:11.170000	2022-12-10 12:55:11.170000	online	accepted
2	2	4	2022-12-08 11:55:11.170000	2022-12-08 12:55:11.170000	online	accepted
3	3	4	2022-12-10 11:55:11.170000	2022-12-10 12:55:11.170000	online	rejected
4	4	6	2022-12-08 11:55:11.170000	2022-12-08 12:55:11.170000	online	accepted
5	5	6	2022-12-09 11:55:11.170000	2022-12-09 12:55:11.170000	online	accepted
6	6	6	2022-12-07 11:55:11.170000	2022-12-07 12:55:11.170000	online	accepted
7	7	7	2022-12-08 11:55:11.170000	2022-12-08 12:55:11.170000	online	accepted
8	8	8	2022-12-10 11:55:11.170000	2022-12-10 12:55:11.170000	online	accepted
9	9	8	2022-12-10 11:55:11.170000	2022-12-10 12:55:11.170000	online	rejected
10	10	10	2022-12-10 07:55:11.170000	2022-12-10 08:55:11.170000	online	accepted
11	11	11	2022-12-10 11:55:11.170000	2022-12-10 12:55:11.170000	online	accepted

Query executed successfully.

LAPTOP-6LLHCP92 (15.0 RTM) | LAPTOP-6LLHCP92\sidhh ... Recruitment 00:00:00 | 32 rows

SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\sidhh (67)) - Microsoft SQL Server Management Studio

```

File Edit View Query Project Tools Window Help
New Query Execute
SQLQuery2.sql - LA...LHCP92\sidhh (67)* SQLQuery1.sql - LA...LHCP92\sidhh (51)*
1: SELECT * FROM Tests;
2: SELECT * FROM GraderMapping;
3: SELECT * FROM InterviewRemarks;
4:

```

107 % Results Messages

TestID	InterviewID	Type	StartTime	EndTime	Answers	Grades
1	1	online	2022-12-10 11:55:11.173333	2022-12-10 11:55:11.173333	1-a-2-c-3-d-4-d	passed
2	2	onsite	2022-12-10 11:55:11.173333	2022-12-10 12:55:11.173333	1>true-2>false	passed
3	3	online	2022-12-08 11:55:11.173333	2022-12-09 12:55:11.173333	1>false-2>false	failed
4	4	onsite	2022-12-10 11:55:11.173333	2022-12-10 12:55:11.173333	1-d-2=c	passed
5	5	online	2022-12-10 11:55:11.173333	2022-12-10 12:55:11.173333	1-a-2-c-3-d-4-d	passed
6	6	onsite	2022-12-10 11:55:11.173333	2022-12-10 12:55:11.173333	1>false-2>true-3>true-4>false-5>a-6>b	passed
7	7	online	2022-12-10 06:55:11.173333	2022-12-05 12:55:11.173333	2>a-3=e	passed
8	8	onsite	2022-12-10 11:55:11.173333	2022-12-10 12:55:11.173333	1-a-2-c-3-d-4-d	passed
9	9	online	2022-12-10 11:55:11.173333	2022-12-10 12:55:11.173333	1>false	failed

InterviewerID	TestID
1	1
2	2
3	3
4	4
5	2
6	1
7	3
8	4
9	2
n	9

RemarkID	InterviewerRemarks	CandidateRemarks
1	Candidate seems to be good based on the 1st interview.	Cooperative Interviewer
2	Candidate can proceed with round 2. Candidate...	NULL
3	Candidate couldn't answer the 50% questions in th...	Tough Interview
4	Candidate cleared the 1st technical round.	NULL
5	Candidate cleared the 2nd technical round. Can pr...	NULL
6	Candidate cleared the 3rd round of interview	NULL
7	Candidate is a good fit for our requirements.	NULL
8	Candidate cleared the 2nd round of interview. Can...	NULL
9	Candidate is rejected in the 1st interview. Doesn't m...	Rude interviewer. Ask...
n	Candidate did well in the 1st round.	AMM

Query executed successfully.

LAPTOP-6LLHCP92 (15.0 RTM) | LAPTOP-6LLHCP92\sidhh ... Recruitment 00:00:00 | 29 rows

SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\sidhh (67)) - Microsoft SQL Server Management Studio

```

1 SELECT * FROM ReimbursementType;
2 SELECT * FROM Reimbursement;

```

107 %

Results Messages

RequestTypeID	RequestName	StandardAmt
1	Airline reservation details	700.00
2	Hotel reservation details	300.00
3	Car Rental	150.00
4	Food Expenses	100.00

ReimbursementID	InterviewID	RequestTypeID	RequestedAmt	ProcessedAmt	ReceiptLink	IsProcessed	
1	1	2	500.00	500.00	1	emirates.pdf	
2	2	7	100.00	100.00	1	mirador.docx	
3	3	4	100.00	100.00	1	zipcar.docx	
4	4	8	1	700.00	700.00	1	elite.pdf
5	5	2	350.00	300.00	1	dropoff.pdf	
6	6	4	100.00	100.00	1	sandeep.docx	
7	7	2	100.00	0.00	0	refill.pdf	
8	8	9	1	600.00	600.00	1	airindia.docx
9	9	10	2	300.00	300.00	1	
10	10	9	2	350.00	0.00	0	ranch.docx
11	11	9	4	50.00	50.00	1	chipotle.pdf
12	12	8	2	200.00	200.00	1	fivestar.pdf

Query executed successfully.

LAPTOP-6LLHCP92 (15.0 RTM) | LAPTOP-6LLHCP92\sidhh .. | Recruitment | 00:00:00 | 16 rows

SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\sidhh (62)) - Microsoft SQL Server Management Studio

```

1 select * from DocumentType;
2 select * from Documents;
3 select * from Onboarding;
4 select * from Offers;

```

129 %

Results Messages

DocumentTypeID	DocumentTypeName
1	CV
2	Reference Letter
3	Cover Letter

DocumentID	CandidateID	DocumentTypeID	DocumentLink
1	1	1	cv.pdf
2	2	1	cv_developer.pdf
3	3	2	rl_developer.pdf
4	4	3	cv.pdf
5	5	4	cv_manager.pdf
6	6	5	cv.pdf
7	7	6	cover1.pdf
8	8	8	resume_BA.pdf
9	9	8	rl_BA.pdf
10	10	9	resume3.pdf
11	11	10	resume.pdf
12	12	8	cover2.pdf

CandidateID	JobID	StartDate
1	5	2022-12-31 05:25:54.6633333
2	7	2023-01-10 05:25:54.6633333

OfferID	CandidateID	ApplicationID	OfferStatus
1	3	1	actionNeeded
2	4	5	accepted
		11	

Query executed successfully.

LAPTOP-6LLHCP92 (15.0 RTM) | LAPTOP-6LLHCP92\sidhh .. | Recruitment | 00:00:00 | 3 rows

TESTING

VIEWS

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

```
USE Recruitment
GO
```

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

```
select * from InterviewRebursements;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'Recruitment' is selected. In the main query window, a new view 'InterviewRebursements' is being created with the following T-SQL code:

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

Below the code, a 'Results' tab displays the output of the 'select * from InterviewRebursements;' query. The results show 17 rows of reimbursement data, including columns: InterviewID, ReimbursementID, RequestName, RequestedAmt, ProcessedAmt, and IsProcessed. Some rows have NULL values in the RequestName and ProcessedAmt columns. The RequestName column contains entries like 'Airline reservation details', 'Food Expenses', 'Car Rental', etc. The IsProcessed column has values like 'emirates.pdf', 'dragontail.pdf', 'mirador.docx', etc.

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

```

USE Recruitment
GO

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

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

```

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery2.sql' is open, displaying the T-SQL code for creating a view. The results pane below shows the output of the 'SELECT' statement, which returns two rows of data. The code and results are as follows:

```

USE Recruitment
GO

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

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

```

InterviewerID	TotalNoOfInterviews
1	4
2	4

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

```

USE Recruitment
GO

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

```

```
SELECT * FROM ApplicationComplaintsRaised;
```

```
SQLQuery2.sql - LAPTOP-6LHCP92.Recruitment (LAPTOP-6LHCP92\sidhh (67)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
Object Explorer
Connect ▾ Recruitment
SQLQuery2.sql - LA...LHCP92\sidhh (67)* SQLQuery1.sql - LA...LHCP92\sidhh (51)*
1 USE Recruitment
2 GO
3
4 CREATE VIEW ApplicationComplaintsRaised AS
5 SELECT a.ApplicationID, c.ComplaintID, c.Description, c.Status
6 FROM Applications a LEFT JOIN Complaint c ON a.ApplicationID = c.ApplicationID
7 WHERE c.ComplaintID IS NOT NULL
8
9 SELECT * FROM ApplicationComplaintsRaised;
10
107 %
Results Messages
ApplicationID ComplaintID Description Status
1 8 1 Interviewer asked questions not aligned with th... Under Review
1 Query executed successfully.
LAPTOP-6LHCP92 (15.0 RTM) LAPTOP-6LHCP92\sidhh ... Recruitment 00:00:00 1 rows
```

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

```
USE Recruitment
GO
```

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

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

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'Recruitment' is selected. In the center pane, there are two tabs: 'SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\sidhh (67))' and 'SQLQuery1.sql - LA...LHCP92\sidhh (51)'. The SQLQuery2.sql tab contains the following T-SQL code:

```

1 USE Recruitment
2 GO
3
4 CREATE VIEW ApplicationsReceived AS
5 SELECT a.ApplicationID, c.CandidateID, c.CandidateName, c.ShortProfile, a.ApplicationStatus, a.LastUpdateDate
6 FROM Applications a LEFT JOIN Candidates c ON a.CandidateID = c.CandidateID
7 WHERE lower(a.ApplicationStatus) = 'applied';
8
9 |SELECT * FROM ApplicationsReceived ORDER BY LastUpdateDate ASC;
10

```

The Results grid displays the following data:

	ApplicationID	CandidateID	CandidateName	ShortProfile	ApplicationStatus	LastUpdateDate
1	7	6	Tarik Mehta	IT Manager 3 years experience	Applied	2022-12-01 13:55:11.1666667
2	10	10	Nawaraj Udhawa	Senior Business Analyst 7 years experience	Applied	2022-12-03 13:55:11.1666667
3	1	1	Bidhita Nilam	Java Developer 5 Years Experience	Applied	2022-12-08 13:55:11.1666667
4	3	5	Roshan Singh Sodhi	Java Developer No experience	Applied	2022-12-10 13:55:11.1666667
5	9	9	Abdul Sheikh	Software Developer 2 years experience	Applied	2022-12-10 13:55:11.1666667

At the bottom of the results grid, it says 'Query executed successfully.' and shows the session details: 'LAPTOP-6LLHCP92 (15.0 RTM) LAPTOP-6LLHCP92\sidhh ... Recruitment | 00:00:00 | 5 rows'.

STORED PROCEDURES

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

Query :

```

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

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

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

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'Recruitment' is selected. In the main pane, the code for the stored procedure 'interviewsCount' is displayed. The code creates a temporary table #Interviews, performs a join between Interviewer and InterviewerMapping, and then joins with Interviews to count the number of interviews for each interviewer within specified date ranges. Finally, it executes the stored procedure with parameters '1', '2022-12-10 01:00:00', and '2022-12-11 12:00:00'. The Results pane shows the output:

InterviewerID	CountOfInterviews
1	3

At the bottom, a message indicates 'Query executed successfully.'

2. Create a stored procedure named spCountApplication that accepts status. The procedure should return a result set consisting of Number of candidates having the given status.

Database has 5 applicants with applied status.

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

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

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'Recruitment' is selected. In the center pane, a query window titled 'SQLQuery2.sql - LAPTOP-6LLHCP92\sidhh (67)* - Microsoft SQL Server Management Studio' contains the following T-SQL code:

```

23
24
25 CREATE PROC spCountApplication
26     @Status varchar(25) = NULL
27     AS
28 IF @Status IS NULL
29     SET @Status = ''
30 DECLARE @CountCandidates int = 0
31 IF @CountCandidates = 0
32     SELECT @CountCandidates = COUNT(*) FROM Applications WHERE ApplicationStatus= @Status
33     SELECT @Status AS ApplicationStatus, @CountCandidates AS NoOfApplications;
34
35 EXEC spCountApplication @Status = 'applied';

```

The results pane shows a single row of data:

ApplicationStatus	NoOfApplications
applied	5

At the bottom of the interface, a status bar indicates: 'Query executed successfully.' and 'LAPTOP-6LLHCP92 (15.0 RTM) LAPTOP-6LLHCP92\sidhh ... Recruitment 0000000 0 rows'.

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

```

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

```

```
EXEC spReimbursementsDone 0;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'Recruitment' is selected. In the center pane, two queries are running: 'SQLQuery2.sql' and 'SQLQuery1.sql'. The code in 'SQLQuery2.sql' is a stored procedure definition:

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

The results of the query show 12 rows of reimbursement details:

ReimbursementID	RequestName	ProcessedAmt	RequestedAmt
1	Food Expenses	100.00	200.00
2	Food Expenses	100.00	100.00
3	Car Rental	100.00	100.00
4	Airline reservation details	700.00	700.00
5	Hotel reservation details	300.00	350.00
6	Food Expenses	100.00	100.00
7	Airline reservation details	600.00	600.00
8	Hotel reservation details	300.00	300.00
9	Hotel reservation details	200.00	200.00
10	Food Expenses	100.00	100.00
11	Car Rental	100.00	100.00
12	Airline reservation details	700.00	700.00

At the bottom, a message indicates "Query executed successfully." and the status bar shows "LAPTOP-6LLHCP92 (15.0 RTM) LAPTOP-6LLHCP92\sidhh... Recruitment 00:00:00 9 rows".

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

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

```
EXEC candidatesRejected;
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'Recruitment' is selected. In the center pane, a query window titled 'SQLQuery2.sql' contains the following T-SQL code:

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

EXEC candidatesRejected;
```

Below the code, the 'Results' tab displays the output of the executed query. The table has columns: CandidateID, ApplicationID, CandidateName, ApplicationStatus, and LastUpdateDate. The data is as follows:

CandidateID	ApplicationID	CandidateName	ApplicationStatus	LastUpdateDate
1	7	Popal Pandey	Waiting	2022-12-06 13:55:11.1666667
2	8	Babita Yer	Rejected	2022-12-09 13:55:11.1666667

At the bottom of the screen, a message bar indicates 'Query executed successfully.'

USER DEFINED FUNCTIONS

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

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

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

```

File Edit View Query Project Tools Window Help
File Edit View Query Project Tools Window Help
SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\sidhh (67)) - Microsoft SQL Server Management Studio
SQLQuery1.sql - LA...LHCP92\sidhh (51)*
Object Explorer
Connect ...
SQLQuery2.sql - LAPTOP-6LLHCP92 (SQL Server 15.0.2000)
  Databases
  Security
  Server Objects
  Replication
  PolyBase
  Always On High Availability
  Management
  Integration Services Catalogs
  SQL Server Agent (Agent XPs disabled)
  XEvent Profiler
53
54
55 =CREATE FUNCTION fnPassedTestEarly()
56   RETURNS INT
57   BEGIN
58     RETURN
59     (SELECT TestID
60      FROM Tests
61     WHERE lower(Grades) = 'passed'
62     ORDER BY EndTime asc
63     OFFSET 0 ROWS
64     FETCH FIRST 1 ROWS ONLY);
65
66
67
68   SELECT i.ApplicationID,i.ApplicationID,i.RemarkID,i.Type,i.Result
69   FROM TESTS t JOIN Interviews i ON t.InterviewID = i.InterviewID
70   WHERE t.TestID = dbo.fnPassedTestEarly();
71
72
73
117%  Messages
Results
ApplicationID ApplicationID RemarkID Type Result
1 5 7 onsite accepted
Query executed successfully.

```

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

```

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

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

```

```

File Edit View Query Project Tools Window Help
File Edit View Query Project Tools Window Help
SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\sidhh (67)) - Microsoft SQL Server Management Studio
SQLQuery1.sql - LA...LHCP92\sidhh (51)*
Quick Launch (Ctrl+Q) P X
Object Explorer
Connect - > 
LAPTOP-6LLHCP92 (SQL Server 15.0.2000)
Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent (Agent XPs disabled) XEvent Profiler
53
54
55 =CREATE FUNCTION fnPassedTestEarly()
56 RETURNS INT
57 BEGIN
58 RETURN
59 (SELECT TestID
60 FROM Tests
61 WHERE lower(Grades) = 'passed'
62 ORDER BY EndTime asc
63 OFFSET 0 ROWS
64 FETCH FIRST 1 ROWS ONLY);
65
66
67
68 SELECT i.ApplicationID,i.ApplicationID,i.RemarkID,i.Type,i.Result
69 FROM TESTS t JOIN Interviews i ON t.InterviewID = i.InterviewID
70 WHERE t.TestID = dbo.fnPassedTestEarly();
71
72
73
117% Messages
ApplicationID ApplicationID RemarkID Type Result
1 5 7 onsite accepted
Query executed successfully.
LAPTOP-6LLHCP92 (15.0 RTM) LAPTOP-6LLHCP92\sidhh ... Recruitment | 00:00:00 | 1 rows

```

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

```

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

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

SELECT *

```

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

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer displays the database structure for 'LAPTOP-6LHCP92'. In the center, a query window titled 'SQLQuery2.sql - LAPTOP-6LHCP92\sidhh (67)*' contains the following T-SQL code:

```

CREATE FUNCTION fnInterviewsByInterviewer (@InterviewerID int, @DateMin datetime2,
                                         @DateMax datetime2)
RETURNS TABLE
RETURN
(SELECT i.InterviewerID, COUNT(ins.InterviewID) as CountOfInterviews
FROM Interviewer i LEFT JOIN InterviewerMapping im ON i.InterviewerID = im.InterviewerID LEFT JOIN Interviews ins ON im.InterviewID = ins.
WHERE ins.StartTime > @DateMin
    and ins.EndTime < @DateMax
    and i.InterviewerID = @InterviewerID
GROUP BY i.InterviewerID)
SELECT *
FROM dbo.fnInterviewsByInterviewer(1, '2022-12-10 01:00:00', '2022-12-11 12:00:00');

```

The results pane shows a single row of data:

InterviewerID	CountOfInterviews
1	3

At the bottom of the screen, a status bar indicates 'Query executed successfully.' and the session details: 'LAPTOP-6LHCP92 (15.0 RTM) | LAPTOP-6LHCP92\sidhh... | Recruitment | 00:00:00 | 1 rows'.

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

```

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

```

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

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane displays the database structure for 'LAPTOP-6LLHCP92'. In the center, the 'SQLQuery2.sql' window contains the following T-SQL code:

```
123
124
125
126 CREATE FUNCTION fnOffersByCandidate (@CandidateID int = 0)
127 RETURNS TABLE
128 RETURN
129 (SELECT c.CandidateID, c.CandidateName, c.ShortProfile, o.OfferStatus
130 FROM Candidates c JOIN Offers o ON c.CandidateID = o.CandidateID
131 WHERE c.CandidateID = @CandidateID);
132
133 SELECT * FROM dbo.fnOffersByCandidate(5);
```

The 'Results' tab in the bottom pane shows the output of the query:

	CandidateID	CandidateName	ShortProfile	OfferStatus
1	5	Roshan Singh Sodhi	Java Developer No experience	accepted

A status bar at the bottom indicates: 'Query executed successfully.' and 'LAPTOP-6LLHCP92 (15.0 RTM) LAPTOP-6LLHCP92\sidhh... Recruitment 00:00:00 | 1 rows'.

TRANSACTIONS:

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

```
BEGIN TRAN
```

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

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

```
COMMIT TRAN
```

```
SELECT * FROM OnBoarding
SELECT * FROM Applications
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer pane, a database named 'Recruitment' is selected. In the Results pane, two queries are displayed:

```

58 BEGIN TRAN
59
60
61 INSERT INTO Onboarding VALUES
62 (6,3,dateadd(w, -20,GETDATE()))
63
64 UPDATE Applications SET ApplicationStatus = 'Onboarding' WHERE CandidateID = 6 and
65 JobOpeningID = (SELECT JobOpeningID FROM JobOpenings WHERE JobID=3)
66
67 COMMIT TRAN
68
69
70
71 SELECT * FROM OnBoarding
72 SELECT * FROM Applications

```

The results of the second query show three rows in the 'OnBoarding' table:

CandidateID	JobID	StartDate
1	5	2022-12-31 13:55:11.1833333
2	5	2023-01-10 13:55:11.1833333
3	6	2022-11-21 20:51:10.7766667

The results of the third query show 11 rows in the 'Applications' table:

ApplicationID	JobOpeningID	CandidateID	ApplicationStatus	ApplicationStartDate	LastUpdateDate
1	1	1	Applied	2022-12-11 13:55:11.1666667	2022-12-08 13:55:11.1666667
2	2	1	Waiting	2022-12-11 13:55:11.1666667	2022-12-06 13:55:11.1666667
3	3	4	Applied	2022-12-11 13:55:11.1666667	2022-12-10 13:55:11.1666667
4	4	5	Rejected	2022-12-11 13:55:11.1666667	2022-12-09 13:55:11.1666667
5	5	5	Offer Extended	2022-12-11 13:55:11.1666667	2022-12-08 13:55:11.1666667
6	6	3	Onboarding	2022-12-11 13:55:11.1666667	2022-12-08 13:55:11.1666667
7	7	1	Applied	2022-12-11 13:55:11.1666667	2022-12-01 13:55:11.1666667
8	8	2	Waiting	2022-12-11 13:55:11.1666667	2022-12-06 13:55:11.1666667
9	9	4	Applied	2022-12-11 13:55:11.1666667	2022-12-10 13:55:11.1666667
10	10	4	Applied	2022-12-11 13:55:11.1666667	2022-12-03 13:55:11.1666667
11	11	5	Onboarding	2022-12-11 13:55:11.1666667	2022-12-08 13:55:11.1666667

At the bottom of the Results pane, it says "Query executed successfully."

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

```

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

```

GO

BEGIN TRAN

```

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

```

DELETE FROM Reimbursement

```

WHERE ReimbursementID IN(SELECT ReimbursementID FROM
ReimbursementArchive);

COMMIT TRAN;

SELECT * FROM Reimbursement;

SELECT * FROM ReimbursementArchive;

```

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer displays a connection to 'LAPTOP-6LLHCP92' (SQL Server 15.0). In the center, there are two query panes. The top pane contains the following SQL code:

```

159
160
161 SELECT * FROM Reimbursement;
162
163 SELECT * FROM ReimbursementArchive;
164
165
166
167

```

The bottom pane shows the results of the second query, which is a subset of the data shown in the top pane. The results are as follows:

ReimbursementID	InterviewID	RequestTypeID	RequestedAmt	ProcessedAmt	ReceiptLink	IsProcessed
7	2	4	100.00	0.00	restBill.pdf	0
10	9	2	350.00	0.00	ranch.docx	0

Below this, another set of results is shown for the 'ReimbursementArchive' table, identical to the ones above:

ReimbursementID	InterviewID	RequestTypeID	RequestedAmt	ProcessedAmt	ReceiptLink	IsProcessed
1	2	1	500.00	500.00	emirates.pdf	1
2	7	4	100.00	100.00	mriadoor.docx	1
3	4	3	100.00	100.00	zipcar.docx	1
4	8	1	700.00	700.00	ethad.pdf	1
5	2	2	350.00	300.00	dragonfly.pdf	1
6	6	4	100.00	100.00	sandeep.docx	1
7	9	1	600.00	600.00	airindia.docx	1
8	10	2	300.00	300.00		1
9	11	4	50.00	50.00	chiptole.pdf	1
10	12	2	200.00	200.00	fuerstar.pdf	1

At the bottom of the interface, a message says 'Query executed successfully.' and the status bar indicates 'LAPTOP-6LLHCP92 (15.0 RTM) | LAPTOP-6LLHCP92\sidhh ... Recruitment | 00:00:00 | 12 rows'.

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

```

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

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

COMMIT TRAN;

SELECT * FROM Complaint
SELECT * FROM Applications

```

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane displays the database structure for 'Recruitment'. In the center, the 'SQLQuery1.sql' window contains a transaction script:

```

73 BEGIN TRAN
74 INSERT INTO Complaint VALUES (7, 'Very rude behavior from interviewer', 'Under Review')
75
76 UPDATE Applications
77 SET ApplicationStatus = 'waiting'
78 WHERE ApplicationID = 7;
79
80 COMMIT TRAN;
81
82 SELECT * FROM Complaint
83 SELECT * FROM Applications

```

The 'Results' tab shows the output of the 'SELECT' statements. The 'Complaint' table has three rows with ComplaintID 1, 2, and 3. The 'Applications' table has 11 rows, with ApplicationID 7 being updated to 'waiting' status.

At the bottom, a green message indicates: 'Query executed successfully.'

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

```
BEGIN TRAN
```

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

```
COMMIT TRAN;
```

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

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer pane displays the database structure for 'Recruitment'. In the center, the 'SQLQuery1.sql' window contains a transaction script:

```

87 BEGIN TRAN
88
89 UPDATE Job SET Medium = 'online' WHERE JobID IN (SELECT DISTINCT JobID FROM JobOpenings)
90
91 COMMIT TRAN;
92
93 SELECT * FROM Job JOIN JobOpenings ON Job.JobID = JobOpenings.JobID

```

The 'Results' tab shows the output of the 'SELECT' statement. It lists 13 rows of data from the 'Job' and 'JobOpenings' tables joined on 'JobID'.

At the bottom, a green message indicates: 'Query executed successfully.'

SCRIPTS

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

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

The screenshot shows the Microsoft SQL Server Management Studio interface. On the left, the Object Explorer displays the database structure for 'Recruitment'. In the center, two query panes are open: 'SQLQuery2.sql' and 'SQLQuery1.sql'. The code in 'SQLQuery2.sql' is the script provided above. The code in 'SQLQuery1.sql' is a placeholder. The status bar at the bottom indicates 'Query executed successfully.' and shows the completion time as 2022-12-11T19:44:57.0007766-05:00. The bottom right corner shows the session details: LAPTOP-6LHCP92 (15.0 RTM) | LAPTOP-6LHCP92\sidhh | Recruitment | 00:00:00 | 0 rows.

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

```

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

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows a connection to 'LAPTOP-6LLHCP92.Reruitment'. The main pane displays a T-SQL script named 'SQLQuery2.sql' with the following content:

```

203 USE Recruitment;
204 DECLARE @sqlstr VARCHAR(4000)
205 IF EXISTS(select 1 FROM SYS.VIEWS WHERE NAME = 'LatestApplication')
206 BEGIN
207     SELECT *
208     FROM LatestApplication;
209 END
210 ELSE
211 BEGIN
212     SET @sqlstr =
213     'CREATE VIEW latestApplication
214     AS
215     SELECT CandidateID, MAX(LastUpdateDate) AS Latest_Invoice_Date
216     FROM Applications
217     GROUP BY CandidateID'
218     EXEC(@sqlstr)
219 END
220

```

The 'Results' tab at the bottom shows the output of the query:

CandidateID	Latest_Invoice_Date
1	2022-12-08 13:55:11.1666687
2	2022-12-06 13:55:11.1666687
3	2022-12-10 13:55:11.1666687
4	2022-12-01 13:55:11.1666687
5	2022-12-06 13:55:11.1666687
6	2022-12-09 13:55:11.1666687
7	2022-12-10 13:55:11.1666687
8	2022-12-03 13:55:11.1666687
9	
10	

A status bar at the bottom right indicates 'Query executed successfully.' and 'LAPTOP-6LLHCP92 (15.0 RTM) | LAPTOP-6LLHCP92\sidhh ... | Recruitment | 00:00:00 | 8 rows'.

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

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

```

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

```

The screenshot shows the Microsoft SQL Server Management Studio interface. The left pane displays the Object Explorer with a connection to 'LAPTOP-6LLHCP92.Recruitment'. The right pane contains two tabs: 'SQLQuery2.sql' and 'SQLQuery1.sql'. The 'SQLQuery2.sql' tab shows the provided dynamic SQL script. The 'Results' tab shows the output of the script, which is a table named 'LatestInterview' with two columns: 'InterviewerID' and 'LatestInterview'. The data is as follows:

InterviewerID	LatestInterview
1	2022-12-10 12:55:11.170000
2	2022-12-10 12:55:11.170000
3	2022-12-10 12:55:11.170000
4	2022-12-10 12:55:11.170000
5	2022-12-10 12:55:11.170000

At the bottom of the interface, a yellow bar indicates 'Query executed successfully.'

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

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

```
USE Recruitment;
```

```
DECLARE @First_tab varchar(4000);
SET @First_tab =(SELECT TOP 1 name
FROM sys.tables
WHERE name <> 'dtproperties' OR name <> 'sysdiagrams'
ORDER BY name);
EXEC ('SELECT COUNT(*) AS CountOf' + @First_tab + ' FROM '
+@First_tab+');';
```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a connection to 'LAPTOP-6LLHCP92 (SQL Server 15.0.200)' is selected. In the center pane, there are two tabs: 'SQLQuery2.sql - LAPTOP-6LLHCP92\sidhh (67)*' and 'SQLQuery1.sql - LAPTOP-6LLHCP92\sidhh (51)*'. The 'SQLQuery1.sql' tab contains the provided T-SQL code. The 'Results' tab shows the output of the executed query, which is a single row with a value of 11. At the bottom, a status bar indicates 'Query executed successfully.' and the session details: 'LAPTOP-6LLHCP92 (15.0 RTM) LAPTOP-6LLHCP92\sidhh ... Recruitment 00:00:00 | 1 rows'.

Script Roles

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

```

100 CREATE ROLE ComplaintRaiser
101 GO
102 GRANT INSERT,UPDATE,DELETE on Complaint to ComplaintRaiser
103 GO
104 GRANT SELECT on DATABASE::Recruitment to ComplaintRaiser;
105 GO
106
107

```

Messages
Commands completed successfully.
Completion time: 2022-12-11T21:48:00.7722891-08:00

Query executed successfully.

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

```

108 USE Recruitment
109 GO
110
111
112 CREATE ROLE InterviewingDept
113 GO
114 GRANT SELECT ON Interviews to InterviewingDept
115 GO
116 GRANT SELECT ON Interviewer to InterviewingDept
117 GO
118 GRANT SELECT ON InterviewRemarks to InterviewingDept
119 GO
120 GRANT SELECT ON InterviewerMapping to InterviewingDept
121 GO
122 GRANT SELECT, UPDATE ON JobOpenings to InterviewingDept;

```

Messages
Commands completed successfully.
Completion time: 2022-12-11T21:52:22.7247909-08:00

Query executed successfully.

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

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\sidhh (55)) - Microsoft SQL Server Management Studio". The left pane is the Object Explorer, displaying the database structure of "LAPTOP-6LLHCP92 (SQL Server 15.0.2000.5 - LAPTOP-6LLHCP92\sidhh)". The "Recruitment" database is expanded, showing its objects like Tables, Views, External Resources, Synonyms, Programmability, Service Broker, Storage, Security, and School. The right pane contains two query windows. The top window, titled "SQLQuery2.sql - LA...LHC92\sidhh (55)*", contains the following T-SQL code:

```
126 CREATE ROLE InsertJobs;
127 GRANT INSERT, UPDATE ON JobOpenings TO InsertJobs
128 GO
129
130
131
```

The status bar at the bottom indicates "Query executed successfully." and "LAPTOP-6LLHCP92 (15.0 RTM) | LAPTOP-6LLHCP92\sidhh... | Recruitment | 00:00:00 | 0 rows".

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

SQLQuery2.sql - LAPTOP-6LLHCP92.Recruitment (LAPTOP-6LLHCP92\sidhh (55)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Recruitment New Query Execute

Object Explorer

Connect

SQLQuery2.sql - LA... - LAPTOP-6LLHCP92\sidhh (55)* SQLQuery1.sql - LA... - LAPTOP-6LLHCP92\sidhh (66)*

```
144
145 CREATE ROLE CandidateRecord;
146 GRANT UPDATE ON Candidates TO CandidateRecord
147 GO
148 
149 CREATE LOGIN Siddhita WITH PASSWORD = 'snikam234',
150 DEFAULT_DATABASE = Recruitment;
151 CREATE USER Siddhita FOR LOGIN Siddhita;
152 ALTER ROLE CandidateRecord ADD MEMBER SiddhitaN;
153
154
```

96 %

Messages

Commands completed successfully.

Completion time: 2022-12-11T22:07:14.0777144+06:00

Query executed successfully.

TRIGGERS

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

```
CREATE TRIGGER RecruitmentDB_CreateTable
```

```

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

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

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

```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'LAPTOP-6LLHCP92' is selected. In the center pane, a new query window titled 'SQLQuery2.sql - LAPTOP-6LLHCP92\sidhh (67)*' is open, displaying the T-SQL code for creating a trigger. The trigger is named 'Recruitment_CreateTable' and is defined to fire after a table is created ('AFTER CREATE_TABLE'). It declares variables for XML data and event type, and prints a message indicating a new table has been created if the event type matches 'CREATE_TABLE'. Below the code, the 'Messages' pane shows the execution results, including the creation of the 'Onboarded_candidate' table and a message indicating a new table has been created.

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

```

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

```

```

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

```

```

USE Recruitment
GO
DROP TABLE Onboarded_candidate

```

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'LAPTOP-6LLHCP92' is selected. In the center pane, a query window titled 'SQLQuery2.sql - LAPTOP-6LLHCP92.master (LAPTOP-6LLHCP92\sidhh (67)) - Microsoft SQL Server Management Studio' contains the following T-SQL code:

```

269
270
271
272
273 CREATE TRIGGER Recruitment_CreateTable
274 ON DATABASE
275 AFTER CREATE_TABLE
276 AS
277 DECLARE @EventData xml;
278 SELECT @EventData = EVENTDATA();
279 DECLARE @EventType varchar(100);
280 SET @EventType =
281 @EventData.value('/EVENT_INSTANCE/EventType{1}', 'varchar(100)');
282 IF @EventType = 'CREATE_TABLE'
283 PRINT 'A new table has been created.';
284 ELSE
285 PRINT 'A table has been dropped.';
286 PRINT CONVERT(varchar(max), @EventData);
287
288 CREATE TABLE OnboardedCandidate
289 (OnboardingID int NOT NULL PRIMARY KEY,
290 CandidateFirstName varchar(50) NOT NULL,
291 CandidateLastName varchar(50),
292 StartDate datetime2 NOT NULL);
293
294

```

Below the code, the 'Messages' section displays the execution results:

```

A new table has been created.
<EVENT_INSTANCE><EventType>CREATE_TABLE</EventType><PostTime>2022-12-11T20:00:08.430+05'<SPID>47</SPID><ServerName>LAPTOP-6LLHCP92</ServerName><LoginName>LAPTOP-6LLHCP92\sidhh</LoginName><DatabaseName>master</DatabaseName><CommandText>CREATE TABLE OnboardedCandidate</CommandText></EVENT_INSTANCE>
Completion time: 2022-12-11T20:00:08.4472418+05:00

```

The status bar at the bottom right indicates 'Query executed successfully.' and shows the session details: LAPTOP-6LLHCP92 (15.0 RTM) | LAPTOP-6LLHCP92\sidhh ... | master | 00:00:00 | 0 rows.

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

```

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

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

```

```

1  USE Recruitment
2  GO
3
4  CREATE TRIGGER ComplaintInserts
5  ON Complaint
6  AFTER INSERT
7  AS
8
9  UPDATE Applications
10 SET ApplicationStatus = 'waiting'
11 WHERE ApplicationID IN (SELECT ApplicationID FROM Complaint WHERE ComplaintID = (SELECT MAX(ComplaintID) FROM Complaint));
12
13 =INSERT INTO Complaint VALUES
14 (2,'Very tough questions and rude behavior from interviewer', 'Under Review')
15

```

Completion time: 2022-12-11T20:34:35.8347428-08:00

Messages

(1 row affected)

(1 row affected)

(1 row affected)

Query executed successfully.

Below rows are affected

```

15
16  SELECT * FROM Applications
17
18  SELECT * FROM Complaint;

```

ApplicationID	JobOpeningID	CandidateID	ApplicationStatus	ApplicationStartDate	LastUpdateDate
1	1	1	Applied	2022-12-11 13:55:11.1666667	2022-12-08 13:55:11.1666667
2	2	1	Waiting	2022-12-11 13:55:11.1666667	2022-12-08 13:55:11.1666667
3	3	4	Applied	2022-12-11 13:55:11.1666667	2022-12-10 13:55:11.1666667
4	4	5	Rejected	2022-12-11 13:55:11.1666667	2022-12-09 13:55:11.1666667
5	5	5	Offer Extended	2022-12-11 13:55:11.1666667	2022-12-08 13:55:11.1666667
6	6	3	Interview 3	2022-12-11 13:55:11.1666667	2022-12-05 13:55:11.1666667
7	7	1	Applied	2022-12-11 13:55:11.1666667	2022-12-01 13:55:11.1666667
8	8	2	Waiting	2022-12-11 13:55:11.1666667	2022-12-06 13:55:11.1666667
9	9	4	Applied	2022-12-11 13:55:11.1666667	2022-12-10 13:55:11.1666667
10	10	4	Applied	2022-12-11 13:55:11.1666667	2022-12-03 13:55:11.1666667
11	11	7	Onboarding	2022-12-11 13:55:11.1666667	2022-12-08 13:55:11.1666667

ComplaintID	ApplicationID	Description	Status
1	8	Interviewer asked questions not aligned with the r...	Under Review
2	2	Very tough questions and rude behavior from inte...	Under Review

Query executed successfully.

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

```

CREATE TRIGGER OnboardingInsert3
ON Onboarding
AFTER INSERT
AS

```

```

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

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

SELECT * FROM Applications;
SELECT * from Onboarding;

```

The screenshot shows the Microsoft SQL Server Management Studio interface with two query panes and a results pane.

Object Explorer: Shows the database structure for 'LAPTOP-6LLHCP92'.

SQLQuery2.sql: Contains the following T-SQL code:

```

CREATE TRIGGER OnboardingInsert3
ON Onboarding
AFTER INSERT
AS
BEGIN
    UPDATE Applications
    SET ApplicationStatus = 'Onboarding'
    WHERE JobOpeningID IN
        (SELECT JobOpeningID FROM JobOpenings WHERE JobID = (SELECT JobID FROM inserted) AND CandidateID = (SELECT CandidateID FROM inserted));

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

    SELECT * FROM Applications
    select * from Onboarding
END

```

SQLQuery1.sql: Contains the following T-SQL code:

```

SELECT * FROM Applications;
SELECT * from Onboarding;

```

Results: Displays the results of the executed queries.

ApplicationID	JobOpeningID	CandidateID	ApplicationStatus	ApplicationStartDate	LastUpdateDate
1	1	1	Applied	2022-12-11 13:55:11.1666667	2022-12-08 13:55:11.1666667
2	2	1	Waiting	2022-12-11 13:55:11.1666667	2022-12-06 13:55:11.1666667
3	3	4	Applied	2022-12-11 13:55:11.1666667	2022-12-10 13:55:11.1666667
4	4	5	Rejected	2022-12-11 13:55:11.1666667	2022-12-09 13:55:11.1666667
5	5	1	Offer Extended	2022-12-11 13:55:11.1666667	2022-12-08 13:55:11.1666667
6	6	3	Onboarding	2022-12-11 13:55:11.1666667	2022-12-05 13:55:11.1666667
7	7	1	Applied	2022-12-11 13:55:11.1666667	2022-12-01 13:55:11.1666667
8	8	2	Waiting	2022-12-11 13:55:11.1666667	2022-12-06 13:55:11.1666667
9	9	4	Applied	2022-12-11 13:55:11.1666667	2022-12-10 13:55:11.1666667
10	10	4	Applied	2022-12-11 13:55:11.1666667	2022-12-03 13:55:11.1666667
11	11	7	Onboarding	2022-12-11 13:55:11.1666667	2022-12-08 13:55:11.1666667

CandidateID	JobID	Start Date
1	5	2023-12-31 13:55:11.1833333
2	5	2023-01-10 13:55:11.1833333
3	6	2022-11-21 20:51:10.7766667

Query executed successfully.

Conclusion

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

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

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