Each student will generate the following:

1. **Two (2) stored procedures to populate transactional tables**
2. **Definition of a business rule**
3. **One (1) user-defined function to enforce the business rule defined above**
4. **One (1) user-defined function to enable a computed column**
5. **One (1) view generating a 'complex' query (includes multiple JOINs, GROUP BY/HAVING statements, and aggregate function)**

Jacinda Eng

Creating Tables

```
CREATE TABLE tblMAJOR_TYPE
(Major_TypeID INTEGER IDENTITY(1,1) primary key,
Major_TypeName varchar(50) not null,
Major_TypeDescr varchar (500) NULL)
GO


INSERT INTO tblMAJOR_TYPE (Major_TypeName, Major_TypeDescr)
VALUES ('STEM', 'College and university degree programs in science, technology,
engineering and mathematics (STEM) are considered STEM degrees, and they are in high
demand across many industries.'),
('Non_STEM', 'Non_stem majors')

CREATE TABLE tblMAJOR
(MajorID INTEGER IDENTITY(1,1) primary key not null,
MajorName varchar(50) not null,
Major_TypeID INTEGER FOREIGN KEY REFERENCES tblMAJOR_TYPE (Major_TypeID) NOT NULL,
MajorDescr varchar(500) NULL)
GO

INSERT INTO tblMAJOR (MajorName, Major_TypeID, MajorDescr)
VALUES ('Accounting', (SELECT Major_TypeID FROM tblMAJOR_TYPE WHERE Major_TypeName =
'Non_STEM'), 'Processing financial information for businesses and corporation'),
```

```
('Computer Science', (SELECT Major_TypeID FROM tblMAJOR_TYPE WHERE Major_TypeName =
'STEM'), 'Programming processes that interact with digital information'),
('Communication', (SELECT Major_TypeID FROM tblMAJOR_TYPE WHERE Major_TypeName =
'Non_STEM'), 'Studying human communication and behavior'), ('Finance', (SELECT
Major_TypeID FROM tblMAJOR_TYPE WHERE Major_TypeName = 'Non_STEM'), 'Assisting
businesses in making financial decisions and maximizing revenue'),('Biology', (SELECT
Major_TypeID FROM tblMAJOR_TYPE WHERE Major_TypeName = 'STEM'), 'Studying the
evolution, genetics, and anatomy of all forms of life from animals to plants'),
('Data Science'', (SELECT Major_TypeID FROM tblMAJOR_TYPE WHERE Major_TypeName =
'STEM'), 'Using algorithms and processes to find information from structured and
unstructured data')

INSERT INTO tblUNIVERSITY_MAJOR (UniversityID, MajorID)
VALUES ((SELECT UniversityID FROM tblUNIVERSITY WHERE UniversityName = 'University of
Washington'), (SELECT MajorID FROM tblMAJOR WHERE MajorName = 'Computer Science')),
((SELECT UniversityID FROM tblUNIVERSITY WHERE UniversityName = 'Seattle University'),
(SELECT MajorID FROM tblMAJOR WHERE MajorName = 'Biology')),
((SELECT UniversityID FROM tblUNIVERSITY WHERE UniversityName = 'Stanford'), (SELECT
MajorID FROM tblMAJOR WHERE MajorName = 'Communication')), ((SELECT UniversityID FROM
tblUNIVERSITY WHERE UniversityName = 'University of Florida'), (SELECT MajorID FROM
tblMAJOR WHERE MajorName = 'Accounting')), ((SELECT UniversityID FROM tblUNIVERSITY
WHERE UniversityName = 'Western Washington University'),
(SELECT MajorID FROM tblMAJOR WHERE MajorName = 'Finance'))

INSERT INTO tblUNIVERSITY_MAJOR (University_MajorID, StudentID)
VALUES ((SELECT University_MajorID FROM tblUNIVERSITY_MAJOR WHERE UniversityName =
'University of Washington'), (SELECT MajorID FROM tblMAJOR WHERE MajorName = 'Computer
Science')), ((SELECT UniversityID FROM tblUNIVERSITY WHERE UniversityName = 'Seattle
University'), (SELECT MajorID FROM tblMAJOR WHERE MajorName = 'Biology')),
((SELECT UniversityID FROM tblUNIVERSITY WHERE UniversityName = 'Stanford'), (SELECT
MajorID FROM tblMAJOR WHERE MajorName = 'Communication')), ((SELECT UniversityID FROM
tblUNIVERSITY WHERE UniversityName = 'University of Florida'), (SELECT MajorID FROM
tblMAJOR WHERE MajorName = 'Accounting')), ((SELECT UniversityID FROM tblUNIVERSITY
WHERE UniversityName = 'Western Washington University'),
(SELECT MajorID FROM tblMAJOR WHERE MajorName = 'Finance'))
```

Stored Procedures

```
CREATE PROCEDURE addNewCompany
@CompName varchar(30),
@CompTypeName varchar(30),
@CompDate DATE,
@LocName varchar(50),
@LocCity varchar(50),
@LocState varchar(50),
@LocZip char(10)

AS

DECLARE @L_ID INT
DECLARE @CT_ID INT

SET @CT_ID = (SELECT Company_TypeID
FROM tblCOMPANY_TYPE
WHERE Company_TypeName = @CompTypeName)

SET @L_ID = (SELECT LocationID
FROM tblLOCATION
WHERE LocationName = @LocName
AND LocationCity = @LocCity
AND LocationState = @LocState
AND LocationZip = @LocZip)
BEGIN TRAN T1
INSERT INTO tblCOMPANY (CompanyID, Company_TypeID, DateFounded, LocationID)
VALUES (@CompName, @CT_ID, @CompDate, @L_ID)
COMMIT TRAN T1
GO



CREATE PROCEDURE addNewUniversity
@UniName varchar(50),
@UniDescr varchar(50),
@UniTypeName varchar(50),
@UniTypeDescr varchar(50),
@LocName varchar(50),
```

```
@LocDescr varchar(50),
@LocCity varchar(50),
@LocState varchar(50),
@LocZip char(10)

AS

DECLARE @L_ID INT
DECLARE @UT_ID INT

SET @L_ID = (SELECT LocationID
FROM tblLOCATION
WHERE LocationName = @LocName
AND LocationDescr = @LocDescr
AND LocationCity = @LocCity
AND LocationState = @LocState
AND LocationZip = @LocZip)

SET @UT_ID = (SELECT University_TypeID
FROM tblUniversity_Type
WHERE University_TypeName = @UniTypeName
AND University_TypeDescr = @UniTypeDescr)

BEGIN TRAN T2
INSERT INTO tblUNIVERSITY (University_TypeID, UniversityName, UniversityDescr,
LocationID)
VALUES (@UT_ID, @L_ID, @UniName, @UniDescr)
COMMIT TRAN T2
```

Business Rule

'Individuals younger than 16 can't be a student'

```
CREATE FUNCTION test16Oldstudent()
RETURNS INT
AS
BEGIN

DECLARE @RET INT = 0
```

```
IF EXISTS (
   SELECT *
   FROM tblSTUDENT S
   WHERE S.DateOfBirth > (SELECT (GetDate() - 365.25 * 16))
)
BEGIN
   SET @RET = 1
END
RETURN @RET
END
GO

ALTER TABLE tblSTUDENT
ADD CONSTRAINT ck_noTooYoung16Student
CHECK (dbo.test16Oldstudent() = 0)

GO
```

Computed Column

'Measure the total number of students who started working in a public company in the past five years'

```
CREATE FUNCTION fn_PublicEmployees (@PK INT)
RETURNS INT
AS
BEGIN

DECLARE @RET INT = (
    SELECT COUNT (*)
    FROM tblSTUDENT S
        JOIN tblSTUDENT_UNIVERSITY_MAJOR STUM ON S.StudentID = STUM.StudentID
        JOIN tblUNIVERSITY_MAJOR UM ON UM.University_MajorID = STUM.University_MajorID
        JOIN tblJOB J ON J.Student_University_MajorID =
STUM.Student_University_MajorID
        JOIN tblUNIVERSITY U ON U.UniversityID = UM.UniversityID
```

```
        JOIN tblLOCATION L ON L.LocationID = U.LocationID
        JOIN tblCOMPANY C ON C.LocationID = L.LocationID
        JOIN tblCOMPANY_TYPE CT ON CT.Company_TypeID = C.Company_TypeID
    WHERE S.StudentID = @PK
    AND CT.Company_TypeName = 'Public'
    AND J.BeginDate > (SELECT (GetDate() - 365.25 * 5))
)
RETURN @RET
END
GO
ALTER TABLE tblSTUDENT
ADD PublicFiveYearStudents AS (dbo.fn_PublicEmployees(StudentID))
```

Complex Query

```
--'Query the oldest student born who have ended at least two jobs in a public
company?'

SELECT TOP 1 with ties S.StudentID, S.StudentFirstName, S.StudentLastName,
S.DateOfBirth, COUNT(J.JobID) AS NumberOfPrivateJobs
FROM tblSTUDENT S
       JOIN tblSTUDENT_UNIVERSITY_MAJOR STUM ON S.StudentID = STUM.StudentID
       JOIN tblUNIVERSITY_MAJOR UM ON UM.University_MajorID = STUM.University_MajorID
       JOIN tblJOB J ON J.Student_University_MajorID = STUM.Student_University_MajorID
       JOIN tblUNIVERSITY U ON U.UniversityID = UM.UniversityID
       JOIN tblLOCATION L ON L.LocationID = U.LocationID
       JOIN tblCOMPANY C ON C.LocationID = L.LocationID
       JOIN tblCOMPANY_TYPE CT ON CT.Company_TypeID = C.Company_TypeID
WHERE CT.Company_TypeName = 'Public'
GROUP BY S.StudentID, S.StudentFirstName, S.StudentLastName, S.DateOfBirth
HAVING COUNT(J.JobID) >= 2
```

```
ORDER BY S.DateOfBirth DESC
```

Code for adding Student_University_Major

```
EXEC AddStudent_University_Major 'Data Science', 'University Of Washington',
'Lucas', 'Woo', '1996-03-25', 'dsa@ww.com', 'University District',
'Seattle', 'Washington'

EXEC AddStudent_University_Major 'Computer Science', 'University Of Washington',
'Andy', 'Chen', '1998-08-25', 'chhdjsa@sds.com', 'University District',
'Seattle', 'Washington'

EXEC AddStudent_University_Major 'Biology', 'Seattle University',
'kk', 'Smith', '1991-01-25', 'kksina@gmail.com', 'Queen Anne',
'Seattle', 'Washington'


EXEC AddStudent_University_Major 'Finance', 'Western Washington University',
'Grace', 'Mark', '2002-04-29', 'gmark@uw.edu', 'Bellingham',
'Seattle', 'Washington'


EXEC AddStudent_University_Major 'Biology', 'Seattle University',
'Kelly', 'James', '1998-01-26', 'ddsadsa@gmail.com', 'Queen Anne',
'Seattle', 'Washington'
```

Yu Che Lin

Creating Tables

```
CREATE TABLE tblJOB_TYPE
(Job_TypeID INTEGER IDENTITY(1,1) primary key,
Job_TypeName varchar(50) not null,
Job_TypeDescr varchar (500) null)
GO

CREATE TABLE tblPOSITION
(PositionID INTEGER IDENTITY(1,1) primary key not null,
PositionName varchar(50) not null,
PositionDescr varchar(500) null,
Salary numeric(9,2) not null)
GO

CREATE TABLE tblJOB
(JobID INTEGER IDENTITY(1,1) primary key not null,
Job_TypeID INTEGER FOREIGN KEY REFERENCES tblJOB_TYPE (Job_TypeID) not null,
Student_University_MajorID INTEGER FOREIGN KEY REFERENCES
tblSTUDENT_UNIVERSITY_MAJOR (Student_University_MajorID) not null,
PositionID INTEGER FOREIGN KEY REFERENCES tblPOSITION (PositionID) not null,
CompanyID INTEGER FOREIGN KEY REFERENCES tblCOMPANY (CompanyID) not null,
BeginDate Date not null,
EndDate Date null)
GO
```

Stored Procedures

```
CREATE PROCEDURE AddStudent
@Fname varchar(50),
@Lname varchar(50),
@Bdate Date,
@Gdate Date null,
@City varchar(50),
@State varchar(50),
@Zip varchar(25),
@Phone varchar(15),
@Email varchar(80),
@Gender char(1),
@Gpa decimal(3,2) null

AS

BEGIN TRAN Y1
INSERT INTO tblSTUDENT (StudentFirstName, StudentLastName, DateofBirth, DateofGraduate,
PermCity, PermState, PermZip, PhoneNumber, EmailAddress, Gender, AverageGPA)
VALUES (@Fname, @Lname, @Bdate, @Gdate, @City, @State, @Zip, @Phone, @Email,
@Gender, @Gpa)
COMMIT TRAN Y1

CREATE PROCEDURE AddMajor
@Mname varchar(50),
@Mdesc varchar(500),
@Mtname varchar(50)

AS

DECLARE @MT_ID INT

SET @MT_ID = (SELECT Major_TypeID
FROM tblMAJOR_TYPE
WHERE Major_TypeName = @Mtname)

BEGIN TRAN Y2
INSERT INTO tblMAJOR (Major_TypeID, MajorName, MajorDescr)
VALUES (@MT_ID, @Mname, @Mdesc)
COMMIT TRAN Y2
```

Business Rule

A full-time 'Software Developer' cannot have less than salary of 20000

```
CREATE FUNCTION fn_SoftDevMinSalary20k()
RETURNS INT
```

```
AS
BEGIN

DECLARE @RET INT = 0
IF EXISTS (
	SELECT *
	FROM tblPOSITION P
		JOIN tblJOB J ON J.PositionID = P.PositionID
		JOIN tblJOB_TYPE JT ON JT.Job_TypeID = J.Job_TypeID
	WHERE P.PositionName = 'Software Developer'
	AND J.Salary < 20000
	AND JT.Job_TypeName = 'Full Time'
)
BEGIN
	SET @RET = 1
END
RETURN @RET
END
GO

ALTER TABLE tblJOB
ADD CONSTRAINT CK_SoftDevMinSalary
CHECK (dbo.fn_SoftDevMinSalary20k() = 0)
```

Computed Column

Average salary of part-time 'database developer' for each company

```
CREATE FUNCTION fn_DbDevAvgSalary(@PK INT)
RETURNS INT
AS
BEGIN

DECLARE @RET INT = (
	SELECT AVG(J.Salary)
	FROM tblPOSITION P
		JOIN tblJOB J ON J.PositionID = P.PositionID
		JOIN tblCOMPANY C ON C.CompanyID = J.CompanyID
		JOIN tblJOB_TYPE JT ON JT.Job_TypeID = J.Job_TypeID
	WHERE C.CompanyID = @PK
	AND JT.Job_TypeName = 'Part Time'
	AND P.PositionName = 'Database Developer'
)
RETURN @RET
END
```

```
GO
ALTER TABLE tblCOMPANY
ADD DatabaseDevSalary AS (dbo.fn_DbDevAvgSalary(CompanyID))
```

Complex Query

Query companies in Seattle, Washington that has more than 80k average paying salary for full-time 'chemical engineer'

```
SELECT C.CompanyID, C.Company_Name, AVG(J.Salary) AS ChemEngineerSalary
FROM tblPOSITION P
        JOIN tblJOB J ON J.PositionID = P.PositionID
        JOIN tblCOMPANY C ON C.CompanyID = J.CompanyID
        JOIN tblJOB_TYPE JT ON JT.Job_TypeID = J.Job_TypeID
WHERE JT.Job_TypeName = 'Full Time'
        AND P.PositionName = 'Chemical Engineer'
        AND C.CompanyID IN (SELECT C.CompanyID
                FROM tblLOCATION L
                        JOIN tblCOMPANY C ON C.LocationID = L.LocationID
                WHERE L.LocationCity = 'Seattle'
                        AND L.LocationState = 'Washington'
                GROUP BY C.CompanyID)
GROUP BY C.CompanyID, C.Company_Name
HAVING AVG(J.Salary) > 80000
ORDER BY AVG(J.Salary) DESC
```

Ryan Dang

Creating Tables

```
CREATE TABLE tblUNIVERSITY
(UniversityID INT IDENTITY(1,1) primary key,
University_TypeID INT FOREIGN KEY REFERENCES tblUNIVERSITY_TYPE (University_TypeID)
NOT NULL,
UniversityName varchar(50) NOT NULL,
UniversityDescr varchar(500) NULL,
LocationID INT FOREIGN KEY REFERENCES tblLOCATION (LocationID) NOT NULL)
GO

INSERT INTO tblUNIVERSITY (UniversityName, UniversityDescr, LocationID)
```

VALUES('**University of Washington**', 'The UW is one of the world's preeminent public universities. Our impact on individuals, our region and the world is profound — whether we are launching young people into a boundless future or confronting the grand challenges of our time through undaunted research and scholarship.', (SELECT LocationID FROM tblLOCATION),
('**Seattle University**', 'Seattle University, founded in 1891, is a Jesuit Catholic university located on 50 acres in Seattle's Capitol Hill neighborhood. More than 7,200 students are enrolled in undergraduate and graduate programs within eight schools and colleges.', (SELECT LocationID FROM tblLOCATION)),
('**University of Florida**', 'At the University of Florida, we are a people of purpose. We're committed to challenging convention and ourselves. We see things not as they are, but as they could be. And we strive for a greater impact: one measured in people helped and lives improved.', (SELECT LocationID FROM tblLOCATION)),
('**Stanford**', 'Stanford University is one of the world's leading teaching and research universities. Since its opening in 1891, Stanford has been dedicated to finding solutions to big challenges and to preparing students for leadership in a complex world.', (SELECT LocationID FROM tblLOCATION)),
('**Western Washington University**', 'Western is an energized campus community with nearly 16,000 students and over 160 academic programs. Western offers the focus on students and the faculty access of a smaller college with the academic choice, opportunities for research, multicultural diversity, and room to grow of a large university.', (SELECT LocationID FROM tblLOCATION))

```
CREATE TABLE tblUNIVERSITY_TYPE
(University_TypeID INT IDENTITY(1,1) primary key,
University_TypeName varchar(50) NOT NULL,
University_TypeDescr varchar(500) NULL)
GO

INSERT INTO tblUNIVERSITY_TYPE (University_TypeName, University_TypeDescr)
VALUES('Public', 'Public, Research, Sea-grant, Space-grant'),
('Private', 'Private, Jesuit, Nonprofit'),
('Public', 'Public, Research, Land-grant, Sea-grant, Space-grant'),
('Private', 'Private, Research'),
('Public', 'Public')


CREATE TABLE tblLOCATION
(LocationID INT IDENTITY(1,1) primary key,
LocationName varchar(50) NOT NULL,
LocationDescr varchar(500) NULL,
LocationCity varchar(30) NOT NULL,
LocationState varchar(20) NOT NULL,
LocationZip varchar(20) NOT NULL)
GO

INSERT INTO tblLOCATION (LocationName, LocationDescr, LocationCity, LocationState,
LocationZip)
```

```
VALUES('University of Washington', 'public research university', 'Seattle', 'Washington', '98195'),
('Seattle University', 'private university', 'Seattle', 'Washington', '98122'),
('University of Florida', 'public research university', 'Gainesville', 'Florida', '32611'),
('Stanford', 'private research university', 'Stanford', 'California', '94305'),
('Western Washington University', 'public university', 'Bellingham', 'Washington', '98225')
```

Stored Procedures

```
CREATE PROCEDURE uspNewMajor
@MajName varchar(30),
@MajDescr varchar(500),
@MajTypeName varchar(30),
@MajTypeDescr varchar(500)

AS

DECLARE @MT_ID INT

SET @MT_ID INT = (SELECT Major_TypeID
FROM tblMAJOR_TYPE
WHERE Major_TypeName = @MajTypeName
AND Major_TypeDescr = @MajTypeDescr)

BEGIN TRAN R1
INSERT INTO tbl_MAJOR (Major_TypeID, MajorName, MajorDescr)
VALUES(@MT_ID, @MajName, @MajDescr)
COMMIT TRAN R1


CREATE PROCEDURE uspNewJob
@JobTypeName varchar(30),
@JobTypeDescr varchar(500),
@MajName varchar(30),
@MajDescr varchar(500),
@UniTypeName varchar(30),
@UniTypeDescr varchar(500),
@StuFirstName varchar(30),
@StuLastName varchar(30),
@PosName varchar(30),
@PosDescr varchar(500),
@Sal numeric(9,2),
@CompName varchar(30),
@DateFound date,
@BegDate date,
@EndDate date
```

```
AS

DECLARE @JT_ID INT
DECLARE @SUM_ID INT
DECLARE @UM_ID INT
DECLARE @U_ID INT
DECLARE @M_ID INT
DECLARE @S_ID INT
DECLARE @P_ID INT
DECLARE @C_ID INT

SET @SUM_ID INT = (SELECT Student_University_MajorID
FROM tblSTUDENT_UNIVERSITY_MAJOR)

SET @UM_ID INT = (SELECT University_MajorID
FROM tblUNIVERSITY_MAJOR)

SET @U_ID INT = (SELECT UniversityID
FROM tblUNIVERSITY
WHERE UniversityName = @UniName
AND UniversityDescr = @UniDescr)

SET @M_ID = (SELECT MajorID
FROM tblMAJOR
WHERE MajorName = @MajName
AND MajorDescr = @MajDescr)

SET @S_ID = (SELECT StudentID
FROM tblSTUDENT
WHERE StudentFirstName = @StuFirstName
AND StudentLastName = @StuLastName)

SET @JT_ID INT = (SELECT Job_TypeID
FROM tblJOB_TYPE
WHERE Job_TypeName = @JobTypeName
AND Job_TypeDescr = @JobTypeDescr)

SET @P_ID INT = (SELECT PositionID
FROM tblPOSITION
WHERE PositionName = @PosName
AND PositionDescr = @PosDescr
AND Salary = @Sal)

SET @C_ID INT = (SELECT CompanyID
```

```
FROM tblCOMPANY
WHERE CompanyName = @CompName
AND DateFounded = @DateFound)

BEGIN TRAN R2
INSERT INTO tbl_JOB (Job_TypeID, Student_University_MajorID, PositionID, CompanyID,
BeginDate, EndDate)
VALUES(@JT_ID, @SUM_ID, @P_ID, @C_ID, @BegDate, @EndDate)
COMMIT TRAN R2
```

Business Rule
A student from the University of Washington must have a minimum average GPA of 2.0.

```
CREATE FUNCTION fn_GpaMoreThanTwo()
RETURNS INT
AS
BEGIN

DECLARE @Ret INT = 0

IF EXISTS (SELECT *
FROM tblSTUDENT S
        JOIN tblSTUDENT_UNIVERSITY_MAJOR STUM ON S.StudentID = STUM.StudentID
        JOIN tblUNIVERSITY_MAJOR UM ON STUM.Student_University_MajorID =
UM.Student_University_MajorID
        JOIN tblUNIVERSITY U ON UM.University_MajorID = U.University_MajorID
WHERE S.AverageGPA > 2
AND U.UniversityName = 'University of Washington')

BEGIN
SET @Ret = 1
END

RETURN @Ret
END
GO

ALTER TABLE tblSTUDENT_UNIVERSITY_MAJOR
ADD CONSTRAINT CK_GPAHigherThan2
CHECK (dbo.fn_GpaMoreThanTwo() = 0)
GO
```

Computed Column
Count the total number of students working at Amazon who have also graduated with a major in Computer Science.

```
CREATE FUNCTION fn_CSAmazon(@PK INT)
RETURNS INT
AS
BEGIN

DECLARE @Ret INT = (
SELECT COUNT(*)
FROM tblSTUDENT S
        JOIN tblSTUDENT_UNIVERSITY_MAJOR STUM ON S.StudentID = STUM.StudentID
        JOIN tblUNIVERSITY_MAJOR UM ON STUM.Student_University_MajorID =
UM.Student_University_MajorID
        JOIN tblMAJOR M on UM.University_MajorID = M.UniversityMajorID
        JOIN tblJOB J ON STUM.Student_University_MajorID = J.Student_University_MajorID
        JOIN tblCOMPANY C ON J.JobID = C.JobID
WHERE M.MajorName = Computer Science
AND C.CompanyName = 'Amazon'
AND S.StudentID = @PK)

RETURN @Ret
END
GO

ALTER TABLE tblSTUDENT
ADD TotalCSAmazon AS (dbo.fn_CSAmazon(StudentID))
```

Complex Query

Query students born before February 26, 1998 residing in California state that graduated with a GPA of more than 3.0 who have majored in Electrical Engineering who also work full time at a company founded after 2000.

```
SELECT S.StudentID, S.StudentFirstName, S.StudentLastName, STUM(S.AverageGPA) AS
GoodGPA
FROM tblSTUDENT S
        JOIN tblSTUDENT_UNIVERSITY_MAJOR STUM ON S.StudentID = STUM.StudentID
        JOIN tblUNIVERSITY_MAJOR UM ON STUM.Student_University_MajorID =
UM.Student_University_MajorID
        JOIN tblMAJOR M ON UM.University_MajorID = M.University_Major
        JOIN tblJOB J ON STUM.Student_University_MajorID = J.Student_University_MajorID
        JOIN tblJOB_TYPE JT ON J.JobID = JT.Job_ID
        JOIN tblCOMPANY C ON J.JobID = C.JobID
WHERE S.DateOfBirth < 'February 26, 1998'
AND S.PermState = 'California'
AND M.MajorName = 'Electrical Engineering'
AND JT.Job_TypeName = 'Full Time'
```

```
AND C.DateFounded > 2000
GROUP BY S.StudentID, S.StudentFirstName, S.StudentLastName
HAVING STUM(S.AverageGPA) >= 3
```

Yueh-Chi Lin

Creating Tables

```
CREATE TABLE tblSTUDENT_UNIVERSITY_MAJOR
(Student_University_MajorID INT IDENTITY(1,1) primary key,
University_MajorID INT FOREIGN KEY REFERENCES tblUNIVERSITY_MAJOR
(University_MajorID) NOT NULL,
StudentID INT FOREIGN KEY REFERENCES tblSTUDENT (StudentID) NOT NULL)
GO

CREATE TABLE tblStudent
(StudentID INT IDENTITY(1,1) primary key,
StudentFirstName varchar(30) not NULL,
StudentLastName varchar(30) not NULL,
DateOfBirth Date not NULL,
DateOfGraduate Date not NULL,
PermCity varchar(30) not NULL,
PermState varchar(30) not NULL,
PermZip varchar(30) not NULL,
PhoneNumber varchar(20) not NULL,
EmailAddress varchar(50) not NULL,
Gender varchar(20) not NULL,
AverageGPA numeric(2,2) is NULL)
GO

CREATE TABLE tblSTUDENT_UNIVERSITY_MAJOR_STATUS
(Sudent_University_Major_StatusID INT IDENTITY(1,1) primary key,
Student_University_MajorID INT FOREIGN KEY REFERENCES
tblSTUDENT_UNIVERSITY_MAJOR (Student_University_MajorID) NOT NULL,
StatusID INT FOREIGN KEY REFERENCES tblSTATUS (StatusID) NOT NULL
)
GO
```

Stored Procedure

```
CREATE PROCEDURE uspAddLocation
@LocationName varchar(30),
@LocationDescr varchar(30),
```

```
@LocationCity varchar(30),
@LocationState varchar(30),
@LocationZip varchar(25)
AS

BEGIN TRAN T1
INSERT INTO tblLOCATION (LocationName, LocationDescr, LocationCity, LocationState,
LocationZip)
VALUES(@LocationName, @LocationDescr, @LocationCity, @LocationState, @LocationZip)
COMMIT TRAN T1

CREATE PROCEDURE uspNewStatus
@StatusName varchar(30),
@StatusDescr varchar(30),
@BeginDate Date,
@EndDate Date
AS

BEGIN TRAN T1
INSERT INTO tblSTATUS (StatusName, StatusDescr, BeginDate, EndDate)
VALUES(@StatusName, @StatusDescr, @BeginDate, @EndDate)
COMMIT TRAN T1
```

Business Rule
No students under 12 years old from California that can have major of ‘Computer Science’

```
CREATE FUNCTION fn_No12CACSStudents()
RETURNS INT
AS
BEGIN
DECLARE @RET INT = 0
IF EXISTS (
        SELECT *
        FROM tblSTUDENT S
                JOIN tblSTUDENT_UNIVERSITY_MAJOR SUM ON S.StudentID = SUM.StudentID
                JOIN tblUNIVERSITY_MAJOR UM ON SUM.Student_University_MajorID =
                UM.Student_University_MajorID
                JOIN tblMAJOR M on UM.University_MajorID = M.UniversityMajorID
        WHERE S.DateOfBirth > (SELECT GetDate() - (365.25 * 12))
        AND S.PermState = ‘California, CA’
        ANd M.MajorName = ‘Computer Science’)
)
BEGIN
        SET @RET = 1
END
```

```
RETURN @RET
END
GO

ALTER TABLE tblSTUDENT
ADD CONSTRAINT CK_No12CACSStudents
CHECK (dbo.fn_No12CACSStudents() = 0)
```

Computed Column

Total number of major a student have

```
CREATE FUNCTION fn_TotalNumOfMajor(@PK INT)
RETURNS INT
AS
BEGIN
DECLARE @RET INT = (
        SELECT Total(UM.University_MajorID)
        FROM tblSTUDENT S
                JOIN tblSTUDENT_UNIVERSITY_MAJOR SUM ON S.StudentID = SUM.StudentID
        WHERE S.StudentID = @PK
)
RETURN @RET
END
GO
ALTER TABLE tblSTUDENT
ADD TotalNumOfMajor AS (dbo.fn_TotalNumOfMajor(StudentID))
```

Complex Query

Select top 5 students from California that has GPA of 4.0 and a job with average salary more than 50000

```
SELECT TOP 5 S.StudentFirstName, S.StudentLastName, S.AverageGPA, AVG(P.Salary)
FROM tblSTUDENT S
        JOIN tblSTUDENT_UNIVERSITY_MAJOR SUM ON S.StudentID = SUM.StudentID
        JOIN tblJOB J ON J.Student_University_MajorID = P.Student_University_MajorID
        JOIN tblPOSITION P ON J.PositionID = P.PositionID
WHERE S.AverageGPA=4.0
AND S.PermState = 'California, CA'
GROUP BY S.StudentFirstName, S.StudentLastName, S.AverageGPA
HAVING AVG(P.Salary) > 50000
ORDER BY AVG(P.Salary) DESC
```

He Chen

Creating Tables

```
CREATE TABLE tblCOMPANY_TYPE
(Company_TypeID INTEGER IDENTITY(1,1) primary key,
Company_TypeName varchar(50) not null,
Company_TypeDescr varchar (500) NULL)
GO

CREATE TABLE tblStatus
(StatusID INTEGER IDENTITY(1,1) primary key,
StatusName varchar(50) not null,
StatusDescr varchar (500) NULL)
GO

CREATE TABLE tblCompany
(CompanyID INTEGER IDENTITY(1,1) primary key NOT NULL,
Company_TypeID INTEGER FOREIGN KEY REFERENCES tblCOMPANY_TYPE (Company_TypeID) NOT
NULL,
Company_Name varchar(50) not null,
Company_Descr varchar(500) not null,
DateFounded DATE not null,
LocationID INTEGER FOREIGN KEY REFERENCES tblLOCATION (LocationID) NOT NULL)
GO
```

Stored Procedure

```
-- INSERT INTO Student_University_Major
CREATE PROCEDURE AddStudent_University_Major
@MajorName varchar(30),
@UniversityName varchar(30),
```

```
@SFName varchar(30),
@SLName varchar(30),
@DateofBirth Date,
@EmailAddress varchar(50),
@LocName varchar(30),
@LocCity varchar(30),
@LocState varchar(30)

AS

DECLARE @Loc_ID INT
DECLARE @M_ID INT
DECLARE @U_ID INT
DECLARE @S_ID INT
DECLARE @Uni_M_ID INT



SET @Loc_ID = (SELECT LocationID
FROM tblLOCATION
WHERE LocationName = @LocName
AND LocationCity = @LocCity
AND LocationState = @LocState)

SET @S_ID = (SELECT s.StudentID
FROM tblStudent s
WHERE s.StudentFirstName = @SFName
AND s.StudentLastName = @SLName
AND s.DateOfBirth = @DateofBirth
AND s.EmailAddress = @EmailAddress)

SET @M_ID = (SELECT MajorID
FROM tblMAJOR m
WHERE m.MajorName = @MajorName)

SET @U_ID = (SELECT u.UniversityID
FROM tblUNIVERSITY u
WHERE u.UniversityName = @UniversityName
AND u.LocationID = @Loc_ID)
```

```
SET @Uni_M_ID = (SELECT um.University_MajorID
FROM tblUNIVERSITY_MAJOR um
WHERE um.UniversityID = @U_ID
AND um.MajorID = @M_ID)


BEGIN TRAN T1
INSERT INTO tblSTUDENT_UNIVERSITY_MAJOR (University_MajorID, StudentID)
VALUES(@Uni_M_ID, @S_ID)
COMMIT TRAN T1


-- Add major status to student
CREATE PROCEDURE AddStudent_University_Major_Status
@MajorName varchar(30),
@UniversityName varchar(30),
@SFName varchar(30),
@SLName varchar(30),
@DateofBirth Date,
@EmailAddress varchar(50),
@LocName varchar(30),
@LocCity varchar(30),
@LocState varchar(30),
@StatusName varchar(30),
@BeginDate Date

AS

DECLARE @Loc_ID INT
DECLARE @M_ID INT
DECLARE @U_ID INT
DECLARE @S_ID INT
DECLARE @Uni_M_ID INT
DECLARE @Uni_M_S_ID INT
DECLARE @StatusID INT
```

```
SET @Loc_ID = (SELECT LocationID
FROM tblLOCATION
WHERE LocationName = @LocName
AND LocationCity = @LocCity
AND LocationState = @LocState)

SET @S_ID = (SELECT s.StudentID
FROM tblStudent s
WHERE s.StudentFirstName = @SFName
AND s.StudentLastName = @SLName
AND s.DateOfBirth = @DateofBirth
AND s.EmailAddress = @EmailAddress)

SET @M_ID = (SELECT MajorID
FROM tblMAJOR m
WHERE m.MajorName = @MajorName)

SET @U_ID = (SELECT u.UniversityID
FROM tblUNIVERSITY u
WHERE u.UniversityName = @UniversityName
AND u.LocationID = @Loc_ID)

SET @Uni_M_ID = (SELECT um.University_MajorID
FROM tblUNIVERSITY_MAJOR um
WHERE um.UniversityID = @U_ID
AND um.MajorID = @M_ID)

SET @StatusID = (SELECT s.StatusID
FROM tblStatus s
WHERE s.StatusName = @StatusName)

SET @Uni_M_S_ID = (SELECT stum.Student_University_MajorID
FROM  tblSTUDENT_UNIVERSITY_MAJOR stum
WHERE stum.University_MajorID = @Uni_M_ID
AND stum.StudentID = @S_ID)

BEGIN TRAN T1
```

```
INSERT INTO tblSTUDENT_UNIVERSITY_MAJOR_STATUS (Student_University_MajorID, StatusID,
BeginDate, EndDate)
VALUES(@Uni_M_S_ID, @StatusID, @BeginDate, NULL)
COMMIT TRAN T1

-- Add to major to university
CREATE PROCEDURE AddUniversity_Major
@MajorName varchar(30),
@UniversityName varchar(30),
@LocName varchar(30),
@LocCity varchar(30),
@LocState varchar(30)

AS

DECLARE @Loc_ID INT
DECLARE @M_ID INT
DECLARE @U_ID INT

SET @Loc_ID = (SELECT LocationID
FROM tblLOCATION
WHERE LocationName = @LocName
AND LocationCity = @LocCity
AND LocationState = @LocState)

SET @M_ID = (SELECT MajorID
FROM tblMAJOR m
WHERE m.MajorName = @MajorName)

SET @U_ID = (SELECT u.UniversityID
FROM tblUNIVERSITY u
WHERE u.UniversityName = @UniversityName
AND u.LocationID = @Loc_ID)

BEGIN TRAN T1
INSERT INTO tblUNIVERSITY_MAJOR (UniversityID, MajorID)
VALUES(@U_ID, @M_ID)
COMMIT TRAN T1
```

Business Rule

Full time student can't take full time job

```
CREATE FUNCTION fn_NoFullTimeJobForFullTimeStudent()
RETURNS INT
AS
BEGIN

DECLARE @RET INT = 0
IF EXISTS (
    SELECT *
    FROM tblJOB j
        JOIN tblJOB_TYPE jt ON jt.Job_TypeID = j.Job_TypeID
       JOIN tblSTUDENT_UNIVERSITY_MAJOR tblsum ON tblsum.Student_University_MajorID =
j.Student_University_MajorID
       JOIN tblSTUDENT_UNIVERSITY_MAJOR_STATUS tblsums ON
tblsums.Student_University_MajorID = tblsum.Student_University_MajorID
       JOIN tblStatus s ON s.StatusID = tblsums.StatusID
   WHERE s.StatusName = 'Full Time'
   AND jt.Job_TypeName = 'Full Time'
   AND ((j.BeginDate >= tblsums.BeginDate)  AND ((j.BeginDate <= tblsums.EndDate) OR
(tblsums.EndDate = NULL)))
)
BEGIN
    SET @RET = 1
END
RETURN @RET
END
GO

ALTER TABLE tblJOB
ADD CONSTRAINT CK_FullTimeJobs
CHECK (dbo.fn_NoFullTimeJobForFullTimeStudent() = 0)
GO
```

Computed Column

```
CREATE FUNCTION fn_countSTEM (@PK INT)
RETURNS INT
AS
BEGIN

DECLARE @RET INT = (
    SELECT COUNT (*)
    FROM tblSTUDENT S
        JOIN tblSTUDENT_UNIVERSITY_MAJOR tblsum ON S.StudentID = tblsum.StudentID
        JOIN tblUNIVERSITY_MAJOR um ON um.University_MajorID =
tblsum.University_MajorID
        JOIN tblUNIVERSITY u ON u.UniversityID = um.UniversityID
       JOIN tblMAJOR m ON m.MajorID = um.MajorID
       JOIN tblSTUDENT_UNIVERSITY_MAJOR_STATUS tblsums ON
tblsums.Student_University_MajorID = tblsum.Student_University_MajorID
       JOIN tblMAJOR_TYPE mt ON mt.Major_TypeID = m.Major_TypeID
    WHERE u.UniversityID = @PK
    AND mt.Major_TypeName = 'STEM'
    AND tblsums.EndDate = NULL
)
RETURN @RET
END
GO
ALTER TABLE tblUNIVERSITY
ADD CurrentSTEMStudents AS (dbo.fn_countSTEM(UniversityID))
GO

SELECT *
FROM tblUNIVERSITY
```

Complex Query:

```
-- List all people who had at least 3 jobs after 2018-01-1 and started his academic
life on data science later than 2005 in order
SELECT s.StudentFirstName, s.StudentLastName, COUNT(j.JobID) AS total_jobs
FROM tblStudent s
```

```
JOIN tblSTUDENT_UNIVERSITY_MAJOR tblsum ON tblsum.StudentID = s.StudentID
JOIN tblSTUDENT_UNIVERSITY_MAJOR_STATUS sms ON sms.Student_University_MajorID =
tblsum.Student_University_MajorID
JOIN tblJOB j ON j.Student_University_MajorID = tblsum.Student_University_MajorID
JOIN tblJOB_TYPE jt ON jt.Job_TypeID = j.Job_TypeID
JOIN tblUNIVERSITY_MAJOR um ON um.University_MajorID = tblsum.University_MajorID
JOIN tblMAJOR m ON m.MajorID = um.MajorID
JOIN (
   SELECT s.StudentID
   FROM tblStudent s
   JOIN tblSTUDENT_UNIVERSITY_MAJOR tblsum ON tblsum.StudentID = s.StudentID
   JOIN tblSTUDENT_UNIVERSITY_MAJOR_STATUS sms ON sms.Student_University_MajorID =
tblsum.Student_University_MajorID
   JOIN tblJOB j ON j.Student_University_MajorID = tblsum.Student_University_MajorID
   JOIN tblJOB_TYPE jt ON jt.Job_TypeID = j.Job_TypeID
   JOIN tblUNIVERSITY_MAJOR um ON um.University_MajorID = tblsum.University_MajorID
   JOIN tblMAJOR m ON m.MajorID = um.MajorID
   WHERE m.MajorName = 'Data Science'
   AND sms.BeginDate > '2005-01-1'
) as sub ON sub.StudentID = s.StudentID
WHERE j.BeginDate > '2018-01-1'
GROUP BY s.StudentID, s.StudentFirstName, s.StudentLastName
HAVING COUNT(j.JobID) >= 3
ORDER BY total_jobs
```