Aleksandra Georgievska

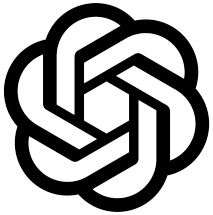
Project Lead

Project Management - Tools

Github

Azure Data Studio

SSMS

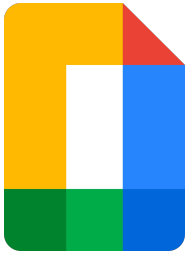


ChatGPT

Redgate

Discord

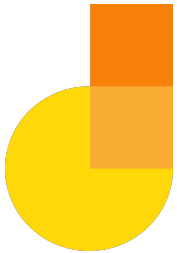
Google Docs



Google Drive

Google Sheets

Google Slides



Google Jamboard

Davinci Resolve

Project Management - Life Cycle

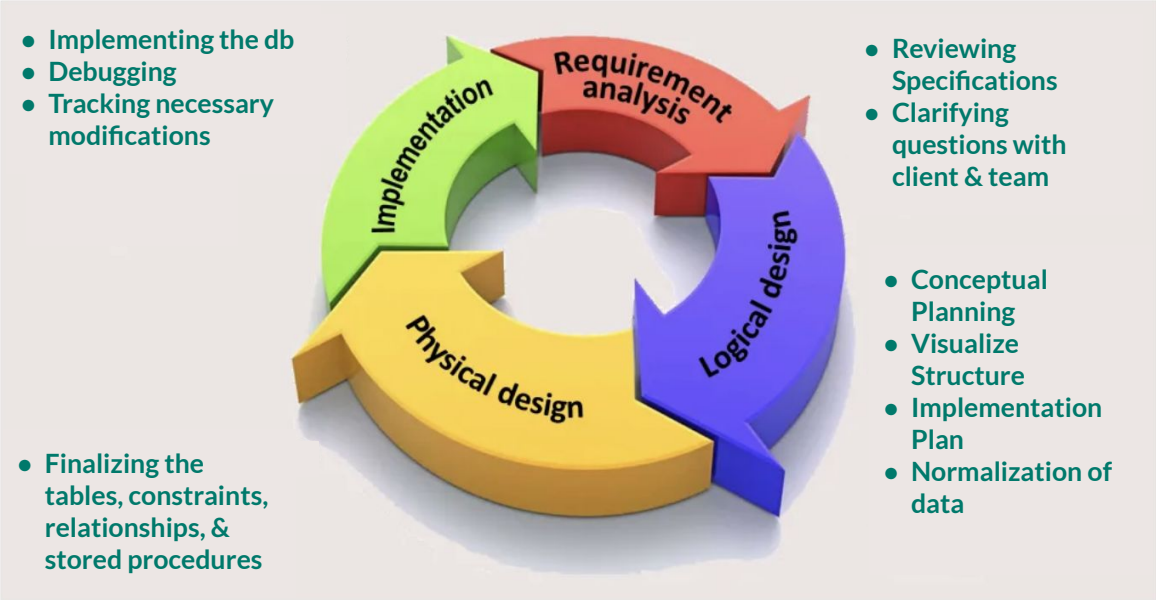


Figure 1 Database life cycle

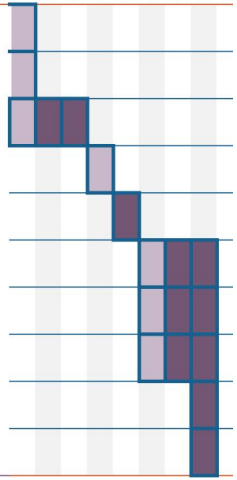
Project Management - Team Planning

Project Planner

Select a period to highlight at right. A legend describing the charting follows.

Period Highlight: | Plan Duration

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Review Project Specs	1	1	1	1	100%
Have first meeting to review specs/Understand project	1	1	1	1	100%
Make list of table planning and normalization ideas	1	3	1	3	100%
2nd Meeting to plan distribution of work	4	1	4	1	100%
TL make project template & distribute to team	5	1	5	1	100%
Individual code portion	6	3	6	3	100%
Merge different branches of code contributions	6	3	6	3	100%
Verify changes work	6	3	6	3	100%
Record Presentations	9	1	9	1	100%
Consolidate Presetnations & Create submission package	9	1	9	1	100%



Project Management - Personal Planning

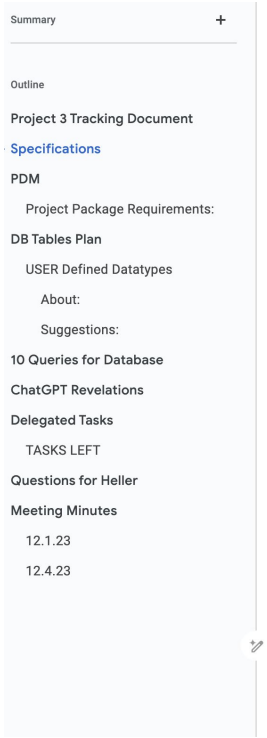
To-do list

To be completed by:		**Name Aleksandra Georgievska**	
Deadline:	**12/10/23**	**Date 12/1/23**	

Project 1

% done	Phase	Start By	Original Due By	Revised Due By	Number Of Days	Revision Notes
100%	Review Specs & Create plan for 1st Meeting	12/1/23	12/1/23		1	
100%	1st Team Meeting	12/1/23	12/1/23		1	
100%	Log Table/Normalization ideas	12/1/23	12/4/23		4	
100%	2nd Team Meeting	12/4/23	12/4/23		1	
100%	Create template of project 3 to distribute to team	12/4/23	12/4/23		1	
100%	Merge individual Scripts & Test code	12/4/23	12/8/23	9-Dec-23	4	granted extension
100%	Create Presentation materials	12/8/23	12/8/23	9-Dec-23	1	granted extension
100%	Create Submission Package	12/8/23	12/8/23	12/10/23	1	granted extension
100%	Submit	12/8/23	12/8/23	12/10/23	1	granted extension

Project Management - Project Planning



Project 3 Tracking Document

T/Th 9:15am - Group 1

DUE DATE: Thursday, December 7, 2023 11:59 PM EST

Team Lead: Aleksandra Georgievska
Project Lead Backup: Aryeh Richman
Team Members: Nicholas Kong, Aleksandra Georgievska, Aryeh Richman, Edwin Wray, Sigalita Yakubova, Ahnaf Ahmed

Quicklinks:
Project 2.5 Group Google Drive

Specifications

Project Requirements:

- Use of the swing app
  - One of the requirements is that once we finish creating the db and putting the data in he gives us 3 or 4 queries that we should be able to run on on the database (propositions) and then asks us to come up with 10 or more as a group (think the medium queries project 1):
    - 1. Show all instructors who are teaching in classes in multiple departments
    - 2. How many instructors are in each department?
    - 3. How many classes are being taught that semester grouped by course and aggregating the total enrollment, total class limit and the percentage of enrollment.
    - 4. 10 more queries of your choice and their proposition.
    - Load all 10 queries into the JDBC app
- Upload the final design your database as backup file (.bak)
- Use the schema Udt & create User Defined Datatypes where possible

Mp4 Presentation Requirements:

- Database:
- Project Lifecyle:

12.4.23

12:00 pm - 1:00 pm

Attendance: Nicholas Kong, Aleksandra Georgievska, Aryeh Richman, Sigalita Yakubova, Ahnaf Ahmed

Agenda

- [ ] Discuss project ideas
- [ ] Delegate tasks
- [ ] Set due dates

Meeting Minutes

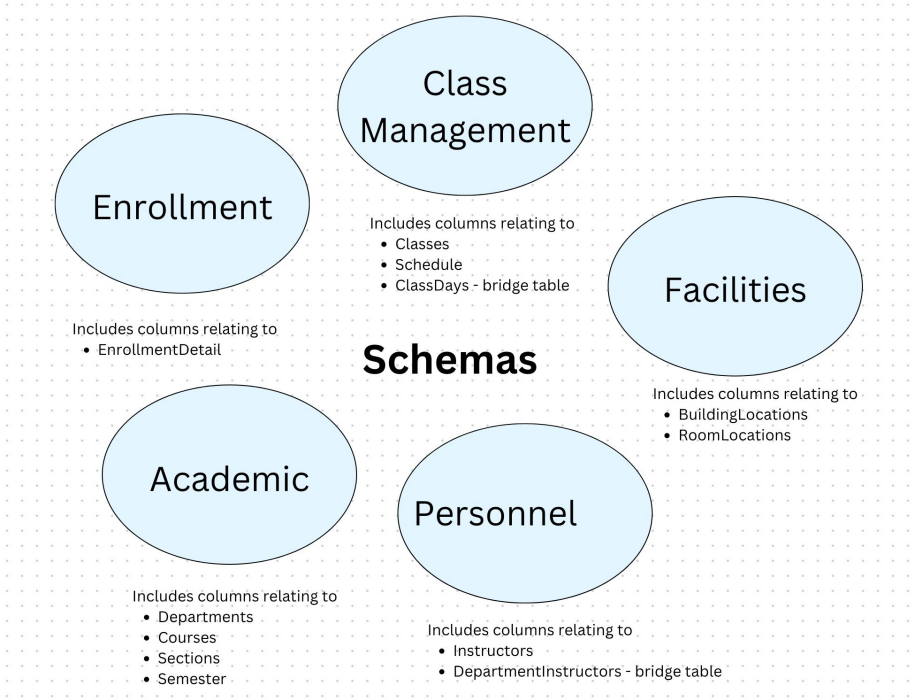
- Discussed tables plan
- Reviewed the UDT suggestions
  - Agreed upon **BuildingCode**, **TimeSlot**, **CreditHours**, **DayAbbreviations** Types
- ISSUE: Any table that has a foreign key the value of that foreign key depends on what the value of the primary key was, so the table of primary key needs to be defined first
- Any table that has a PK can be filled in first and then we move on to the tables with the foreign keys
  - Make a digraph to determine prereqs of the tables - Aryeh doing
- Aryeh demod the graph prereqs

Action Items

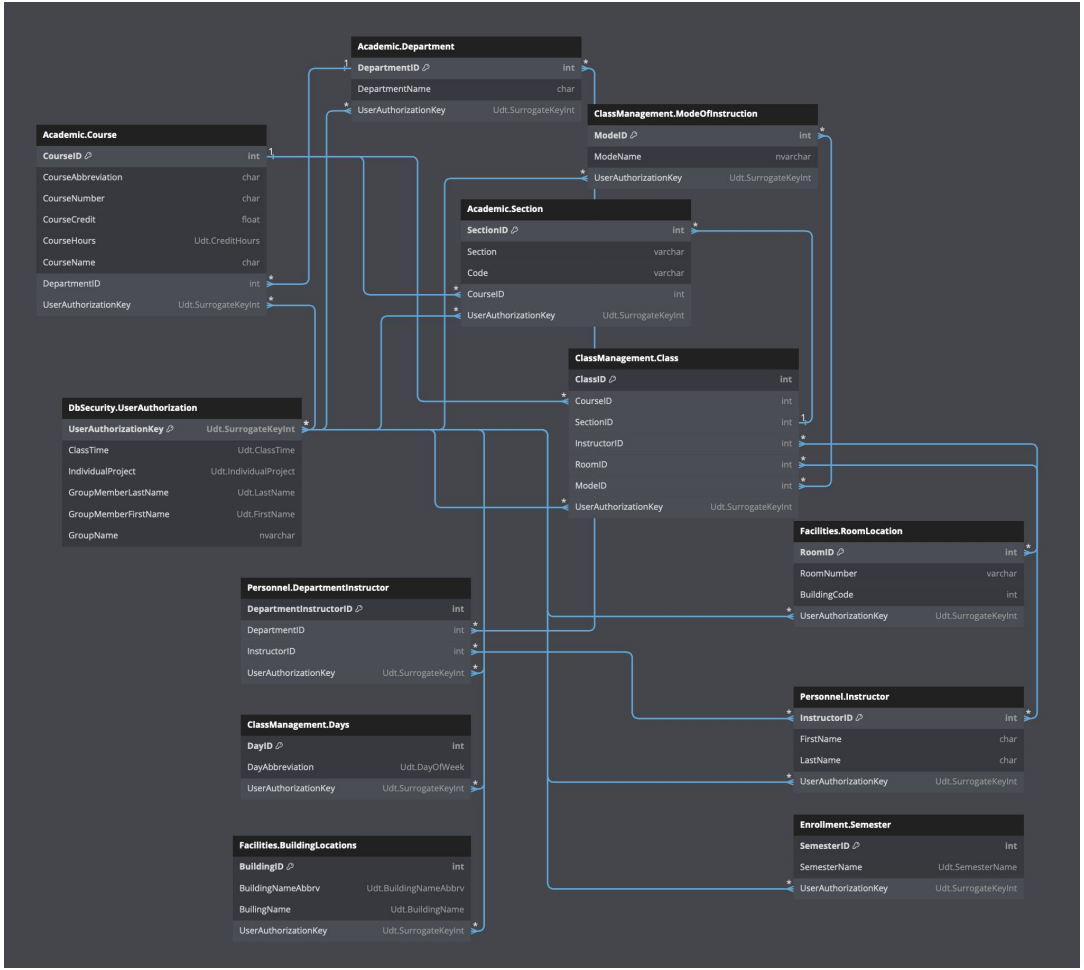
Due Tuesday night:

- Make the these tables
- make the constraints
- make the stored procedures
- let Aleks know when you're ready to merge your changes

Project Management - Project Planning



Physical Data Model



Project Management - Sharing & Version Control

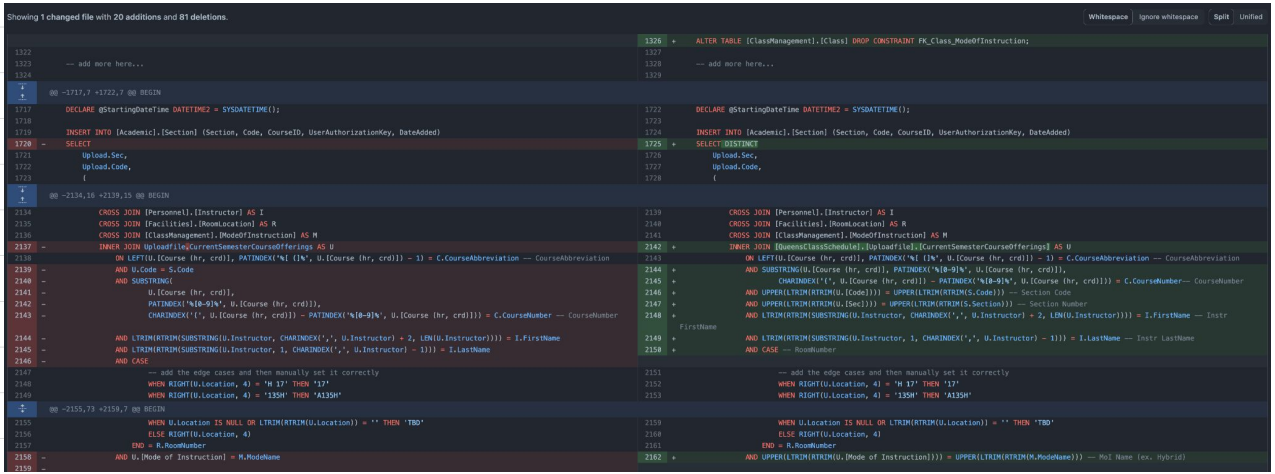
Computers › ··· › CS 331 Database Syste... › Project3

Type | People | Modified

74% of storage used You use storage when you save to Drive, back up to Google Photos, and send and r

Name ↑

- 915Group1_Project3_SubmissionFolder
- PowerPointSlides
- Recordings
- SQLFiles
- CollegeScheduleDatabaseDesign
- CollegeScheduleDatabaseDesign.txt
- How to create a Database Diagram in SSMS.pdf
- Project3 Tracking Document
- Screenshot 2023-12-04 at 2.45.31 PM.png
- T/TH 9:15 -Group 1 – Gantt Project3 planner.xlsx
- T/TH 9:15 -Group 1 – To-do list for Project3.xlsx



Code

```
USE [ClassSchedule_9:15_Group1];
GO

DROP SCHEMA IF EXISTS [Academic];
GO
CREATE SCHEMA [Academic];
GO

DROP SCHEMA IF EXISTS [Personnel];
GO
CREATE SCHEMA [Personnel];
GO

DROP SCHEMA IF EXISTS [ClassManagement];
GO
CREATE SCHEMA [ClassManagement];
GO

DROP SCHEMA IF EXISTS [Facilities];
GO
CREATE SCHEMA [Facilities];
GO

DROP SCHEMA IF EXISTS [Enrollment];
GO
CREATE SCHEMA [Enrollment];
GO

DROP SCHEMA IF EXISTS [DbSecurity];
GO
CREATE SCHEMA [DbSecurity];
GO
```

```
DROP SCHEMA IF EXISTS [Process];
GO
CREATE SCHEMA [Process];
GO

DROP SCHEMA IF EXISTS [PkSequence];
GO
CREATE SCHEMA [PkSequence];
GO

DROP SCHEMA IF EXISTS [Project3];
GO
CREATE SCHEMA [Project3];
GO

DROP SCHEMA IF EXISTS [G9_1];
GO
CREATE SCHEMA [G9_1];
GO

DROP SCHEMA IF EXISTS [Uploadfile]
GO
CREATE SCHEMA [Uploadfile]
GO

DROP SCHEMA IF EXISTS [Udt]
GO
CREATE SCHEMA [Udt]
GO
```

```
------------------------------------ Create User Defined Datatypes ---

--Aleks
CREATE TYPE [Udt].[DateAdded] FROM [datetime2] NOT NULL
GO
CREATE TYPE [Udt].[DateOfLastUpdate] FROM [datetime2] NOT NULL
GO
CREATE TYPE [Udt].[SurrogateKeyInt] FROM [int] NOT NULL
GO
CREATE TYPE [Udt].[ClassTime] FROM nchar(19) NOT NULL
GO
CREATE TYPE [Udt].[IndividualProject] FROM nvarchar (60) NOT NULL
GO
CREATE TYPE [Udt].[LastName] FROM  nvarchar(35) NOT NULL
GO
CREATE TYPE [Udt].[FirstName] FROM nvarchar(20) NOT NULL
GO
CREATE TYPE [Udt].[GroupName] FROM nvarchar(20) NOT NULL
GO
CREATE TYPE [Udt].[CreditHours] FROM [FLOAT] NOT NULL
GO
```

Code

```
177     CREATE TABLE [DbSecurity].[UserAuthorization]
178     (
179         -- all tables must have the following 3 columns:
180         [UserAuthorizationKey] [Udt].[SurrogateKeyInt] NOT NULL IDENTITY(1,1), -- primary key
181         [DateAdded] [Udt].[DateAdded] NOT NULL,
182         [DateOfLastUpdate] [Udt].[DateOfLastUpdate] NOT NULL,
183         --
184         [ClassTime] [Udt].[ClassTime] NULL,
185         [IndividualProject] [Udt].[IndividualProject] NULL,
186         [GroupMemberLastName] [Udt].[LastName] NOT NULL,
187         [GroupMemberFirstName] [Udt].[FirstName] NOT NULL,
188         [GroupName] [nvarchar](20) NOT NULL
189         PRIMARY KEY CLUSTERED
190     (
191             [UserAuthorizationKey] ASC
192     )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
193     ) ON [PRIMARY]
194     GO
```

```
632     ALTER TABLE [DbSecurity].[UserAuthorization] ADD  DEFAULT ('9:15') FOR [ClassTime]
633     GO
634     ALTER TABLE [DbSecurity].[UserAuthorization] ADD  DEFAULT ('PROJECT 3') FOR [IndividualProject]
635     GO
636     ALTER TABLE [DbSecurity].[UserAuthorization] ADD  DEFAULT ('GROUP 1') FOR [GroupName]
637     GO
638     ALTER TABLE [DbSecurity].[UserAuthorization] ADD  DEFAULT (sysdatetime()) FOR [DateAdded]
639     GO
640     ALTER TABLE [DbSecurity].[UserAuthorization] ADD  DEFAULT (sysdatetime()) FOR [DateOfLastUpdate]
641     GO
```

```
1080    CREATE OR ALTER PROCEDURE [Project3].[Load_UserAuthorization]
1081        @UserAuthorizationKey INT
1082    AS
1083    BEGIN
1084        -- SET NOCOUNT ON added to prevent extra result sets from
1085        -- interfering with SELECT statements.
1086        DECLARE @StartingDateTime DATETIME2 = SYSDATETIME();
1087
1088        INSERT INTO [DbSecurity].[UserAuthorization]
1089            ([GroupMemberLastName],[GroupMemberFirstName])
1090        VALUES
1091
1092            ('Georgievska', 'Aleksandra'),
1093            ('Yakubova', 'Sigalita'),
1094            ('Kong', 'Nicholas'),
1095            ('Wray', 'Edwin'),
1096            ('Ahmed', 'Ahnaf'),
1097            ('Richman', 'Aryeh');
1098
1099        DECLARE @WorkFlowStepTableRowCount INT;
1100        SET @WorkFlowStepTableRowCount = 6;
1101        DECLARE @EndingDateTime DATETIME2 = SYSDATETIME();
1102        DECLARE @QueryTime BIGINT = CAST(DATEDIFF(MILLISECOND, @StartingDateTime, @EndingDateTime) AS bigint);
1103        EXEC [Process].[usp_TrackWorkFlow] 'Add Users',
1104                                           @WorkFlowStepTableRowCount,
1105                                           @StartingDateTime,
1106                                           @EndingDateTime,
1107                                           @QueryTime,
1108                                           @UserAuthorizationKey;
1109    END;
1110    GO
```

Code

```
214     CREATE TABLE [Process].[WorkflowSteps]
215     (
216         [WorkFlowStepKey] [Udt].[SurrogateKeyInt] NOT NULL IDENTITY(1,1), -- primary key
217         [WorkFlowStepDescription] [nvarchar](100) NOT NULL,
218         [WorkFlowStepTableRowCount] [int] NULL,
219         [StartingDateTime] [datetime2](7) NULL,
220         [EndingDateTime] [datetime2](7) NULL,
221         [QueryTime (ms)] [bigint] NULL,
222         [Class Time] [char](5) NULL,
223         -- all tables must have the following 3 columns:
224         [UserAuthorizationKey] [Udt].[SurrogateKeyInt] NOT NULL,
225         [DateAdded] [Udt].[DateAdded] NOT NULL,
226         [DateOfLastUpdate] [Udt].[DateOfLastUpdate] NOT NULL,
227         PRIMARY KEY CLUSTERED(
228             [WorkFlowStepKey] ASC
229     )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS
230     ) ON [PRIMARY]
231     GO
```

```
642     ALTER TABLE [Process].[WorkflowSteps] ADD  DEFAULT ((0)) FOR [WorkFlowStepTableRowCount]
643     GO
644     ALTER TABLE [Process].[WorkflowSteps] ADD  DEFAULT ('09:15') FOR [Class Time]
645     GO
646     ALTER TABLE [Process].[WorkflowSteps] ADD  DEFAULT (sysdatetime()) FOR [StartingDateTime]
647     GO
648     ALTER TABLE [Process].[WorkflowSteps] ADD  DEFAULT (sysdatetime()) FOR [EndingDateTime]
649     GO
650     ALTER TABLE [Process].[WorkflowSteps] ADD  DEFAULT (sysdatetime()) FOR [DateAdded]
651     GO
652     ALTER TABLE [Process].[WorkflowSteps] ADD  DEFAULT (sysdatetime()) FOR [DateOfLastUpdate]
653     GO
```

```
725     ALTER TABLE [Process].[WorkflowSteps]  WITH CHECK ADD  CONSTRAINT [FK_WorkFlowSteps_UserAuthorization] FOREIGN KEY([UserAuthorizationKey])
726     REFERENCES [DbSecurity].[UserAuthorization] ([UserAuthorizationKey])
727     GO
728     ALTER TABLE [Process].[WorkflowSteps] CHECK CONSTRAINT [FK_WorkFlowSteps_UserAuthorization]
729     GO
```

```
1028    CREATE OR ALTER PROCEDURE [Process].[usp_TrackWorkFlow]
1029        -- Add the parameters for the stored procedure here
1030        @WorkflowDescription NVARCHAR(100),
1031        @WorkFlowStepTableRowCount INT,
1032        @StartingDateTime DATETIME2,
1033        @EndingDateTime DATETIME2,
1034        @QueryTime BIGINT,
1035        @UserAuthorizationKey INT
1036    AS
1037    BEGIN
1038        -- SET NOCOUNT ON added to prevent extra result sets from
1039        -- interfering with SELECT statements.
1040        SET NOCOUNT ON;
1041
1042        -- Insert statements for procedure here
1043        INSERT INTO [Process].[WorkflowSteps]
1044            (
1045            WorkFlowStepDescription,
1046            WorkFlowStepTableRowCount,
1047            StartingDateTime,
1048            EndingDateTime,
1049            [QueryTime (ms)],
1050            UserAuthorizationKey
1051            )
1052        VALUES
1053            (@WorkflowDescription,
1054            @WorkFlowStepTableRowCount,
1055            @StartingDateTime,
1056            @EndingDateTime,
1057            @QueryTime,
1058            @UserAuthorizationKey);
1059
1060    END;
1061    GO
```

Code

```
250    CREATE TABLE [Personnel].[Instructor]
251    (
252        InstructorID [int] NOT NULL IDENTITY(1, 1), -- primary key
253        FirstName [char](25) NULL,
254        LastName [char](25) NULL,
255        -- all tables must have the following 3 columns:
256        [UserAuthorizationKey] [Udt].[SurrogateKeyInt] NOT NULL,
257        [DateAdded] [Udt].[DateAdded] NOT NULL,
258        [DateOfLastUpdate] [Udt].[DateOfLastUpdate] NOT NULL,
259        PRIMARY KEY CLUSTERED(
260            [InstructorID] ASC
261    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
262    ) ON [PRIMARY]
263    GO
```

```
654    ALTER TABLE [Personnel].[Instructor] ADD  DEFAULT (sysdatetime()) FOR [DateAdded]
655    GO
656    ALTER TABLE [Personnel].[Instructor] ADD  DEFAULT (sysdatetime()) FOR [DateOfLastUpdate]
657    GO
658    ALTER TABLE [Personnel].[Instructor] ADD DEFAULT ('none') FOR [LastName]
659    GO
660    ALTER TABLE [Personnel].[Instructor] ADD DEFAULT ('none') FOR [FirstName]
661    GO
```

```
730    ALTER TABLE [Personnel].[Instructor] WITH CHECK ADD  CONSTRAINT [FK_Instructor_UserAuthorization] FOREIGN KEY([UserAuthorizationKey])
731    REFERENCES [DbSecurity].[UserAuthorization] ([UserAuthorizationKey])
732    GO
733    ALTER TABLE [Personnel].[Instructor] CHECK CONSTRAINT [FK_Instructor_UserAuthorization]
734    GO
```

```
1358   CREATE OR ALTER PROCEDURE [Project3].[LoadInstructors] @UserAuthorizationKey INT
1359   AS
1360   BEGIN
1361       SET NOCOUNT ON;
1362       DECLARE @DateAdded DATETIME2 = SYSDATETIME();
1363       DECLARE @StartingDateTime DATETIME2 = SYSDATETIME();
1364
1365       INSERT INTO [Personnel].[Instructor](
1366           FirstName, LastName, UserAuthorizationKey, DateAdded
1367       )
1368       SELECT DISTINCT
1369           -- use COALESCE/NULLIF to check when importing the data from the original table to prevent importing nulls
1370           COALESCE(NULLIF(LTRIM(RTRIM(SUBSTRING(Instructor, CHARINDEX(',', Instructor) + 2, LEN(Instructor)))), ''), 'none') AS FirstName,
1371           COALESCE(NULLIF(LTRIM(RTRIM(SUBSTRING(Instructor, 1, CHARINDEX(',', Instructor) - 1))), ''), 'none') AS LastName,
1372           @UserAuthorizationKey,
1373           @DateAdded
1374       FROM
1375       [Uploadfile].[CurrentSemesterCourseOfferings]
1376       ORDER BY LastName;
1377
1378       DECLARE @WorkFlowStepTableRowCount INT;
1379       SET @WorkFlowStepTableRowCount = (
1380                                       SELECT COUNT(*)
1381                                       FROM [Personnel].[Instructor]
1382                                       );
1383       DECLARE @EndingDateTime DATETIME2 = SYSDATETIME();
1384       DECLARE @QueryTime BIGINT = CAST(DATEDIFF(MILLISECOND, @StartingDateTime, @EndingDateTime) AS bigint);
1385       EXEC [Process].[usp_TrackWorkFlow] 'Add Instructor Data',
1386                                           @WorkFlowStepTableRowCount,
1387                                           @StartingDateTime,
1388                                           @EndingDateTime,
1389                                           @QueryTime,
1390                                           @UserAuthorizationKey;
1391   END;
1392   GO
```

Code

```
281  CREATE TABLE [Academic].[Course]
282  (
283      CourseId INT NOT NULL IDENTITY(1, 1), -- primary key
284      CourseAbbreviation CHAR(5) NOT NULL, -- needs check constraint
285      CourseNumber CHAR(5) NOT NULL,
286      CourseCredit FLOAT NOT NULL, -- (Needs Check Constraint to be positive)
287      CreditHours [Udt].[CreditHours] NOT NULL, -- CreditHours (Needs Check Constraint) -> should be positive,
288      CourseName CHAR(35) NOT NULL, -- CourseDescription
289      DepartmentID [int] NOT NULL, -- FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
290      -- all tables must have the following 3 columns:
291      [UserAuthorizationKey] [Udt].[SurrogateKeyInt] NOT NULL,
292      [DateAdded] [Udt].[DateAdded] NOT NULL,
293      [DateOfLastUpdate] [Udt].[DateOfLastUpdate] NOT NULL,
294      PRIMARY KEY CLUSTERED(
295          [CourseId] ASC
296  )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE
297  ) ON [PRIMARY]
298  GO
```

```
662  ALTER TABLE [Academic].[Course] ADD  DEFAULT (sysdatetime()) FOR [DateAdded]
663  GO
664  ALTER TABLE [Academic].[Course] ADD  DEFAULT (sysdatetime()) FOR [DateOfLastUpdate]
665  GO
666  ALTER TABLE [Academic].[Course] ADD DEFAULT ('unknown') FOR [CourseName]
667  GO
```

```
741  ALTER TABLE [Academic].[Course] CHECK CONSTRAINT [FK_Course_UserAuthorization]
742  GO
743  ALTER TABLE [Academic].[Course] ADD CONSTRAINT [CHK_CreditHours_Positive] CHECK (CreditHours >= 0)
744  GO
745  ALTER TABLE [Academic].[Course] ADD CONSTRAINT [CHK_CourseCredit_Positive] CHECK (CourseCredit >= 0)
746  GO
```

```
1404  CREATE OR ALTER PROCEDURE [Project3].[LoadCourse] @UserAuthorizationKey INT
1405  AS
1406  BEGIN
1407      SET NOCOUNT ON;
1408      DECLARE @DateAdded DATETIME2 = SYSDATETIME();
1409      DECLARE @StartingDateTime DATETIME2 = SYSDATETIME();
1410
1411      INSERT INTO [Academic].[Course](
1412          [CourseAbbreviation] -- Course (parse letters)
1413          ,[CourseNumber] -- Course (parse number)
1414          ,[CourseCredit] -- Course (parse second number in (,))
1415          ,[CreditHours] -- Course (parse first number in (,))
1416          ,[CourseName] -- Description
1417          ,[DepartmentID] -- fk
1418          ,UserAuthorizationKey
1419          ,DateAdded
1420      )
1421      SELECT DISTINCT
1422          LEFT([Course (hr, crd)], PATINDEX('%[ ()]%', [Course (hr, crd)]) - 1) -- CourseAbbreviation
1423          ,SUBSTRING(
1424                  [Course (hr, crd)],
1425                  PATINDEX('%[0-9]%', [Course (hr, crd)]),
1426                  CHARINDEX('(', [Course (hr, crd)]) - PATINDEX('%[0-9]%', [Course (hr, crd)])
1427              ) -- CourseNumber
1428          ,CAST(SUBSTRING(
1429                  [Course (hr, crd)],
1430                  CHARINDEX(',', [Course (hr, crd)]) + 2,
1431                  CHARINDEX(')', [Course (hr, crd)]) - CHARINDEX(',', [Course (hr, crd)]) - 2
1432                  ) AS FLOAT) --CourseCredit
1433          ,CAST(SUBSTRING(
1434                  [Course (hr, crd)],
1435                  CHARINDEX('(', [Course (hr, crd)]) + 1,
1436                  CHARINDEX(',', [Course (hr, crd)]) - CHARINDEX('(', [Course (hr, crd)]) - 1
1437                  ) AS FLOAT) -- CreditHours
1438          ,C.Description -- CourseName
1439          , ( SELECT D.DepartmentID
1440              FROM [Academic].[Department] AS D
1441              WHERE D.DepartmentName = LEFT([Course (hr, crd)], PATINDEX('%[ ()]%', [Course (hr, crd)]) - 1))
1442          ,@UserAuthorizationKey
1443          ,@DateAdded
1444      FROM
1445      [Uploadfile].[CurrentSemesterCourseOfferings] AS C;
1446
1447      DECLARE @WorkFlowStepTableRowCount INT;
1448      SET @WorkFlowStepTableRowCount = (
1449                                      SELECT COUNT(*)
1450                                      FROM [Academic].[Course]
1451                                      );
1452      DECLARE @EndingDateTime DATETIME2 = SYSDATETIME();
1453      DECLARE @QueryTime BIGINT = CAST(DATEDIFF(MILLISECOND, @StartingDateTime, @EndingDateTime) AS bigint);
1454      EXEC [Process].[usp_TrackWorkFlow] 'Add Instructor Data',
1455                                      @WorkFlowStepTableRowCount,
1456                                      @StartingDateTime,
1457                                      @EndingDateTime,
1458                                      @QueryTime,
1459                                      @UserAuthorizationKey;
1460  END;
1461  GO
```

Queries

```
/* ============================================
-- Author:              Aleksandra Georgievska
-- Create date: 12/8/23
-- Proposition: Which department offers the most number of courses
-- ============================================*/

GO

WITH DepartmentCourseCount AS (
    SELECT
        DepartmentID,
        COUNT(CourseId) AS NumberOfCoursesOffered
    FROM [Academic].[Course]
    GROUP BY DepartmentID
)
SELECT TOP 1
    DepartmentID,
    NumberOfCoursesOffered
FROM DepartmentCourseCount
ORDER BY NumberOfCoursesOffered DESC;
```

Results | Messages

	DepartmentID	NumberOfCoursesOffered
1	66	146

Queries

```sql
/* ========================================
-- Author:              Aleksandra Georgievska
-- Create date: 12/8/23
-- Proposition: what instructors teach the course with the most number of sections?
-- ========================================*/

GO

WITH CourseMostSections (CourseName, CourseSectionCounts, DepartmentID)
AS
(
    SELECT TOP 1
        C.CourseName
        , COUNT(S.Section) AS CourseSectionCounts
        , DepartmentID
    FROM [Academic].[Course] AS C
        INNER JOIN [Academic].[Section] AS S
        ON C.CourseId = S. CourseId
    GROUP BY CourseName, DepartmentID
    ORDER BY CourseSectionCounts DESC
)
SELECT   C.CourseName
        , C.CourseSectionCounts
        , C.DepartmentId
        , D.InstructorId
        , (I.FirstName + ' ' + I.LastName) AS InstructorFullName
FROM CourseMostSections AS C
INNER JOIN [Personnel].[DepartmentInstructor] AS D
ON C.DepartmentID = D.DepartmentId
INNER JOIN [Personnel].[Instructor] AS I
ON D.InstructorId = I.InstructorID
```

Results Messages

	CourseName	CourseSectionCounts	DepartmentId	InstructorId	InstructorFullName	
1	Private Instruction Jazz Perf	62	51	17	David	Adler
2	Private Instruction Jazz Perf	62	51	56	Yoshio	Aomori
3	Private Instruction Jazz Perf	62	51	58	Darcy	Argue
4	Private Instruction Jazz Perf	62	51	59	Maria	Argyros
5	Private Instruction Jazz Perf	62	51	60	Timothy	Armacost
6	Private Instruction Jazz Perf	62	51	119	David	Berkman
7	Private Instruction Jazz Perf	62	51	148	Rogerio	Boccato
8	Private Instruction Jazz Perf	62	51	154	Paul	Bollenback
9	Private Instruction Jazz Perf	62	51	158	Luis	Bonilla
10	Private Instruction Jazz Perf	62	51	209	Peter	Calandra
11	Private Instruction Jazz Perf	62	51	262	Vincent	Cherico
12	Private Instruction Jazz Perf	62	51	418	John	Ellis
13	Private Instruction Jazz Perf	62	51	488	Ian	Froman
14	Private Instruction Jazz Perf	62	51	606	Antonio	Hart
15	Private Instruction Jazz Perf	62	51	607	William	Hart
16	Private Instruction Jazz Perf	62	51	676	Aubrey	Johnson
17	Private Instruction Jazz Perf	62	51	868	Jeannette	Lovetri
18	Private Instruction Jazz Perf	62	51	886	Dennis	Mackrel
19	Private Instruction Jazz Perf	62	51	917	Joseph	Martin
20	Private Instruction Jazz Perf	62	51	995	Michael	Moreno
21	Private Instruction Jazz Perf	62	51	999	Michael	Mossman
22	Private Instruction Jazz Perf	62	51	1090	Jeb	Patton
23	Private Instruction Jazz Perf	62	51	1126	Lonnie	Plaxico
24	Private Instruction Jazz Perf	62	51	1474	Charenee	Wade
25	Private Instruction Jazz Perf	62	51	1546	David	Wong

NACE

- Teamwork: Learned to collaborate and communicate efficiently in a team, valuing diverse contributions and working towards common goals.
- Communication: Enhanced verbal and written communication skills, fostering clear and concise messages to interact effectively with others.
- Technology: Acquired proficiency in relevant technologies, leveraging them to improve efficiency and productivity in job-related tasks.

Aryeh Richman

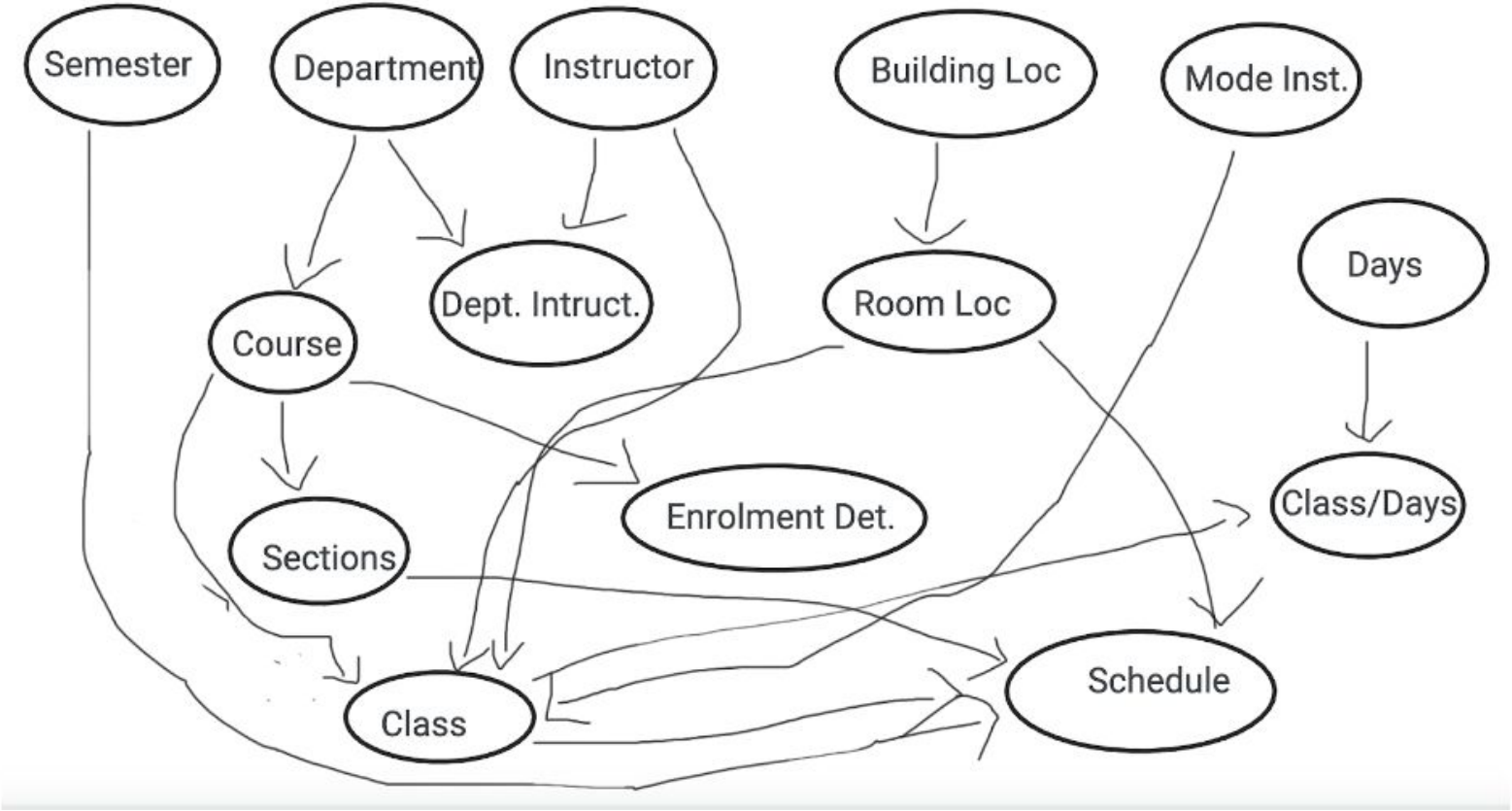
Group 1 - Project 3

To-do list

To be completed by:		Name: Aryeh Richman	
Deadline:	12/10	Date	

Project 3

% done	Phase	Start By	Original Due By	Revised Due By	Number Of Days	Revision Notes
100%	Group meeting #1	12/1/23	12/1/23		1	
100%	Data Modeling (ERD Design)	12/2/23	12/4/23		3	
100%	Group meeting #2	12/4/23	12/4/23		1	
100%	Table Hierarchy Relationship	12/4/23	12/4/23		1	
100%	Tier 1 Table: Departments	12/5/23	12/5/23		1	
100%	Tier 2 Table: DepartmentInstructors	12/6/23	12/6/23		1	
100%	Queries	12/7/23	12/7/23	9-Dec-23	1	
100%	Make Slideshow	12/8/23	12/8/23	9-Dec-23	1	Extension
100%	Recording	12/8/23	12/8/23	10-Dec-23	1	Extension
100%	Submit	12/10/23	12/10/23		1	


Semester
Department
Instructor
Building Loc
Mode Inst.
Days
Dept. Intruct.
Room Loc
Course
Enrolment Det.
Class/Days
Sections
Schedule
Class

```
/*

Table: [Personnel].[DepartmentInstructor]

-- ============================================
-- Author:              Aryeh Richman
-- Create date: 12/6/23
-- Description: Create a bridge table between departments and instructors
-- ============================================

*/
DROP TABLE IF EXISTS [Personnel].[DepartmentInstructor]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [Personnel].[DepartmentInstructor]
(
    DepartmentInstructorID [int] NOT NULL IDENTITY(1, 1), -- primary key
    DepartmentID [int] NOT NULL,
    InstructorID [int] NOT NULL,
    -- all tables must have the following 3 columns:
    [UserAuthorizationKey] [Udt].[SurrogateKeyInt] NOT NULL,
    [DateAdded] [Udt].[DateAdded] NOT NULL,
    [DateOfLastUpdate] [Udt].[DateOfLastUpdate] NOT NULL,
    PRIMARY KEY CLUSTERED(
        [DepartmentInstructorID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
) ON [PRIMARY]
GO
```

```
/*

Table: [Academic].[Department]

-- ============================================
-- Author:              Aryeh Richman
-- Create date: 12/5/23
-- Description: Create a Department table with id and name
-- ============================================

*/
DROP TABLE IF EXISTS [Academic].[Department]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [Academic].[Department]
(
    DepartmentID [int] NOT NULL IDENTITY(1, 1), -- primary key
    DepartmentName [char](5) NOT NULL,
    -- all tables must have the following 3 columns:
    [UserAuthorizationKey] [Udt].[SurrogateKeyInt] NOT NULL,
    [DateAdded] [Udt].[DateAdded] NOT NULL,
    [DateOfLastUpdate] [Udt].[DateOfLastUpdate] NOT NULL,
    PRIMARY KEY CLUSTERED(
        [DepartmentID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
) ON [PRIMARY]
GO
```

```
-- Aryeh
ALTER TABLE [Academic].[Department]  WITH CHECK ADD  CONSTRAINT [FK_Department_UserAuthorization] FOREIGN KEY([UserAuthorizationKey])
REFERENCES [DbSecurity].[UserAuthorization] ([UserAuthorizationKey])
GO
ALTER TABLE [Academic].[Department] CHECK CONSTRAINT [FK_Department_UserAuthorization]
GO
ALTER TABLE [Personnel].[DepartmentInstructor]  WITH CHECK ADD  CONSTRAINT [FK_DepartmentInstructor_UserAuthorization] FOREIGN KEY([UserAuthorizationKey])
REFERENCES [DbSecurity].[UserAuthorization] ([UserAuthorizationKey])
GO
ALTER TABLE [Personnel].[DepartmentInstructor] CHECK CONSTRAINT [FK_DepartmentInstructor_UserAuthorization]
GO
ALTER TABLE [Personnel].[DepartmentInstructor]  WITH CHECK ADD  CONSTRAINT [FK_DepartmentInstructor_Department] FOREIGN KEY([DepartmentID])
REFERENCES [Academic].[Department] ([DepartmentID])
GO
ALTER TABLE [Personnel].[DepartmentInstructor] CHECK CONSTRAINT [FK_DepartmentInstructor_Department]
GO
```

```
-- Aryeh
ALTER TABLE [Academic].[Department] ADD  DEFAULT (sysdatetime()) FOR [DateAdded]
GO
ALTER TABLE [Academic].[Department] ADD  DEFAULT (sysdatetime()) FOR [DateOfLastUpdate]
GO
ALTER TABLE [Personnel].[DepartmentInstructor] ADD  DEFAULT (sysdatetime()) FOR [DateAdded]
GO
ALTER TABLE [Personnel].[DepartmentInstructor] ADD  DEFAULT (sysdatetime()) FOR [DateOfLastUpdate]
GO
```

```
/*
Stored Procedure: [Project3].[LoadDepartmentInstructor]

-- ============================================
-- Author:              Aryeh Richman
-- Create date: 12/6/23
-- Description: Adds the values to the Department / Instructor bridge table
-- ============================================

*/

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE OR ALTER PROCEDURE [Project3].[LoadDepartmentInstructor] @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @DateAdded DATETIME2 = SYSDATETIME();
    DECLARE @StartingDateTime DATETIME2 = SYSDATETIME();

    INSERT INTO [Personnel].[DepartmentInstructor] (
        DepartmentID, InstructorID, UserAuthorizationKey, DateAdded
    )
    SELECT DISTINCT D.DepartmentID, I.InstructorID, @UserAuthorizationKey, @DateAdded
    FROM Academic.Department AS D
        CROSS JOIN Personnel.Instructor AS I
        INNER JOIN Uploadfile.CurrentSemesterCourseOfferings AS U
            ON LEFT(U.[Course (hr, crd)], CHARINDEX(' ', U.[Course (hr, crd)]) - 1) = D.DepartmentName
            AND LTRIM(RTRIM(SUBSTRING(U.Instructor, CHARINDEX(',', U.Instructor) + 2, LEN(U.Instructor)))) = I.FirstName
            AND LTRIM(RTRIM(SUBSTRING(U.Instructor, 1, CHARINDEX(',', U.Instructor) - 1))) = I.LastName

    DECLARE @WorkFlowStepTableRowCount INT;
    SET @WorkFlowStepTableRowCount = 0;
    DECLARE @EndingDateTime DATETIME2 = SYSDATETIME();
    DECLARE @QueryTime BIGINT = CAST(DATEDIFF(MILLISECOND, @StartingDateTime, @EndingDateTime) AS bigint);
    EXEC [Process].[usp_TrackWorkFlow] 'Add Department Data',
                                       @WorkFlowStepTableRowCount,
                                       @StartingDateTime,
                                       @EndingDateTime,
                                       @QueryTime,
                                       @UserAuthorizationKey;
END;
GO
```

```
/*
Stored Procedure: [Project3].[LoadDepartments]

-- ============================================
-- Author:              Aryeh Richman
-- Create date: 12/5/23
-- Description: Adds the Departments to the Department Table
-- ============================================

*/

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE OR ALTER PROCEDURE [Project3].[LoadDepartments] @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @DateAdded DATETIME2 = SYSDATETIME();
    DECLARE @StartingDateTime DATETIME2 = SYSDATETIME();

    INSERT INTO [Academic].[Department] (
        DepartmentName, UserAuthorizationKey, DateAdded
    )
    SELECT DISTINCT
        LEFT([Course (hr, crd)], CHARINDEX(' ', [Course (hr, crd)]) - 1) AS DepartmentName,
        @UserAuthorizationKey,
        @DateAdded
    FROM [Uploadfile].[CurrentSemesterCourseOfferings]
    ORDER BY DepartmentName

    DECLARE @WorkFlowStepTableRowCount INT;
    SET @WorkFlowStepTableRowCount = 0;
    DECLARE @EndingDateTime DATETIME2 = SYSDATETIME();
    DECLARE @QueryTime BIGINT = CAST(DATEDIFF(MILLISECOND, @StartingDateTime, @EndingDateTime) AS bigint);
    EXEC [Process].[usp_TrackWorkFlow] 'Add Department Data',
                                       @WorkFlowStepTableRowCount,
                                       @StartingDateTime,
                                       @EndingDateTime,
                                       @QueryTime,
                                       @UserAuthorizationKey;
END;
GO
```

Queries

Which Instructors teach courses with over 150 students enrolled?

```sql
SELECT DISTINCT CONCAT(I.FirstName, ' ', I.LastName) AS Professor
FROM Enrollment.EnrollmentDetail AS E
    INNER JOIN Academic.Section AS S
        ON S.SectionID = E.SectionID
        INNER JOIN Academic.Course AS C
            ON C.CourseId = S.CourseID
        INNER JOIN Personnel.DepartmentInstructor AS DI
            ON DI.DepartmentID = C.DepartmentID
        INNER JOIN Personnel.Instructor AS I
            ON DI.InstructorID = I.InstructorID
WHERE E.CurrentEnrollment > 150
```

Results | Messages

	Professor
1	Emnet Gammada
2	Maria Stalias
3	Renee Goodwin
4	Karl Fath
5	Jose Anadon
6	Riddhi Chauhan
7	Erica Doran
8	Jeffrey Farber
9	Anna Budd
1	Vivek Upadhyay

Which classes have over enrollment and by how many students?

```sql
SELECT DISTINCT CONCAT(C.CourseAbbreviation, C.CourseNumber) AS Course,
                C.CourseName,
                S.SectionID,
                E.CurrentEnrollment,
                E.MaxEnrollmentLimit,
                (E.CurrentEnrollment - E.MaxEnrollmentLimit) AS Overflow
FROM Academic.Course AS C
    INNER JOIN Academic.Section AS S
        ON C.CourseId = S.CourseID
    INNER JOIN Enrollment.EnrollmentDetail AS E
        ON S.SectionID = E.SectionID
WHERE E.CurrentEnrollment > E.MaxEnrollmentLimit
```

Results | Messages

	Course	CourseName	SectionID	CurrentEnrollment	MaxEnrollmentLimit	Overflow
1	ACCT 202	Inter Acct 2	834	55	53	2
2	ACCT 202	Inter Acct 2	5021	55	53	2
3	ACCT 202	Inter Acct 2	9208	55	53	2
4	ACCT 261	Business Law l	2139	46	44	2
5	ACCT 261	Business Law l	6326	46	44	2
6	ACCT 261	Business Law l	10513	46	44	2
7	ACCT 306	Quant Techniq Pln & Ctrl	2249	55	53	2
8	ACCT 306	Quant Techniq Pln & Ctrl	6436	55	53	2
9	ACCT 306	Quant Techniq Pln & Ctrl	10623	55	53	2
1…	ACCT 322	Auditing 2	867	51	50	1
1…	ACCT 322	Auditing 2	5054	51	50	1

Edwin Wray

Group 1 - Project 3

To-do List

Project 1						
% done	Phase	Start By	Original Due By2	Revised Due By	Number Of Days	Revision Notes
100%	Preparation	11/29/2023	12/1/2023			
100%	Group Meeting	12/1/2023	12/1/2023			
100%	Planning	12/1/2023	12/4/2023			
100%	Follow-up Group Meeting	12/4/2023	12/4/2023			
100%	Tier 1 tables: BuildingLocation table	12/6/2023	12/6/2023			
100%	Tier 3 tables: Class table	12/9/2023	12/7/2023	12/9/2023	2	Multiple Bugs
100%	Queries	12/10/2023	12/8/2023	12/10/2023	2	Delayed by Class table
100%	Create Slides	12/10/2023	12/8/2023	12/10/2023	2	Delayed by Class table
100%	Record Presentation	12/10/2023	12/8/2023	12/10/2023	2	Delayed by Class table
0%	Submit	12/10/2023	12/8/2023	12/10/2023	2	Extension Granted

Table: BuildingLocations

```
DROP TABLE IF EXISTS [Facilities].[BuildingLocations]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [Facilities].[BuildingLocations]
(
    BuildingID [int] NOT NULL IDENTITY(1, 1), -- primary key
    BuildingNameAbbrv [Udt].[BuildingNameAbbrv] NOT NULL,
    BuildingName [Udt].[BuildingName] NOT NULL,
    -- all tables must have the following 3 columns:
    [UserAuthorizationKey] [Udt].[SurrogateKeyInt] NOT NULL,
    [DateAdded] [Udt].[DateAdded] NOT NULL,
    [DateOfLastUpdate] [Udt].[DateOfLastUpdate] NOT NULL,
    PRIMARY KEY CLUSTERED(
    [BuildingID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
    = OFF, ALLOW_ROW_LOCKS =
    ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
CREATE TYPE [Udt].[BuildingNameAbbrv] FROM NVARCHAR(3) NOT NULL;
GO
CREATE TYPE [Udt].[BuildingName] FROM NVARCHAR(50) NOT NULL;
GO
```

Constraints

```
ALTER TABLE [Facilities].[BuildingLocations]  WITH CHECK ADD  CONSTRAINT [FK_BuildingLocations_UserAuthorization] FOREIGN KEY([UserAuthorizationKey])
REFERENCES [DbSecurity].[UserAuthorization] ([UserAuthorizationKey])
GO
ALTER TABLE [Facilities].[BuildingLocations] CHECK CONSTRAINT [FK_BuildingLocations_UserAuthorization]
GO
```

UDT: GetBuildingNameAbbrv

```
-- Create a function to determine the BuildingName
CREATE FUNCTION [Udt].[GetBuildingNameAbbrv](@Location VARCHAR(50))
RETURNS [Udt].[BuildingNameAbbrv]
AS
BEGIN
    IF @Location IS NULL OR LTRIM(RTRIM(@Location)) = ''
        RETURN 'TBD';

    DECLARE @BuildingNameAbbrv [Udt].[BuildingNameAbbrv];
    SET @BuildingNameAbbrv =
        CASE
            WHEN CHARINDEX(' ', @Location) > 0
            THEN LEFT(@Location, CHARINDEX(' ', @Location) - 1)
            ELSE 'TBD'
        END;

    RETURN @BuildingNameAbbrv;
END;
GO
```

UDT: GetBuildingName

```sql
-- Create a function to determine the BuildingName
CREATE FUNCTION [Udt].[GetBuildingName](@Location VARCHAR(50))
RETURNS [Udt].[BuildingName]
AS
BEGIN
    -- Return 'TBD' if the input is NULL or an empty string
    IF @Location IS NULL OR LTRIM(RTRIM(@Location)) = ''
        RETURN 'TBD';
    DECLARE @BuildingNameAbbrv NVARCHAR(2);
    SET @BuildingNameAbbrv = [Udt].[GetBuildingNameAbbrv](@Location);
    DECLARE @BuildingName [Udt].[BuildingName];
    SET @BuildingName =
        CASE
            WHEN @BuildingNameAbbrv = 'AE' THEN 'Alumni Hall'
            WHEN @BuildingNameAbbrv = 'CD' THEN 'Campbell Dome'
            WHEN @BuildingNameAbbrv = 'CA' THEN 'Colden Auditorium'
            WHEN @BuildingNameAbbrv = 'CH' THEN 'Colwin Hall'
            WHEN @BuildingNameAbbrv = 'CI' THEN 'Continuing Ed 1'
            WHEN @BuildingNameAbbrv = 'DY' THEN 'Delany Hall'
            WHEN @BuildingNameAbbrv = 'DH' THEN 'Dining Hall'
            WHEN @BuildingNameAbbrv = 'FG' THEN 'FitzGerald Gym'
            WHEN @BuildingNameAbbrv = 'FH' THEN 'Frese Hall'
            WHEN @BuildingNameAbbrv = 'GB' THEN 'G Building'
            WHEN @BuildingNameAbbrv = 'GC' THEN 'Gertz Center'
            WHEN @BuildingNameAbbrv = 'GT' THEN 'Goldstein Theatre'
            WHEN @BuildingNameAbbrv = 'HH' THEN 'Honors Hall'
            WHEN @BuildingNameAbbrv = 'IB' THEN 'I Building'
            WHEN @BuildingNameAbbrv = 'JH' THEN 'Jefferson Hall'
            WHEN @BuildingNameAbbrv = 'KY' THEN 'Kiely Hall'
            WHEN @BuildingNameAbbrv = 'KG' THEN 'King Hall'
            WHEN @BuildingNameAbbrv = 'KS' THEN 'Kissena Hall'
            WHEN @BuildingNameAbbrv = 'KP' THEN 'Klapper Hall'
            WHEN @BuildingNameAbbrv = 'MU' THEN 'Music Building'
            WHEN @BuildingNameAbbrv = 'PH' THEN 'Powdermaker Hall'
            WHEN @BuildingNameAbbrv = 'QH' THEN 'Queens Hall'
            WHEN @BuildingNameAbbrv = 'RA' THEN 'Rathaus Hall'
            WHEN @BuildingNameAbbrv = 'RZ' THEN 'Razran Hall'
            WHEN @BuildingNameAbbrv = 'RE' THEN 'Remsen Hall'
            WHEN @BuildingNameAbbrv = 'RO' THEN 'Rosenthal Library'
            WHEN @BuildingNameAbbrv = 'SB' THEN 'Science Building'
            WHEN @BuildingNameAbbrv = 'SU' THEN 'Student Union'
            WHEN @BuildingNameAbbrv = 'C2' THEN 'Tech Incubator'
            ELSE 'TBD' -- Default case for any other or unexpected abbreviation
        END;
    RETURN @BuildingName;
END;
GO
```

Stored Procedure: LoadBuildingLocations

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE OR ALTER PROCEDURE [Project3].[LoadBuildingLocations] @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @DateAdded DATETIME2 = SYSDATETIME();
    DECLARE @StartingDateTime DATETIME2 = SYSDATETIME();

    INSERT INTO [Facilities].[BuildingLocations] (
        BuildingNameAbbrv, BuildingName, UserAuthorizationKey, DateAdded
    )
    SELECT DISTINCT
        [UDT].[GetBuildingNameAbbrv]([Location]) AS BuildingNameAbbrv,
        [UDT].[GetBuildingName]([Location]) AS BuildingName,
        @UserAuthorizationKey,
        @DateAdded
    FROM [Uploadfile].[CurrentSemesterCourseOfferings]
    ORDER BY BuildingName

    DECLARE @WorkFlowStepTableRowCount INT;
    SET @WorkFlowStepTableRowCount = 0;
    DECLARE @EndingDateTime DATETIME2 = SYSDATETIME();
    DECLARE @QueryTime BIGINT = CAST(DATEDIFF(MILLISECOND, @StartingDateTime, @EndingDateTime) AS bigint);
    EXEC [Process].[usp_TrackWorkFlow] 'Add BuildingLocations Data',
                                       @WorkFlowStepTableRowCount,
                                       @StartingDateTime,
                                       @EndingDateTime,
                                       @QueryTime,
                                       @UserAuthorizationKey;
END;
GO
```

Table Data: BuildingLocations

Results Messages

	BuildingID	Buildin…	BuildingName	UserAuthori…	DateAdded	DateOfLastUpdate
1	1	CD	Campbell Dome	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
2	2	CH	Colwin Hall	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
3	3	DY	Delany Hall	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
4	4	FG	FitzGerald Gym	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
5	5	GB	G Building	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
6	6	GC	Gertz Center	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
7	7	GT	Goldstein Theatre	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
8	8	HH	Honors Hall	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
9	9	IB	I Building	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
1…	10	JH	Jefferson Hall	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
1…	11	KY	Kiely Hall	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
1…	12	KG	King Hall	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
1…	13	KP	Klapper Hall	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
1…	14	MU	Music Building	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
1…	15	PH	Powdermaker Hall	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
1…	16	QH	Queens Hall	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
1…	17	RA	Rathaus Hall	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
1…	18	RZ	Razran Hall	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
1…	19	RE	Remsen Hall	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
2…	20	RO	Rosenthal Library	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
2…	21	SB	Science Building	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
2…	22	SU	Student Union	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
2…	23	AR	TBD	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333
2…	24	TBD	TBD	4	2023-12-10 16:04:50.6733333	2023-12-10 16:04:50.6733333

Table: Class

```
DROP TABLE IF EXISTS [ClassManagement].[Class]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [ClassManagement].[Class]
(
    ClassID [int] NOT NULL IDENTITY(1, 1), -- primary key
    CourseID [int] NOT NULL,
    SectionID [int] NOT NULL,
    InstructorID [int] NOT NULL,
    RoomID [int] NOT NULL,
    ModeID [int] NOT NULL,
    -- all tables must have the following 3 columns:
    [UserAuthorizationKey] [Udt].[SurrogateKeyInt] NOT NULL,
    [DateAdded] [Udt].[DateAdded] NOT NULL,
    [DateOfLastUpdate] [Udt].[DateOfLastUpdate] NOT NULL,
    PRIMARY KEY CLUSTERED(
    [ClassID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
    = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS
    = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Constraints

```
ALTER TABLE [ClassManagement].[Class]  WITH CHECK ADD  CONSTRAINT [FK_Class_UserAuthorization] FOREIGN KEY([UserAuthorizationKey])
REFERENCES [DbSecurity].[UserAuthorization] ([UserAuthorizationKey])
GO
ALTER TABLE [ClassManagement].[Class] CHECK CONSTRAINT [FK_Class_UserAuthorization]
GO
ALTER TABLE [ClassManagement].[Class]  WITH CHECK ADD  CONSTRAINT [FK_Class_Course] FOREIGN KEY([CourseID])
REFERENCES [Academic].[Course] ([CourseID])
GO
ALTER TABLE [ClassManagement].[Class] CHECK CONSTRAINT [FK_Class_Course]
GO
ALTER TABLE [ClassManagement].[Class]  WITH CHECK ADD  CONSTRAINT [FK_Class_Section] FOREIGN KEY([SectionID])
REFERENCES [Academic].[Section] ([SectionID])
GO
ALTER TABLE [ClassManagement].[Class] CHECK CONSTRAINT [FK_Class_Section]
GO
ALTER TABLE [ClassManagement].[Class]  WITH CHECK ADD  CONSTRAINT [FK_Class_Instructor] FOREIGN KEY([InstructorID])
REFERENCES [Personnel].[Instructor] ([InstructorID])
GO
ALTER TABLE [ClassManagement].[Class] CHECK CONSTRAINT [FK_Class_Instructor]
GO
ALTER TABLE [ClassManagement].[Class]  WITH CHECK ADD  CONSTRAINT [FK_Class_RoomLocation] FOREIGN KEY([RoomID])
REFERENCES [Facilities].[RoomLocation] ([RoomID])
GO
ALTER TABLE [ClassManagement].[Class] CHECK CONSTRAINT [FK_Class_RoomLocation]
GO
ALTER TABLE [ClassManagement].[Class]  WITH CHECK ADD  CONSTRAINT [FK_Class_ModeOfInstruction] FOREIGN KEY([ModeID])
REFERENCES [ClassManagement].[ModeOfInstruction] ([ModeID])
GO
ALTER TABLE [ClassManagement].[Class] CHECK CONSTRAINT [FK_Class_ModeOfInstruction]
GO
```

Stored Procedure: LoadClass

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE OR ALTER PROCEDURE [Project3].[LoadClass] @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @DateAdded DATETIME2 = SYSDATETIME();
    DECLARE @StartingDateTime DATETIME2 = SYSDATETIME();

    INSERT INTO [ClassManagement].[Class] (
        CourseID, SectionID, InstructorID, RoomID, ModeID, UserAuthorizationKey, DateAdded
    )
    SELECT DISTINCT
        ( SELECT TOP 1 C.CourseID
            FROM [Academic].[Course] AS C
            WHERE C.CourseAbbreviation = LEFT([Course (hr, crd)], PATINDEX('%[ ()]%', [Course (hr, crd)]) - 1) -- CourseAbbreviation
                AND C.CourseNumber = SUBSTRING([Course (hr, crd)], PATINDEX('%[0-9]%', [Course (hr, crd)]),
                        CHARINDEX('(', [Course (hr, crd)]) - PATINDEX('%[0-9]%', [Course (hr, crd)])) -- CourseNumber
        )
        , ( SELECT TOP 1 S.SectionID
            FROM [Academic].[Section] AS S
            WHERE S.Code = U.Code -- Section Code
                AND S.Section = U.Sec -- Section Number
        )
        , ( SELECT TOP 1 I.InstructorID
            FROM [Personnel].[Instructor] AS I
            WHERE I.FirstName = COALESCE(NULLIF(LTRIM(RTRIM(SUBSTRING(U.Instructor,
                CHARINDEX(',', U.Instructor) + 2, LEN(U.Instructor)))), ''), 'none') -- FirstName
                AND I.LastName = COALESCE(NULLIF(LTRIM(RTRIM(SUBSTRING(U.Instructor, 1,
                    CHARINDEX(',', U.Instructor) - 1))), ''), 'none') -- LastName
        )
        , ( SELECT TOP 1 R.RoomID
            FROM [Facilities].[RoomLocation] AS R
            WHERE R.RoomNumber = CASE
                                    -- add the edge cases and then manually set it correctly
                                    WHEN RIGHT(U.Location, 4) = 'H 17' THEN '17'
                                    WHEN RIGHT(U.Location, 4) = '135H' THEN 'A135H'
                                    WHEN RIGHT(U.Location, 4) = '135B' THEN 'A135B'
                                    WHEN RIGHT(U.Location, 4) = 'H 09' THEN '09'
                                    WHEN RIGHT(U.Location, 4) = 'H 12' THEN '12'

                                    -- checks for null and empty string, if so set default string named TBD
                                    WHEN U.Location IS NULL OR LTRIM(RTRIM(U.Location)) = '' THEN 'TBD'
                                        ELSE RIGHT(U.Location, 4)
                                END
        )
        , ( SELECT TOP 1 M.ModeID
            FROM [ClassManagement].[ModeOfInstruction] AS M
            WHERE M.ModeName = U.[Mode of Instruction]
        )
        , @UserAuthorizationKey
        , @DateAdded
    FROM [Uploadfile].[CurrentSemesterCourseOfferings] AS U;

    DECLARE @WorkFlowStepTableRowCount INT;
    SET @WorkFlowStepTableRowCount = 0;
    DECLARE @EndingDateTime DATETIME2 = SYSDATETIME();
    DECLARE @QueryTime BIGINT = CAST(DATEDIFF(MILLISECOND, @StartingDateTime, @EndingDateTime) AS bigint);
    EXEC [Process].[usp_TrackWorkFlow] 'Add Class Data',
                                        @WorkFlowStepTableRowCount,
                                        @StartingDateTime,
                                        @EndingDateTime,
                                        @QueryTime,
                                        @UserAuthorizationKey;
END;
GO
```

Table Data: Class

Results Messages

	ClassID	CourseID	SectionID	InstructorID	RoomID	ModeID	UserAuthorizationKey	DateAdded	DateOfLastUpdate
1	1	1	510	969	1	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
2	2	1	1291	969	1	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
3	3	1	1649	630	2	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
4	4	2	831	333	9	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
5	5	2	1430	581	4	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
6	6	2	1765	424	13	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
7	7	2	1973	1229	9	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
8	8	2	2127	424	11	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
9	9	2	2244	445	4	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
10	10	2	2341	1347	10	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
11	11	2	2424	1229	9	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
12	12	2	2492	1584	6	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
13	13	2	2962	1347	7	3	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
14	14	2	3024	240	6	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
15	15	2	3105	972	5	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
16	16	3	832	581	4	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
17	17	3	1431	424	13	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
18	18	3	1766	445	4	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
19	19	3	1974	334	17	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
20	20	3	2128	1387	17	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
21	21	3	2245	717	17	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
22	22	3	2342	112	6	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
23	23	3	2425	722	5	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
24	24	3	2493	112	20	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
25	25	4	833	635	6	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
26	26	4	1432	1517	7	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
27	27	4	1767	1517	7	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
28	28	4	1975	817	9	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
29	29	4	2129	817	9	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
30	30	4	2246	770	28	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
31	31	4	2343	1045	20	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000
32	32	4	2426	1501	31	2	4	2023-12-10 16:04:51.0900000	2023-12-10 16:04:51.0900000

Query

Proposition: Which instructors are teaching classes, in multiple departments?

```
SELECT DI.InstructorID, I.FirstName, I.LastName
FROM Personnel.DepartmentInstructor AS DI
    INNER JOIN Personnel.Instructor AS I
        ON DI.InstructorID = I.InstructorID
GROUP BY DI.InstructorID,  I.FirstName, I.LastName
HAVING COUNT(DISTINCT DI.DepartmentID) > 1
ORDER BY DI.InstructorID;
```

Results | Messages

	InstructorID	FirstName	LastName
1	10	Dorian	Abreu
2	17	David	Adler
3	28	Christina	Alaimo
4	56	Yoshio	Aomori
5	59	Maria	Argyros
6	62	Alice	Artzt
7	119	David	Berkman
8	133	Brenda	Biddle
9	139	Jeffrey	Bird
10	150	Richard	Bodnar
11	188	Joshua	Brumberg
12	201	Harvey	Burstein
13	205	Fred	Cadieu
14	209	Peter	Calandra
15	220	Nora	Carr
16	223	Clare	Carroll
17	245	Riddhi	Chauhan
18	259	Yu	Chen
19	270	Euna	Cho
20	285	Yves	Cloarec
21	287	Joseph	Cohen
22	299	Lewis	Cook
23	306	Catherine	Cordeiro
24	314	Georges	Courtadon
25	315	Sarah	Covington
26	351	David	Debora
27	361	John	Dennehy
28	386	Antonio	Donato
29	393	Abigail	Doukhan
30	416	Omri	Elisha
31	421	Robert	Engel
32	434	Jeffrey	Farber

Query

Proposition: How many instructors are in each department?

```
SELECT D.DepartmentID, D.DepartmentName, COUNT(DI.InstructorID) AS NumberOfInstructors
FROM Academic.Department AS D
    LEFT JOIN Personnel.DepartmentInstructor AS DI
        ON D.DepartmentID = DI.DepartmentID
GROUP BY D.DepartmentID, D.DepartmentName
ORDER BY D.DepartmentID;
```

	DepartmentID	DepartmentName	NumberOfInstructors
1	1	ACCT	58
2	2	AFST	1
3	3	AMST	2
4	4	ANTH	32
5	5	ARAB	2
6	6	ARTH	12
7	7	ARTS	64
8	8	ASTR	3
9	9	BALA	10
10	10	BIOCH	2
11	11	BIOL	62
12	12	BUS	19
13	13	CERT	0
14	14	CESL	3
15	15	CHEM	50
16	16	CHIN	6
17	17	CLAS	6
18	18	CMAL	1
19	19	CMLIT	23
20	20	CO-OP	1
21	21	CSCI	53
22	22	CUNBA	0
23	23	DANCE	11
24	24	DRAM	22
25	25	EAST	8

Query

Proposition: How many classes are being taught this semester? Group by course and aggregating total enrollment, total class limit and the percentage of Enrollment.

```
SELECT DISTINCT
    C.CourseID, C.CourseName
    , SUM(DISTINCT CS.ClassID) AS TotalClasses
    , E.CurrentEnrollment AS CurrentEnrollment
    , E.MaxEnrollmentLimit AS MaxEnrollmentLimit
FROM
    ClassManagement.Class AS CS
        JOIN Academic.Course AS C
            ON CS.CourseID = C.CourseID
        LEFT JOIN Academic.Section AS S
            ON C.CourseID = S.CourseId
        LEFT JOIN Enrollment.EnrollmentDetail AS E
            ON S.SectionID = E.SectionID
GROUP BY
    C.CourseID, C.CourseName, E.CurrentEnrollment, E.MaxEnrollmentLimit
ORDER BY
    C.CourseID;
```

	CourseID	CourseName	TotalClasses	CurrentEnrollment	MaxEnrollmentLimit
1	1	Fin & Mgr Acct	6	20	22
2	1	Fin & Mgr Acct	6	21	22
3	1	Fin & Mgr Acct	6	22	22
4	2	Int Theo & Prac Acct 1	114	20	30
5	2	Int Theo & Prac Acct 1	114	29	55
6	2	Int Theo & Prac Acct 1	114	30	30
7	2	Int Theo & Prac Acct 1	114	39	55
8	2	Int Theo & Prac Acct 1	114	42	55
9	2	Int Theo & Prac Acct 1	114	45	45
10	2	Int Theo & Prac Acct 1	114	47	55
11	2	Int Theo & Prac Acct 1	114	54	55
12	2	Int Theo & Prac Acct 1	114	55	55
13	3	Intro Theo & Prac Acct 2	180	16	55
14	3	Intro Theo & Prac Acct 2	180	17	55
15	3	Intro Theo & Prac Acct 2	180	19	30
16	3	Intro Theo & Prac Acct 2	180	41	55
17	3	Intro Theo & Prac Acct 2	180	44	55
18	3	Intro Theo & Prac Acct 2	180	51	55
19	3	Intro Theo & Prac Acct 2	180	55	55
20	4	Inter Acct 1	228	29	30
21	4	Inter Acct 1	228	39	42
22	4	Inter Acct 1	228	40	42
23	4	Inter Acct 1	228	42	42
24	4	Inter Acct 1	228	43	46
25	4	Inter Acct 1	228	49	51

Query

Proposition: What courses are offered by a specific department? Include the courses names and credit hours.

```
DECLARE @DepartmentID INT;
-- Set the value of the specific department
SET @DepartmentID = 1;

SELECT @DepartmentID AS DepartmentID, D.DepartmentName
    , C.CourseId, C.CourseName, C.CreditHours
FROM Academic.Department AS D
    INNER JOIN Academic.Course AS C
        ON D.DepartmentID = C.DepartmentID
WHERE D.DepartmentID = @DepartmentID
```

Results | Messages

	DepartmentID	DepartmentName	CourseId	CourseName	CreditHours
1	1	ACCT	1	Fin & Mgr Acct	3
2	1	ACCT	2	Int Theo & Prac Acct 1	4
3	1	ACCT	3	Intro Theo & Prac Acct 2	4
4	1	ACCT	4	Inter Acct 1	4
5	1	ACCT	5	Inter Acct 2	3
6	1	ACCT	6	Business Law l	3
7	1	ACCT	7	Cost Acct	3
8	1	ACCT	8	Quant Techniq Pln & Ctrl	4
9	1	ACCT	9	Advcd Acct	4
10	1	ACCT	10	Auditing 1	4
11	1	ACCT	11	Auditing 2	3
12	1	ACCT	12	Acct Info Syst	3
13	1	ACCT	13	Microcomp Apps Acct	3
14	1	ACCT	14	Finan Stat Analys	3
15	1	ACCT	15	Business Law 2	3
16	1	ACCT	16	Business Law 2	3
17	1	ACCT	17	Business Law 3	3
18	1	ACCT	18	Fed & Ny State Tax	4
19	1	ACCT	19	Gov & Nfp Acct & Aud	3
20	1	ACCT	20	Seminar in Accounting	3
21	1	ACCT	21	Internship	3
22	1	ACCT	22	Issues In Mgmt Acct	3
23	1	ACCT	23	Adv Fin Acct Theory	3
24	1	ACCT	24	Adv Auditing Theory	3
25	1	ACCT	25	Comm & Accountants	3
26	1	ACCT	26	Adv Acct Info Systms	3
27	1	ACCT	27	Adv Stdy In Bus Law	3
28	1	ACCT	28	Taxation Of Bus Ents	3
29	1	ACCT	29	State & Local Taxatn	3
30	1	ACCT	30	Govt Acct & Audit	3

Sigalita Yakubova

To-do list

To be completed by:	Name Sigalita Yakubova
Deadline:	Date 12/6/2023

Project 1

% done	Phase	Start By	Original Due By	Revised Due By	Nu	Revision Notes
100%	Planning	12/4/2023	12/4/2023		1	
100%	Preparation	12/4/2023	12/4/2023		1	
100%	Assess the data and figure out a good database design to fit the data	12/5/2023	12/5/2023		1	
100%	Create the Semester Table	12/7/2023	12/7/2023		1	
100%	Create the Section Table	12/8/2023	12/8/2023		1	
100%	Create the LDM	12/9/2023	12/9/2023		1	
100%	Create 4 Queries	12/10/2023	12/10/2023		1	
0%	Record Presentation and make slides	12/10/2023	12/10/2023		1	
0%	Hand-off	12/10/2023	12/10/2023		1	

Semester

Creating the Table

```
DROP TABLE IF EXISTS [Enrollment].[Semester]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [Enrollment].[Semester]
(
    SemesterID [int] NOT NULL IDENTITY(1, 1), -- primary key
    SemesterName [Udt].SemesterName NULL,
    -- all tables must have the following 3 columns:
    [UserAuthorizationKey] [Udt].[SurrogateKeyInt] NOT NULL,
    [DateAdded] [Udt].[DateAdded] NOT NULL,
    [DateOfLastUpdate] [Udt].[DateOfLastUpdate] NOT NULL,
    PRIMARY KEY CLUSTERED(
    [SemesterID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_D
) ON [PRIMARY]
GO
```

Stored Procedure

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE OR ALTER PROCEDURE [Project3].[LoadSemesters] @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @DateAdded DATETIME2 = SYSDATETIME();
    DECLARE @StartingDateTime DATETIME2 = SYSDATETIME();

    --Right now this should only fill the table with Fall 2023 but in the future this will be a useful feature
    INSERT INTO [Enrollment].[Semester](
        SemesterName, UserAuthorizationKey, DateAdded
    )
    SELECT [Udt].GetSemesterName(@DateAdded), @UserAuthorizationKey, @DateAdded
    FROM
    [Uploadfile].[CurrentSemesterCourseOfferings]

    DECLARE @WorkFlowStepTableRowCount INT;
    SET @WorkFlowStepTableRowCount = 0;
    DECLARE @EndingDateTime DATETIME2 = SYSDATETIME();
    DECLARE @QueryTime BIGINT = CAST(DATEDIFF(MILLISECOND, @StartingDateTime, @EndingDateTime) AS bigint);
    EXEC [Process].[usp_TrackWorkFlow] 'Add Semester Data',
                                       @WorkFlowStepTableRowCount,
                                       @StartingDateTime,
                                       @EndingDateTime,
                                       @QueryTime,
                                       @UserAuthorizationKey;
END;
GO
```

Functions

```sql
-- Sigi
CREATE TYPE [Udt].SemesterName FROM NVARCHAR(20)
GO
```

```sql
-- Sigi
-- Create a function to determine the season
CREATE FUNCTION [Udt].GetSeason(@DateAdded DATETIME2)
RETURNS NVARCHAR(10)
AS
BEGIN
    DECLARE @Season NVARCHAR(10);

    SET @Season =
        CASE
            WHEN MONTH(@DateAdded) BETWEEN 1 AND 3 THEN 'Winter'
            WHEN MONTH(@DateAdded) BETWEEN 4 AND 6 THEN 'Spring'
            WHEN MONTH(@DateAdded) BETWEEN 7 AND 9 THEN 'Summer'
            WHEN MONTH(@DateAdded) BETWEEN 10 AND 12 THEN 'Fall'
        END;

    RETURN @Season;
END;
GO
```

```sql
-- Create a function to get the formatted SemesterName
CREATE FUNCTION [Udt].GetSemesterName(@DateAdded DATETIME2)
RETURNS [Udt].SemesterName
AS
BEGIN
    DECLARE @Season NVARCHAR(10);
    DECLARE @Year NVARCHAR(4);
    DECLARE @SemesterName [Udt].SemesterName;

    SET @Season = [Udt].GetSeason(@DateAdded);
    SET @Year = FORMAT(@DateAdded, 'yyyy');
    SET @SemesterName = @Season + ' ' + @Year;

    RETURN @SemesterName;
END;

GO
```

Section

Creating the Table

```
DROP TABLE IF EXISTS [Academic].[Section]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [Academic].[Section]
(
    SectionID INT NOT NULL IDENTITY(1, 1), -- primary key
    Section varchar(20) NOT NULL,
    Code varchar(20) NOT NULL,
    CourseID [int] NOT NULL, -- FOREIGN KEY (CourseID)
    -- all tables must have the following 3 columns:
    [UserAuthorizationKey] [Udt].[SurrogateKeyInt] NOT NULL,
    [DateAdded] [Udt].[DateAdded] NOT NULL,
    [DateOfLastUpdate] [Udt].[DateOfLastUpdate] NOT NULL,
    PRIMARY KEY CLUSTERED(
    [SectionId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_K
) ON [PRIMARY]
GO
```

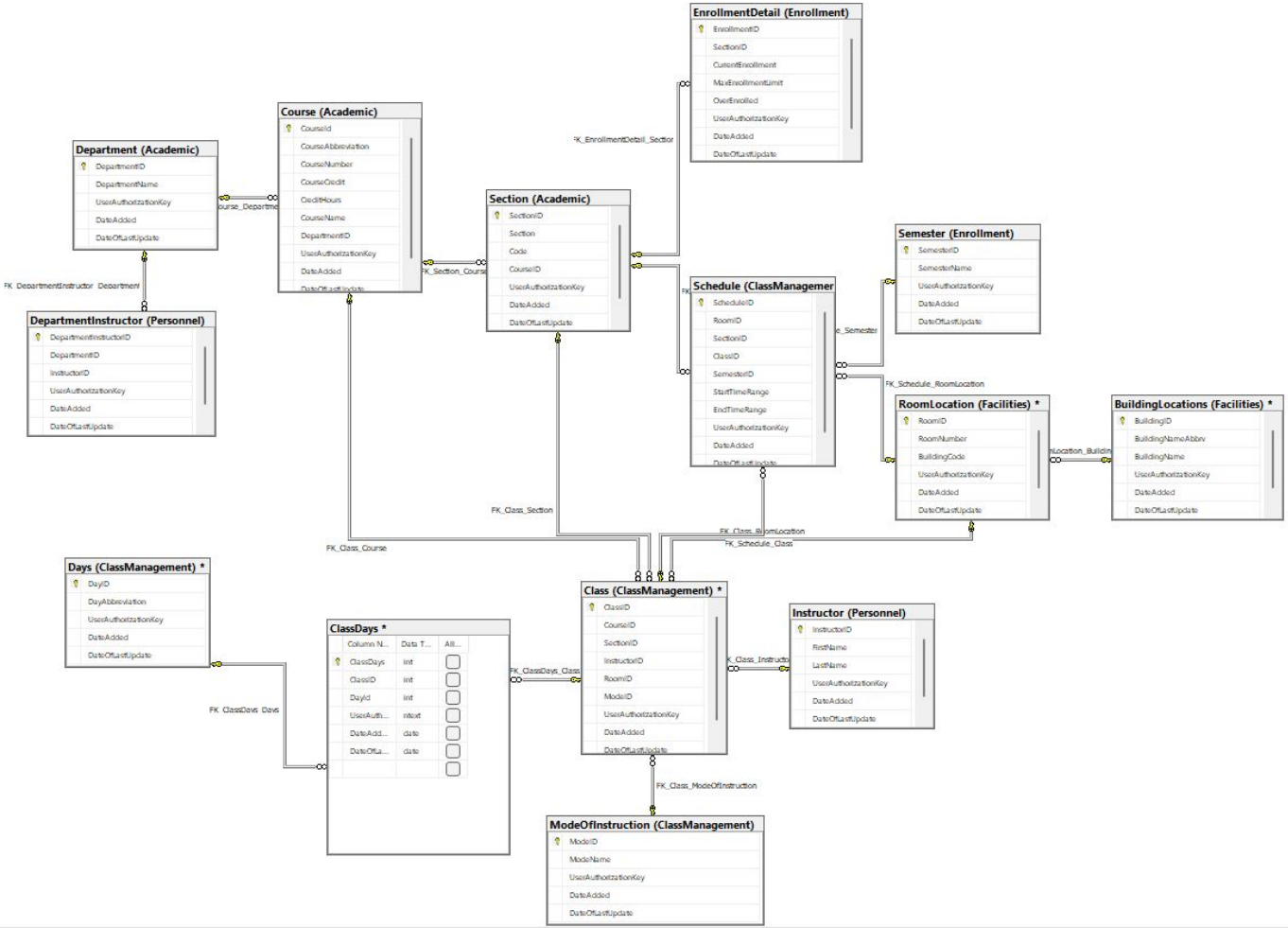
Stored Procedure

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE OR ALTER PROCEDURE [Project3].[LoadSections] @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @DateAdded DATETIME2 = SYSDATETIME();
    DECLARE @StartingDateTime DATETIME2 = SYSDATETIME();

    INSERT INTO [Academic].[Section] (Section, Code, CourseID, UserAuthorizationKey, DateAdded)
    SELECT
        Upload.Sec,
        Upload.Code,
        (
            SELECT TOP 1 C.CourseId
            FROM [Academic].[Course] AS C
            WHERE
                C.CourseAbbreviation = LEFT(Upload.[Course (hr, crd)], PATINDEX('%[ (]%', Upload.[Course (hr, crd)]) - 1) AND
                C.CourseNumber = SUBSTRING(
                    Upload.[Course (hr, crd)],
                    PATINDEX('%[0-9]%', Upload.[Course (hr, crd)]),
                    CHARINDEX('(', Upload.[Course (hr, crd)]) - PATINDEX('%[0-9]%', Upload.[Course (hr, crd)])
                )
        ) AS CourseID,
        @UserAuthorizationKey,
        @DateAdded
    FROM
        [Uploadfile].[CurrentSemesterCourseOfferings] AS Upload;

    DECLARE @WorkFlowStepTableRowCount INT;
    SET @WorkFlowStepTableRowCount = 0;
    DECLARE @EndingDateTime DATETIME2 = SYSDATETIME();
    DECLARE @QueryTime BIGINT = CAST(DATEDIFF(MILLISECOND, @StartingDateTime, @EndingDateTime) AS bigint);
    EXEC [Process].[usp_TrackWorkFlow] 'Add Section Data',
                                       @WorkFlowStepTableRowCount,
                                       @StartingDateTime,
                                       @EndingDateTime,
                                       @QueryTime
```

LDM


EnrollmentDetail (Enrollment)
EnrollmentID
SectionID
CurrentEnrollment
MaxEnrollmentLimit
OverEnrolled
UserAuthorizationKey
DateAdded
DateOfLastUpdate
Course (Academic)
CourseId
CourseAbbreviation
CourseNumber
CourseCredit
CreditHours
CourseName
DepartmentID
UserAuthorizationKey
DateAdded
Department (Academic)
DepartmentID
DepartmentName
UserAuthorizationKey
DateAdded
DateOfLastUpdate
Section (Academic)
SectionID
Section
Code
CourseID
UserAuthorizationKey
DateAdded
DateOfLastUpdate
Semester (Enrollment)
SemesterID
SemesterName
UserAuthorizationKey
DateAdded
DateOfLastUpdate
Schedule (ClassManagemer
ScheduleID
RoomID
SectionID
ClassID
SemesterID
StartTimeRange
EndTimeRange
UserAuthorizationKey
DateAdded
DateOfLastUpdate
DepartmentInstructor (Personnel)
DepartmentInstructorID
DepartmentID
InstructorID
UserAuthorizationKey
DateAdded
DateOfLastUpdate
RoomLocation (Facilities) *
RoomID
RoomNumber
BuildingCode
UserAuthorizationKey
DateAdded
DateOfLastUpdate
BuildingLocations (Facilities) *
BuildingID
BuildingNameAbbrv
BuildingName
UserAuthorizationKey
DateAdded
DateOfLastUpdate
Days (ClassManagement) *
DayID
DayAbbreviation
UserAuthorizationKey
DateAdded
DateOfLastUpdate
ClassDays *
Column N...
Data T...
All...
ClassDays
int
ClassID
int
DayId
int
UserAuth...
ntext
DateAdd...
date
DateOfLa...
date
Class (ClassManagement) *
ClassID
CourseID
SectionID
InstructorID
RoomID
ModeID
UserAuthorizationKey
DateAdded
DateOfLastUpdate
Instructor (Personnel)
InstructorID
FirstName
LastName
UserAuthorizationKey
DateAdded
DateOfLastUpdate
ModeOfInstruction (ClassManagement)
ModeID
ModeName
UserAuthorizationKey
DateAdded
DateOfLastUpdate
FK_EnrollmentDetail_Sectior
FK_Section_Course
FK_Schedule_RoomLocation
FK_Class_Section
FK_Class_Course
FK_Class_RoomLocation
FK_Schedule_Class
FK_ClassDays_Class
FK_ClassDays_Days
FK_Class_ModeOfInstruction

Query 1

```
57
58  /* ============================================
59  -- Author:      Sigalita Yakubova
60  -- Create date: 12/10/23
61  -- Proposition: Show all the courses being taught in a specific department
62  -- ============================================*/
63  DECLARE  @DeptID INT = 1;
64
65  SELECT D.DepartmentID, D.DepartmentName, C.CourseName, C.CourseAbbreviation, C.CourseNumber, C.CourseId
66  FROM [Academic].Department AS D
67  INNER JOIN [Academic].Course AS C ON C.DepartmentID = D.DepartmentID
68  WHERE D.DepartmentID = @DeptID
```

Results | Messages

	DepartmentID	DepartmentName	CourseName	CourseAbbreviation	CourseNumber	CourseId
1	1	ACCT	Fin & Mgr Acct	ACCT	100	1
2	1	ACCT	Int Theo & Prac Acct 1	ACCT	101	2
3	1	ACCT	Intro Theo & Prac Acct 2	ACCT	102	3
4	1	ACCT	Inter Acct 1	ACCT	201	4
5	1	ACCT	Inter Acct 2	ACCT	202	5
6	1	ACCT	Business Law 1	ACCT	261	6
7	1	ACCT	Cost Acct	ACCT	305	7
8	1	ACCT	Quant Techniq Pln & Ctrl	ACCT	306	8

Query 2

```
71  /* ============================================
72  -- Author:      Sigalita Yakubova
73  -- Create date: 12/10/23
74  -- Proposition: Show all teachers in the Accounting Department whose first name starts with A
75  -- ============================================*/
76
77  SELECT D.DepartmentID, D.DepartmentName, I.FirstName, I.LastName
78  FROM [Personnel].DepartmentInstructor AS DI
79  INNER JOIN [Personnel].Instructor AS I ON I.InstructorID = DI.InstructorID
80  INNER JOIN [Academic].Department AS D ON D.DepartmentID = DI.DepartmentID
81  WHERE D.DepartmentName = 'ACCT' AND I.FirstName LIKE 'A%'
82
83
```

Results Messages

DepartmentID	DepartmentName	FirstName	LastName
1	ACCT	Arthur	Adelberg
1	ACCT	Amy	David
1	ACCT	Arthur	Dignam
1	ACCT	Anita	Feisullin
1	ACCT	Anique	Qureshi
1	ACCT	Arthur	Silverman

Query 3

```
138
139  /* ============================================
140  -- Author:      Sigalita Yakubova
141  -- Create date: 12/10/23
142  -- Proposition: Identify instructors who are not assigned to any class
143  -- ============================================*/
144  --Empty because all instructors included in the data are teaching this semester
145  SELECT
146      I.InstructorID,
147      I.LastName,
148      I.FirstName
149  FROM
150      [Personnel].Instructor AS I
151  WHERE
152      NOT EXISTS (
153          SELECT 1
154          FROM ClassManagement.Class AS C
155          WHERE C.InstructorID = I.InstructorID
156      )
157  ORDER BY I.LastName
158
159  /* ============================================
```

Results | Messages

Instruc…	LastName	FirstName

Query 4

```sql
/* ========================================
-- Author:      Sigalita Yakubova
-- Create date: 12/10/23
-- Proposition: Most Popular Courses: Rank courses based on total enrollment for the semester, showing the most to least popular
-- ========================================*/

WITH CourseEnrollment AS (
    SELECT
        C.CourseID,
        C.CourseName,
        ED.CurrentEnrollment,
        ED.MaxEnrollmentLimit
    FROM
        [Academic].Section AS S
    INNER JOIN [Academic].Course AS C ON S.CourseID = C.CourseID
    INNER JOIN Enrollment.EnrollmentDetail AS ED ON ED.SectionID = S.SectionID
    GROUP BY
        C.CourseID, C.CourseName, ED.CurrentEnrollment, ED.MaxEnrollmentLimit
)

SELECT
    CE.CourseID,
    CE.CourseName,
    CE.CurrentEnrollment,
    CE.MaxEnrollmentLimit,
    RANK() OVER (ORDER BY
        CASE
            WHEN CE.MaxEnrollmentLimit > 0 THEN CE.CurrentEnrollment * 1.0 / CE.MaxEnrollmentLimit
            ELSE 0 -- Handle the case where MaxEnrollmentLimit is 0
        END DESC
    ) AS EnrollmentRank
FROM
    CourseEnrollment CE;
```

CourseName	CurrentEnrollment	MaxEnrollmentLimit
Spec Proj Painting	5	1
Spec Proj Drawing	5	1
Spec Proj Ceramics	5	1

Basic Biochem Lab	13	18
Special Topics	13	18
Principles of Genetics	13	18

Ahnaf Ahmed

To-do list

To be completed by:		**Name: Ahnaf Ahmed**
Deadline:	**12/10/2023**	**Date: 12/09/2023**

Project 3

% done	Phase	Start By	Original Due By2	Revised Due By	Number Of Days	Revision Notes
100%	Preparation	11/29/2023	12/1/2023		2	
100%	Group Meeting	12/1/2023	12/1/2023		1	
100%	Planning	12/1/2023	12/4/2023		3	
100%	Follow-up Group Meeting	12/4/2023	12/4/2023		1	
100%	Tier 1 tables: Days table	12/4/2023	12/5/2023		2	
100%	Tier 3 tables: EnrollmentDetail table	12/5/2023	12/7/2023		3	
100%	Queries	12/8/2023	12/9/2023		2	
100%	Create Slides	12/9/2023	12/9/2023		1	
100%	Record Presentation	12/9/2023	12/9/2023		1	
0%	Submit	12/9/2023	12/10/2023		1	

Table: Days

```
DROP TABLE IF EXISTS [ClassManagement].[Days]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [ClassManagement].[Days]
(
    [DayID] [int] NOT NULL IDENTITY(1, 1), -- primary key
    [DayAbbreviation] [Udt].[DayOfWeek] NOT NULL,
    -- all tables must have the following 3 columns:
    [UserAuthorizationKey] [Udt].[SurrogateKeyInt] NOT NULL,
    [DateAdded] [Udt].[DateAdded] NOT NULL,
    [DateOfLastUpdate] [Udt].[DateOfLastUpdate] NOT NULL,
    PRIMARY KEY CLUSTERED(
    [DayID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
) ON [PRIMARY]
GO
```

```
CREATE TYPE [Udt].[DayOfWeek] FROM CHAR(2) NULL
GO
```

Constraints

```
ALTER TABLE [ClassManagement].[Days] ADD  DEFAULT (sysdatetime()) FOR [DateAdded]
GO
ALTER TABLE [ClassManagement].[Days] ADD  DEFAULT (sysdatetime()) FOR [DateOfLastUpdate]
GO
```

```
ALTER TABLE [ClassManagement].[Days]  WITH CHECK ADD  CONSTRAINT [FK_Days_UserAuthorization] FOREIGN KEY([UserAuthorizationKey])
REFERENCES [DbSecurity].[UserAuthorization] ([UserAuthorizationKey])
GO
ALTER TABLE [ClassManagement].[Days] CHECK CONSTRAINT [FK_Days_UserAuthorization]
GO
ALTER TABLE [ClassManagement].[Days] ADD CONSTRAINT CHK_DayOfWeek
CHECK (DayAbbreviation IN ('M', 'T', 'W', 'TH', 'F', 'S', 'SU'))
GO
```

Stored Procedure: LoadDays

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE OR ALTER PROCEDURE [Project3].[LoadDays] @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @DateAdded DATETIME2 = SYSDATETIME();
    DECLARE @StartingDateTime DATETIME2 = SYSDATETIME();

    INSERT INTO [ClassManagement].[Days](
        [DayAbbreviation], [UserAuthorizationKey]
    )
    VALUES

        ('M', @UserAuthorizationKey),
        ('T', @UserAuthorizationKey),
        ('W', @UserAuthorizationKey),
        ('TH', @UserAuthorizationKey),
        ('F', @UserAuthorizationKey),
        ('S', @UserAuthorizationKey),
        ('SU', @UserAuthorizationKey)

    DECLARE @WorkFlowStepTableRowCount INT;
    SET @WorkFlowStepTableRowCount = 7;
    DECLARE @EndingDateTime DATETIME2 = SYSDATETIME();
    DECLARE @QueryTime BIGINT = CAST(DATEDIFF(MILLISECOND, @StartingDateTime, @EndingDateTime) AS bigint);
    EXEC [Process].[usp_TrackWorkFlow] 'Procedure: [Project3].[LoadDays] loads DayAbbreviation into the [Days] table',
                                       @WorkFlowStepTableRowCount,
                                       @StartingDateTime,
                                       @EndingDateTime,
                                       @QueryTime,
                                       @UserAuthorizationKey;
END;
GO
```

```
EXEC [Project3].[LoadDays] @UserAuthorizationKey = 5
```

Table Data

	DayID	DayAbbreviation	UserAuthorizationKey	DateAdded	DateOfLastUpdate
1	1	M	5	2023-12-10 01:57:15.6200000	2023-12-10 01:57:15.6200000
2	2	T	5	2023-12-10 01:57:15.6200000	2023-12-10 01:57:15.6200000
3	3	W	5	2023-12-10 01:57:15.6200000	2023-12-10 01:57:15.6200000
4	4	TH	5	2023-12-10 01:57:15.6200000	2023-12-10 01:57:15.6200000
5	5	F	5	2023-12-10 01:57:15.6200000	2023-12-10 01:57:15.6200000
6	6	S	5	2023-12-10 01:57:15.6200000	2023-12-10 01:57:15.6200000
7	7	SU	5	2023-12-10 01:57:15.6200000	2023-12-10 01:57:15.6200000

(7 rows affected)

Table: EnrollmentDetail

```
DROP TABLE IF EXISTS [Enrollment].[EnrollmentDetail]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [Enrollment].[EnrollmentDetail]
(
    [EnrollmentID] INT NOT NULL IDENTITY(1, 1), -- primary key
    [SectionID] INT NOT NULL, -- Foreign Key (SectionID)
    [CurrentEnrollment] INT NOT NULL,
    [MaxEnrollmentLimit] INT NOT NULL,
    [OverEnrolled] NCHAR(3),
    -- all tables must have the following 3 columns:
    [UserAuthorizationKey] [Udt].[SurrogateKeyInt] NOT NULL,
    [DateAdded] [Udt].[DateAdded] NOT NULL,
    [DateOfLastUpdate] [Udt].[DateOfLastUpdate] NOT NULL,
    PRIMARY KEY CLUSTERED(
    [EnrollmentID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY
) ON [PRIMARY]
GO
```

Constraints

```
ALTER TABLE [Enrollment].[EnrollmentDetail] ADD  DEFAULT (sysdatetime()) FOR [DateAdded]
GO
ALTER TABLE [Enrollment].[EnrollmentDetail] ADD  DEFAULT (sysdatetime()) FOR [DateOfLastUpdate]
GO


ALTER TABLE [Enrollment].[EnrollmentDetail]  WITH CHECK ADD  CONSTRAINT [FK_EnrollmentDetail_UserAuthorization] FOREIGN KEY([UserAuthorizationKey])
REFERENCES [DbSecurity].[UserAuthorization] ([UserAuthorizationKey])
GO
ALTER TABLE [Enrollment].[EnrollmentDetail] CHECK CONSTRAINT [FK_EnrollmentDetail_UserAuthorization]
GO
ALTER TABLE [Enrollment].[EnrollmentDetail] WITH CHECK ADD CONSTRAINT [FK_EnrollmentDetail_Section] FOREIGN KEY([SectionID])
REFERENCES [Academic].[Section] ([SectionID])
GO
ALTER TABLE [Enrollment].[EnrollmentDetail] CHECK CONSTRAINT [FK_EnrollmentDetail_Section]
GO
```

Stored Procedure: LoadEnrollmentDetail

```sql
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE OR ALTER PROCEDURE [Project3].[LoadEnrollmentDetail] @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @DateAdded DATETIME2 = SYSDATETIME();
    DECLARE @StartingDateTime DATETIME2 = SYSDATETIME();

    INSERT INTO [Enrollment].[EnrollmentDetail]
    (SectionID,
        CurrentEnrollment,
        MaxEnrollmentLimit,
        OverEnrolled,
        UserAuthorizationKey,
        DateAdded)
    SELECT DISTINCT
        S.SectionID,
        CAST(Upload.Enrolled AS INT),
        CAST(Upload.Limit AS INT),
        CASE
            WHEN CAST(Upload.Enrolled AS INT) <= CAST(Upload.Limit AS INT) THEN 'No'
            ELSE 'Yes'
        END,
        @UserAuthorizationKey,
        @DateAdded
    FROM
        [Uploadfile].[CurrentSemesterCourseOfferings] AS Upload
        INNER JOIN [Academic].[Section] AS S
        ON Upload.Code = S.Code

    DECLARE @WorkFlowStepTableRowCount INT;
    SET @WorkFlowStepTableRowCount = 0;
    DECLARE @EndingDateTime DATETIME2 = SYSDATETIME();
    DECLARE @QueryTime BIGINT = CAST(DATEDIFF(MILLISECOND, @StartingDateTime, @EndingDateTime) AS bigint);
    EXEC [Process].[usp_TrackWorkFlow] 'Procedure: [Project3].[LoadEnrollmentDetail] loads [EnrollmentDetail] table',
                                       @WorkFlowStepTableRowCount,
                                       @StartingDateTime,
                                       @EndingDateTime,
                                       @QueryTime,
                                       @UserAuthorizationKey;
END;
GO
```

```sql
EXEC [Project3].[LoadEnrollmentDetail] @UserAuthorizationKey = 5
```

Table Data

	EnrollmentID	SectionID	CurrentEnrollment	MaxEnrollmentLimit	OverEnrolled	UserAuthorizationKey	DateAdded	DateOfLastUpdate
1	1	1	42	50	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
2	2	2	9	50	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
3	3	3	26	26	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
4	4	4	26	26	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
5	5	5	26	26	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
6	6	6	24	24	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
7	7	7	20	24	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
8	8	8	25	24	Yes	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
9	9	9	26	26	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
10	10	10	30	26	Yes	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
11	11	11	22	24	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
12	12	12	23	24	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
13	13	13	18	16	Yes	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
14	14	14	24	24	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
15	15	15	12	14	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
16	16	16	19	20	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
17	17	17	4	15	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
18	18	18	3	10	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
19	19	19	5	4	Yes	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
20	20	20	8	16	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
21	21	21	7	15	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
22	22	22	7	15	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
23	23	23	0	1	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
24	24	24	0	10	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333
25	25	25	26	26	No	5	2023-12-10 01:57:35.2633333	2023-12-10 01:57:35.2633333

(4187 rows affected)

Query

Proposition: Show the number of students enrolled in each course.
Display the course, course name, and total enrollment.

```
SELECT CONCAT(C.CourseAbbreviation, C.CourseNumber) AS [Course],
       C.CourseName,
       SUM(E.CurrentEnrollment) AS [TotalEnrollment]
FROM Academic.Course AS C
    INNER JOIN Academic.Section AS S
       ON C.CourseId = S.CourseID
    INNER JOIN Enrollment.EnrollmentDetail AS E
       ON S.SectionID = E.SectionID
GROUP BY C.CourseAbbreviation, C.CourseNumber, C.CourseName
```

Result Table

	Course	CourseName	TotalEnrollment
1	ACCT 100	Fin & Mgr Acct	63
2	ACCT 101	Int Theo & Prac Acct 1	500
3	ACCT 102	Intro Theo & Prac Acct 2	342
4	ACCT 201	Inter Acct 1	311
5	ACCT 202	Inter Acct 2	289
6	ACCT 261	Business Law l	352
7	ACCT 305	Cost Acct	296
8	ACCT 306	Quant Techniq Pln & Ctrl	267
9	ACCT 311	Advcd Acct	263
10	ACCT 321	Auditing 1	193

NACE

- Communication: Enhanced verbal and written communication skills, fostering clear and concise messages to interact effectively with others.
- Technology: Acquired proficiency in relevant technologies, leveraging them to improve efficiency and productivity in job-related tasks.
- Critical Thinking: Developed analytical and problem-solving abilities to approach challenges with a systematic and innovative mindset.

THANK YOU!

Nicholas Kong

To be completed by:		Nicholas Kong
Deadline:	12/10/2023	11/30/23

Project 1

% done	Phase	Start By	Original Due By	Revised Due By	Nu	Revision Notes
100%	Review Specifications for 1st Team Meeting	11/30/23	11/30/2023	Extension by Sunday 12/10/23		
100%	1st Team Meeting	12/1/23	12/1/23			
100%	Model Entity Relationships -TextFile in Shared Drive	12/2/23	12/3/23			
100%	2nd Team Meeting	12/4/23	12/4/23			
100%	Implement Mode Of Instruction Table	12/4/23	12/5/23			
100%	Implement Room Location Table	12/5/23	12/6/23			
100%	Implement Schedule Table	12/6/23	12/9/23			
100%	Presentation	12/10/23	12/10/23			
100%	Recording WorkFlowStep	12/10/23	12/10/23			

Mode Of Instruction

```
IF EXISTS [ClassManagement].[ModeOfInstruction] GO
    SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [ClassManagement].[ModeOfInstruction] (
    ModeID INT IDENTITY(1, 1) NOT NULL
    ,ModeName NVARCHAR(12) NOT NULL
    ,
    -- all tables must have the following 3 columns:
    [UserAuthorizationKey] [Udt].[SurrogateKeyInt] NOT NULL
    ,[DateAdded] [Udt].[DateAdded] NOT NULL
    ,[DateOfLastUpdate] [Udt].[DateOfLastUpdate] NOT NULL
    ,PRIMARY KEY CLUSTERED ([ModeID] ASC) WITH (
        PAD_INDEX = OFF
        ,STATISTICS_NORECOMPUTE = OFF
        ,IGNORE_DUP_KEY = OFF
        ,ALLOW_ROW_LOCKS = ON
        ,ALLOW_PAGE_LOCKS = ON
        ,OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF
        ) ON [PRIMARY]
    ) ON [PRIMARY]
GO
```

```
CREATE
    OR

ALTER PROCEDURE [Project3].[LoadModeOfInstruction]
    -- Add parameters if needed
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @DateAdded DATETIME2 = SYSDATETIME();
    DECLARE @DateOfLastUpdate DATETIME2 = SYSDATETIME();
    DECLARE @StartingDateTime DATETIME2 = SYSDATETIME();
    DECLARE @WorkFlowStepTableRowCount INT = 0;

    -- INSERT INTO [Enrollment].[Semester](
    --      SemesterName, UserAuthorizationKey, DateAdded
    -- )
    -- SELECT [Udt].GetSemesterName(@DateAdded), @UserAuthorizationKey, @DateAdded
    INSERT INTO ClassManagement.ModeOfInstruction (
        ModeName
        ,UserAuthorizationKey
        ,DateAdded
        )
    SELECT DISTINCT Q.[Mode of Instruction]
        ,@UserAuthorizationKey
        ,@DateAdded
    FROM [QueensClassSchedule].[Uploadfile].[CurrentSemesterCourseOfferings] AS Q

    -- Additional statements or constraints can be added here
    DECLARE @EndingDateTime DATETIME2 = SYSDATETIME();
    DECLARE @QueryTime BIGINT = CAST(DATEDIFF(MILLISECOND, @StartingDateTime, @EndingDateTime) AS BIGINT);

    EXEC [Process].[usp_TrackWorkFlow] 'Procedure: Project3[LoadModeOfInstruction] loads data into ShowTableStatusRowCount'
        ,@WorkFlowStepTableRowCount
        ,@StartingDateTime
        ,@EndingDateTime
        ,@QueryTime
        ,@UserAuthorizationKey;
END;
GO
```

Room Location

```
DROP TABLE

IF EXISTS [Facilities].[RoomLocation] GO
    SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [Facilities].[RoomLocation] (
    RoomID INT IDENTITY(1, 1) NOT NULL
    ,RoomNumber VARCHAR(12) NULL
    ,BuildingCode INT
    ,-- Assuming BuildingCode is INT; adjust the data type as needed
    -- FOREIGN KEY (BuildingCode) REFERENCES BuildingLocation(BuildingCode),
    -- all tables must have the following 3 columns:
    [UserAuthorizationKey] [Udt].[SurrogateKeyInt] NOT NULL
    ,[DateAdded] [Udt].[DateAdded] NOT NULL
    ,[DateOfLastUpdate] [Udt].[DateOfLastUpdate] NOT NULL
    ,PRIMARY KEY CLUSTERED ([RoomID] ASC) WITH (
        PAD_INDEX = OFF
        ,STATISTICS_NORECOMPUTE = OFF
        ,IGNORE_DUP_KEY = OFF
        ,ALLOW_ROW_LOCKS = ON
        ,ALLOW_PAGE_LOCKS = ON
        ,OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF
        ) ON [PRIMARY]
    ) ON [PRIMARY]
GO
```

```sql
CREATE OR ALTER PROCEDURE [Project3].[LoadRoomLocation]
    -- Add parameters if needed
    @UserAuthorizationKey INT
AS
BEGIN

    SET NOCOUNT ON;

    DECLARE @DateAdded DATETIME2 = SYSDATETIME();
    DECLARE @DateOfLastUpdate DATETIME2 = SYSDATETIME();
    DECLARE @StartingDateTime DATETIME2 = SYSDATETIME();
    DECLARE @WorkFlowStepTableRowCount INT = 0;

    INSERT INTO [Facilities].[RoomLocation](RoomNumber,UserAuthorizationKey, DateAdded)
    SELECT
    CASE
        -- add the edge cases and then manually set it correctly
        WHEN RIGHT(Q.Location, 4) = 'H 17' THEN '17'
        WHEN RIGHT(Q.Location, 4) = '135H' THEN 'A135H'
        WHEN RIGHT(Q.Location, 4) = '135B' THEN 'A135B'
        WHEN RIGHT(Q.Location, 4) = 'H 09' THEN '09'
        WHEN RIGHT(Q.Location, 4) = 'H 12' THEN '12'

        -- checks for null and empty string, if so set default string named TBD
       WHEN Q.Location IS NULL OR LTRIM(RTRIM(Q.Location)) = '' THEN 'TBD'
        ELSE RIGHT(Q.Location, 4)
    END AS RoomNumber, @UserAuthorizationKey, @DateAdded
    FROM [QueensClassSchedule].[Uploadfile].[CurrentSemesterCourseOfferings] as Q

    -- Additional statements or constraints can be added here

    DECLARE @EndingDateTime DATETIME2 = SYSDATETIME();
    DECLARE @QueryTime BIGINT = CAST(DATEDIFF(MILLISECOND, @StartingDateTime, @EndingDateTime) AS bigint);
    EXEC [Process].[usp_TrackWorkFlow] 'Procedure: Project3[LoadRoomLocation] loads data into ShowTableStatusRowCoun
                                       @WorkFlowStepTableRowCount,
                                       @StartingDateTime,
                                       @EndingDateTime,
```

Schedule

```
IF EXISTS [ClassManagement].[Schedule] GO
    SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [ClassManagement].[Schedule] (
    ScheduleID INT IDENTITY(1, 1) NOT NULL
    ,RoomID INT NULL
    ,SectionID INT NULL
    ,ClassID INT NULL
    ,SemesterID INT NULL
    ,StartTimeRange [Udt].[ClassTime] NOT NULL CHECK (
        StartTimeRange >= '00:00:00.0000000'
        AND StartTimeRange <= '24:00:00.0000000'
        )
    ,EndTimeRange [Udt].[ClassTime] NOT NULL CHECK (
        EndTimeRange >= '00:00:00.0000000'
        AND EndTimeRange <= '24:00:00.0000000'
        )
    ,
    -- all tables must have the following 3 columns:
    [UserAuthorizationKey] [Udt].[SurrogateKeyInt] NOT NULL
    ,[DateAdded] [Udt].[DateAdded] NOT NULL
    ,[DateOfLastUpdate] [Udt].[DateOfLastUpdate] NOT NULL
    ,PRIMARY KEY CLUSTERED ([ScheduleID] ASC) WITH (
        PAD_INDEX = OFF
        ,STATISTICS_NORECOMPUTE = OFF
        ,IGNORE_DUP_KEY = OFF
        ,ALLOW_ROW_LOCKS = ON
        ,ALLOW_PAGE_LOCKS = ON
        ,OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF
        ) ON [PRIMARY]
    ) ON [PRIMARY]
GO
```

```
CREATE
    OR

ALTER PROCEDURE [Project3].[LoadSchedule]
    --Add parameters if needed
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @DateAdded DATETIME2 = SYSDATETIME();
    DECLARE @DateOfLastUpdate DATETIME2 = SYSDATETIME();
    DECLARE @StartingDateTime DATETIME2 = SYSDATETIME();
    DECLARE @WorkFlowStepTableRowCount INT = 0;

    INSERT INTO [ClassManagement].[Schedule] (
        StartTimeRange
        ,EndTimeRange
        ,UserAuthorizationKey
        ,DateAdded
        )
    SELECT CONVERT(TIME, NULLIF(LEFT(Q.TIME, CHARINDEX('-', Q.TIME) - 1), 'TBD'), 108) AS ConvertedStartTime
        ,CONVERT(TIME, NULLIF(RIGHT(Q.TIME, LEN(Q.TIME) - CHARINDEX('-', Q.TIME)), 'TBD'), 108) AS ConvertedEndTime
        ,@UserAuthorizationKey
        ,@DateAdded
    FROM [QueensClassSchedule].[Uploadfile].[CurrentSemesterCourseOfferings] AS Q;

    -- Additional statements or constraints can be added here
    DECLARE @EndingDateTime DATETIME2 = SYSDATETIME();
    DECLARE @QueryTime BIGINT = CAST(DATEDIFF(MILLISECOND, @StartingDateTime, @EndingDateTime) AS BIGINT);

    EXEC [Process].[usp_TrackWorkFlow] 'Procedure: Project3[LoadSchedule] loads data into ShowTableStatusRowCount'
        ,@WorkFlowStepTableRowCount
        ,@StartingDateTime
        ,@EndingDateTime
        ,@QueryTime
        ,@UserAuthorizationKey;
END;
GO
```

```
/* ================================================================================
   Author: Nicholas Kong
   Create date: 12/10/23
   Proposition: Show the same instructor who teaches from different deparments
================================================================================
*/

SELECT D.DepartmentID, D.DepartmentName, I.FirstName, I.LastName
FROM [Personnel].DepartmentInstructor AS DI
INNER JOIN [Personnel].Instructor AS I ON I.InstructorID = DI.InstructorID
INNER JOIN [Academic].Department AS D ON D.DepartmentID = DI.DepartmentID
WHERE I.FirstName = 'David' AND I.LastName = 'Lahti'


---------------------------------------- EXEC COMMANDS TO MANAGE THE DB ----------
```

Results | Messages

DepartmentID	DepartmentName	FirstName	LastName
11	BIOL	David	Lahti
45	HMNS	David	Lahti

NACE Competencies

Teamwork
Critical Thinking
Technology
Communication