Project Presentation

Database Design and Implementation

Loyd Flores | Roni MIkhaylov | Mohammed Hasan
Yasir Jawwad Shoeb | Kishan Shah | Zhihan Chen

Presentation Agenda

I. NACE Competencies
II. Project Management
    A. Tools used
    B. ToDo / Gant Chart
    C. Naming Conventions
III. Project Specifications
    A. Task
    B. Design / Normalization of each Table
    C. Anomalies / Discoveries
IV. Stored Procedure Design
V. Closing Remarks & Introduction to Implementation (JDBC & Queries)

I. Nace Competencies

1 CRITICAL THINKING PROBLEM SOLVING

2 ORAL AND WRITTEN COMMUNICATIONS

3 TEAMWORK COLLABORATION

4 DIGITAL TECHNOLOGY

CAREER READINESS COMPETENCIES

ADOPTED FROM THE NACE CAREER READINESS COMPETENCIES FRAMEWORK

5 LEADERSHIP

6 PROFESSIONALISM WORK ETHIC

7 CAREER MANAGEMENT

8 GLOBAL INTERCULTURAL FLUENCY

II. Project Management

Project Management

Gant Project Planner Group #3

To-do list

DISCORD

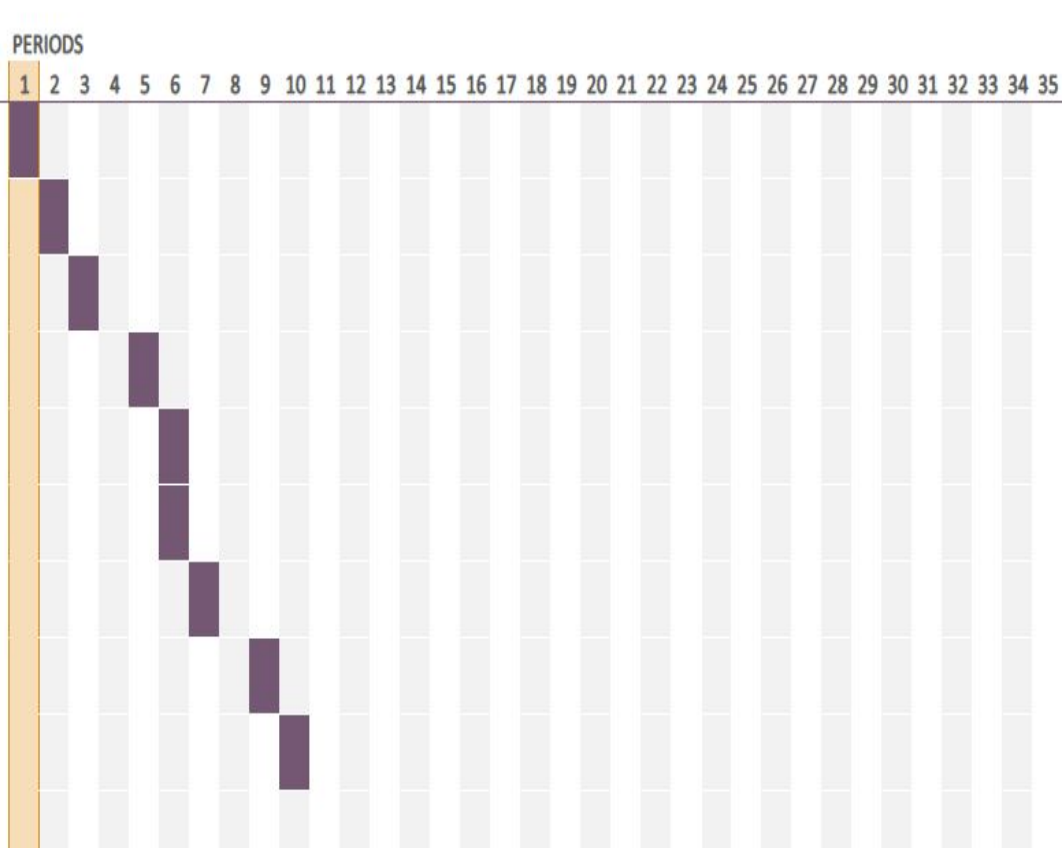
Gant Project Planner Group #3

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

Period Highlight: 1

Plan Duration | Actual Start | % Complete | Actual (beyond plan) | % Complete (beyond plan)

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Through Project specs Regarding	1	1	1	1	100%
database design tasks to groupmates	2	1	2	1	100%
	3	1	3	1	100%
Work on tables Compilation of code	5	1	5	1	100%
tasks to groupmates description	6	1	6	1	100%
meeting	6	1	6	1	100%
Finish Queries Compilation and Recording	7	1	7	1	100%
	9	1	9	1	100%
	10	1	10	1	100%



To-do list

To be completed by:	12/10/2023	Name	Loyd Flores
Deadline:	12/10/2023	Date	

Project 1

% done	Phase	Start By	Original Due By	Revised Due By	Number Of Days	Revision Notes
100%	Group reading of project specifications	12/1/2023	12/1/2023	n/a	1 day	Manage Ambiguity
100%	Discussion and task dissemenation	12/2/2023	12/2/2023	n/a	1 day	Manage Ambiguity
100%	Work on assigned tables	3-Dec	3-Dec	n/a	1 day	Manage Ambiguity
100%	tables and assist other members	4-Dec	4-Dec	n/a	1 day	& Stress management
100%	Compilation of work	5-Dec	5-Dec	n/a	1 day	Leadership, collaboration
100%	Complete Assigned Views and tables	6-Dec	8-Nov		1 day	Managing Ambiguitiy
100%	Check up on queries	8-Dec	8-Dec		1 day	stress management
100%	presentation and submission	9-Nov	9-Nov		1 day	stress management

III. Project Specifications

Task Given : Design a Database out from a file

	Semester	Sec	Code	Course (hr, crd)	Description	Day	Time	Instructor	Location	Enrolled	Limit	Mode of Instruction
1	Current Semester	02	37366	ACCT 100 (3, 3)	Fin & Mgr Acct	T, TH	3:10 PM - 4:25 PM	Milo, Michael	KY 419	20	22	In-Person
2	Current Semester	03	37823	ACCT 100 (3, 3)	Fin & Mgr Acct	M	3:10 PM - 6:00 PM	Ho, Vivian	HH 17	21	22	In-Person
3	Current Semester	01	37365	ACCT 100 (3, 3)	Fin & Mgr Acct	T, TH	10:45 AM - 12:00 PM	Milo, Michael	KY 419	22	22	In-Person
4	Current Semester	06	7351	ACCT 101 (4, 3)	Int Theo & Prac Acct 1	T, TH	12:10 PM - 2:00 PM	Feisullin, Anita	RA 201	30	30	In-Person
5	Current Semester	12	7357	ACCT 101 (4, 3)	Int Theo & Prac Acct 1	SU	8:20 AM - 12:00 PM	Mintz, Chana	PH 204	39	55	In-Person
6	Current Semester	11	7356	ACCT 101 (4, 3)	Int Theo & Prac Acct 1	S	8:20 AM - 12:00 PM	Chan, Joseph	PH 110	54	55	In-Person
7	Current Semester	10	7355	ACCT 101 (4, 3)	Int Theo & Prac Acct 1	F	6:30 PM - 10:30 PM	Solarsh, Eva	PH 212	30	30	Hybrid
8	Current Semester	09	7354	ACCT 101 (4, 3)	Int Theo & Prac Acct 1	T, TH	8:50 PM - 10:30 PM	Zapf, Michael	PH 110	29	55	In-Person
9	Current Semester	08	7353	ACCT 101 (4, 3)	Int Theo & Prac Acct 1	M, W	6:55 PM - 8:45 PM	Ruthizer, Scott	PH 130	47	55	In-Person
10	Current Semester	07	7352	ACCT 101 (4, 3)	Int Theo & Prac Acct 1	T, TH	10:05 AM - 11:55 AM	Solarsh, Eva	SB D133	45	45	In-Person

Our Design


Group3.Building
BuildingId udt.SurrogateKeyInt PK
BuildingName udt.BuildingNameType
Group3.Room
RoomId udt.SurrogateKeyInt PK
RoomNo udt.RoomNo
BuildingId udt.SurrogateKeyInt
BuildingId_1 udt.SurrogateKeyInt FK
Group3.Department
DepartmentID udt.SurrogateKeyInt PK
DepartmentName udt.NameType
Group3.Instructors
InstructorId udt.SurrogateKeyInt PK
FirstName udt.NameType
LastName udt.NameType
FullName udt.NameType
Group3.ModeOfInstruction
ModeId udt.SurrogateKeyInt PK
ModeDescription udt.Description
Group3.Class
ClassId udt.SurrogateKeyInt PK
Semester udt.SemesterType PK
Section Udt.SectionType PK
StartTime Udt.Timetype PK
EndTime Udt.Timetype PK
Day Udt.DayName PK
Enrolled Udt.CountType PK
Limit Udt.CountType PK
RoomId udt.SurrogateKeyInt FK
CourseId udt.SurrogateKeyInt FK
InstructorId udt.SurrogateKeyInt FK
ModeOfInstructionID_1 udt.SurrogateKeyInt FK
Group3.Course
CourseId udt.SurrogateKeyInt PK
CourseCode udt.CodeType
CourseName udt.NameType
Course Description udt.Description
Credits udt.smallInt
HoursPerWeek udt.smallInt

Our Approach : Implementation of Levels

Group3.Instructors [Zhihan][Level 0]

- InstructorId (PK): `udt.SurrogateKeyInt` (int)
- FirstName: `udt.NameType` (nvarchar(40))
- LastName: `udt.NameType` (nvarchar(40))
- FullName: `udt.NameType` (nvarchar(40))

Group3.Building [Mo][Level 0]

- BuildingId (PK): `udt.SurrogateKeyInt` (int)
- BuildingName: `udt.BuildingNameType` (nvarchar(20))

Group3.Room [Mo][Level 1]

- RoomId (PK): `udt.SurrogateKeyInt` (int)
- RoomNo: `udt.RoomNo` (nvarchar(10))
- BuildingId (FK): `udt.SurrogateKeyInt` (int)

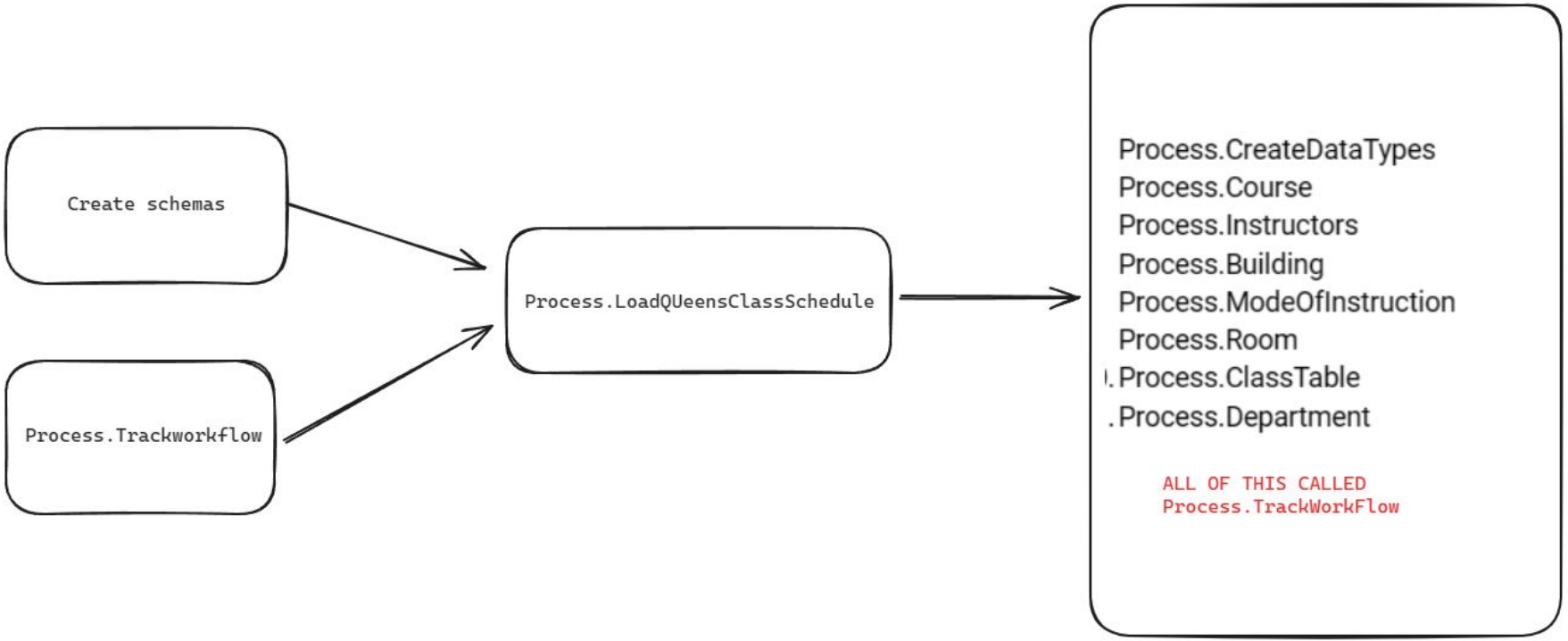
Group3.Class [Jawwad] [Level 2]

- ClassId (PK): `udt.SurrogateKeyInt` (int)
- Semester udt.SemesterType (nvarchar20)
- Section udt.SectionType(int)
- CourseId (FK): `udt.SurrogateKeyInt` (int)
- InstructorID (FK): `udt.SurrogateKeyInt` (int)
- StartTime: `udt.Timetype` (datetime)
- EndTime: `udt.Timetype` (datetime)
- Day: `udt.DayName` (nvarchar(20))
- Enrolled: `udt.countType` (int)
- Limit: `udt.countType` (int)
- RoomId (FK): `udt.SurrogateKeyInt` (int)
- ModeOfInstructionID (FK): `udt.SurrogateKeyInt` (int)

Process.LoadQueensClassSchedule

```sql
CREATE OR ALTER PROCEDURE Process.LoadQueensClassSchedule
AS
BEGIN

    -- Call Process.usp_UserAuthorization
    EXEC Process.usp_UserAuthorization @UserAuthorizationKey = 1;

    -- Call Process.CreateDataTypes
    EXEC Process.CreateDataTypes @UserAuthorizationKey = 1;

    -- Call Process.Course
    EXEC Process.Course @UserAuthorizationKey = 1;

    -- Call Process.Instructors
    EXEC Process.Instructors @UserAuthorizationKey = 6;

    -- Call Process.Building
    EXEC Process.Building @UserAuthorizationKey = 4;

    -- Call Process.Room
    EXEC Process.Room @UserAuthorizationKey = 4;

    -- Call Process.ModeOfInstruction
    EXEC Process.ModeOfInstruction @UserAuthorizationKey = 5;

    -- Call Process.ClassTable
    EXEC Process.ClassTable @UserAuthorizationKey = 3;

    -- Call Process.Department
    EXEC Process.Department @UserAuthorizationKey = 1;

END;
GO
```

Our approach : Stored Procedure Design



Output

	WorkFlowStepKey	WorkFlowStepDescription	WorkFlowStepTableRowCount	StartingDateTime	EndingDateTime	ClassTime	UserAuthorization
1	1	Create User Authorization Table	6	2023-12-07 03:53:20.1753510	2023-12-07 03:53:20.2880314	09:15	1
2	2	Create User Authorization Table	12	2023-12-07 03:55:27.7700550	2023-12-07 03:55:27.7784457	09:15	1
3	3	Create User Defined Data Types	0	2023-12-07 03:55:27.7867540	2023-12-07 03:55:27.9329252	09:15	1
4	4	Create and Populate Group3.Course	4522	2023-12-07 03:55:27.9453439	2023-12-07 03:55:28.0453897	09:15	1
5	5	Create and Populate Group3.Instructors	1596	2023-12-07 03:55:28.0660982	2023-12-07 03:55:28.1368517	09:15	6
6	6	Create and Populate Group3.Building	23	2023-12-07 03:55:28.1493317	2023-12-07 03:55:28.2325050	09:15	4
7	7	Create and Populate Group3.Room	318	2023-12-07 03:55:28.2407491	2023-12-07 03:55:28.3281228	09:15	4
8	8	Create and Populate Group3.ModeOfInstruction	4	2023-12-07 03:55:28.3447871	2023-12-07 03:55:28.4031985	09:15	5
9	9	Create and Populate Group3.Class	4522	2023-12-07 03:55:28.4156812	2023-12-07 03:55:28.7893087	09:15	3
10	11	Create and Populate Group3.Department	85	2023-12-07 04:19:25.4933543	2023-12-07 04:19:25.5772841	09:15	1

Table Design : Create Data Types & Schema

```
CREATE OR ALTER PROCEDURE Process.CreateDataTypes
    @UserAuthorizationKey INT
AS
BEGIN
    DECLARE
        @StartTime DATETIME2 = sysdatetime(),
        @WorkFlowDescription NVARCHAR(100) = 'Create User Defined Data Types',
        @WorkFlowStepTableRowCount INT = 0;

    -- Check if the type exists before creating
    IF TYPE_ID('udt.SurrogateKeyInt') IS NULL
        CREATE TYPE udt.SurrogateKeyInt FROM int;

    IF TYPE_ID('udt.NameType') IS NULL
        CREATE TYPE udt.NameType FROM nvarchar(40);

    IF TYPE_ID('udt.CodeType') IS NULL
        CREATE TYPE udt.CodeType FROM nvarchar(10);

    IF TYPE_ID('udt.Description') IS NULL
        CREATE TYPE udt.Description FROM nvarchar(500);

    IF TYPE_ID('udt.smallNum') IS NULL
        CREATE TYPE udt.smallNum FROM float;

    IF TYPE_ID('udt.Instruction') IS NULL
        CREATE TYPE udt.Instruction FROM nvarchar(255);

    IF TYPE_ID('udt.BuildingNameType') IS NULL
        CREATE TYPE udt.BuildingNameType FROM nvarchar(30);

    IF TYPE_ID('Udt.SemesterType') IS NULL
        CREATE TYPE Udt.SemesterType FROM nvarchar(20);

    IF TYPE_ID('Udt.SectionType') IS NULL
        CREATE TYPE Udt.SectionType FROM nvarchar(20);

    IF TYPE_ID('Udt.Timetype') IS NULL
        CREATE TYPE Udt.Timetype FROM nvarchar(20);

    IF TYPE_ID('Udt.DayName') IS NULL
        CREATE TYPE Udt.DayName FROM nvarchar(20);

    IF TYPE_ID('Udt.CountType') IS NULL
        CREATE TYPE Udt.CountType FROM int;

    IF TYPE_ID('udt.RoomNo') IS NULL
        CREATE TYPE udt.RoomNo FROM nvarchar(50);

    EXEC [Process].[usp_TrackWorkFlow] @StartTime, @WorkFlowDescription, @WorkFlowStepTableRowCount, @UserAuthorizationKey;
END;
```

```
-- Create the stored procedure
CREATE OR ALTER PROCEDURE dbo.CreateSchemas

AS
BEGIN

    -- Create the Group3 schema
    IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.SCHEMATA WHERE SCHEMA_NAME = 'Group3')
    BEGIN
        EXEC('CREATE SCHEMA Group3;');
    END;

    -- Create the Udt schema
    IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.SCHEMATA WHERE SCHEMA_NAME = 'udt')
    BEGIN
        EXEC('CREATE SCHEMA udt;');
    END;

    -- Create the Process schema
    IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.SCHEMATA WHERE SCHEMA_NAME = 'Process')
    BEGIN
        EXEC('CREATE SCHEMA Process;');
    END;

END;
```

Table Design : Group3.Course

```
CREATE OR ALTER PROCEDURE Process.Course
    @UserAuthorizationKey INT
AS
BEGIN
    DECLARE
        @StartTime DATETIME2 = sysdatetime(),
        @WorkFlowDescription NVARCHAR(100) = 'Create and Populate Group3.Course',
        @WorkFlowStepTableRowCount INT;

    -- Create Group3 schema if it does not exist
    IF NOT EXISTS (SELECT * FROM sys.schemas WHERE name = 'Group3')
    BEGIN
        EXEC('CREATE SCHEMA Group3');
    END

    -- Drop table if it exists
    IF OBJECT_ID('Group3.Course', 'U') IS NOT NULL
    BEGIN
        DROP TABLE Group3.Course;
    END

    -- Create the table Group3.Course using the UDTs
    CREATE TABLE Group3.Course (
        CourseId udt.SurrogateKeyInt PRIMARY KEY IDENTITY(1,1),
        CourseCode udt.CodeType,
        CourseName udt.NameType,
        CourseDescription udt.Description,
        Credits udt.smallNum,
        HoursPerWeek udt.smallNum
    );

    -- Populate the table Group3.Course
    INSERT INTO Group3.Course (CourseCode, CourseName, CourseDescription, Credits, HoursPerWeek)
    SELECT DISTINCT
        Code,
        SUBSTRING([Course (hr, crd)], 1, CHARINDEX(' (', [Course (hr, crd)]) - 1),
        Description,
        CAST(SUBSTRING([Course (hr, crd)], CHARINDEX('(', [Course (hr, crd)]) + 1, CHARINDEX(',', [Course (hr, crd)]) - CHARINDEX('(', [Course (hr, crd)]) - 1) AS float),
        CAST(SUBSTRING([Course (hr, crd)], CHARINDEX(',', [Course (hr, crd)]) + 2, CHARINDEX(')', [Course (hr, crd)]) - CHARINDEX(',', [Course (hr, crd)]) - 2) AS float)
    FROM
        Uploadfile.CurrentSemesterCourseOfferings;

    SELECT @WorkFlowStepTableRowCount = COUNT(*) FROM Group3.Course;
    EXEC [Process].[usp_TrackWorkFlow] @StartTime, @WorkFlowDescription, @WorkFlowStepTableRowCount, @UserAuthorizationKey;
END;
```

Workflow Steps

Query to obtain Group3.Course

Table Design : Group3.Department

```sql
INSERT INTO Group3.Department (Department)
SELECT DISTINCT
    SUBSTRING([Course (hr, crd)], 1, CHARINDEX(' ', [Course (hr, crd)]) - 1) AS Department
FROM
    uploadfile.currentsemestercourseofferings
WHERE
    [Course (hr, crd)] IS NOT NULL AND
    CHARINDEX(' ', [Course (hr, crd)]) > 0 AND
    NOT EXISTS (
        SELECT 1 FROM Group3.Department
        WHERE Department = SUBSTRING([Course (hr, crd)], 1, CHARINDEX(' ', [Course (hr, crd)]) - 1)
    );
```

Table Design : Group3.Instructor

```sql
-- Insert data into Group3.Instructors
INSERT INTO Group3.Instructors(FullName, LastName, FirstName)
SELECT DISTINCT
    Instructor,
    SUBSTRING(Instructor, 1, CHARINDEX(',', Instructor + ',') - 1) AS LastName,
    SUBSTRING(Instructor, CHARINDEX(',', Instructor + ',') + 1, LEN(Instructor)) AS FirstName
FROM
    Uploadfile.CurrentSemesterCourseOfferings
WHERE
    Instructor != ',';
```

Table Design : Group3.Building

```
    -- Insertion logic
    WITH UniqueBuildingNames AS (
        SELECT DISTINCT LEFT(csco.Location, 2) AS ShortLocation
        FROM Uploadfile.CurrentSemesterCourseOfferings AS csco
        WHERE csco.Location IS NOT NULL AND LEN(csco.Location) >= 2
    )
    INSERT INTO Group3.Building (BuildingId, BuildingName)
    SELECT
        (COALESCE((SELECT MAX(BuildingId) FROM Group3.Building), 0) + ROW_NUMBER() OVER (ORDER BY (SELECT NULL))) AS NewBuildingId,
        ShortLocation
    FROM
        UniqueBuildingNames
    WHERE
        NOT EXISTS (
            SELECT 1
            FROM Group3.Building
            WHERE BuildingName = ShortLocation
        );

END TRY
BEGIN CATCH
    DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
    RAISERROR (@ErrorMessage, 16, 1);
```

Table Design : Group3.ModeOfInstruction

```sql
-- Populate the ModeOfInstruction table
INSERT INTO Group3.ModeOfInstruction (ModeOfInstruction)
SELECT DISTINCT([Mode of Instruction])
FROM Uploadfile.CurrentSemesterCourseOfferings;
```

Table Design : Group3.Room

```sql
-- Insert data into Group3.Room
INSERT INTO Group3.Room (RoomNo, BuildingId)
SELECT
    RoomNo,
    BuildingId
FROM
    (
        SELECT DISTINCT
            SUBSTRING(cso.Location, CHARINDEX(' ', cso.Location) + 1, LEN(cso.Location)) AS RoomNo,
            b.BuildingId
        FROM
            Uploadfile.CurrentSemesterCourseOfferings cso
        INNER JOIN
            Group3.Building b ON b.BuildingName = LEFT(cso.Location, CHARINDEX(' ', cso.Location) - 1)
        WHERE
            cso.Location IS NOT NULL AND CHARINDEX(' ', cso.Location) > 0
    ) AS UniqueRooms
GROUP BY
    RoomNo, BuildingId;
```

Table Design : Group3.Class

```sql
INSERT INTO Group3.Class(Semester, Section, CourseId, InstructorID,
StartTime, EndTime, [Day], Enrolled, Limit, RoomId, ModeOfInstructionID)
SELECT
    OFFERINGS.Semester,
    OFFERINGS.Sec,
    Course.CourseId,
    Instructors.InstructorID,
    CASE
        WHEN CHARINDEX(' - ', Time) > 0 THEN
            SUBSTRING(Time, 1, CHARINDEX(' - ', Time) - 1)
        ELSE
            Time -- Or NULL, if you want to return NULL when ' - ' is not found
    END AS StartTime,
    CASE
        WHEN CHARINDEX(' - ', Time) > 0 THEN
            SUBSTRING(Time, CHARINDEX(' - ', Time) + 3, LEN(Time))
        ELSE
            NULL -- Assuming EndTime should be NULL if ' - ' is not found
    END AS EndTime,
    OFFERINGS.[Day],
    OFFERINGS.Enrolled,
    OFFERINGS.Limit,
    r.RoomId,
    ModeOfInstruction.ModeID
FROM
    [Uploadfile].[CurrentSemesterCourseOfferings] AS OFFERINGS
LEFT JOIN
    Group3.Course AS Course ON OFFERINGS.[Course (hr, crd)] = Course.CourseName
LEFT JOIN
    Group3.Instructors AS Instructors ON OFFERINGS.Instructor = Instructors.FullName
LEFT JOIN
    (Group3.Room r INNER JOIN Group3.Building as b ON r.BuildingId=b.buildingId)
ON
    SUBSTRING(Location, CHARINDEX(' ', Location) + 1, LEN(Location)) = r.RoomNo
AND
    LEFT(Location, CHARINDEX(' ', Location + ' ') - 1) = b.BuildingName
INNER JOIN
    Group3.ModeOfInstruction AS ModeOfInstruction ON OFFERINGS.[Mode of Instruction] = ModeOfInstruction.ModeOfInstruction;
```

V. Closing Remarks & Introduction to Implementation (JDBC & Queries)