Assignment 2: Database Development using SQL Server

Total points: 20

This assignment helps you practice building and querying a relational database using **SQL Server**. Please review the relevant lecture slides and SQL materials before starting.

Assignment Overview & Mark Distribution

Create tables	4 points
Define integrity constraints	4 points
Populate the database with sample data	3 points
SQL queries for information retrieval	8 points
Report writing	1 point

Tools & Setup (if not already completed)

You may either:

- Install **SQL Server** locally **(Windows only)**, or
- **Use Azure SQL Database (free tier)**

To connect and run your queries, you can use:

- **VS Code** (Recommended; cross-platform, supports multiple languages)
  - If using **VS Code**, please install the "MSSQL" extension in VS Code.
- **Azure Data Studio** (Cross-platform, retiring in Feb 2026)
- **SSMS (SQL Server Management Studio)** (Windows only)

Please refer to SQL installation/setup guide in eClass, Module 05: SQL DDL → "SQLServer_InstallationGuide" and watch the corresponding lecture videos for setup help.

Notes Before You Start

- Stick to the provided table and column names (use UPPERCASE with no spaces).
- Data types and relationships must reflect the structure in the problem.
- Save all your queries and include them in your report. **You must submit SQL files, in addition to your report**.
- Only run CREATE TABLE statements **once** unless you drop the table first.
- Insert **your own employee record and assignment** as part of the data entry.

- You can work individually or with one partner only.

Problem Definition:

A consulting company needs to store data to track the charges incurred for various projects. Charges are calculated based on the number of hours each employee works on a project.

- An employee can be assigned to **multiple projects**, and a project can have **multiple employees** working on it.
- Each project is **managed by exactly one employee**, but an employee may manage **zero or more projects**.

The relational database structure and contents are shown below:

Relational Schema

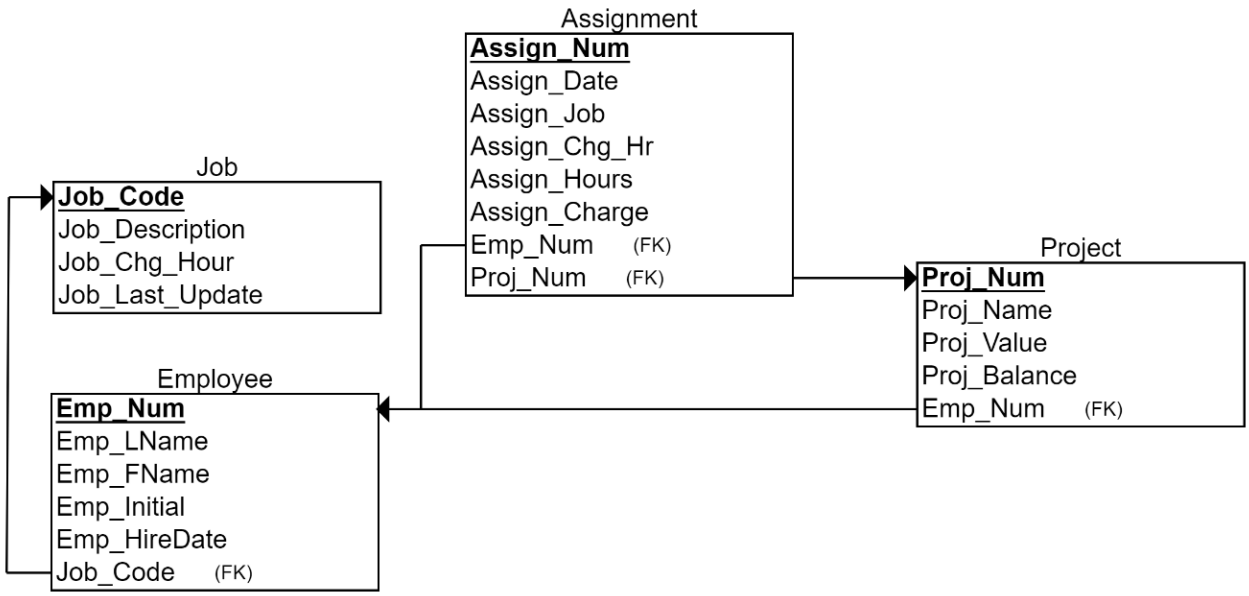

Table: JOB

JOB_CODE	JOB_DESCRIPTION	JOB_CHG_HOUR	JOB_LAST_UPDATE
500	Programmer	35.75	20-Nov-2017
501	Systems Analyst	96.75	20-Nov-2017
502	Database Designer	125.00	21-Mar-2018
503	Electrical Engineer	84.50	20-Nov-2017
504	Mechanical Engineer	67.90	20-Nov-2017
505	Civil Engineer	55.78	20-Nov-2017
506	Clerical Support	26.87	20-Nov-2017
507	DSS Analyst	45.95	20-Nov-2017
508	Applications Designer	48.10	21-Mar-2018
509	Bio Technician	34.55	20-Nov-2017
510	General Support	18.36	20-Nov-2017

Table: EMPLOYEE

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIREDATE	JOB_CODE
101	News	John	G	08-Nov-2000	502
102	Senior	David	H	12-Jul-1989	501
103	Arbough	June	E	01-Dec-1996	500
104	Ramoras	Anne	K	15-Nov-1987	501
105	Johnston	Alice	K	01-Feb-1993	502
106	Smithfield	William		22-Jun-2004	500
107	Alonzo	Maria	D	10-Oct-1993	500
108	Washington	Ralph	B	22-Aug-1991	501
109	Smith	Larry	W	18-Jul-1997	501
110	Olenko	Gerald	A	11-Dec-1995	505
111	Wabash	Geoff	B	04-Apr-1991	506
112	Smithson	Darlene	M	23-Oct-1994	507
113	Joenbrood	Delbert	K	15-Nov-1996	508
114	Jones	Annelise		20-Aug-1993	508
115	Bawangi	Travis	B	25-Jan-1992	501
116	Pratt	Gerald	L	05-Mar-1997	510
117	Williamson	Angie	H	19-Jun-1996	509
118	Frommer	James	J	04-Jan-2005	510

Table: PROJECT

PROJ_NUM	PROJ_NAME	PROJ_VALUE	PROJ_BALANCE	EMP_NUM
15	Evergreen	1453500.00	1002350.00	103
18	Amber Wave	3500500.00	2110346.00	108
22	Rolling Tide	805000.00	500345.20	102
25	Star flight	2650500.00	2309880.00	107

Table: ASSIGNMENT

ASSIGN_NUM	ASSIGN_DATE	PROJ_NUM	EMP_NUM	ASSIGN_JOB	ASSIGN_CHR_HR	ASSIGN_HOURS	ASSIGN_CHARGE
1001	22-Mar-2018	18	103	500	35.75	3.5	125.13
1002	22-Mar-2018	22	117	509	34.55	4.2	145.11
1003	23-Mar-2018	18	117	509	34.55	2.0	69.10
1004	23-Mar-2018	18	103	500	35.75	5.9	210.93
1005	23-Mar-2018	25	108	501	96.75	2.2	212.85
1006	23-Mar-2018	22	104	501	96.75	4.2	406.35
1007	23-Mar-2018	25	113	508	48.10	3.8	182.78
1008	24-Mar-2018	18	103	500	35.75	0.9	32.18
1009	24-Mar-2018	15	115	501	96.75	5.6	541.80
1010	24-Mar-2018	15	117	509	34.55	2.4	82.92
1011	24-Mar-2018	25	105	502	125.00	4.3	537.50
1012	24-Mar-2018	18	108	501	96.75	3.4	328.95
1013	25-Mar-2018	25	115	501	96.75	2.0	193.50
1014	25-Mar-2018	22	104	501	96.75	2.8	270.90
1015	25-Mar-2018	15	103	500	35.75	6.1	218.08
1016	25-Mar-2018	22	105	502	125.00	4.7	587.50
1017	25-Mar-2018	18	117	509	34.55	3.8	131.29
1018	26-Mar-2018	25	117	509	34.55	2.2	76.01
1019	26-Mar-2018	25	104	501	96.75	4.9	474.08
1020	26-Mar-2018	15	101	502	125.00	3.1	387.50
1021	26-Mar-2018	22	108	501	96.75	2.7	261.23
1022	26-Mar-2018	22	115	501	96.75	4.9	474.08
1023	26-Mar-2018	22	105	502	125.00	3.5	437.50
1024	26-Mar-2018	15	103	500	35.75	3.3	117.98
1025	27-Mar-2018	18	117	509	34.55	4.2	145.11

Task 1: Table Creation (4 Points)

Write and run SQL statements to create all the tables described in the problem using **SQL Server** (you may use SSMS, Azure Data Studio, VS Code, or another SQL client).

- Use the **exact table and column names** shown in the schema (**UPPERCASE, no spaces**).
- Choose **appropriate data types** for each column based on the sample data. For example:
  - For JOB_CHG_HOUR, use NUMERIC(6,2) to allow values like 35.75, and up to 9999.99.
    - Why not NUMERIC(4,2)? Because it only allows values up to 99.99.
  - JOB_CODE → INT, NUMERIC(3,0), or CHAR(3) (e.g., '002' if leading zeros are needed)
- Define **primary keys** and **foreign keys** in your CREATE TABLE statements.
- Ensure the **data types of foreign keys match** those of the referenced primary keys.
- If a column contains **blank (missing) values** in the sample data, define it to **allow NULL** values (e.g., EMP_INITIAL CHAR(1) NULL).

Note on the ASSIGNMENT Table:

The provided ASSIGNMENT table contains redundancy and violates Third Normal Form (3NF). You may either:

- Create the table **as-is**, or
- **Normalize** it using the simplified ASSIGN table below:

```
CREATE TABLE ASSIGN (
ASSIGN_NUM INT NOT NULL,
ASSIGN_DATE DATE,
PROJ_NUM INT,
EMP_NUM INT,
ASSIGN_HOURS NUMERIC (5,1),
PRIMARY KEY (ASSIGN_NUM),
FOREIGN KEY (PROJ_NUM) REFERENCES PROJECT,
FOREIGN KEY (EMP_NUM) REFERENCES EMPLOYEE);
```

If you use the normalized design, you may later create a VIEW to replicate the ASSIGNMENT structure using joins.

Implementation Notes:

- Run your CREATE TABLE statements in the correct order so that FK constraints work. For example, in the Student Registration example, the "enrollment" table could only be created after the "course" and "student" tables because "enrollment" table had foreign keys referring to the columns of these two tables.
- Notice that the create table statement can only run once. If you need to change a table, **drop it first** before recreating it.

- **Save all your queries** and include them in your final report. You must submit the SQL file and your report.

Task 2: Integrity Constraints and Relationships (4 points)

Specify the integrity constraints and table relationships you implemented as part of your CREATE TABLE statements. Briefly describe the following.

- **Entity Integrity**: identifying the **primary key** for each table.
- **Referential Integrity**: Foreign keys that link related tables (e.g., EMP_NUM in ASSIGNMENT referencing EMPLOYEE). Mention any actions you defined such as: ON DELETE RESTRICT, ON UPDATE CASCADE, etc.
- **Domain Integrity**: Proper data types (e.g., DATE for ASSIGN_DATE) and value constraints that ensure valid input.

Optional: Generate Database Diagram

If you're using **SQL Server Management Studio (SSMS)**, you may create a database diagram to show the table structures and relationships visually. You may refer to **module 5** slides (SQL DDL), pages **48-51** to see how you can generate Database Diagrams and check integrity controls in SSMS.

If you're using **VS Code** or **Azure Data Studio**, this feature is not available. You may either:

- Use an external tool to draw your relational diagram (**optional**), or
- Simply rely on your written explanation of **foreign key constraints** to show how tables are related.

Recommended free diagramming tools (optional):

- dbdiagram.io
- QuickDBD
- DrawSQL

Task 3: Enter sample data for the tables you created (3 points)

Insert sample data into your tables using INSERT statements (do **not** use table editors or GUI-based data entry tools). Make sure all entries comply with your integrity constraints.

Guidelines:

- Insert data **in the correct order**, based on foreign key dependencies.
- Use **four-digit years** for all dates (e.g., '2017-11-20' or '20-Nov-2017', not '20-Nov-17').
- Do **not re-run** the same INSERT statement twice. Inserting duplicate primary keys will cause errors.
- You may insert **multiple rows** in a single query to save time: INSERT INTO ... VALUES (...), (...);

Note: If any values are missing in the sample tables provided (i.e., cells are blank), treat those as NULL when inserting data.

Personalized Record:

To make your submission unique:

- Add **your name** as a new employee in the EMPLOYEE table (with a new EMP_NUM)
- Add an assignment for yourself to one project in the ASSIGNMENT (or ASSIGN) table
- **Report the insert statements** you used to enter your own data.

This helps ensure your work is unique and identifiable.
Include the SQL statements you used to insert your personalized records in your final report.

Design Note (Optional):

The ASSIGNMENT table contains some data that could be derived from other tables (e.g., job information or hourly charges). While this introduces redundancy, such designs are sometimes used to preserve historical records even when job roles or pay rates change.

If you're interested, you may experiment with a more normalized design by using a simpler ASSIGN table and creating a VIEW to reconstruct the full structure.

Here is a reminder of the basic structure for creating a view:

```
CREATE VIEW view_name AS
SELECT ….
FROM ….
JOIN … ON  …
WHERE ….;
```

You may check if the data in the virtual ASSIGNMENT table (view) is the same as expected.

Task 4: Design queries for information search (8 points)

Write and run SQL queries to retrieve useful information from your database. Each query is designed to test your understanding of SQL fundamentals, including filtering, sorting, and, where applicable, joins, grouping, and aggregation.

For each query:

- Write the SQL statement using correct syntax.
- Run it on your database and ensure the results are correct.
- Take a screenshot showing both the query and the output.

- Include each SQL query and its screenshot in your report.
- You must submit the **SQL file** along with your **report (MS word, or PDF)**.

Complete the following 5 queries:

1. List all employees who were hired **before January 1, 1995 (1 point).**
2. List the **job code**, **job description**, and **employee last name and first name**. Sort the result by **job description**, then by **employee last name (1.5 point).**
3. List the **first and last names** of all employees assigned to the project named **"Evergreen" (1.5 point).**
4. For each project, list the **project name**, the **number of employees** worked for the project, the **total number of hours worked**, and the **total charges** assigned to each project. Sort the result by project name **(2 points).**
5. List all employees who have been assigned to **more than one project**. Show their EMP_NUM, EMP_LNAME, EMP_FNAME, and the number of distinct projects they were assigned to **(2 points).**

After running each query:

- Save both the **SQL code** and the **result**.
- You must submit the **SQL file** along with your **report (MS word, or PDF)**.
- Make sure your results are **correct**, **readable**, and clearly show both the query and the output.

Task 5: Compile Your Assignment Report (1 Point)

Prepare a single, well-organized report that includes the results and explanations for Tasks 1–4. Your report should:

- Include all required SQL code, screenshots of results, and explanations where needed.
- Be organized by task (e.g., clearly labeled sections for Task 1, Task 2, etc.).
- Begin with a cover page including your full name and student ID on the first page.

You do not need to write a separate report for this task, your completed Assignment 2 is the report.

Final Notes:

1. This assignment must be completed **individually** or in **pairs (maximum 2 students)**. You may discuss general concepts with others, but your implementation must be your own. **Copying another student's work or report is not permitted.**
2. You must submit both SQL files and your report.

3. Submit your report as a **MS Word document**. Use the following **file naming format**: **ASS2_YourFullName_SectionLetter (**For example: ASS2_MonaNasery_M).
4. Submit your assignment by the **due date** specified in the course outline or class schedule.
5. Upload your completed assignment to **eClass** under: Deliverables → Assignment 2.