CSU 44D01
Information Management for Engineering
TechStackTracker Application

Ujjayant Kadian
22330954

20 November 2023

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Contents

1 Application Description **3**

2 Entity Relation Diagram for the Application **4**

3 Mapping to Relational Tables **4**

4 Functional Dependencies **5**

5 Explanation and SQL Code for Creating the Database Tables (including any constraints) **6**

6 Explanation and SQL Code for Altering tables **8**

7 Explanation and SQL Code for Creating Views **8**

8 Explanation and SQL Code for Populating the Tables **10**

9 Explanation and SQL Code for retrieving information from the database (including Joins and use of functions) **13**

10 Explanation and SQL Code for Triggers **14**

11 Explanation and SQL Code for Security (Roles & Permissions) **16**

1 Application Description

TechStackTracker is a powerful and user-friendly MySQL-based application designed to help students (especially computer engineering students and developers) effectively manage their technology stack and simplify and simplify the complexities of juggling various libraries, frameworks and dependencies across multiple projects.

The key features of this application are:

- It keeps track of all the technologies, libraries and dependencies a user has used or plans to use in the coding projects.
- Have all the related details to a particular technology (such as documentation links, version details, and compatibility notes).
- Seamlessly link the technological stack with the projects. Keep track of all the technologies a project is using.
- Have all the necessary details belonging to a particular project and have a clear classification of projects to help users pick technologies for new similar projects.
- Keep track of technology versions, updates and releases.

How does the Application work?

- Add all the technologies, libraries, and frameworks you have used or intend to use in your projects.
- Add all the projects that you have done and are doing. Associate each project with the relevant technologies from your stack.
- Keep track of all the versions, dependencies and updates so that you remain up to date with the technologies.
- Filter the technology stack to find what you need for your current project or assignment. Fetch all the details related to the technology.
- Use this application to find the projects and their related technologies to make it easier for future projects.

2 Entity Relation Diagram for the Application

The entity relation diagram for the application was developed as follows:

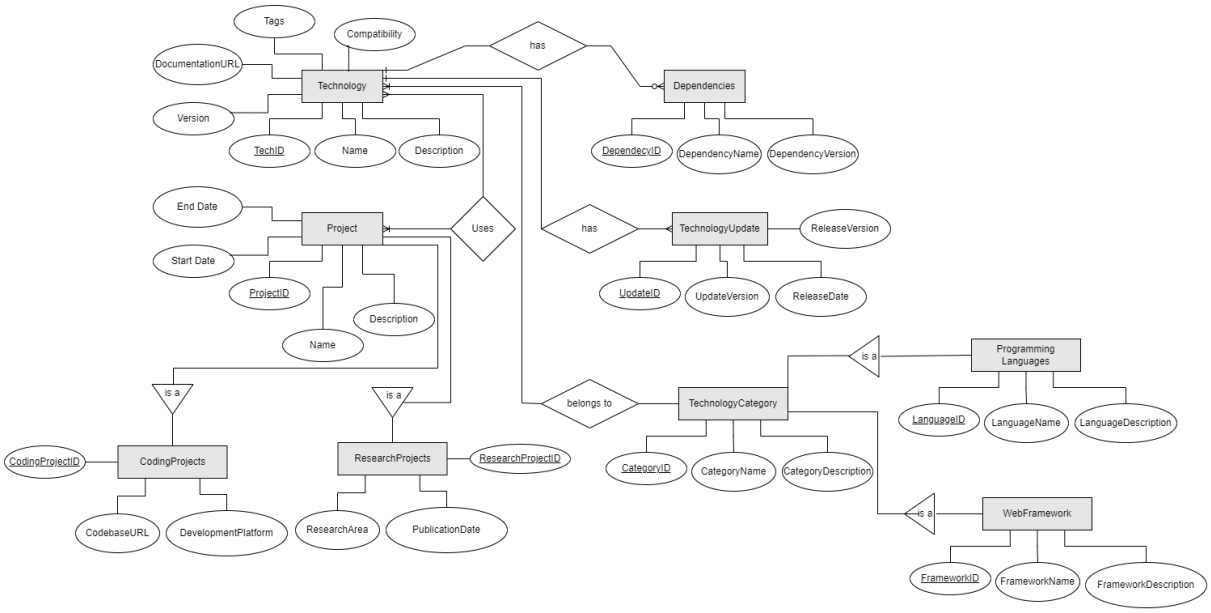

Figure 1: EER Diagram

The database broadly has five entities - Technology, Dependencies, TechnologyUpdates, Project and TechnologyCategory. Technology, Dependecies, TechnologyUpdates and TechnologyCategory records all the related information to a technology. Project records all the information concerned with a project. There are two subclasses of TechnologyCategory, ProgrammingLanguages and WebFramework. These subclasses help to better classify technologies. There are two subclasses of Projects, CodingProjects and ResearchProjects. This helps to store some additional information about a project based on how coding or research-oriented the project is. The relationship between each entity can be observed from the above ER Diagram.

3 Mapping to Relational Tables

The EER Diagram can be mapped to the following tables with the following data types:

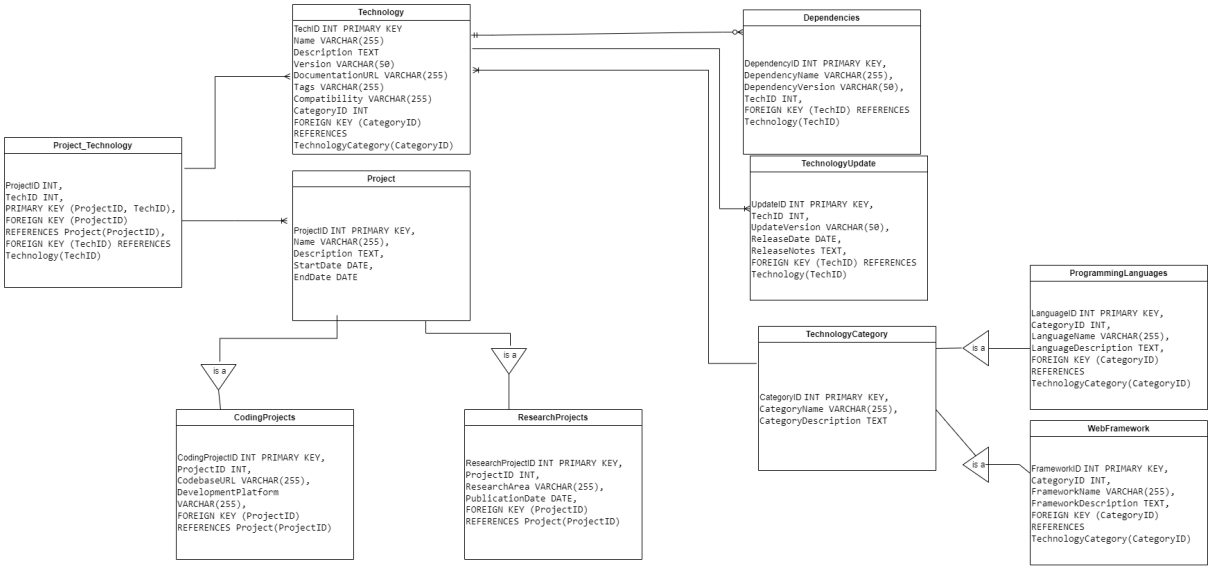

Figure 2: Mapping Relational Schema to Relational Tables

To establish the relationship between Technology and TechnologyCategory, the CategoryID is used as a foreign key in Technology.

To establish the relationship between Technology and Project, a new table is created with TechID and ProjectID as the attributes (TechID referencing Technology and ProjectID referencing Project). This helps to effectively link multiple technologies with multiple projects.

To link the subclasses, a foreign key is referencing to their parents primary key.

4 Functional Dependencies

```
Technology:
TechID-> (Name, Description, Version, DocumentationURL, Tags, Compatibility, CategoryID)

Dependencies:
DependencyID-> (DependencyName, DependencyVersion, TechID)

TechnologyUpdate:
UpdateID-> (TechID, UpdateVersion, ReleaseDate, ReleaseNotes)

TechnologyCategory:
CategoryID-> (CategoryName, CategoryDescription)

ProgrammingLanguages:
LanguageID-> (CategoryID, Language Name, LanguageDescription)

WebFramework:
FrameworkID-> (CategoryID, FrameworkName, FrameworkDescription)

Project:
ProjectID-> (Name, Description, StartDate, EndDate)

CodingProjects:
CodingProjectID-> (ProjectID, CodebaseURL, DevelopmentPlatform)
```

```
ResearchProjects:
ResearchProjectID-> (ResearchArea, PublicationDate)

Project_Technology:
(ProjectID, TechID) are primary keys
```

From the above functional dependencies, it can be seen that:

- All attributes are atomic. - 1NF
- There is only one candidate key in all tables. All attributes are dependent on the whole primary key. - 2NF
- There are no transitive dependencies. - 3NF

Thus it can be said the relational schema is appropriately normalised.

5 Explanation and SQL Code for Creating the Database Tables (including any constraints)

The SQL Code for creating the database tables are as follows:

```
-- Creating the Database for TechStackTracker
CREATE DATABASE TechStackTrackerDB;
-- Using the Database
USE TechStackTrackerDB;

-- Creating Tables according to the EER Diagram:
-- Technology Table: Showing all the Technologies
CREATE TABLE TechStackTrackerDB.Technology (
    TechID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL UNIQUE,
    Description TEXT,
    Version VARCHAR(50) NOT NULL,
    DocumentationURL VARCHAR(255) NOT NULL UNIQUE,
    Tags VARCHAR(255) NOT NULL,
    Compatibility VARCHAR(255) NOT NULL,
    CategoryID INT NOT NULL,
    FOREIGN KEY (CategoryID) REFERENCES TechnologyCategory(CategoryID)
);

-- Project Table: Showing all the Projects
CREATE TABLE TechStackTrackerDB.Project (
    ProjectID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL UNIQUE,
    Description TEXT,
    StartDate DATE NOT NULL,
    EndDate DATE NOT NULL,
    CHECK (StartDate <= EndDate)
);

-- CodingProject Table: Specializing Projects into Coding Projects:
CREATE TABLE TechStackTrackerDB.CodingProject (
```

```
    CodingProjectID INT PRIMARY KEY,
    ProjectID INT NOT NULL,
    CodebaseURL VARCHAR(255) NOT NULL UNIQUE,
    DevelopmentPlatform VARCHAR(255) NOT NULL,
    FOREIGN KEY (ProjectID) REFERENCES Project(ProjectID)
);

-- ResearchProject Table: Specializing Projects into Research Projects:
CREATE TABLE TechStackTrackerDB.ResearchProject (
    ResearchProjectID INT PRIMARY KEY,
    ProjectID INT NOT NULL,
    ResearchArea VARCHAR(255) NOT NULL UNIQUE,
    PublicationDate DATE NOT NULL,
    FOREIGN KEY (ProjectID) REFERENCES Project(ProjectID)
);

-- Dependency Table: Showing all the Dependencies related to a Technology
CREATE TABLE TechStackTrackerDB.Dependency (
    DependencyID INT PRIMARY KEY,
    DependencyName VARCHAR(255) NOT NULL UNIQUE,
    DependencyVersion VARCHAR(50) NOT NULL,
    TechID INT NOT NULL,
    FOREIGN KEY (TechID) REFERENCES Technology(TechID)
);

-- TechnologyUpdate Table: Showing all the Technology Updates
CREATE TABLE TechStackTrackerDB.TechnologyUpdate (
    UpdateID INT PRIMARY KEY,
    TechID INT NOT NULL,
    UpdateVersion VARCHAR(50) NOT NULL,
    ReleaseDate DATE NOT NULL,
    ReleaseNotes TEXT,
    FOREIGN KEY (TechID) REFERENCES Technology(TechID)
);

-- TechnologyCategoryTable: Showing all the types of the categories
CREATE TABLE TechStackTrackerDB.TechnologyCategory (
    CategoryID INT PRIMARY KEY,
    CategoryName VARCHAR(255) NOT NULL UNIQUE,
    CategoryDescription TEXT
);

-- ProgrammingLanguage Table: Technologies classified as languages
CREATE TABLE TechStackTrackerDB.ProgrammingLanguage (
    LanguageID INT PRIMARY KEY,
    CategoryID INT NOT NULL,
    LanguageName VARCHAR(255) NOT NULL UNIQUE,
    LanguageDescription TEXT,
    FOREIGN KEY (CategoryID) REFERENCES TechnologyCategory(CategoryID)
);
```

```
-- WebFramework Table: Technologies classified as web frameworks
CREATE TABLE TechStackTrackerDB.WebFramework (
    FrameworkID INT PRIMARY KEY,
    CategoryID INT NOT NULL,
    FrameworkName VARCHAR(255) NOT NULL UNIQUE,
    FrameworkDescription TEXT,
    FOREIGN KEY (CategoryID) REFERENCES TechnologyCategory(CategoryID)
);

-- Project_Technology Table: Relationship between Technology and Project
CREATE TABLE TechStackTrackerDB.Project_Technology (
    ProjectID INT,
    TechID INT,
    PRIMARY KEY (ProjectID, TechID),
    FOREIGN KEY (ProjectID) REFERENCES Project(ProjectID),
    FOREIGN KEY (TechID) REFERENCES Technology(TechID)
);
```

Constraints:

- Primary Key and Foreign Key: The keys are defined as shown in the code to depict the relationships.
- NOT NULL and UNIQUE: The names in all tables are kept as not null and unique for the sake of the application. It ensures no duplicate entries are inserted.
- CHECK: Project's StartDate is compared with EndDate to ensure that the start date is lesser than the end date.

6 Explanation and SQL Code for Altering tables

There was no reason to alter tables except to rename ProgrammingLanguage table to ProgrammingLanguages to be consistent with the EER Diagram. Code:

```
-- Renaming ProgrammingLanguage Table
ALTER TABLE TechStackTrackerDB.ProgrammingLanguage
RENAME TO TechStackTrackerDB.ProgrammingLanguages;
```

7 Explanation and SQL Code for Creating Views

The following Views are Created:

```
- VIEWS:
-- Technology and Corresponding Category:
CREATE VIEW TechView AS
SELECT
    t.TechID,
    t.Name,
    t.Description,
    t.Version,
    t.DocumentationURL,
    t.Tags,
    t.Compatibility,
```

```
    tc.CategoryName
FROM TechStackTrackerDB.Technology t
LEFT JOIN TechStackTrackerDB.TechnologyCategory tc ON t.CategoryID = tc.CategoryID;

-- Project to Coding Projects and Research Projects
CREATE VIEW ProjectView AS
SELECT
    p.ProjectID,
    p.Name,
    p.Description,
    p.StartDate,
    p.EndDate,
    cp.CodebaseURL,
    cp.DevelopmentPlatform,
    rp.ResearchArea,
    rp.PublicationDate
FROM Project p
LEFT JOIN TechStackTrackerDB.CodingProject cp ON p.ProjectID = cp.ProjectID
LEFT JOIN TechStackTrackerDB.ResearchProject rp ON p.ProjectID = rp.ProjectID;

-- Dependency with corresponding Technology:
CREATE VIEW DependencyView AS
SELECT
    d.DependencyID,
    d.DependencyName,
    d.DependencyVersion,
    t.Name AS TechName
FROM TechStackTrackerDB.Dependency d
JOIN TechStackTrackerDB.Technology t ON d.TechID = t.TechID;

-- TechnologyUpdate with Technology:
CREATE VIEW TechnologyUpdateView AS
SELECT
    tu.UpdateID,
    tu.UpdateVersion,
    tu.ReleaseDate,
    tu.ReleaseNotes,
    t.Name AS TechName
FROM TechStackTrackerDB.TechnologyUpdate tu
JOIN TechStackTrackerDB.Technology t ON tu.TechID = t.TechID;

-- Project and Technology:
CREATE VIEW Project_TechnologyView AS
SELECT
    pt.ProjectID,
    pt.TechID,
    p.Name AS ProjectName,
    t.Name AS TechName
FROM TechStackTrackerDB.Project_Technology pt
JOIN TechStackTrackerDB.Project p ON pt.ProjectID = p.ProjectID
JOIN TechStackTrackerDB.Technology t ON pt.TechID = t.TechID;
```

- TechView: Displays the technology and its corresponding category by left joining TechnologyCategory with Technology.
- ProjectView: Displays all the projects with the corresponding special details (if the project is a CodingProject, a ResearchProject or both). Left join both CodingProject and ResearchProject with Project.
- DependencyView: Displays all the dependencies corresponding to Technology. Dependency Inner Join Technology since not all Technologies have their entry in Dependency.
- TechnologyUpdate: Displays the updates corresponding to a technology. TechnologyUpdate Inner Join Technology as there are many updates corresponding to a Technology.
- Project_TechnologyView: Displaying all the projects and the corresponding technologies they are using. Inner Joining Project and Technology with Project_Technology as there are many projects which have many technologies corresponding to them.

8 Explanation and SQL Code for Populating the Tables

Code:

```
-- INSERTING VALUES INTO THESE TABLES:
-- TechnologyCategory:
INSERT INTO TechStackTrackerDB.TechnologyCategory (CategoryID, CategoryName,
CategoryDescription)
VALUES
(1, 'Database', 'Technologies related to databases'),
(2, 'Programming Language', 'Languages used for coding'),
(3, 'Web Framework', 'Frameworks for web development'),
(4, 'Artificial Intelligence', 'Technologies related to AI and Machine
Learning'),
(5, 'Computer Security', 'Technologies for Computer Security'),
(6, 'JavaScript', 'Technologies primarily using JavaScript and are not web
frameworks'),
(7, 'Documentation Tool', 'Allows you to write documents easily');

-- ProgrammingLanguages:
INSERT INTO TechStackTrackerDB.ProgrammingLanguages (LanguageID, CategoryID,
LanguageName, LanguageDescription)
VALUES
(1, 2, 'Python', 'High-level programming language'),
(2, 6, 'JavaScript', 'High-level programming language for the web'),
(3, 2, 'Java', 'General-purpose programming language'),
(4, 2, 'C++', 'General-purpose programming language'),
(5, 1, 'SQL', 'Structured Query Language for databases');

-- WebFrameworks:
INSERT INTO TechStackTrackerDB.WebFramework (FrameworkID, CategoryID,
FrameworkName, FrameworkDescription)
VALUES
(1, 3, 'Django', 'High-level Python web framework'),
(2, 6, 'React', 'JavaScript library for building user interfaces'),
(3, 3, 'Express.js', 'Web application framework for Node.js'),
```

```
(4, 3, 'Flask', 'Lightweight web application framework for Python'),
(5, 3, 'Spring Boot', 'Java-based framework for building web applications');

-- Technology:
INSERT INTO TechStackTrackerDB.Technology (TechID, Name, Description, Version,
DocumentationURL, Tags, Compatibility, CategoryID)
VALUES
(1, 'MySQL', 'Relational Database Management System', '8.0',
'https://dev.mysql.com/doc/', 'Database', 'Compatible with most web development
stacks', 1),
(2, 'Python', 'High-level programming language', '3.9',
'https://www.python.org/', 'Programming Language', 'Widely used for web
development and scripting', 2),
(3, 'Django', 'Web framework for Python', '3.2',
'https://www.djangoproject.com/', 'Web Framework', 'Follows the model-view-
controller architectural pattern', 3),
(4, 'React', 'JavaScript library for building user interfaces', '17.0',
'https://reactjs.org/', 'JavaScript, Frontend', 'Component-based UI
development', 6),
(5, 'Node.js', 'JavaScript runtime for server-side development', '14.17',
'https://nodejs.org/', 'JavaScript, Backend', 'Enables building scalable
network applications', 6),
(6, 'Overleaf', 'Documetation Software', '14.1', 'https://overleaf.org/',
'LaTex', 'Allows to write documents using Code', 7),
(7, 'C++', 'Low-Level programming language', '3.1', 'https://C++.org/', 'GNU',
'Allows low level programming and object oriented programming', 2);

-- Dependency
INSERT INTO TechStackTrackerDB.Dependency (DependencyID, DependencyName,
DependencyVersion, TechID)
VALUES
(1, 'Flask', '2.0', 2),
(2, 'React Router', '5.2', 4),
(3, 'Express.js', '4.17', 5),
(4, 'NumPy', '1.21', 2),
(5, 'D3.js', '7.0', 4);

-- TechnologyUpdate:
INSERT INTO TechStackTrackerDB.TechnologyUpdate (UpdateID, TechID,
UpdateVersion, ReleaseDate, ReleaseNotes)
VALUES
(1, 1, '8.0.27', '2023-07-15', 'Security updates and bug fixes'),
(2, 4, '17.0.2', '2023-06-30', 'Performance improvements'),
(3, 2, '3.9.7', '2023-08-05', 'New features and enhancements'),
(4, 5, '14.17.6', '2023-07-20', 'Stability improvements'),
(5, 3, '3.2.8', '2023-06-15', 'Bug fixes and documentation updates'),
(6, 6, '14.1', '2023-06-15', 'Current updated version'),
(7, 7, '3.1', '2023-06-15', 'Current updated version');

-- Project:
INSERT INTO TechStackTrackerDB.Project (ProjectID, Name, Description,
```

```
StartDate, EndDate)
VALUES
(1, 'E-commerce Website', 'Building an online store', '2023-01-01', '2023-12-
31'),
(2, 'Data Analysis Tool', 'Analyzing large datasets', '2023-02-01', '2023-06-
30'),
(3, 'Portfolio Website', 'Showcasing personal projects', '2023-03-01', '2023-03-
31'),
(4, 'Research Paper on AI', 'Investigating the impact of AI on society', '2023-
04-01', '2023-09-30'),
(5, 'Task Management App', 'Developing an application for task tracking', '2023-
05-01', '2023-08-31'),
(6, 'Finding Probable Causes for Brain Tumour', 'Listing probable reasons for
brain tumour', '2023-05-01', '2023-08-31'),
(7, 'Dijkstra Algorithm', 'Implementing Dijkstra Algorithm', '2023-01-01',
'2023-01-02');

-- CodingProject:
INSERT INTO TechStackTrackerDB.CodingProject (CodingProjectID, ProjectID,
CodebaseURL, DevelopmentPlatform)
VALUES
(1, 1, 'https://github.com/example/e-commerce', 'Node.js and React'),
(2, 3, 'https://github.com/example/portfolio', 'Django'),
(3, 5, 'https://github.com/example/task-management', 'Node.js and React'),
(4, 2, 'https://github.com/example/data-analysis', 'Python and NumPy'),
(5, 4, 'https://github.com/example/ai-research', 'Python and Flask'),
(6, 7, 'https://github.com/example/Dijkstra Algorithm', 'C++');

-- ResearchProject:
INSERT INTO TechStackTrackerDB.ResearchProject (ResearchProjectID, ProjectID,
ResearchArea, PublicationDate)
VALUES
(1, 4, 'Artificial Intelligence', '2023-10-01'),
(2, 2, 'Data Science', '2023-07-05'),
(3, 5, 'Software Engineering', '2023-09-15'),
(4, 3, 'Web Development', '2023-04-20'),
(5, 1, 'E-commerce and Business', '2023-11-01'),
(6, 6, 'Cancer Research', '2023-08-31');

-- Project_Technology:
INSERT INTO TechStackTrackerDB.Project_Technology (ProjectID, TechID)
VALUES
(1, 1),
(1, 4),
(1, 5),
(2, 2),
(3, 1),
(3, 2),
(3, 3),
(3, 4),
(3, 5),
```

```
(4, 2),
(4, 6),
(5, 2),
(5, 3),
(6, 6),
(7, 7);
```

The values are inserted in such a way which covers all the possible different scenarios. A project using many technologies, there are some projects that are coding projects, research projects or both. The technologies are appropriately divided into some categories and specialised into either programming languages or web frameworks or none. The values try to display all the relevant information related to projects and technologies.

9 Explanation and SQL Code for retrieving information from the database (including Joins and use of functions)

Some examples of information retrieval are shown below:

```
-- Retrieving Information:
-- Technologies with their latest update information
SELECT
    t.TechID,
t.Name,
    t.Description,
    t.Version,
    t.DocumentationURL,
    t.Tags,
    t.Compatibility,
    tu.UpdateVersion,
    tu.ReleaseDate,
    tu.ReleaseNotes
FROM Technology t
LEFT JOIN TechnologyUpdate tu ON t.TechID = tu.TechID
WHERE tu.UpdateID IS NULL OR tu.UpdateID = (
    SELECT MAX(UpdateID) FROM TechnologyUpdate tu WHERE t.TechID = tu.TechID
);

-- Count of projects corresponding to each technology
SELECT
    t.Name AS TechnologyName,
    COUNT(p.ProjectID) AS ProjectCount
FROM Technology t
LEFT JOIN Project_Technology pt ON t.TechID = pt.TechID
LEFT JOIN Project p ON pt.ProjectID = p.ProjectID
GROUP BY t.TechID;

-- Average Project Duration
SELECT
    t.Name AS TechnologyName,
    AVG(DATEDIFF(p.EndDate, p.StartDate)) AS AverageProjectDuration
FROM Technology t
LEFT JOIN Project_Technology pt ON t.TechID = pt.TechID
```

```
LEFT JOIN Project p ON pt.ProjectID = p.ProjectID
GROUP BY t.TechID;
```

- First Retrieval: By left joining TechnologyUpdate with Technology, all the updates related to a technology are fetched. In the where clause, the subquery fetches the most recent UpdateID (max) for a particular TechID. The query then displays the most recent update of all the technologies in the Technology table.
- Second Retrieval: It counts the number of projects using a particular technology. By left joining Technology with Project_Technology and Project it fetches all the projects belonging to a technology. Grouping by each technology (TechID) and then calculating the number of rows, it displays the number of projects using a particular technology.
- Third Retrieval: It finds the average project duration when using a particular technology. It fetches all the projects belonging to a technology as in the second retrieval and groups them by each TechID. Using the DATEDIFF function difference in dates is calculated and using the AVG the average of this difference is calculated for each group.

10 Explanation and SQL Code for Triggers

Code:

```
-- Triggers:
-- Update the version of the technology in Technology Table when you insert in
TechnologyUpdate (when it is updated)
delimiter //
CREATE TRIGGER update_tech_version
BEFORE INSERT ON TechnologyUpdate
FOR EACH ROW
BEGIN
    UPDATE Technology
    SET Version = NEW.UpdateVersion WHERE TechID = NEW.TechID;
END;
//
delimiter ;

-- Delete Entries from Project_Technology, CodingProject and Research Project
when a entry in Project is deleted
delimiter //
CREATE TRIGGER delete_project_technologies_cp_rp
BEFORE DELETE ON Project
FOR EACH ROW
BEGIN
    DELETE FROM Project_Technology WHERE ProjectID = OLD.ProjectID;
    DELETE FROM CodingProject WHERE ProjectID IS NOT NULL AND ProjectID =
    OLD.ProjectID;
    DELETE FROM ResearchProject WHERE ProjectID IS NOT NULL AND ProjectID =
    OLD.ProjectID;
END;
//
delimiter ;
```

```
-- Trigger to prevent change in Project Start Dates
delimiter //
CREATE TRIGGER prevent_project_start_date_change
BEFORE UPDATE ON Project
FOR EACH ROW
BEGIN
    IF NEW.StartDate != OLD.StartDate THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot change the start date of a project';
    END IF;
END;
//
delimiter ;

-- Prevent technology deletion if it has associated projects:
delimiter //
CREATE TRIGGER prevent_technology_deletions
BEFORE DELETE ON Technology
FOR EACH ROW
BEGIN
    DECLARE proj_count INT;

    -- Check if the project has associated technologies
    SELECT COUNT(*) INTO proj_count
    FROM Project_Technology
    WHERE TechID = OLD.TechID;

    -- If there are associated technologies, cancel the deletion
    IF proj_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot delete project with associated technologies';
    END IF;
END
//
delimiter ;
-- DELETE FROM technology WHERE TechID = 1; -- Error Code: 1644. Cannot delete
project with associated technologies
```

- First Trigger: Before every insert in TechnologyUpdate, this trigger will run. It will update the Version in the Technology table for every row whose TechID is the same as the inserted entry's TechID (accessed using New).
- Second Trigger: If you delete an entry from the Project table, the tuples corresponding to this ProjectID (accessed using OLD) in Project_Technology, CodingProject and ResearchProject will also be deleted.
- Third Trigger: This trigger will prevent the change of the start date in the Project Table. It will compare the OLD start date with the NEW start date and if both are different (due to updation) it will signal an error.
- Fourth Trigger: This trigger will prevent an entry from being deleted from technology if it has projects associated with it. A variable proj_count is used to see if there are projects associated with a TechID or not. First, it counts the number of projects associated with

the particular TechID (stored in OLD before deleting) and stores it in proj_count. Then, using the if statement it is checked if the proj_count is not 0, it will throw an error (as shown in the comment of the code).

11 Explanation and SQL Code for Security (Roles & Permissions)

Before Assigning Permissions and Roles, there must be multiple users defined. A user is defined using 'Create User' as shown below (username and password are filled). To assign different permissions (SELECT, INSERT, DELETE, UPDATE - CRUD) to the user 'Grant' is used.

CODE:

```
-- Security:
-- Granting less permissions to a USER
CREATE USER 'tech_user_demo'@'localhost' IDENTIFIED BY 'password';
GRANT SELECT, INSERT ON Technology TO 'tech_user_demo'@'localhost';
GRANT SELECT, INSERT ON Project TO 'tech_user_demo'@'localhost';
GRANT SELECT, INSERT ON Dependency TO 'tech_user_demo'@'localhost';
GRANT SELECT, INSERT ON TechnologyUpdate TO 'tech_user_demo'@'localhost';
GRANT SELECT ON TechnologyCategory TO 'tech_user_demo'@'localhost';
GRANT SELECT, INSERT ON CodingProject TO 'tech_user_demo'@'localhost';
GRANT SELECT, INSERT ON ResearchProject TO 'tech_user_demo'@'localhost';
GRANT SELECT, INSERT ON ProgrammingLanguages TO 'tech_user_demo'@'localhost';
GRANT SELECT, INSERT ON WebFramework TO 'tech_user_demo'@'localhost';
GRANT SELECT, INSERT ON Project_Technology TO 'tech_user_demo'@'localhost';
-- Now if this user signs in with this username and password, he will have no updatation and

-- 03:28:13 DELETE FROM codingproject WHERE CodingID < 2 Error Code: 1142.
DELETE command denied to user 'tech_user_demo'@'localhost' for table
'codingproject'0.000 sec
```

If the user signs in with this particular username and password, he will not have DELETION and UPDATION permissions and an error with code 1142 will be thrown.