University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 4

Date: 5th April 2024

Group Number: 10

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Kanish Khanna	20186961	e2p2p@ugrad.cs.ubc.ca	Kanishkhanna2020@gmail.com
Yiquan Liu	33205998	c7m8c@ugrad.cs.ubc.ca	Springliu2003@gmail.com
Aaditya Suri	41935511	f5o3r@ugrad.cs.ubc.ca	Aadityasuri01@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

1. **A single SQL script that can be used to create all the tables and data in the database. If you are using multiple scripts while developing, ensure you concatenate them and hand in only a SINGLE SQL script.**

2. **A PDF file containing:**

a. A short description of the final project, and what it accomplished.

Our project is for software engineering project management and tracking. The final GUI has a home page consisting of all projects in the database. The home page contains options to add, delete, filter, and reset projects, as well as the more general searches like viewing any attributes from any tables in the database and the average number of team members. By clicking each project box, users can see all the attributes regarding the project and perform searches based on the project. The project accomplishes enhanced team collaboration, task and bug tracking, deadline and status tracking, project information, repositories and file updates.

b. A description of how your final schema differed from the schema you turned in.

The final schema is unchanged from what we planned.

c. A copy of the schema and screenshots that show what data is present in each relation after the SQL script from item #2 is run.

<u>Final Schema</u>

```
USE cpsc304;
DROP TABLE Designers CASCADE CONSTRAINTS;
DROP TABLE Managers CASCADE CONSTRAINTS;
DROP TABLE QAs CASCADE CONSTRAINTS;
DROP TABLE Engineers CASCADE CONSTRAINTS;
DROP TABLE BlockedBy CASCADE CONSTRAINTS;
DROP TABLE TaskHaveBugs CASCADE CONSTRAINTS;
DROP TABLE Bugs CASCADE CONSTRAINTS;
DROP TABLE Files CASCADE CONSTRAINTS;
DROP TABLE Repositories CASCADE CONSTRAINTS;
DROP TABLE Releases CASCADE CONSTRAINTS;
DROP TABLE WorkOnBy CASCADE CONSTRAINTS;
DROP TABLE TeamMembers CASCADE CONSTRAINTS;
DROP TABLE AssignTo CASCADE CONSTRAINTS;
DROP TABLE Teams CASCADE CONSTRAINTS;
DROP TABLE Projects CASCADE CONSTRAINTS;
DROP TABLE Employees CASCADE CONSTRAINTS;
DROP TABLE PostNormTasksR1 CASCADE CONSTRAINTS;
DROP TABLE PostNormTasksR2 CASCADE CONSTRAINTS;
DROP TABLE PostNormTasksR3 CASCADE CONSTRAINTS;
DROP DATABASE cpsc304;

CREATE DATABASE cpsc304;
USE cpsc304;

CREATE TABLE Projects (
  Name VARCHAR(255),
  Description VARCHAR(255) UNIQUE,
  Deadline DATE,
  PRIMARY KEY (Name)
);

CREATE TABLE Teams (
  TeamID INT,
  TeamSize INT,
  TeamFunction VARCHAR(255),
  PRIMARY KEY (TeamID)
);

CREATE TABLE TeamMembers (
  EmployeeID INT,
  TeamID INT,
  Name VARCHAR(255),
  Seniority INT,
  PRIMARY KEY (EmployeeID),
  FOREIGN KEY (TeamID) REFERENCES Teams(TeamID) ON DELETE CASCADE ON UPDATE
CASCADE
);

CREATE TABLE Releases (
  Version VARCHAR(255),
  Name VARCHAR(255),
  ProjectName VARCHAR(255) NOT NULL,
  Changes VARCHAR(255),
  ReleaseDate DATE,
  PRIMARY KEY (Version, Name),
  FOREIGN KEY (ProjectName) REFERENCES Projects(Name) ON DELETE CASCADE ON UPDATE
CASCADE
);

CREATE TABLE Repositories (
  URL VARCHAR(255) PRIMARY KEY,
  ProjectName VARCHAR(255) NOT NULL,
  Name VARCHAR(255),
  FOREIGN KEY (ProjectName) REFERENCES Projects(Name) ON DELETE CASCADE ON UPDATE
CASCADE
);

CREATE TABLE Files (
  Path VARCHAR(255),
  URL VARCHAR(255) NOT NULL,
  FileName VARCHAR(255),
  PRIMARY KEY (Path),
  FOREIGN KEY (URL) REFERENCES Repositories(URL) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Bugs (
  ID INT,
  Description VARCHAR(255),
  Severity INT,
  PRIMARY KEY (ID)
);

CREATE TABLE AssignTo (
  Name VARCHAR(255),
  TeamID INT,
  PRIMARY KEY (Name, TeamID),
  FOREIGN KEY (Name) REFERENCES Projects(Name) ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (TeamID) REFERENCES Teams(TeamID) ON DELETE CASCADE ON UPDATE
CASCADE
```

```
);

CREATE TABLE WorkOnBy (
  EmployeeID INT,
  Version VARCHAR(255),
  ReleaseName VARCHAR(255),
  PRIMARY KEY (EmployeeID, Version, ReleaseName),
  FOREIGN KEY (EmployeeID) REFERENCES TeamMembers(EmployeeID) ON DELETE CASCADE ON
UPDATE CASCADE,
  FOREIGN KEY (Version, ReleaseName) REFERENCES Releases(Version, Name) ON DELETE
CASCADE ON UPDATE CASCADE
);

CREATE TABLE Engineers (
  EmployeeID INT NOT NULL,
  TechStack VARCHAR(255),
  MainPushAccess CHAR(1) CHECK (MainPushAccess IN ('Y', 'N')),
  PRIMARY KEY (EmployeeID),
  FOREIGN KEY (EmployeeID) REFERENCES TeamMembers(EmployeeID) ON DELETE CASCADE ON
UPDATE CASCADE
);

CREATE TABLE QAs (
  EmployeeID INT NOT NULL,
  AutomationLevel INT,
  PRIMARY KEY (EmployeeID),
  FOREIGN KEY (EmployeeID) REFERENCES TeamMembers(EmployeeID) ON DELETE CASCADE ON
UPDATE CASCADE
);

CREATE TABLE Managers (
  EmployeeID INT NOT NULL,
  Tools VARCHAR(255),
  PRIMARY KEY (EmployeeID),
  FOREIGN KEY (EmployeeID) REFERENCES TeamMembers(EmployeeID) ON DELETE CASCADE ON
UPDATE CASCADE
);
```

```
CREATE TABLE Designers (
  EmployeeID INT NOT NULL,
  Specialization VARCHAR(255),
  PRIMARY KEY (EmployeeID),
  FOREIGN KEY (EmployeeID) REFERENCES TeamMembers(EmployeeID) ON DELETE CASCADE ON
UPDATE CASCADE
);

CREATE TABLE PostNormTasksR3 (
  Description VARCHAR(255) UNIQUE,
  TaskID INT,
  ProjectName VARCHAR(255),
  Progress INT,
  PRIMARY KEY (TaskID, ProjectName),
  FOREIGN KEY (ProjectName) REFERENCES Projects(Name) ON DELETE CASCADE ON UPDATE
CASCADE
);

CREATE TABLE PostNormTasksR2 (
  Description VARCHAR(255),
  Priority INT UNIQUE,
  PRIMARY KEY (Description),
  FOREIGN KEY (Description) REFERENCES PostNormTasksR3(Description) ON DELETE CASCADE
ON UPDATE CASCADE
);

CREATE TABLE PostNormTasksR1 (
  Priority INT,
  Deadline DATE,
  PRIMARY KEY (Priority),
  FOREIGN KEY (Priority) REFERENCES PostNormTasksR2(Priority) ON DELETE CASCADE ON
UPDATE CASCADE
);

CREATE TABLE TaskHaveBugs (
  BugID INT,
  TaskID INT,
  ProjectName VARCHAR(255),
  PRIMARY KEY (BugID, TaskID, ProjectName),
  FOREIGN KEY (BugID) REFERENCES Bugs(ID) ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (TaskID) REFERENCES PostNormTasksR3(TaskID) ON DELETE CASCADE ON
UPDATE CASCADE,
  FOREIGN KEY (ProjectName) REFERENCES Projects(Name) ON DELETE CASCADE ON UPDATE
CASCADE
);

CREATE TABLE BlockedBy (
  BlockerTaskID INT,
  BlockedTaskID INT,
  PRIMARY KEY (BlockerTaskID, BlockedTaskID),
  FOREIGN KEY (BlockerTaskID) REFERENCES PostNormTasksR3(TaskID) ON DELETE CASCADE ON
UPDATE CASCADE,
  FOREIGN KEY (BlockedTaskID) REFERENCES PostNormTasksR3(TaskID) ON DELETE CASCADE ON
UPDATE CASCADE
);

CREATE TABLE ReportedBy (
  BugID INT,
  EmployeeID INT NOT NULL,
  PRIMARY KEY (BugID, EmployeeID),
  FOREIGN KEY (BugID) REFERENCES Bugs(ID) ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (EmployeeID) REFERENCES TeamMembers(EmployeeID) ON DELETE CASCADE ON
UPDATE CASCADE
);

INSERT INTO Projects (Name, Description, Deadline) VALUES
('Search', 'Search Engine', '2024-03-15'),
('Ads', 'Ad integration', '2024-04-30'),
('Networking', 'Network routing', '2024-05-20'),
('Classifier', 'CNN image classification', '2024-06-10'),
('Frontend', 'UI for website', '2024-07-25');

INSERT INTO Teams (TeamID, TeamSize, TeamFunction) VALUES
```

```
(1, 2, 'Development'),
(2, 1, 'Testing'),
(3, 4, 'Design'),
(4, 2, 'Support'),
(5, 1, 'Marketing');

INSERT INTO AssignTo (Name, TeamID) VALUES
('Search', 1),
('Ads', 2),
('Networking', 3),
('Classifier', 4),
('Frontend', 5),
('Ads', 1),
('Networking', 1),
('Classifier', 1),
('Frontend', 1);

INSERT INTO TeamMembers (EmployeeID, TeamID, Name, Seniority) VALUES
(1, 1, 'John', 3),
(2, 1, 'Alice', 2),
(3, 3, 'Bob', 4),
(4, 4, 'Emma', 1),
(5, 4, 'Tom', 5),
(6, 3, 'Jim', 3),
(7, 3, 'Jane', 2),
(8, 3, 'Sam', 4),
(9, 2, 'ABC', 1),
(10, 5, 'XYZ', 2);

-- Ensure the Version and ReleaseName columns are correctly defined in the Releases table
before inserting into WorkOnBy
INSERT INTO Releases (Version, Name, ProjectName, Changes, ReleaseDate) VALUES
('1', 'Release1', 'Search', 'Bug fixes and enhancements', '2024-03-10'),
('2', 'Release2', 'Ads', 'New feature additions', '2024-04-20'),
('3', 'Release3', 'Networking', 'Performance improvements', '2024-05-15'),
('4', 'Release4', 'Classifier', 'Major overhaul and redesign', '2024-06-30'),
('5', 'Release5', 'Frontend', 'Marketing campaign updates', '2024-07-20');
```

```
-- Assuming WorkOnBy table's structure is aligned with these columns
INSERT INTO WorkOnBy (EmployeeID, Version, ReleaseName) VALUES
(1, '1', 'Release1'),
(2, '2', 'Release2'),
(3, '3', 'Release3'),
(4, '3', 'Release3'),
(5, '3', 'Release3');

INSERT INTO Repositories (URL, ProjectName, Name) VALUES
('http://example.com/repo1', 'Search', 'Repo1'),
('http://example.com/repo2', 'Ads', 'Repo2'),
('http://example.com/repo3', 'Networking', 'Repo3'),
('http://example.com/repo4', 'Classifier', 'Repo4'),
('http://example.com/repo5', 'Frontend', 'Repo5');

INSERT INTO Files (Path, URL, FileName) VALUES
('/path1', 'http://example.com/repo1', 'file1.txt'),
('/path2', 'http://example.com/repo2', 'file2.txt'),
('/path3', 'http://example.com/repo3', 'file3.txt'),
('/path4', 'http://example.com/repo4', 'file4.txt'),
('/path5', 'http://example.com/repo5', 'file5.txt');

INSERT INTO Bugs (ID, Description, Severity) VALUES
(1, 'Bug description 1', 3),
(2, 'Bug description 2', 2),
(3, 'Bug description 3', 1),
(4, 'Bug description 4', 2),
(5, 'Bug description 5', 3),
(6, 'Bug Description 6', 1),
(7, 'Bug description 7', 2),
(8, 'Bug description 8', 1),
(9, 'Bug description 9', 3),
(10, 'Bug description 10', 2),
(11, 'Bug description 11', 1);

INSERT INTO PostNormTasksR3 (TaskID, ProjectName, Description, Progress) VALUES
```

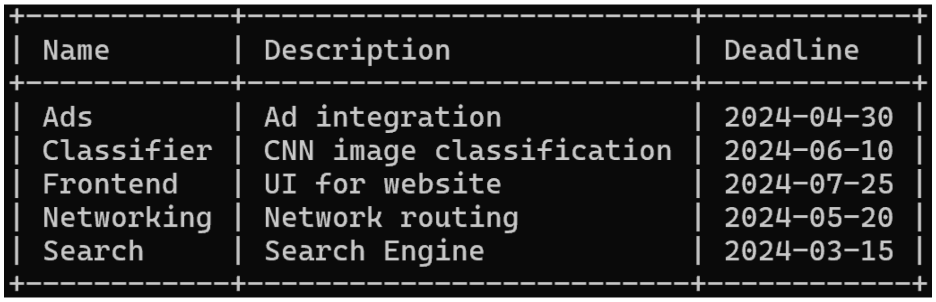
```sql
(1, 'Search', 'Task description 1', 50),
(2, 'Ads', 'Task description 2', 75),
(3, 'Networking', 'Task description 3', 30),
(4, 'Classifier', 'Task description 4', 90),
(5, 'Frontend', 'Task description 5', 60),
(6, 'Classifier', 'Task description 6', 82);

INSERT INTO PostNormTasksR2 (Description, Priority) VALUES
('Task description 1', 1),
('Task description 2', 2),
('Task description 3', 3),
('Task description 4', 4),
('Task description 5', 5),
('Task description 6', 6);

-- Note: Ensure PostNormTasks tables are created and interlinked correctly before these inserts
INSERT INTO PostNormTasksR1 (Priority, Deadline) VALUES
(1, '2024-03-20'),
(2, '2024-04-25'),
(3, '2024-05-30'),
(4, '2024-06-15'),
(5, '2024-07-30'),
(6, '2024-05-15');

INSERT INTO Engineers (EmployeeID, TechStack, MainPushAccess) VALUES
(1, 'Java, Spring', 'Y'),
(2, 'Python, Django', 'N'),
(3, 'JavaScript, React', 'Y'),
(4, 'C#, .NET', 'N'),
(5, 'Ruby, Rails', 'Y');

INSERT INTO QAs (EmployeeID, AutomationLevel) VALUES
(1, 3),
(2, 2),
(3, 1),
(4, 2),
(5, 3);
```

```
INSERT INTO Managers (EmployeeID, Tools) VALUES
(1, 'Jira, Confluence'),
(2, 'Trello, Slack'),
(3, 'Asana, Basecamp'),
(4, 'GitHub, GitLab'),
(5, 'Bitbucket, Jenkins');

INSERT INTO Designers (EmployeeID, Specialization) VALUES
(1, 'UI/UX design'),
(2, 'Graphic design'),
(3, 'Web design'),
(4, 'Product design'),
(5, 'Interior design');

-- Note: Ensure Tasks and Bugs tables are populated before TaskHaveBugs due to FK constraints
INSERT INTO TaskHaveBugs (BugID, TaskID, ProjectName) VALUES
(1, 1, 'Search'),
(2, 2, 'Ads'),
(3, 3, 'Networking'),
(4, 4, 'Classifier'),
(5, 5, 'Frontend'),
(6, 4, 'Classifier'),
(7, 4, 'Classifier'),
(8, 4, 'Classifier'),
(9, 4, 'Classifier'),
(10, 4, 'Classifier'),
(11, 6, 'Classifier');

INSERT INTO BlockedBy (BlockerTaskID, BlockedTaskID) VALUES
(1, 2),
(2, 3),
(3, 4),
(4, 5),
(1, 5);

INSERT INTO ReportedBy (BugID, EmployeeID)
```

VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(1, 5);

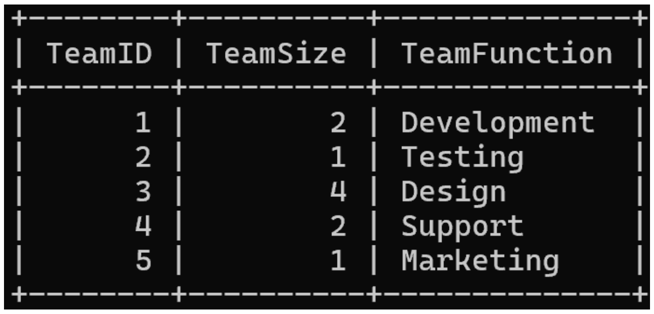
List of Table Definitions

Projects (Name: varchar, Description: varchar, Deadline: date)



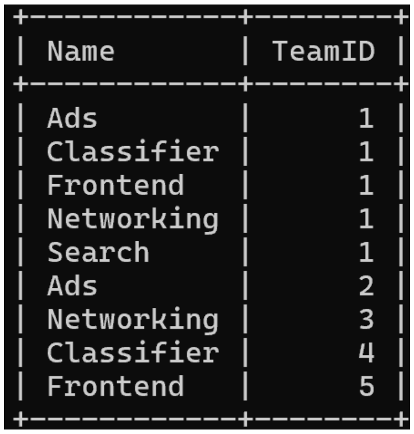
```
+------------+-------------------------+------------+
| Name       | Description             | Deadline   |
+------------+-------------------------+------------+
| Ads        | Ad integration          | 2024-04-30 |
| Classifier | CNN image classification | 2024-06-10 |
| Frontend   | UI for website          | 2024-07-25 |
| Networking | Network routing         | 2024-05-20 |
| Search     | Search Engine           | 2024-03-15 |
+------------+-------------------------+------------+
```

Teams (TeamID: integer, Size: integer, Function: varchar)



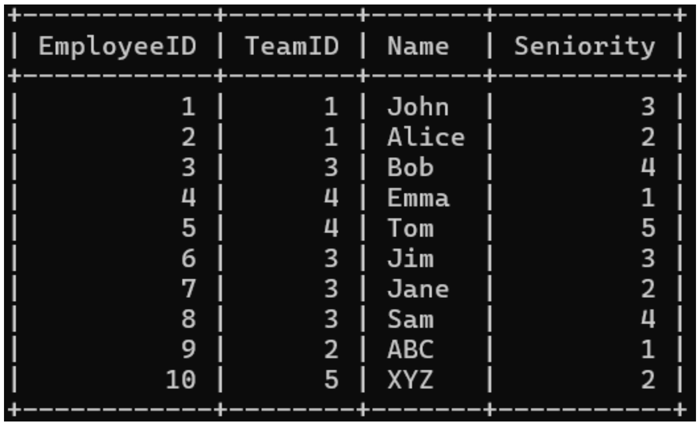
```
+--------+----------+--------------+
| TeamID | TeamSize | TeamFunction |
+--------+----------+--------------+
|      1 |        2 | Development  |
|      2 |        1 | Testing      |
|      3 |        4 | Design       |
|      4 |        2 | Support      |
|      5 |        1 | Marketing    |
+--------+----------+--------------+
```

AssignTo (**Name: varchar**, **TeamID: integer**)



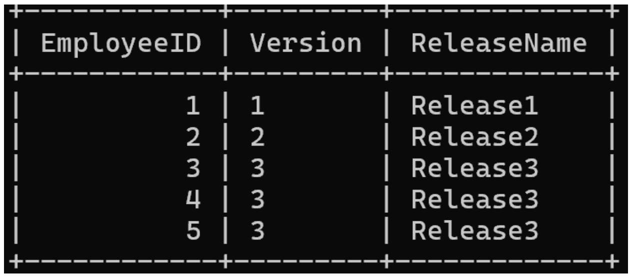
```
+------------+--------+
| Name       | TeamID |
+------------+--------+
| Ads        |      1 |
| Classifier |      1 |
| Frontend   |      1 |
| Networking |      1 |
| Search     |      1 |
| Ads        |      2 |
| Networking |      3 |
| Classifier |      4 |
| Frontend   |      5 |
+------------+--------+
```

TeamMembers (EmployeeID: integer, **TeamID: integer**, Name: varchar, Seniority: integer)



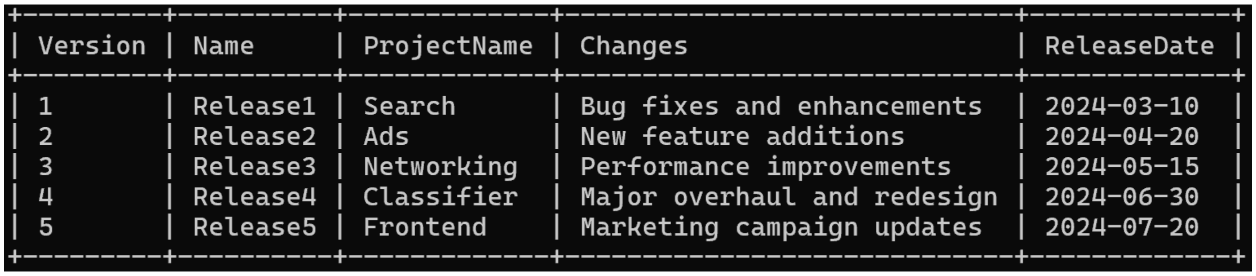
```
+------------+--------+-------+-----------+
| EmployeeID | TeamID | Name  | Seniority |
+------------+--------+-------+-----------+
|          1 |      1 | John  |         3 |
|          2 |      1 | Alice |         2 |
|          3 |      3 | Bob   |         4 |
|          4 |      4 | Emma  |         1 |
|          5 |      4 | Tom   |         5 |
|          6 |      3 | Jim   |         3 |
|          7 |      3 | Jane  |         2 |
|          8 |      3 | Sam   |         4 |
|          9 |      2 | ABC   |         1 |
|         10 |      5 | XYZ   |         2 |
+------------+--------+-------+-----------+
```

WorkOnBy (**EmployeeID: integer, Version: integer, ReleaseName: varchar**)



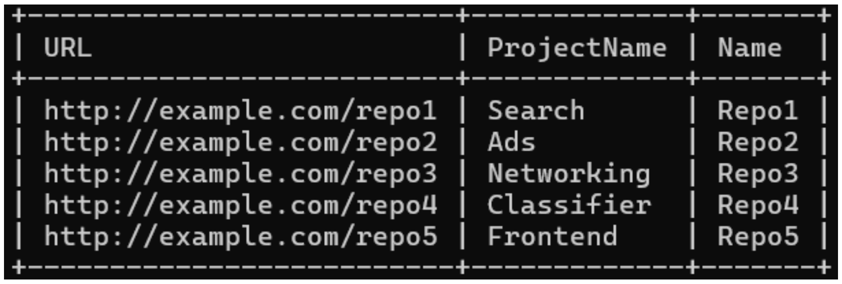
```
+------------+---------+-------------+
| EmployeeID | Version | ReleaseName |
+------------+---------+-------------+
|          1 | 1       | Release1    |
|          2 | 2       | Release2    |
|          3 | 3       | Release3    |
|          4 | 3       | Release3    |
|          5 | 3       | Release3    |
+------------+---------+-------------+
```

Releases (Version:varchar, Name:varchar, **ProjectName: varchar**, Changes: varchar, Date: date)



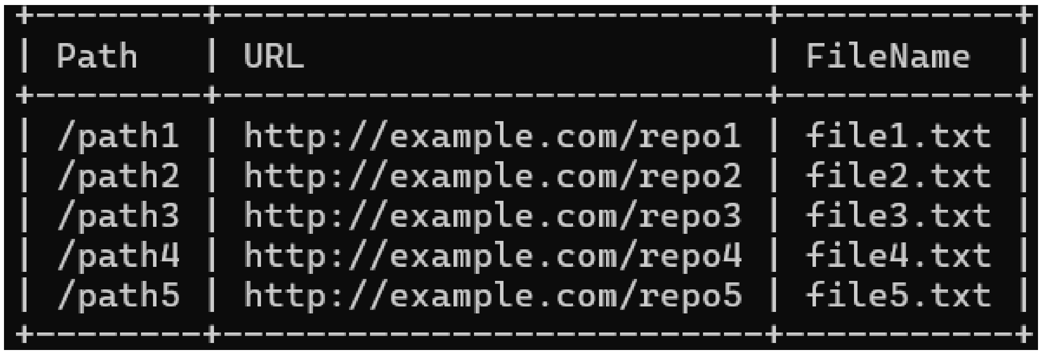
```
+---------+----------+-------------+-----------------------------+-------------+
| Version | Name     | ProjectName | Changes                     | ReleaseDate |
+---------+----------+-------------+-----------------------------+-------------+
| 1       | Release1 | Search      | Bug fixes and enhancements  | 2024-03-10  |
| 2       | Release2 | Ads         | New feature additions       | 2024-04-20  |
| 3       | Release3 | Networking  | Performance improvements    | 2024-05-15  |
| 4       | Release4 | Classifier  | Major overhaul and redesign | 2024-06-30  |
| 5       | Release5 | Frontend    | Marketing campaign updates  | 2024-07-20  |
+---------+----------+-------------+-----------------------------+-------------+
```

Repositories (URL: varchar, **ProjectName: varchar**, Name:varchar)



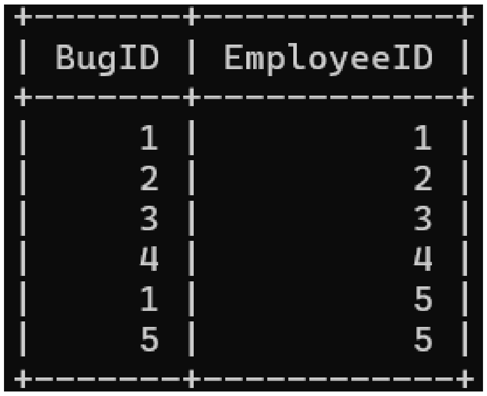
```
+--------------------------+-------------+-------+
| URL                      | ProjectName | Name  |
+--------------------------+-------------+-------+
| http://example.com/repo1 | Search      | Repo1 |
| http://example.com/repo2 | Ads         | Repo2 |
| http://example.com/repo3 | Networking  | Repo3 |
| http://example.com/repo4 | Classifier  | Repo4 |
| http://example.com/repo5 | Frontend    | Repo5 |
+--------------------------+-------------+-------+
```

Files(Path:varchar, **URL:varchar**, FileName:varchar)



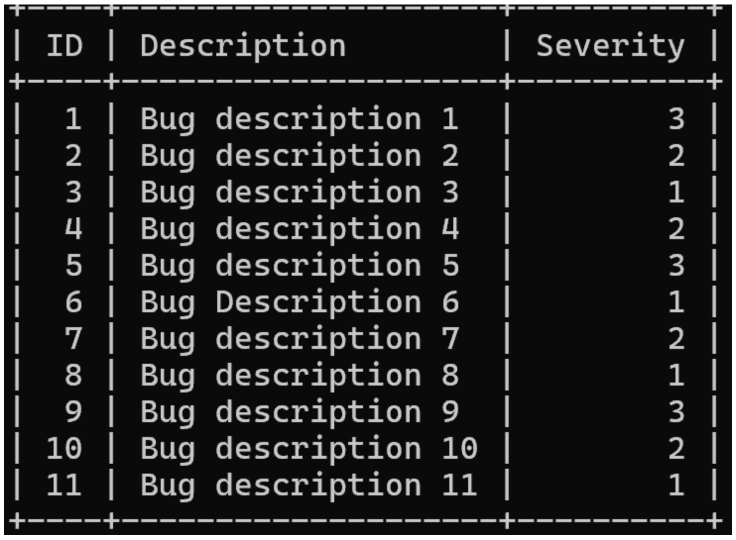
```
+--------+--------------------------+-----------+
| Path   | URL                      | FileName  |
+--------+--------------------------+-----------+
| /path1 | http://example.com/repo1 | file1.txt |
| /path2 | http://example.com/repo2 | file2.txt |
| /path3 | http://example.com/repo3 | file3.txt |
| /path4 | http://example.com/repo4 | file4.txt |
| /path5 | http://example.com/repo5 | file5.txt |
+--------+--------------------------+-----------+
```

ReportedBy(**BugID:integer**, **EmployeeID:integer**)



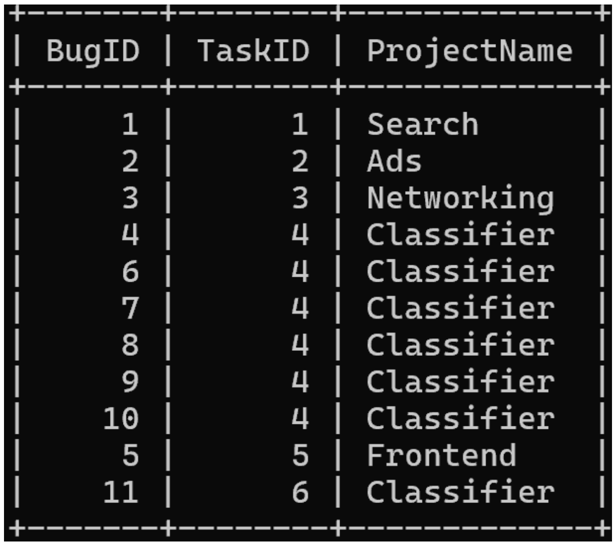
```
+-------+------------+
| BugID | EmployeeID |
+-------+------------+
|     1 |          1 |
|     2 |          2 |
|     3 |          3 |
|     4 |          4 |
|     1 |          5 |
|     5 |          5 |
+-------+------------+
```

Bugs(ID: integer, Description: varchar, Severity: integer)



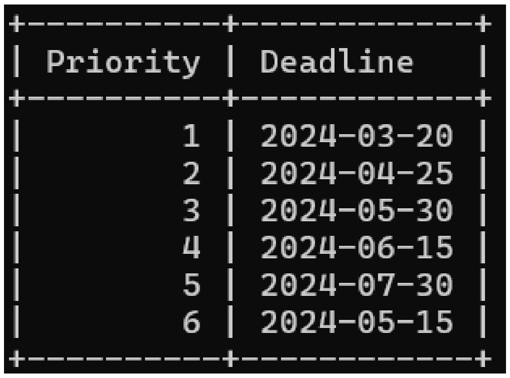
```
+----+--------------------+----------+
| ID | Description        | Severity |
+----+--------------------+----------+
|  1 | Bug description 1  |        3 |
|  2 | Bug description 2  |        2 |
|  3 | Bug description 3  |        1 |
|  4 | Bug description 4  |        2 |
|  5 | Bug description 5  |        3 |
|  6 | Bug Description 6  |        1 |
|  7 | Bug description 7  |        2 |
|  8 | Bug description 8  |        1 |
|  9 | Bug description 9  |        3 |
| 10 | Bug description 10 |        2 |
| 11 | Bug description 11 |        1 |
+----+--------------------+----------+
```

TaskHaveBugs(**BugID: integer**, **TaskID: integer**, **ProjectName: varchar**)



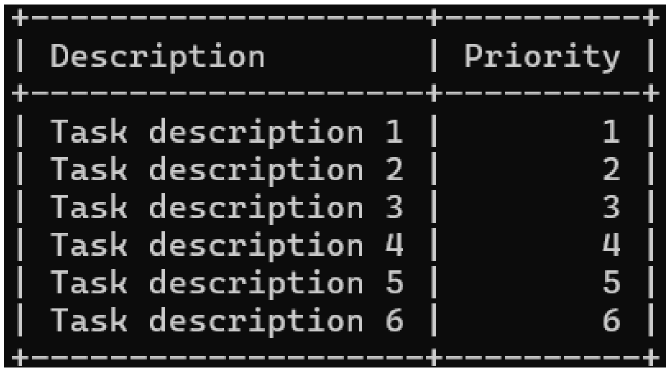
```
+-------+--------+-------------+
| BugID | TaskID | ProjectName |
+-------+--------+-------------+
|     1 |      1 | Search      |
|     2 |      2 | Ads         |
|     3 |      3 | Networking  |
|     4 |      4 | Classifier  |
|     6 |      4 | Classifier  |
|     7 |      4 | Classifier  |
|     8 |      4 | Classifier  |
|     9 |      4 | Classifier  |
|    10 |      4 | Classifier  |
|     5 |      5 | Frontend    |
|    11 |      6 | Classifier  |
+-------+--------+-------------+
```

PostNormTasksR1(**Priority: integer**, Deadline: date)



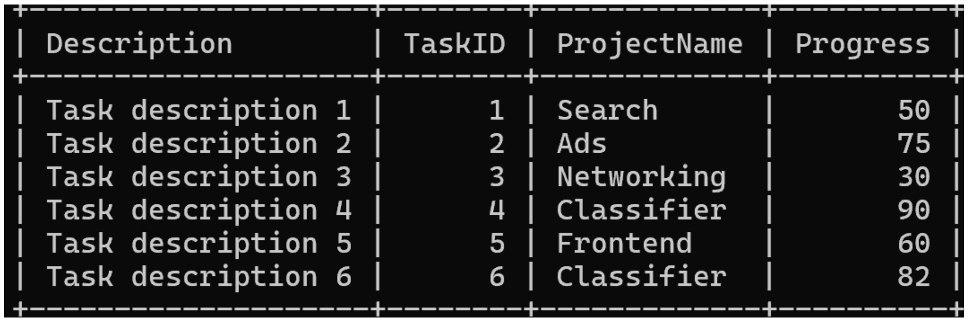
```
+----------+------------+
| Priority | Deadline   |
+----------+------------+
|        1 | 2024-03-20 |
|        2 | 2024-04-25 |
|        3 | 2024-05-30 |
|        4 | 2024-06-15 |
|        5 | 2024-07-30 |
|        6 | 2024-05-15 |
+----------+------------+
```

PostNormTasksR2(**Description: varchar**, Priority: integer)



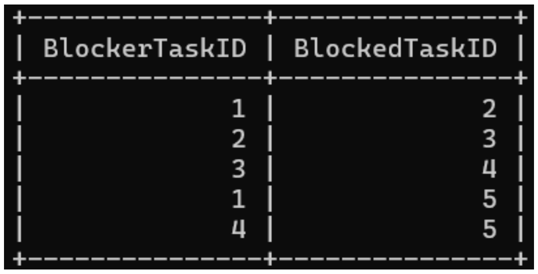
```
+--------------------+----------+
| Description        | Priority |
+--------------------+----------+
| Task description 1 |        1 |
| Task description 2 |        2 |
| Task description 3 |        3 |
| Task description 4 |        4 |
| Task description 5 |        5 |
| Task description 6 |        6 |
+--------------------+----------+
```

PostNormTasksR3(TaskID: integer, ProjectName: varchar, Description: varchar, Progress: integer)



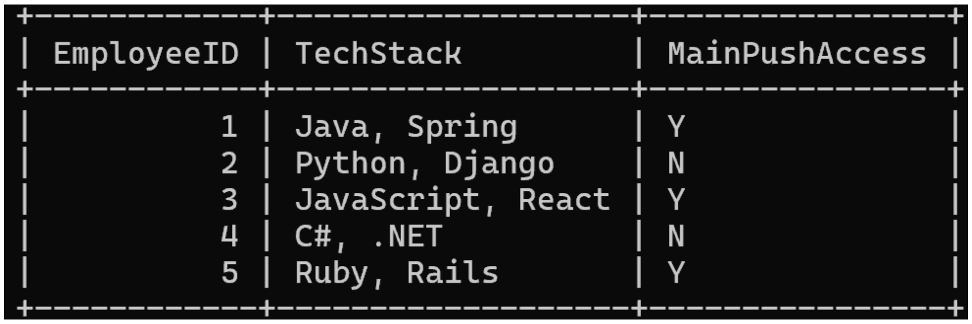
```
+--------------------+--------+-------------+----------+
| Description        | TaskID | ProjectName | Progress |
+--------------------+--------+-------------+----------+
| Task description 1 |      1 | Search      |       50 |
| Task description 2 |      2 | Ads         |       75 |
| Task description 3 |      3 | Networking  |       30 |
| Task description 4 |      4 | Classifier  |       90 |
| Task description 5 |      5 | Frontend    |       60 |
| Task description 6 |      6 | Classifier  |       82 |
+--------------------+--------+-------------+----------+
```

BlockedBy(**BlockerTaskID: integer, BlockedTaskID: integer**)



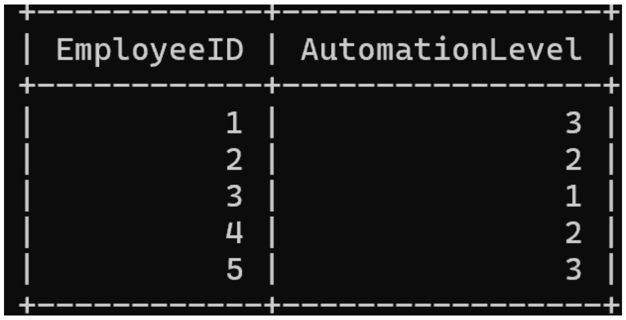
```
+---------------+---------------+
| BlockerTaskID | BlockedTaskID |
+---------------+---------------+
|             1 |             2 |
|             2 |             3 |
|             3 |             4 |
|             1 |             5 |
|             4 |             5 |
+---------------+---------------+
```

Engineers(**EmployeeID: integer**, TechStack: varchar, MainPushAccess: boolean)



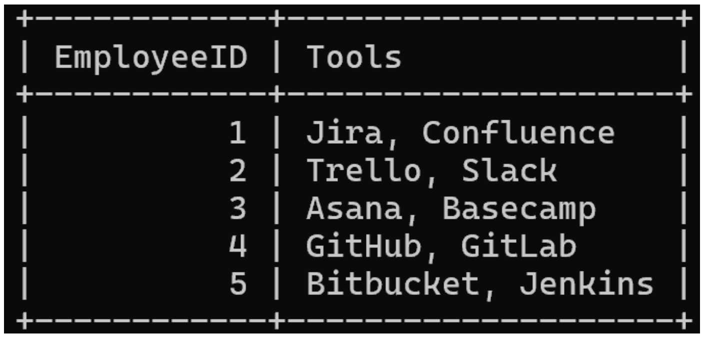
```
+------------+-------------------+----------------+
| EmployeeID | TechStack         | MainPushAccess |
+------------+-------------------+----------------+
|          1 | Java, Spring      | Y              |
|          2 | Python, Django    | N              |
|          3 | JavaScript, React | Y              |
|          4 | C#, .NET          | N              |
|          5 | Ruby, Rails       | Y              |
+------------+-------------------+----------------+
```

QAs(**EmployeeID: integer**, AutomationLevel: integer)



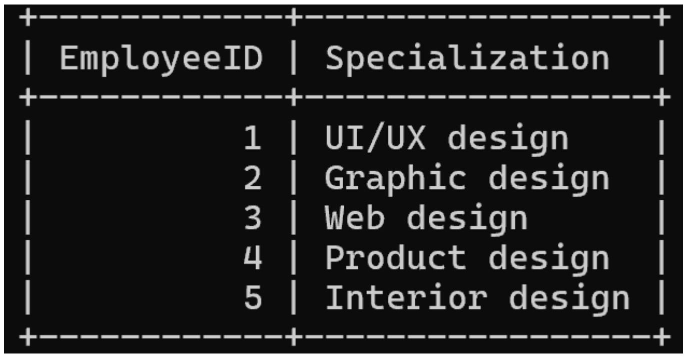
```
+------------+-----------------+
| EmployeeID | AutomationLevel |
+------------+-----------------+
|          1 |               3 |
|          2 |               2 |
|          3 |               1 |
|          4 |               2 |
|          5 |               3 |
+------------+-----------------+
```

Managers(**<u>EmployeeID: integer,</u>** Tools: varchar)



```
+------------+--------------------+
| EmployeeID | Tools              |
+------------+--------------------+
|          1 | Jira, Confluence   |
|          2 | Trello, Slack      |
|          3 | Asana, Basecamp    |
|          4 | GitHub, GitLab     |
|          5 | Bitbucket, Jenkins |
+------------+--------------------+
```

Designers(**<u>EmployeeID: integer</u>**, Specialization: varchar)



```
+------------+-------------------+
| EmployeeID | Specialization    |
+------------+-------------------+
|          1 | UI/UX design      |
|          2 | Graphic design    |
|          3 | Web design        |
|          4 | Product design    |
|          5 | Interior design   |
+------------+-------------------+
```

d. A list of all SQL queries used and where it can be found in the code (i.e., file name and line number(s)). For SQL query requirements, check the rubric listed on Canvas for Milestone 4.

1. <u>selection: selecting project name and deadline of Projects based on a condition entered by user in the WHERE clause</u>

File Name, Line Number: app.py, 90

2. <u>projection: selecting any number of attributes entered by the user from any tables</u>

File Name, Line Number: app.py, 158

3. <u>insertion: inserting attributes of a TeamMember tuple into the database, where EmployeeID and TeamID are not null; if TeamID does not exist, create it in the Teams table then perform the insertion of the TeamMember tuple</u>

File Name, Line Number: app.py, (293-300, 310-313)

4. <u>update: allow users to update any attributes from TeamMembers table with a specified EmployeeID</u>

File Name, Line Number: app.py, (372 - 376)

5. <u>delete: allow users to delete any tuples from TeamMembers table with a specified EmployeeID</u>

File Name, Line Number: app.py, (331 - 337)

6. <u>Aggregation with group by: finds the number of bugs associated with each task for a specified project</u>

File Name, Line Number: app.py, 195

7. <u>aggregation with having: finds the TaskID with a condition on number of bugs specified by the user for a specific project</u>

File Name, Line Number: app.py, 254

8. <u>join: joins the Bugs and ReportedBy tables based on EmployeeID</u>

File Name, Line Number: app.py, 223

9. <u>nested aggregation: find the average Teams size from Teams that are assigned to Projects</u>

File Name, Line Number: app.py, 115

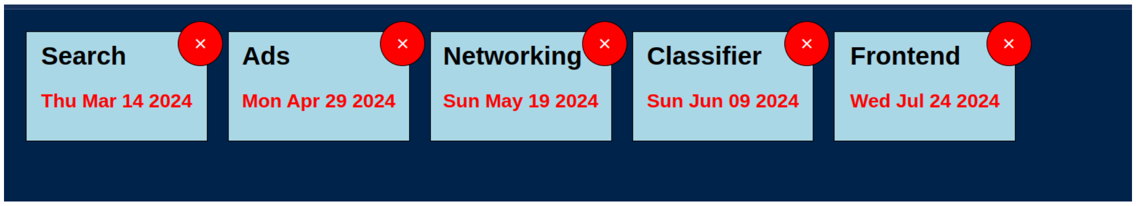
10. <u>division: finds the TeamID of Teams who are assigned to all projects</u>

File Name, Line Number: app.py, 136

e. Screenshots demonstrating the functionality of each query using the GUI. We want to see a before/during/after progression of events. For example, the before screenshot would be what data is in the table before you run the query, the during screenshot(s) is how the query is triggered using the GUI, and the after screenshot is what data is in your table afterwards. Please label each set of screenshots with the name of the query it is meant to address (e.g., "Insert Operation").

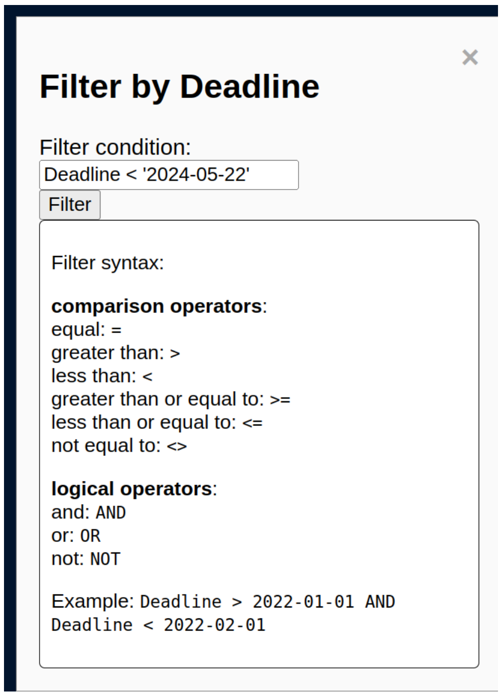
<u>selection: selecting project name and deadline of Projects based on a condition entered by user in the WHERE clause</u>

before



Search	Ads	Networking	Classifier	Frontend
Thu Mar 14 2024	Mon Apr 29 2024	Sun May 19 2024	Sun Jun 09 2024	Wed Jul 24 2024

during



Filter by Deadline

Filter condition:
Deadline < '2024-05-22'

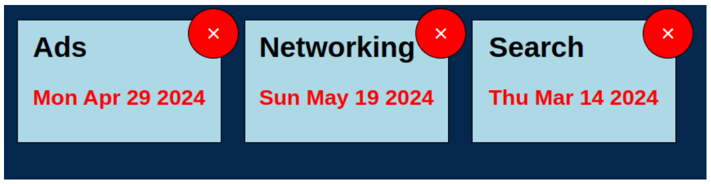
Filter syntax:

comparison operators:
equal: =
greater than: >
less than: <
greater than or equal to: >=
less than or equal to: <=
not equal to: <>

logical operators:
and: `AND`
or: `OR`
not: `NOT`

Example: `Deadline > 2022-01-01 AND Deadline < 2022-02-01`

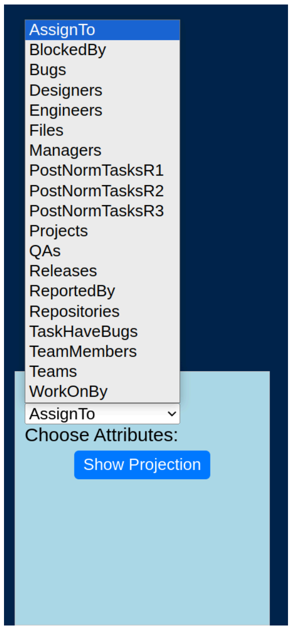
after



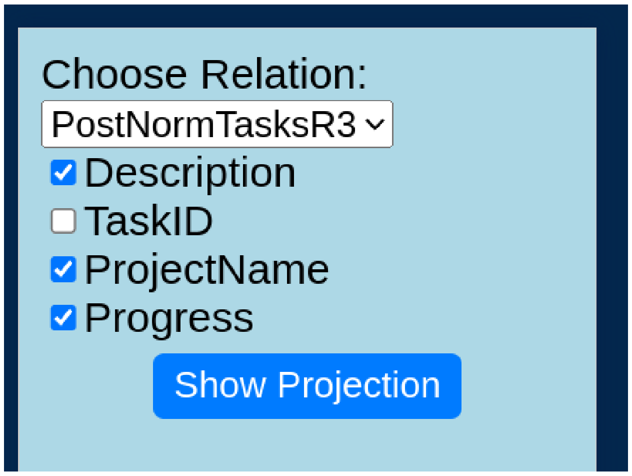
Ads	Networking	Search
Mon Apr 29 2024	Sun May 19 2024	Thu Mar 14 2024

<u>projection: selecting any number of attributes entered by the user from any table</u>

<u>before</u>

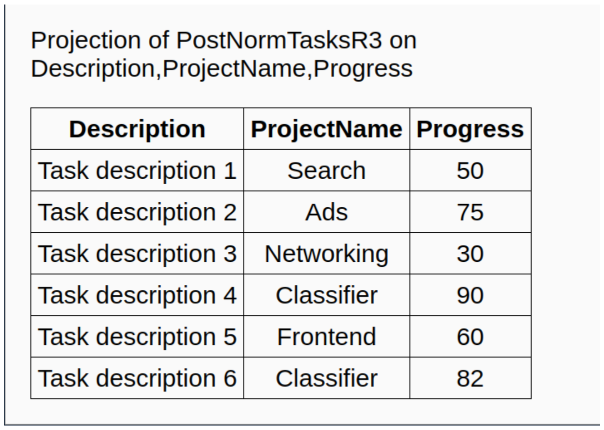


during


Choose Relation: PostNormTasksR3

- [x] Description
- [ ] TaskID
- [x] ProjectName
- [x] Progress

after

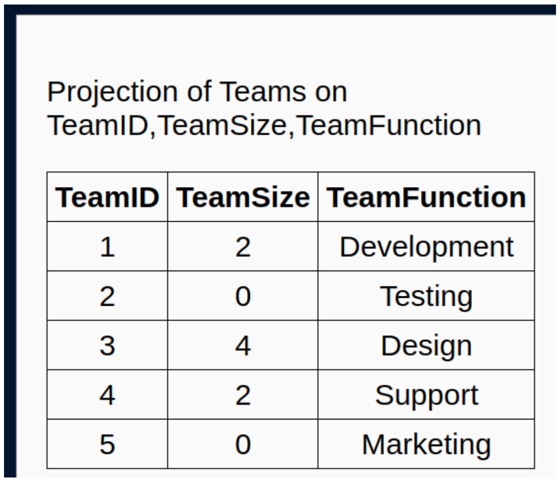

Projection of PostNormTasksR3 on Description,ProjectName,Progress

Description	ProjectName	Progress
Task description 1	Search	50
Task description 2	Ads	75
Task description 3	Networking	30
Task description 4	Classifier	90
Task description 5	Frontend	60
Task description 6	Classifier	82

insertion: inserting attributes of a TeamMember tuple into the database, where EmployeeID and TeamID are not null; if TeamID does not exist, create it in the Teams table then perform the insertion of the TeamMember tuple

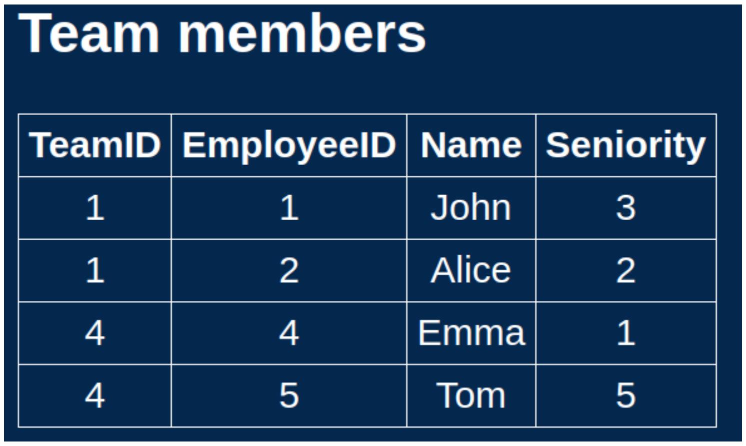
case 1: TeamID exists

before



Projection of Teams on TeamID,TeamSize,TeamFunction

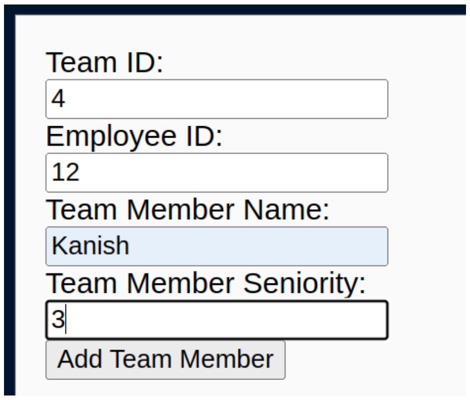
TeamID	TeamSize	TeamFunction
1	2	Development
2	0	Testing
3	4	Design
4	2	Support
5	0	Marketing



Team members

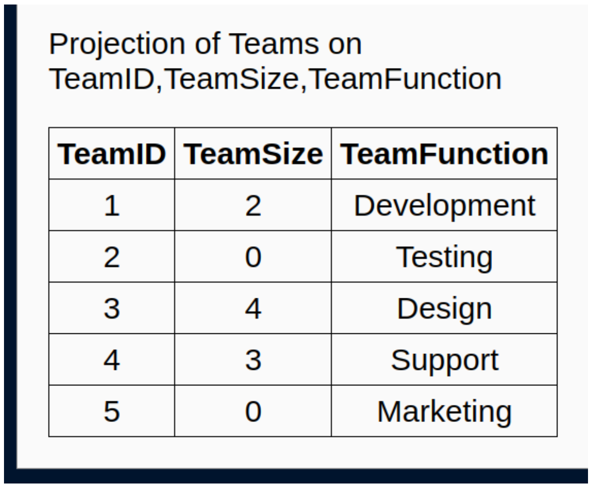
TeamID	EmployeeID	Name	Seniority
1	1	John	3
1	2	Alice	2
4	4	Emma	1
4	5	Tom	5

during



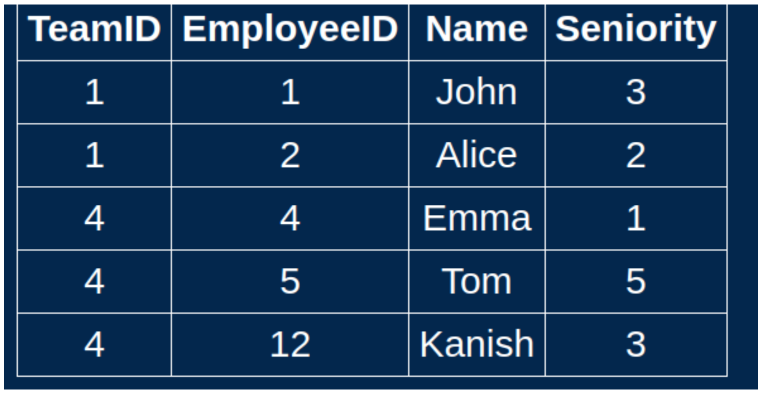
Team ID: 4
Employee ID: 12
Team Member Name: Kanish
Team Member Seniority: 3

after



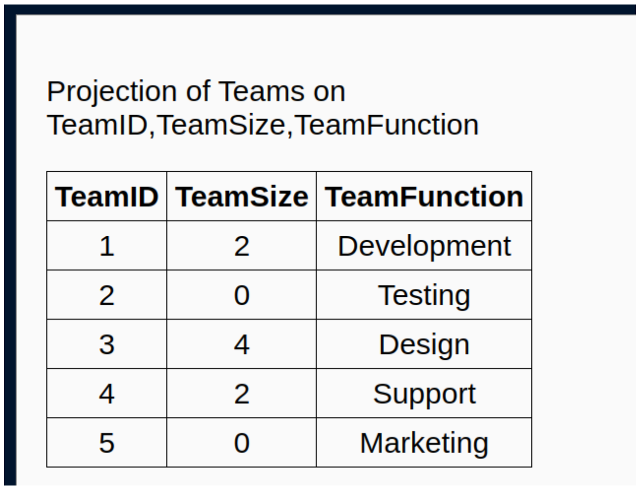
Projection of Teams on TeamID,TeamSize,TeamFunction

TeamID	TeamSize	TeamFunction
1	2	Development
2	0	Testing
3	4	Design
4	3	Support
5	0	Marketing



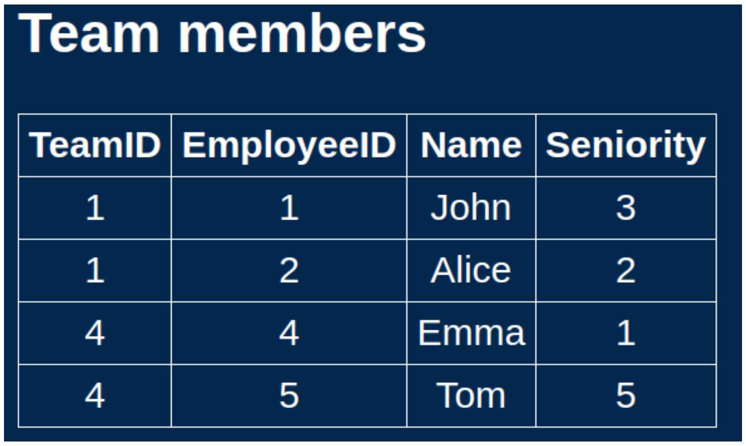
TeamID	EmployeeID	Name	Seniority
1	1	John	3
1	2	Alice	2
4	4	Emma	1
4	5	Tom	5
4	12	Kanish	3

case 2: TeamID does not exist
before



Projection of Teams on TeamID,TeamSize,TeamFunction

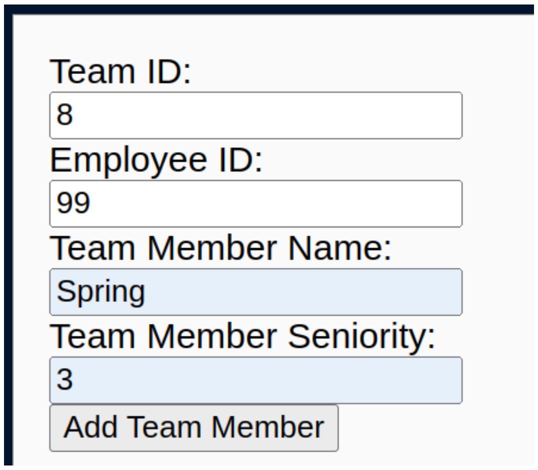
TeamID	TeamSize	TeamFunction
1	2	Development
2	0	Testing
3	4	Design
4	2	Support
5	0	Marketing



Team members

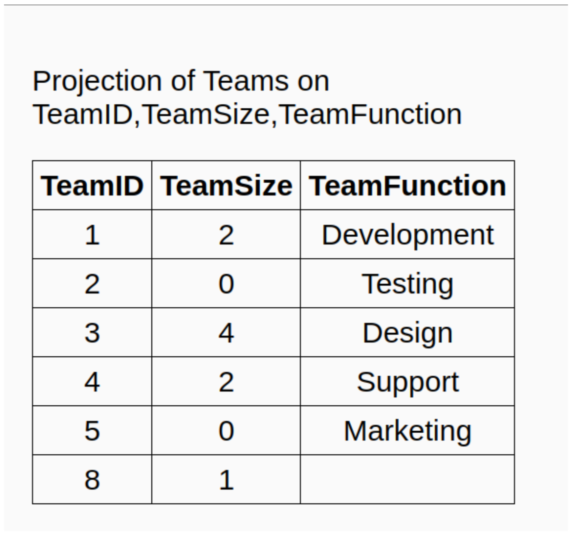
TeamID	EmployeeID	Name	Seniority
1	1	John	3
1	2	Alice	2
4	4	Emma	1
4	5	Tom	5

during



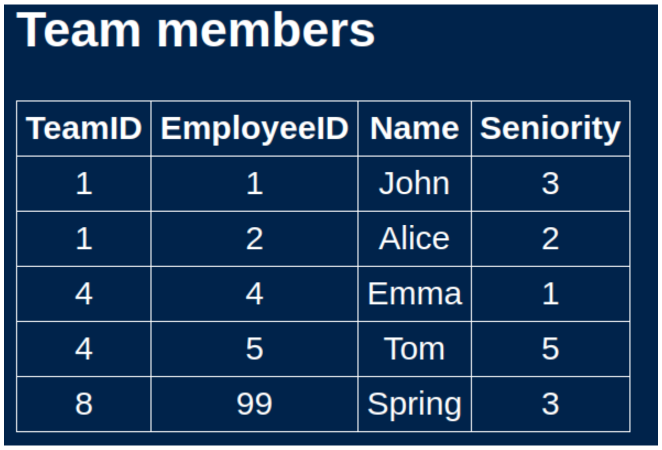
Team ID:	8
Employee ID:	99
Team Member Name:	Spring
Team Member Seniority:	3

after



Projection of Teams on TeamID,TeamSize,TeamFunction

TeamID	TeamSize	TeamFunction
1	2	Development
2	0	Testing
3	4	Design
4	2	Support
5	0	Marketing
8	1	

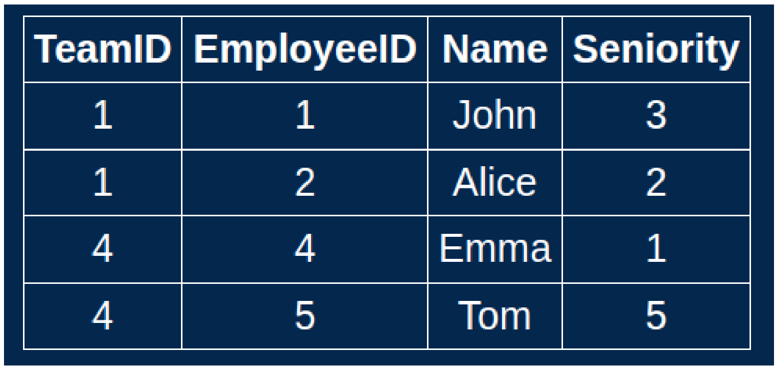


Team members

TeamID	EmployeeID	Name	Seniority
1	1	John	3
1	2	Alice	2
4	4	Emma	1
4	5	Tom	5
8	99	Spring	3

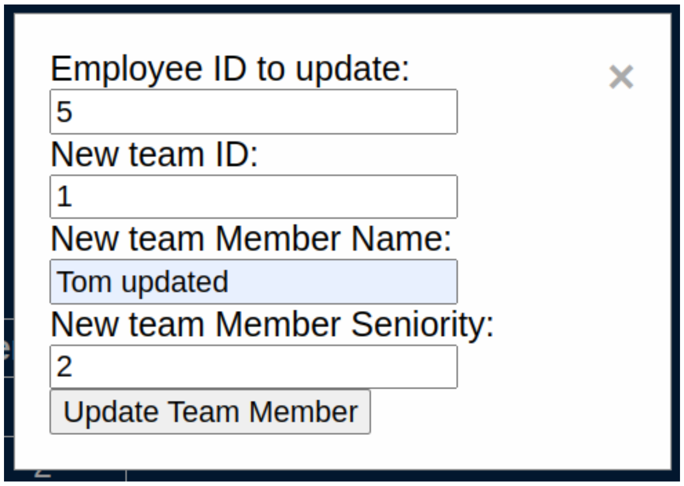
update: allow users to update any attributes from TeamMembers table with a specified EmployeeID

before



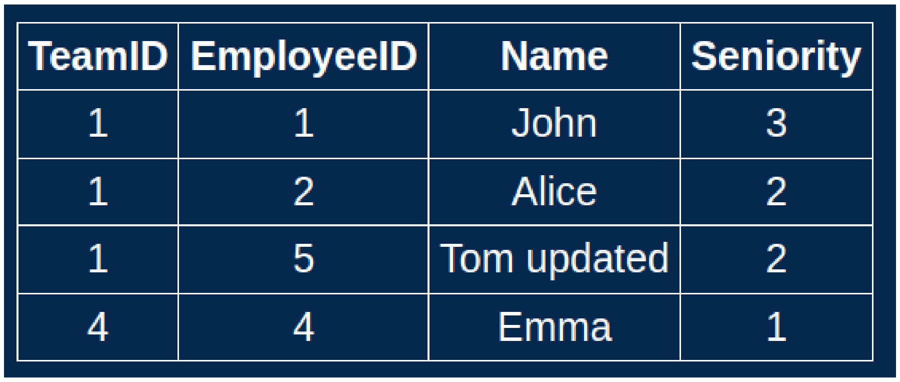
TeamID	EmployeeID	Name	Seniority
1	1	John	3
1	2	Alice	2
4	4	Emma	1
4	5	Tom	5

during



Employee ID to update:	5
New team ID:	1
New team Member Name:	Tom updated
New team Member Seniority:	2

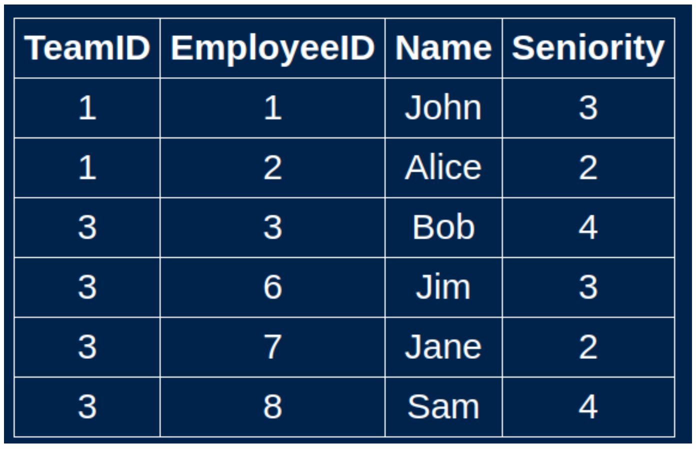
after



TeamID	EmployeeID	Name	Seniority
1	1	John	3
1	2	Alice	2
1	5	Tom updated	2
4	4	Emma	1

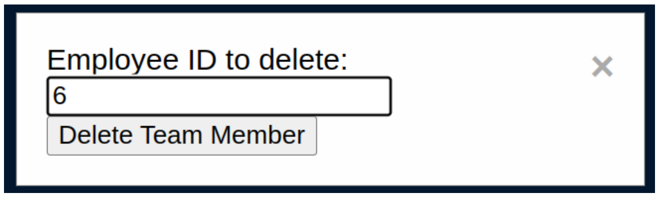
delete: allow users to delete any tuples from TeamMembers table with a specified EmployeeID

before



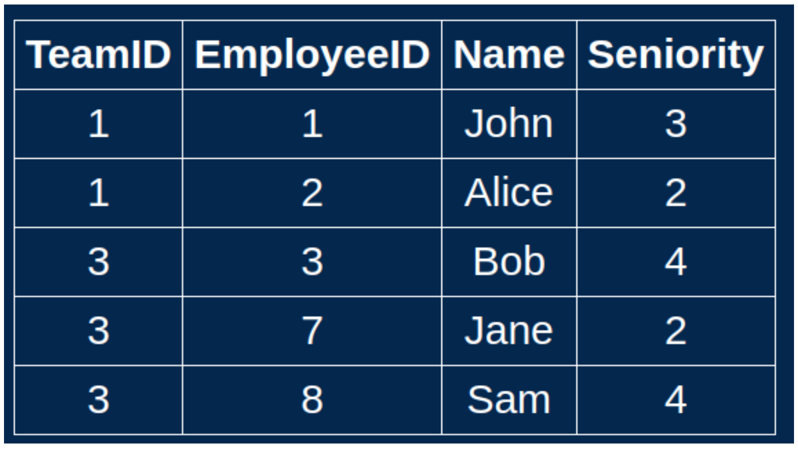
TeamID	EmployeeID	Name	Seniority
1	1	John	3
1	2	Alice	2
3	3	Bob	4
3	6	Jim	3
3	7	Jane	2
3	8	Sam	4

during



Employee ID to delete:
6
Delete Team Member

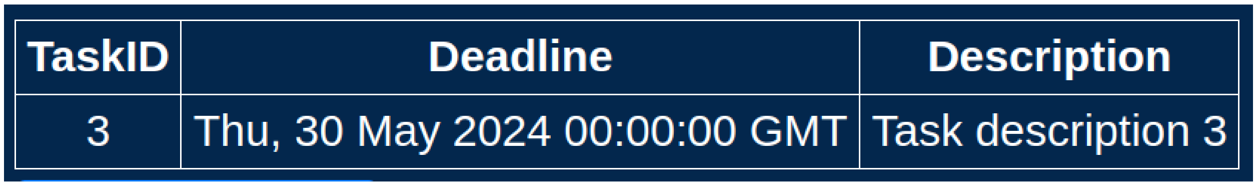
after



TeamID	EmployeeID	Name	Seniority
1	1	John	3
1	2	Alice	2
3	3	Bob	4
3	7	Jane	2
3	8	Sam	4

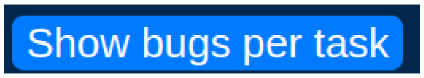
aggregation with group by: finds the number of bugs associated with each task for a specified project

before

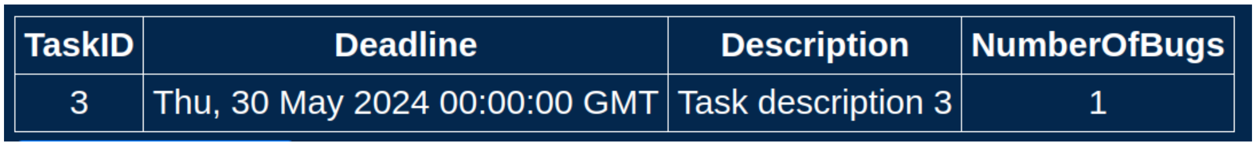


TaskID	Deadline	Description
3	Thu, 30 May 2024 00:00:00 GMT	Task description 3

during



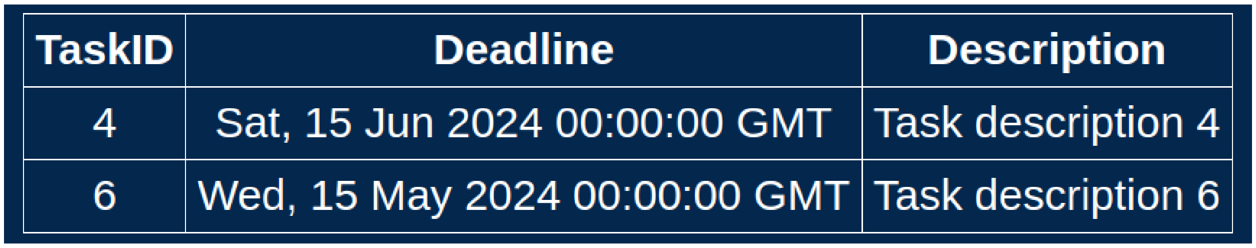
after



TaskID	Deadline	Description	NumberOfBugs
3	Thu, 30 May 2024 00:00:00 GMT	Task description 3	1

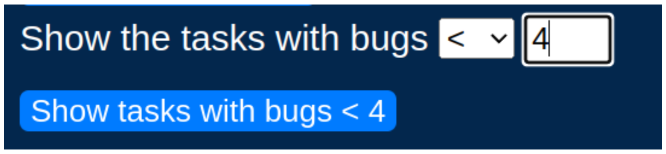
aggregation with having: finds the TaskID with a condition on number of bugs specified by the user for a specific project

before

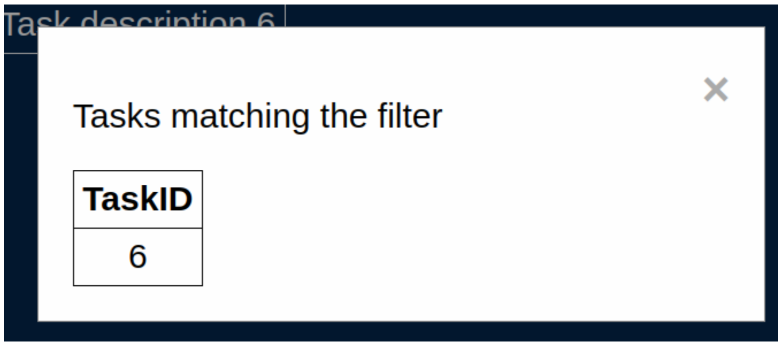


TaskID	Deadline	Description
4	Sat, 15 Jun 2024 00:00:00 GMT	Task description 4
6	Wed, 15 May 2024 00:00:00 GMT	Task description 6

during



after

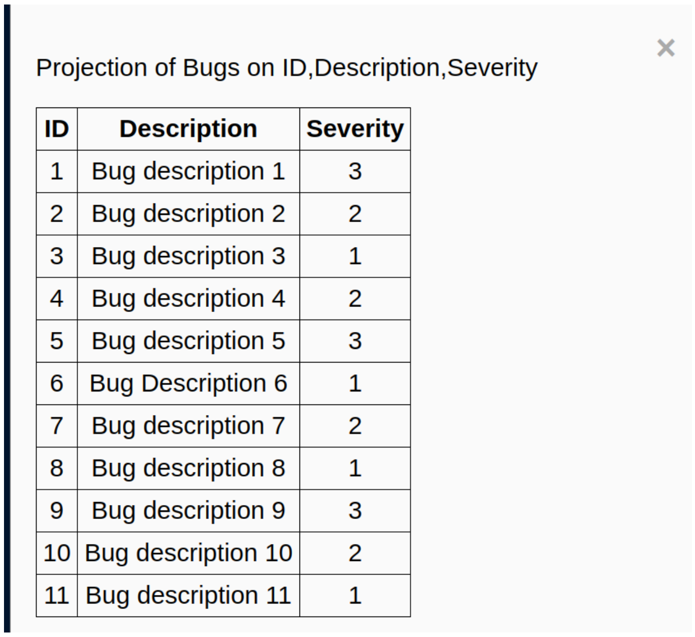


Tasks matching the filter

TaskID
6

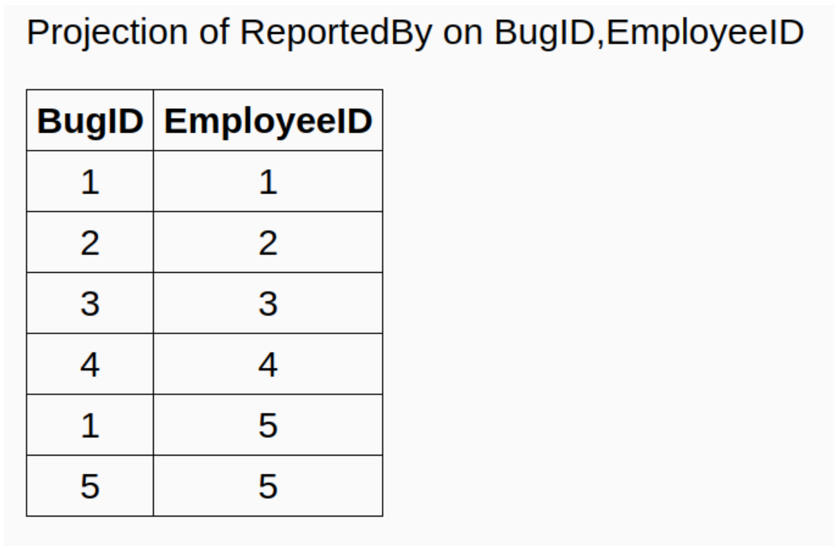
join: joins the Bugs and ReportedBy tables based on EmployeeID

before


×

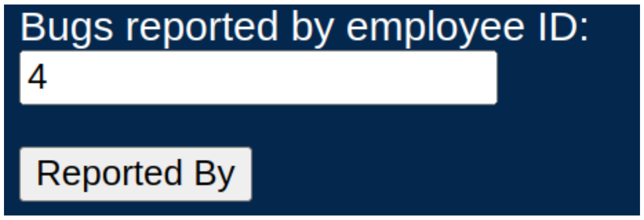
Projection of Bugs on ID,Description,Severity

ID	Description	Severity
1	Bug description 1	3
2	Bug description 2	2
3	Bug description 3	1
4	Bug description 4	2
5	Bug description 5	3
6	Bug Description 6	1
7	Bug description 7	2
8	Bug description 8	1
9	Bug description 9	3
10	Bug description 10	2
11	Bug description 11	1

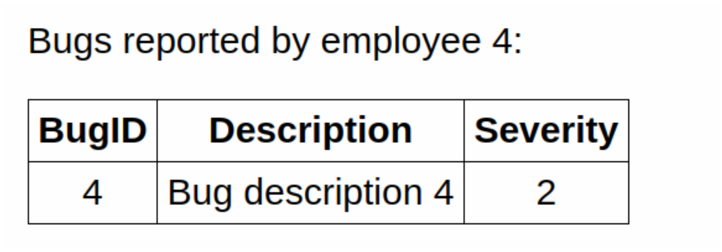

Projection of ReportedBy on BugID,EmployeeID

BugID	EmployeeID
1	1
2	2
3	3
4	4
1	5
5	5

during


Bugs reported by employee ID:
4

after

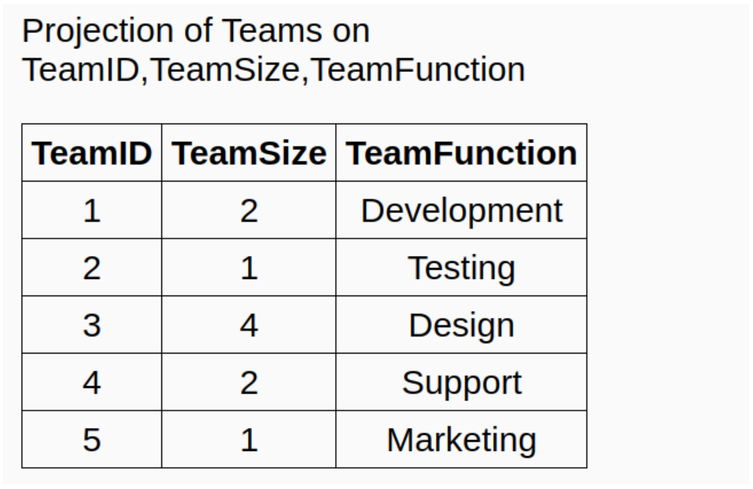


Bugs reported by employee 4:

BugID	Description	Severity
4	Bug description 4	2

<u>nested aggregation: find the average Teams size from Teams that are assigned to Projects</u>

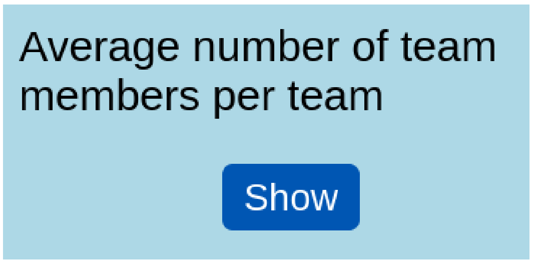
before



Projection of Teams on TeamID,TeamSize,TeamFunction

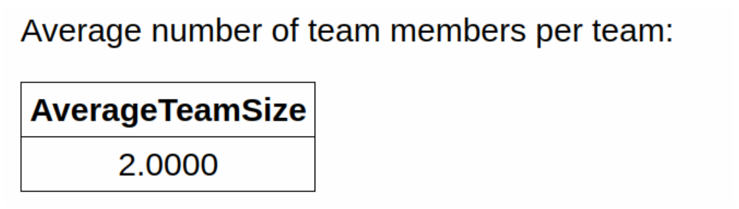
TeamID	TeamSize	TeamFunction
1	2	Development
2	1	Testing
3	4	Design
4	2	Support
5	1	Marketing

during



Average number of team members per team

after



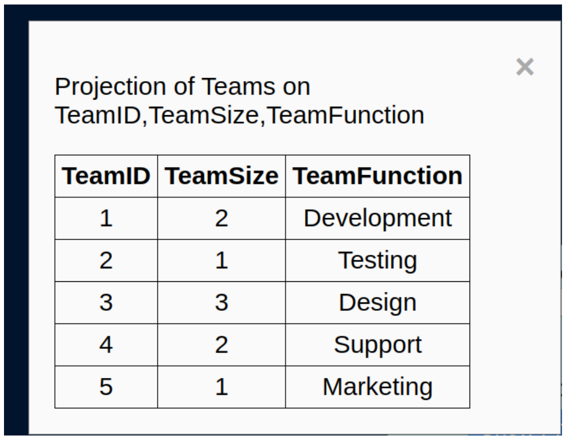
Average number of team members per team:

AverageTeamSize
2.0000

division: finds the employeeID from TeamMembers who are assigned to all projects

before

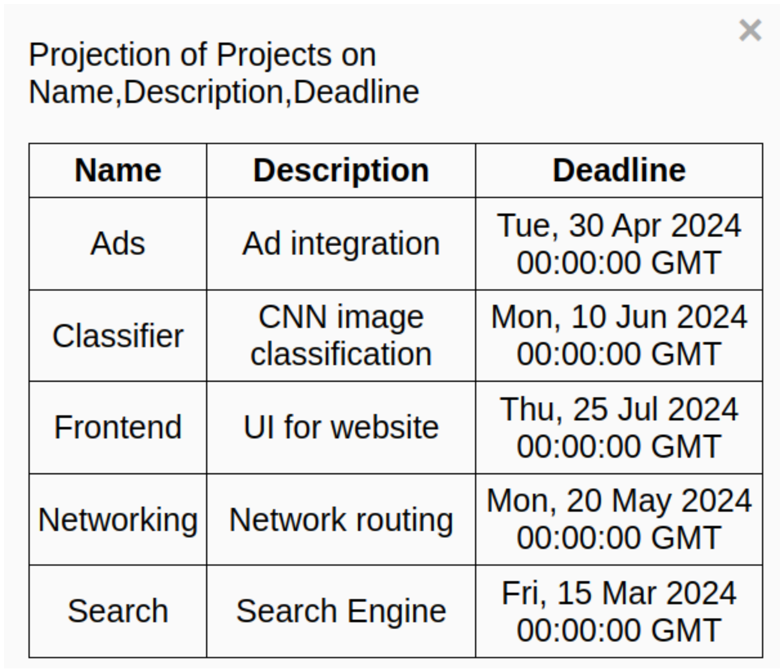
Teams Table:



Projection of Teams on TeamID,TeamSize,TeamFunction

TeamID	TeamSize	TeamFunction
1	2	Development
2	1	Testing
3	3	Design
4	2	Support
5	1	Marketing

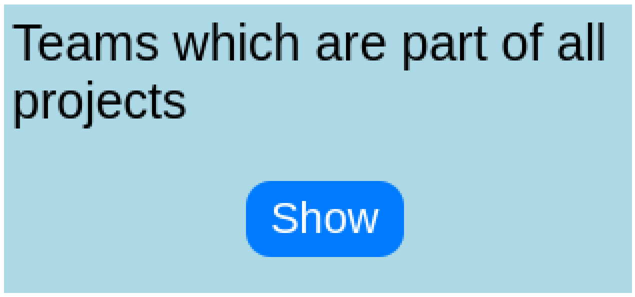
Projects:



Projection of Projects on Name,Description,Deadline

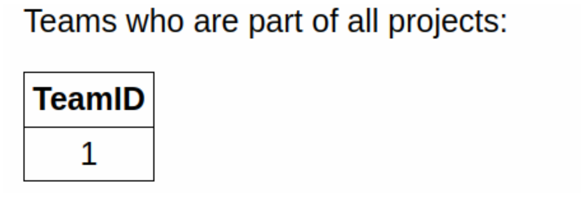
Name	Description	Deadline
Ads	Ad integration	Tue, 30 Apr 2024 00:00:00 GMT
Classifier	CNN image classification	Mon, 10 Jun 2024 00:00:00 GMT
Frontend	UI for website	Thu, 25 Jul 2024 00:00:00 GMT
Networking	Network routing	Mon, 20 May 2024 00:00:00 GMT
Search	Search Engine	Fri, 15 Mar 2024 00:00:00 GMT

during



Teams which are part of all projects

after



Teams who are part of all projects:

TeamID
1