

# MySQL Week 8 Exercises

## Instructions

Points possible: 75


URL to GitHub Repository: <https://github.com/SkHahn/Projects>

URL to Public Link of your Video : <https://youtu.be/oyPal-RyLNw>


---

### Instructions :

1. Follow the [Exercises](#) below to complete this assignment.

- In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed.
- Create a new repository on GitHub for this week's assignment and push your completed code to this dedicated repo, including your entire Maven Project Directory (e.g., mysql-java) and any .sql files that you create. In addition, screenshot your ERD and push the screenshot to your GitHub repo.
- Include the screenshots into this Assignment Document indicated by: 
- Create a video showcasing your work:
  - In this video: record and present your project verbally while showing the results of the working project.
  - Easy way to Create a video: Start a meeting in Zoom, share your screen, open Eclipse with the code and your Console window, start recording & record yourself describing and running the program showing the results.
  - Your video should be a maximum of 5-minutes.
  - Upload your video with a public link.
  - Easy way to Create a Public Video Link: Upload your video recording to YouTube with a public link.

2. In addition, please include the following in your Coding Assignment Document:

- The requested screenshots, indicated by: 
- The URL for this week's GitHub repository.
- The URL of the public link of your video.

3. Save the Coding Assignment Document as a .pdf and do the following:

- Push the .pdf to the GitHub repo for this week.
  - Upload the .pdf to the LMS in your Coding Assignment Submission.
-

# MySQL Week 8 Exercises

## Background


You are in the process of creating an application that will perform CRUD (Create, Read, Update, and Delete) operations on a MySQL database. In the week 1 exercises you wrote code to connect to a MySQL database using the Java Database Connectivity (JDBC) API. In these exercises, you will diagram the database tables using Draw.io to create an Entity-Relationship Diagram. You will then write the SQL statements to create the five project tables. Lastly, you will use a MySQL client (DBeaver) to create the tables.

## Objectives

In these exercises, you will:

- Learn how to create an Entity-Relationship Diagram (ERD) in Draw.io with entities and relationship lines.
- Learn about crow's foot notation and how to apply that knowledge in an Entity Relationship Diagram.
- Apply your knowledge of DROP TABLE and CREATE TABLE statements to create tables using a MySQL client (DBeaver),

## Important

In the exercises below, you will see this icon: . This means to take a screen shot or snip showing the results of the action or the code in the editor.

## Exercises

Use the column definitions in the table below for the Entity-Relationship Diagram (ERD) and the Create Table statements.

Table	Column	Data Type	Nullable	Comment
<b>project</b>	project_id	int	No	primary key
	project_name	varchar(128)	No	
	estimated_hours	decimal(7,2)	Yes	
	actual_hours	decimal(7,2)	Yes	
	difficulty	int	Yes	
	notes	text	Yes	
<b>material</b>	material_id	int	No	primary key
	project_id	int	No	foreign key

## MySQL Week 8 Exercises

	material_name	varchar(128)	No	
	num_required	int	Yes	
	cost	decimal(7,2)	Yes	
<b>step</b>	step_id	int	No	primary key
	project_id	int	No	foreign key
	step_text	text	No	
	step_order	int	No	
<b>category</b>	category_id	int	No	primary key
	category_name	varchar(128)	No	
<b>project_category</b>	project_id	int	No	foreign key, unique key with category_id
	category_id	int	No	foreign key, unique key with project_id

### Entity-Relationship Diagram

Documenting a project is an essential skill so that the project will make sense to others who want to know about the project. This includes network diagrams, Entity-Relationship Diagrams, well-commented code and readme files.

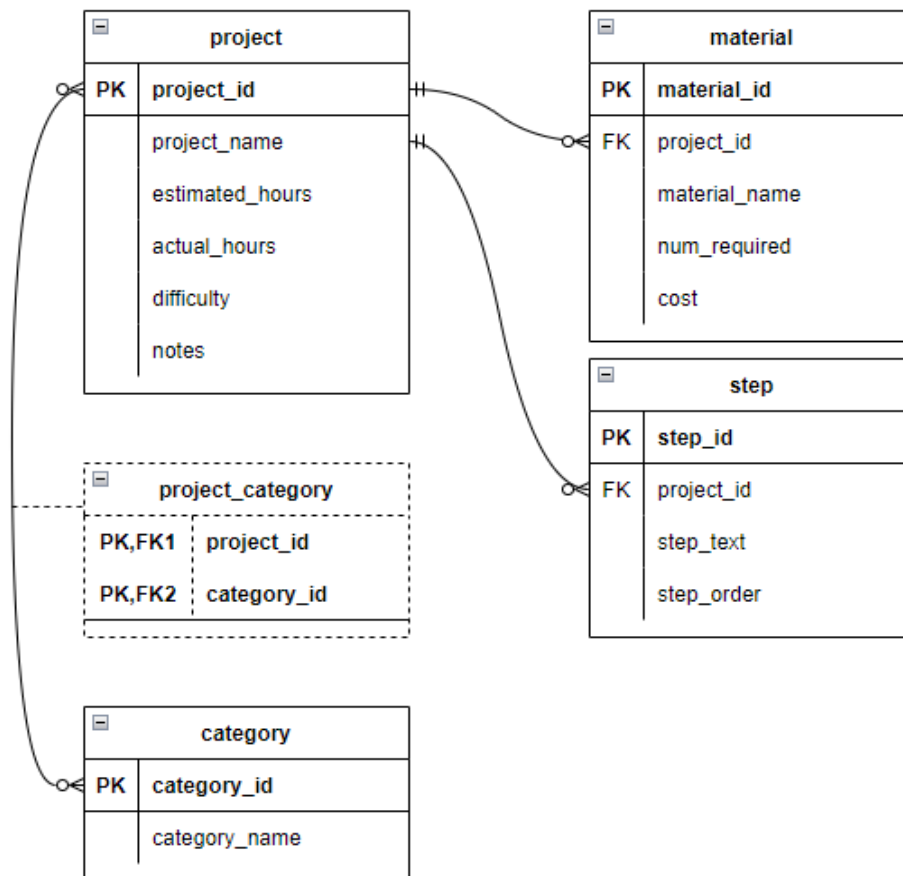
In this section, you will create an Entity-Relationship Diagram. This diagram will contain the five table entities and show the relationships between the tables. A good ERD can quickly orient future developers to the data that the application will work with.

Use your knowledge learned in the videos and follow the instructions in this section to create an Entity-Relationship Diagram of the DIY Projects schema.

1. Follow the instructions in the Week 2 Installation Instructions found in the resources packet to either download Draw.io or to use the online tool.
2. In Draw.io, create a new drawing and expand "Entity Relation" in the tool palette on the left.

## MySQL Week 8 Exercises

3. Use Draw.io to create an Entity-Relationship Diagram. Save the file. The file must be uploaded to your GitHub repository for Week 2. Note that it should look similar to the diagram below.




### Project Schema

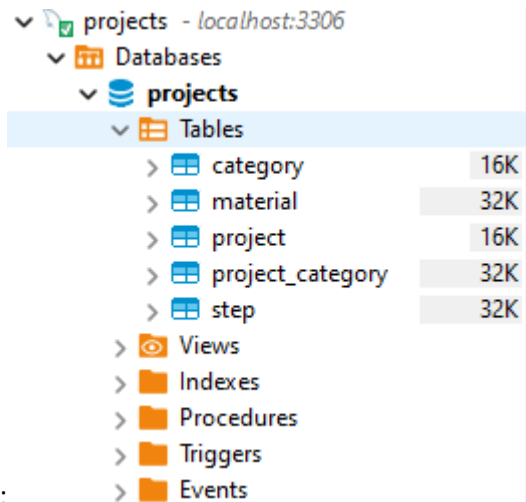
In this section, you will create a new file and write DROP TABLE and CREATE TABLE SQL statements for the five tables in the DIY Projects schema. The goal for this section is to copy the SQL statements from the file, drop them into the DBeaver SQL editor, and have DBeaver send the instructions to the MySQL server so that the five tables are created with no errors.

Creating the tables is the first step to having the application read from and write to them. Next week you will begin that process.

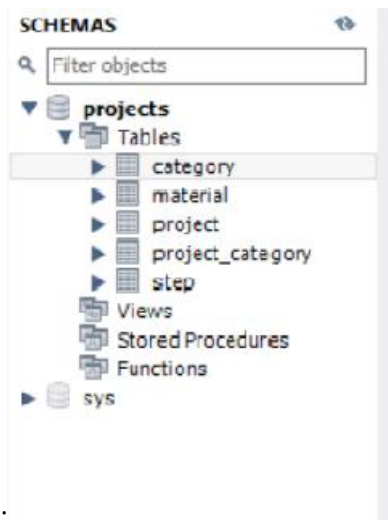
1. In the mysql-java project in Eclipse, create a file named "projects-schema.sql" in the src/main/resources directory.
2. Add DROP TABLE statements at the top of the file to drop the tables in the correct order. There can be some variation but the tables with dependencies (foreign key references to other tables) must be dropped first. The tables should only be dropped if they exist.
3. Write the CREATE TABLE statements in the inverse of the order that they were dropped. Include the following:

## MySQL Week 8 Exercises

- a. Auto-increment the primary key columns.
  - b. Primary key statements.
  - c. Foreign key statements with appropriate ON DELETE CASCADE.
4. Remember to close each DROP TABLE and CREATE TABLE statement with a semicolon.
  5. Paste the SQL into the DBeaver SQL editor. Run all DROP TABLE and CREATE TABLE statements. Include a screen shot of the created tables as shown in DBeaver's connection explorer (leftmost panel).  It should look like this:



Example:



Mine:

6. Push your project to GitHub.

# MySQL Week 8 Exercises

**SCHEMAS**

Filter objects

- ▼ projects
  - ▼ Tables
    - category
    - material
    - project
    - project\_category
    - step
  - Views
  - Stored Procedures
  - Functions
- ▼ sys

Administration Schemas

Information

**Table: category**

Columns:

category_id	int AI PK
category_name	varchar(128)

```

1 DROP TABLE IF EXISTS material;
2 DROP TABLE IF EXISTS step;
3 DROP TABLE IF EXISTS project_category;
4 DROP TABLE IF EXISTS category;
5 DROP TABLE IF EXISTS project;
6
7 CREATE TABLE project (
8     project_id INT AUTO_INCREMENT NOT NULL,
9     project_name VARCHAR(128) NOT NULL,
10    estimated_hours DECIMAL(7,2),
11    actual_hours DECIMAL(7,2),
12    difficulty INT,
13    notes TEXT,
14    PRIMARY KEY (project_id)
15 );
  
```

Output

#	Time	Action
✓ 115	18:05:37	DROP TABLE IF EXISTS material
✓ 116	18:05:37	DROP TABLE IF EXISTS step
✓ 117	18:05:37	DROP TABLE IF EXISTS project_category
✓ 118	18:05:37	DROP TABLE IF EXISTS category
✓ 119	18:05:37	DROP TABLE IF EXISTS project
✓ 120	18:05:37	CREATE TABLE project (project_id INT AUTO_INCREMENT NOT NULL, project_name ...
✓ 121	18:05:37	CREATE TABLE category (category_id INT AUTO_INCREMENT NOT NULL, category_...
✓ 122	18:05:37	CREATE TABLE project_category (project_id INT NOT NULL, category_id INT NOT NU...
✓ 123	18:05:37	CREATE TABLE step (step_id INT AUTO_INCREMENT NOT NULL, project_id INT NOT...
✓ 124	18:05:37	CREATE TABLE material (material_id INT AUTO_INCREMENT NOT NULL, project_id IN...

```

erDiagram
    project ||--o{ step : "has"
    project ||--o{ material : "has"
    project ||--o{ project_category : "has"
    project ||--o{ category : "has"
    project_category ||--o{ project : "belongs to"
    project_category ||--o{ category : "belongs to"
    step ||--o{ project : "belongs to"
    material ||--o{ project : "belongs to"
  
```

The ER diagram illustrates the following tables and their attributes:

- project**: project\_id (PK), project\_name, estimated\_hours, actual\_hours, difficulty, notes
- step**: step\_id (PK), project\_id (FK), step\_text, step\_order
- material**: material\_id (PK), project\_id (FK), material\_name, num\_required, cost
- project\_category**: project\_id (PK, FK1), category\_id (PK, FK2)
- category**: category\_id (PK), category\_name

Relationships are shown as follows:

- project** is the primary entity, with one-to-many relationships to **step**, **material**, **project\_category**, and **category**.
- project\_category** is a junction table with foreign keys to **project** and **category**.
- step** and **material** have foreign keys to **project**.