

Design DB model for Guvi Zen class:

Entities:

1. users
2. courses
3. modules
4. tasks
5. queries
6. leaveApplication

Attributes:

1. users
 - user_id
 - first_name
 - last_name
 - email
 - password
 - role
2. courses
 - course_id
 - course_name
 - course_description
 - start_date
 - end_date
 - instructor_id
3. modules
 - module_id
 - module_name
 - module_description
 - course_id

4. tasks

- task_id
- module_id
- task_description
- due_date

5. queries

- query_id
- module_id
- user_id
- question
- answer
- status

6. leaveApplication

- leave_id
- user_id
- start_date
- end_date
- reason
- status

Relationships:

1. Users - Courses (One-to-Many)
 2. Courses - Modules (One-to-Many)
 3. Modules - Tasks (One-to-Many)
 4. Users - Queries (One-to-Many)
 5. Modules - Queries (One-to-Many)
 6. Users - LeaveApplications (One-to-Many)
-

1.Users:

```
```sql
create table users (
 user_id int not null auto_increment,
 first_name varchar(50) not null,
 last_name varchar (50) not null,
 email varchar (100) not null unique,
 password varchar(255) not null,
 role varchar(20) not null,
 primary key (user_id)
);

insert into users (first_name, last_name, email, password, role)
values
('Alice', 'Smith', 'alice.smith@example.com', 'password123', 'Instructor'),
('Bob', 'Johnson', 'bob.johnson@example.com', 'password123', 'Instructor'),
('Charlie', 'Brown', 'charlie.brown@example.com', 'password123', 'Student');
```
```

Output:

The screenshot shows a database management interface. On the left, a sidebar lists the database schema 'GuvizZenClassDB' with various tables and views. The 'Users' table is selected. The main area displays the 'Users' table with columns: user_id, first_name, last_name, email, password, and role. The table contains three rows of data: Alice Smith (Instructor), Bob Johnson (Instructor), and Charlie Brown (Student).

| user_id | first_name | last_name | email | password | role |
|---------|------------|-----------|---------------------------|-------------|------------|
| 1 | Alice | Smith | alice.smith@example.com | password123 | Instructor |
| 2 | Bob | Johnson | bob.johnson@example.com | password123 | Instructor |
| 3 | Charlie | Brown | charlie.brown@example.com | password123 | Student |

List all users, showing their full name and role from users table ?

The screenshot shows a database management interface with a dark theme. On the left, a 'SCHEMAS' sidebar lists a database named 'GuviZenClassDB' with several tables, including 'Users'. The main area displays a SQL query: `SELECT CONCAT(first_name, ' ', last_name) AS full_name, role FROM Users;`. Below the query editor, a 'Result Grid' shows the output of the query. The grid has two columns: 'full_name' and 'role'. It contains three rows of data: 'Alice Smith' (Instructor), 'Bob Johnson' (Instructor), and 'Charlie Brown' (Student). The interface also includes a search bar, a 'Filter Rows' button, and an 'Export' button.

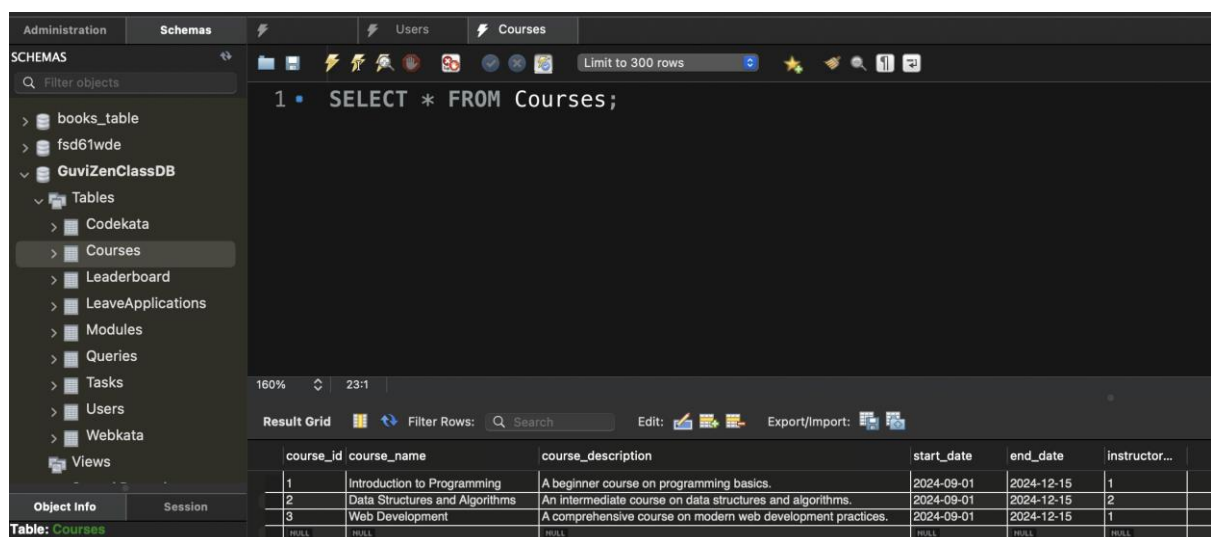
| full_name | role |
|---------------|------------|
| Alice Smith | Instructor |
| Bob Johnson | Instructor |
| Charlie Brown | Student |

2.Courses:

```
```sql
create table courses (
 course_id int not null auto_increment,
 course_name varchar(100) not null,
 course_description text,
 start_date date not null,
 end_date date not null,
 instructor_id int not null,
 primary key (course_id),
 foreign key (instructor_id) references users(user_id)
);

insert into courses (course_name, course_description, start_date, end_date,
instructor_id)
values
('Introduction to Programming', 'A beginner course on programming basics.',
'2024-09-01', '2024-12-15', 1),
('Data Structures and Algorithms', 'An intermediate course on data structures
and algorithms.', '2024-09-01', '2024-12-15', 2),
('Web Development', 'A comprehensive course on modern web development
practices.', '2024-09-01', '2024-12-15', 1);
```
```

Output:



The screenshot shows a database management interface with a sidebar on the left containing a tree view of schemas and tables. The main area displays a SQL query: `SELECT * FROM Courses;` and the resulting data in a table format. The table has 6 columns: `course_id`, `course_name`, `course_description`, `start_date`, `end_date`, and `instructor...`. There are three rows of data.

| course_id | course_name | course_description | start_date | end_date | instructor... |
|-----------|--------------------------------|---|------------|------------|---------------|
| 1 | Introduction to Programming | A beginner course on programming basics. | 2024-09-01 | 2024-12-15 | 1 |
| 2 | Data Structures and Algorithms | An intermediate course on data structures and algorithms. | 2024-09-01 | 2024-12-15 | 2 |
| 3 | Web Development | A comprehensive course on modern web development practices. | 2024-09-01 | 2024-12-15 | 1 |

List all courses along with start and end date ?

The screenshot shows a database management interface with a dark theme. The top navigation bar includes 'Administration', 'Schemas', 'Users', and 'Courses'. The left sidebar, titled 'SCHEMAS', contains a tree view with 'books_table', 'fsd61wde', 'GuviZenClassDB' (expanded), 'Tables' (expanded), 'Codekata', 'Courses' (selected), 'Leaderboard', 'LeaveApplications', 'Modules', and 'Queries'. The main area displays a SQL query: `1 • SELECT course_name, start_date, end_date FROM Courses;`. Below the query editor, there are controls for 'Result Grid', 'Filter Rows', and 'Export'. The results are shown in a table with the following data:

| course_name | start_date | end_date |
|--------------------------------|------------|------------|
| Introduction to Programming | 2024-09-01 | 2024-12-15 |
| Data Structures and Algorithms | 2024-09-01 | 2024-12-15 |
| Web Development | 2024-09-01 | 2024-12-15 |

3.Modules:

```
```sql
create table modules (
 module_id int not null auto_increment,
 module_name varchar(100) not null,
 module_description text,
 course_id int not null,
 primary key (module_id),
 foreign key (course_id) references courses(course_id)
);

insert into modules (module_name, module_description, course_id)
values
('Introduction to Variables', 'Learning about variables in programming.', 1),
('Linked Lists', 'Understanding the basics of linked lists.', 2),
('HTML Basics', 'Introduction to HTML and its basic elements.', 3);
```
```

Output:

The screenshot shows a database management interface with a sidebar on the left displaying a tree view of schemas. The 'GuvizZenClassDB' schema is expanded, showing a list of tables including 'books_table', 'fsd61wde', 'Codekata', 'Courses', 'Leaderboard', 'LeaveApplications', 'Modules', 'Queries', and 'Tasks'. The 'Modules' table is selected. The main window displays the SQL query 'SELECT * FROM Modules;' and the resulting data in a table format. The table has four columns: 'module_id', 'module_name', 'module_description', and 'course_id'. The data is as follows:

| module_id | module_name | module_description | course_id |
|-----------|---------------------------|--|-----------|
| 1 | Introduction to Variables | Learning about variables in programming. | 1 |
| 2 | Linked Lists | Understanding the basics of linked lists. | 2 |
| 3 | HTML Basics | Introduction to HTML and its basic elements. | 3 |
| HULL | HULL | HULL | HULL |

Module along with description for specific course identify by the course_id:

The screenshot shows a database management interface with a dark theme. At the top, there are tabs for 'Administration', 'Schemas', 'Users', 'Courses', and 'Modules'. The 'Schemas' tab is active, showing a tree view on the left with 'GuviZenClassDB' expanded to show 'Tables'. The 'Modules' table is selected. The main area displays a SQL query:

```
1 • SELECT module_name, module_description
2   FROM modules WHERE course_id = 1;
3
```

Below the query editor, there is a 'Result Grid' section. It shows a table with two columns: 'module_name' and 'module_description'. The first row of data is:

| module_name | module_description |
|---------------------------|--|
| Introduction to Variables | Learning about variables in programming. |

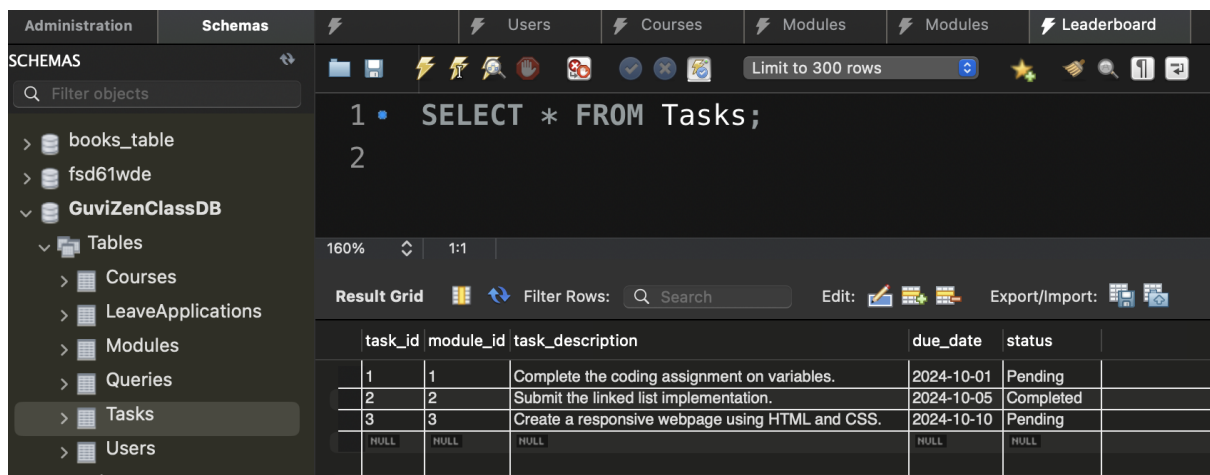
The interface also includes a search bar, a 'Filter Rows' button, and an 'Export' button. The query results are limited to 300 rows.

4.Tasks:

```
```sql
create table tasks (
 task_id int not null auto_increment,
 module_id int not null,
 task_description text,
 due_date date,
 status varchar(20),
 primary key (task_id),
 foreign key (module_id) references modules(module_id)
);

insert into tasks (module_id, task_description, due_date, status) values
(1, 'complete the coding assignment on variables.', '2024-10-01', 'pending'),
(2, 'submit the linked list implementation.', '2024-10-05', 'completed'),
(3, 'create a responsive webpage using html and css.', '2024-10-10', 'pending');
```
```

Output:



The screenshot shows a database management interface with a sidebar on the left displaying a schema tree. The 'Tasks' table is selected. The main area shows the SQL query 'SELECT * FROM Tasks;' and the resulting data grid.

| task_id | module_id | task_description | due_date | status |
|---------|-----------|---|------------|-----------|
| 1 | 1 | Complete the coding assignment on variables. | 2024-10-01 | Pending |
| 2 | 2 | Submit the linked list implementation. | 2024-10-05 | Completed |
| 3 | 3 | Create a responsive webpage using HTML and CSS. | 2024-10-10 | Pending |
| | | | | |

List all tasks that are currently pending along with their due dates:

The screenshot shows a database management tool interface. The top navigation bar includes tabs for Administration, Schemas, Users, Courses, and Modules. The left sidebar, titled 'SCHEMAS', shows a tree view with 'GuviZenClassDB' expanded, containing 'Tables' (Courses, LeaveApplications, Modules, Queries, Tasks, Users) and 'Users'. The main area displays a SQL query:

```
1 • select task_description, due_date
2   from tasks where status = 'pending';
3
```

 Below the query editor, the 'Result Grid' shows two columns: 'task_description' and 'due_date'. The results are as follows:

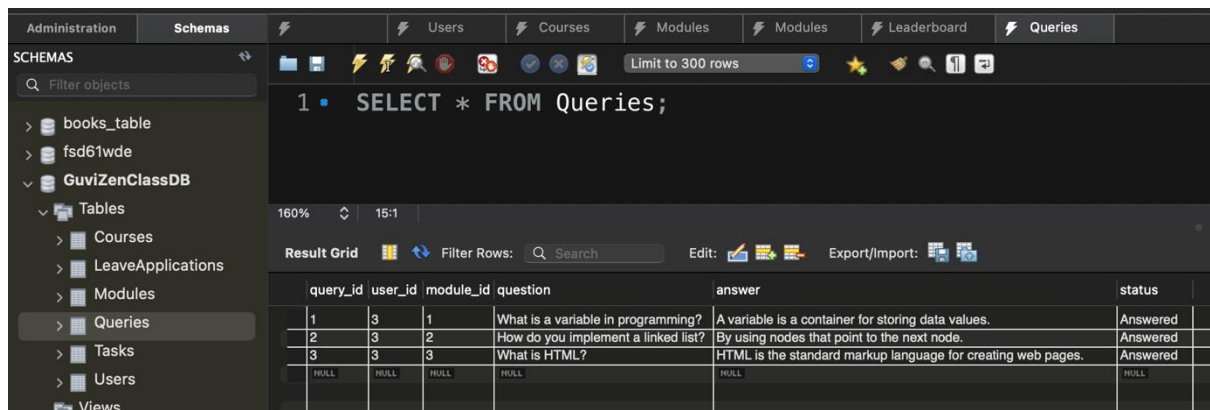
| task_description | due_date |
|---|------------|
| Complete the coding assignment on variables. | 2024-10-01 |
| Create a responsive webpage using HTML and CSS. | 2024-10-10 |

5.Queries:

```
```sql
create table queries (
 query_id int not null auto_increment,
 user_id int not null,
 module_id int not null,
 question text,
 answer text,
 status varchar(20),
 primary key (query_id),
 foreign key (user_id) references users(user_id),
 foreign key (module_id) references modules(module_id)
);

insert into queries (user_id, module_id, question, answer, status) values
(3, 1, 'what is a variable in programming?', 'a variable is a container for storing
data values.', 'answered'),
(3, 2, 'how do you implement a linked list?', 'by using nodes that point to the
next node.', 'answered'),
(3, 3, 'what is html?', 'html is the standard markup language for creating web
pages.', 'answered');
```
```

Output:



The screenshot shows a database management interface with a sidebar on the left containing a tree view of schemas and tables. The main area displays a SQL query: `1 • SELECT * FROM Queries;` and a 'Result Grid' below it. The grid shows the contents of the 'Queries' table, which has six columns: query_id, user_id, module_id, question, answer, and status. The data is as follows:

| query_id | user_id | module_id | question | answer | status |
|----------|---------|-----------|-------------------------------------|--|----------|
| 1 | 3 | 1 | What is a variable in programming? | A variable is a container for storing data values. | Answered |
| 2 | 3 | 2 | How do you implement a linked list? | By using nodes that point to the next node. | Answered |
| 3 | 3 | 3 | What is HTML? | HTML is the standard markup language for creating web pages. | Answered |
| | NULL | NULL | NULL | NULL | NULL |
| | | | | | |

Retrieve all the questions and their corresponding answers for a specific module:

The screenshot shows a database management interface with a dark theme. The top navigation bar includes tabs for Administration, Schemas, Users, Courses, Modules, and Leads. The left sidebar, titled 'SCHEMAS', contains a search bar and a tree view with the following structure:

- books_table
- fsd61wde
- GuviZenClassDB
 - Tables
 - Courses
 - LeaveApplications
 - Modules
 - Queries (highlighted)
 - Tasks

The main area displays a SQL query in a text editor:

```
1 • select question, answer from queries
2   where module_id = 1;
3
```

Below the query editor, there is a 'Result Grid' section. It includes a zoom level of 160%, a 25:1 aspect ratio, and a 'Filter Rows' search bar. The results are shown in a table with two columns: 'question' and 'answer'.

| question | answer |
|------------------------------------|--|
| What is a variable in programming? | A variable is a container for storing data values. |

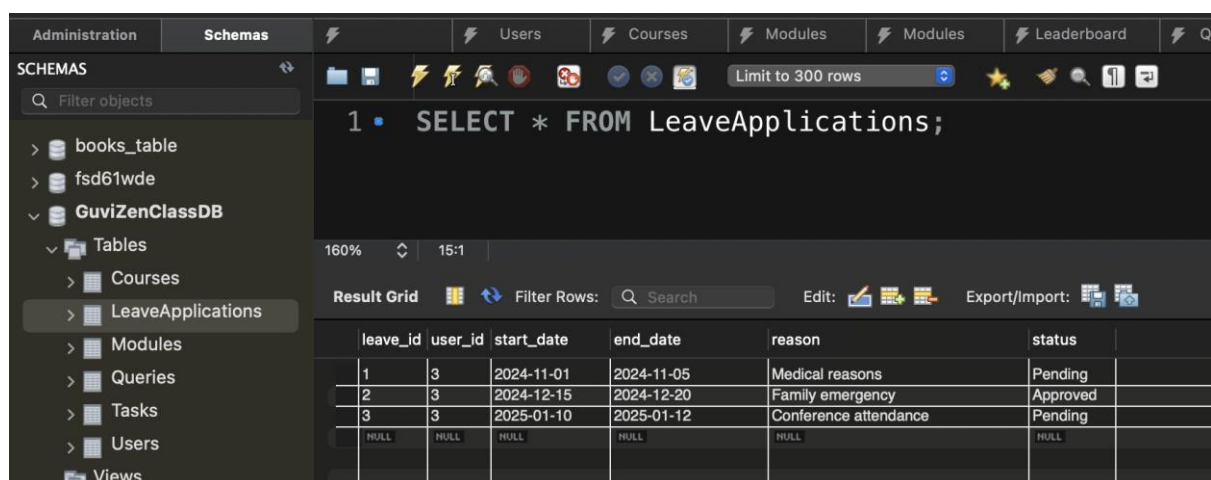
6.LeaveApplication:

```
```sql
create table leaveapplications (
 leave_id int not null auto_increment,
 user_id int not null,
 start_date date not null,
 end_date date not null,
 reason text,
 status varchar(20) default 'pending',
 primary key (leave_id),
 foreign key (user_id) references users(user_id)
);
```

```
insert into leaveapplications (user_id, start_date, end_date, reason, status)
values
(3, '2024-11-01', '2024-11-05', 'medical reasons', 'pending'),
(3, '2024-12-15', '2024-12-20', 'family emergency', 'approved'),
(3, '2025-01-10', '2025-01-12', 'conference attendance', 'pending');
``
```

### Output:

---



The screenshot shows a database management interface with a sidebar on the left containing a tree view of schemas and tables. The main area displays a SQL query: `SELECT * FROM LeaveApplications;` and a 'Result Grid' below it. The grid shows three rows of data for the LeaveApplications table, with columns for leave\_id, user\_id, start\_date, end\_date, reason, and status. The first row shows a pending leave for user 3 from 2024-11-01 to 2024-11-05 for medical reasons. The second row shows an approved leave for user 3 from 2024-12-15 to 2024-12-20 for family emergency. The third row shows a pending leave for user 3 from 2025-01-10 to 2025-01-12 for conference attendance. The grid also includes a 'Filter Rows' search bar and 'Edit' and 'Export/Import' buttons.

leave_id	user_id	start_date	end_date	reason	status
1	3	2024-11-01	2024-11-05	Medical reasons	Pending
2	3	2024-12-15	2024-12-20	Family emergency	Approved
3	3	2025-01-10	2025-01-12	Conference attendance	Pending

*List all leave applications that have been approved:*

---

The screenshot shows a database management interface with a dark theme. On the left, a 'SCHEMAS' sidebar lists various database objects, with 'LeaveApplications' selected under the 'GuviZenClassDB' schema. The main area displays a SQL query in a text editor:

```
1 • select leave_id, start_date, end_date, reason
2 from leaveapplications
3 where status = 'approved';
4
```

Below the query editor, a 'Result Grid' shows the query results. The grid has five columns: 'leave\_id', 'start\_date', 'end\_date', 'reason', and an empty column. The first row of data shows '2', '2024-12-15', '2024-12-20', and 'Family emergency'. Subsequent rows are empty, with 'NULL' values visible in the first three columns of the second row.

leave_id	start_date	end_date	reason	
2	2024-12-15	2024-12-20	Family emergency	
NULL	NULL	NULL	NULL	

## Join Clause:

List all approved leave applications along with the full name of the user who applied for the leave?

---

The screenshot shows a database management interface with a sidebar on the left containing a tree view of schemas. The 'GuviZenClassDB' schema is selected, and the 'Users' table is highlighted. The main area displays a SQL query that joins the 'leaveapplications' table with the 'users' table to retrieve approved leave applications. Below the query editor, the 'Result Grid' shows the output of the query, displaying columns for first\_name, last\_name, start\_date, and end\_date. The first row of data shows Charlie Brown with a leave application starting on 2024-12-15 and ending on 2024-12-20.

```
1 • use GuviZenClassDB;
2
3 • select
4 u.first_name,
5 u.last_name,
6 l.start_date,
7 l.end_date
8 from
9 leaveapplications l
10 join
11 users u
12 on
13 l.user_id = u.user_id
14 where
15 l.status = 'approved';
16
```

Table: **Users**

Columns:

- user\_id int AI PK
- first\_name varchar(50)
- last\_name varchar(50)
- email

Result Grid

first_name	last_name	start_date	end_date
Charlie	Brown	2024-12-15	2024-12-20

List all modules for each course, showing the course name and module name ?

---

The screenshot shows a database management interface with a sidebar on the left and a main workspace on the right. The sidebar, under the 'Schemas' tab, shows a tree view of the database structure. The 'GuviZenClassDB' schema is expanded, showing tables, views, stored procedures, and functions. The 'Users' table is selected. The main workspace displays a SQL query in a dark-themed editor. The query is a SELECT statement that joins the 'courses' and 'modules' tables on 'course\_id'. Below the query editor, there is a 'Result Grid' section showing the results of the query. The results are displayed in a table with two columns: 'course\_name' and 'module\_name'. The table contains three rows of data. The interface also includes a top navigation bar with tabs for 'Administration', 'Schemas', 'Users', 'Courses', 'Modules', 'Leaderboard', and 'Quizzes'. A search bar and a 'Limit to 300 rows' dropdown are also visible in the top right of the workspace.

```
1 use GuviZenClassDB;
2
3 select
4 c.course_name,
5 m.module_name
6 from
7 courses c
8 join
9 modules m
10 on
11 c.course_id = m.course_id;
12
```

130% 1:3

Result Grid Filter Rows: Search Export:

course_name	module_name
Introduction to Programming	Introduction to Variables
Data Structures and Algorithms	Linked Lists
Web Development	HTML Basics

Object Info Session

Table: Users

Columns: user\_id