

# library-management-system

---

A frontend application for managing the resources and memberships of a university library stored in MySQL Server.

Optimization Tasks of Assignment 8 is at the end.

## Steps to run this project

---

Installation requirements :

```
python3 -m venv env_flask  
pip3 install flask  
pip3 install flask-mysqldb  
pip3 install pyyaml
```



MySQL Server (Workbench) should also be installed. Dump the library schema with dummy values using the following command in MySQL:

```
source dumpfile.sql;
```

One must configure the db.yaml file to connect to the MySQL Server.

```
mysql_host: 'localhost'  
mysql_user: 'root'  
mysql_password: '<enter your password>'  
mysql_db: '<enter the name of the database which can be imported from the dump file>'
```

After cloning the repository and entering into the `library-management-system` folder :

```
cd flask-website  
python app.py
```

Click on the URL `http://127.0.0.1:5000` to visit the Library Management System website.

## Contents of this repository

---

- The relations were added to the database from the MySQL client command line. The schema with dummy values of our database can be found in the `final_values_final.sql` file.
- A Python script was used to add dummy data to our tables. This script can be found in the `insert_dummy_values.py` file.
- The Flask website code is found inside the `flask-website` folder.

## Snapshots of the functioning of the website

---

The functioning of the INSERT, DELETE and UPDATE queries has been shown through the following snapshots of the website. We have also shown some cases where our web app gives errors like foreign key constraints are violated, duplicate primary keys, etc.

### View Of `Users` Relation

## Library Management System

[Users](#) [Users\\_Phone](#) [Student](#) [Faculty](#) [Staff](#) [Library\\_Staff](#) [Other\\_Staff](#) [Library\\_Systems](#) [Books](#) [Books\\_Purchase](#) [Publishers](#) [Publishers\\_Phone](#)  
[Authors](#) [Genres](#) [Location](#) [Transaction](#) [Penalties](#) [Purchase](#) [Book\\_Issue](#) [User\\_Issue](#) [Strike](#) [Book\\_Pub](#) [Book\\_Auth](#) [Book\\_Genre](#) [Book\\_Location](#)  
[System\\_Allocation](#) [Library\\_Staff\\_Working\\_Hours](#)

Show  entries

Search:

user_id	user_name	Delete
101	Kim	X
102	Bob	X
103	Slya	X
123	Snape	X
200	Rachel	X
301	Sam	X
302	Sam	X
401	Gita	X
561	Monica	X
601	Harry	X
602	Ginny	X
603	Hermoine	X

602	Ginny	X
603	Hermoine	X
604	Ron	X
605	Neville	X
721	Chandler	X
723	Lily	X
811	Ross	X
922	Phoebe	X

Showing 1 to 18 of 18 entries

Input Values to be Inserted

user_id	
user_name	
<input type="button" value="Add"/>	

## View Of Users Relation in Workbench

MySQLWorkbench File Edit View Query Database Server Tools Scripting Help 4% Wed 30 Mar 7:05:15 PM

Local instance 3306 MySQL Workbench

Administration Schemas Query 4 Users - Table Users Context Help Snippets

Result Grid Filter Rows: Search Edit: Export/Import:

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid Form Editor Field Types Query Stats Execution Plan

SCHEMAS Publishers Publishers\_Phone Purchase Staff Strike Student System\_Allocation Transaction User\_Issue Users

Columns Indexes Foreign Keys Triggers Views

Tables Views Standard Procedures

Object Info Session

Table: Users

Columns:

user_ID	int PK
User_name	varchar(45)

Users 1 Apply Revert

Action Output

Time	Action	Response	Duration / Fetch Time
19:04:46	SELECT * FROM assignment_6.Users LIMIT 0, 1000	18 row(s) returned	0.00047 sec / 0.0000...

Query Completed

## Adding Entry in Users Relation

301	Sam	X
302	Sam	X
401	Gita	X
561	Monica	X
601	Harry	X
602	Ginny	X
603	Hermoine	X
604	Ron	X
605	Neville	X
721	Chandler	X
723	Lily	X
811	Ross	X
922	Phoebe	X

Showing 1 to 18 of 18 entries

Previous 1 Next

Input Values to be Inserted

user_id	989
user_name	Stefan

Add

user_id	user_name	Delete
101	Kim	X
102	Bob	X
103	Siya	X
123	Snape	X
200	Rachel	X
301	Sam	X
302	Sam	X
401	Gita	X
561	Monica	X
601	Harry	X
602	Ginny	X
603	Hermoine	X
604	Ron	X
605	Neville	X
721	Chandler	X
723	Lily	X
811	Ross	X
922	Phoebe	X
989	Stefan	X

Showing 1 to 19 of 19 entries

Previous 1 Next

MySQLWorkbench File Edit View Query Database Server Tools Scripting Help

Local instance 3306 MySQL Workbench

Administration Schemas Query 4 Users - Table Users Context Help Snippets

Result Grid Filter Rows: Search Edit: Export/Import:

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid Form Editor Field Types Query Stats Execution Plan

SCHEMAS

Publishers Publishers\_Phone Purchase Staff Strike Student System\_Allocation Transaction User\_Issue Users

Columns Indexes Foreign Keys Triggers Users\_Phone Views

Table: Users

Object Info Session

Table: Users

Columns:

<b>user_ID</b>	int PK
User_name	varchar(45)

Action Output

	Time	Action	Response	Duration / Fetch Time
1	19:04:46	SELECT * FROM assignment_6.Users LIMIT 0, 1000	18 row(s) returned	0.00047 sec / 0.0000...
2	19:07:10	SELECT * FROM assignment_6.Users LIMIT 0, 1000	19 row(s) returned	0.00042 sec / 0.0000...

Users 1 Apply Revert

Query Completed

user_ID	User_name
101	Kim
102	Bob
103	Siyia
123	Snape
200	Rachel
301	Sam
302	Sam
401	Ginn
501	Monica
601	Hank
602	Ginny
603	Hermione
604	Ron
605	Neville
721	Chandler
723	Lily
811	Ross
922	Phoebe
989	Stefan
NULL	NULL

## Updating Entry in Users Relation

user_id	user_name	Delete
101	Kim	X
102	Bob	X
103	Siya	X
123	Snape	X
200	Rachel	X
301	Sam	X
302	Sam	X
401	Gita	X
561	Monica	X
601	Harry	X
602	Ginny	X
603	Hermoine	X
604	Ron	X
605	Neville	X
721	Chandler	X

update

Stefan

989 Stefan X

Showing 1 to 19 of 19 entries

Previous 1 Next

user_id	user_name	Delete
101	Kim	X
102	Bob	X
103	Siya	X
123	Snape	X
200	Rachel	X
301	Sam	X
302	Sam	X
401	Gita	X
561	Monica	X
601	Harry	X
602	Ginny	X
603	Hermoine	X
604	Ron	X
605	Neville	X
721	Chandler	X

update

Damon

989 Stefan X

Showing 1 to 19 of 19 entries

Previous 1 Next

user_id	user_name	Delete
101	Kim	X
102	Bob	X
103	Siya	X
123	Snape	X
200	Rachel	X
301	Sam	X
302	Sam	X
401	Gita	X
561	Monica	X
601	Harry	X
602	Ginny	X
603	Hermoine	X
604	Ron	X
605	Neville	X
721	Chandler	X
723	Lily	X
811	Ross	X
922	Phoebe	X
989	Damon	X

Showing 1 to 19 of 19 entries

Previous 1 Next

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The title bar indicates "MySQL Workbench" and the date "Wed 30 Mar 7:08:40 PM". The left sidebar displays the database schema with the "Users" table selected. The main area shows a "Result Grid" with the following data:

user_ID	User_name
101	Kim
102	Bob
103	Siya
123	Snape
200	Rachel
301	Sam
302	Sam
401	Gita
561	Monica
601	Harry
602	Ginny
603	Hermoine
604	Ron
605	Neville
721	Chandler
723	Lily
811	Ross
922	Phoebe
989	Damon
NULL	NULL

The right side of the interface includes various toolbars and a context help panel stating: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help." The bottom status bar shows "Query Completed".

## Deleting Entry in Users Relation

user_id	user_name	Delete
101	Kim	X
102	Bob	X
103	Siya	X
123	Snape	X
200	Rachel	X
301	Sam	X
302	Sam	X
401	Gita	X
561	Monica	X
601	Harry	X
602	Ginny	X
603	Hermoine	X
604	Ron	X
605	Neville	X
721	Chandler	X
723	Lily	X
811	Ross	X
922	Phoebe	X
989	Damon	X

Showing 1 to 19 of 19 entries

[Previous](#) [1](#) [Next](#)Show [All](#) entriesSearch: 

user_id	user_name	Delete
101	Kim	X
102	Bob	X
103	Siya	X
123	Snape	X
200	Rachel	X
301	Sam	X
302	Sam	X
401	Gita	X
561	Monica	X
601	Harry	X
602	Ginny	X
603	Hermoine	X
604	Ron	X
605	Neville	X
721	Chandler	X
723	Lily	X
811	Ross	X
922	Phoebe	X

Showing 1 to 18 of 18 entries

[Previous](#) [1](#) [Next](#)

## View Of Transactions Relation

transaction_ID	issue_date	expected_return_date	actual_return_date	Delete
4211	2021-07-15 00:00:00	2021-07-24 00:00:00	2021-07-20 00:00:00	X
4230	2021-07-14 00:00:00	2021-07-20 00:00:00	2021-07-22 00:00:00	X
4232	2021-07-12 00:00:00	2021-07-20 00:00:00	2021-07-23 00:00:00	X
4238	2021-07-12 00:00:00	2021-07-20 00:00:00	2021-07-24 00:00:00	X
4278	2021-03-10 00:00:00	2022-07-15 00:00:00	2022-07-20 00:00:00	X
4473	2021-07-11 00:00:00	2021-07-24 00:00:00	2021-07-20 00:00:00	X
4898	2022-03-03 00:00:00	2022-04-05 00:00:00	None	X

## Adding Entry in Transactions Relation

Show 10 entries Search:

transaction_ID	issue_date	expected_return_date	actual_return_date	Delete
4211	2021-07-15 00:00:00	2021-07-24 00:00:00	2021-07-20 00:00:00	X
4230	2021-07-14 00:00:00	2021-07-20 00:00:00	2021-07-22 00:00:00	X
4232	2021-07-12 00:00:00	2021-07-20 00:00:00	2021-07-23 00:00:00	X
4238	2021-07-12 00:00:00	2021-07-20 00:00:00	2021-07-24 00:00:00	X
4278	2021-03-10 00:00:00	2022-07-15 00:00:00	2022-07-20 00:00:00	X
4473	2021-07-11 00:00:00	2021-07-24 00:00:00	2021-07-20 00:00:00	X
4898	2022-03-03 00:00:00	2022-04-05 00:00:00	None	X

Showing 1 to 7 of 7 entries

Previous 1 Next

Input Values to be Inserted

transaction_ID	4980
issue_date	2022-03-30
expected_return_date	2022-04-05
actual_return_date	NULL

Show 10 entries Search:

Showing 1 to 8 of 8 entries

transaction_ID	issue_date	expected_return_date	actual_return_date	Delete
4211	2021-07-15 00:00:00	2021-07-24 00:00:00	2021-07-20 00:00:00	X
4230	2021-07-14 00:00:00	2021-07-20 00:00:00	2021-07-22 00:00:00	X
4232	2021-07-12 00:00:00	2021-07-20 00:00:00	2021-07-23 00:00:00	X
4238	2021-07-12 00:00:00	2021-07-20 00:00:00	2021-07-24 00:00:00	X
4278	2021-03-10 00:00:00	2022-07-15 00:00:00	2022-07-20 00:00:00	X
4473	2021-07-11 00:00:00	2021-07-24 00:00:00	2021-07-20 00:00:00	X
4898	2022-03-03 00:00:00	2022-04-05 00:00:00	None	X
4980	2022-03-30 00:00:00	2022-04-05 00:00:00	None	X

Showing 1 to 8 of 8 entries

Previous 1 Next

Input Values to be Inserted

transaction_ID	
issue_date	
expected_return_date	
actual_return_date	

MySQL Workbench File Edit View Query Database Server Tools Scripting Help

Local instance 3306 MySQL Workbench

Administration Schemas Query 4 Transaction Context Help Snippets

**Result Grid** Filter Rows: Search Export/Import:

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

**SCHEMAS**

- > Publishers
- > Publishers\_Photo
- > Purchase
- > Staff
- > Strike
- > Student
- > System\_Allocation
- > Transaction
- > User\_Issue
- Users
  - > Columns
  - > Indexes
  - > Foreign Keys
  - > Triggers
  - > Users\_Photo
- Views

**Object Info Session**

**Table: Transaction**

**Columns:**

transaction_ID	int PK
issue_date	datetime
expected_return_date	datetime
actual_return_date	datetime

Transaction 1

Action Output

Time	Action	Response	Duration / Fetch Time
4	SELECT * FROM assignment_6.Transaction LIMIT 0, 1000	8 row(s) returned	0.00000 sec / 0.000...
5	19:34:17 SELECT * FROM assignment_6.Transaction LIMIT 0, 1000	8 row(s) returned	0.00048 sec / 0.000...

Query Completed

## Adding Entry in User\_Issue Relation

## Library Management System

[Users](#) [Users\\_Phone](#) [Student](#) [Faculty](#) [Staff](#) [Library\\_Staff](#) [Other\\_Staff](#) [Library\\_Systems](#) [Books](#) [Books\\_Purchase](#) [Publishers](#) [Publishers\\_Phone](#)  
[Authors](#) [Genres](#) [Location](#) [Transaction](#) [Penalties](#) [Purchase](#) [Book\\_Issue](#) [User\\_Issue](#) [Strike](#) [Book\\_Pub](#) [Book\\_Auth](#) [Book\\_Genre](#) [Book\\_Location](#)  
[System\\_Allocation](#) [Library\\_Staff\\_Working\\_Hours](#)

Show 10 entries

Search:

transaction_ID	user_ID	Delete
4238	101	X
4278	103	X
4898	200	X

Showing 1 to 3 of 3 entries

[Previous](#) [1](#) [Next](#)

**Input Values to be Inserted**

transaction_ID	<input type="text" value="4980"/>
user_ID	<input type="text" value="922"/>
<a href="#" style="border: 1px solid black; padding: 2px 10px; background-color: #e0f2e0;">Add</a>	

## Library Management System

[Users](#) [Users\\_Phone](#) [Student](#) [Faculty](#) [Staff](#) [Library\\_Staff](#) [Other\\_Staff](#) [Library\\_Systems](#) [Books](#) [Books\\_Purchase](#) [Publishers](#) [Publishers\\_Phone](#)  
[Authors](#) [Genres](#) [Location](#) [Transaction](#) [Penalties](#) [Purchase](#) [Book\\_Issue](#) [User\\_Issue](#) [Strike](#) [Book\\_Pub](#) [Book\\_Auth](#) [Book\\_Genre](#) [Book\\_Location](#)  
[System\\_Allocation](#) [Library\\_Staff\\_Working\\_Hours](#)

Show 10 entries

Search:

transaction_ID	user_ID	Delete
4238	101	X
4278	103	X
4898	200	X
4980	922	X

Showing 1 to 4 of 4 entries

[Previous](#) [1](#) [Next](#)

**Input Values to be Inserted**

transaction_ID	<input type="text"/>
user_ID	<input type="text"/>
<a href="#" style="border: 1px solid black; padding: 2px 10px; background-color: #e0f2e0;">Add</a>	

MySQLWorkbench File Edit View Query Database Server Tools Scripting Help

Local instance 3306 MySQL Workbench

Administration Schemas Query 4 User\_Issue Context Help Snippets

Result Grid Filter Rows: Search Edit: Export/Import:

transaction\_ID user\_ID

> 4238	101
> 4278	103
> 4898	200
> 4980	922
NULL	NULL

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Form Editor

Field Types

Query Stats

Execution Plan

Object Info Session

Table: User\_Issue

Columns:

transaction_ID	int PK
user_ID	int

User\_Issue 1

Action Output

Time	Action	Response	Duration / Fetch Time
6 19:34:40	SELECT * FROM assignment_6.User_Issue LIMIT 0, 1000	3 row(s) returned	0.00029 sec / 0.0000...

Apply Revert

Query Completed

The screenshot shows the MySQL Workbench interface. The left sidebar displays the schema structure with the User\_Issue table selected. The main area shows the table's columns (transaction\_ID, user\_ID) and a result grid with four rows of data. A sidebar on the right provides context help and links to other tools like Form Editor and Field Types. The bottom section shows the execution plan and action output for a SELECT query.

## Adding Entry in Book\_Issue Relation

## Library Management System

[Users](#) [Users\\_Phone](#) [Student](#) [Faculty](#) [Staff](#) [Library\\_Staff](#) [Other\\_Staff](#) [Library\\_Systems](#) [Books](#) [Books\\_Purchase](#) [Publishers](#) [Publishers\\_Phone](#)  
[Authors](#) [Genres](#) [Location](#) [Transaction](#) [Penalties](#) [Purchase](#) [Book\\_Issue](#) [User\\_Issue](#) [Strike](#) [Book\\_Pub](#) [Book\\_Auth](#) [Book\\_Genre](#) [Book\\_Location](#)  
[System\\_Allocation](#) [Library\\_Staff\\_Working\\_Hours](#)

Show 10 entries

Search:

transaction_ID	book_ID	Delete
4232	45678	X
4238	12345	X
4278	34567	X
4898	23456	X

Showing 1 to 4 of 4 entries

Previous 1 Next

**Input Values to be Inserted**

transaction_ID	<input type="text" value="4980"/>
book_ID	<input type="text" value="67890"/>
<input type="button" value="Add"/>	

## Library Management System

[Users](#) [Users\\_Phone](#) [Student](#) [Faculty](#) [Staff](#) [Library\\_Staff](#) [Other\\_Staff](#) [Library\\_Systems](#) [Books](#) [Books\\_Purchase](#) [Publishers](#) [Publishers\\_Phone](#)  
[Authors](#) [Genres](#) [Location](#) [Transaction](#) [Penalties](#) [Purchase](#) [Book\\_Issue](#) [User\\_Issue](#) [Strike](#) [Book\\_Pub](#) [Book\\_Auth](#) [Book\\_Genre](#) [Book\\_Location](#)  
[System\\_Allocation](#) [Library\\_Staff\\_Working\\_Hours](#)

Show 10 entries

Search:

transaction_ID	book_ID	Delete
4232	45678	X
4238	12345	X
4278	34567	X
4898	23456	X
4980	67890	X

Showing 1 to 5 of 5 entries

Previous 1 Next

**Input Values to be Inserted**

transaction_ID	<input type="text"/>
book_ID	<input type="text"/>
<input type="button" value="Add"/>	

The screenshot shows the MySQL Workbench interface with the following details:

- Toolbar:** Standard MySQL Workbench icons for file operations, database management, and search.
- Left Panel (schemas):** Shows the current schema structure. Under the **Book\_Issue** table, the **Columns** node is selected, displaying the table's columns: **transaction\_ID** (int PK) and **book\_ID** (int).
- Result Grid:** A table titled "transaction...\_book\_ID" showing five rows of data:

	transaction_ID	book_ID
1	4238	12345
2	4898	23456
3	4278	34567
4	4232	45678
5	4980	67890
- Right Panel (context help):** A vertical panel with links to various tools: Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan. A note at the top states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."
- Bottom Panel (query history):** Shows the executed query: "SELECT \* FROM assignment\_6.Book\_Issue LIMIT 0, 1000". The output indicates 5 row(s) returned and a duration of 0.00039 sec / 0.000... ms.

## Updating Entry in Library\_Systems Relation

## Library Management System

[Users](#) [Users.Phone](#) [Student](#) [Faculty](#) [Staff](#) [Library\\_Staff](#) [Other\\_Staff](#) [Library\\_Systems](#) [Books](#) [Books\\_Purchase](#) [Publishers](#) [Publishers.Phone](#)  
[Authors](#) [Genres](#) [Location](#) [Transaction](#) [Penalties](#) [Purchase](#) [Book\\_Issue](#) [User\\_Issue](#) [Strike](#) [Book\\_Pub](#) [Book\\_Auth](#) [Book\\_Genre](#) [Book\\_Location](#)  
[System\\_Allocation](#) [Library\\_Staff\\_Working\\_Hours](#)

Show  entries

Search:

system_ID	system_specs	Delete
1011	OS: Windows, NVIDIA GPU	X
1023	'IDIA GPU 256 GB RAM	X
1024		X
1035	OS: Windows, NVIDIA GPU	X

Showing 1 to 4 of 4 entries

Previous  Next

Input Values to be Inserted

system_ID	<input type="text"/>
system_specs	<input type="text"/>
<input type="button" value="Add"/>	

## Library Management System

[Users](#) [Users.Phone](#) [Student](#) [Faculty](#) [Staff](#) [Library\\_Staff](#) [Other\\_Staff](#) [Library\\_Systems](#) [Books](#) [Books\\_Purchase](#) [Publishers](#) [Publishers.Phone](#)  
[Authors](#) [Genres](#) [Location](#) [Transaction](#) [Penalties](#) [Purchase](#) [Book\\_Issue](#) [User\\_Issue](#) [Strike](#) [Book\\_Pub](#) [Book\\_Auth](#) [Book\\_Genre](#) [Book\\_Location](#)  
[System\\_Allocation](#) [Library\\_Staff\\_Working\\_Hours](#)

Show  entries

Search:

system_ID	system_specs	Delete
1011	OS: Windows, NVIDIA GPU	X
1023	OS: Windows, NO GPU	X
1024	OS: MacOS, TPU	X
1035	OS: Windows, NVIDIA GPU 256 GB RAM	X

Showing 1 to 4 of 4 entries

Previous  Next

Input Values to be Inserted

system_ID	<input type="text"/>
system_specs	<input type="text"/>
<input type="button" value="Add"/>	

MySQLWorkbench File Edit View Query Database Server Tools Scripting Help

Local instance 3306 MySQL Workbench

Administration Schemas Query 4 Library\_Systems

Result Grid Filter Rows: Search Edit: Export/Import:

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

System\_ID system\_specs

1011	OS: Windows, NVIDIA GPU
1023	OS: Windows, NO GPU
1024	OS: MacOS, TPU
1035	OS: Windows, NVIDIA GPU 256 GB RAM
HULL	HULL

Object Info Session

Table: Library\_Systems

Columns:

system_ID	int PK
system_specs	text

Action Output

Time	Action	Response	Duration / Fetch Time
19:37:45	SELECT * FROM assignment_6.Library_Systems LIMIT 0, 1000	4 row(s) returned	0.00039 sec / 0.000...
8	19:37:45	SELECT * FROM assignment_6.Library_Systems LIMIT 0, 1000	0.00039 sec / 0.000...

Query Completed

The screenshot shows the MySQL Workbench interface with the 'Library\_Systems' table selected. The results grid displays the following data:

System ID	System Specs
1011	OS: Windows, NVIDIA GPU
1023	OS: Windows, NO GPU
1024	OS: MacOS, TPU
1035	OS: Windows, NVIDIA GPU 256 GB RAM

## Deleting Entry in Library\_Systems Relation

## Library Management System

[Users](#) [Users\\_Phone](#) [Student](#) [Faculty](#) [Staff](#) [Library\\_Staff](#) [Other\\_Staff](#) [Library\\_Systems](#) [Books](#) [Books\\_Purchase](#) [Publishers](#) [Publishers\\_Phone](#)  
[Authors](#) [Genres](#) [Location](#) [Transaction](#) [Penalties](#) [Purchase](#) [Book\\_Issue](#) [User\\_Issue](#) [Strike](#) [Book\\_Pub](#) [Book\\_Auth](#) [Book\\_Genre](#) [Book\\_Location](#)  
**System\_Allocation** [Library\\_Staff\\_Working\\_Hours](#)

Show  entries

Search:

user_ID	system_ID	Delete
123	1035	X
561	1011	X
603	1023	X
604	1024	X

Showing 1 to 4 of 4 entries

[Previous](#) [1](#) [Next](#)

### Input Values to be Inserted

user\_ID   
 system\_ID

[Add](#)

## Library Management System

[Users](#) [Users\\_Phone](#) [Student](#) [Faculty](#) [Staff](#) [Library\\_Staff](#) [Other\\_Staff](#) [Library\\_Systems](#) [Books](#) [Books\\_Purchase](#) [Publishers](#) [Publishers\\_Phone](#)  
[Authors](#) [Genres](#) [Location](#) [Transaction](#) [Penalties](#) [Purchase](#) [Book\\_Issue](#) [User\\_Issue](#) [Strike](#) [Book\\_Pub](#) [Book\\_Auth](#) [Book\\_Genre](#) [Book\\_Location](#)  
**System\_Allocation** [Library\\_Staff\\_Working\\_Hours](#)

Show  entries

Search:

user_ID	system_ID	Delete
123	1035	X
561	1011	X
604	1024	X

Showing 1 to 3 of 3 entries

[Previous](#) [1](#) [Next](#)

### Input Values to be Inserted

user\_ID   
 system\_ID

[Add](#)

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the tabs include Administration, Schemas, Query 4, System\_Allocation, and Context Help. The main area displays a Result Grid for the 'System\_Allocation' table, which has columns 'user\_ID' and 'system\_ID'. The data shows three rows: (561, 1011), (604, 1024), and (123, 1035). To the right of the grid is a sidebar with various tools: Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan. A note in the sidebar states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help." Below the grid, the Object Info tab is selected, showing details for the 'Library.Systems' table, including its columns: system\_ID (int PK) and system\_specs (text). The Action Output section shows a single query execution: "SELECT \* FROM assignment\_6.System\_Allocation LIMIT 0, 1000" with a duration of 0.00029 sec / 0.0000... rows returned. At the bottom left, it says "Query Completed".

## Error due to Foreign Key Constraint in Users Relation

Showing 1 to 18 of 18 entries		
<a href="#">Show All</a> <a href="#">entries</a> <input type="text" value="Search:"/> <a href="#">Next</a>		
user_id	user_name	Delete
101	Kim	X
102	Bob	X
103	Siya	X
123	Snape	X
200	Rachel	X
301	Sam	X
302	Sam	X
401	Gita	X
561	Monica	X
601	Harry	X
602	Ginny	X
603	Hermoine	X
604	Ron	X
605	Neville	X
721	Chandler	X
723	Lily	X
811	Ross	X
922	Phoebe	X

The screenshot shows a Chrome browser window with the URL "127.0.0.1:5000/delete/Users?Users\_user\_id=101&Users\_user\_name=Kim&". The page displays an error message: "There was an issue adding the entry:(1451, 'Cannot delete or update a parent row: a foreign key constraint fails ('assignment\_6`.`user\_issue`, CONSTRAINT `user\_ID\_user` FOREIGN KEY (`user\_ID`) REFERENCES `users` (`user\_ID`))')".

## Error due to Duplicate Primary Key in Books Relation

Authors Genres Location Transaction Penalties Purchase Book\_Issue User\_Issue Strike Book\_Pub Book\_Auth Book\_Genre Book\_Location  
 System\_Allocation Library\_Staff\_Working\_Hours

Show 10 entries

Search:

book_ID	book_name	num_pages	availability	Delete
12345	Science Marvels	560	True	X
23456	Archaeology	245	False	X
34567	Astro Physics	500	True	X
45670	Compilers	679	True	X
45678	Panchatantra	200	True	X
56789	Computer Organisation and Architecture	1200	True	X
67890	Sherlock Holmes	820	True	X

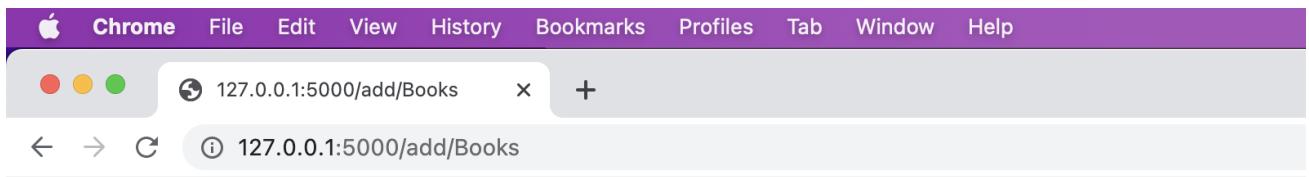
Showing 1 to 7 of 7 entries

Previous 1 Next

Input Values to be Inserted

book_ID	67890
book_name	Databases
num_pages	623
availability	True

Add



## Tasks Assignment 8

### Task 1

#### Query 1

```
select * from publishers where publisher_name like 'Denise%' or street_name
like '%Scott';
```

#### Optimized query 1

```
select * from publishers where publisher_name like 'Denise%'
union all
select * from publishers where street_name like '%Scott';
```

```

mysql> set profiling = 1;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> select * from publishers where publisher_name like 'Denise%' or street_name like '%Scott';
+-----+-----+-----+-----+-----+-----+-----+
| publisher_ID | publisher_name | street_num | street_name | city | state | zip_code | email |
+-----+-----+-----+-----+-----+-----+-----+
| 10492 | Denise Hammond | 123 | Barbara Croft | abc | abc | 302363 | abc_612 |
| 10888 | William Fundora | 123 | Heather Scott | abc | abc | 935344 | abc_677 |
| 58182 | Denise Beaver | 123 | Connie Downs | abc | abc | 404124 | abc_135 |
| 62834 | Allen Russell | 123 | Janet Scott | abc | abc | 171932 | abc_587 |
| 95239 | Sandra Nova | 123 | Michael Scott | abc | abc | 788562 | abc_545 |
| 96184 | Pamela Bolton | 123 | John Prescott | abc | abc | 810302 | abc_856 |
+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.08 sec)

mysql> set profiling = 0;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> explain select * from publishers where publisher_name like 'Denise%' or street_name like '%Scott';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | publishers | NULL | ALL | NULL | NULL | NULL | NULL | 1000 | 20.99 | Using where |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)

mysql> CREATE FULLTEXT INDEX pub_name_index ON publishers(publisher_name);
Query OK, 0 rows affected (0.75 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> CREATE FULLTEXT INDEX street_name_index ON publishers(street_name);
Query OK, 0 rows affected (0.50 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> set profiling = 1;
Query OK, 0 rows affected, 1 warning (0.00 sec)

```

```

MySQL 8.0 Command Line Client

mysql> select * from publishers where match(publisher_name) against ('Denise*' in boolean mode) union all select * from publishers where match(street_name) aga
inst ('*Scott' in boolean mode);
+-----+-----+-----+-----+-----+-----+-----+
| publisher_ID | publisher_name | street_num | street_name | city | state | zip_code | email |
+-----+-----+-----+-----+-----+-----+-----+
| 10492 | Denise Hammond | 123 | Barbara Croft | abc | abc | 302363 | abc_612 |
| 58182 | Denise Beaver | 123 | Connie Downs | abc | abc | 404124 | abc_135 |
| 10888 | William Fundora | 123 | Heather Scott | abc | abc | 935344 | abc_677 |
| 62834 | Allen Russell | 123 | Janet Scott | abc | abc | 171932 | abc_587 |
| 81960 | Maria Torres | 123 | Scott Robledo | abc | abc | 929974 | abc_726 |
| 95239 | Sandra Nova | 123 | Michael Scott | abc | abc | 788562 | abc_545 |
+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> set profiling = 0;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> explain select * from publishers where match(publisher_name) against ('Denise*' in boolean mode) union all select * from publishers where match(street_
name) against ('*Scott' in boolean mode);
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | publishers | NULL | fulltext | pub_name_index | pub_name_index | 0 | const | 1 | 100.00 | Using where; Ft_hints: n
o_ranking |
| 2 | UNION | publishers | NULL | fulltext | street_name_index | street_name_index | 0 | const | 1 | 100.00 | Using where; Ft_hints: n
o_ranking |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql> show profiles;
+-----+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+-----+
| 1 | 0.08033325 | select * from publishers where publisher_name like 'Denise%' or street_name like '%Scott' |
| 2 | 0.00825325 | select * from publishers where match(publisher_name) against ('Denise*' in boolean mode) union all select * from publishers where m
atch(street_name) against ('*Scott' in boolean mode) |
+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql>
```

## Query 1

Rows: 1000

Duration: 0.08033325 s

Optimized query 1

Rows: 2 (reduced)

Duration: 0.00825325 s (reduced)

If we run queries using the comparison operator 'or' on different columns in the where clause, there are chances that the MySQL optimizer may incorrectly choose a full table scan to retrieve a record. However, having different indices to optimize two separate queries on the two different attributes and taking the union of those results can make the query run faster. The query 1 above can run far much slower compared to optimized query 1 which uses a union operator to merge the results of 2 separate fast queries that take advantage of the indexes.

## Task 2

---

Query: `SELECT * FROM users WHERE user_name LIKE "m%" ;`

[pattern selected: x% , where x is any character from english alphabet]

Number of rows hit: 2000

Execution time: 0.00189500 sec

In the query mentioned above, we are querying a name that starts with the English alphabet 'm'. The query is expected to return the rows from the `users` table having the value of attribute `user_name` as a string starting with 'm'. Clearly, this can be modeled as a prefix selection problem. In our query, the prefix required is 'm'. Therefore, we can create a prefix index on the attribute `user_name` using:

```
CREATE INDEX user_name_ind ON users(user_name(1));
```

Now, we run the query again for the same pattern. To force the use of index, we run the optimized query as:

```
SELECT * FROM users USE INDEX(user_name_ind) WHERE user_name LIKE "m%" ;
```

Number of rows hit: 221

Execution time: 0.00142900 sec

Results:

```
[mysql> EXPLAIN SELECT * from users where user_name LIKE "m%";  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 1 | SIMPLE | users | NULL | ALL | NULL | NULL | NULL | NULL | 2000 | 11.11 | Using where |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
1 row in set, 1 warning (0.01 sec)  
  
[mysql> CREATE INDEX user_name_ind ON users(user_name(1));  
Query OK, 0 rows affected (0.08 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
| 1 | SIMPLE | users | NULL | range | user_name_ind | user_name_ind | 6 | NULL | 221 | 100.00 | Using where |  
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+  
1 row in set, 1 warning (0.00 sec)  
  
mysql> show profiles;  
+-----+-----+-----+  
| Query_ID | Duration | Query |  
+-----+-----+-----+  
| 1 | 0.00189500 | SELECT * FROM users WHERE user_name LIKE 'm%' |  
| 2 | 0.03234700 | CREATE INDEX user_name_ind ON users(user_name(1)) |  
| 3 | 0.00142900 | SELECT * FROM users USE INDEX(user_name_ind) WHERE user_name LIKE 'm%' |  
+-----+-----+-----+
```

## Task 3

Table: Publishers

Column:

- **zip\_code:** Change to BIGINT

The longest postal code used around the world is 10 digits long. So, space required to store zip\_code will be 8 bytes. INT type comparisons are faster than VARCHAR as the former takes up less space than varchars. If we store zip\_code as varchar(10) space required will be 11 bytes per entry.

Results:

```
[mysql]> describe publishers;
+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| publisher_ID | int        | NO   | PRI | NULL    |       |
| publisher_name | varchar(225) | NO   |     | NULL    |       |
| street_num    | varchar(45)  | YES  |     | NULL    |       |
| street_name   | varchar(45)  | YES  |     | NULL    |       |
| city          | varchar(45)  | YES  |     | NULL    |       |
| state         | varchar(45)  | YES  |     | NULL    |       |
| zip_code      | varchar(10)   | YES  |     | NULL    |       |
| email         | varchar(45)  | YES  | UNI | NULL    |       |
+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

```
[mysql]> set profiling = 1;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

```
[mysql]> select count(*) from publishers where zip_code > 980000;
+-----+
| count(*) |
+-----+
|      22 |
+-----+
1 row in set (0.01 sec)
```

```
[mysql]> set profiling = 0;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

```
[mysql]> alter table publishers modify column zip_code bigint;
Query OK, 1000 rows affected (0.03 sec)
Records: 1000  Duplicates: 0  Warnings: 0
```

```
[mysql]> describe publishers;
+-----+-----+-----+-----+-----+
| Field          | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| publisher_ID   | int        | NO   | PRI | NULL    |       |
| publisher_name | varchar(225) | NO   |     | NULL    |       |
| street_num     | varchar(45)  | YES  |     | NULL    |       |
| street_name    | varchar(45)  | YES  |     | NULL    |       |
| city           | varchar(45)  | YES  |     | NULL    |       |
| state          | varchar(45)  | YES  |     | NULL    |       |
| zip_code       | bigint      | YES  |     | NULL    |       |
| email          | varchar(45)  | YES  | UNI | NULL    |       |
+-----+-----+-----+-----+-----+
8 rows in set (0.01 sec)
```

```
[mysql]> set profiling = 1;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

```
[mysql]> select count(*) from publishers where zip_code > 980000;
+-----+
| count(*) |
+-----+
|      22 |
+-----+
1 row in set (0.01 sec)
```

```
[mysql]> set profiling = 0;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

```
[mysql]> SHOW PROFILES;
+-----+-----+
| Query_ID | Duration | Query
+-----+-----+
|      1 | 0.00160600 | select count(*) from publishers where zip_code > 980000 |
|      2 | 0.00149400 | select count(*) from publishers where zip_code > 980000 |
+-----+-----+
2 rows in set, 1 warning (0.00 sec)
```

Similarly, we can only change the following column types:

Table: Publishers\_Phone

Column:

- **phone\_number**: Change to BIGINT

Phone numbers are 10 digits long. So, space required to store phone\_numbers will be 8 bytes using BIGINT. INT type comparisons are faster than VARCHAR as the former takes up less space than varchars. If we store phone\_number as varchar(10) space required will be 11 bytes per entry.

#### Table: Faculty

Column:

- **salary**: No salary will be greater than 1,67,77,215. So we can keep the salary field as MEDIUMINT.

#### Table: Book\_Location

Column:

- **shelf\_ID**: Change to TINYINT

At most there will be 100 shelves in the library.

## Task 4

---

Query 1: `SELECT * FROM transaction WHERE issue_date = "2022-09-07"`

This task was implemented on the Transaction table. Initially, the issue\_date column had the data type as `VARCHAR(20)`.

The time taken by the query as given by the profiling functionality in MySQL is:

`0.00618500`

```
[mysql]>
[mysql]> DESCRIBE transaction;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| transaction_ID | int | NO | PRI | NULL |
| issue_date | varchar(20) | NO | | NULL |
| expected_return_date | varchar(20) | NO | | NULL |
| actual_return_date | varchar(20) | YES | | NULL |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

[mysql]>
[mysql]> SELECT * FROM transaction WHERE issue_date = "2022-09-07";
+-----+-----+-----+-----+
| transaction_ID | issue_date | expected_return_date | actual_return_date |
+-----+-----+-----+-----+
| 1021 | 2022-09-07 | 2022-09-14 | 2022-09-09 |
| 2575 | 2022-09-07 | 2022-09-14 | 2022-09-12 |
| 2599 | 2022-09-07 | 2022-09-14 | 2022-09-15 |
| 2698 | 2022-09-07 | 2022-09-14 | 2022-09-15 |
| 3064 | 2022-09-07 | 2022-09-14 | 2022-09-15 |
| 3274 | 2022-09-07 | 2022-09-14 | 2022-09-16 |
| 3280 | 2022-09-07 | 2022-09-14 | 2022-09-09 |
| 3304 | 2022-09-07 | 2022-09-14 | 2022-09-14 |
| 3397 | 2022-09-07 | 2022-09-14 | 2022-09-09 |
| 3739 | 2022-09-07 | 2022-09-14 | 2022-09-14 |
| 3913 | 2022-09-07 | 2022-09-14 | 2022-09-11 |
+-----+-----+-----+-----+
11 rows in set (0.01 sec)

[mysql]> show profiles;
+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+
| 1 | 0.00072700 | set profiling = 1 |
| 2 | 0.00618500 | SELECT * FROM transaction WHERE issue_date = "2022-09-07" |
+-----+-----+
2 rows in set, 1 warning (0.00 sec)

mysql> ALTER TABLE `assignment_6`.`Transaction`
    -> CHANGE COLUMN `issue_date` `issue_date` DATE NOT NULL ,
    -> CHANGE COLUMN `expected_return_date` `expected_return_date` DATE NOT NULL ,
    -> CHANGE COLUMN `actual_return_date` `actual_return_date` DATE NULL DEFAULT NULL ;
Query OK, 1000 rows affected (0.04 sec)
Records: 1000  Duplicates: 0  Warnings: 0
```

The data type was then changed to DATE for the attribute issue\_date.

```
[mysql]> DESCRIBE transaction;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| transaction_ID | int | NO | PRI | NULL |
| issue_date | date | NO | | NULL |
| expected_return_date | date | NO | | NULL |
| actual_return_date | date | YES | | NULL |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Time taken for the same query now reduces to 0.00364000, which is almost half of the earlier execution time.

```
[mysql> show profiles;
+-----+-----+
| Query_ID | Duration | Query
+-----+-----+
| 1 | 0.00072700 | set profiling = 1
| 2 | 0.00618500 | SELECT * FROM transaction WHERE issue_date = "2022-09-07"
| 3 | 0.03670500 | ALTER TABLE `assignment_6`.`Transaction` CHANGE COLUMN `issue_date` `issue_date` DATE NOT NULL ,
CHANGE COLUMN `expected_return_date` `expected_return_date` DATE NOT NULL ,
CHANGE COLUMN `actual_return_date` `actual_return_date` DATE NULL DEFAULT NULL |
| 4 | 0.00364000 | SELECT * FROM transaction WHERE issue_date = "2022-09-07"
+-----+-----+
4 rows in set, 1 warning (0.00 sec)
```

## Task 5

---

Initially the description of all the fee receipts in the penalties column is NULL. Therefore counting over that column returns 0. However, when a NOT NULL description is added, count over description = 1 (i.e. the NOT NULL description is counted). Similarly, when more NOT NULL description values are added to the table, count(description) returns the count of the total NOT NULL description rows from the table.

```
mysql> select * from penalties;
+-----+-----+
| fee_receipt_ID | description |
+-----+-----+
| 368882 | NULL      |
| 768131 | NULL      |
| 888284 | NULL      |
| 930558 | NULL      |
| 955515 | NULL      |
+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> select count(description) from penalties;
+-----+
| count(description) |
+-----+
| 0 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> insert into penalties values(455525, "Returned Late");
Query OK, 1 row affected (0.00 sec)
```

```
[mysql> select * from penalties;
+-----+-----+
| fee_receipt_ID | description   |
+-----+-----+
| 368882 | NULL      |
| 455525 | Returned Late |
| 768131 | NULL      |
| 888284 | NULL      |
| 930558 | NULL      |
| 955515 | NULL      |
+-----+-----+
6 rows in set (0.00 sec)
```

```
mysql> select count(description) from penalties;
+-----+
| count(description) |
+-----+
|          1         |
+-----+
1 row in set (0.01 sec)
```

```
[mysql> insert into penalties values(455526, "Returned Late");
Query OK, 1 row affected (0.00 sec)
```

```
[mysql> select count(description) from penalties;
+-----+
| count(description) |
+-----+
|          2         |
+-----+
1 row in set (0.00 sec)
```

## Task 6

System not compatible to cache queries

## Task 7

Query: To find the names of all users working in the library.

```
select u.user_name
from users as u
where u.user_ID in
(select user_ID
from library_staff as l
where u.user_ID = l.user_ID);
```

Optimized query using joins-

```
select u.user_name
from users as u, library_staff as l
where u.user_ID = l.user_ID;
```

```

mysql> set profiling = 1;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> select u.user_name from users as u where u.user_ID in (select user_ID from library_staff as l where u.user_ID = l.user_ID);
+-----+
| user_name |
+-----+
| James Cintron
| Sterling Wilson
| Dorothy Howell
| Kenneth Zuckerman
| Alan Taylor
|
| Ricky Costa
| Robert Kane
| Trista Owens
+
1016 rows in set (0.12 sec)

mysql> select u.user_name from users as u, library_staff as l where u.user_ID = l.user_ID;
+-----+
| user_name |
+-----+
| James Cintron
| Sterling Wilson
| Dorothy Howell
| Kenneth Zuckerman
| Alan Taylor
|
mysql> set profiling = 0;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> show profiles;
+-----+-----+
| Query_ID | Duration   | Query
+-----+-----+
| 1 | 0.12107075 | select u.user_name from users as u where u.user_ID in (select user_ID from library_staff as l where u.user_ID = l.user_ID)
| 2 | 0.00407425 | select u.user_name from users as u, library_staff as l where u.user_ID = l.user_ID
+
2 rows in set, 1 warning (0.00 sec)

mysql>

```

We can observe that the first query takes `0.12s` however, the optimized query using the join gives the correct output in `0.004s` which is very less compared to the first query.

The reason is because in the first query, for every row of the users table, the complete library\_staff table is checked for a matching ID. This is a slow process as for each row the entire library\_staff table has to be fetched from the disc into memory for comparison. However, in the optimized query, first a temporary table is formed by joining the two tables, resulting in their cartesian product and just the two ID columns of both the tables are compared.

#### *Drawbacks of multiple joins:-*

As the number of tables increases, the cost of optimizing such complex queries increases. This is because for  $n$  joins, there are  $n!$  Possible orderings. A 2 table join can be performed in 2 ways (A join B or B join A), However, for a three table join, there are 6 possible orders and for a 10 table join there are 3,628,800 unique join orders. Thus the time required to inspect all possible query plans is much longer however the difference in actually executing those different join orders is not much. Also, more space is needed to store the intermediate results in temporary tables.

## Task 8

Query: To find all the user\_names who are part of the staff and working in the job\_profile with highest salary

Nested Query:

```
select user_name
from users
where user_id in
(select user_id
from staff
where job_profile in
(select job_profile
from job_salary
where salary =
(select max(salary) from job_salary)));
```

Optimized Query:

```
select distinct(user_name)
from (select job_profile
from job_salary
order by salary desc limit 1) as a
natural join staff
natural join users
```

```
mysql> explain select user_name from users where user_id in (select user_id from staff where job_profile in (select job_profile from job_salary where salary = (select max(salary) from job_salary)));
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | ref |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | job_salary | NULL | ALL | PRIMARY | NULL | NULL |
| 1 | PRIMARY | staff | NULL | ALL | PRIMARY | NULL | NULL |
| 1 | PRIMARY | users | NULL | eq_ref | PRIMARY | PRIMARY | 4 |
| 4 | SUBQUERY | job_salary | NULL | ALL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
| rows | filtered | Extra |
+-----+-----+
| 16 | 10.00 | Using where |
| 1000 | 10.00 | Using where; Using join buffer (hash join) |
| 1 | 100.00 | NULL |
| 16 | 100.00 | NULL |
+-----+-----+
4 rows in set, 1 warning (0.00 sec)

mysql> explain select distinct(user_name) from (select job_profile from job_salary order by salary desc limit 1) as a natural join staff natural join users;
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | ref |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | <derived2> | NULL | system | NULL | NULL | NULL |
| 1 | PRIMARY | staff | NULL | ALL | PRIMARY | NULL | NULL |
| 1 | PRIMARY | users | NULL | eq_ref | PRIMARY | PRIMARY | 4 |
| 2 | DERIVED | job_salary | NULL | ALL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+
| rows | filtered | Extra |
+-----+-----+
| 1 | 100.00 | Using temporary |
| 1000 | 10.00 | Using where |
| 1 | 100.00 | NULL |
| 16 | 100.00 | Using filesort |
+-----+-----+
4 rows in set, 1 warning (0.00 sec)

mysql> show profiles;
+-----+-----+
| Query_ID | Duration |
+-----+-----+
| 1 | 0.00087000 |
| 2 | 0.00085250 |
+-----+-----+
2 rows in set, 1 warning (0.00 sec)
```

We can observe that the original nested query takes 0.00087000 sec., however the optimized query which is written using join operations and order by, limit clauses takes 0.00085250 sec. which is less than the execution time for original nested query.

The reason is clear from the number of scan rows shown in the above image. The max clause in the nested query scans all the 16 rows of job\_salary, however the order by and limit clause effectively scans 1 row from the derived table which makes the query faster.

## Additional features in the website

---

For Task 1 , we have added a search functionality that takes substrings of the publisher name and the street name and returns results about all the publishers whose attributes contain those substrings.

For Task 2 , we have added a search functionality that takes a string and returns the details of all users whose user\_name starts with the input string (i.e. searching from the prefix).

For Task 4 , we added a feature where the user can enter a date, and he will be shown all the transactions where the issue date was the date that he entered.

For Task 7 , we have implemented a join functionality where the user can find the names of all users working in the library in a faster manner due to our optimized query using join.

## References

<https://www.bridge-global.com/blog/tips-to-improve-mysql-query-performance/>