

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

View Of Users Relation in Workbench

MySQLWorkbench File Edit View Query Database Server Tools Scripting Help

Local instance 3306 MySQL Workbench

Administration Schemas

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

Context Help Snippets

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

Object Info Session

Table: Users

Columns:

user_ID	int PK
User_name	varchar(45)

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...

Users 1 Apply Revert

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.

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:

user_ID	int PK
User_name	varchar(45)

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...
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

The screenshot shows the MySQL Workbench interface with the 'Users' table selected in the 'Schemas' tab. The table structure is displayed with columns 'user_ID' and 'User_name'. Data rows are listed, including names like Kim, Bob, Siyia, Snape, Rachel, Sam, Sara, Gita, Monica, Harry, Hermione, Ginny, Ron, Neville, Chandler, Lily, Ross, Phoebe, and Stefan. The 'Actions' panel at the bottom shows two recent queries: one for 18 rows and another for 19 rows, both executed quickly. The status bar indicates the date and time as 'Wed 30 Mar 7:07:25 PM'.

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



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



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 says "MySQL Workbench" and "Local instance 3306". The left sidebar has a tree view under "SCHEMAS" showing various tables like Publishers, Staff, Student, System_Allocation, Transaction, User_Issue, and Users. The "Users" table is selected. The main area displays a "Result Grid" with columns "user_ID" and "User_name". The grid shows 19 rows of data. A vertical toolbar on the right contains icons for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan. A message in the center of the toolbar says: "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 entries

Search:

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

MySQL Workbench - Local instance 3306

Administration Schemas

Result Grid

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

user_ID	User_name
101	Kim
102	Bob
103	Sya
123	Snape
200	Rachel
301	Sam
302	Suri
401	Gita
501	Monica
601	Harry
602	Ginny
603	Hermoine
604	Ron
605	Neville
721	Chandler
723	Lily
811	Ross
922	Phoebe
HULL	HULL

Object Info Session

Table: Users

Columns:

- user_ID** int PK
- User_name** varchar(45)

Users 1

Action Output

Time	Action	Response	Duration / Fetch Time
19:09:32	SELECT * FROM assignment_6.Users LIMIT 0, 1000	18 row(s) returned	0.00030 sec / 0.000...
4	19:09:31 SELECT * FROM assignment_6.Users LIMIT 0, 1000	18 row(s) returned	0.00030 sec / 0.000...

Query Completed

View Of Transactions Relation

Library Management System

Transaction

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

Show 10 entries Search: _____

Showing 1 to 7 of 7 entries Previous 1 Next

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:

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	<input type="text"/>
issue_date	<input type="text"/>
expected_return_date	<input type="text"/>
actual_return_date	<input type="text"/>

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 Edit: Export/Import: Result Grid Form Editor Field Types Query Stats Execution Plan

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_Phone
- > Purchase
- > Staff
- > Strike
- > Student
- > System_Allocation
- > Transaction
- > User_Issue
- Users
 - > Columns
 - > Indexes
 - > Foreign Keys
 - > Triggers
 - > Users_Phone
- Views

Object Info Session

Table: Transaction

Columns:

transaction_ID	int PK
issue_date	datetime
expected_return_date	datetime
actual_return_date	datetime

Action Output

Time	Action	Response	Duration / Fetch Time
1	19:34:17	SELECT * FROM assignment_6.Transaction LIMIT 0, 1000	8 row(s) returned 0.00048 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"/>
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	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"/>
Add	

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
HULL	HULL

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	4 row(s) returned	0.00029 sec / 0.0000...

Apply Revert

Query Completed

The screenshot shows the MySQL Workbench interface with the 'User_Issue' table selected. The table has four rows of data. The 'transaction_ID' column contains values 4238, 4278, 4898, and 4980. The 'user_ID' column contains values 101, 103, 200, and 922. A message bar at the top right states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help." The bottom section displays the execution plan for a query, showing one row with a duration of 0.00029 sec and a fetch time of 0.0000... sec.

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 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 Next

Input Values to be Inserted

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

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:

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 Next

Input Values to be Inserted

transaction_ID	<input type="text"/>
book_ID	<input type="text"/>

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the schema tree with the 'Book_Issue' table selected. The main area shows a 'Result Grid' with the following data:

	transaction_ID	book_ID
1	4238	12345
2	4898	23456
3	4278	34567
4	4232	45678
5	4980	67890
6	HULL	HULL

The 'Object Info' tab shows the table definition:

Table: Book_Issue

Columns:

Column Name	Type	PK
transaction_ID	int	PK
book_ID	int	

The 'Session' tab shows the current session details.

The 'Query Output' pane shows the executed query and its execution statistics:

Action	Time	Response	Duration / Fetch Time
SELECT * FROM assignment_6.Book_Issue LIMIT 0, 1000	19:36:13	5 row(s) returned	0.00039 sec / 0.000...

A message in the top right corner states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

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.

Result Grid Form Editor Field Types Query Stats Execution Plan

Table: Library_Systems

Columns:

system_ID	int PK
system_specs	text

Library_Systems 1

Action Output

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

Query Completed

The screenshot shows the MySQL Workbench interface with the 'Library_Systems' table selected. The 'Result Grid' pane displays four rows of data from the 'system_specs' column. A tooltip on the right side of the interface states: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.' The 'Execution Plan' sidebar is visible on the right.

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

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

MySQL Workbench - Local instance 3306

Administration Schemas

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

user_ID	system_ID
561	1011
604	1024
123	1035
NULL	NULL

Object Info Session

Table: Library_Systems

Columns:

- system_ID int PK
- system_specs text

Action Output

Time	Action	Response	Duration / Fetch Time
19:39:17	SELECT * FROM assignment_6.System_Allocation LIMIT 0, 1000	3 row(s) returned	0.00029 sec / 0.0000...

Query Completed

Error due to Foreign Key Constraint in Users Relation

Show All entries

Search:

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

127.0.0.1:5000/delete/Users?Users_user_id=101&Users_user_name=Kim&

Previous 1 Next

File Edit View History Bookmarks Profiles Tab Window Help

127.0.0.1:5000/delete/Users?Users_user_id=101&Users_user_name=Kim&

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

The screenshot shows a web application interface for managing books. At the top, there is a navigation bar with various links: Authors, Genres, Location, Transaction, Penalties, Purchase, Book_Issue, User_Issue, Strike, Book_Pub, Book_Auth, Book_Genre, Book_Location, System_Allocation, and Library_Staff_Working_Hours. Below the navigation bar, there is a search bar labeled "Search:" and a dropdown menu showing "Show 10 entries". A table displays seven book entries with columns: book_ID, book_name, num_pages, availability, and Delete. The table data is as follows:

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

The screenshot shows a Chrome browser window. The address bar shows the URL `127.0.0.1:5000/add/Books`. The main content area displays an "Input Values to be Inserted" dialog box with the following fields:

book_ID	67890
book_name	Databases
num_pages	623
availability	True

Below the dialog box is a button labeled "Add".

There was an issue adding the entry: (1062, "Duplicate entry '67890' for key 'books.PRIMARY'"")

Tasks Assignment 8

Task 1

Query 1

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

Optimized query 1

```
select * from publishers where publisher_name like 'David%'
union
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 'David%' or street_name like '%Scott';
+-----+-----+-----+-----+-----+-----+-----+-----+
| publisher_ID | publisher_name | street_num | street_name | city | state | zip_code | email |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 10148 | David Shin | 123 | John Alvarado | abc | abc | 567841 | abc_11 |
| 11561 | David Arroyo | 123 | Isaac Besaw | abc | abc | 363326 | abc_707 |
| 28135 | David Tyler | 123 | Janet Pfeiffer | abc | abc | 355331 | abc_731 |
| 33429 | David Kennedy | 123 | Kelly Thomason | abc | abc | 291835 | abc_517 |
| 44402 | Jamel Boham | 123 | Susan Scott | abc | abc | 615007 | abc_718 |
| 46106 | David Crews | 123 | Andrew Cleaves | abc | abc | 752133 | abc_831 |
| 48120 | David Stott | 123 | John Pillar | abc | abc | 264312 | abc_159 |
| 48250 | David High | 123 | Melissa Corr | abc | abc | 216646 | abc_741 |
| 53521 | David Tara | 123 | Maria Wickham | abc | abc | 766889 | abc_142 |
| 61885 | David Brown | 123 | Myrtle Tipton | abc | abc | 675876 | abc_494 |
| 63690 | David Ascencio | 123 | Raymond Gagnon | abc | abc | 663837 | abc_442 |
| 85831 | David Gehl | 123 | Barry Schwartz | abc | abc | 723095 | abc_723 |
+-----+-----+-----+-----+-----+-----+-----+-----+
12 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 publisher_name like 'David%' 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 | 1 NULL | NULL | 1 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, 1 warning (0.06 sec)
Records: 0 Duplicates: 0 Warnings: 1

mysql> CREATE FULLTEXT INDEX street_name_index ON publishers(street_name);
Query OK, 0 rows affected (0.02 sec)
Records: 0 Duplicates: 0 Warnings: 0

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

mysql> select * from publishers where match(publisher_name) against ('David*' in boolean mode) union select * from publishers where match(street_name) against ('*Scott' in boolean mode);
+-----+-----+-----+-----+-----+-----+-----+-----+
| publisher_ID | publisher_name | street_num | street_name | city | state | zip_code | email |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 10148 | David Shin | 123 | John Alvarado | abc | abc | 567841 | abc_11 |
| 11561 | David Arroyo | 123 | Isaac Besaw | abc | abc | 363326 | abc_707 |
| 28135 | David Tyler | 123 | Janet Pfeiffer | abc | abc | 355331 | abc_731 |
| 33429 | David Kennedy | 123 | Kelly Thomason | abc | abc | 291835 | abc_517 |
| 44402 | Jamel Boham | 123 | Susan Scott | abc | abc | 615007 | abc_718 |
| 46106 | David Crews | 123 | Andrew Cleaves | abc | abc | 752133 | abc_831 |
| 48120 | David Stott | 123 | John Pillar | abc | abc | 264312 | abc_159 |
| 48250 | David High | 123 | Melissa Corr | abc | abc | 216646 | abc_741 |
| 53521 | David Tara | 123 | Maria Wickham | abc | abc | 766889 | abc_142 |
| 61885 | David Brown | 123 | Myrtle Tipton | abc | abc | 675876 | abc_494 |
| 63690 | David Ascencio | 123 | Raymond Gagnon | abc | abc | 663837 | abc_442 |
| 85831 | David Gehl | 123 | Barry Schwartz | abc | abc | 723095 | abc_723 |
| 98760 | Elsa Akin | 123 | Scott Groner | abc | abc | 218124 | abc_985 |
+-----+-----+-----+-----+-----+-----+-----+-----+
15 rows in set (0.00 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 ('David*' in boolean mode) union 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: no_ranking |
| 2 | UNION | publishers | NULL | fulltext | street_name_index | street_name_index | 0 | const | 1 | 100.00 | Using where; Ft_hints: no_ranking |
| NULL | UNION RESULT | <union1,2> | NULL | ALL | 1 NULL | Using temporary |
+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set, 1 warning (0.00 sec)

mysql> show profiles;
+-----+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+-----+
| 1 | 0.00130500 | select * from publishers where publisher_name like 'David%' or street_name like '%Scott' |
| 2 | 0.00037700 | select * from publishers where match(publisher_name) against ('David*' in boolean mode) union select * from publishers where match(street_name) against ('*Scott' in boolean mode) |
+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)

```

Query 1

Rows: 1000

Duration: 0.00130500 s

Optimized query 1

Rows: 2 (reduced)

Duration: 0.00037700 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`

The decision to make date from varchar to date type results in decrease of execution time as the date type is inherently more structured. When searching for the date, instead of searching for every character and pattern in varchar, a date can be divided by hyphens over day, month and year. The search values over these three divisions help in faster retrieval of data.

```
[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 | 0.000618500 | 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

Select statements like

- Location: Getting the different books locations of the library
- System_Allocation: Selecting all the allotted library systems
- Genres: Seeing the different genres variety for books
and many more are some of the frequently run select statements. Caching them will reduce the execution time significantly in the next run. If a query has been cached, the next time it is run, it will be fetched from the cache to give the output in lesser time. Also, the above three tables will be updated less frequently. So, it is better to use cached data for select queries on them as cached data gets removed if we run update/insert query on that particular table.

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
+-----+
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
+-----+
2 rows in set, 1 warning (0.00 sec)

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 | key_len | ref |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | job_salary | NULL | ALL | PRIMARY | NULL | NULL | NULL |
| 1 | PRIMARY | staff | NULL | ALL | PRIMARY | NULL | NULL | NULL |
| 1 | PRIMARY | users | NULL | eq_ref | PRIMARY | PRIMARY | 4 | assignment8.staff.user_ID |
| 4 | SUBQUERY | job_salary | NULL | ALL | NULL | NULL | NULL | 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 | key_len | ref |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | <derived2> | NULL | system | NULL | NULL | NULL | NULL |
| 1 | PRIMARY | staff | NULL | ALL | PRIMARY | NULL | NULL | NULL |
| 1 | PRIMARY | users | NULL | eq_ref | PRIMARY | PRIMARY | 4 | assignment8.staff.user_ID |
| 2 | DERIVED | job_salary | NULL | ALL | NULL | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
4 rows in set, 1 warning (0.00 sec)

mysql> show profiles;
+-----+-----+
| Query_ID | Duration |
+-----+-----+
| 1 | 0.00087000 | 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))) |
| 2 | 0.00085250 | 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 |
+-----+-----+
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.

Contribution

G1: All the members had equal contribution (12.5% each out of 50%)

- 19110080: Chetan Kishore
- 19110093: Manas Malpuri

- 19110106: Gaurav Viramgami
- 19110136: Shrreya Singh

G2: All the members had equal contribution (12.5% each out of 50%)

- 19110090: Shubh Lavti
- 19110092: Mahika Om Jaguste
- 19110127: Nipun Mahajan
- 19110128: Paras Gupta

References

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