

DBMS PROJECT

TITLE OF THE PROJECT:

BUS BOOKING MANAGEMENT SYSTEM

Name: Rajesh M Mysoremath

Description and Scope of Project:

Online bus booking management system is a project which presents a portal for bus ticket reservation. This application lets in users to book bus tickets from somewhere and anytime. The customer can without difficulty e book their tickets and cancel tickets. The user can additionally view the details of the journey.

Features We can see in this project,

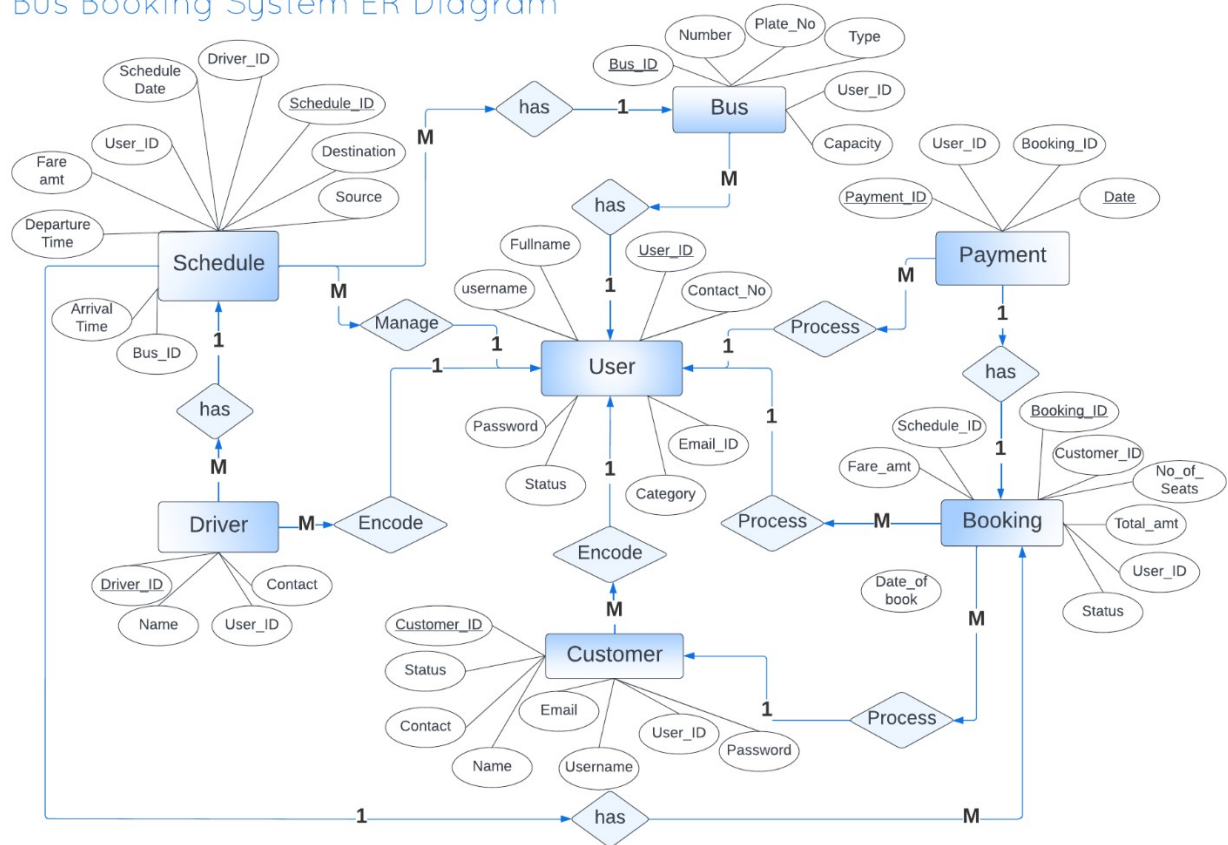
Thread operations in

- Users
- Bus
- Driver
- Customer
- Schedule
- Booking
- Payment
- Query Box

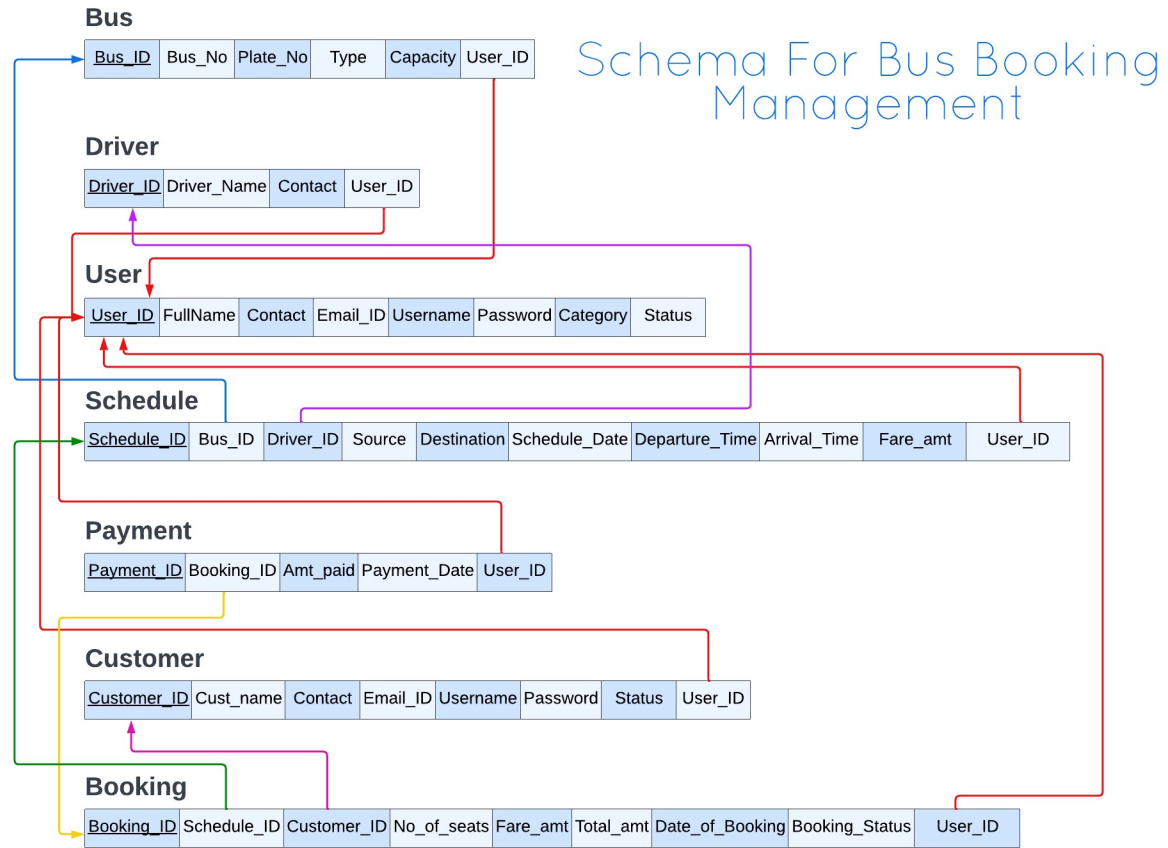
The customer to enquire the availability of seats in a particular bus at particular date. To reserve a ticket and he can cancel a reserved ticket. The current bus booking system relies on buying tickets from the conductor for commuting to and from a location through public transportation. The task can be tedious if the number of commuters is large. Also, payment in cash can be difficult if the payable denominations are uneven. Online Bus Booking system allows the computer to either have a specific amount of money on his Android Based mobile, from which the ticket can be charged. Or, the costumer can buy the ticket on the bus.

ER Diagram

Bus Booking System ER Diagram



Relational Schema



DDL Statements

Building The Database

1) Create User Table

```
CREATE TABLE `user` (  
  `users_id` int(11) NOT NULL,  
  `full_name` varchar(50) NOT NULL,  
  `contact_no` varchar(15) NOT NULL,  
  `email_address` varchar(30) NOT NULL,  
  `username` varchar(30) NOT NULL,  
  `userpassword` varchar(30) NOT NULL,  
  `account_category` varchar(100) NOT NULL,  
  `account_status` varchar(100) NOT NULL)
```

```
ALTER TABLE `user`  
  ADD PRIMARY KEY (`users_id`);
```

2) Create Bus Table

```
CREATE TABLE `bus` (  
  `bus_id` int(11) NOT NULL,  
  `bus_number` varchar(15) NOT NULL,  
  `bus_plate_number` varchar(15) NOT NULL,  
  `bus_type` int(1) NOT NULL,  
  `capacity` int(3) NOT NULL,  
  `users_id` int(11) NOT NULL)
```

```
ALTER TABLE `bus`  
  ADD PRIMARY KEY (`bus_id`),  
  ADD KEY `user_id` (`users_id`);
```

3) Create Driver Table

```
CREATE TABLE `driver` (  
  `driver_id` int(11) NOT NULL,  
  `driver_name` varchar(50) NOT NULL,  
  `driver_contact` varchar(15) NOT NULL,  
  `users_id` int(11) NOT NULL)
```

```
ALTER TABLE `driver`  
  ADD PRIMARY KEY (`driver_id`),  
  ADD KEY `user_id` (`users_id`);
```

4) Create Customer Table

```
CREATE TABLE `customer` (  
  `customer_id` int(11) NOT NULL,  
  `customer_name` varchar(50) NOT NULL,  
  `customer_contact` varchar(15) NOT NULL,  
  `customer_email` varchar(30) NOT NULL,  
  `username` varchar(30) NOT NULL,  
  `cust_password` varchar(30) NOT NULL,  
  `account_status` varchar(100) NOT NULL,  
  `users_id` int(11) NOT NULL)
```

```
ALTER TABLE `customer`  
  ADD PRIMARY KEY (`customer_id`),  
  ADD KEY `user_id` (`users_id`);
```

5)Create Schedule Table

```
CREATE TABLE `schedule` (  
  `schedule_id` int(11) NOT NULL,  
  `bus_id` int(11) NOT NULL,  
  `driver_id` int(11) NOT NULL,  
  `starting_point` varchar(30) NOT NULL,  
  `destination` varchar(30) NOT NULL,  
  `schedule_date` date NOT NULL,  
  `departure_time` varchar(100) NOT NULL,  
  `estimated_arrival_time` varchar(100) NOT NULL,  
  `fare_amount` int(11) NOT NULL,  
  `remarks` varchar(100) NOT NULL,  
  `users_id` int(11) NOT NULL)
```

```
ALTER TABLE `schedule`  
  ADD PRIMARY KEY (`schedule_id`),  
  ADD KEY `bus_id` (`bus_id`,`driver_id`,`users_id`),  
  ADD KEY `user_id` (`users_id`),  
  ADD KEY `driver_id` (`driver_id`);
```

6) Create Booking Table

```
CREATE TABLE `booking` (  
  `booking_id` int(11) NOT NULL,  
  `schedule_id` int(11) NOT NULL,  
  `customer_id` int(11) NOT NULL,  
  `number_of_seats` int(2) NOT NULL,  
  `fare_amount` int(11) NOT NULL,  
  `total_amount` int(11) NOT NULL,  
  `date_of_booking` date NOT NULL,  
  `booking_status` varchar(100) NOT NULL,  
  `users_id` int(11) NOT NULL)
```

```
ALTER TABLE `booking`  
  ADD PRIMARY KEY (`booking_id`),  
  ADD KEY `schedule_id` (`schedule_id`,`customer_id`,`users_id`),  
  ADD KEY `user_id` (`users_id`),  
  ADD KEY `customer_id` (`customer_id`);
```

7) Create Payment Table

```
CREATE TABLE `payment` (  
  `payment_id` int(11) NOT NULL,  
  `booking_id` int(11) NOT NULL,  
  `amount_paid` int(11) NOT NULL,  
  `payment_date` date NOT NULL,  
  `users_id` int(11) NOT NULL)
```



```
ALTER TABLE `payment`  
  ADD PRIMARY KEY (`payment_id`),  
  ADD KEY `booking_id` (`booking_id`,`users_id`),  
  ADD KEY `user_id` (`users_id`);
```

Populating The Database

1) Populating Users

```
INSERT INTO `user` (`users_id`, `full_name`, `contact_no`, `email_address`, `username`, `userpassword`, `account_category`, `account_status`) VALUES ('1564679', 'Tasmai', '78945013.0', 'Tasmai123@gmail.com', 'Tasmai', 'Tasmai@123', '1', 'Active'), ('4789658', 'Sagar', '798441614.0', 'sagar123@gmail.com', 'Sagar', 'Sagar@123', '1', 'Deactive')
```

2) Populating Buses

```
INSERT INTO `bus` (`bus_id`, `bus_number`, `bus_plate_number`, `bus_type`, `capacity`, `users_id`) VALUES ('4561187', '7896254', 'KA124579', 'Non-AC Sleeper', '45', '4789658'), ('7862157', '156467', 'KA214357', 'Non-AC', '57', '1564679')
```

3) Populating Drivers

```
INSERT INTO `bus` (`bus_id`, `bus_number`, `bus_plate_number`, `bus_type`, `capacity`, `users_id`) VALUES ('4561187', '7896254', 'KA124579', 'Non-AC Sleeper', '45', '4789658'), ('7862157', '156467', 'KA214357', 'Non-AC', '57', '1564679')
```

4) Populating Customers

```
INSERT INTO `customer` (`customer_id`, `customer_name`, `customer_contact`, `customer_email`, `username`, `cust_password`, `account_statuses`, `users_id`) VALUES ('4547987', 'hari', '79461648', 'hari123@gmail.com', 'Hari', 'Hari@123', 'Not Done', '1564679'), ('6014064', 'Sharu', '244600747', 'sharu@gmail.com', 'Sharu', 'Sharu@123', 'Done', '4789658')
```

5) Populating Schedules

```
INSERT INTO `schedule` (`schedule_id`, `bus_id`, `driver_id`, `starting_point`, `destination`, `schedule_date`, `departure_time`, `estimated_arrival_time`, `fare_amount`, `remarks`, `users_id`) VALUES ('154578', '4561187', '165461', 'Pune', 'Banglore', '2022-12-14', '05:45:00', '06:15:00', '2540', 'via: Bangalore', '4789658'), ('7916548', '7862157', '781647', 'Bangalore', 'Mumbai', '2022-11-29', '01:58:00', '01:58:00', '4578', 'via: Pune', '1564679')
```

6) Populating Booking

```
INSERT INTO `booking` (`booking_id`, `schedule_id`, `customer_id`, `number_of_seats`, `fare_amount`, `total_amount`, `date_of_booking`, `booking_status`, `users_id`) VALUES ('4679416', '7916548', '4547987', '15', '1235', '1300', '2022-11-29', 'Confirmed', '4789658'), ('7841647', '154578', '6014064', '25', '4578', '4600', '2022-12-15', 'Not Confirmed', '1564679')
```

7)Populating Payment

```
INSERT INTO `payment` (`payment_id`, `booking_id`, `amount_paid`, `payment_date`, `users_id`) VALUES ('164878', '4679416', '4578', '2022-12-22', '4789658'), ('4679167', '7841647', '1542', '2022-11-29', '1564679')
```

JOIN QUERIES

Query 1:

Joining bus and driver table using users_id attribute

The screenshot shows a web application interface for a 'BUS BOOKING MANAGEMENT SYSTEM'. On the left, there is a sidebar with a 'SERVICES' section and a 'QUERY BOX' dropdown. The main area displays the title 'BUS BOOKING MANAGEMENT SYSTEM' and a query input field. The query entered is: `SELECT * from bus JOIN driver using(users_id)`. Below the query input is a 'Submit' button. The result is displayed as a table with 8 columns: users_id, bus_id, bus_number, bus_plate_number, bus_type, capacity, driver_id, and driver_name. The table contains two rows of data.

	users_id	bus_id	bus_number	bus_plate_number	bus_type	capacity	driver_id	driver_name
0	4789658	4561187	7896254	KA124579	Non-AC Sleeper	45	165461	Darsha
1	1564679	7862157	156467	KA214357	Non-AC	57	781647	Shreya

Made with Streamlit

Query 2:

Joining Booking table and Customer table using customer_id attribute

The screenshot shows the same web application interface as the previous one. The query input field now contains: `SELECT * from Booking JOIN customer using(customer_id)`. Below the query input is a 'Submit' button. The result is displayed as a table with 8 columns: customer_id, booking_id, schedule_id, number_of_seats, fare_amount, totalAmount, and date. The table contains two rows of data.

	customer_id	booking_id	schedule_id	number_of_seats	fare_amount	totalAmount	date
0	4547987	4679416	7916548	15	1235	1300	2022-01-01
1	6014064	7841647	154578	25	4578	4600	2022-01-01

	customer_id	booking_id	schedule_id	number_of_seats	fare_amount	total_amount	date_of_booking	booking_status	users_id	customer_name	customer_contact	customer_email	username	cust_password	account_status
0	4547987	4679416	7916548	15	1235	1300	2022-11-29	Confirmed	1564679	hari	79461648	hari123@gmail.com	Hari	Hari@123	Not Done
1	6014064	7841647	154578	25	4578	4600	2022-12-15	Not Confirmed	4789658	Sharu	244600747	sharu@gmail.com	Sharu	Sharu@123	Done

Query 3:

Joining 3 tables (User table, driver table, Customer table) using common attributes

```
SELECT u.users_id,u.full_name,d.driver_id,d.driver_name,
c.customer_id, c.customer_name from user u JOIN driver d
ON u.users_id = d.users_id JOIN customer c ON u.users_id =
c.users_id;
```

SERVICES

QUERY BOX

BUS BOOKING MANAGEMENT SYSTEM

Enter the query here:

```
SELECT
u.users_id,u.full_name,d.driver_id,d.driver_name,c.customer_id,c.c
ustomer_name from user u
JOIN driver d
ON u.users_id = d.users_id
JOIN customer c
ON u.users_id = c.users_id;
```

Submit

	users_id	full_name	driver_id	driver_name	customer_id	customer_name
0	1564679	Tasmai	781647	Shreyas	4547987	hari
1	4789658	Sagar	165461	Darshan	6014064	Sharu

Query 4:

Joining 4 tables (User table, Customer table, Schedule and bus table) using common attributes

```
SELECT u.users_id, u.username, u.contact_no, c.customer_name,
c.customer_email,s.starting_point,s.schedule_date,
b.bus_plate_number,b.bus_type from user u JOIN customer c
ON u.users_id = c.users_id JOIN schedule s ON c.users_id =
s.users_id JOIN bus b ON s.users_id = b.users_id;
```

SERVICES

QUERY BOX

BUS BOOKING MANAGEMENT SYSTEM

Enter the query here:

```
SELECT u.users_id, u.username, u.contact_no,
c.customer_name,c.customer_email,s.starting_point,s.schedule_date,
b.bus_plate_number,b.bus_type from user u
JOIN customer c
ON u.users_id = c.users_id
JOIN schedule s
ON c.users_id = s.users_id
JOIN bus b
ON s.users_id = b.users_id;
```

Submit

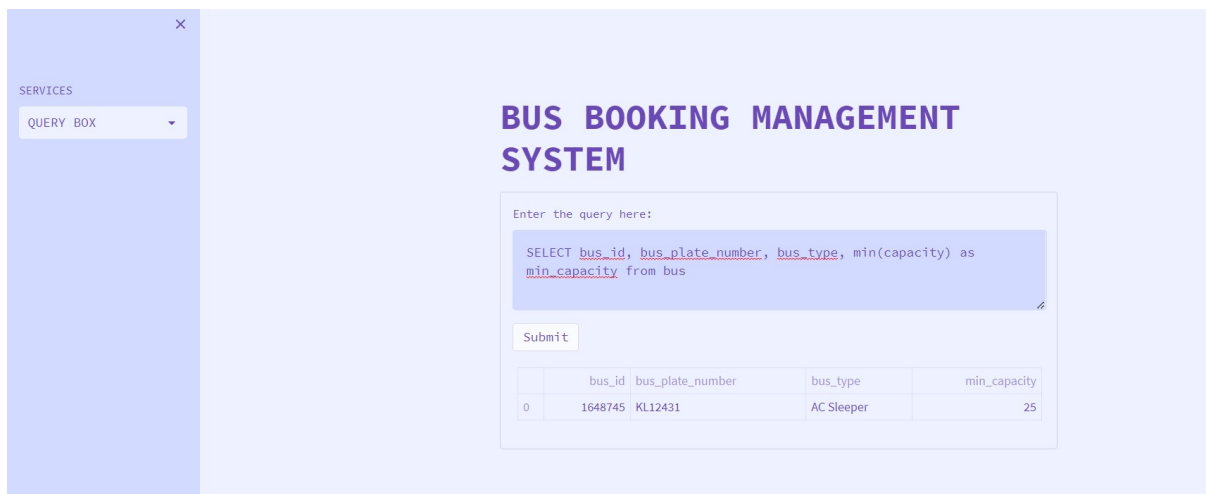
	users_id	username	contact_no	customer_name	customer_email	starting_point	scl
0	4789658	Sagar	798441614.0	Sharu	sharu@gmail.com	Pune	20.
1	1564679	Tasmai	78945013.0	hari	hari123@gmail.com	Bangalore	20.

	users_id	username	contact_no	customer_name	customer_email	starting_point	schedule_date	bus_plate_number	bus_type
0	4789658	Sagar	798441614.0	Sharu	sharu@gmail.com	Pune	2022-12-14	KA124579	Non-AC Sleeper
1	1564679	Tasmai	78945013.0	hari	hari123@gmail.com	Bangalore	2022-11-29	KA214357	Non-AC

Aggregate Functions

Query 1:

Getting the Details of minimum number of capacities in bus table



The screenshot shows a web application titled "BUS BOOKING MANAGEMENT SYSTEM". On the left, there is a sidebar with a "SERVICES" section containing a "QUERY BOX" dropdown. The main area has a text input field labeled "Enter the query here:" containing the SQL query: `SELECT bus_id, bus_plate_number, bus_type, min(capacity) as min_capacity from bus`. Below the input is a "Submit" button. The result is displayed in a table with the following data:

	bus_id	bus_plate_number	bus_type	min_capacity
0	1648745	KL12431	AC Sleeper	25

Query 2:

Listing Number of Users in Bus table, and sorted low to high according to user id.

```
SELECT COUNT(users id) as count, users id FROM bus GROUP BY users_id ORDER BY COUNT (users_id) DESC;
```

SERVICES
QUERY BOX

BUS BOOKING MANAGEMENT SYSTEM

Enter the query here:

```
SELECT COUNT(users_id) as count, users_id  
FROM bus  
GROUP BY users_id  
ORDER BY COUNT(users_id) DESC;
```

Submit

	count	users_id
0	2	1564679
1	1	3254160
2	1	4789658

Query 3:

List the number of entries in our bus table having capacity > 46

SERVICES
QUERY BOX

BUS BOOKING MANAGEMENT SYSTEM

Enter the query here:

```
SELECT COUNT(capacity)  
FROM bus  
WHERE capacity > 46;
```

Submit

	COUNT(capacity)
0	2

Query 4:

Using min () and max () functions to display max and min no. of Capacity in bus table

```
SELECT MIN(capacity) AS Minimum_Capacity, MAX (capacity)  
AS Maximum_Capacity FROM bus
```

SERVICES

QUERY BOX

BUS BOOKING MANAGEMENT SYSTEM

Enter the query here:

```
SELECT MIN(capacity) AS Minimum_Capacity,  
      MAX(capacity) AS Maximum_Capacity  
FROM bus
```

Submit

	Minimum_Capacity	Maximum_Capacity
0	25	57

Set Operations

Query 1:

Union operation between user and Customer table of our database

SERVICES
QUERY BOX

BUS BOOKING MANAGEMENT SYSTEM

Enter the query here:

```
SELECT * FROM user  
UNION  
SELECT * FROM customer;
```

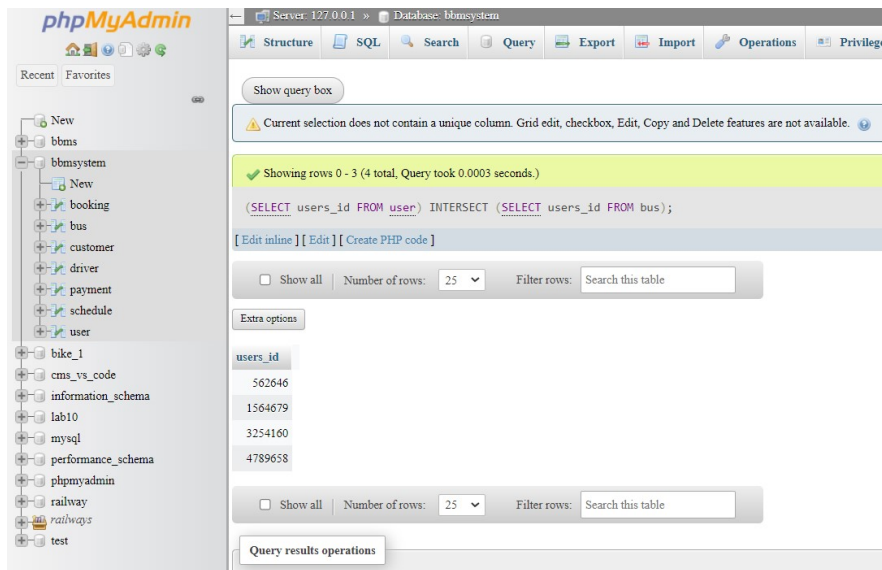
Submit

	users_id	full_name	contact_no	email_address	username	userpassword	accou
0	1564679	Tasmai	78945013.0	Tasmai123@gmail.com	Tasmai	Tasmai@123	1
1	3254160	Ganesh	46741347.0	Ganesh123@gmail.com	Ganesh	Ganesh@123	2
2	4789658	Sagar	798441614.0	sagar123@gmail.com	Sagar	Sagar@123	1
3	4547987	hari	79461648	hari123@gmail.com	Hari	Hari@123	Not D
4	6014064	Sharu	244600747	sharu@gmail.com	Sharu	Sharu@123	Done

	users_id	full_name	contact_no	email_address	username	userpassword	account_category	account_status
0	1564679	Tasmai	78945013.0	Tasmai123@gmail.com	Tasmai	Tasmai@123	1	Active
1	3254160	Ganesh	46741347.0	Ganesh123@gmail.com	Ganesh	Ganesh@123	2	Deactive
2	4789658	Sagar	798441614.0	sagar123@gmail.com	Sagar	Sagar@123	1	Deactive
3	4547987	hari	79461648	hari123@gmail.com	Hari	Hari@123	Not Done	1564679
4	6014064	Sharu	244600747	sharu@gmail.com	Sharu	Sharu@123	Done	4789658

Query 2:

Intersection between user and bus table



Query 3:

Using union operation over bus table to differentiate users as availability low or high.

SELECT bus_id,bus_plate_number,bus_type, ' low' as availability

From bus

Where capacity<=40

UNION

SELECT bus_id,bus_plate_number,bus_type, 'high' as availability

From bus

Where capacity>=40

SERVICES
QUERY BOX

BUS BOOKING MANAGEMENT SYSTEM

Enter the query here:

```

SELECT bus_id,bus_plate_number,bus_type, 'low' as availability
from bus
where capacity<=40
UNION
SELECT bus_id,bus_plate_number,bus_type, 'high' as availability
from bus
where capacity>40

```

Submit

	bus_id	bus_plate_number	bus_type	availability
0	312124	KA322314	Non-AC	low
1	648789	KA23231	AC Sleeper	low
2	1648745	KL12431	AC Sleeper	low
3	4561187	KA124579	Non-AC Sleeper	high
4	7862157	KA214357	Non-AC	high
5	7981554	KA12344	General	high

Query 4:

Find all users except account status is deactive.

SELECT * FROM user

EXCEPT;

SELECT * FROM user WHERE user.acoount_status='Deactive';

The screenshot shows the phpMyAdmin interface with the following details:

- Database:** bbmsystem
- Table:** user
- Query:**

```

SELECT * FROM user EXCEPT SELECT * FROM user WHERE user.account_status="Deactive";

```
- Results:** Showing rows 0 - 2 (3 total, Query took 0.0005 seconds.)
- Table Structure:**

users_id	full_name	contact_no	email_address	username	userpassword	account_category	account_status
562646	Vivian	29672761.0	Vivian123@gmail.com	Vivian	Vivian@123	3	Active
1564679	Tasmai	78945013.0	Tasmai123@gmail.com	Tasmai	Tasmai@123	1	Active
8467641	Vismai	351467700.0	vismai123@gmail.com	Vismai	Vismai@123	2	Active

Functions and Procedures

Query 1:

The below query is function to get if the fare amount is less than 1600 then cheap otherwise expensive in schedule table

Function:

```
DELIMITER $$
```

```
CREATE DEFINER=`root`@`localhost` FUNCTION `price`(`fare_amount`  
INT) RETURNS varchar(20) CHARSET utf8mb4
```

```
    DETERMINISTIC
```

```
BEGIN
```

```
DECLARE v varchar(100);
```

```
if fare_amount<1600 THEN
```

```
SET v = "Cheap";
```

```
ELSE
```

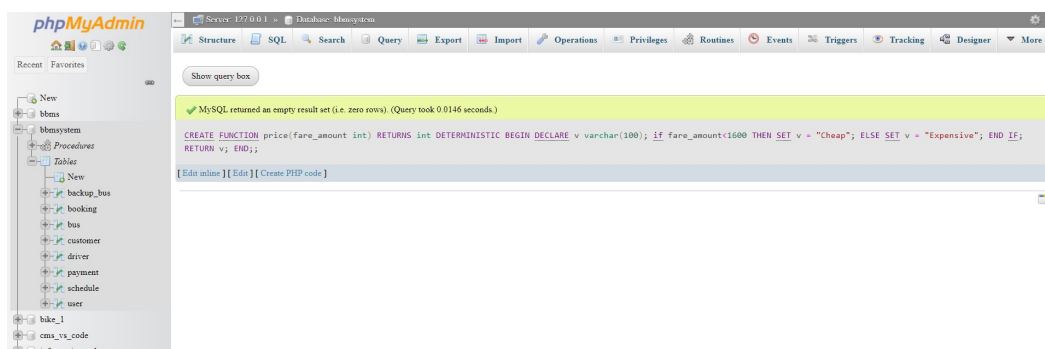
```
SET v = "Expensive";
```

```
END IF;
```

```
RETURN v;
```

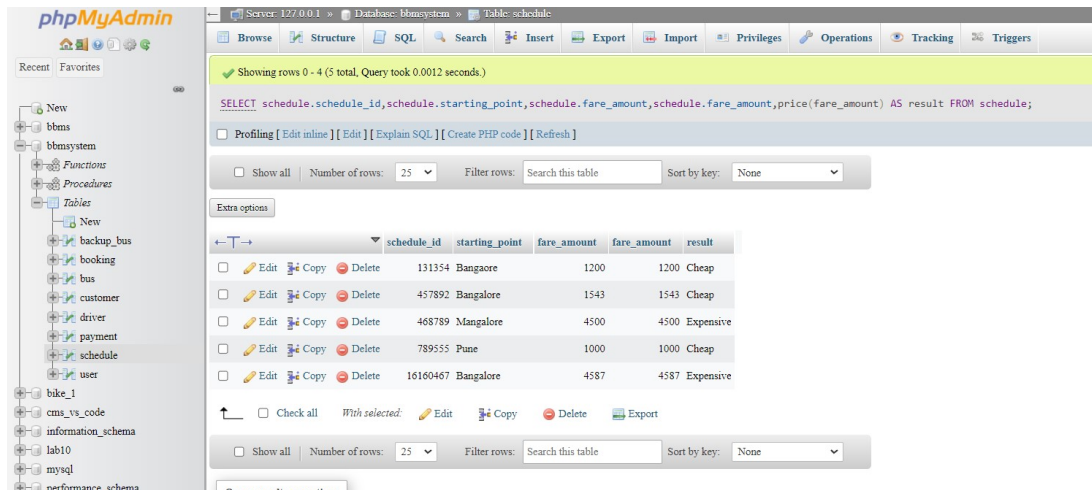
```
END$$
```

```
DELIMITER ;
```



Check Result:

SELECT schedule.schedule_id,schedule.starting point,schedule.fare_amount,price(fare_amount) AS result FROM schedule;

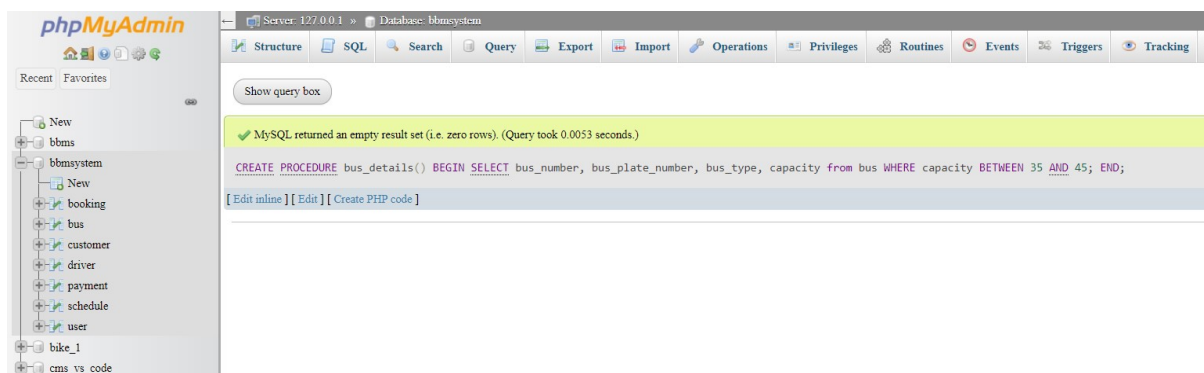


The screenshot shows the phpMyAdmin interface with the 'schedule' table selected. The query results are displayed in a table with 5 rows. The columns are: schedule_id, starting_point, fare_amount, fare_amount, and result. The results show various bus routes and their fares, categorized as Cheap or Expensive.

schedule_id	starting_point	fare_amount	fare_amount	result
131354	Bangalore	1200	1200	Cheap
457892	Bangalore	1543	1543	Cheap
408789	Mangalore	4500	4500	Expensive
789555	Pune	1000	1000	Cheap
16160467	Bangalore	4587	4587	Expensive

Query 2:

The below query is to write a procedure to get bus number, plate number, bus types from bus table where bus capacity is between 35 and 45



The screenshot shows the phpMyAdmin interface with the 'Query' tab selected. The query results show that the MySQL returned an empty result set (i.e. zero rows). The query executed was: CREATE PROCEDURE bus_details() BEGIN SELECT bus_number, bus_plate_number, bus_type, capacity from bus WHERE capacity BETWEEN 35 AND 45; END;

```
CREATE PROCEDURE bus_details() BEGIN SELECT bus_number, bus_plate_number, bus_type, capacity from bus WHERE capacity BETWEEN 35 AND 45; END;
```

phpMyAdmin

Recent Favorites

New

bbms

bbmsystem

Procedures

Tables

New

booking

bus

customer

driver

payment

schedule

user

bike_1

cms_vs_code

information_schema

lab10

mysql

performance_schema

phpmyadmin

railway

railways

test

Server: 127.0.0.1 » Database: bbmsystem

Structure SQL Search Query Export Import Operations Privileges

Show query box

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 1 (2 total, Query took 0.0004 seconds.)

CALL bus_details();

[Edit inline] [Edit] [Create PHP code]

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

bus_number	bus_plate_number	bus_type	capacity
1213548	KA23231	AC Sleeper	35
7896254	KA124579	Non-AC Sleeper	45

☐ Show all | Number of rows: 25 | Filter rows: Search this table

Query results operations

Print Copy to clipboard Create view

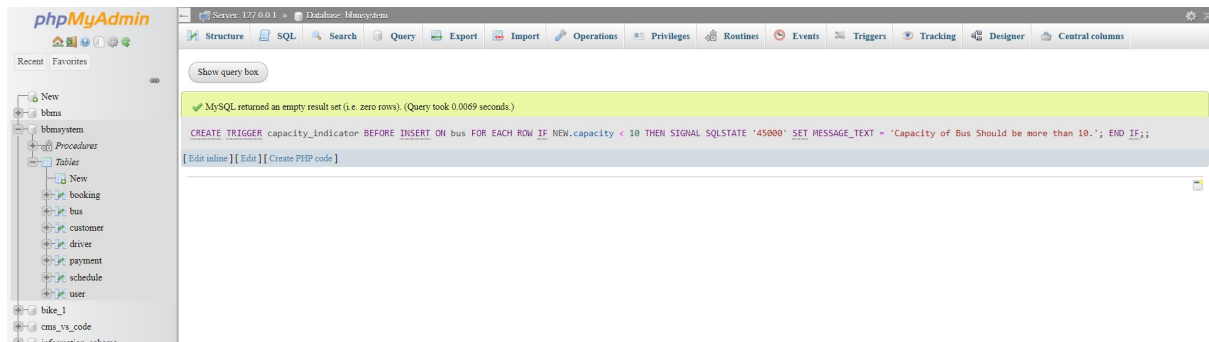
Bookmark this SQL query

Let every user access this bookmark

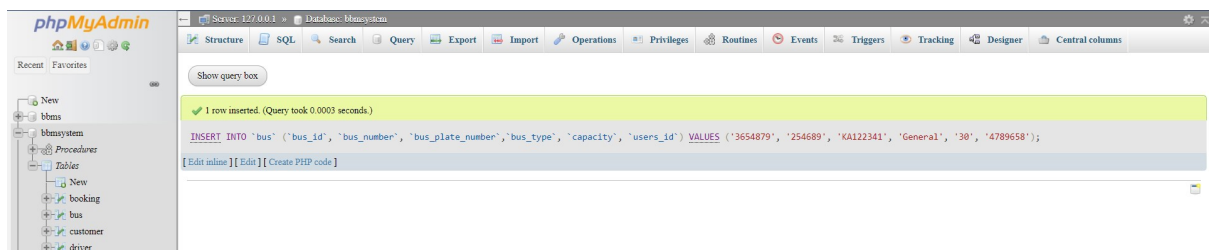
Triggers and Cursors

Trigger:

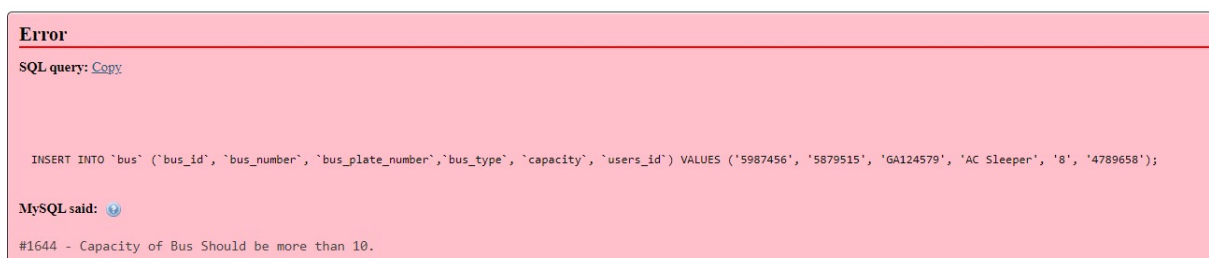
Trigger named capacity_indicator which is created for bus table we won't take bus capacity which are less than 10.



When we insert capacity more than 10.



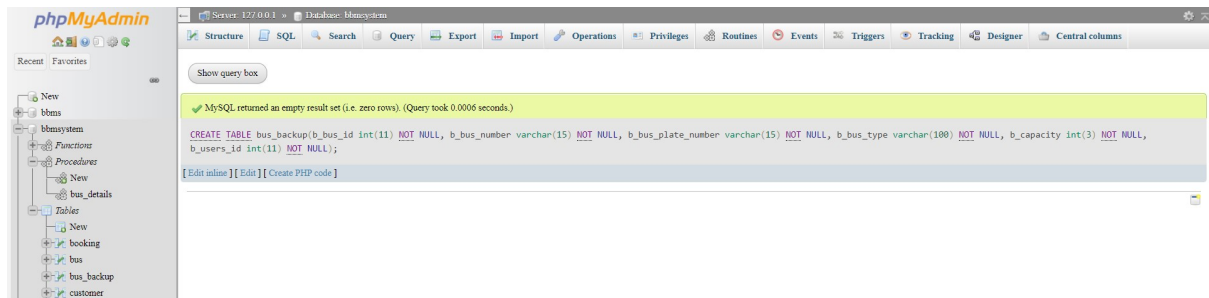
When we insert capacity less than 10.



Cursors:

Create a cursor to make a backup of bus table

Creating cursor table:



#Procedure with cursor

DELIMITER \$\$

CREATE DEFINER='root'@'localhost' PROCEDURE `backup_bus`()

BEGIN

 DECLARE done int DEFAULT 0;

 DECLARE bus_id int(11);

 DECLARE bus_number varchar(15);

 DECLARE bus_plate_number varchar(15);

 DECLARE bus_type varchar(100);

 DECLARE capacity int(3);

 DECLARE users_id int(11);

 DECLARE cur CURSOR FOR SELECT * FROM bus;

 DECLARE CONTINUE HANDLER FOR NOT Found SET done=1;

 OPEN cur;

label: LOOP

 FETCH cur INTO bus_id,bus_number,bus_plate_number,bus_type,capacity,users_id;

 INSERT INTO bus_backup
VALUES(bus_id,bus_number,bus_plate_number,bus_type,capacity,users_id);

 IF done=1 THEN LEAVE label;

```

END IF;

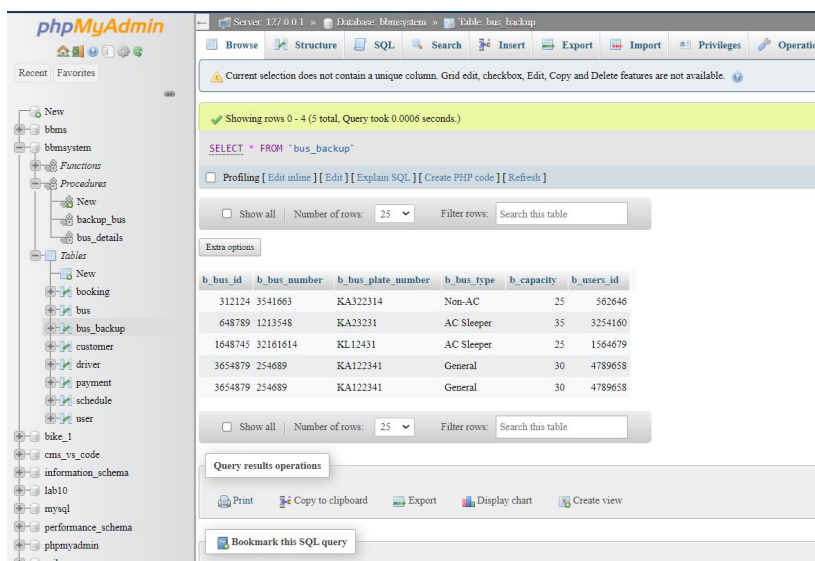
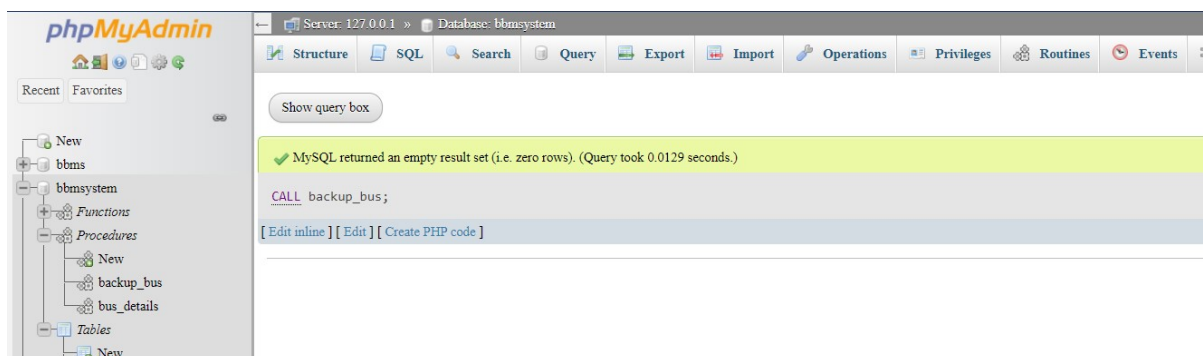
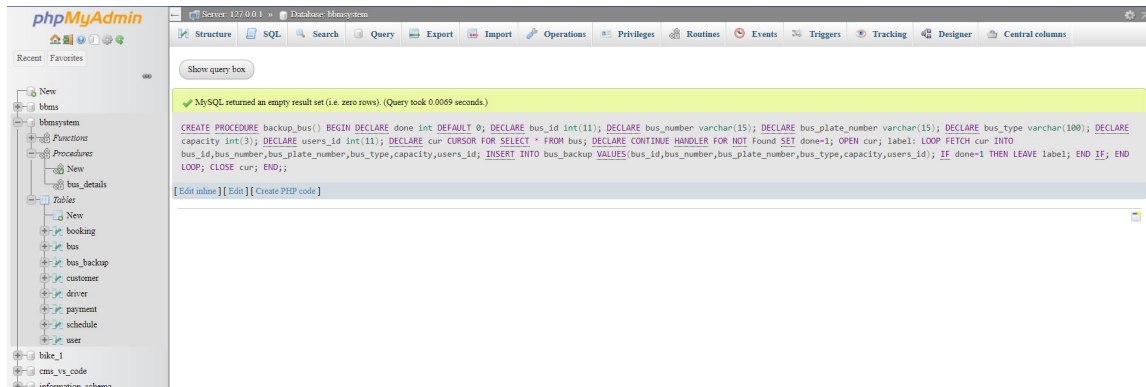
END LOOP;

CLOSE cur;

END$$

DELIMITER ;

```



Frontend Using Streamlit

User adding Page

×

SERVICES

USER

MENU

ADD_USER

BUS BOOKING MANAGEMENT SYSTEM

ENTER A USER DETAILS

User ID:

4789658.00

−

+

User Name:

Sagar

Full Name:

Sagar

User Password:

Sagar@123

Contact Number:

798441614.00

−

+

Account Category:

1

Email Address:

sagar123@gmail.com

Account Status:

Deactive

Add User Sagar

Successfully added user Sagar

Details of Bus Page

×

SERVICES

BUS

MENU

VIEW_BUS

BUS BOOKING MANAGEMENT SYSTEM

DETAILIES OF ALL BUS

View all Bus Details

	Bus ID	Bus Number	Plate Number	Type	Capacity	User ID
0	4561187	7896254	KA124579	Non-AC Sleeper	45	4789658
1	7862157	156467	KA214357	Non-AC	57	1564679

Made with Streamlit

Driver Delete Page

SERVICES

DRIVER

MENU

DELETE_DRIVER

BUS BOOKING MANAGEMENT SYSTEM

SELECT DRIVER TO DELETE

Current_driver

	Driver ID	Driver Name	Contact	User ID
0	165461	Darshan	6106466	4789658
1	781647	Shreyas	71066447	1564679

Driver to edit

781647

Do you want to delete '781647'?

Delete driver

Updated data

	Driver ID	Driver Name	Contact	User ID
0	165461	Darshan	6106466	4789658
1	781647	Shreyas	71066447	1564679

Updating Customer Page

SERVICES

CUSTOMER

MENU

UPDATE_CUSTOMER

BUS BOOKING MANAGEMENT SYSTEM

SELECT CUSTOMER TO UPDATE

Current Customer

	customer_id	customer_name	customer_contact	customer_email	username
0	4547987	hari	79461648	hari123@gmail.com	Hari
1	6014864	Sharu	244606747	sharu@gmail.com	Sharu

Customer to edit

4547987

Customer ID

4547987

User Name

Hari

Customer Name

hari

Customer Password

Hari@123

Customer Contact

79461648

account status

Done

Customer Email

hari123@gmail.com

Users ID

562646

Update Customer

Updated data

	customer_id	customer_name	customer_contact	customer_email	username
0	4547987	hari	79461648	hari123@gmail.com	Hari

Query Page

×

SERVICES

QUERY BOX ▾

BUS BOOKING MANAGEMENT SYSTEM

Enter the query here:

Submit