PES UNIVERSITY
Department of Computer Science & Engineering

DBMS - UE20CS301

Mini Project

Title of the Project

CLOTHING STORE MANAGEMENT SYSTEM

Submitted to:

Dr. Geetha D

Associate Professor

Submitted by:

Name : Suhas T J

SRN : PES2UG20CS355

V Semester Section F

Table of Contents

Sl.No	Title	Page No
1	A short description about the project and scope	1
2	ER Diagram	2
3	Relational Schema	3
4	DDL statements to build the database and populating the database	4 - 6
5	JOIN queries	7 - 11
6	Aggregate Functions	12 - 16
7	SET Operations	17 - 22
8	Functions and Procedure	23 - 26
9	Triggers and cursors	27 – 32
10	Developing a front end	33 – 54
11	Conclusion	55
12	References	56

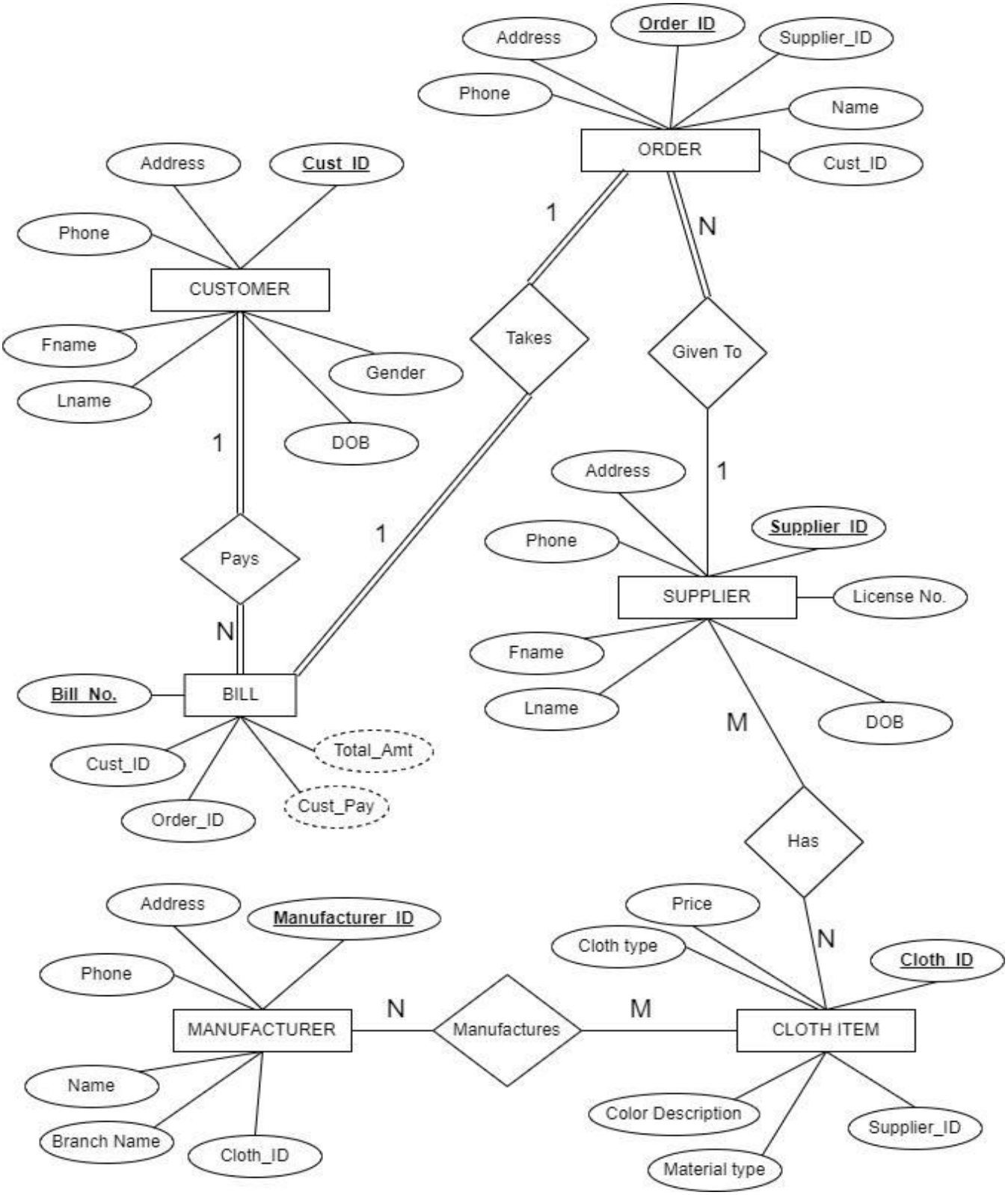
1. Short Description and Scope of the Project

The Cloth Store Management System is a simple MySQL project that aims to help a small-scale clothing or apparel business/shop to manage its sales through a simple online portal.
It gives a clear picture of the suppliers of different types of clothes and strictly evaluates the bill per customer. The admin can easily view the transactions and dealings through the dynamically designed database. The customers enjoy the facility of being able to order the cloth of their choice from a vast collection of apparels at a glance.

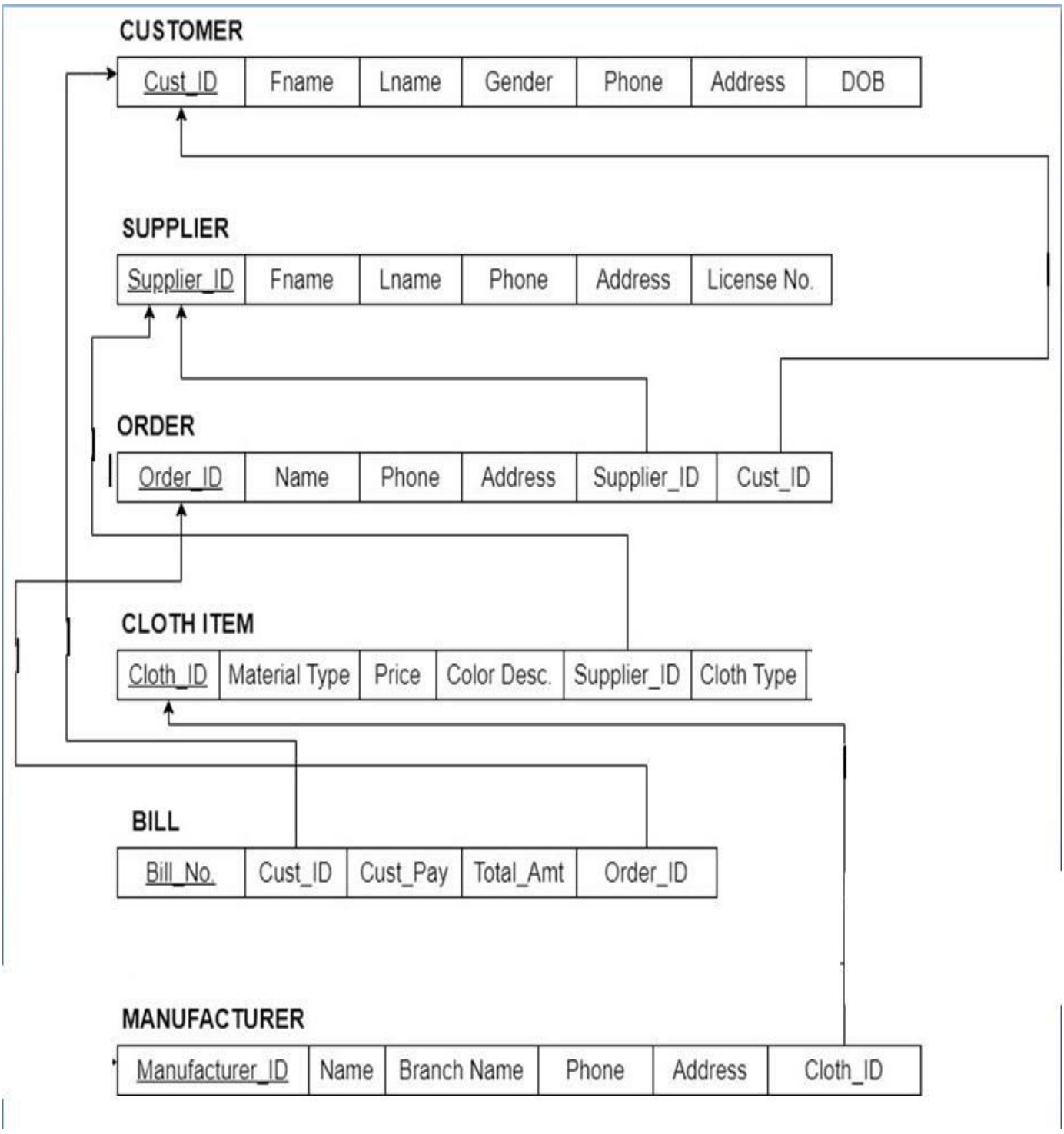
SCOPE OF THIS PROJECT

- This project adopts a general database schema which makes it portable for any clothing store present in any locality.
- It aims to give the online support to these local suppliers of clothes and make their shop reach out to a larger audience.
- The billings and dealings are monitored and recorded thus preventing any kind of frauds whatsoever.
- Users get the benefit to shop online right from the comfort of their home.
- The suppliers also have the option to order clothes from different manufacturers thus expanding their potential.

2. ER Diagram


Address
Order_ID
Supplier_ID
Phone
Name
ORDER
Cust_ID
Address
Cust_ID
Phone
CUSTOMER
1
N
Takes
Given To
Fname
Gender
Lname
DOB
1
1
Address
Pays
1
Supplier_ID
Phone
SUPPLIER
License No.
N
Fname
Lname
M
DOB
Bill_No.
BILL
Total_Amt
Cust_ID
Order_ID
Cust_Pay
Has
Address
Manufacturer_ID
Price
N
Cloth type
Cloth_ID
Phone
N
M
MANUFACTURER
Manufactures
CLOTH ITEM
Name
Color Description
Supplier_ID
Branch Name
Cloth_ID
Material type

3. Relational Schema



4. DDL statements - Building the database

```
CREATE TABLE `bill` (
  `bill_num` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `order_id` int(11) NOT NULL,
  `Amt_paid` int(11) NOT NULL,
  `total_amt` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `clothes` (
  `cloth_id` int(30) NOT NULL,
  `item_code` varchar(100) NOT NULL,
  `name` varchar(200) NOT NULL,
  `description` text NOT NULL,
  `size` varchar(10) NOT NULL,
  `Supp_ID` int(11) DEFAULT NULL,
  `cost` float NOT NULL,
  `date_created` datetime NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `order_c` (
  `Order_ID` int(11) NOT NULL COMMENT 'Primary Key',
  `Supp_ID` int(11) NOT NULL,
  `Order_qty` int(11) NOT NULL,
  `Order_date` date NOT NULL,
  `user_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `suppliers` (
  `Supp_ID` int(30) NOT NULL,
  `fname` varchar(20) NOT NULL,
  `lname` varchar(20) NOT NULL,
  `address` text NOT NULL,
  `contact` varchar(50) NOT NULL,
  `date_created` datetime NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `users` (
  `user_id` int(30) NOT NULL,
  `name` varchar(20) NOT NULL,
  `username` varchar(200) NOT NULL,
  `password` text NOT NULL,
  `Gender` varchar(10) NOT NULL,
  `contact` int(10) NOT NULL,
  `Address` varchar(50) NOT NULL,
  `Amt_paid` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `manufacturer` (
  `manufacturer_ID` int(11) NOT NULL,
  `name` varchar(11) NOT NULL,
  `phone` int(11) NOT NULL,
  `address` varchar(11) NOT NULL,
  `cloth_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```sql
ALTER TABLE `bill`
  ADD PRIMARY KEY (`bill_num`),
  ADD KEY `fk1` (`user_id`),
  ADD KEY `fk2` (`order_id`);

ALTER TABLE `clothes`
  ADD PRIMARY KEY (`cloth_id`),
  ADD KEY `Supp_ID` (`Supp_ID`);

ALTER TABLE `order_c`
  ADD PRIMARY KEY (`Order_ID`),
  ADD KEY `user_id` (`user_id`),
  ADD KEY `given_to` (`Supp_ID`);

ALTER TABLE `suppliers`
  ADD PRIMARY KEY (`Supp_ID`);

ALTER TABLE `users`
  ADD PRIMARY KEY (`user_id`);

ALTER TABLE `clothes`
  MODIFY `cloth_id` int(30) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=908;

ALTER TABLE `suppliers`
  MODIFY `Supp_ID` int(30) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7566;

ALTER TABLE `users`
  MODIFY `user_id` int(30) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=904;

ALTER TABLE `bill`
  ADD CONSTRAINT `fk1` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`),
  ADD CONSTRAINT `fk2` FOREIGN KEY (`order_id`) REFERENCES `order_c` (`Order_ID`);

ALTER TABLE `clothes`
  ADD CONSTRAINT `clothes_ibfk_1` FOREIGN KEY (`Supp_ID`) REFERENCES `suppliers` (`Supp_ID`);

ALTER TABLE `order_c`
  ADD CONSTRAINT `given_to` FOREIGN KEY (`Supp_ID`) REFERENCES `suppliers` (`Supp_ID`),
  ADD CONSTRAINT `order_c_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`),
  ADD CONSTRAINT `order_c_ibfk_2` FOREIGN KEY (`user_id`) REFERENCES `users` (`user_id`);
```

5. Populating the Database

```
INSERT INTO `bill` (`bill_num`, `user_id`, `order_id`, `Amt_paid`, `total_amt`) VALUES
(216, 152, 109, 2134, 5000),
(258, 112, 134, 6785, 11000),
(312, 512, 134, 5678, 10000),
(478, 210, 329, 4319, 8000),
(548, 112, 867, 3489, 12000),
(562, 131, 109, 2893, 5097),
(867, 104, 356, 2346, 4000),
(965, 210, 290, 3244, 6000);

INSERT INTO `clothes` (`cloth_id`, `item_code`, `name`, `description`, `size`, `Supp_ID`, `cost`, `date_created`) VALUES
(2, '774420957142', 'Cotton T-shirt', 'Type : T-shirt\r\nColor : Yellow and Radium\r\nCompany : Levi Strauss & Co.', 'L', 234, 300, '2022-
(134, '904563', 'Unknown apparel', 'Color : Blood red Cotton', 'XL', 234, 3570, '2022-11-12 00:00:00'),
(156, '298644', 'SHORTS', 'Color : Brown\r\nDesign : Codra', 'S', 7564, 150, '2022-11-04 10:55:56'),
(214, '216908', 'Salwar Kameez', 'Color : Yellow and blue\r\nPattern : Banaras\r\n', 'XL', 234, 576, '2022-11-04 10:52:07'),
(231, '123908', 'Jeggings', 'Color : Black\r\nPatter : Desi', 'M', 2167, 213, '2022-11-02 10:52:07'),
(278, '905673', 'Track Pants ', 'Color : Blue\r\nMaterial : Cotton', 'L', 2167, 420, '2022-11-02 10:55:56'),
(289, '298641', 'BLAZER', 'Type:Hard\nColor:Black and dark blue mix', 'M', 234, 6079, '2022-03-04 09:51:02'),
(367, '435622', 'Formal Pants', 'Mens wear deluxe', 'L', 214, 552, '2022-11-05 09:24:56'),
(427, '784533', '3/4th Shorts', 'Color : Military green\\nMaterial type : Cotton+polyester', 'XL', 214, 2000, '2022-11-16 00:00:00'),
(459, '56347832', 'Black T-shirt', 'Colar blue', 'L', 214, 500, '2022-11-09 09:21:11'),
(907, '568432', 'Dhoti', 'Color : Bright Gold', 'L', 7564, 500, '2022-11-19 00:00:00');

INSERT INTO `order_c` (`Order_ID`, `Supp_ID`, `Order_qty`, `Order_date`, `user_id`) VALUES
(109, 2582, 90, '2022-11-10', 152),
(134, 1132, 80, '2022-11-14', 104),
(290, 7564, 102, '2022-11-09', 112),
(329, 2167, 47, '2022-11-08', 512),
(356, 2167, 106, '2022-11-06', 18),
(867, 1132, 70, '2022-11-05', 131);
```

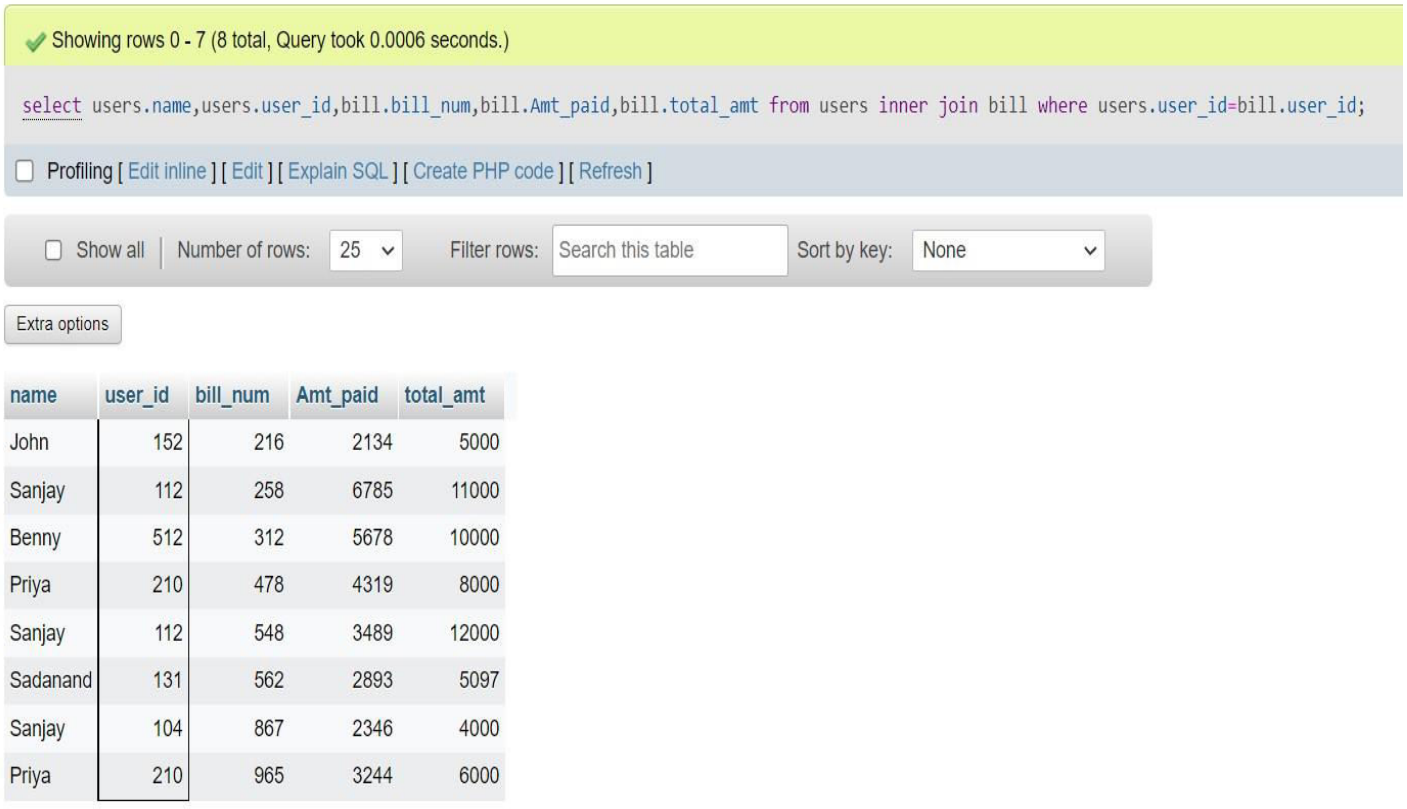
```
INSERT INTO `users` (`user_id`, `name`, `username`, `password`, `Gender`, `contact`, `Address`, `Amt_paid`) VALUES
(1, 'admin', 'admin', '0192023a7bbd73250516f069df18b500', '', 0, '', 0),
(18, 'Sheetal', 'shetal', 'shetal125', 'Female', 2147483647, 'Sakarnagar', 4213),
(32, 'Skanda', 'skunk', 'list', 'Male', 2147483647, 'Mumbai', 3200),
(104, 'Sanjay', 'sanjay', '12345', 'Male', 2147483647, 'RR NAGAR', 4500),
(112, 'Sanjay', 'sanjay', '12345', 'Male', 2147483647, 'RR NAGAR', 4500),
(131, 'Shanbogh', 'sandy', 'prep123', 'Female', 2147483647, 'Ram NAGAR', 5500),
(152, 'John', 'john123', '8756j', 'Male', 2147483647, 'JP NAGAR', 42134),
(186, 'Kali', 'kalyani123', 'kr@12345', 'Female', 2147483647, 'Bengal', 3200),
(210, 'Priya', 'priyaa', '12125', 'Female', 2147483647, 'RR NAGAR', 4420),
(512, 'Benny', 'ben', 'benny@12345', 'Male', 2147483647, 'Jakkur', 45231);

INSERT INTO `suppliers` (`Supp_ID`, `fname`, `lname`, `address`, `contact`, `date_created`) VALUES
(214, 'Balushahi', 'Fashions', 'Coimbatore', '8905673207', '2022-11-09 00:00:00'),
(234, 'Saiyan', 'Stores', 'JAI NAGAR', '9876523409', '2020-11-05 09:33:01'),
(1132, 'Sunnyside', 'Fashions', 'CBD St., EFG City', '7411324190', '2022-11-01 09:33:26'),
(2167, 'Arrow', 'Wears', 'JP nagar', '8750982143', '2022-11-03 09:34:21'),
(2582, 'SunSide', 'Fashions', 'Balajinagar', '8956731295', '2022-10-13 00:00:00'),
(7564, 'Pranjal', 'Stores', 'Bhairavanagar', '8747808789', '2022-11-02 09:34:50');

INSERT INTO `manufacturer` (`manufacturer_ID`, `name`, `phone`, `address`, `cloth_id`) VALUES
(185, 'RUDRA Texti', 9896509, 'Thiruvanant', 134),
(368, 'Kalyan Text', 78695768, 'Pondicherry', 278),
(458, 'Varghese Te', 897590687, 'Hyderabad', 214),
(509, 'GokulDas Ex', 996578437, 'Madurai', 231);
```

6. Join Queries

Showcase at least 4 join queries
Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

1. Taking inner join (common field between table **users** and table **bill** being user_id) so it displays the records correspondingly.

Select users.name, users.user_id, bill.bill_num, bill.Amt_paid, bill.total_amt from users inner join bill where users.user_id=bill.user_id;



name	user_id	bill_num	Amt_paid	total_amt
John	152	216	2134	5000
Sanjay	112	258	6785	11000
Benny	512	312	5678	10000
Priya	210	478	4319	8000
Sanjay	112	548	3489	12000
Sadanand	131	562	2893	5097
Sanjay	104	867	2346	4000
Priya	210	965	3244	6000

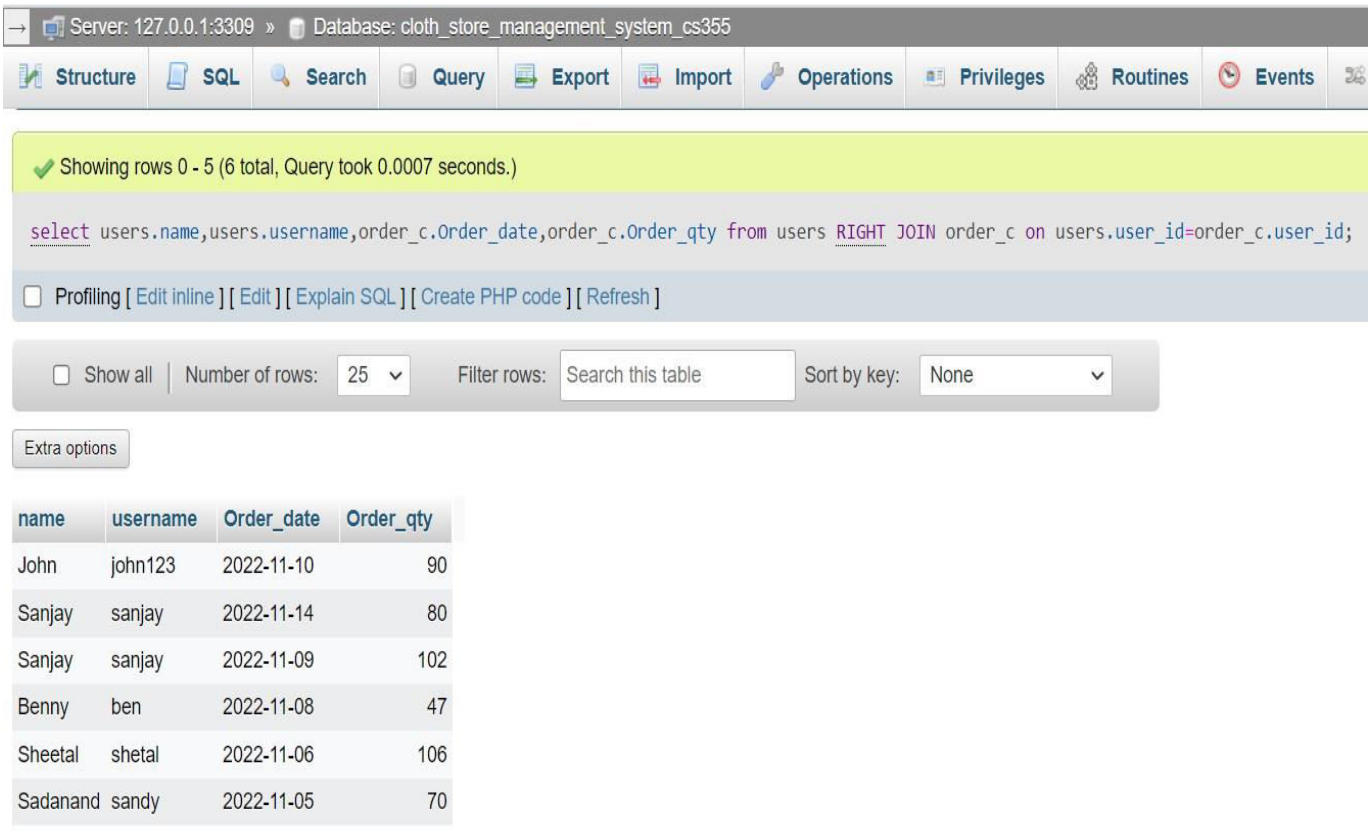
2. Returning all the rows from the table **users (left** table) and the corresponding matching rows in the table **order (right** table). NULL is displayed for those not matching.

Select users.name, users.username, order_c.Order_date, order_c.Order_qty from users left JOIN order_c on users.user_id = order_c.user_id;

name	username	Order_date	Order_qty
admin	admin	*NULL*	*NULL*
Sheetal	shetal	2022-11-06	106
Sanjay	sanjay	2022-11-14	80
Sanjay	sanjay	2022-11-09	102
Sadanand	sandy	2022-11-05	70
John	john123	2022-11-10	90
Priya	priyaa	*NULL*	*NULL*
Benny	ben	2022-11-08	47

3. Returning all the rows from the table **order_c (right** table) and the corresponding matching rows in the table **users (left** table). NULL is displayed for those not matching.

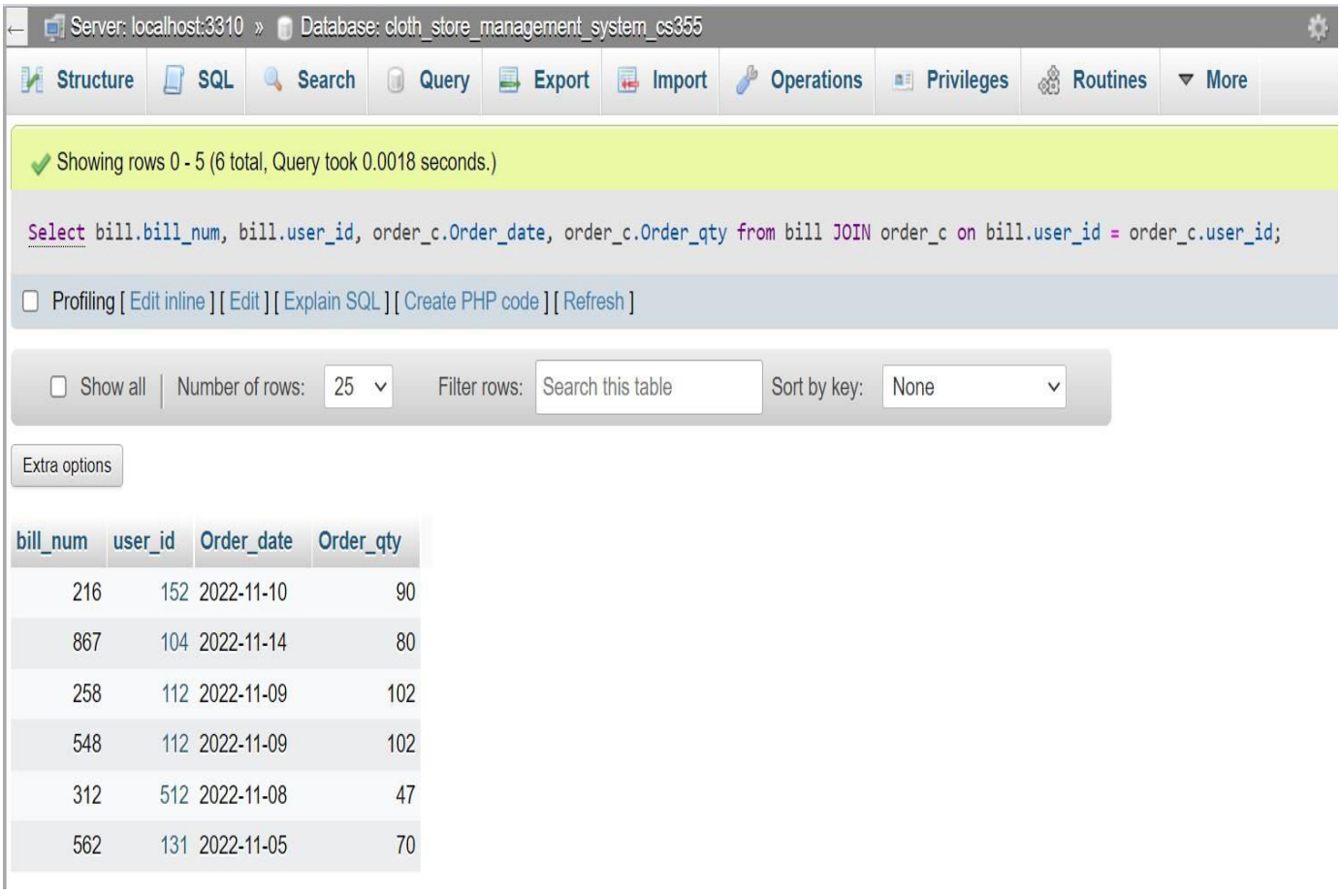
Select users.name, users.username, order_c.Order_date, order_c.Order_qty from users RIGHT JOIN order_c on users.user_id = order_c.user_id;



name	username	Order_date	Order_qty
John	john123	2022-11-10	90
Sanjay	sanjay	2022-11-14	80
Sanjay	sanjay	2022-11-09	102
Benny	ben	2022-11-08	47
Sheetal	shetal	2022-11-06	106
Sadanand	sandy	2022-11-05	70

4. Displays the combined rows from the tables **bill** and **order_c** based on the related column user_id.

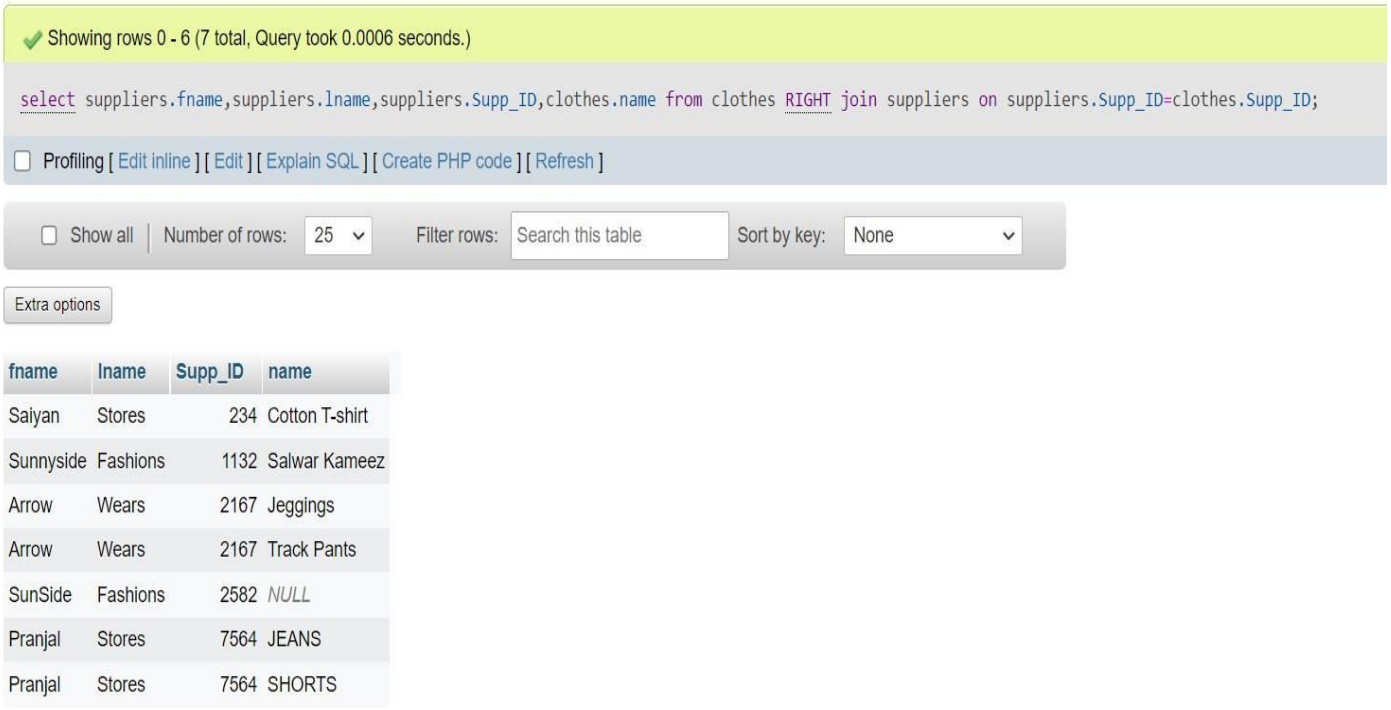
Select bill.bill_num, bill.user_id, order_c.Order_date, order_c.Order_qty from bill JOIN order_c on bill.user_id = order_c.user_id;



bill_num	user_id	Order_date	Order_qty
216	152	2022-11-10	90
867	104	2022-11-14	80
258	112	2022-11-09	102
548	112	2022-11-09	102
312	512	2022-11-08	47
562	131	2022-11-05	70

5. Returning all the rows from the table **suppliers (right** table) and the corresponding matching rows in the table **clothes (left** table). NULL is displayed for those not matching.

Select suppliers.fname, suppliers.lname, suppliers.Supp_ID, clothes.name from clothes RIGHT JOIN suppliers on suppliers.Supp_id = clothes.Supp_ID;



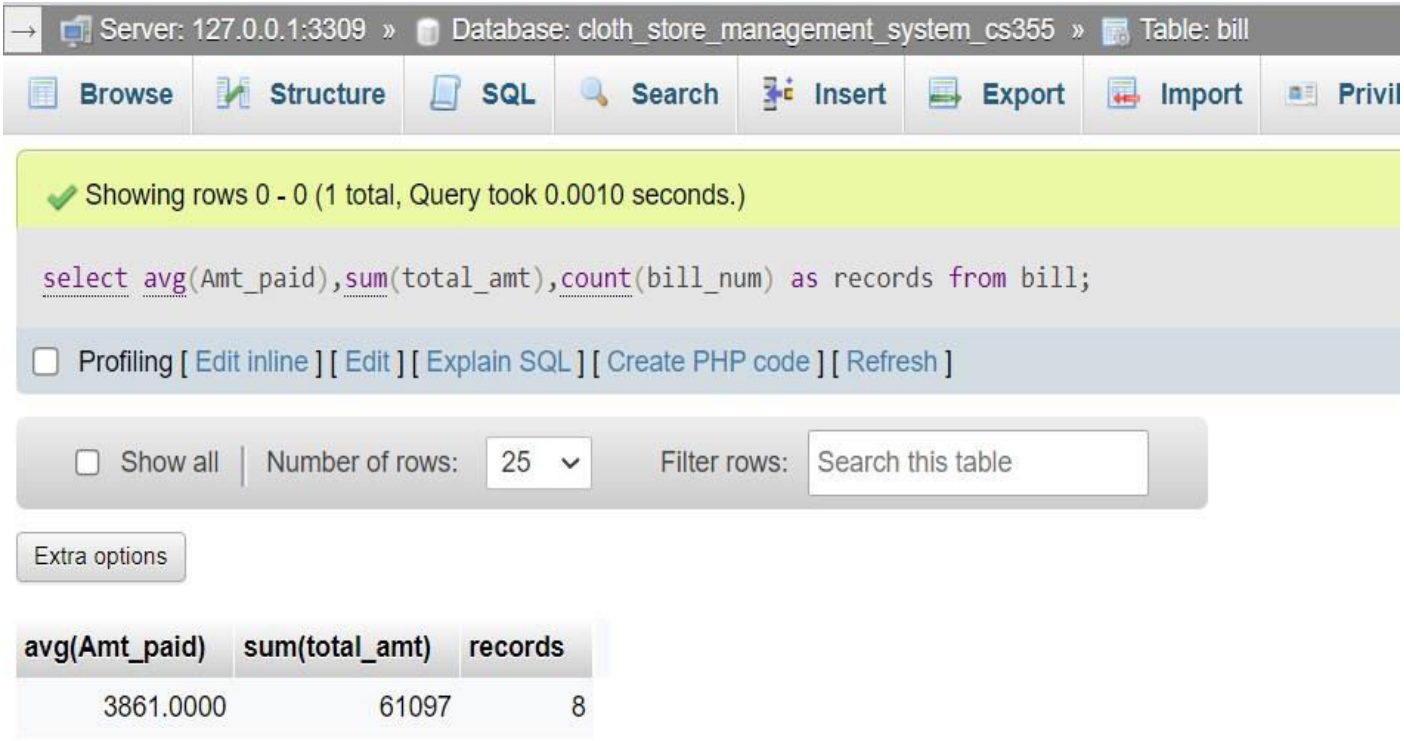
fname	lname	Supp_ID	name
Saiyan	Stores	234	Cotton T-shirt
Sunnyside	Fashions	1132	Salwar Kameez
Arrow	Wears	2167	Jeggings
Arrow	Wears	2167	Track Pants
SunSide	Fashions	2582	*NULL*
Pranjal	Stores	7564	JEANS
Pranjal	Stores	7564	SHORTS

7. Aggregate Functions

Showcase at least 4 Aggregate function queries
Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results.

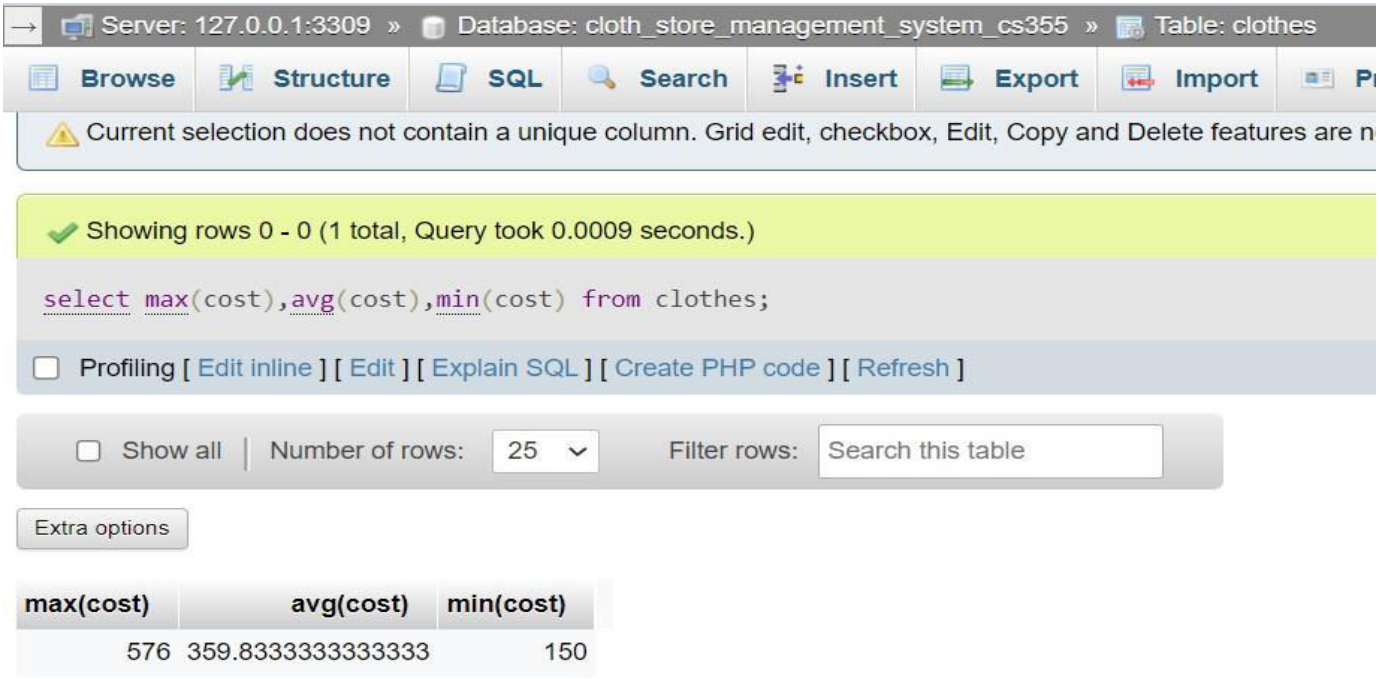
1. Fetching the average of the amount paid, sum of total amount paid and counting the total number of bills from the table **bill**.

Select avg(Amt_paid),sum(total_amt),count(bill_num) as records from bill;



2. Calculating the average, minimum and maximum cost in the table **clothes.**

   Select max(cost), avg(cost), min(cost) from clothes;

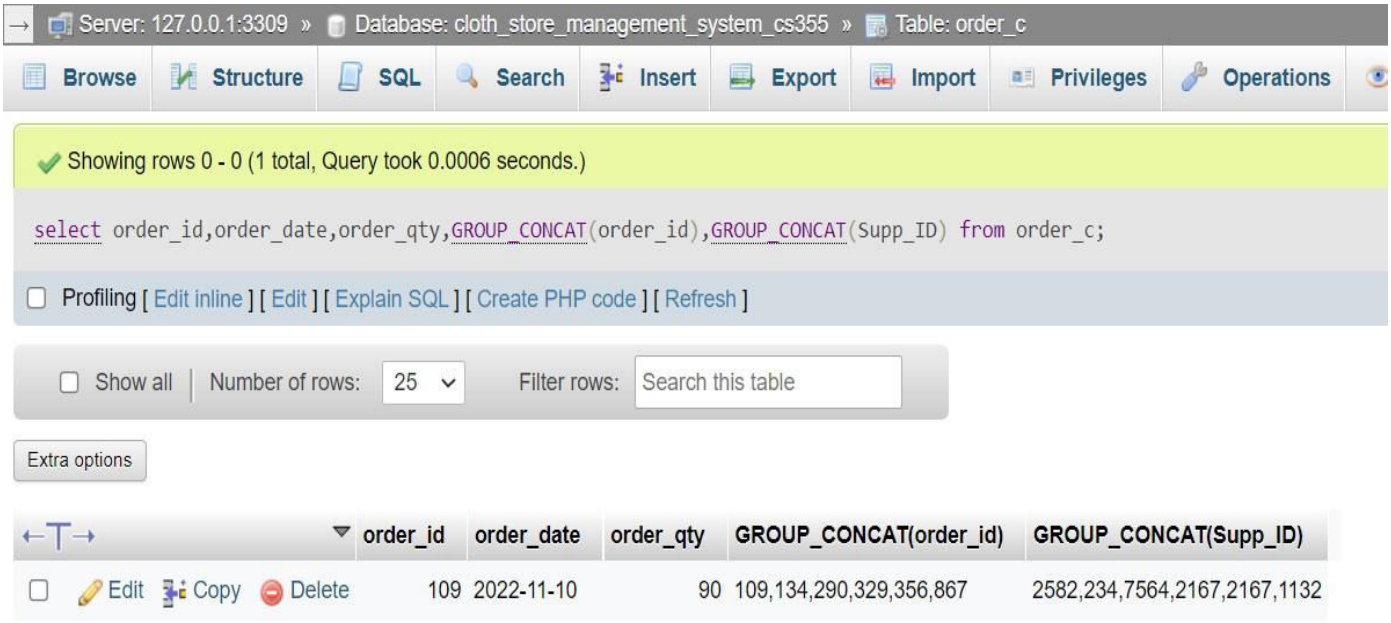


3. Calculating the average, sum of all and counting the total number of order quantity from the table **order_c.**

   Select avg(order_qty), sum(order_qty), count(order_id) from order_c;

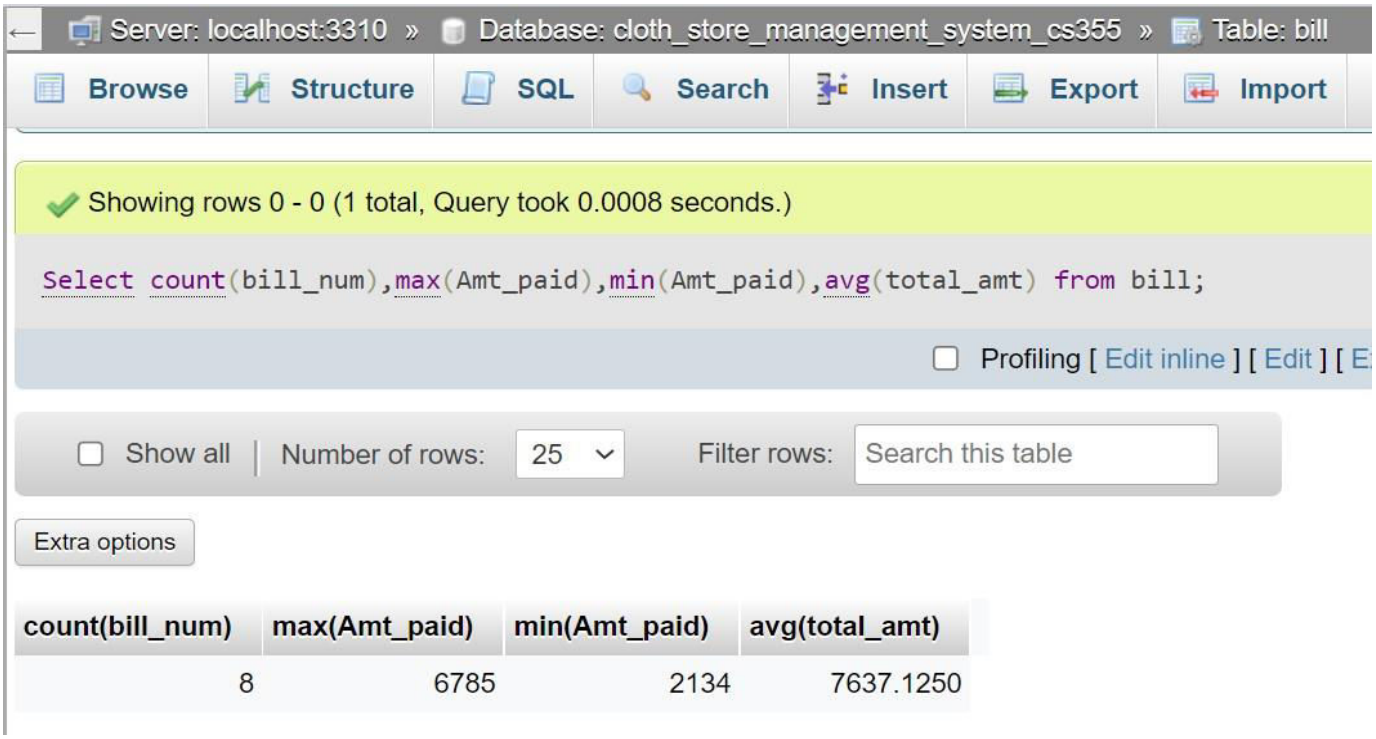
4. Getting all the order_id values and Supp_ID values from the table **order_c**.

Select order_id, order_date, order_qty, GROUP_CONCAT(order_id), GROUP_CONCAT(Supp_ID) from order_c;



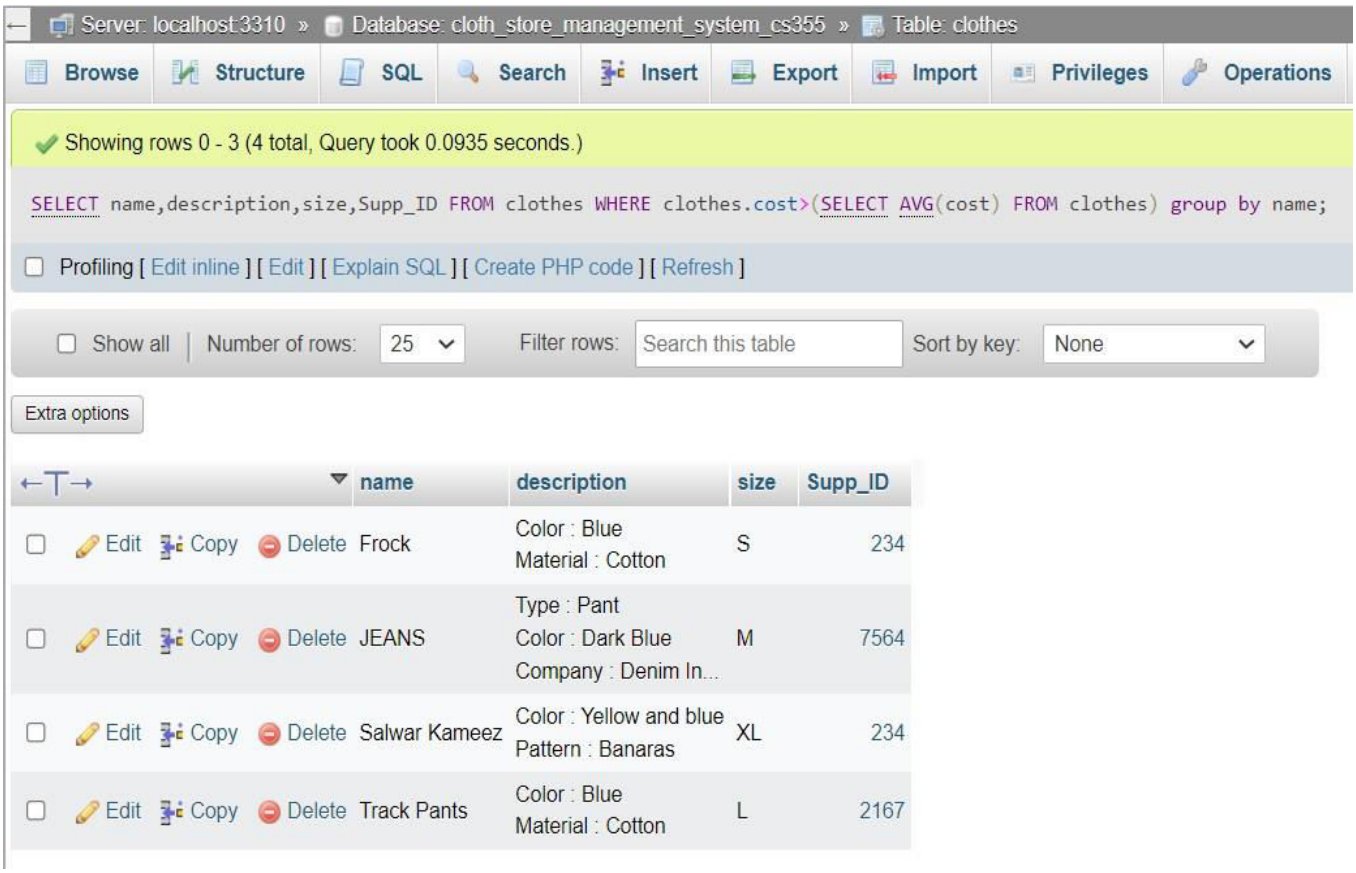
5. Calculating the average, minimum and maximum Amount paid in the table **users.**

Select count(bill_num), max(Amt_paid), min(Amt_paid), avg(total_amt) from users;



6. Displaying the names, sizes, description and the associated supplier_id of all the clothes whose cost is above average. It is displayed in ascending order of name.

Select name, description, size, Supp_ID FROM clothes WHERE clothes.cost > (SELECT avg(cost) FROM clothes) group by name;

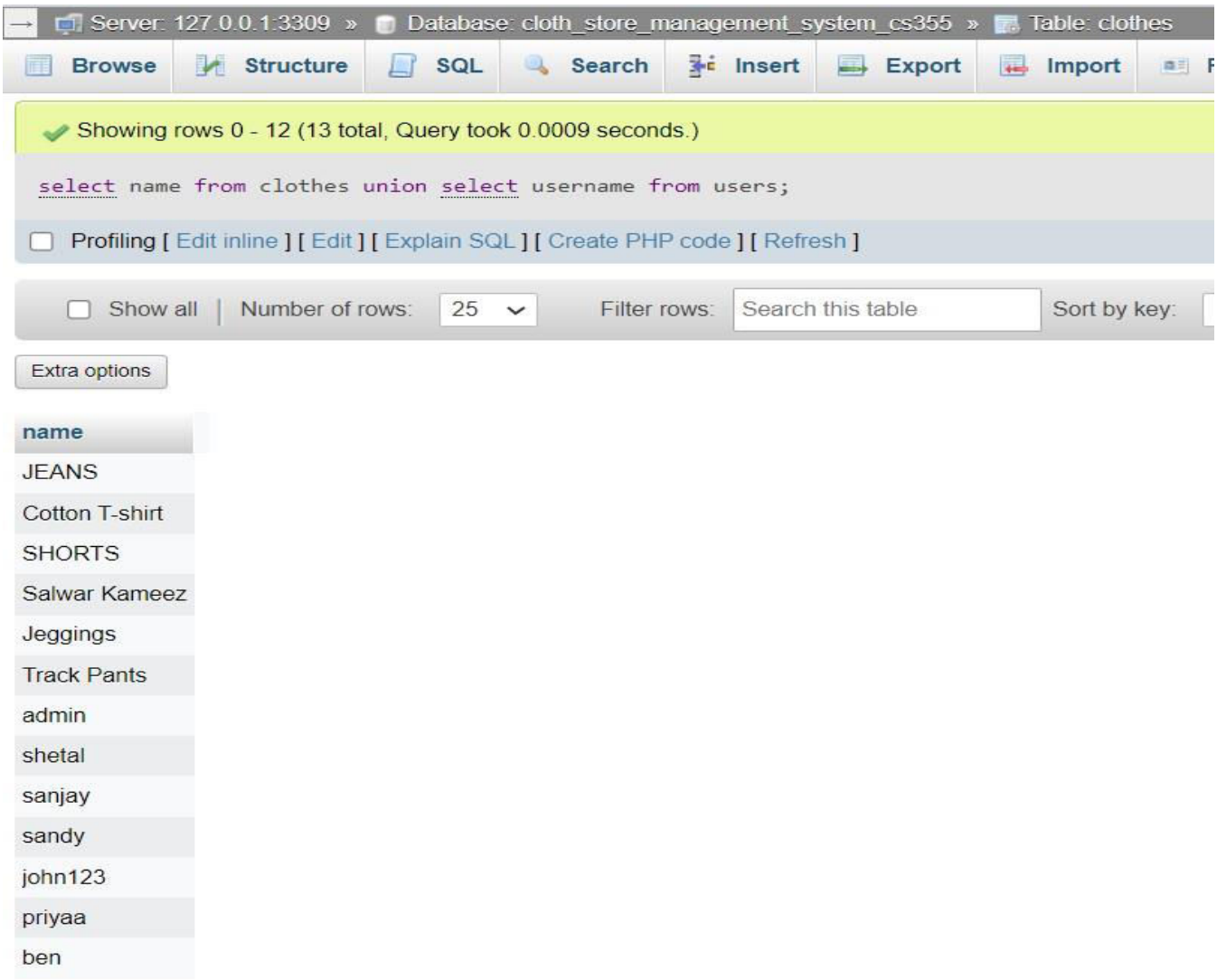


8. Set Operations

Showcase at least 4 Set Operations queries
Write the query in English Language, Show the equivalent SQL statement and also ascreenshot of the query and the results

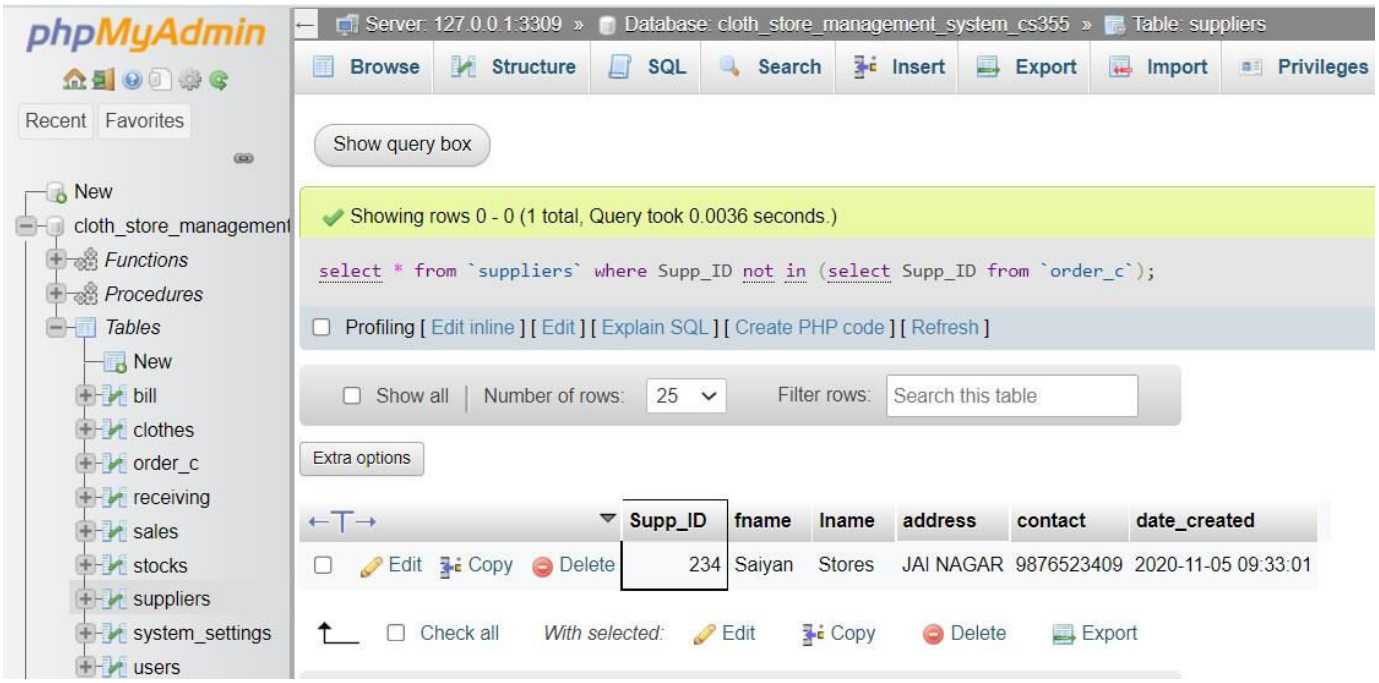
1. Displaying the column which contains the values after doing union between all the names in table **clothes** and the usernames in table **users**.

   Select name from clothes union select username from users;



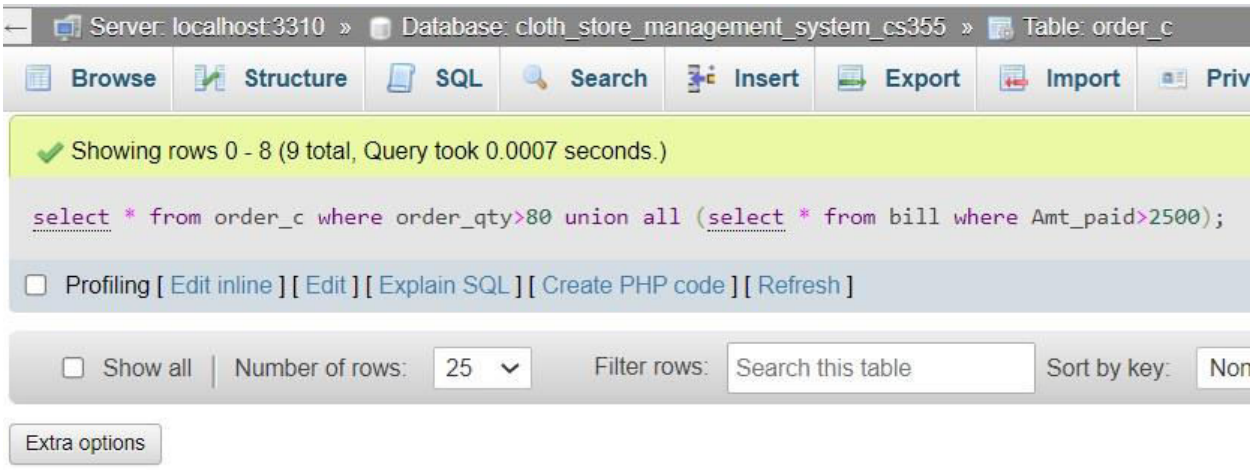
2. Displaying all the records in table **suppliers** which are not in table **orders.** This means the displayed supplier doesn't have any orders currently.

select * from suppliers where Supp_ID not in (select Supp_ID from order_c);



3. Displaying all the records from table **orders** whose quantity is more than 80 or amount paid in bill is more than 2500.

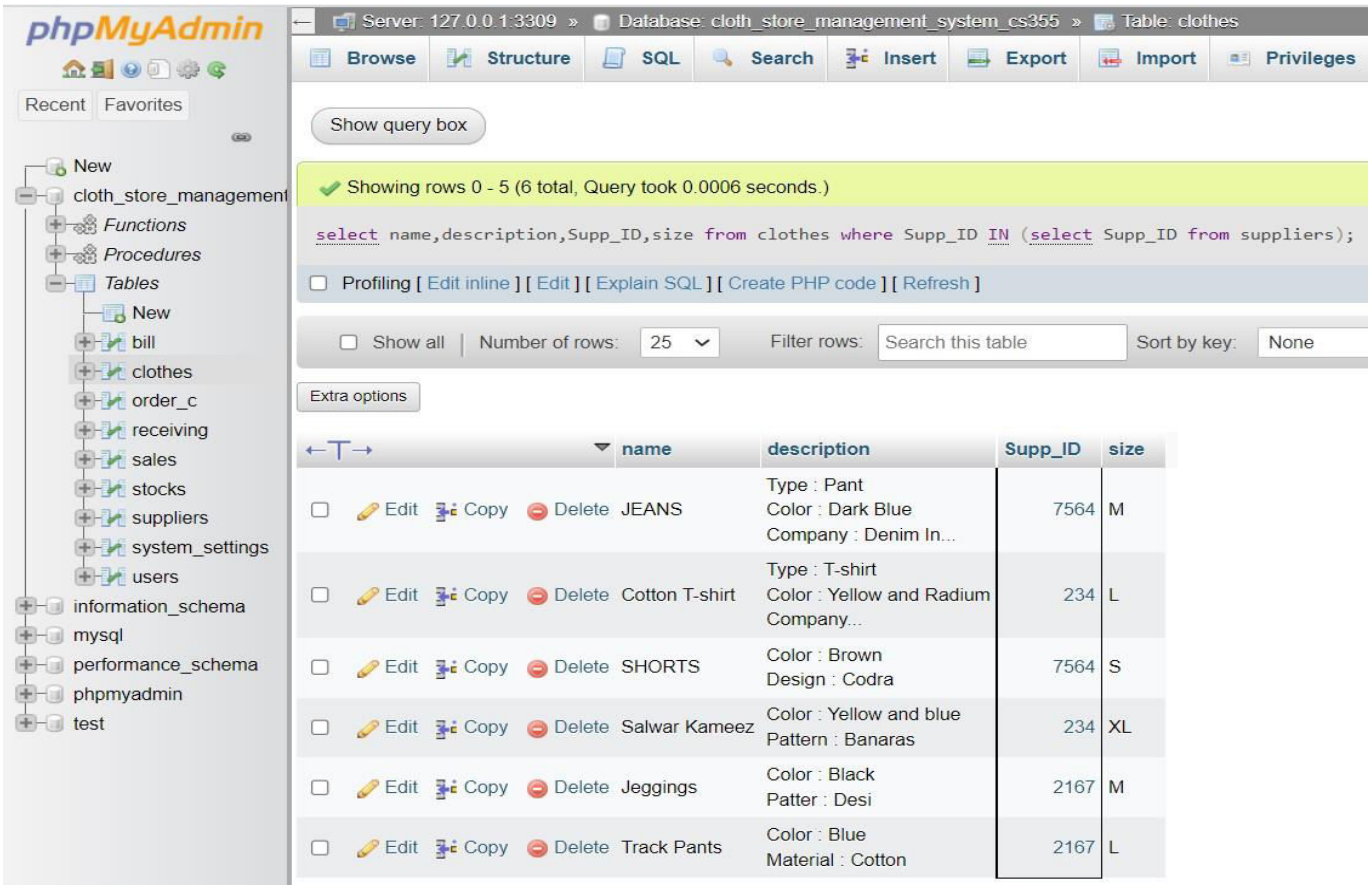
select * from order_c where order_qty>80 union all (select * from bill where Amt_paid>2500);



Order_ID	Supp_ID	Order_qty	Order_date	user_id
109	2582	90	2022-11-10	152
290	7564	102	2022-11-09	112
356	2167	106	2022-11-06	18
258	112	134	6785	11000
312	512	134	5678	10000
478	210	329	4319	8000
548	112	867	3489	12000
562	131	109	2893	5097
965	210	290	3244	6000

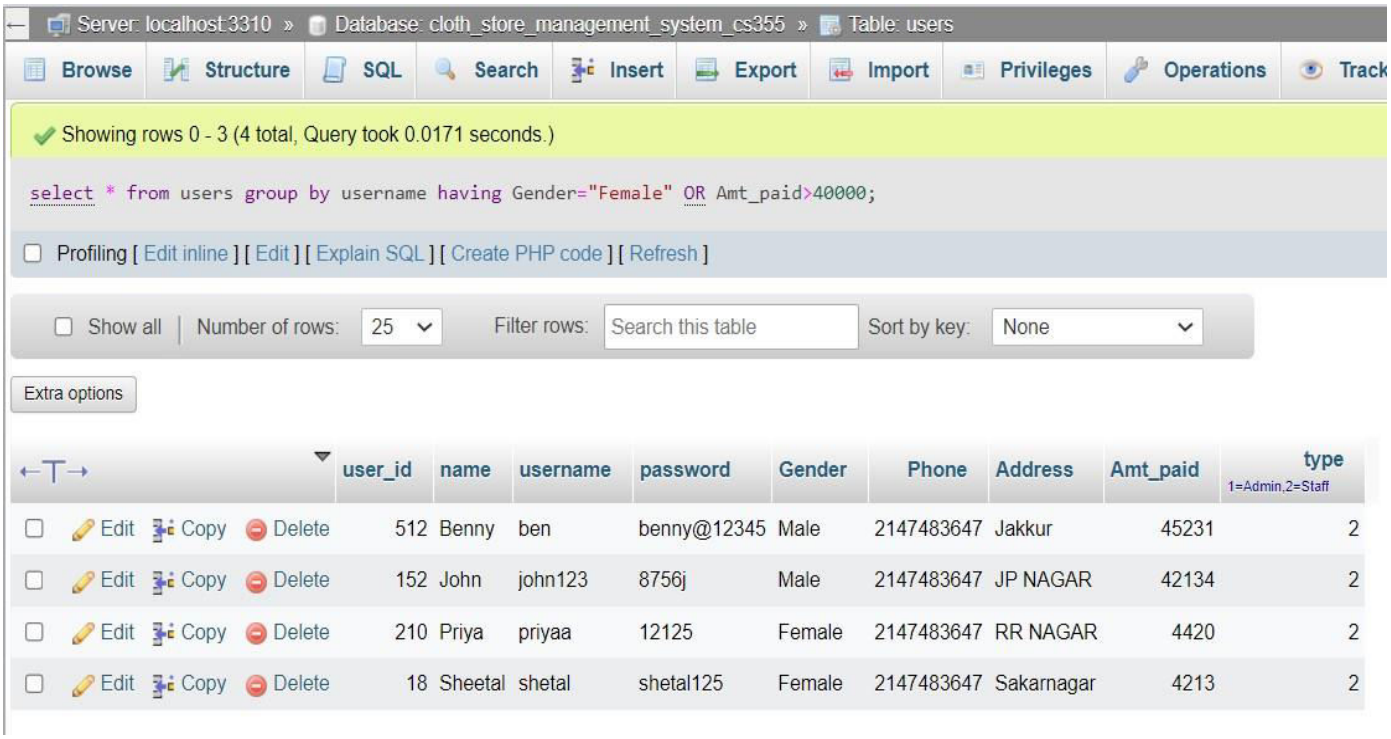
4. Displaying only the names, description, Supp_ID and size of those clothes whose Supp_ID is also present in table **suppliers.** Finding the records who have a defined supplier.

select name, description, Supp_ID, size from clothes where Supp_ID IN (select Supp_ID from suppliers);



5. Displaying all the records from table **users** who are females or have paid an amount more than 40000.

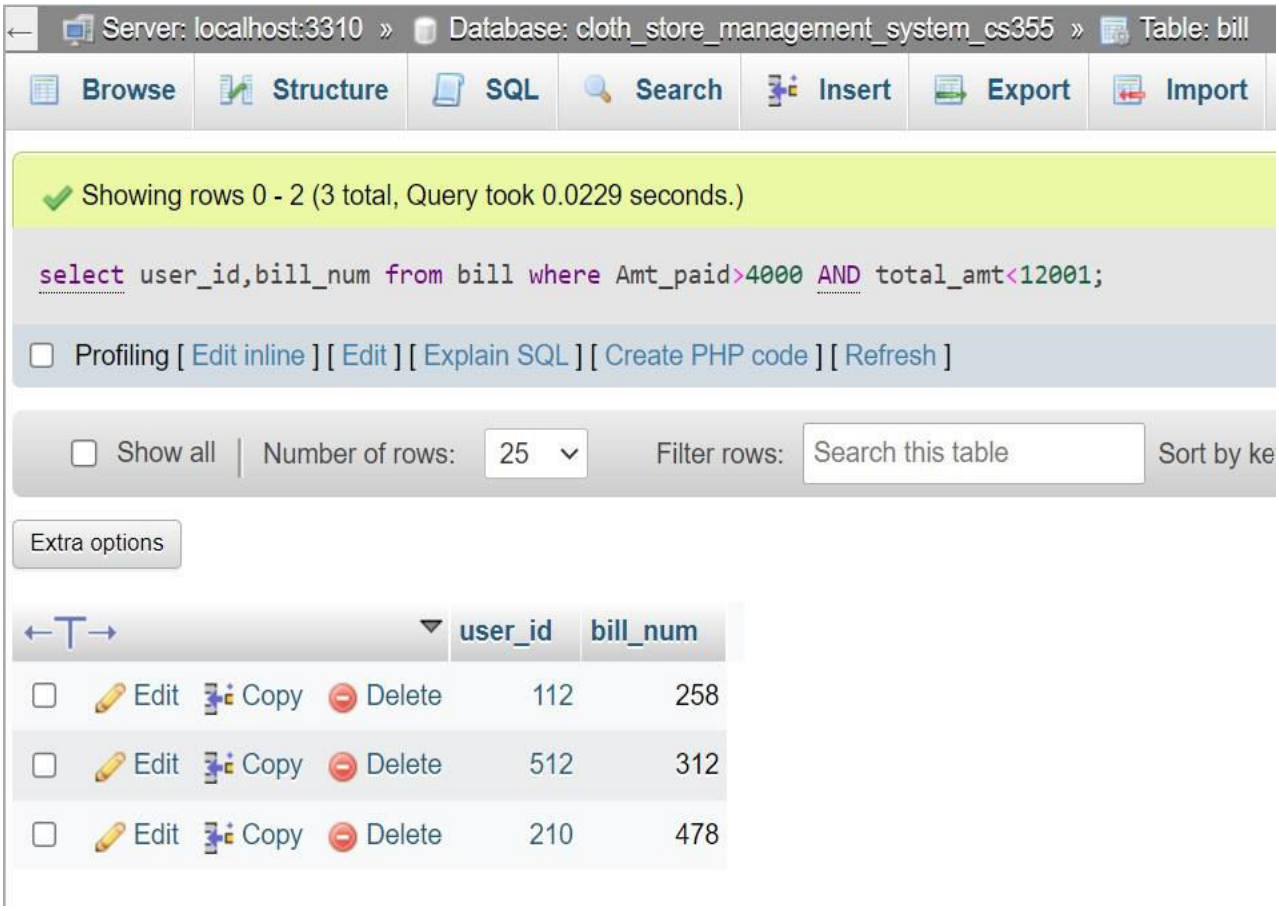
select * from users group by username having Gender = “Female” OR Amt_paid>40000;



	user_id	name	username	password	Gender	Phone	Address	Amt_paid	type (1=Admin,2=Staff)
Edit Copy Delete	512	Benny	ben	benny@12345	Male	2147483647	Jakkur	45231	2
Edit Copy Delete	152	John	john123	8756j	Male	2147483647	JP NAGAR	42134	2
Edit Copy Delete	210	Priya	priyaa	12125	Female	2147483647	RR NAGAR	4420	2
Edit Copy Delete	18	Sheetal	shetal	shetal125	Female	2147483647	Sakarnagar	4213	2

6. Displaying the user_id and bill_num of the users who have purchased for more than INR 4000 till now and yet to pay a final bill lesser than INR 12001.

select user_id, bill_num from bill where Amt_paid>4000 AND total_amt<12001;



9. Functions and Procedures

Create a Function and Procedure. State the objective of the function / Procedure. Run anddisplay the results.

FUNCTION

I have implemented quite a simple function which takes the input as the cloth_id and returns the name of that apparel. Specific apparels are chosen here to indicate that they are the best-selling in the shop right now. So, this function encourages the customers to purchase that particular apparel by providing them clarity on it.

The function is written as follows:

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION
`Supp_ID_Order`(`cloth_id` INT) RETURNS varchar(50)
CHARSET utf8mb4
begin
      declare orderSupp int(10) DEFAULT 0;
   declare var varchar(50);
   set var = 'no values';
   IF cloth_id=214 then
      set var='This ID corresponds to the apparel Salwar
Kameez !';
   end if;
   IF cloth_id=156 THEN
      set var = 'This ID corresponds to the apparel SHORTS !';
   end if;
   IF cloth_id=289 THEN
      set var= 'This ID corresponds to the apparel BLAZER. !';
   end if;
   return var;
end$$
DELIMITER ;
```

Server: localhost:3310 » Database: frontend_cloth_store_management_system_cs355

Export

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `Supp_ID_Order`(`cloth_id` INT) RETURNS varchar(50) CHARSET utf8mb4
begin
    declare orderSupp int(10) DEFAULT 0;
    declare var varchar(50);
    set var = 'no values';
    IF cloth_id=214 then
        set var='This ID corresponds to the apparel Salwar Kameez !';
    end if;
    IF cloth_id=156 THEN
        set var = 'This ID corresponds to the apparel SHORTS !';
    end if;
    IF cloth_id=289 THEN
        set var= 'This ID corresponds to the apparel BLAZER. !';
    end if;
    return var;
```

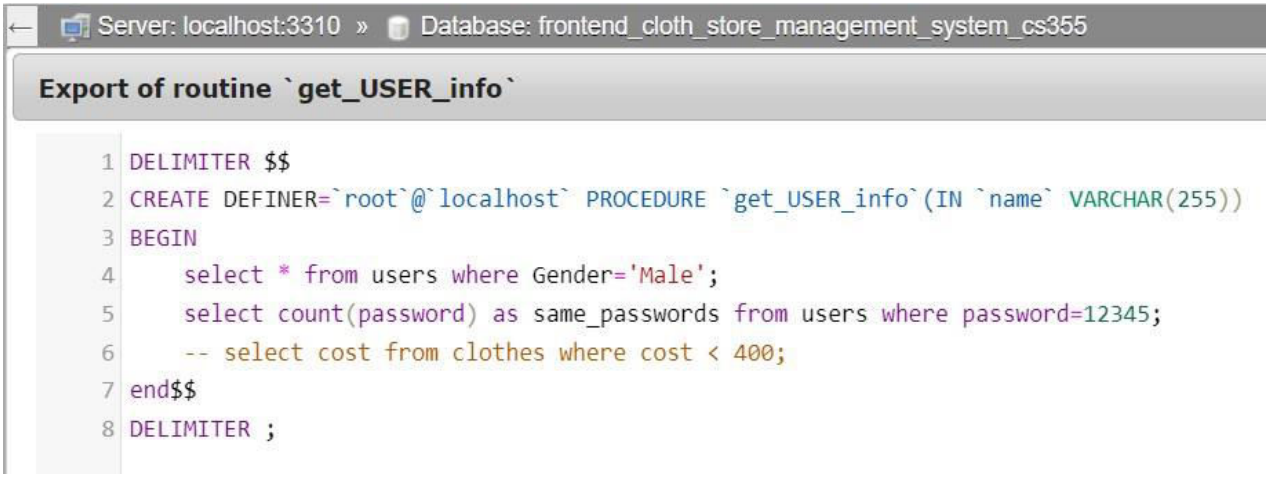
Calling the function given an input value:

PROCEDURE

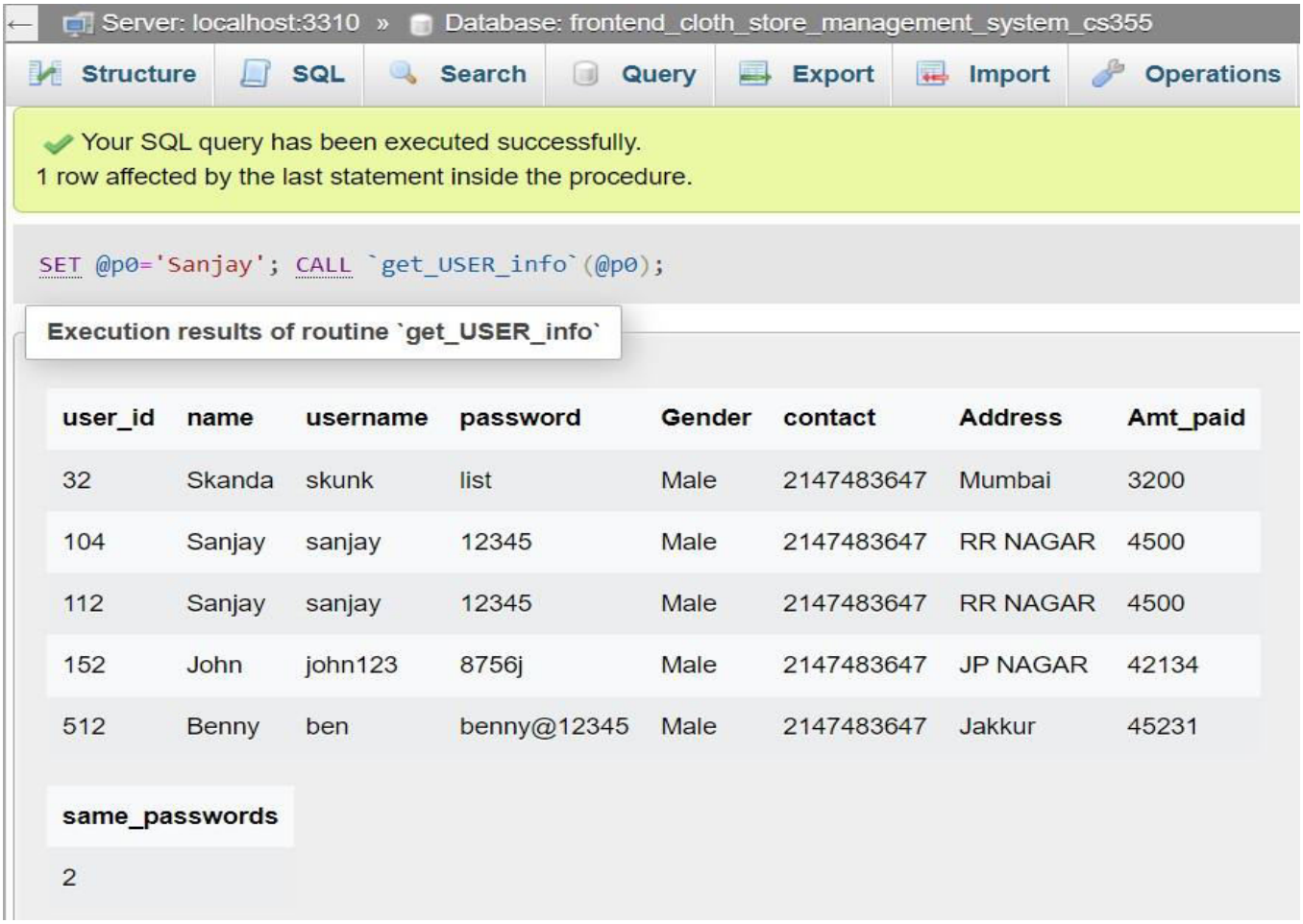
I have implemented this simple procedure to select all the males in the users table. It also counts the number of users whose passwords are exactly the same. This procedure reinforces the users to maintain a unique identity. It urges them to have a stricter password to ensure the safety of their account.

The procedure is written as follows:

```
DELIMITER $$
CREATE DEFINER = `root`@`localhost` PROCEDURE
`get_USER_info`(IN `name` VARCHAR(255))
BEGIN
      select * from users where Gender='Male';
   select count(password) as same_passwords from users
where password=12345;
   -- select cost from clothes where cost < 400;
end$$
DELIMITER ;
```



Calling the procedure given an input value:



10. Triggers and Cursors

Create a Trigger and a Cursor. State the objective. Run and display the results.

TRIGGER

I have implemented a trigger to increase the cost price of the cloth by a margin. This is done typically to provide for the labor and transport charge required by the supplier for getting the cloth from its place of manufacture to the shop and eventually to deliver it to the customer.

The trigger is written as follows:

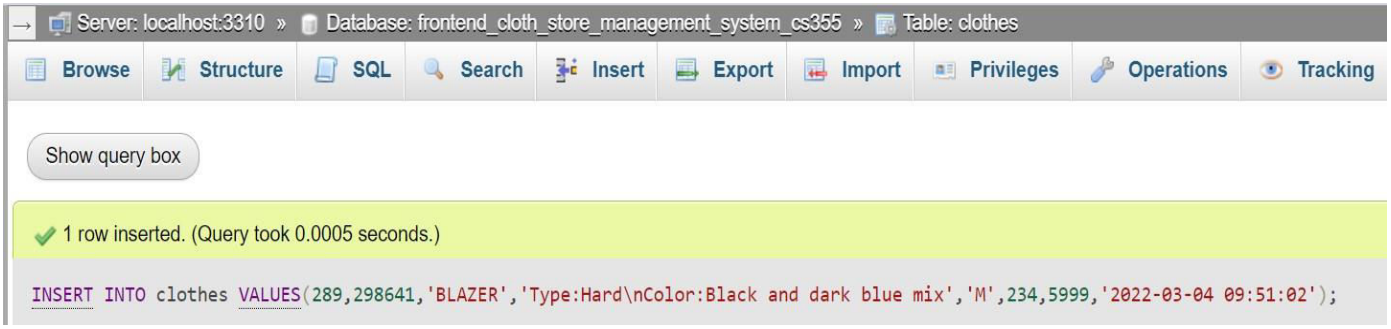
```
CREATE TRIGGER `before_insert_cost` BEFORE
INSERT ON `clothes`
 FOR EACH ROW begin
   if new.cost < =100 then
      set new.cost = new.cost + 20 ;
      end if ;
   if new.cost > 100 and new.cost <= 500 THEN
      set new.cost = new.cost + 30 ;
   end if ;
   if new.cost > 500 and new.cost <= 1000 THEN
      set new.cost = new.cost + 50 ;
   end if ;
   if new.cost > 1000 and new.cost<=5000 THEN
      set new.cost = new.cost + 70 ;
   end if ;
   if new.cost > 5000 and new.cost < 10000 THEN
      set new.cost = new.cost + 80 ;
   end if ;

end
```

Server: localhost:3310 » Database: frontend_cloth_store_management_system_cs355 » Table: clothes

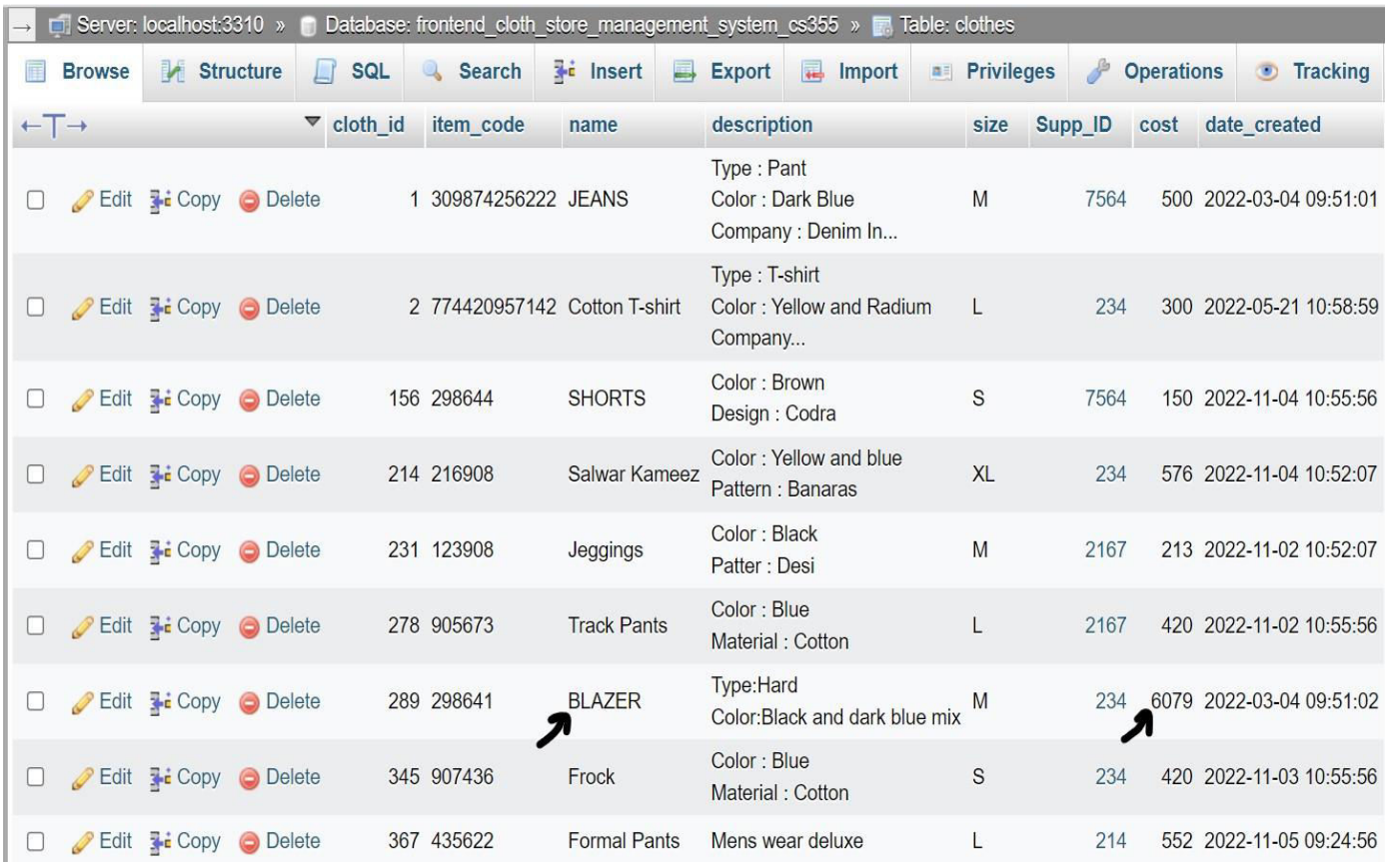
Export of trigger `before_insert_cost`

```
1  CREATE TRIGGER `before_insert_cost` BEFORE INSERT ON `clothes`
2   FOR EACH ROW begin
3      if new.cost<=100 then
4          set new.cost = new.cost+20 ;
5      end if ;
6      if new.cost>100 and new.cost<=500 THEN
7          set new.cost = new.cost+30 ;
8      end if ;
9      if new.cost>500 and new.cost<=1000 THEN
10         set new.cost = new.cost+50 ;
11     end if ;
12     if new.cost>1000 and new.cost<=5000 THEN
13         set new.cost = new.cost+70 ;
14     end if ;
15     if new.cost>5000 and new.cost<10000 THEN
16         set new.cost = new.cost+80 ;
```

Inserting a value into the table to demonstrate the working of this trigger:



Indicating that the cost price of the added apparel is modified and taken (i.e. 5999 + 80)



Server: localhost:3310 » Database: frontend_cloth_store_management_system_cs355 » Table: clothes

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Tracking

	cloth_id	item_code	name	description	size	Supp_ID	cost	date_created
Edit Copy Delete	1	309874256222	JEANS	Type : Pant Color : Dark Blue Company : Denim In...	M	7564	500	2022-03-04 09:51:01
Edit Copy Delete	2	774420957142	Cotton T-shirt	Type : T-shirt Color : Yellow and Radium Company...	L	234	300	2022-05-21 10:58:59
Edit Copy Delete	156	298644	SHORTS	Color : Brown Design : Codra	S	7564	150	2022-11-04 10:55:56
Edit Copy Delete	214	216908	Salwar Kameez	Color : Yellow and blue Pattern : Banaras	XL	234	576	2022-11-04 10:52:07
Edit Copy Delete	231	123908	Jeggings	Color : Black Patter : Desi	M	2167	213	2022-11-02 10:52:07
Edit Copy Delete	278	905673	Track Pants	Color : Blue Material : Cotton	L	2167	420	2022-11-02 10:55:56
Edit Copy Delete	289	298641	BLAZER	Type:Hard Color:Black and dark blue mix	M	234	6079	2022-03-04 09:51:02
Edit Copy Delete	345	907436	Frock	Color : Blue Material : Cotton	S	234	420	2022-11-03 10:55:56
Edit Copy Delete	367	435622	Formal Pants	Mens wear deluxe	L	214	552	2022-11-05 09:24:56

CURSOR

I have implemented a simple cursor which transfers all the records from the table **clothes** into another table **backup_routine**. If at all the original table containing the details of the clothes gets deleted then, we can retrieve the information safely from this backup table.

The cursor is written as follows:

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE
`backup_routine`()
BEGIN
	DECLARE done INT DEFAULT 0;
	DECLARE cloth_id,Supp_ID INTEGER;
	DECLARE
item_code,name,description,size,cost,date_created VARCHAR
(255);
	DECLARE cur CURSOR FOR SELECT * FROM clothes;
	DECLARE CONTINUE HANDLER FOR NOT Found
SET done=1;
	OPEN cur;
	label:LOOP
	FETCH cur INTO
cloth_id,item_code,name,description,size,Supp_ID,cost,date_cre
ated;
	INSERT INTO backup_routine
VALUES(cloth_id,item_code,name,description,size,Supp_ID,cos
t,date_created);
	IF done = 1 THEN LEAVE label;
	END IF;
	END LOOP;
	CLOSE cur;
END$$
DELIMITER ;
```

Server: localhost:3310 » Database: frontend_cloth_store_management_system_cs355

Export of routine `backup_routine`

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `backup_routine`()
BEGIN
        DECLARE done INT DEFAULT 0;
        DECLARE cloth_id,Supp_ID INTEGER;
        DECLARE item_code,name,description,size,cost,date_created VARCHAR
(255);
        DECLARE cur CURSOR FOR SELECT * FROM clothes;
        DECLARE CONTINUE HANDLER FOR NOT Found SET done=1;
        OPEN cur;
        label:LOOP
        FETCH cur INTO
cloth_id,item_code,name,description,size,Supp_ID,cost,date_created;
        INSERT INTO backup_routine
VALUES(cloth_id,item_code,name,description,size,Supp_ID,cost,date_created);
        IF done = 1 THEN LEAVE label;
        END IF;
        END LOOP;
        CLOSE cur;
END$$
DELIMITER ;
```

Calling the cursor:

All the records from the table **clothes** are copied onto the table **backup_routine** upon calling the cursor.

Server: localhost:3310 » Database: frontend_cloth_store_management_system_cs355 » Table: backup_routine

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking

Showing rows 0 - 11 (12 total, Query took 0.0015 seconds.)

```
SELECT * FROM `backup_routine`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

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

Extra options

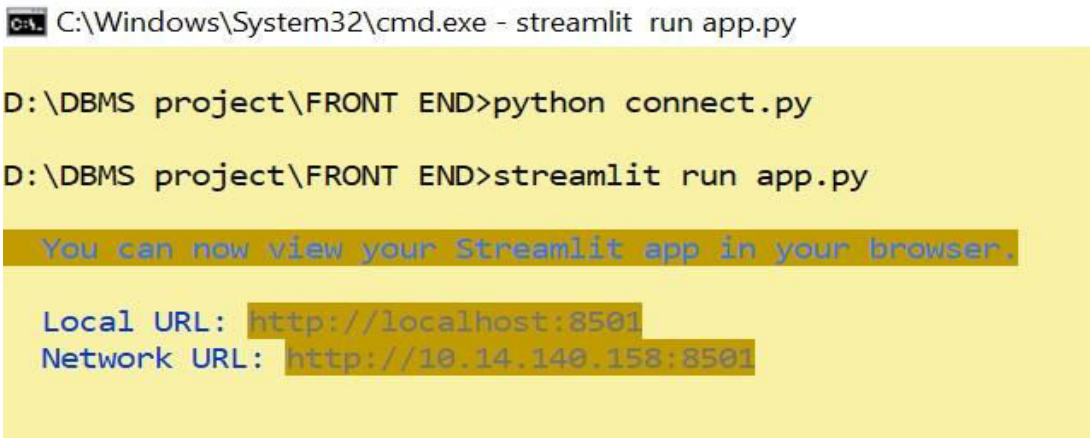
cloth_id	item_code	name	description	size	Supp_ID	cost	date_created
2	774420957142	Cotton T-shirt	Type : T-shirt Color : Yellow and Radium Company...	L	234	300	2022-05-21 10:58:59
134	904563	Unknown apparel	Color : Blood red Cotton	XL	234	3570	2022-11-12 00:00:00
156	298644	SHORTS	Color : Brown Design : Codra	S	7564	150	2022-11-04 10:55:56
214	216908	Salwar Kameez	Color : Yellow and blue Pattern : Banaras	XL	234	576	2022-11-04 10:52:07
231	123908	Jeggings	Color : Black Patter : Desi	M	2167	213	2022-11-02 10:52:07
278	905673	Track Pants	Color : Blue Material : Cotton	L	2167	420	2022-11-02 10:55:56
289	298641	BLAZER	Type:Hard Color:Black and dark blue mix	M	234	6079	2022-03-04 09:51:02
367	435622	Formal Pants	Mens wear deluxe	L	214	552	2022-11-05 09:24:56
Console	784533	3/4th Shorts	Color : Military green\nMaterial type : Cotton+pol...	XL	214	2000	2022-11-16 00:00:00

11. Developing a Frontend

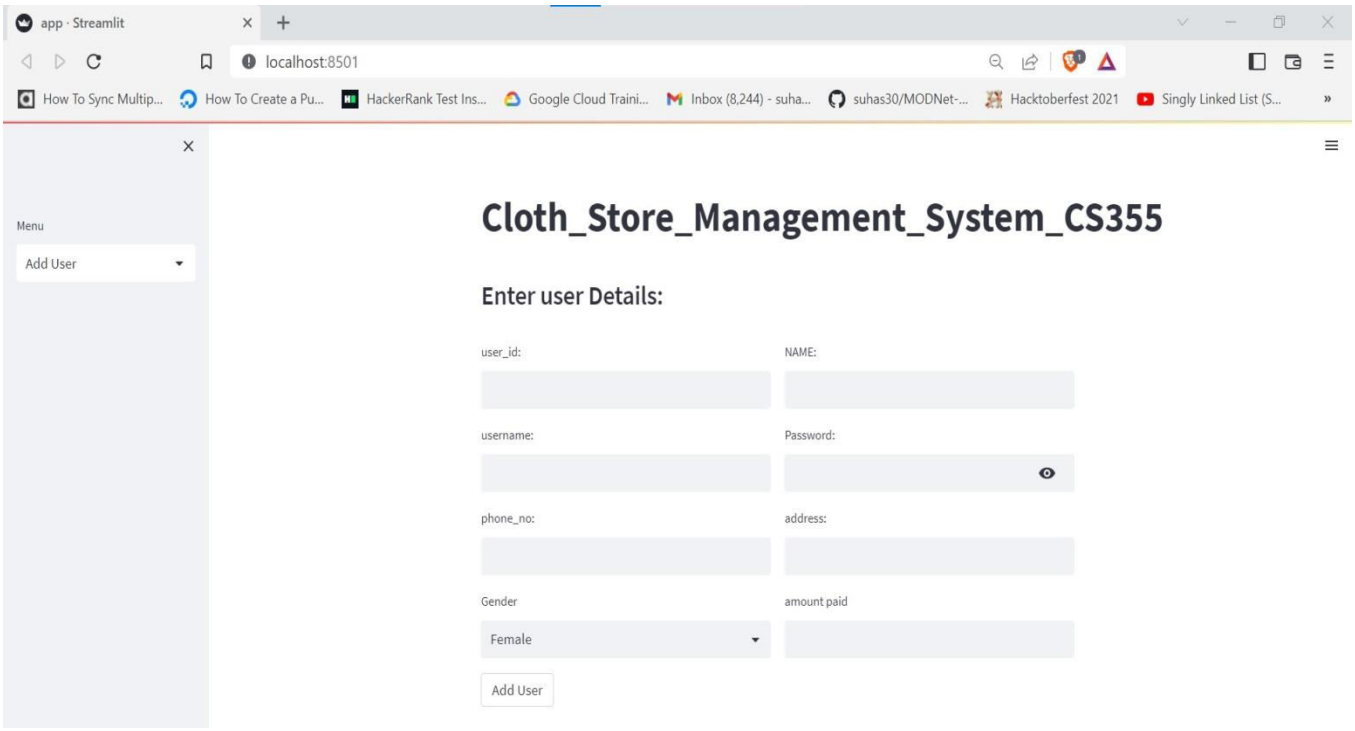
The frontend should support

1. Addition, Modification and Deletion of records from any chosen table
2. There should be an window to accept and run any SQL statement and display the result

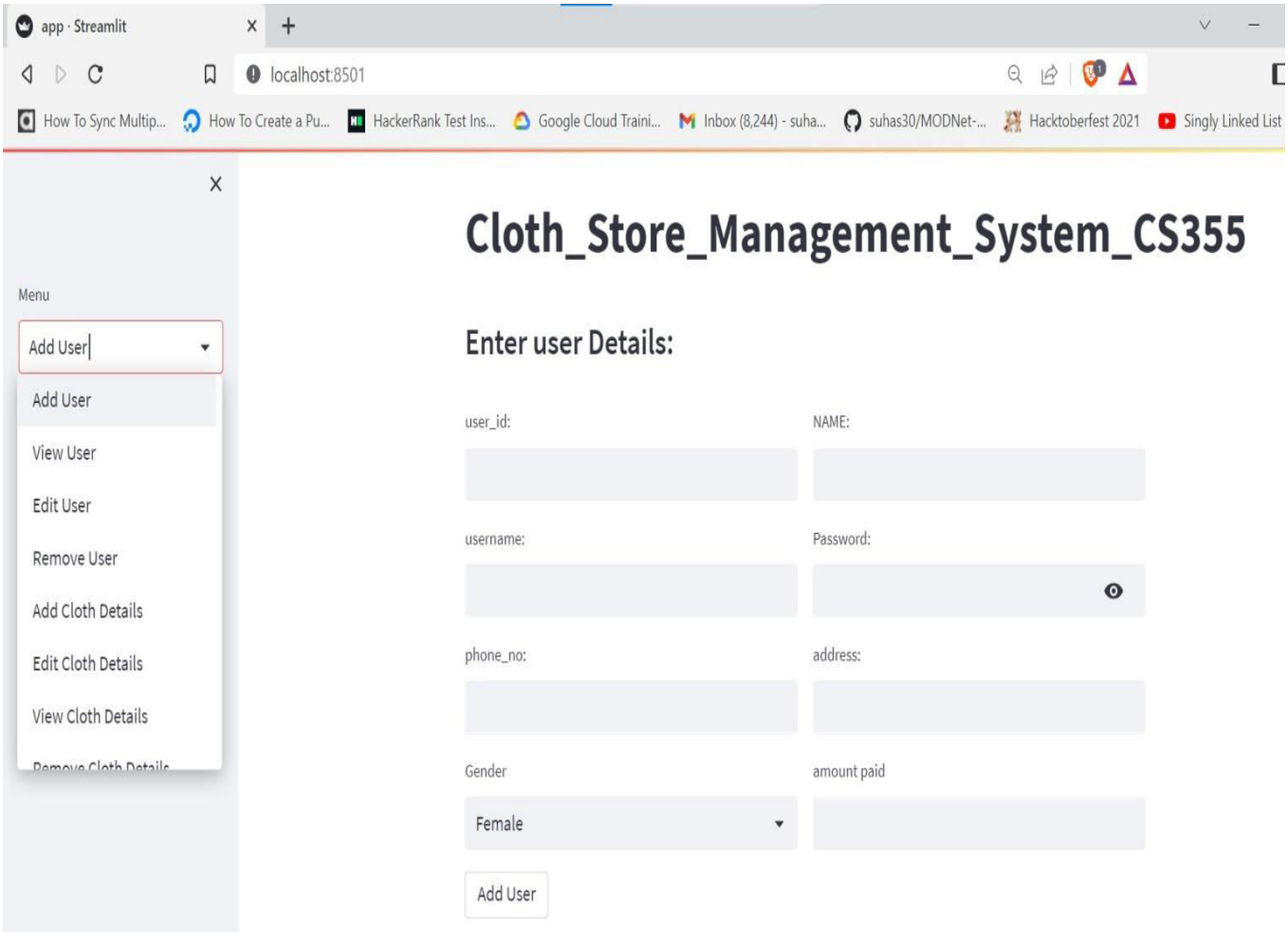
Establishing connection by running connect.py and then hosting it on website via streamlit.



The home page which appears upon running streamlit run app.py



The sidebar containing the menu which shows the different operations that can be performed.



Viewing the details of the created users

Cloth_Store_Management_System_CS355

View created users

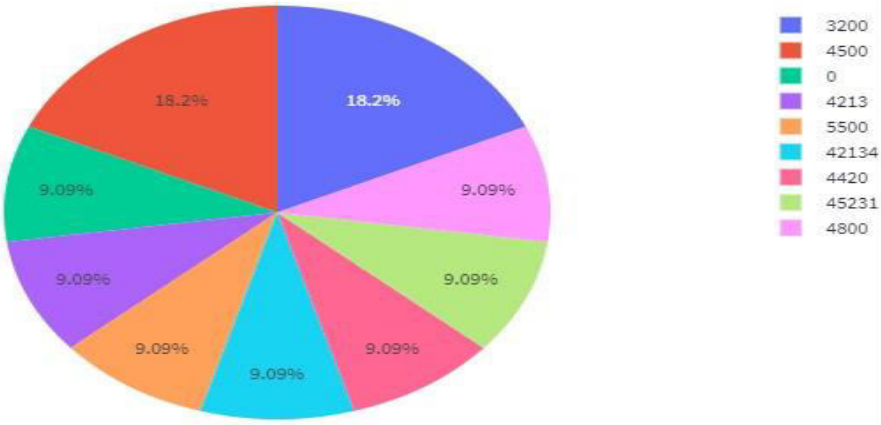
View all Users

	user_id	name	username	password	gender	contact	address	amt_paid
0	1	admin	admin	0192023a7bbd7		0		0
1	18	Sheetal	shetal	shetal125	Female	2147483647	Sakarnagar	4213
2	32	Skanda	skunk	list	Male	2147483647	Mumbai	3200
3	104	Sanjay	sanjay	12345	Male	2147483647	RR NAGAR	4500
4	112	Sanjay	sanjay	12345	Male	2147483647	RR NAGAR	4500
5	131	Shanbogh	sandy	prep123	Female	2147483647	Ram NAGAR	5500
6	152	John	john123	8756j	Male	2147483647	JP NAGAR	42134
7	186	Kali	kalyani123	kr@12345	Female	2147483647	Bengal	3200
8	210	Priya	priyaa	12125	Female	2147483647	RR NAGAR	4420
9	512	Benny	ben	benny@12345	Male	2147483647	Jakkur	45231

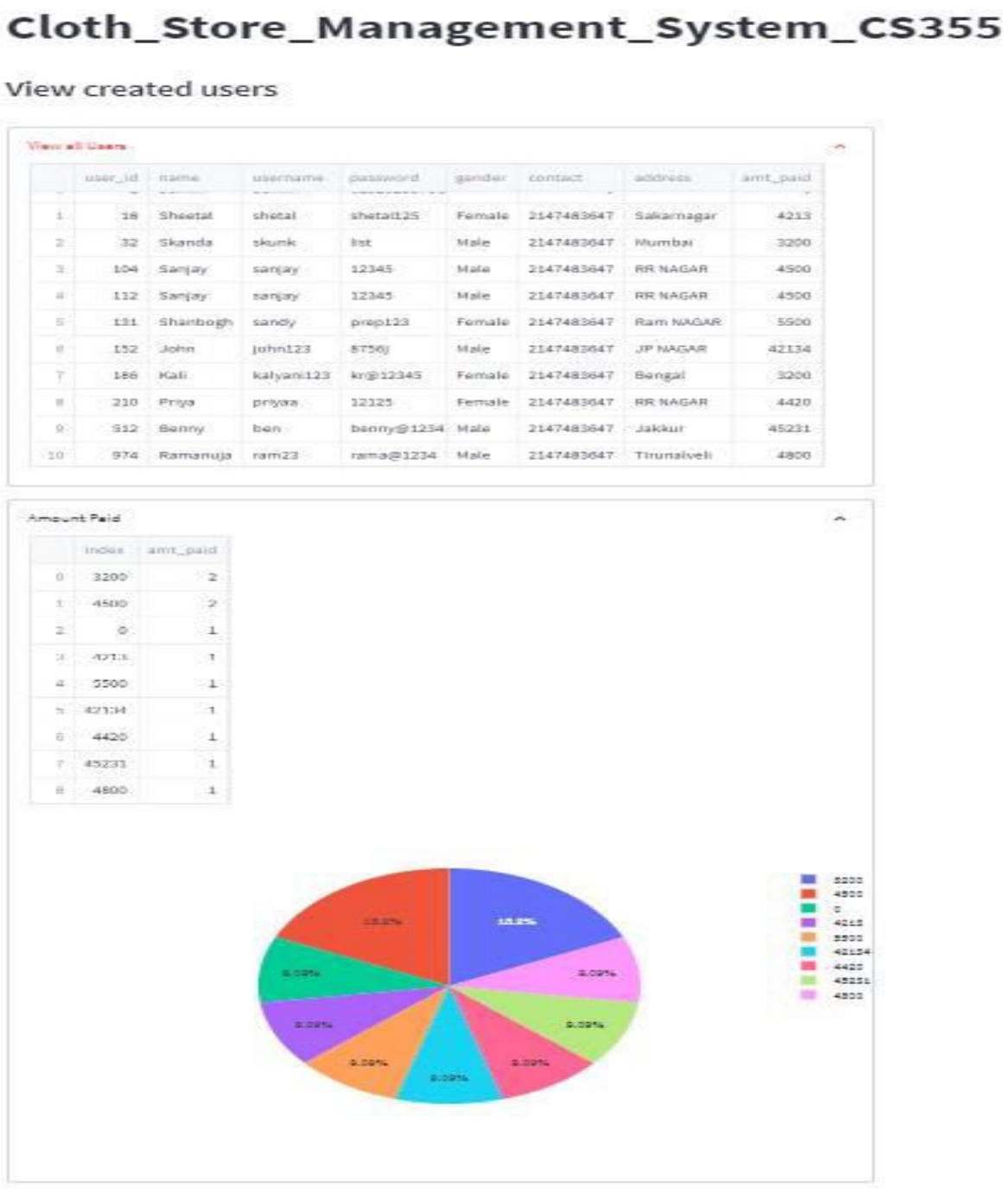
Analyzing the amount paid from users to see how to fix the rate for the upcoming apparel.

Amount Paid

	index	amt_paid
0	3200	2
1	4500	2
2	0	1
3	4213	1
4	5500	1
5	42134	1
6	4420	1
7	45231	1
8	4800	1



This is how the whole page looks like when we opt to view users.



Adding a new user

Cloth_Store_Management_System_CS355

Enter user Details:

user_id: 974

NAME: Ramanuja

username: ram23

Password: ••••••••••

phone_no: 9087634517

address: Tirunalveli

Gender: Male

amount paid: 4800

Add User

Successfully added this new user

Showing that new user has been added into the table

Cloth_Store_Management_System_CS355

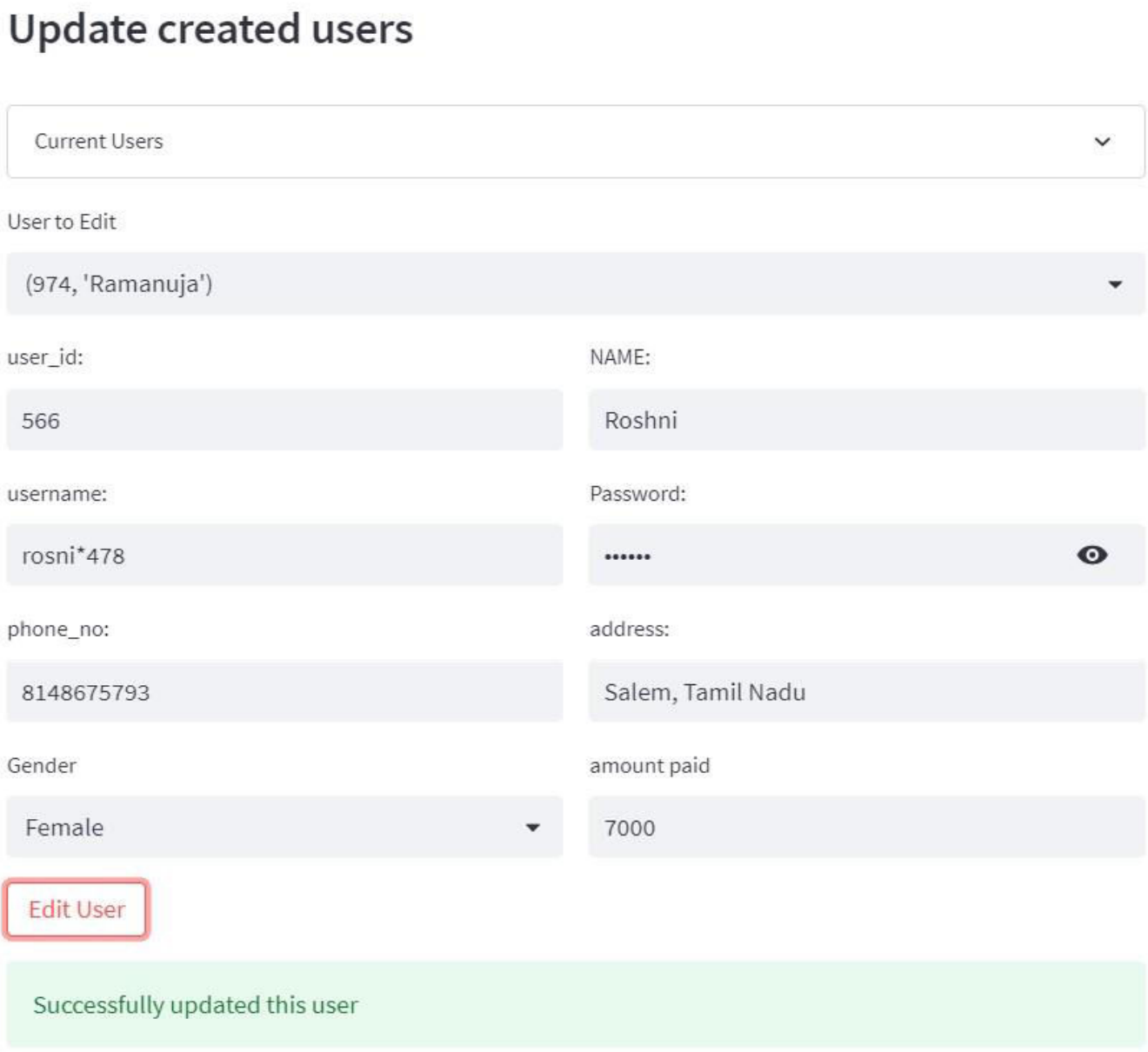
View created users

View all Users

	user_id	name	username	password	gender	contact	address	amt_paid
1	18	Sheetal	shetal	shetal125	Female	2147483647	Sakarnagar	4213
2	32	Skanda	skunk	list	Male	2147483647	Mumbai	3200
3	104	Sanjay	sanjay	12345	Male	2147483647	RR NAGAR	4500
4	112	Sanjay	sanjay	12345	Male	2147483647	RR NAGAR	4500
5	131	Shanbogh	sandy	prep123	Female	2147483647	Ram NAGAR	5500
6	152	John	john123	8756j	Male	2147483647	JP NAGAR	42134
7	186	Kali	kalyani123	kr@12345	Female	2147483647	Bengal	3200
8	210	Priya	priyaa	12125	Female	2147483647	RR NAGAR	4420
9	512	Benny	ben	benny@1234	Male	2147483647	Jakkur	45231
10	974	Ramanuja	ram23	rama@1234	Male	2147483647	Tirunalveli	4800

Amount Paid

Updating the created user



Showing that the selected record has been updated with the new values (see last row)

Edit User

Successfully updated this user

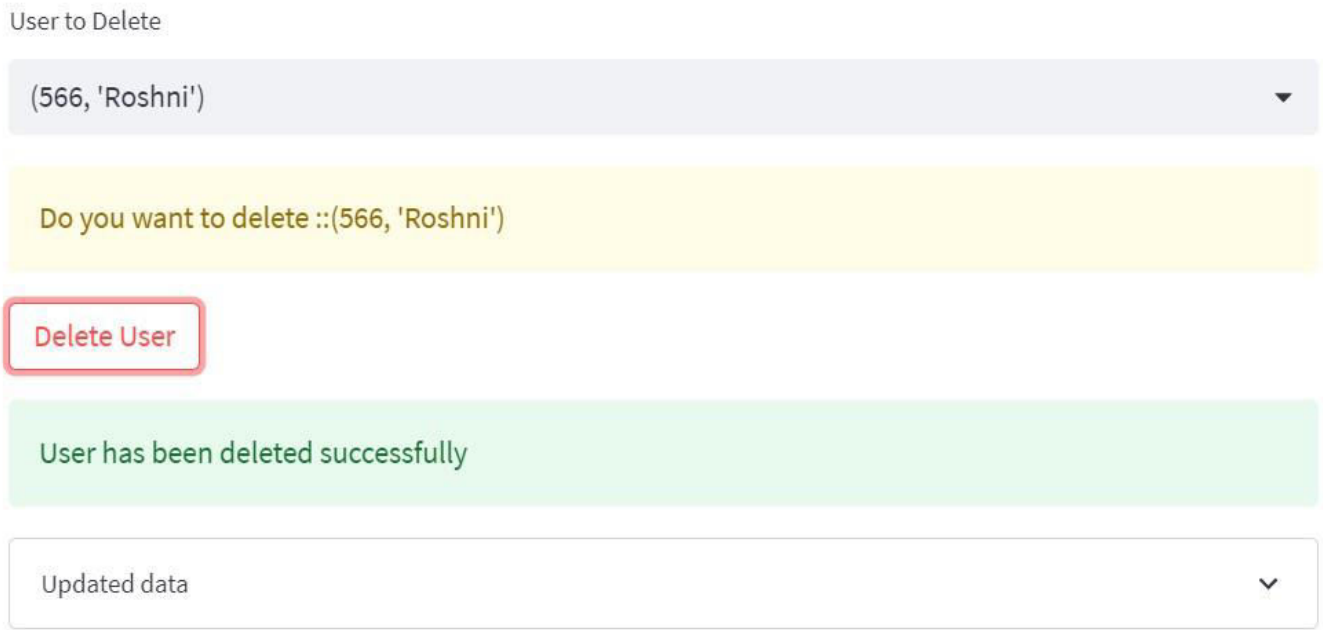
Updated data

	user_id	name	username	password	gender	contact	address	amt_paid
1	18	Sheetal	shetal	shetal125	Female	2147483647	Sakarnagar	4213
2	32	Skanda	skunk	list	Male	2147483647	Mumbai	3200
3	104	Sanjay	sanjay	12345	Male	2147483647	RR NAGAR	4500
4	112	Sanjay	sanjay	12345	Male	2147483647	RR NAGAR	4500
5	131	Shanbogh	sandy	prep123	Female	2147483647	Ram NAGAR	5500
6	152	John	john123	8756j	Male	2147483647	JP NAGAR	42134
7	186	Kali	kalyani123	kr@12345	Female	2147483647	Bengal	3200
8	210	Priya	priyaa	12125	Female	2147483647	RR NAGAR	4420
9	512	Benny	ben	benny@123	Male	2147483647	Jakkur	45231
10	566	Roshni	rosni*478	sun642	Female	2147483647	Salem, Tamil Nadu	7000

Moving to the delete operation for user. Selecting the user to delete.

7	186	Kali	kalyani123	kr@12345	Female	2147483647	Bengal	3200
8	210	Priya	priyaa	12125	Female	2147483647	RR NAGAR	4420
9	512	Benny	ben	benny@123	Male	2147483647	Jakkur	45231
10	566	Roshni	rosni*478	sun642	Female	2147483647	Salem, Tamil Nadu	7000

Successfully showing the selected user is deleted.



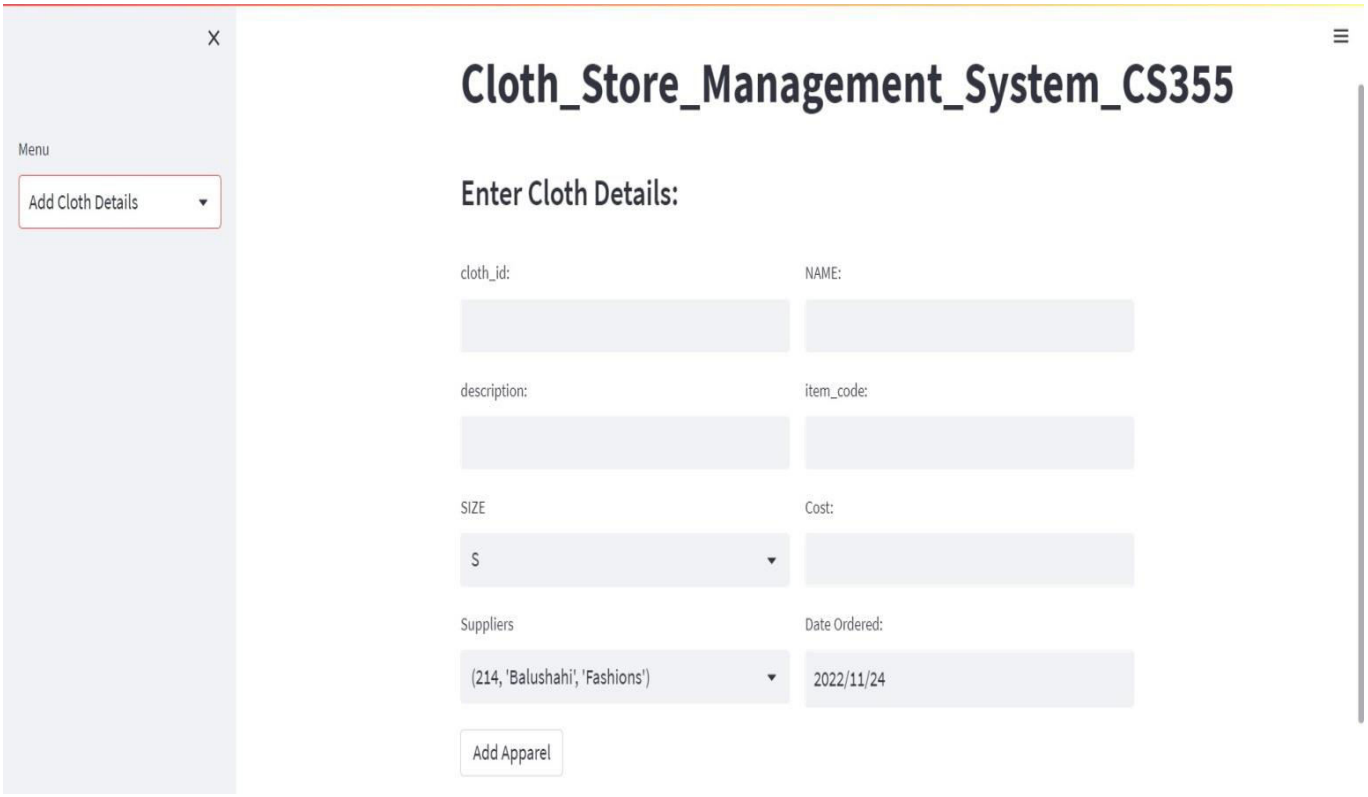
Showing in the table that the selected user's record is removed from the table.

User has been deleted successfully

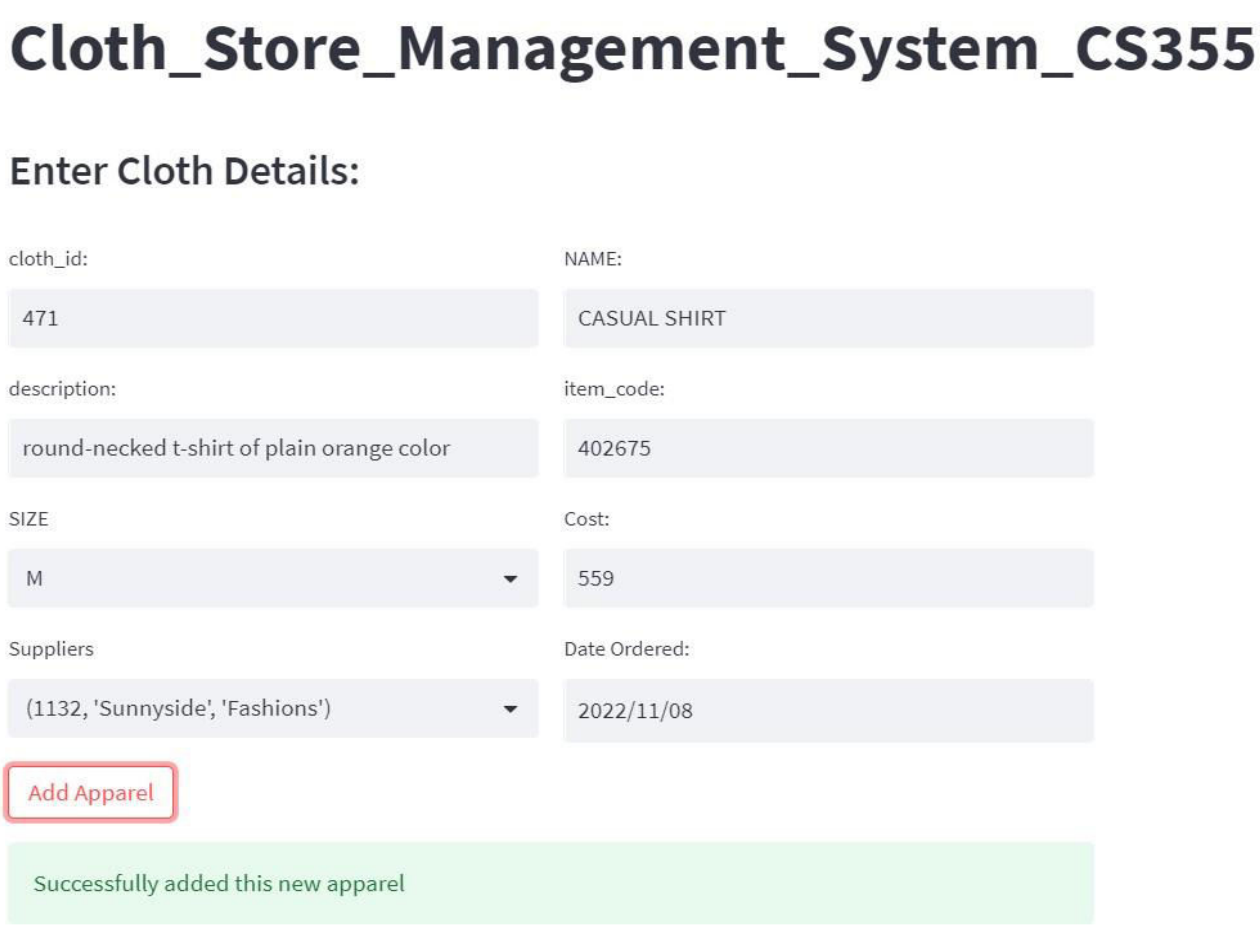
Updated data

	user_id	name	username	password	gender	contact	address	amt_paid
0	1	admin	admin	0192023a7bbd7		0		0
1	18	Sheetal	shetal	shetal125	Female	2147483647	Sakarnagar	4213
2	32	Skanda	skunk	list	Male	2147483647	Mumbai	3200
3	104	Sanjay	sanjay	12345	Male	2147483647	RR NAGAR	4500
4	112	Sanjay	sanjay	12345	Male	2147483647	RR NAGAR	4500
5	131	Shanbogh	sandy	prep123	Female	2147483647	Ram NAGAR	5500
6	152	John	john123	8756j	Male	2147483647	JP NAGAR	42134
7	186	Kali	kalyani123	kr@12345	Female	2147483647	Bengal	3200
8	210	Priya	priyaa	12125	Female	2147483647	RR NAGAR	4420
9	512	Benny	ben	benny@12345	Male	2147483647	Jakkur	45231

Performing CRUD operations for clothes and apparels



Entering the details of a new apparel and inserting it into the table



Update page for modifying the details of desired apparels.

Cloth_Store_Management_System_CS355

Update existing Clothes and Apparels

Existing Clothes and Apparels

	cloth_id	item_code	name	description
2	156	298644	SHORTS	Color : Brown Design : Codra
3	214	216908	Salwar Kameez	Color : Yellow and blue Pattern : Banaras
4	231	123908	Jeggings	Color : Black Patter : Desi
5	278	905673	Track Pants	Color : Blue Material : Cotton
6	289	298641	BLAZER	Type:Hard Color:Black and dark blue mix
7	367	435622	Formal Pants	Mens wear deluxe
8	427	784533	3/4th Shorts	Color : Military green\nMaterial type : Cotton+polyester
9	459	56347832	Black T-shirt	Colar blue
10	471	402675	CASUAL SHIRT	round-necked t-shirt of plain orange color
11	907	568432	Dhoti	Color : Bright Gold

Apparel to edit

Modifying the details of a selected apparel

Apparel to edit

(471, '402675', 'CASUAL SHIRT', 1132)

cloth_id:

501

NAME:

Round_neck SHIRT

description:

Cotton cloth of light orange color

item_code:

658333

SIZE

M

Cost:

611

Suppliers

(234, 'Saiyan', 'Stores')

Date Ordered:

2022/11/09

Edit Cloth details

Successfully updated the details of this Cloth

Showing in the table that the selected apparel has been modified with the new values entered for it. (CASUAL SHIRT to Round_neck shirt)

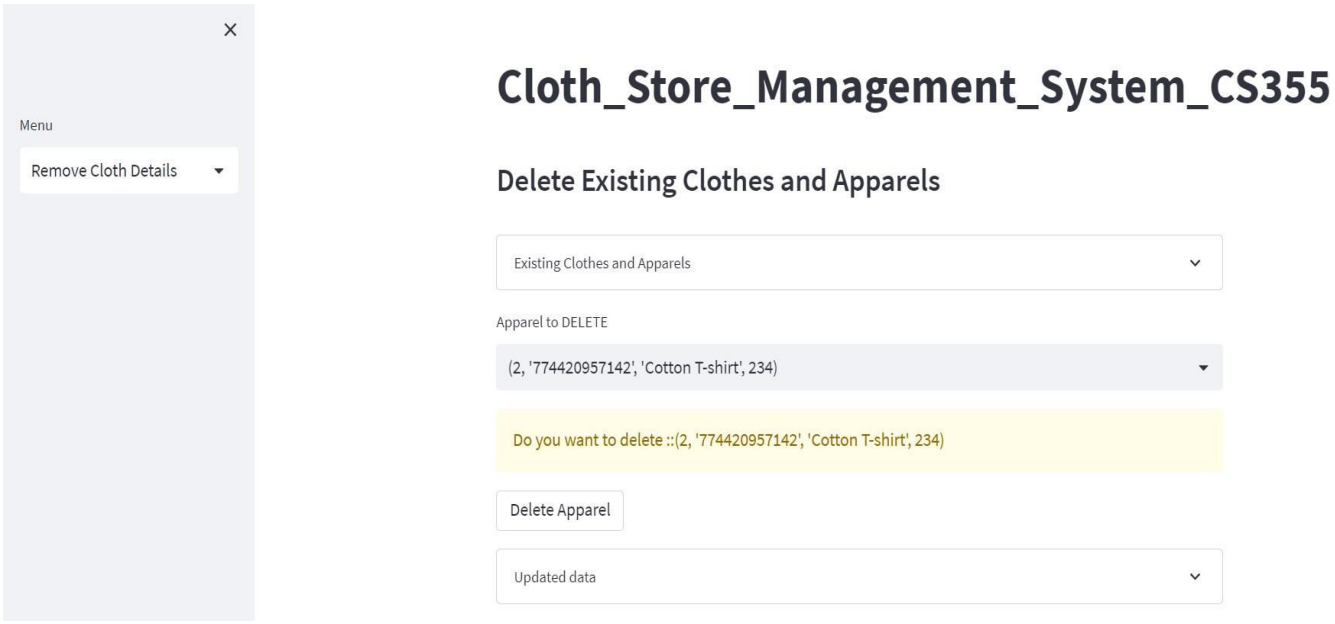
Edit Cloth details

Successfully updated the details of this Cloth

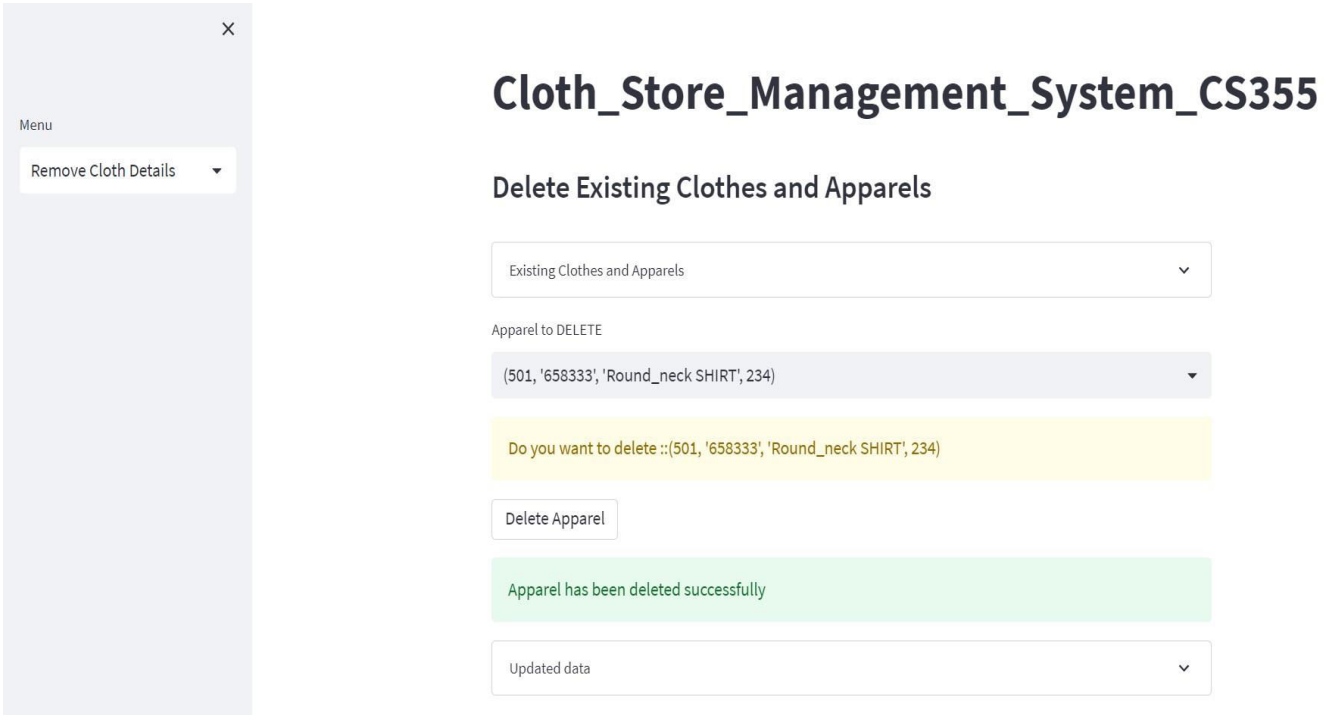
Updated data

	cloth_id	item_code	name	description
2	156	298644	SHORTS	Color : Brown Design : Codra
3	214	216908	Salwar Kameez	Color : Yellow and blue Pattern : Banaras
4	231	123908	Jeggings	Color : Black Patter : Desi
5	278	905673	Track Pants	Color : Blue Material : Cotton
6	289	298641	BLAZER	Type:Hard Color:Black and dark blue mix
7	367	435622	Formal Pants	Mens wear deluxe
8	427	784533	3/4th Shorts	Color : Military green\nMaterial type : Cotton+polyester
9	459	56347832	Black T-shirt	Colar blue
10	501	658333	Round_neck SHIRT	Cotton cloth of light orange color
11	907	568432	Dhoti	Color : Bright Gold

Selecting the apparel to be deleted.



Successfully deleting the selected apparel.



Showing that the apparel selected (in this case, Round_neck SHIRT) has been deleted from the table.

Do you want to delete ::(501, '658333', 'Round_neck SHIRT', 234)

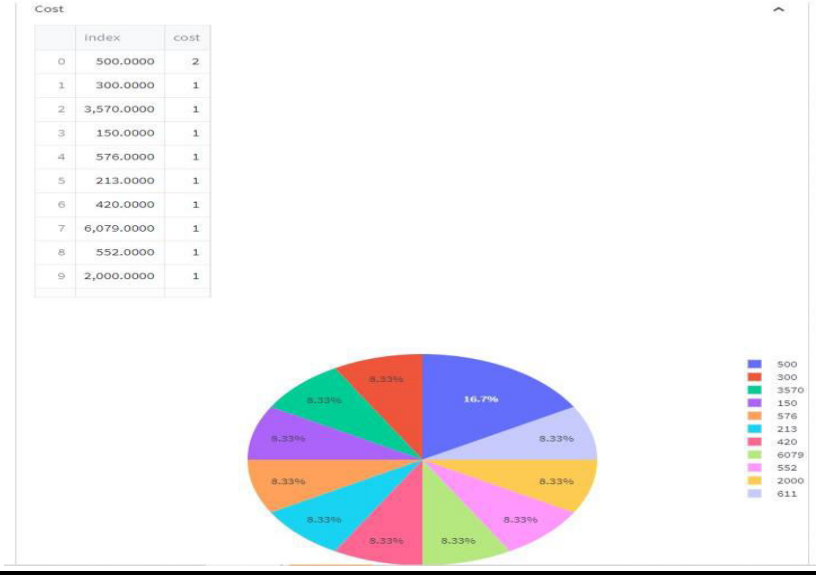
Delete Apparel

Apparel has been deleted successfully

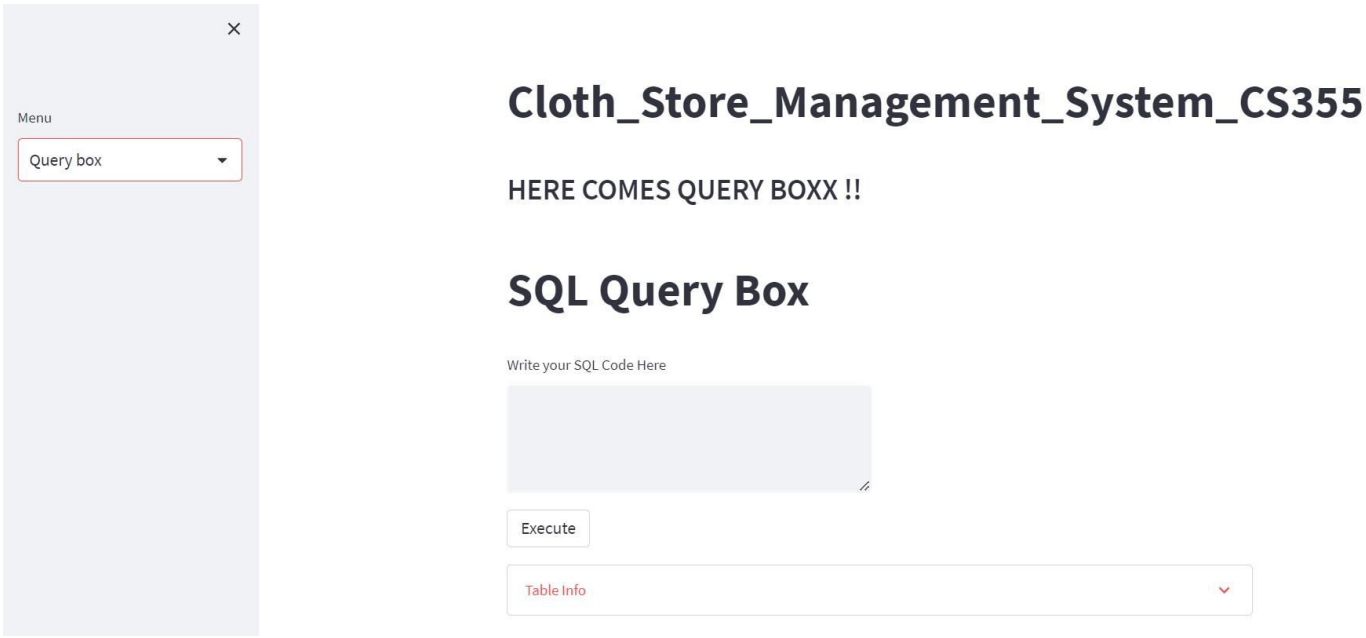
Updated data

	cloth_id	item_code	name	description
1	134	904563	Unknown apparel	Color : Blood red Cotton
2	156	298644	SHORTS	Color : Brown Design : Codra
3	214	216908	Salwar Kameez	Color : Yellow and blue Pattern : Banaras
4	231	123908	Jeggings	Color : Black Patter : Desi
5	278	905673	Track Pants	Color : Blue Material : Cotton
6	289	298641	BLAZER	Type:Hard Color:Black and dark blue mix
7	367	435622	Formal Pants	Mens wear deluxe
8	427	784533	3/4th Shorts	Color : Military green\nMaterial type : Cotton+polyester
9	459	56347832	Black T-shirt	Colar blue
10	907	568432	Dhoti	Color : Bright Gold

Analyzing the cost of the clothes through a simple pie chart



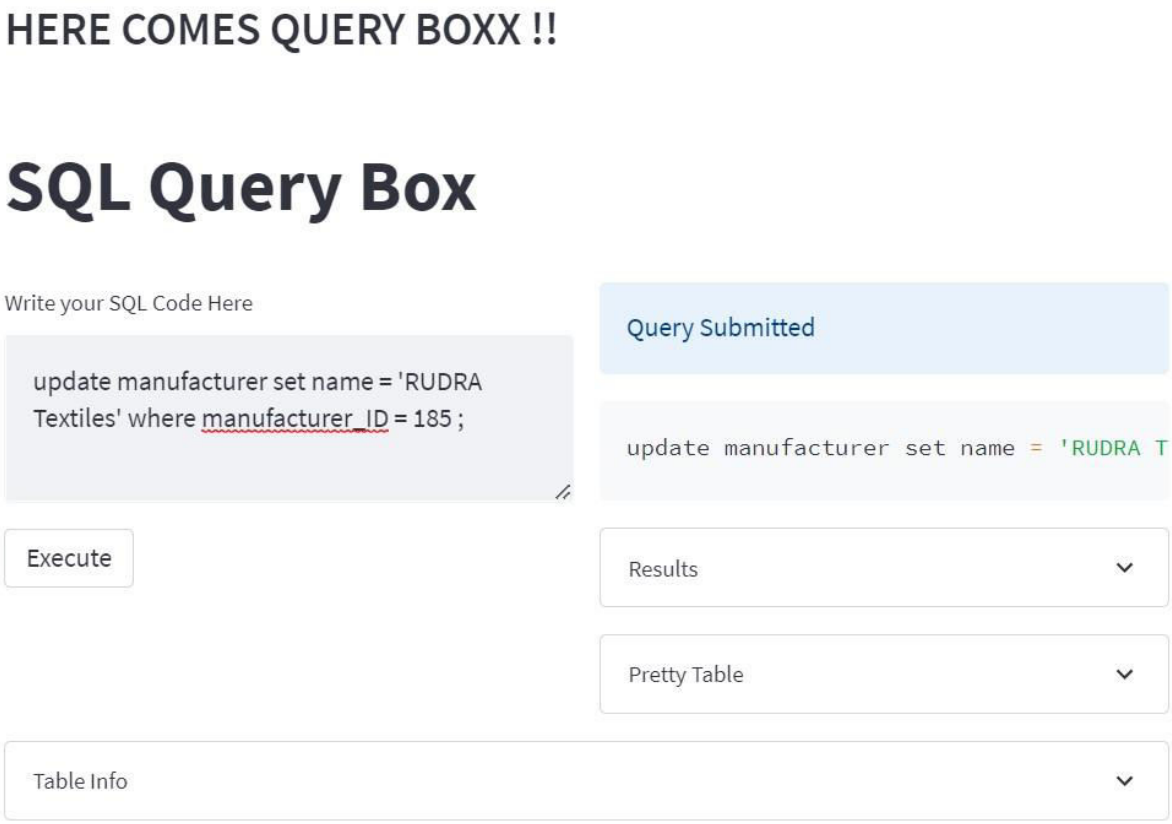
The Query box



Before Update

	manufacturer_ID	name	phone	address	cloth_id
Edit Copy Delete	185	Miriam Expo	9896509	Thiruvanant	134
Edit Copy Delete	368	Kalyan Text	78695768	Pondicherry	278
Edit Copy Delete	458	Varghese Te	897590687	Hyderabad	214
Edit Copy Delete	509	GokulDas Ex	996578437	Madurai	231

Executing MySQL queries through the webpage run via streamlit



After update in the table manufacturer

	manufacturer_ID	name	phone	address	cloth_id
Edit Copy Delete	185	RUDRA Texti	9896509	Thiruvanant	134
Edit Copy Delete	368	Kalyan Text	78695768	Pondicherry	278
Edit Copy Delete	458	Varghese Te	897590687	Hyderabad	214
Edit Copy Delete	509	GokulDas Ex	996578437	Madurai	231

Code:

```
# Importing packages
import update as upd
import streamlit as st
import mysql.connector
import pymysql
import create as cr
import database as dtb
import delete as dlt
import read as rd

def main():
    st.title("Cloth_Store_Management_System_CS355")
    menu = ["Add User", "View User", "Edit User", "Remove User","Add Cloth Details","Edit Cloth Details","View Cloth Details","Remove Clot
    choice = st.sidebar.selectbox("Menu", menu)
    dtb.create_table_user()
    if choice == "Add User":
        st.subheader("Enter user Details:")
        cr.create_new_user()

    if choice == "Add Supplier":
        dtb.create_table_suppliers()

        st.subheader("Enter Supplier Details:")
        cr.create_new_supplier()

    elif choice == "View User":
        st.subheader("View created users")
        rd.read_user()
```

```
        rd.read_user()

    elif choice == "Edit User":
        st.subheader("Update created users")
        upd.update_user()

    elif choice == "Remove User":
        st.subheader("Delete created users")
        dlt.delete_user()

    elif choice == "Add Cloth Details":
        st.subheader("Enter Cloth Details:")
        cr.create_new_cloth()
    elif choice == "View Cloth Details":
        st.subheader("View existing Cloth Details")
        rd.read_clothes()

    elif choice == "Edit Cloth Details":
        st.subheader("Update existing Clothes and Apparels")
        upd.update_clothes()

    elif choice == "Remove Cloth Details":
        st.subheader("Delete Existing Clothes and Apparels")
        dlt.delete_cloth()
    elif choice == "Query box":
        st.subheader("HERE COMES QUERY BOXX !!")
        dtb.queryboxx()
    else:
        st.subheader("About tasks")
```

```
mydb = pymysql.connect(
    host="localhost",
    port=3310,
    user="root",
    passwd="",
    database="frontend_cloth_store_management_system_cs355"
)
c = mydb.cursor()

clothes = ['cloth_id','item_code','name','description','size','Supp_ID','cost','date_created']
users = ['user_id','name','username','password','gender','contact','address','amt_paid']
order_c =['Order_ID','Supp_ID','Order_qty','Order_date','user_id']
suppliers=['fname','lname','address','Supp_ID','contact','date_created']
bill=['bill_num','user_id','order_id','Amt_paid','total_amt']
manufacturers = ['manufacturer_ID','name','phone','address','cloth_id']

def create_table_user():
    c.execute('CREATE TABLE IF NOT EXISTS users(user_id INT, name varchar(255), username TEXT, '
              'password varchar(255),gender TEXT,contact INT,address TEXT,amt_paid INT)')


def create_table_user():
    c.execute('CREATE TABLE IF NOT EXISTS users(user_id INT, name varchar(255), username TEXT, '
              'password varchar(255),gender TEXT,contact INT,address TEXT,amt_paid INT)')

def create_table_cloth():
    c.execute('CREATE TABLE IF NOT EXISTS users(cloth_id INT, item_code varchar(255), name TEXT, '
              'description varchar(255),size TEXT,Supp_ID INT,cost INT,date_created VARCHAR(255))')

def add_new_user_data(user_id,name,username,password,gender,contact,address,amt_paid):
    c.execute('INSERT INTO users(user_id,name,username,password,gender,contact,address,amt_paid) VALUES (%s,%s,%s,'
              '%s,%s,%s,%s,%s)',
              (user_id,name,username,password,gender,contact,address,amt_paid))
    mydb.commit()

def create_table_suppliers():
    c.execute('CREATE TABLE IF NOT EXISTS suppliers(Supp_ID INT, fname varchar(255), lname varchar(20), '
              'contact INT,address TEXT,date_created DATETIME)')

def add_new_supplier_data(Supp_ID, fname,lname,address,contact,date_created):
    c.execute('INSERT INTO suppliers(Supp_ID, fname,lname,address,contact,date_created) VALUES (%s,%s,%s,'
              '%s,%s,%s)',
              (Supp_ID, fname,lname,address,contact,date_created))
    mydb.commit()

def add_new_cloth_data(cloth_id,item_code,name,description,size,Supp_ID,cost,date_created):
    c.execute('INSERT INTO clothes(cloth_id,item_code,name,description,size,Supp_ID,cost,date_created) VALUES (%s,%s,%s,'
              '%s,%s,%s,%s,%s)',
              (cloth_id,item_code,name,description,size,Supp_ID,cost,date_created))
    mydb.commit()


def view_only_user_names():
    c.execute('SELECT user_id,name FROM users')
    data = c.fetchall()
    return data

def view_only_supplier_names():
    c.execute('SELECT Supp_id,fname,lname FROM suppliers')
    data = c.fetchall()
    return data

def get_user(user_id):
    c.execute('SELECT * FROM users WHERE user_id="{}"'.format(user_id))
    data = c.fetchall()
    return data

def get_cloth(cloth_id):
    c.execute('SELECT * FROM clothes WHERE cloth_id="{}"'.format(cloth_id))
    data = c.fetchall()
    return data

def edit_user_data(new_user_id,new_name,new_username,new_password,new_gender,new_contact,new_address,new_amt_paid,user_id,name,username,pa
    c.execute("UPDATE users SET user_id=%s, name=%s, username=%s, password=%s, gender=%s, contact=%s, address=%s, amt_paid=%s WHERE "
              "user_id=%s and name=%s and username=%s and password=%s and gender=%s and contact=%s and address=%s and amt_paid=%s", (new_u
    mydb.commit()
```

```
def edit_cloth_data(new_cloth_id,new_item_code,new_name,new_description,new_size,new_Supp_ID,new_cost,new_date_created,cloth_id,item_code,
    c.execute("UPDATE clothes SET cloth_id=%s, item_code=%s, name=%s, description=%s, size=%s, Supp_ID=%s, cost=%s, date_created=%s WHERE
              "cloth_id=%s and item_code=%s and name=%s and description=%s and size=%s and Supp_ID=%s and cost=%s and date_created=%s", (n
    mydb.commit()


def delete_data(train_no):
    c.execute('DELETE FROM TRAIN WHERE user_id="{}"'.format(train_no))
    mydb.commit()


def delete_user(user_id):
    c.execute('DELETE FROM users WHERE user_id="{}"'.format(user_id))
    mydb.commit()

def view_only_cloth_names():
    c.execute('SELECT user_id,name FROM users')
    data = c.fetchall()
    return data

def view_all_clothes_data():
    c.execute('SELECT * FROM clothes')
    data = c.fetchall()
    return data

def sql_executor(raw_code):
    c.execute(raw_code)
    data = c.fetchall()
    return data


def queryboxx():
            st.title("SQL Query Box")
            col1,col2 = st.columns(2)
            with col1:
                raw_code = st.text_area("Write your SQL Code Here")
                submit_code = st.button("Execute")
            with st.expander("Table Info"):
                table_info = {'clothes':clothes,'users':users,'order_c':order_c,'suppliers':suppliers,'bill':bill,
                'manufacturers':manufacturers}
                st.json(table_info)

            with col2:
                if submit_code:
                    st.info("Query Submitted")
                    st.code(raw_code)

                    query_results = sql_executor(raw_code)

                    with st.expander("Results"):
                        st.write(query_results)

                    with st.expander("Pretty Table"):
                        query_df = pd.DataFrame(query_results)
                        st.dataframe(query_df)
def update_user():
    result = dtb.view_all_user_data()
    # st.write(result)
    df = pd.DataFrame(result, columns=['user_id','name','username','password','gender','contact','address','amt_paid'])
    with st.expander("Current Users"):
        st.dataframe(df)
    list_of_users = [(i[0],i[1]) for i in dtb.view_only_user_names()]
    selected_user = st.selectbox("User to Edit", list_of_users)
    selected_result = dtb.get_user(selected_user[0])
    # st.write(selected_result)
    if selected_result:
        user_id = selected_result[0][0]
        name = selected_result[0][1]
        username = selected_result[0][2]
        password = selected_result[0][3]
        gender = selected_result[0][4]
        contact = selected_result[0][5]
        address = selected_result[0][6]
        amt_paid = selected_result[0][7]


        # Layout of Create
        col1, col2= st.columns(2)
        with col1:
            new_user_id = st.text_input("user_id:")

            new_username = st.text_input("username:")
            new_contact = st.text_input("phone_no:")

            new_gender = st.selectbox("Gender",["Female","Male","Other"])
```

12. Conclusion

The cloth management system has been implemented keeping in mind the local cloth shop owners who aren't too well-versed with technology but have intelligent business strategies to expand their sales to wider horizon. With an aim to help them achieve this, the tools used to develop this system are not too sophisticated yet they impress to meet the necessary requirements. MySQL database is one of the simplest tool one can use for database management systems (Specially RDBMS). This project provides a convenient interface to connect to this backend database by just taking simple user input. Streamlit from python supports the basic necessities essential to provide a good front-end for this project. CRUD operations are implemented via this interface exhibiting its simplicity yet showing its capability to have good control over the database.

This project helped us understand the importance of Database management systems and its various features. Some of the learnings we took were:

- ✓ The planning that goes into implementing a project.
- ✓ The importance of proper planning and an organized methodology.
- ✓ The importance and application of the course UE20CS301: Database Management System

References

(If any)

The practice obtained from doing DBMS lab assignments helped immensely in developing this project.

UE20CS301 - DBMS Lab programs