SVKM's NMIMS
Mukesh Patel School of Technology Management & Engineering
A.Y. 2022 - 23
Course: Database Management Systems

Project Report

Program	BTECH CSBS	
Semester	IV	
Name of the Project:	RETAIL SHOPPING MANAGEMENT SYSTEM	
Details of Project Members		
Batch - B2	Roll No.	Name
	E044	VEDANTI PATKAR
	E049	LAKSH PURI
	E058	KWESHA SHAH
Date of Submission: 04-04-2023		

Contribution of each project Members:

Roll No.	Name	Contribution
E044	VEDANTI PATKAR	
E049	LAKSH PURI	
E058	KWESHA SHAH	

Note:

1. Create a readme file if you have multiple files
2. All files must be properly named (I004_DBMSProject)
3. Submit all relevant files of your work (Report, all SQL files, Any other files)
4. **Plagiarism is highly discouraged (Your report will be checked for plagiarism)**

Rubrics for the Project evaluation:

- Innovative Ideas and self learning (5 Marks) Idea should not be regular such as Hotel, Library Management system etc.
- Implementation and Design (10 Marks) It includes ER model, Relational model and Normalization of tables.
- Project Demonstration and Viva (5 Marks)

Project Report

RETAIL MANAGEMENT SYSTEM

by

Vedanti Patkar, Roll number:E044

Laksh Puri, Roll number:E049

Kwesha Shah, Roll number:E058

Course: DBMS

AY: 2022-23

Table of Contents

I. Storyline

The Database chosen by us is the retail management system. From a Database Design point of view, the following requirements were imperative for the retail management system :

- Table Creation : We needed to formulate various tables and attributes required for this system. Not a very easy task since primarily they need to be in Third Normal Form as well be suitable for all the query requirements. In our system, we have seven tables where every table has a list of attributes and primary keys defined.

- Data Population : To solve various queries, this is a vital step in Database designing and management. In our case, this data is in the form of customer info, payment and product data and staff and administration information.

- Creating SQL queries : It would be completely useless if our system was incompetent to solve various queries that show up in the day to day working of a retail management system. Our system has been efficiently and meticulously designed in a way to tackle every possible query you throw at its way. Solutions to the same are provided in seconds. These queries can be used to generate sales_reports, retrieve the top selling product and customer information too. A completely list has been provided below and will be discussed shortly.
- Security : Another key component when designing, its very important that our information is secure. This point has been further strengthened with the use of grants and views in our database.
- User Interface : Making a simple and easy to understand system is highly beneficial. Our customers manage various product and customer details and manoeuvring their way through the system with ease is the name of the game.

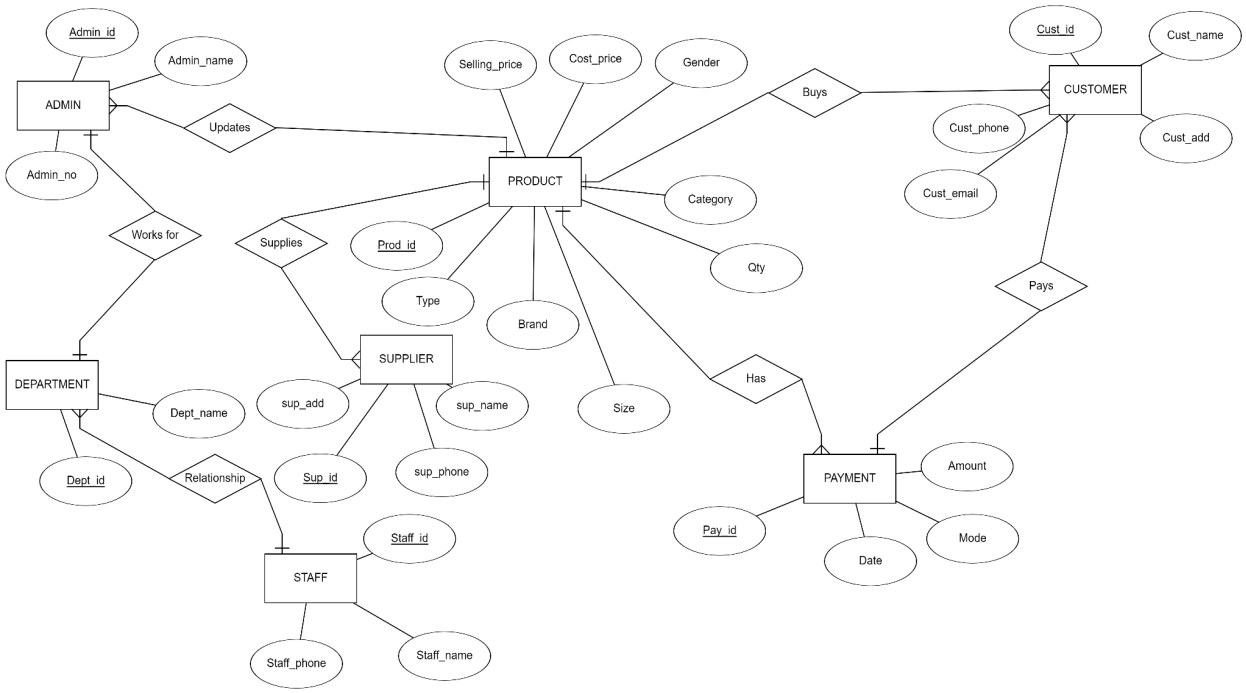
Overall, this project aims to provide a comprehensive solution for managing the retail business's data and processes, helping the businesses to make informed decisions and improve efficiency.

II. Components of Database Design

admin - (admin_id,name,admin_no,dept_id*)
customer -(cust_id,cust_name,cust_add,cust_phone,cust_email)
payment - (pay_id, pay_amt,pay_mode,pay_date,cust_id*,prod_id*)
product
(prod_id,prod_type,prod_category,cost_price,selling_price,prod_gender,prod_size,prod_brand,qty,sup_id*)
supplier - (sup_id,sup_name,sup_phn,sup_add)
staff - (staff_id,staff_name,staff_phn,dept_id*)
department - (dept_id,dept_name)

All highlighted attributes are primary keys whilst the starred attributes are foreign keys in each table.

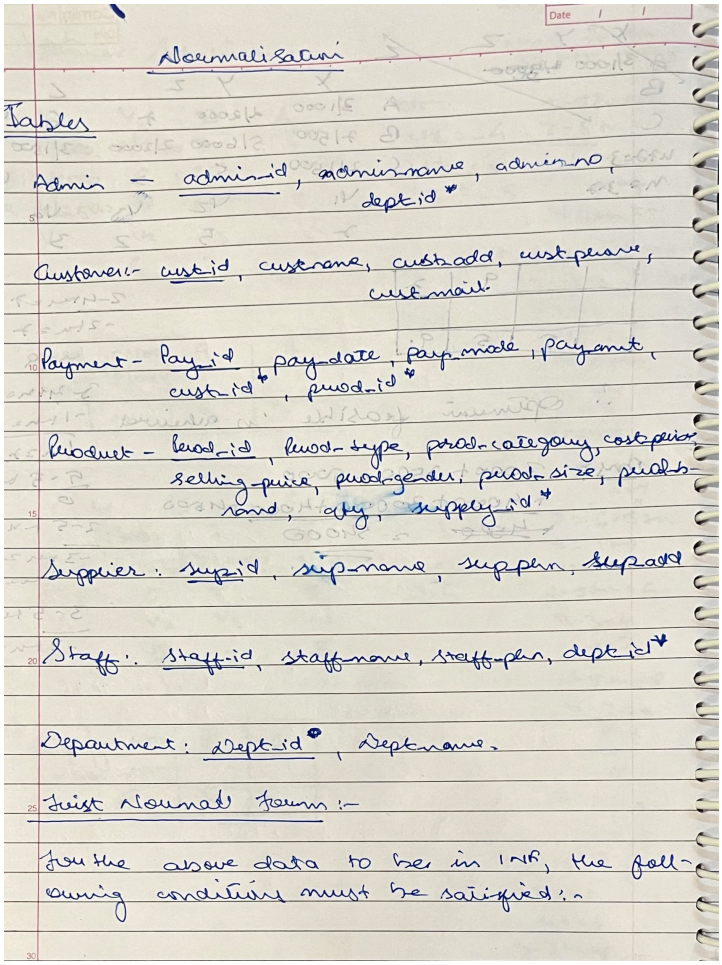
III. Entity Relationship Diagram

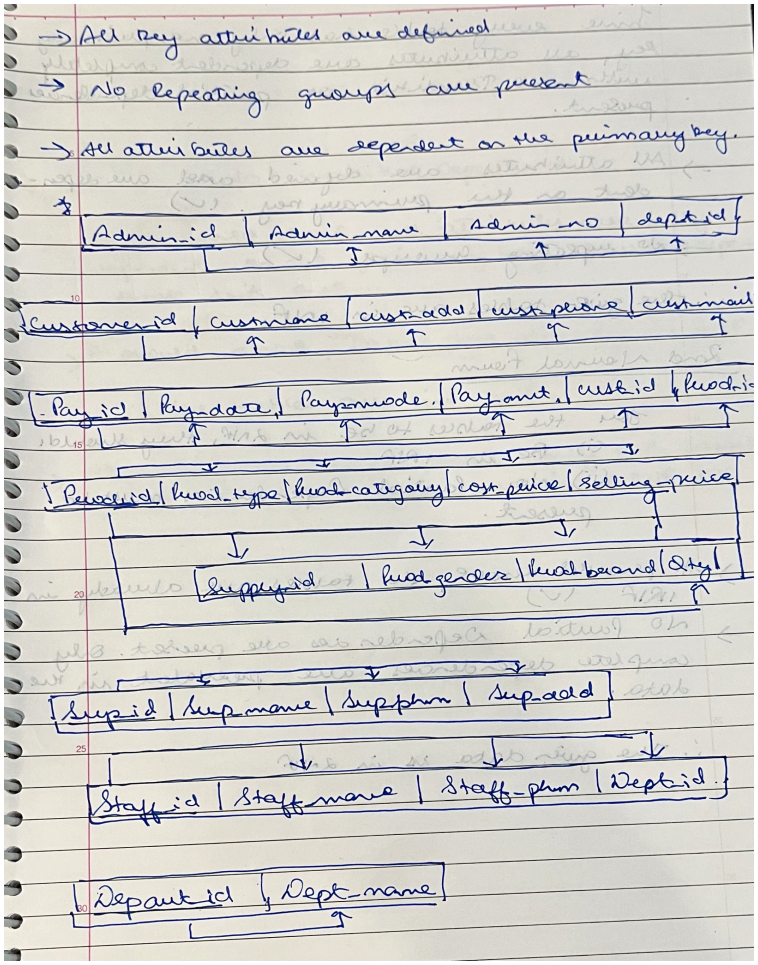


IV. Relational Model

admin - (admin_id,name,admin_no,dept_id*)
customer -(cust_id,cust_name,cust_add,cust_phone,cust_email)
payment - (pay_id, pay_amt,pay_mode,pay_date,cust_id*,prod_id*)
product
(prod_id,prod_type,prod_category,cost_price,selling_price,prod_gender,prod_size,prod_brand,qty,sup_id*)
supplier - (sup_id,sup_name,sup_phn,sup_add)
staff - (staff_id,staff_name,staff_phn,dept_id*)
department - (dept_id,dept_name)

V. Normalization





Since every table has only one primary key, all attributes are dependent completely with no transitive or partial dependencies present.

→ All attributes are defined and are dependent on the primary key; (✓)

→ No repeating Groups (✓)

∴ The given tables are in 1NF.

2nd Normal Form

For the tables to be in 2NF, they should:

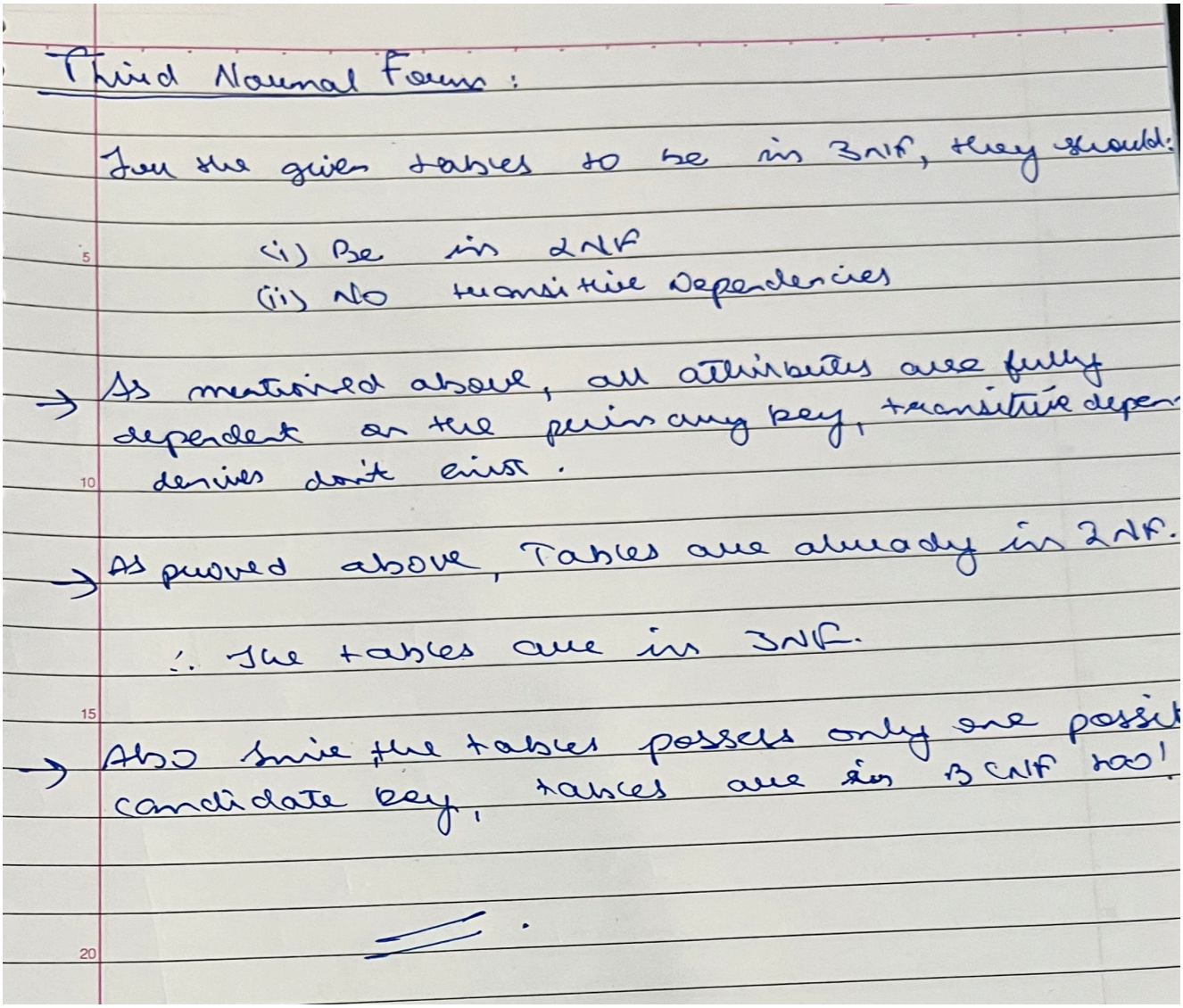
(i) Be in 1NF

(ii) No Partial Dependencies must be present.

→ As proved above, the tables are already in 1NF (✓)

→ NO Partial Dependencies are present. Only complete dependencies are prevalent in the data.

∴ The given data is in 2NF



VI. SQL Queries

#1)Find the details of the customers from Borivali who bought black jeans

```
Select * From customer natural inner join payment
Where cust_add = 'thane' and prod_id in ( select prod_id
From product
Where prod_category = 'Jeans' And prod_color = 'Black' );
```

cust_id	cust_name	cust_phone	cust_email	cust_add	pay_id	pay_mode	pay_amt	pay_date	prod_id
1104	Pallavi	9785295287	pallavi4@gmail.com	thane	2003	cash	2999	2021-04-20	1002

#2)Find the details of all the employees who work in HR
Select *
From staff, department
Where staff.dept_id = department.dept_id And dept_name = 'HR' ;

staff_id	staff_name	staff_phn	dept_id	dept_id	dept_name
505	Aryan	9632552369	3333	3333	HR
707	Akhil	9632112365	3333	3333	HR

#3)Find the details of all suppliers from Khar who supply shirts. Use ON keyword
Select * From supplier inner join product
On supplier.sup_id = product.sup_id
where prod_category = 'shirt' and sup_add='khar';

sup_id	sup_name	sup_phone	sup_add	...	prod_catego...	prod_type	prod_color	prod_brand	prod_gender	prod_size	cost_price	selling_pri...	qty	sup_id
1616	mohan	9632552369	khar	10	Shirt	Oversized	Red	Raymond	Male	xs	2299	3599	10	1616

#4)Find all the customers who spent above 2500/- Use nested queries
Select *
From customer
Where cust_id in (select cust_id From payment
Where pay_amt > 2500) ;

cust_id	cust_name	cust_phone	cust_email	cust_add
1101	Mohini	9009590095	mohini1@gmail.com	Mulund
1104	Pallavi	9785295287	pallavi4@gmail.com	thane
1105	Kamal	9632552369	kamal5@yahoo.com	andheri
1107	Jyoti	9632112365	jyoti6@yahoo.com	worli
1109	Sandeep	9827382738	sandeep9@hotmil.com	andheri
1102	Radhika	9550595505	radhika2@gmail.com	Borivali
1103	Payal	9660696606	payal3@gmail.com	kandivali
1108	Jayesh	8080512236	jayesh8@gmail.com	ghatkopar
NULL	NULL	NULL	NULL	NULL

#5)Provide the details of the admins whose names' second last letter is 'a'.

```
Select *
From admin
Where admin_name like '%a_' ;
```

admin_id	admin_name	admin_phn	dept_id
3	Abhay	9660696606	2222
5	Sanjay	9632552369	1111
6	Soham	9102545429	5555
7	Raj	9632112365	5555
9	Hiral	9827382738	2222
NULL	NULL	NULL	NULL

#6)

```
select prod_category,count(prod_color)
from product
where prod_color='red'
group by prod_category
having count(prod_color)>1;
```

prod_catego...	count(prod_col...
Tops	2
Shirt	2

#7)Find the number of customers who still use Yahoo.
Select count(cust_email) as 'No of users' From customer
Where cust_email like '%yahoo%';

No of users
3

#8)Find the customer who spent the maximum amount in one outing and average amount spent by each customer
Select cust_name, max(pay_amt), avg(pay_amt) as 'Average Value' From customer, payment
Where customer.cust_id = payment.cust_id
Group by cust_name
Order by max(pay_amt) desc;

cust_name	max(pay_amt)	Average Value
Radhika	3599	3099.0000
Jayesh	3599	3599.0000
Mohini	2999	2499.0000
Pallavi	2999	2999.0000
Payal	2999	2999.0000
Kamal	2599	2599.0000
Jyoti	2599	2599.0000
Sandeep	2599	2599.0000

#9)Using the ‘distinct’ keyword, find how many departments are present
Select count(distinct(dept_id)) From admin ;

count(distinct(dept_id))
5

#10)Find all these customers who pay by cash. Use nested queries
Select cust_name, cust_id
From customer
Where cust_id in (select cust_id From payment
Where pay_mode = 'cash') ;

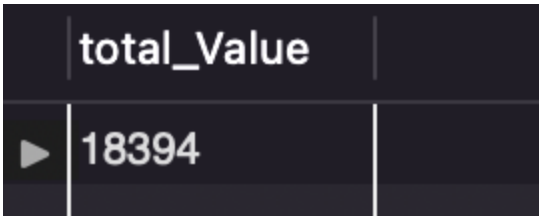
cust_name	cust_id
Mohini	1101
Pallavi	1104
Sandeep	1109
Radhika	1102
Payal	1103
Jayesh	1108
NULL	NULL

#11)List in ascending order the no products sold (low -> high)
Select prod_id, prod_category, prod_type, prod_color, prod_brand, prod_size, prod_gender,
Count(prod_id) as 'Sales_Figures'
From product natural inner join payment Group by prod_id
Order by Sales_Figures ;

prod_id	prod_category	prod_type	prod_color	prod_brand	prod_size	prod_gender	Sales_Figures
1000	Jeans	Skinny	Blue	pepe	xs	Male	1
1002	Jeans	MomJeans	Black	sixth element	s	Female	1
1003	Jeans	boyfriend	Blue	LeeCooper	m	Female	1
1007	Tops	Casual	Blue	Ajile	xs	Female	1
1008	Tops	Formal	Red	H&M	xs	Female	1
1009	Tops	Occasional	Pink	GlobalDesi	l	Female	1
1017	Shirt	half-sleeve	Red	Tommy Hilfiger	s	Female	1
1018	Shirt	Formal	White	Raymond	xl	Male	1
1019	Shirt	full-sleeve	Orange	PeterEngland	m	Female	1
1001	Jeans	Skinny	Blue	levis	xl	Male	2

#12)Retrieve the total amount of sales between the 1st of Jan 2021 and 15th Jan 2022

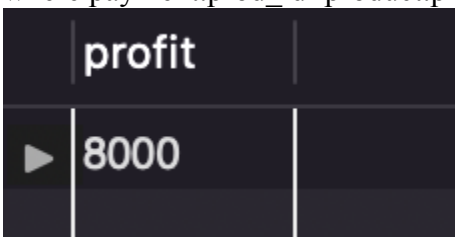
```
Select sum( pay_amt ) as 'total_Value'
From payment
Where pay_date between '2021-01-01' and '2022-01-15' ;
```



total_Value
18394

#13)find the total profit for 2 years

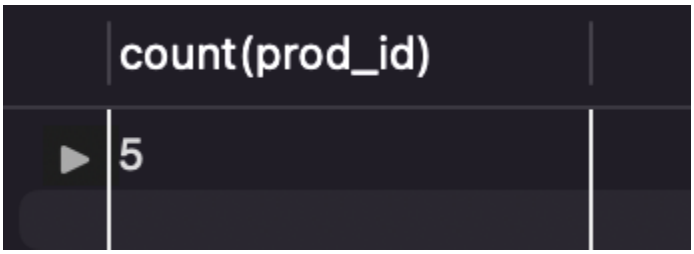
```
select sum(pay_amt-cost_price) as profit
from payment, product
where payment.prod_id=product.prod_id and pay_date between '2021-01-01' and '2023-01-01' ;
```



profit
8000

#14)find total jeans sold

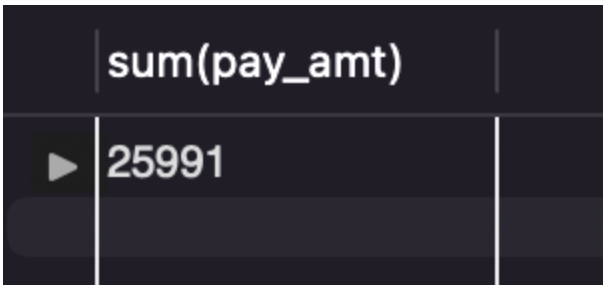
```
select count(prod_id)
from payment natural inner join product
where prod_category='jeans';
```



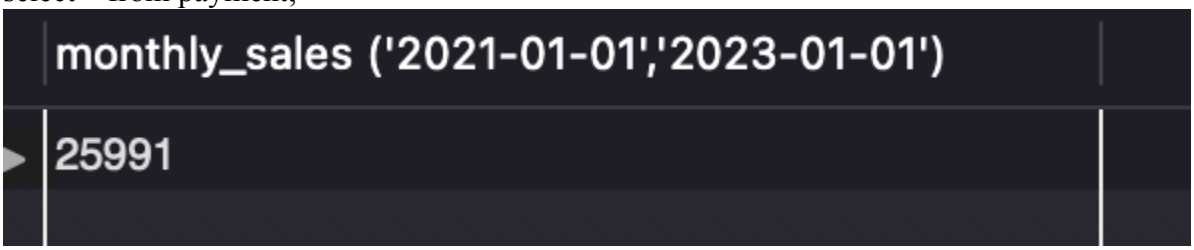
count(prod_id)
5

#15)using normal query to find sale between two dates

```
Select sum(pay_amt)
From payment
Where pay_date between '2021-01-01' and '2023-01-01' ;
```



#16)using functions to find sales between two dates
delimiter //
Create function monthly_sales(start_date date , end_date date)
Returns numeric(10)
deterministic
Begin
Declare total numeric(10) ;
#Total is variable where the summed up value will be stored.
Select sum(pay_amt) into total
From payment
Where pay_date between start_date and end_date;
Return total;
End //
select monthly_sales ('2021-01-01','2023-01-01');
select * from payment;



#17)dense rank:
SELECT
prod_id,prod_category,prod_type,prod_color,prod_brand ,prod_gender,prod_size,cost_price,selli
ng_price,qty,sup_id,
DENSE_RANK() OVER (ORDER BY selling_price) dens_rank
FROM product;

prod_id	prod_category	prod_type	prod_color	prod_brand	prod_gender	prod_size	cost_price	selling_pri...	qty	sup_id	dens_rank
1012	Tshirt	Casual	White	Zara	Female	xs	999	1599	30	1818	1
1000	Jeans	Skinny	Blue	pepe	Male	xs	1499	1999	18	1212	2
1013	Tshirt	V-neck	Purple	H&M	Male	m	1499	1999	20	1919	2
1003	Jeans	boyfriend	Blue	LeeCooper	Female	m	1899	2599	10	1212	3
1005	Jeans	straight	Black	h&m	Female	m	1999	2599	20	1212	3
1007	Tops	Casual	Blue	Ajile	Female	xs	1499	2599	25	1919	3
1008	Tops	Formal	Red	H&M	Female	xs	1599	2599	30	2121	3
1009	Tops	Occasional	Pink	GlobalDesi	Female	l	1699	2599	10	3131	3
1010	Tops	Party Wear	Yellow	HnM	Female	s	1799	2599	5	1414	3
1011	Tops	Evening	Red	LeeCooper	Female	xl	1899	2599	25	1717	3
1001	Jeans	Skinny	Blue	levis	Male	xl	2299	2999	15	1212	4
1002	Jeans	MomJeans	Black	sixth elem...	Female	s	1799	2999	10	1212	4
1004	Jeans	bell bottom	Blue	zara	Male	xs	2299	2999	15	1212	4
1018	Shirt	Formal	White	Raymond	Male	xl	2399	2999	10	1414	4
1019	Shirt	full-sleeve	Orange	PeterEngl...	Female	m	2199	2999	5	1515	4
1006	Jeans	high rise	Blue	durfy	Female	l	2099	3599	25	1818	5
1014	Tshirt	collar-neck	LimeYellow	Tommy Hi...	Female	l	2599	3599	15	2121	5
1015	Tshirt	half-sleeve	Blue	Zara	Female	xs	2699	3599	20	3131	5
1016	Tshirt	full-sleeve	Brown	Raymond	Male	xs	2799	3599	25	2121	5
1017	Shirt	half-sleeve	Red	Tommy Hi...	Female	s	2899	3599	30	3131	5
1020	Shirt	Oversized	Red	Raymond	Male	xs	2299	3599	10	1616	5

#18)percent rank:

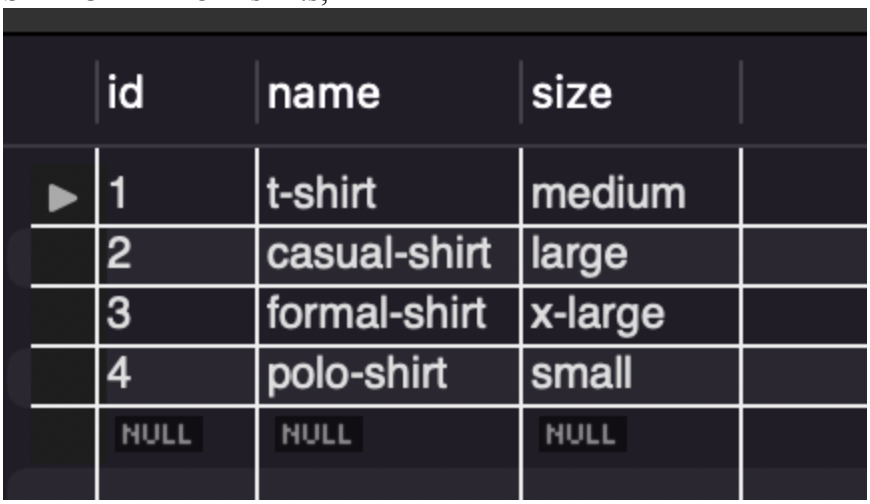
```
SELECT
prod_id,prod_category,prod_type,prod_color,prod_brand ,prod_gender,prod_size,cost_price,selling_price,qty,sup_id,selling_price-cost_price as profit,
PERCENT_RANK() OVER(PARTITION BY prod_category ORDER BY selling_price-cost_price)my_rank
FROM product;
```

prod_id	prod_category	prod_type	prod_color	prod_brand	prod_gender	prod_size	cost_price	selling_pri...	qty	sup_id	profit	my_rank
1000	Jeans	Skinny	Blue	pepe	Male	xs	1499	1999	18	1212	500	0
1005	Jeans	straight	Black	h&m	Female	m	1999	2599	20	1212	600	0.16666666666666666
1001	Jeans	Skinny	Blue	levis	Male	xl	2299	2999	15	1212	700	0.3333333333333333
1003	Jeans	boyfriend	Blue	LeeCooper	Female	m	1899	2599	10	1212	700	0.3333333333333333
1004	Jeans	bell bottom	Blue	zara	Male	xs	2299	2999	15	1212	700	0.3333333333333333
1002	Jeans	MomJeans	Black	sixth element	Female	s	1799	2999	10	1212	1200	0.8333333333333334
1006	Jeans	high rise	Blue	durfy	Female	l	2099	3599	25	1818	1500	1
1018	Shirt	Formal	White	Raymond	Male	xl	2399	2999	10	1414	600	0
1017	Shirt	half-sleeve	Red	Tommy Hilfiger	Female	s	2899	3599	30	3131	700	0.3333333333333333
1019	Shirt	full-sleeve	Orange	PeterEngland	Female	m	2199	2999	5	1515	800	0.6666666666666666
1020	Shirt	Oversized	Red	Raymond	Male	xs	2299	3599	10	1616	1300	1
1011	Tops	Evening	Red	LeeCooper	Female	xl	1899	2599	25	1717	700	0
1010	Tops	Party Wear	Yellow	HnM	Female	s	1799	2599	5	1414	800	0.25
1009	Tops	Occasional	Pink	GlobalDesi	Female	l	1699	2599	10	3131	900	0.5
1008	Tops	Formal	Red	H&M	Female	xs	1599	2599	30	2121	1000	0.75
1007	Tops	Casual	Blue	Ajile	Female	xs	1499	2599	25	1919	1100	1
1013	Tshirt	V-neck	Purple	H&M	Male	m	1499	1999	20	1919	500	0
1012	Tshirt	Casual	White	Zara	Female	xs	999	1599	30	1818	600	0.25
1016	Tshirt	full-sleeve	Brown	Raymond	Male	xs	2799	3599	25	2121	800	0.5
1015	Tshirt	half-sleeve	Blue	Zara	Female	xs	2699	3599	20	3131	900	0.75
1014	Tshirt	collar-neck	LimeYellow	Tommy Hilfiger	Female	l	2599	3599	15	2121	1000	1

#19)auto increment

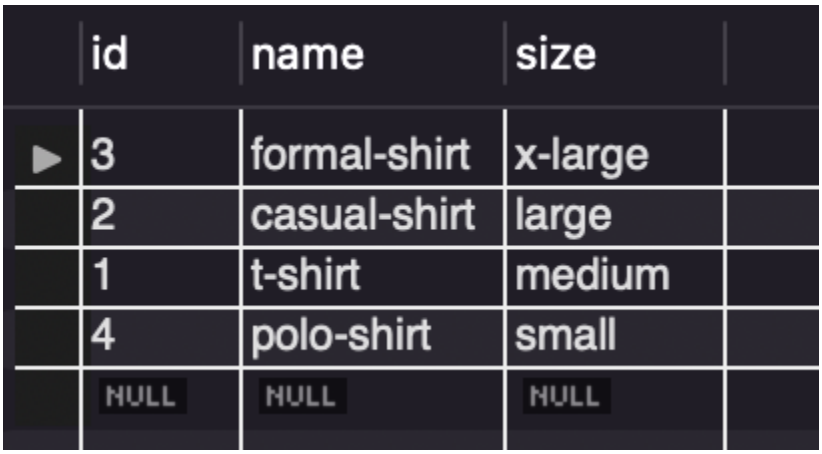
```
CREATE TABLE shirts (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(35),
    size ENUM('small', 'medium', 'large', 'x-large')
);
```

```
INSERT INTO shirts(id, name, size)
VALUES (1,'t-shirt', 'medium'),
(2, 'casual-shirt', 3),
(3, 'formal-shirt', 4),
(4, 'polo-shirt', 'small');
SELECT * FROM shirts;
```



id	name	size
1	t-shirt	medium
2	casual-shirt	large
3	formal-shirt	x-large
4	polo-shirt	small
NULL	NULL	NULL

```
SELECT * FROM shirts ORDER BY  size DESC;
```



id	name	size
3	formal-shirt	x-large
2	casual-shirt	large
1	t-shirt	medium
4	polo-shirt	small
NULL	NULL	NULL

```
#20)Prepared Statement
PREPARE stmt1 FROM
'SELECT staff_name,dept_id FROM staff
WHERE dept_id = ?';
set @dept_id=1111 ;
EXECUTE stmt1 USING @dept_id;
DEALLOCATE PREPARE stmt1;
```

staff_name	dept_id
Tanish	1111
Rushil	1111

Stored Procedure

```
DELIMITER $$
drop procedure if exists get_detail;
create procedure get_detail()
BEGIN
   SELECT count(prod_id) FROM product WHERE selling_price > 2000;
END $$
CALL get_detail();
```

Result Grid | Filter Rows:

count(prod_id)
18

Passing Parameter

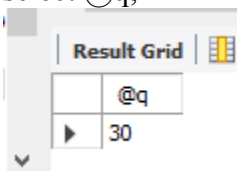
```
DELIMITER &&
drop procedure if exists few_supplier;
CREATE PROCEDURE few_supplier (IN var1 INT)
BEGIN
   SELECT * FROM supplier LIMIT var1;
END &&
DELIMITER ;
call few_supplier(5);
```

Result Grid | Filter Rows: | Export:

sup_id	sup_name	sup_phone	sup_add
1212	rajesh	9009590095	jogeshwari
1313	jay	9550595505	khar
1414	mahesh	9660696606	dombivali
1515	nitin	9785295287	kandivali
1616	mohan	9632552369	khar

Passing with out Parameter

```
DELIMITER &&
drop procedure if exists max_qty;
CREATE PROCEDURE max_qty (OUT highestqty INT)
BEGIN
   SELECT MAX(qty) INTO highestqty FROM product;
END &&
DELIMITER ;
call max_qty(@q);
select @q;
```



Update Quantity

```
delimiter $$
drop procedure if exists pr_buy_products;
create procedure PR_BUY_PRODUCTS()
begin declare pr_prod_id  numeric(4);
select prod_id
into pr_prod_id
from payment
where pay_id='2000';
update product
set qty = (qty - 1)
where prod_id=pr_prod_id;
end $$
select * from payment;
select * from product;
call pr_buy_products();
```

before

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

prod_id	prod_category	prod_type	prod_color	prod_brand	prod_gender	prod_size	cost_price	selling_price	qty	sup_id
1000	Jeans	Skinny	Blue	pepe	Male	xs	1499	1999	19	1212
1001	Jeans	Skinny	Blue	levis	Male	xl	2299	2999	15	1212
1002	Jeans	MomJeans	Black	sixth element	Female	s	1799	2999	10	1212
1003	Jeans	boyfriend	Blue	LeeCooper	Female	m	1899	2599	10	1212
1004	Jeans	bell bottom	Blue	zara	Male	xs	2299	2999	15	1212
1005	Jeans	straight	Black	h&m	Female	m	1999	2599	20	1212
1006	Jeans	high rise	Blue	durfy	Female	l	2099	3599	25	1818
1007	Tops	Casual	Blue	Ajile	Female	xs	1499	2599	25	1919
1008	Tops	Formal	Red	H&M	Female	xs	1599	2599	30	2121

product 4

after

prod_id	prod_category	prod_type	prod_color	prod_brand	prod_gender	prod_size	cost_price	selling_price	qty	sup_id
1000	Jeans	Skinny	Blue	pepe	Male	xs	1499	1999	18	1212
1001	Jeans	Skinny	Blue	levis	Male	xl	2299	2999	15	1212
1002	Jeans	MomJeans	Black	sixth element	Female	s	1799	2999	10	1212
1003	Jeans	boyfriend	Blue	LeeCooper	Female	m	1899	2599	10	1212
1004	Jeans	bell bottom	Blue	zara	Male	xs	2299	2999	15	1212
1005	Jeans	straight	Black	h&m	Female	m	1999	2599	20	1212
1006	Jeans	high rise	Blue	durfy	Female	l	2099	3599	25	1818
1007	Tops	Casual	Blue	Ajile	Female	xs	1499	2599	25	1919
1008	Tops	Formal	Red	H&M	Female	xs	1599	2599	30	2121

#Trigger

```
create table Message(
text varchar(100),
created_at date );

create table Department_news(
dept_idNUMERIC(4) PRIMARY KEY,
dept_name  VARCHAR(15) NOT NULL,
updated_at DATE);

create table cust_history(
cust_id          NUMERIC(4) PRIMARY KEY,
cust_name        VARCHAR(15)  NOT NULL,
cust_phone       NUMERIC(10)  NOT NULL,
cust_email  varchar(20),
cust_add         VARCHAR(25)  NOT NULL,
deleted_at  DATE);
```

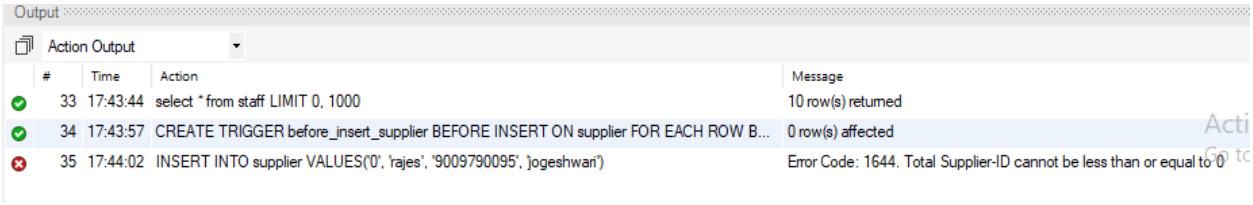
#BEFORE INSERT VALUE

```
set sql_safe_updates=0
Delimiter //
CREATE TRIGGER before_insert_supplier
BEFORE INSERT ON supplier
```

```
FOR EACH ROW
BEGIN
   IF new.sup_id = 0 THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Total Supplier-ID cannot be less than or equal to 0';
   END IF;
END//

INSERT INTO supplier VALUES('0', 'rajes', '9009790095', 'jogeshwari');

INSERT INTO supplier VALUES('15', 'rajes', '9009790095', 'jogeshwari');
```

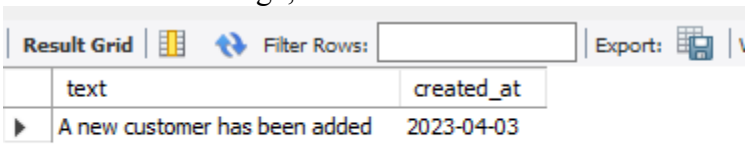


#AFTER INSERT VALUE

```
Delimiter //
CREATE TRIGGER after_insert_customer
AFTER INSERT ON customer
FOR EACH ROW
BEGIN
   INSERT INTO message (text, created_at)
   VALUES ('A new customer has been added', current_date());
END//


insert into customer values ('1121', 'Mohani', '9025590095', 'mohani1@gmail.com', 'Mulund');

select * from message;
```



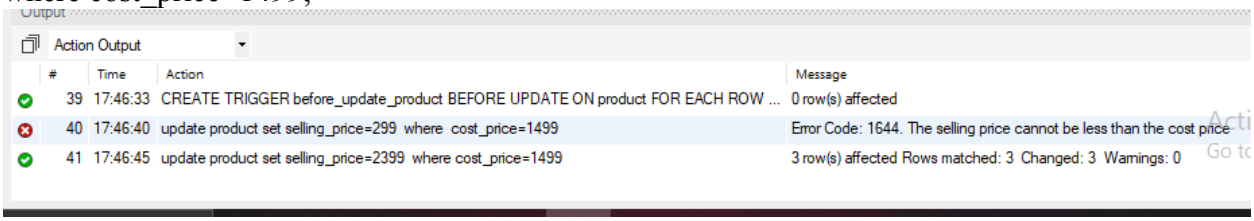
#BEFORE UPDATE VALUE

```
Delimiter //
CREATE TRIGGER before_update_product
BEFORE UPDATE ON product
```

```
FOR EACH ROW
BEGIN
   # Check if the cost price is less than the selling price
   IF NEW.cost_price > NEW.selling_price THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'The selling price cannot be less than the cost price';
   END IF;
END//

update product
set selling_price=299
where  cost_price=1499;

update product
set selling_price=2399
where cost_price=1499;
```



#AFTER UPDATE VALUE

```
Delimiter //
create TRIGGER after_update_department
AFTER UPDATE ON department
FOR EACH ROW
BEGIN
   INSERT INTO department_news(dept_id , dept_name , updated_at)
   values(NEW.dept_id , OLD.dept_name , NOW());
END//

update department
set dept_name='sale'
where dept_name='sales';

select * from Department;
select * from Department_news;
```

Result Grid | Filter Rows:

dept_id	dept_name	updated_at
1111	Sales	2023-04-03
NULL	NULL	NULL

#BEFORE DELETE VALUE

```
Delimiter //
CREATE TRIGGER before_delete_staff
BEFORE DELETE ON staff
FOR EACH ROW
BEGIN
   DECLARE staff_count INT;
   SELECT COUNT(*) INTO staff_count FROM staff WHERE dept_id = OLD.dept_id;

   IF (staff_count <= 1) THEN
         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot delete the last staff member in a department';
   END IF;
END//
```

```
delete from staff
where staff_id=202;


delete from staff
where staff_id=808;

delete from staff
where staff_id=110;
```

Action Output

#	Time	Action	Message
46	17:48:14	delete from staff where staff_id=202	1 row(s) affected
47	17:48:18	delete from staff where staff_id=808	1 row(s) affected
48	17:48:21	delete from staff where staff_id=110	Error Code: 1644. Cannot delete the last staff member in a department

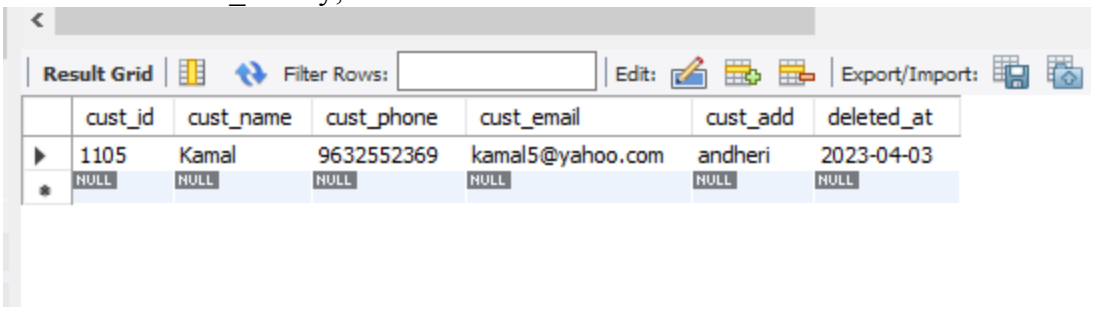
#AFTER DELETE VALUE

```
Delimiter //
CREATE TRIGGER after_delete_customer
AFTER DELETE ON customer
```

```
FOR EACH ROW
BEGIN
   INSERT INTO cust_history (cust_id, cust_name, cust_phone,cust_email, cust_add,deleted_at)
            VALUES (OLD.cust_id, OLD.cust_name, OLD.cust_phone ,
OLD.cust_email,OLD.cust_add , NOW());
END//


DELETE FROM customer where cust_email = 'kamal5@yahoo.com';

select * from cust_history;
```



cust_id	cust_name	cust_phone	cust_email	cust_add	deleted_at
1105	Kamal	9632552369	kamal5@yahoo.com	andheri	2023-04-03
NULL	NULL	NULL	NULL	NULL	NULL

#Grant all Privilege

```
CREATE USER 'william'@'localhost' IDENTIFIED BY 'will123456';
SELECT USER FROM MySQl.user;
GRANT ALL PRIVILEGES ON * . * TO 'william'@'localhost' ;
FLUSH PRIVILEGES;
SHOW GRANTS FOR 'william'@'localhost';
REVOKE all privileges on *.* FROM 'william'@'localhost';
DROP USER 'william'@'localhost';
```

#Views :
Create a view for the promotion team to send discount coupons to the first six customers

```
Create view customer_info as (
Select cust_name, cust_mail, cust_phn
From customer
```

Where cust_id between 1100 and 1106) ;

cust_name	cust_email	cust_phone
Mohini	mohini1@gmail.com	9009590095
Radhika	radhika2@gmail.com	9550595505
Payal	payal3@gmail.com	9660696606
Pallavi	pallavi4@gmail.com	9785295287
Kamal	kamal5@yahoo.com	9632552369
Kashyap	kashyap6@yahoo.com	9102545429

#Query pertaining to the view to retrieve info of all customers that use Gmail:

Select *
From customer_info
Where cust_mail like '%gmail%' ;

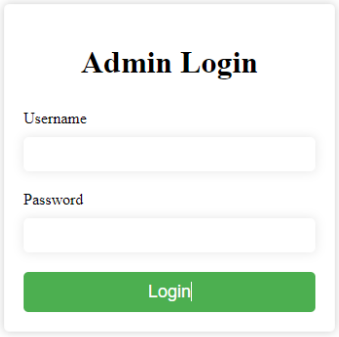
cust_name	cust_email	cust_phone
Mohini	mohini1@gmail.com	9009590095
Radhika	radhika2@gmail.com	9550595505
Payal	payal3@gmail.com	9660696606
Pallavi	pallavi4@gmail.com	9785295287

VI. Project demonstration

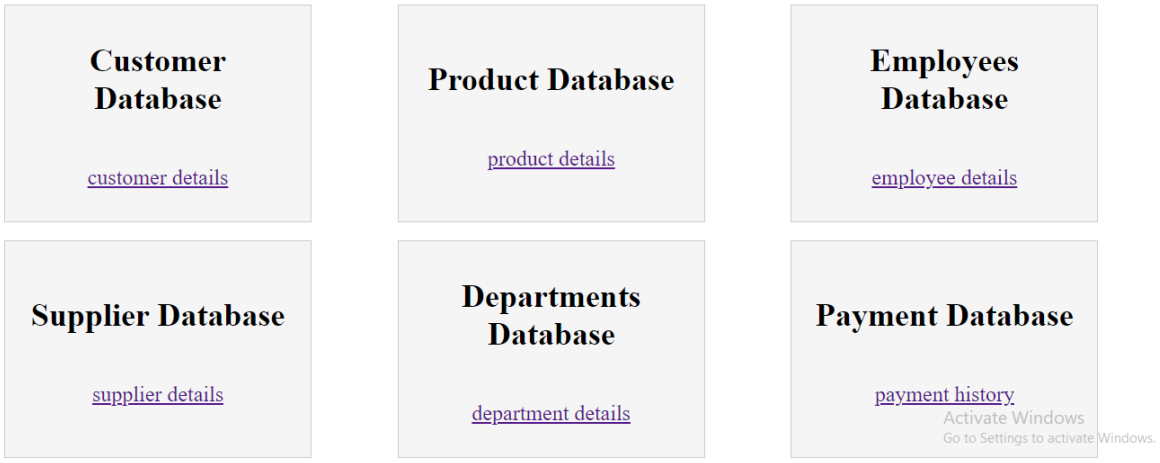
- Tools/software/ libraries used

- MySQL Workbench
- HTML and CSS
- ERD plus
- excel

- GUI



Our Database Collection



Product Database

View Database Table

View the current database table of products.

Add New Product

Add a new product to the database table.

TO ENTER A NEW PRODUCT

PRODUCT_ID :

SELECT A PRODUCT CATEGORY :

- ○ **JEANS**
- ○ **TOPS**
- ○ **TSHIRT**
- ○ **SHIRT**

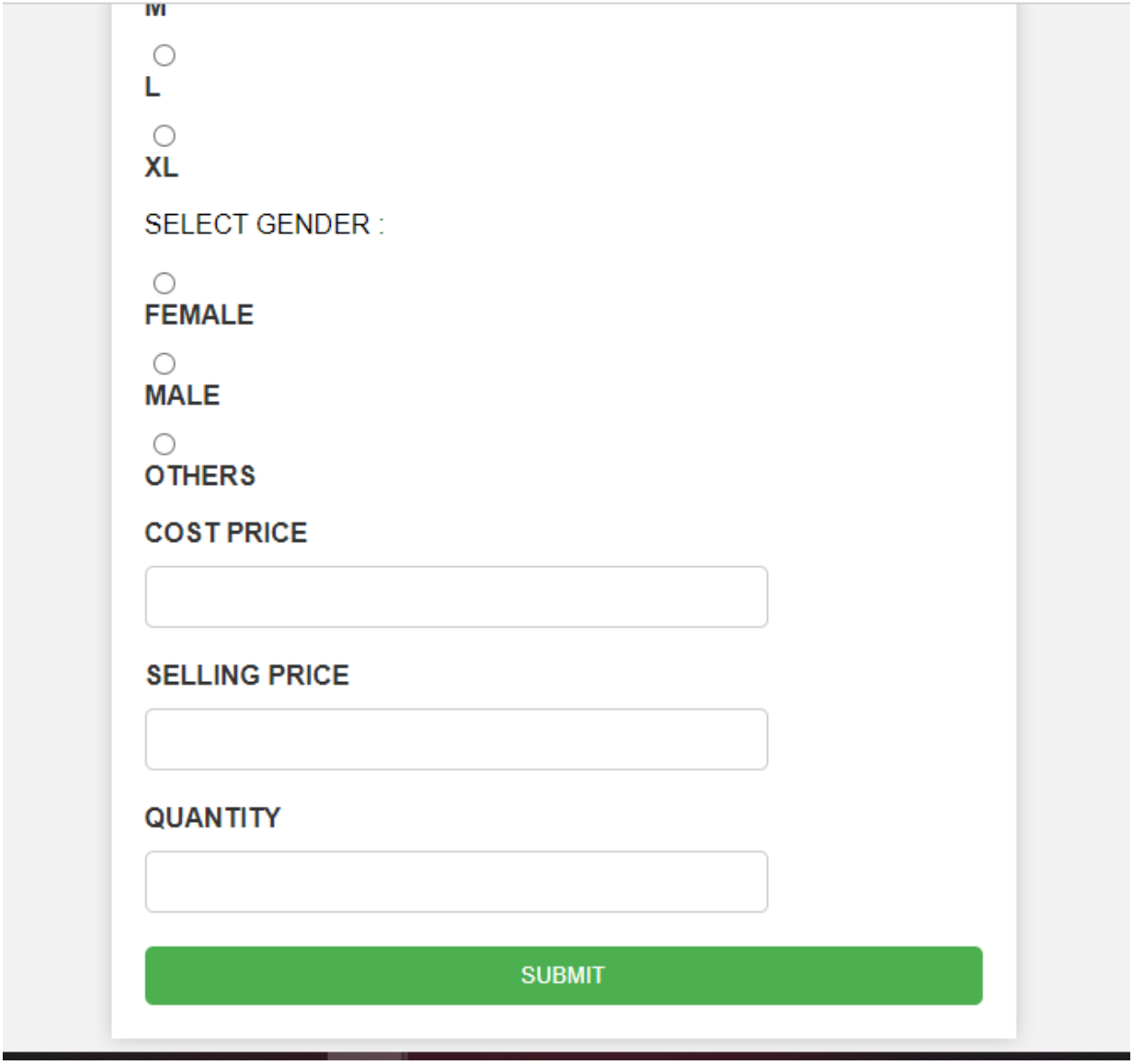
PRODUCT TYPE

Enter product type

PRODUCT BRAND

Enter product brand

PRODUCT COLOR



Added a new product successfully!

VII. Learning from the Project

This project was highly beneficial to us as a team. Some of our learnings are listed below :

- This project taught us the importance of team spirit ; Together Everyone Achieves More.
- Through this project, our overall confidence and morale is boosted towards working in MySQL.
- Through this project, we have understood the basic concepts of database designing ; the same we have implemented in our project.
- This project has given us a head start in the ideology of database designing and management.

VIII. Challenges Faced

- As a team , we found multiple issues in designing the basic tables and attributes.
- Our Designed tables weren't meeting the requirements of various normal forms and query tackling concepts.
- We had to scrap the entire system and build a fresh one again with new ER diagrams so that our project can meet industry requirements.
- When writing queries too, we were getting multiple syntax errors.
- Moreover, the new concepts were a tough nut to crack but with our continuous effort and diligence, we successfully overcame it.

IX. Conclusion

In conclusion, we our, primarily, very thankful for giving us this opportunity to execute such projects. As a Team, we managed to put across an efficient database designed not

only to manage various queries, but also a system that swiftly provides an easy solution to manage the retail management industry by living up to industry standards.