ONLINE SHOPPING OF CLOTHES

REVIEW REPORT

Submitted by

N.ANAND VENKATA SUBBARAJU(19BCE0264)
NADIMPALLI NARASHIMHA RAJU(19BCE2247)
P.SAI KARTHIK REDDY(19BCE0077)

Prepared For

DATABASE MANAGEMENT SYSTEM(CSE2004)

PROJECT COMPONENT

Submitted To

Dr.Nithya S

Associate Professor

School of Computer Science and Engineering

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Table of Content

ABSTRACT:

As we know that,Now-a-days everything is getting connected to internet.Internet on one hand has established a connection between customers and products available in stores.The online shopping of clothes enables vendors to set up online sites for selling various types of products.Customers can easily browse their product through online.The online shopping system of clothes consists of number of products and the type of brands,and number of brands and pieces available in the stock that we use in our day-to-day life.It presents an online display of clothes and an associated delivery window for items (type of brand) selected by the custuomer.Successive items selected for purchase are placed into virtual shopping cart until customer completes their shopping.In this project we mainly discuss about the online shopping of clothes.

1-INTRODUCTION:

- Online Shopping of clothes is the process consumer go through to purchase clothes over the internet. An online shopping of clothes evokes the physical analogy of buying products at a bricks-and mortar retailer.
- Online shopping of clothes is a type of electronic commerce used for business-to-business and business-to-consumer transactions.Online shopping of clothes system provides a virtual cart for each individual for holding items selected for purchase. Virtual shopping carts may be examined at any time, and their contents can be edited or deleted at the option of the customer.
- Once the customer decides to submit a purchase order, the customer may proceed to the payment window where he/she can pay by submitting his/her card details or he/she can pay when the order will be delivered i.e.Cash On Delivery(COD).
- Thus, a person can order any product of his choice over the internet by sitting at home.Some of the real-life examples of the Online Shopping of clothes are Amazon, Flipkart, Myntra, Shopclues, Snapdeal, etc. By this method the product which is ordered is delivered to address of the customer within some days.
- The factors which motivate the shopper to go online can be behavioural or market driven. In the past there has been research done on levels of affects of behaviour factors like shopping convenience, information seeking, immediate possession, social interaction, and variety. Increased convenience and speed of procurement make e-commerce attractive to buyers but also personal motivation factors drive online activity.
- The motivation of building this project is to provide the user the most convenient way to purchase online .By this way customer can choose his/her favourite dress models.
- Customer need not go to the shop for buying products, he/she can order online through many ways. He/she can know about the stock of products of each brand which are registered on our website. Thus, it can become the most convenient way for online shopping for clothes.

CONTRIBUTION OF PROJECT:

Register number	Name	Work assigned
19BCE0214	N.Anand venkata subbaraju	Abstract,Entities &attributes collection,Er diagram,table creation,2nd and bcnf normal forms normalization,quires execution,frontend
19BCE2247	Nadimpalli narshimha raju	Schema diagram,relations and constraints,tables execution,2normal form3th normal form.frontend,sql quires creation and execution
19BCE0077	P.Sai karthik reddy	Introduction and literary survey,documentation of tables,1th,4th normal form,frontend,conversion of queries to relate to our project,pl/sql queries execution.

2-PROJECT RESOURCE REQUIREMENTS:

SOFTWARE REQUIREMENTS:

Word
Sql plus
phpMyAdmin
xampp
windows

HARDWARE REQUIREMENTS:

Internet connectivity
Laptop/pc for admin
Good server

3-LITERATURE SURVEY

In recent times, a number of online shopping ideas and projects has been implemented. Some of them have improvised and improved present online shopping websites. One of our idea is to implement Heuristic Query Optimization in our project. It is often observed in many database industries that a lot of time is spent in executing inefficient SQL queries. This technique optimizes query processing in a great way. There are many such techniques for query processing.

Shibin Chittil proposed a mini “Online Shopping System” project. This project is an attempt to provide the advantages of online shopping to customers of a real shop. It helps

buying the products in the shop anywhere through internet by using an android device. Thus the customer will get the service of online shopping and home delivery from his favourite shop. This system can be implemented to any shop in the locality or to multinational branded shops having retail outlet chains

Aurélia Michaud-Trevinal proposed “Online shopping experiences: a qualitative research”. This research tackles the issue of shopping experiences in an online environment. This paper intends to examine online shopping experiences from three aspects: the physical, ideological and pragmatic dimensions. The results highlighted the three proposed dimensions and underline as core issues online trust (or mistrust), age and online social interactions with friends. The appropriation process of commercial websites is also considered. This paper intends to consider online shopping experiences a whole – and not just purchase experiences, considering shopping practices online and offline, and the appropriation process of commercial websites.

4-DESIGN OF THE PROJECT:

DATA COLLECTION:

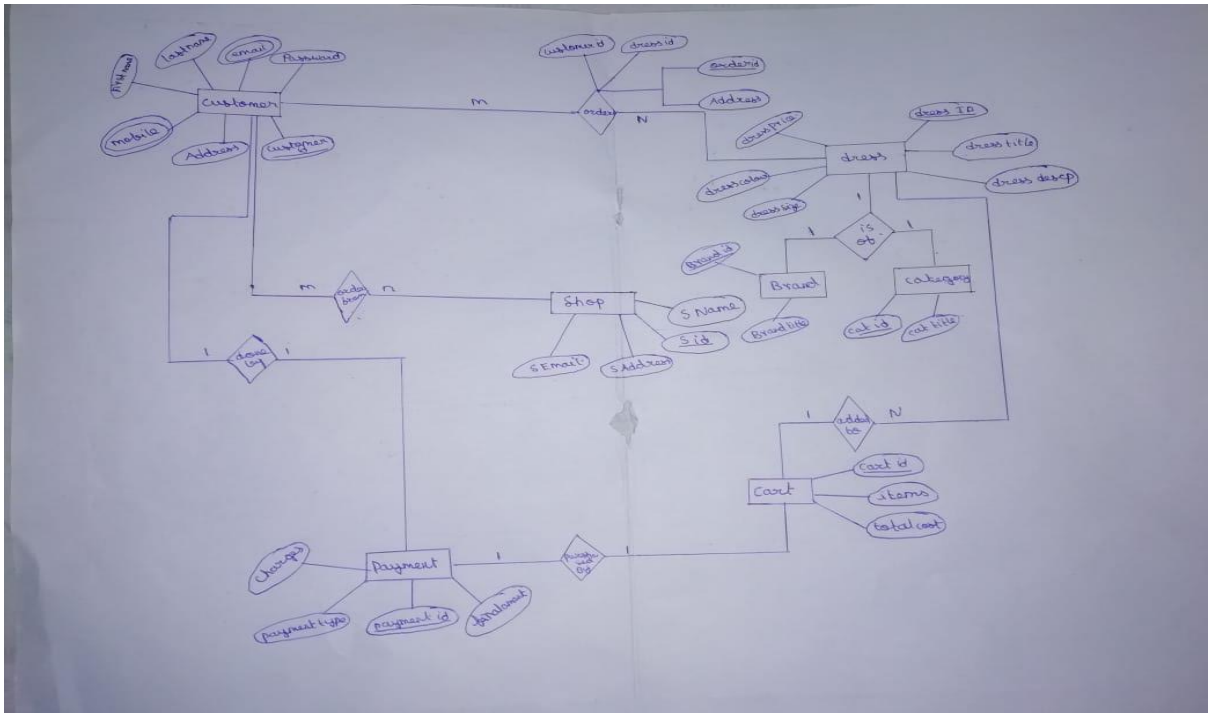
ENTITIES	ATTRIBUTES
Customer	Customer_id(pk) Email_id Address password F_name L_name Mobile
Payments	Payment_type charges Payment_id(pk) Final amount Customer_id(fk) Cart_id(fk)

Dress	dress_id(pk) Dress_title Dress_price Dress_colour Dress_desc Dress_size
Cart	Cart_id(PK) Dress id(fk) Customer_id(fk) Items Total cost
Category	Category_id(pk) Category_title dress_id(fk)

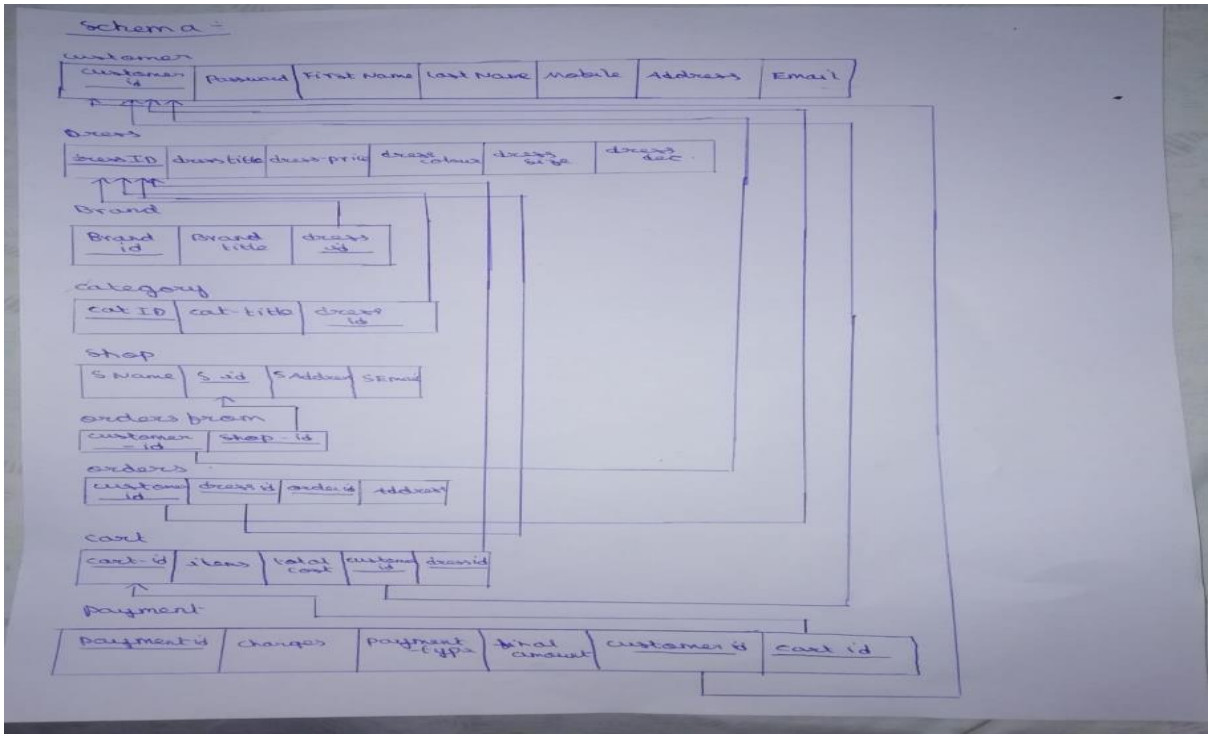
ENTITIES	**ATTRIBUTES**
Shop	Sname Sid(pk) Saddress Semail
Order	Customer_id(fk) Dress_id(fk) Order_id(pk) Address
Order from	Customer_id(fk) Shop_id(fk)
Brand	Brand-id(pk)

	Brand_title Dress_id(fk)

ER DIAGRAM:



ERD TO CONCEPTUAL SCHEMA:



TABLES AND CONSTRAINTS:

TABLE NAME:CUSTOMER

ATTRIBUTES	DATATYPE	CONSTRAINTS
Customer_id	Varchar2(10)	Primary key
Password	Varchar2(10)	Unique
First_name	Varchar2(10)	Not null
Last_name	Varchar2(10)	Unique
Address	Varchar2(50)	Not null
Email	Varchar2(50)	unique
Mobile	int	Unique

TABLE NAME:CATEGORY:

ATTRIBUTES	DATATYPE	CONSTRAINT
Cat_id	Varchar2(10)	Primary key
Cat_title	Varchar2(10)	Not null
dress_id	Varchar2(10)	Foreign key

TABLE NAME:DRESS

ATTRIBUTES	DATATYPE	CONSTRAINT
Dress_id	Varchar2(10)	Primary key
Dress_title	Varchar2(20)	Not null
Dress_price	int	Check(dress price>0)
Dress_colour	Varchar2(10)	Not null
Dress_desc	Varchar2(50)	Not null
Dress_size	Varchar2(10)	Not null

TABLE NAME:CART

ATTRIBUTES	DATATYPE	CONSTRAINTS
Cart_id	Varchar2(10)	Primary key
Dress_id	Varchar2(10)	Foreign key
Items	int	Check(items>0)
Total cost	int	Check(total cost>0)
Customer_id	Varchar2(10)	Foreign key

TABLE NAME:PAYMENT

ATTRIBUTES	DATATYPE	CONSTRAINT
Payment_id	Varchar2(10)	Primary key
Payemnt_type	Varchar2(10)	Not null
charges	int	Check(charges>0)
Final amount	int	Check(final amount>0)
Customer_id	Varchar2(10)	Foreign key
Cart_id	Varchar2(10)	Foreign key

TABLE BRAND:

ATTRIBUTES	DATATYPE	CONSTRAINT
Brand_id	Varchar2(10)	Primary key
Brand_title	Varchar2(20)	Not null
Dress_id	Varchar2(10)	Foreign key

TABLE SHOP:

ATTRIBUTES	DATATYPE	CONSTRAINT
S_name	Varchar2(50)	Not null
S_id	Varchar2(10)	Primary key
S_address	Varchar2(50)	Not null
S_email	Varchar2(25)	unique

TABLE ORDERS:

ATTRIBUTES	**DATATYPE**	**CONSTRAINT**
Customer_id	Varchar2(10)	Foreign key
Dress_id	Varchar2(10)	Foreigh key
Order_id	Varchar(10)	Primary key
address	Varchar2(50)	

TABLE ORDER FROM:

ATTRIBUTES	DATATYPE	CONSTRAINT
Customer_id	Varchar2(10)	Foreign key
Shop_id	Varchar2(10)	Foreign key

TABLE CUSTOMER:

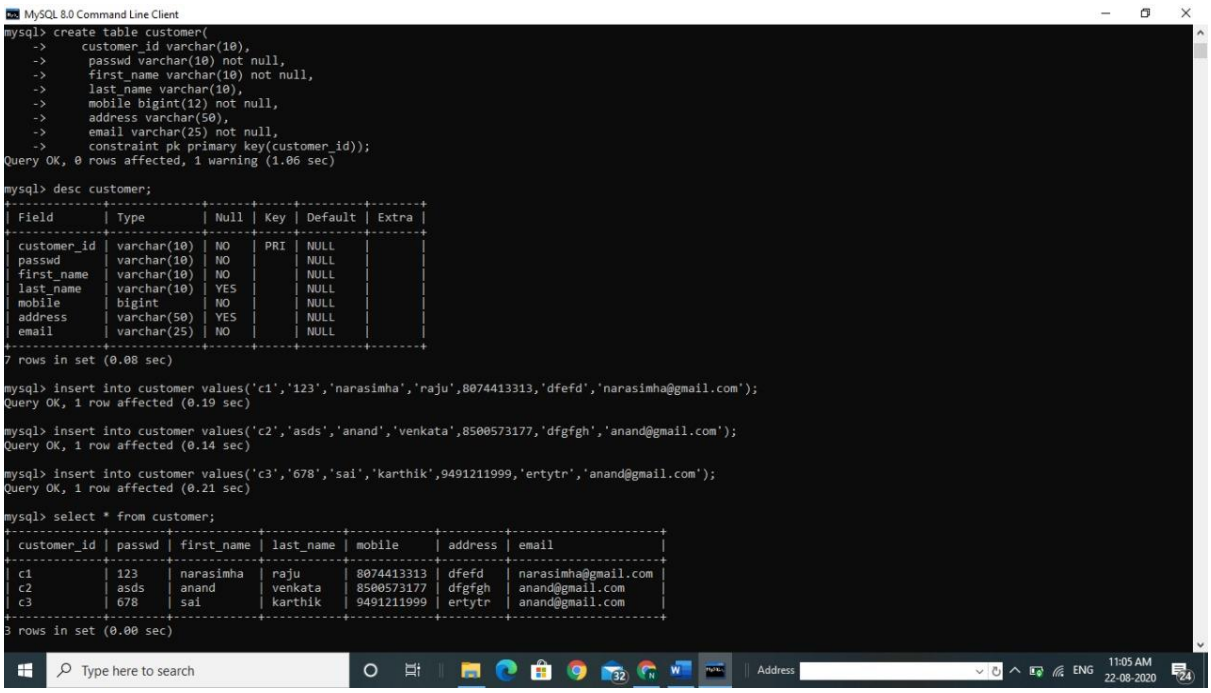
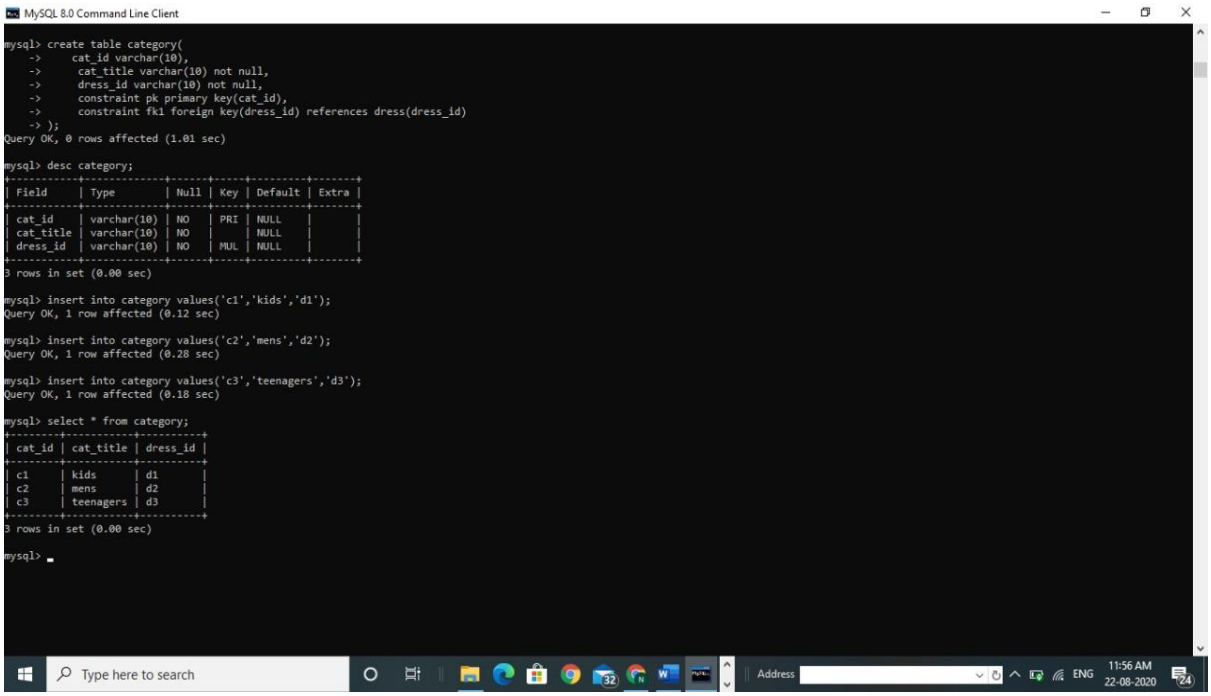
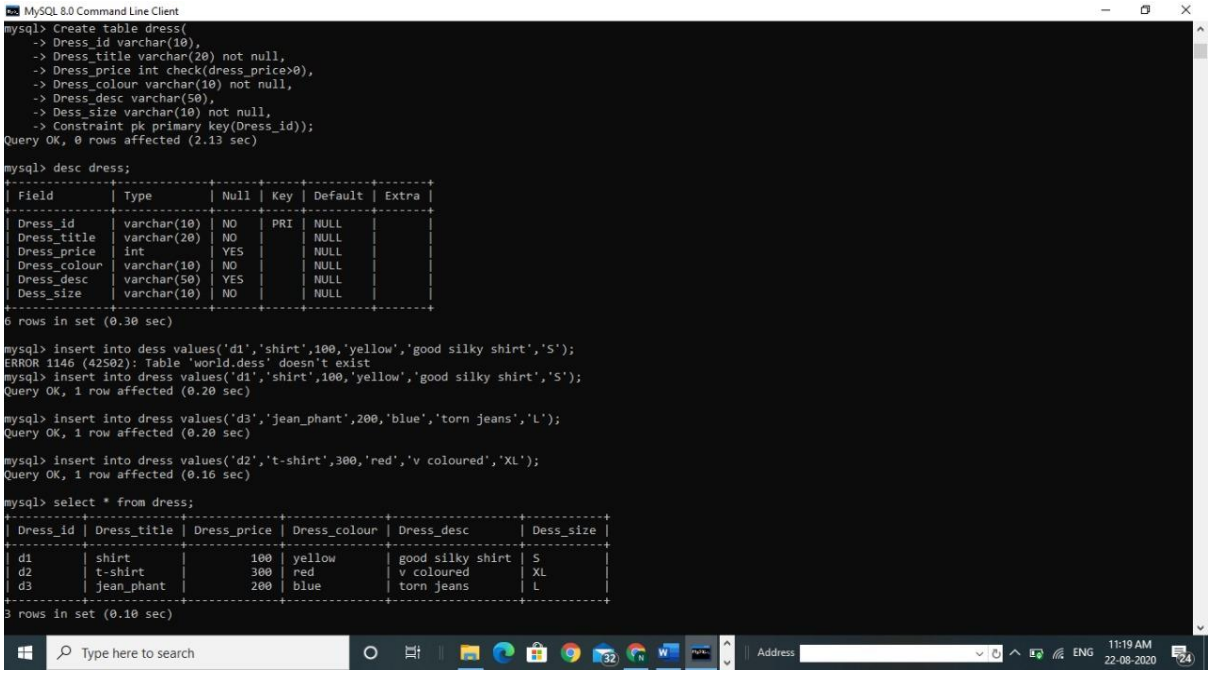


TABLE CATEGORY



```
MySQL 8.0 Command Line Client

mysql> create table category(
    ->     cat_id varchar(10),
    ->      cat_title varchar(10) not null,
    ->      dress_id varchar(10) not null,
    ->      constraint pk primary key(cat_id),
    ->      constraint fk1 foreign key(dress_id) references dress(dress_id)
    -> );
Query OK, 0 rows affected (1.01 sec)

mysql> desc category;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| cat_id    | varchar(10) | NO   | PRI | NULL    |       |
| cat_title | varchar(10) | NO   |     | NULL    |       |
| dress_id  | varchar(10) | NO   | MUL | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> insert into category values('c1','kids','d1');
Query OK, 1 row affected (0.12 sec)

mysql> insert into category values('c2','mens','d2');
Query OK, 1 row affected (0.28 sec)

mysql> insert into category values('c3','teenagers','d3');
Query OK, 1 row affected (0.18 sec)

mysql> select * from category;
+--------+-----------+----------+
| cat_id | cat_title | dress_id |
+--------+-----------+----------+
| c1     | kids      | d1       |
| c2     | mens      | d2       |
| c3     | teenagers | d3       |
+--------+-----------+----------+
3 rows in set (0.00 sec)

mysql>
```

TABLE DRESS:



```
MySQL 8.0 Command Line Client

mysql> Create table dress(
    -> Dress_id varchar(10),
    -> Dress_title varchar(20) not null,
    -> Dress_price int check(dress_price>0),
    -> Dress_colour varchar(10) not null,
    -> Dress_desc varchar(50),
    -> Dess_size varchar(10) not null,
    -> Constraint pk primary key(Dress_id));
Query OK, 0 rows affected (2.13 sec)

mysql> desc dress;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Dress_id     | varchar(10) | NO   | PRI | NULL    |       |
| Dress_title  | varchar(20) | NO   |     | NULL    |       |
| Dress_price  | int         | YES  |     | NULL    |       |
| Dress_colour | varchar(10) | NO   |     | NULL    |       |
| Dress_desc   | varchar(50) | YES  |     | NULL    |       |
| Dess_size    | varchar(10) | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
6 rows in set (0.30 sec)

mysql> insert into dess values('d1','shirt',100,'yellow','good silky shirt','S');
ERROR 1146 (42S02): Table 'world.dess' doesn't exist
mysql> insert into dress values('d1','shirt',100,'yellow','good silky shirt','S');
Query OK, 1 row affected (0.20 sec)

mysql> insert into dress values('d3','jean_phant',200,'blue','torn jeans','L');
Query OK, 1 row affected (0.20 sec)

mysql> insert into dress values('d2','t-shirt',300,'red','v coloured','XL');
Query OK, 1 row affected (0.16 sec)

mysql> select * from dress;
+----------+-------------+-------------+--------------+-----------------+-----------+
| Dress_id | Dress_title | Dress_price | Dress_colour | Dress_desc      | Dess_size |
+----------+-------------+-------------+--------------+-----------------+-----------+
| d1       | shirt       |         100 | yellow       | good silky shirt | S         |
| d2       | t-shirt     |         300 | red          | v coloured      | XL        |
| d3       | jean_phant  |         200 | blue         | torn jeans      | L         |
+----------+-------------+-------------+--------------+-----------------+-----------+
3 rows in set (0.10 sec)
```

TABLE CART:

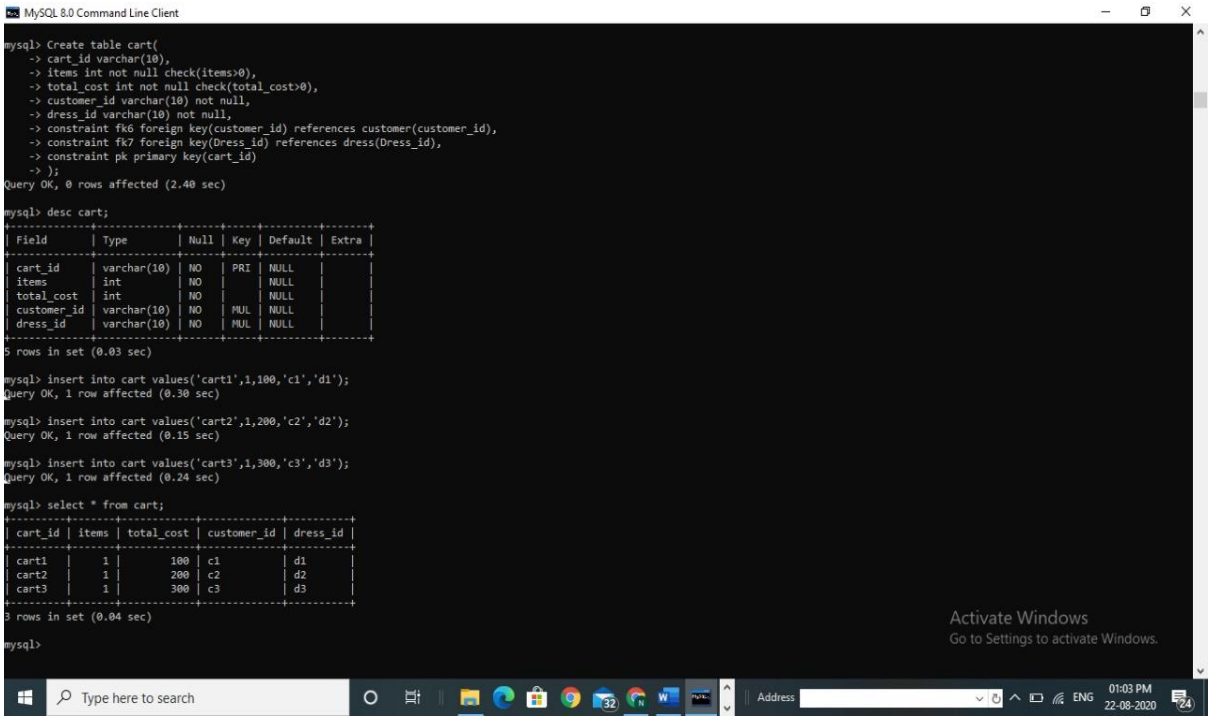


TABLE PAYMENT:

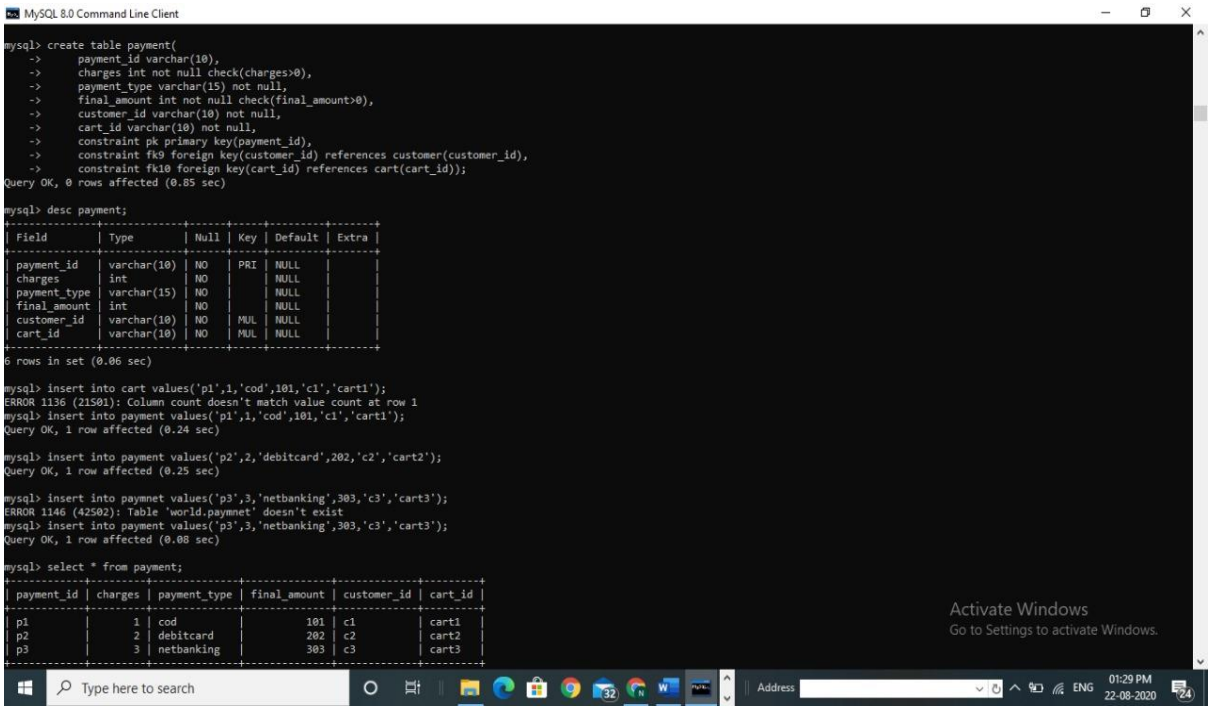


TABLE ORDER

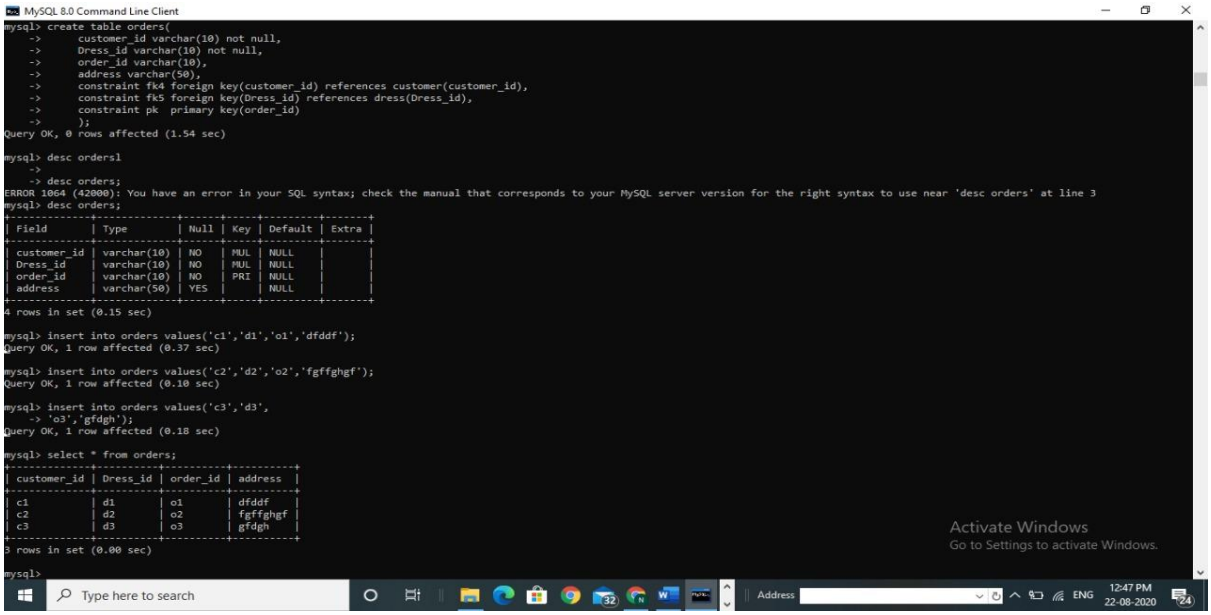


TABLE ORDERS FROM:

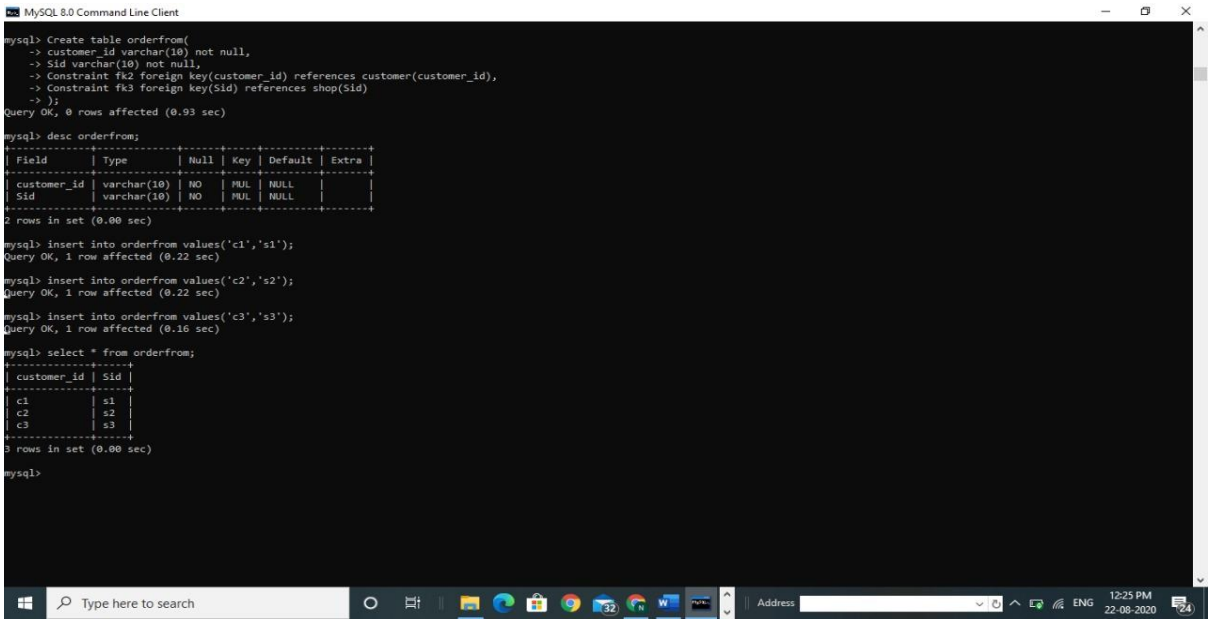


TABLE BRAND:

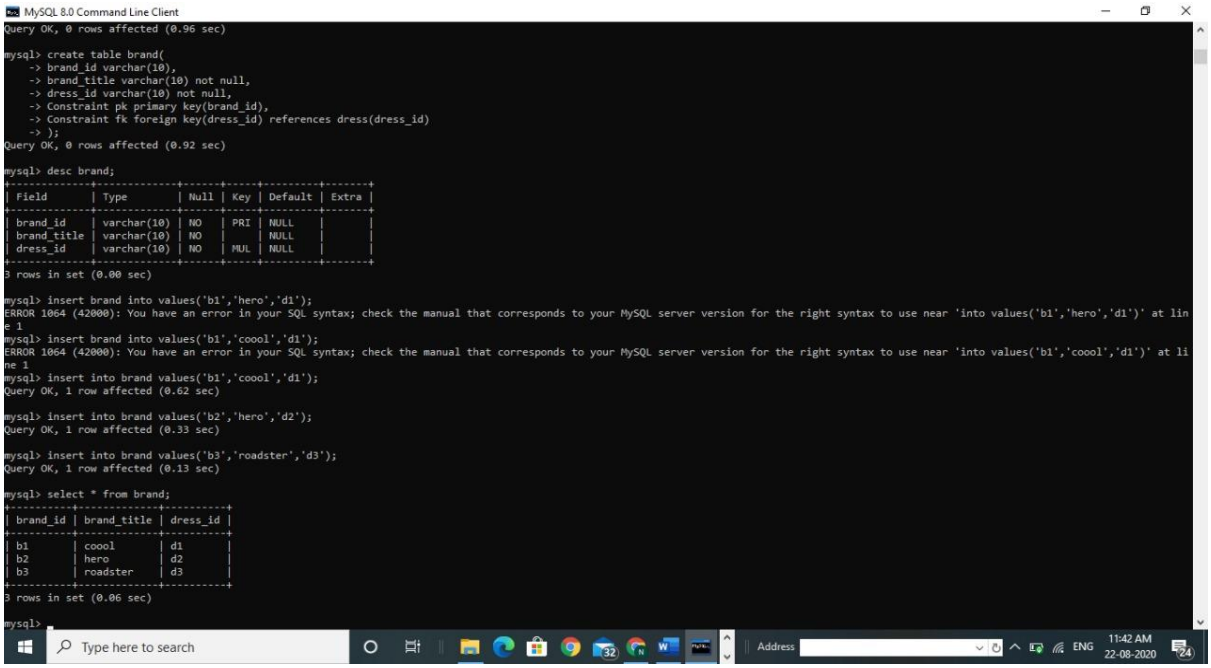
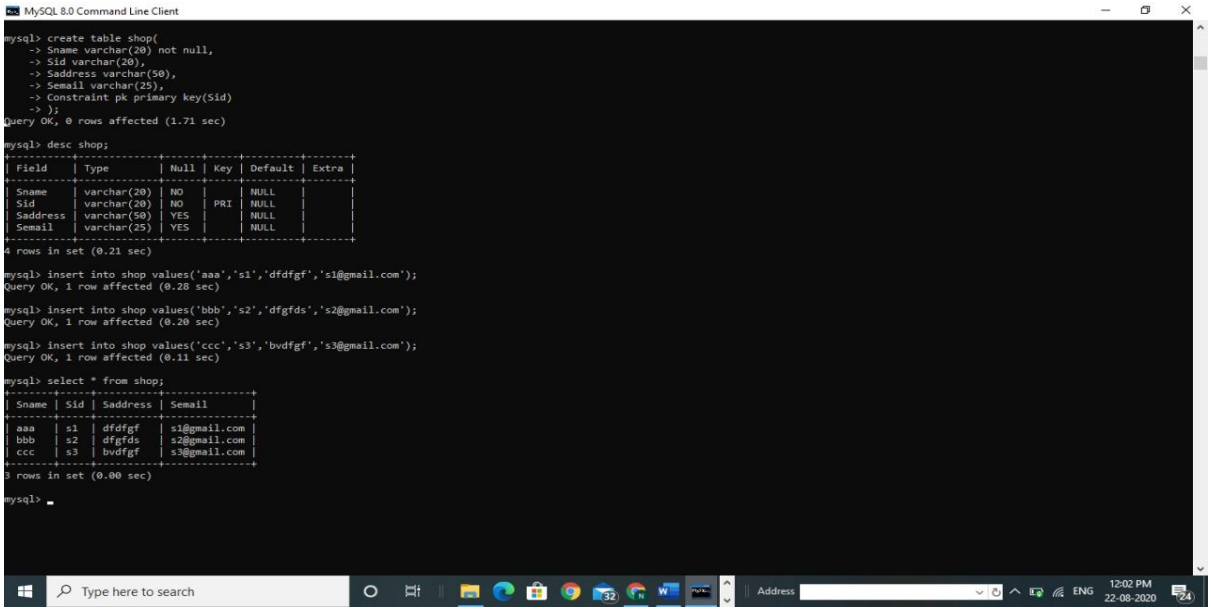


TABLE SHOP:



NORMALISATION:

- **First Normal Form (1nf) :**

First Normal form is a normalization of a table where there are no divisible records and there are no multi-valued attributes. If both the conditions are satisfied, then we can say that the table is in 1NF.

- Rules for First Normal Form

  Rule 1: Single Valued Attributes

  Rule 2: Attribute Domain should not change

  Rule 3: Unique name for Attributes/Columns

- **Second Normal Form:**

- Second Normal form is a normalization of 1NF table in such a way that all non-key attributes are functionally dependent only in one key. That means, there can be no partial key dependency in the table. Partial key dependency is a candidate key where same non-key attributes are dependent on two different key .First the table must be normalized to 1NF then the procedure of 2NF can be carried out.
- For a table to be in the Second Normal Form, it must satisfy two conditions:

1. The table should be in the First Normal Form.
2. There should be no Partial Dependency.
3. Partial Dependency exists, when for a composite primary key, any attribute in the table depends only on a part of the primary key and not on the complete primary key.
4. To remove Partial dependency, we can divide the table, remove the attribute which is causing partial dependency, and move it to some other table where it fits in well.

- **Third Normal Form**

- Third Normal form is a normalization where all transitivity are removed. For example, in a table A, attribute X→Y and Y→Z then X→Z. This rule is called transitivity rule. Third normal form removes such kind of dependency. The table must not any transitivity.Also, the table must have satisfied the conditions of 2NF. If both the conditions are satisfied, then we can say that the table is in 3NF.
- Requirements for Third Normal Form

For a table to be in the third normal form,

1. It should be in the Second Normal form.
2. And it should not have Transitive Dependency.

Advantage of removing Transitive Dependency

- The advantage of removing transitive dependency is,
- Amount of data duplication is reduced.
- Data integrity achieved.

- **BCNF:**

- BCNF is a normalization where LHS of and FD is a super key. Also, the table must be in 3NF. BCNF is also called strict 3NF . For example, in FD X→Y, X attribute must be a super key. Super key is key where all non-key attributes are fully functional dependent. If there is a super key, then table is in BCNF.

- Rules for BCNF

For a table to satisfy the Boyce-Codd Normal Form, it should satisfy the following two conditions:

1. It should be in the **Third Normal Form**.
2. And, for any dependency A → B, A should be a **super key**.

The second point sounds a bit tricky, right? In simple words, it means, that for a dependency A → B, A cannot be a **non-prime attribute**, if B is a **prime attribute**.

4nf:

Fourth Normal Form comes into picture when **Multi-valued Dependency** occur in any relation.

- Rules for 4th Normal Form

For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions:

1-It should be in the **Boyce-Codd Normal Form**.

2- the table should not have any **Multi-valued Dependency**.

A table is said to have multi-valued dependency, if the following conditions are true

*For a dependency A → B, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency.

* a table should have at-least 3 columns for it to have a multi-valued dependency.

*Fora relation R(A,B,C), if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.

If all these conditions are true for any relation(table), it is said to have multi-valued dependency.

TABLE 1:CUSTOMER

Customer(customer_id,Email_id,Address,password,F_name,L_name,mobile)

FUNCTIONAL DEPENDENCIES:

Customer_id -- F_name,L_name,Address,Email_id,password,mobile

Email_id -- customer_id,

Customer_id	Email_id	F_name	L_name	Address	Password	Mobile
C1	Narasimha@gmail.com	Narasimha	raju	dfefd	123	8074413313
C2	anand@gmail.com	Anand	venkata	dfgfgh	asds	8500573177
C3	karthik@gmail.com	sai	karthik	ertytr	678	9491211999
C4	ram@gmail.com ram2@gmail.com	ram	sri	chennai	ram123	9011233111
C5	abbas@gmail.com	Abbas	mohammad	hyderabad	abbos321	7412368905
C6	arup@gmail.com arup2@gmail.com	Anup	kumar	mumbai	anup99	7765345860
C7	maya@gmail.com	maya	kumari	banglore	maya123	9986435200

1NF :

The table above is not in 1NF because Email_id has multiple values per attributes

To normalize into 1NF we make separate records for every multiple attribute

Customer_id	Email_id	F_name	L_name	Address	Password	Mobile
C1	narasimha@gmail.com	narasimha	raju	dfefd	123	8074413313
C2	anand@gmail.com	anand	venkata	dfgfgh	asds	8500573177
C3	karthik@gmail.com	sai	karthik	ertytr	678	9491211999
C4	ram@gmail.com	ram	sri	chennai	ram123	9011233111
C4	ram2@gmail.com	ram	sri	chennai	ram123	9011233111
C5	abbas@gmail.com	abbas	mohammad	hyderabad	abbas321	7412368905
C6	anup@gmail.com	anup	kumar	mumbai	anup99	776534580
C6	anup2@gmail.com	anup	kumar	mumbai	anup99	776534580
C7	maya@gmail.com	maya	kumari	banglore	maya123	9986435200

2nf:

Second normal form is a normalization of 1nf table in such a way that all non-key attributes are functionally dependent only in one key. That means, there can be no partial key dependency in the table. Partial key dependency is a candidate key where same non-key

attributes are dependent on two different key .First the table must be normalized to 1nf then the procedure of 2nf can be carried out.

According to the function dependency, (cus_id, email_id) is a candidate key. That means to get F_name,L_name, address,password,mobile either cus_id or email_id is enough. In other words, all the non-key attributes are dependent on 2 key attributes, i.E. Email_id and cus_id. This tells us that they are partial key dependency. So partial key dependency violates the rule of 2NF. This table is clearly not in 2NF

DEOMPOSITION:

- To decompose the table, we break the table into parts in such a way that all dependent attributes are functionally dependent to only key,i.e Primary key. So in this case email_id should be filled in another table. The other composite key will be included in another table.
- So to normalize this table we will decompose the table into two.
- customer(cus_id, F_name,L_name,address,password,mobile)
- Emails(cus_id,email_id)
- We will make cus_id in emails table as foreign key references to employee. We make a foreign key in order to link these two tables. Also, we make email_id as a primary key so that it uniquely identifies and no two email ids are same.

CUSTOMER_ID

Cus_id	F_name	L_name	Address	Password	Mobile
C1	narasimha	raju	dfefd	123	807413313
C2	anand	venkata	dfgfgh	asds	8500573177
C3	sai	karthik	ertytr	678	9491211999
C4	ram	sri	chennai	ram123	9011233111
C5	abbas	mohammad	hyderabad	abbas321	7412368905
C6	anup	kumar	mumbai	anup99	7765345860
C7	maya	kumari	banglore	maya123	9986435200

EMAILS:

Cus_id	email_id
C1	narasimha@gmail.com
C2	anand@gmail.com
C3	karthik@gmail.com
C4	ram@g,mail.com
C4	ram2@gmail.com
C5	abbas@gmail.com
C6	anup@gmail.com
C6	anup2@gmail.com
C7	maya@gmail.com

3NF:

- Third normal form is a normalization where all transitivity are removed. For example, in a table A, attribute X→Y and Y→Z then X→Z. This rule is called transitivity rule. Third normal form removes such kind of dependency. The table must not any transitivity.Also, the table must have satisfied the conditions of 2NF. If both the conditions are satisfied, then we can say that the table is in 3NF.
- In employee table, there is no transitivity.
- cus_id → F_name,L_name,address,password,mobile
- These are functional dependencies of this table. As shown, this cannot be implied in X→Y and Y→Z formula. Because the RHS of this FD are all non-prime attributes or non-key attributes, hence they are not functionally dependent on anything. So this table clearly does not have transitivity.
- So the table customer is in 3nf already and hence no decomposition is required.

BCNF:

- BCNF is a normalization where LHS of and FD is a super key. Also, the table must be in 3NF. BCNF is also called strict 3NF . for example, in FD X→Y, X attribute must be a super key. Super key is key where all non-key attributes are fully functional dependent. If there is a super key, then table is in BCNF.
- In the table customer the functional dependency is:
- cus_id → F_name,L_name,address,password,mobile
- cus_id is super key because all RHS attributes are fully functionally dependent on cus_id (LHS). Moreover, no more fds can be formed out of this table. Since there is an super key already we can say that this table is in BCNF. Also, email(cus_id,email_id) is a subset of first FD.
- The decomposition is not required as it is already in bcnf.

4nf:

Fourth Normal Form comes into picture when **Multi-valued Dependency** occur in any relation.

- Rules for 4th Normal Form

For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions:

1-It should be in the **Boyce-Codd Normal Form**.

2- the table should not have any **Multi-valued Dependency**.

A table is said to have multi-valued dependency, if the following conditions are true

*For a dependency A → B, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency.

* a table should have at-least 3 columns for it to have a multi-valued dependency.

*For a relation R(A,B,C), if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.

If all these conditions are true for any relation(table), it is said to have multi-valued dependency.

But in this customer table it has only 2 coloumns so we need one more coloumn to satisfy 4nf

TABLE 2:CARTEGORY

Category(cat_id,cat_title,dress_id)

Cat_id	Cat_title	Dress_id
C1	Kids	D1
C2	Mens	D2
C3	Teenagers	D3
C4	Punjabi	D4
C5	Sarees	D5,D6
C6	sports	D7

1NF:

The table above is not in 1NF because Email_id has multiple values per attributes

To normalize into 1NF we make separate records for every multiple attribute

Cat_id	Cat_title	Dress_id
C1	Kids	D1
C2	Mens	D2
C3	Teenagers	D3
C4	Punjabi	D4
C5	Sarees	D5
C5	sarees	D6
C6	Sports	D7

2NF:

Functional dependices:

Cat_id -- cat_title

dress_id -- cat_id

Second normal form is a normalization of 1NF table in such a way that all non-key attributes are functionally dependent only in one key. That means, there can be no partial key dependency in the table. Partial key dependency is a candidate key where same non-key attributes are dependent on two different key .First the table must be normalized to 1NF then the procedure of 2NF can be carried out

2NF:

Category:

Cat_id	Cat_title
C1	Kids
C2	Men
C3	Teenagers
C4	Punjabi
C5	Sarees
C6	sports

Dress_category:

Dress_id	Cat_id
D1	C1
D2	C2
D3	C3
D4	C4
D5	C5
D6	C5
D7	C6

3NF:

- Third normal form is a normalization where all transitivity are removed. For example, in a table A, attribute X→Y and Y→Z then X→Z. This rule is called transitivity rule. Third normal form removes such kind of dependency. The table must not any transitivity.Also, the table must have satisfied the conditions of 2NF. If both the conditions are satisfied, then we can say that the table is in 3NF.
- In category table, there is no transitivity.
- These are functional dependencies of this table. As shown, this cannot be implied in X→Y and Y→Z formula. Because the RHS of this FD are all non-prime attributes or non-key attributes, hence they are not functionally dependent on anything. So this table clearly does not have transitivity.
- So the table category is in 3nf already and hence no decomposition is required.

BCNF:

- BCNF is a normalization where LHS of and FD is a super key. Also, the table must be in 3NF. BCNF is also called strict 3NF . for example, in FD X→Y, X attribute must be a super key. Super key is key where all non-key attributes are fully functional dependent. If there is a super key, then table is in BCNF
- The decomposition is not required as it is already in BCNF.

4nf:

Fourth Normal Form comes into picture when **Multi-valued Dependency** occur in any relation.

- Rules for 4th Normal Form

For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions:

1-It should be in the **Boyce-Codd Normal Form**.

2- the table should not have any **Multi-valued Dependency**.

A table is said to have multi-valued dependency, if the following conditions are true

*For a dependency A → B, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency.

* a table should have at-least 3 columns for it to have a multi-valued dependency.

*Fora relation R(A,B,C), if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.

If all these conditions are true for any relation(table), it is said to have multi-valued dependency.

But the present category table is in 4nf

TABLE -3 DRESS:

Dress(dress_id,dress_price,dress_title,dress_colour,dress_desc,dress_size)

Functional dependencies:

dress_id -- dress_title,dress_colour,dress_desc,dress_price,dress_size

Dress_id	Dress_title	Dress_price	Dress_colour	Dress_desc	Dress_size
D1	Shirt	100	Yellow	Good silky shirt	S
D2	T-shirt	300	Red	V coloured	XL
D3	Jean_pant	200	Blue	Torn jeans	L
D4	Suit	400	White	Very good quality	XXL
D5	Hoddie	500	Pink	Woolen	SL
D6	Tack	600	Black	Cotton	XM
D7	Short	700	green	nylon	XXXL

This is in 1nf,2nf,3nf, and bcnf

1NF:

- First Normal form is a normalization of a table where there are no divisible records and there are no multi-valued attributes. If both the conditions are satisfied, then we can say that the table is in 1NF.

2NF:

- Second Normal form is a normalization of 1NF table in such a way that all non-key attributes are functionally dependent only in one key. That means, there can be no partial key dependency in the table. Partial key dependency is a candidate key where same non-key attributes are dependent on two different key .

3NF:

- Third Normal form is a normalization where all transitivity are removed. For example, in a table A, attribute X→Y and Y→Z then X→Z. This rule is called transitivity rule. Third normal form removes such kind of dependency. The table must not any transitivity.Also, the table must have satisfied the conditions of 2NF. If both the conditions are satisfied, then we can say that the table is in 3NF.

BCNF:

- BCNF is a normalization where LHS of and FD is a super key. Also, the table must be in 3NF. BCNF is also called strict 3NF . For example, in FD X→Y, X attribute must be a super key. Super key is key where all non-key attributes are fully functional dependent. If there is a super key, then table is in BCNF.

4nf:

Fourth Normal Form comes into picture when **Multi-valued Dependency** occur in any relation.

- Rules for 4th Normal Form

For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions:

1-It should be in the **Boyce-Codd Normal Form**.

2- the table should not have any **Multi-valued Dependency**.

A table is said to have multi-valued dependency, if the following conditions are true

*For a dependency A → B, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency.

* a table should have at-least 3 columns for it to have a multi-valued dependency.

*Fora relation R(A,B,C), if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.

If all these conditions are true for any relation(table), it is said to have multi-valued dependency.

But the present dress table is in 4nf

TABLE 4 CART:

- Cart_id -- dress_id,items,total cost,customer_id

Cart_id	Dress_id	Items	Total cost	Customer_id
Cart1	D1	1	100	C1
Cart2	D2	2	200	C2
Cart3	D3	3	300	C3
Cart4	D4	4	400	C4
Cart5	D5	5	500	C5
Cart6	D6	6	600	C6
Cart7	D7	7	700	C7

- **2nf**

  cart(cart_id,items,total cost,customer_id)

Cart id	items	totalcost	Customer_id
Cart1	1	100	C1
Cart2	2	200	C2
Cart3	3	300	C3
Cart4	4	400	C4
Cart5	5	500	C5
Cart6	6	600	C6
Cart7	7	700	C7

2nf

Dress_in_cart(cart_id,dress_id)

Cart_id	Dress_id
Cart1	D1
Cart2	D2
Cart3	D3
Cart4	D4
Cart5	D5
Cart6	D6
Cart7	D7

- **3nf:**

Third Normal form is a normalization where all transitivity are removed. For example, in a table A, attribute X→Y and Y→Z then X→Z. This rule is called transitivity rule. Third normal form removes such kind of dependency. The table must not any transitivity.Also, the table must have satisfied the conditions of 2NF. If both the conditions are satisfied, then we can say that the table is in 3NF.

Bcnf:

BCNF is a normalization where LHS of and FD is a super key. Also, the table must be in 3NF. BCNF is also called strict 3NF . For example, in FD X→Y, X attribute must be a super key. Super key is key where all non-key attributes are fully functional dependent. If there is a super key, then table is in BCNF.

4nf:

Fourth Normal Form comes into picture when **Multi-valued Dependency** occur in any relation.

Rules for 4th Normal Form

For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions:

1-It should be in the **Boyce-Codd Normal Form**.

2- the table should not have any **Multi-valued Dependency**.

A table is said to have multi-valued dependency, if the following conditions are true

*For a dependency A → B, if for a single value of A, multiple value of B exists, then the table may have multi-valued dependency.

* a table should have at-least 3 columns for it to have a multi-valued dependency.

*Fora relation R(A,B,C), if there is a multi-valued dependency between, A and B, then B and C should be independent of each other.

If all these conditions are true for any relation(table), it is said to have multi-valued dependency.

But the present cart table is in 4nf

Table 5: payment

- Payment_id -- payment_type,charges,final amount,cust_id,cart_id
- It is in all 1nf,2nf,3nf and bcnf and 4nf

Payment_id	Payment_type	Charges	Final amount	Customer_id	Cart_id
p1	cod	1	101	C1	Cart1
p2	debitcard	2	202	C2	Cart2
p3	netbanking	3	303	C3	Cart3
p4	upi	4	404	C4	Cart4
p5	paytm	5	505	C5	Cart5
p6	creditcard	6	606	C6	Cart6
p7	mastercard	7	707	C7	Cart7

Table 6: brand

- Brand_id -- brand title,dress_id

Brand_id	Brand_title	Dress_id
B1	coool	D1
B2	hero	D2
B3	roadster	D3
B4	addidas	D4
B5	nike	D5
B6	puma	D6 D7

- **1nf:**

First Normal form is a normalization of a table where there are no divisible records and there are no multi- valued attributes. If both the conditions are satisfied, then we can say that the table is in 1NF.

Brand_id2	Brand_title	Dress_id
B1	coool	D1
B2	hero	D2
B3	roadster	D3
B4	addidas	D4
B5	nike	D5
B6	puma	D6
B6	puma	D7

- **2nf:**

Second Normal form is a normalization of 1NF table in such a way that all non-key attributes are functionally dependent only in one key. That means, there can be no partial key dependency in the table. Partial key dependency is a candidate key where same non-key attributes are dependent on two different key .First the table must be normalized to 1NF then the procedure of 2NF can be carried out.

*Brand(brand_id,brand_title)

*Brand_of_dress(dress_id,brand_id)

Brand_id	Brand_title
B1	coool
B2	hero
B3	roadster
B4	addidas
B5	nike
B6	Puma

Dress_id	Brand_id
D1	B1
D2	B2
D3	B3
D4	B4
D5	B5
D6	B6
D7	B6

- Already in 3nf and bcnf and it is in 4nf

TABLE 7: SHOP

- Sid -- s_name,s_title
- Semail -- sid

Sid	Semail	S_name	S_address
S1	s1@gmail.com	aaa	dfdfgf
S2	s2@gmail.com	bbb	dfgfds
S3	s3@gmail.com	ccc	bvdfgf
S4	sai@gmail.com abc@gmail.com	vasavi	shadnagar
S5	xyz@gmail.com	garments	gadwal
S6	pqr@gmail.com jkl@gmail.com	friends	vijayawada

- **1nf:**

The table above is not in 1NF because Email_id has multiple values per attributes

To normalize into 1NF we make separate records for every multiple attribute

Sid	Semail	S_name	Saddress
S1	s1@.cgmailom	aaa	dfdfgf
S2	s2@gmail.com	bbb	dfgfds
S3	s3@gmail.com	ccc	bvdfgf
S4	sai@gmail.com	vasavi	shadnagar
S4	abc@gmail.com	vasavi	shadnagar
S5	xyz@gmail.com	garments	gadwal
S6	pqr@gmail.com	friends	vijayawada
S6	jkl@gmail.com	friends	vijayawada

- **2nf**

shopid -- (sname,stile)

shopemail -- (sid,semail)

Second normal form is a normalization of 1nf table in such a way that all non-key attributes are functionally dependent only in one key. That means, there can be no partial key dependency in the table. Partial key dependency is a candidate key where same non-key attributes are dependent on two different key .First the table must be normalized to 1nf then the procedure of 2nf can be carried out.

SHOPID

Sid	S_name	Saddress
S1	aaa	dfdfgf
S2	bbb	dfgfds
S3	ccc	bvdfgf
S4	vasavi	shadnagar
S5	garments	gadwal
S6	friends	vijayawada

SHOPEMAIL

Sid	Semail
S1	s1@gmail.com
S2	s2@gmail.com
S3	s3@gmail.com
S4	sai@gmail.com
S4	abc@gmail.com
S5	xyz@gmail.com
S6	pqr@gmail.com
S6	jkl@gmail.com

3nf

Third Normal form is a normalization where all transitivity are removed. For example, in a table A, attribute X→Y and Y→Z then X→Z. This rule is called transitivity rule. Third normal form removes such kind of dependency. The table must not any transitivity.Also, the table must have satisfied the conditions of 2NF. If both the conditions are satisfied, then we can say that the table is in 3NF

BCNF

BCNF is a normalization where LHS of and FD is a super key. Also, the table must be in 3NF. BCNF is also called strict 3NF . For example, in FD X→Y, X attribute must be a super key. Super key is key where all non-key attributes are fully functional dependent. If there is a super key, then table is in BCNF.

THEY ARE IN 3NF AND BCNF 4NF

TABLE 8:ORDER

- Orderid --(customer_id,dress_id,address)

Orderid	Customer_id	Dress_id	Address
O1	C1	D1	dfefd
O2	C2	D2	dfgfgh
O3	C3	D3	ertytr
O4	C4	D4	chennai
O5	C5	D5	hyderabad
O6	C6	D6	mumbai
O7	C7	D7	banglore

ALREADY IN 1NF,2NF,3NF,BCNF AND 4NF

Table 9: orders from

Customer_id -- shop_id

Customer_id	Shop_id
C1	S1
C2	S2
C3	S3
C4	S4
C5	S5
C6,C7	S6

Not in 1nf because it has mutivalues

- 1nf

Customer_id	Shop_id
C1	S1
C2	S2
C3	S3
C4	S4
C5	S5
C6	S6
C7	S6

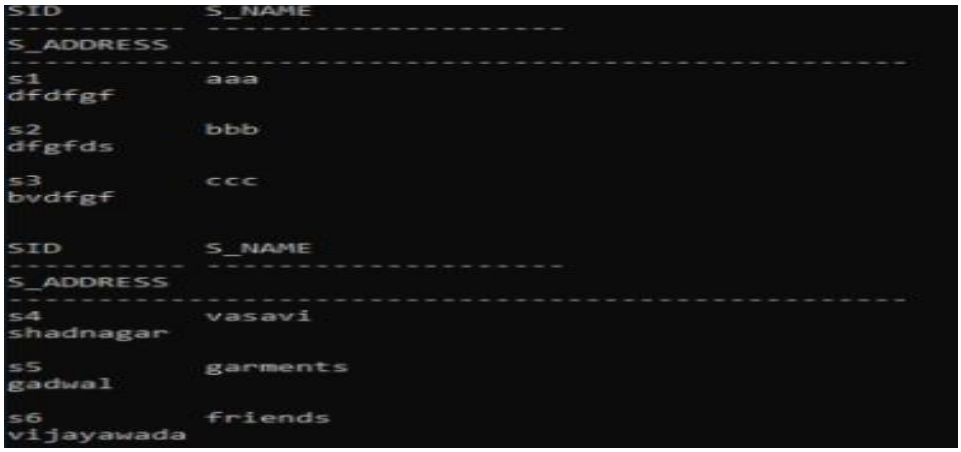
Already in 2nf,3nf,bcnf and 4nf

5-IMPLIMENTATION:

QUERIES EXECUTION

1)Find the details of all shops.SHOP

Select * from shop



2)The last_name of customer hose mobile number is ‘8500573177’

Select last_name from customer where mobile=’8500573177’;

```
SQL> select last_name from customer where mobile='8500573177';

LAST_NAME
---------
venkata
```

3)Find customer_id with number of items brought is =’3’

Select customer_id from cart where items=3;

```
SQL> select customer_id from cart where items=3;

CUSTOMER_I
----------
c3
```

4)Find the cart_id where customer_id=’c1’

Select cart_id from cart where customer_id=’c1’;

```
SQL> select cart_id from cart where customer_id='c1';

CART_ID
----------
cart1
```

5)Find the name of the customer where mobile=’90011233111’

Select last_name from customer where mobile=’9011233111’

```
SQL> select last_name from customer where mobile='9011233111';

LAST_NAME
----------
sai
```

6)Get the total number of items<3

Select count(*) from cart where items<’3’;

```
SQL> select count(*) from cart where items<'3';

  COUNT(*)
----------
         2
```

7)Get the count of customers from Hyderabad

Select count(*) from customer where address=’hyderabad’;

```
SQL> select count(*) from customer where address='hyderabad';

  COUNT(*)
----------
         1
```

8) select of cat_id’s that has cat_title as mens or Punjabi

Select cat_id from category where cat_title=’mens’ or cat_title=’punjabi’;

```
SQL> select cat_id from category where cat_title='mens' or cat_title='punjabi';

CAT_ID
---------
cat4
```

9)Display customer last_names in capital letters

Select upper(llast_name) from customer;

```
SQL> select upper(last_name) from customer;

UPPER(LAST
----------
KUMAR
RAJU
VENKATA
KARTHIK
SAI
MOHAMMAD
KUMARI
```

10)Select sin(mobile) as SIN_VALUE from customer where first_name=’anand’;

```
SQL> select sin(mobile) as SIN_VALUE from customer where first_name='anand';

 SIN_VALUE
----------
.994904839
```

11)Display the length of first_name of customer

Select first_name,length(first_name)as length from customer;

```
SQL> select first_name,length(first_name) as length from customer;

FIRST_NAME     LENGTH
---------- ----------
anup                4
narasimha           9
anand               5
sai                 3
ram                 3
abbas               5
maya                4

7 rows selected.
```

12)Cosine value of mobile number where customer id=’c3’

Select cos(mobile) as COSINE_VALUE from customer where customer_id=’c3’;

```
SQL> select cos(mobile) as COSINE_VALUE from customer where customer_id='c3';

COSINE_VALUE
------------
  -.89395642
```

13)Find addres where customer_id=’c5’

Select address from customer where customer_id=’c5’

```
SQL> select address from customer where customer_id='c5';

ADDRESS
--------------------------------------------------
hyderabad
```

14)Mobile number of customer whose first name start with letter ‘a’

Select first name,mobile from customer where first_name like’a%’;

```
SQL> select first_name,mobile from customer where first_name like 'a%';

FIRST_NAME     MOBILE
---------- ----------
anup         776534580
anand       8500573177
abbas       7412368905
```

15)Find the first name of customer whose customer_id=’c1’

Select first_name from customer where customer.customer_id=’c1’

```
SQL> select first_name from customer where customer.customer_id='c1';

FIRST_NAME
----------
narasimha
```

JOIN OPERATOR:

FULL OUTER JOIN

First name of customer who made a order

Selct first_name from customer full outer join orders on customer.customerid=orders.customer_id;

```
SQL> select first_name from customer full outer join orders on customer.customer_id=orders.customer_id;

FIRST_NAME
----------
narasimha
anand
sai
ram
abbas
anup
maya
```

INNER JOIN

First name of order_id’s

Selct orders.order_id,customer.first_name from orders inner join customer on orders.customer_id=customer.customer_id;

```
SQL> select orders.order_id,customer.first_name from orders inner join customer on orders.customer_id=customer.customer_id;

ORDER_ID   FIRST_NAME
---------- ----------
o6         anup
o1         narasimha
o2         anand
o3         sai
o4         ram
o5         abbas
o7         maya
```

INNER JOIN

Find first name and last_name of a customers by their payment_id

Select payment.payment_id,customer.first_name,customer.last_name from payment inner join customer on payment.customer_id=customer.customer_id;

```
SQL> select payment.payment_id,customer.first_name,customer.last_name from payment inner join customer on payment.customer_id=customer.customer_id;

PAYMENT_ID FIRST_NAME LAST_NAME
---------- ---------- ----------
p6         anup       kumar
p1         narasimha  raju
p2         anand      venkata
p3         sai        karthik
p4         ram        sai
p5         abbas      mohammad
p7         maya       kumari
```

FULL OUTER JOIN

Find first_name and last_name by customer and orders tables

Select first_name,last_name from customer full outer join on customer.customer_id=orders.customer_id;

```
SQL> select first_name,last_name from customer full outer join orders on customer.customer_id=orders.customer_id;

FIRST_NAME LAST_NAME
---------- ----------
narasimha  raju
anand      venkata
sai        karthik
ram        sai
abbas      mohammad
anup       kumar
maya       kumari

7 rows selected.
```

FULL OUTER JOIN

Mobile and address of customers who have done payment

Select mobile,address from customer full outer join payment on customer.customer_id=payment.customer_id;

```
SQL> select mobile,address from customer full outer join payment on customer.customer_id=payment.customer_id;

    MOBILE ADDRESS
---------- --------------------------------------------------
8074413313 dfefd
8500573177 dfgfgh
9491211999 ertytr
9011233111 chennai
7412368905 hyderabad
 776534580 mumbai
9986435200 bangalore
```

LEFT JOIN

Find the first_name and last_name of customer show their resp order_id

Selct customer.first_name,customer.last_name,orders.order_id from customer LEFT JOIN orders on customer.customer_id=orders.customer_id;

```
SQL> select customer.first_name,customer.last_name,orders.order_id from customer LEFT JOIN orders ON customer.customer_id=orders.customer_id;

FIRST_NAME LAST_NAME  ORDER_ID
---------- ---------- ----------
narasimha  raju       o1
anand      venkata    o2
sai        karthik    o3
ram        sai        o4
abbas      mohammad   o5
anup       kumar      o6
maya       kumari     o7
```

RIGHT JOIN

Find the list of first_name and charges charged from customer

Select customer.first_name,customer.mobile,payment.payment_type,payment.charges from customer right join payment on customer.customer_id=payment.customer_id;

```
SQL> select customer.first_name,customer.mobile,payment.payment_type,payment.charges from customer right join payment on customer.customer_id=payment.customer_id;

FIRST_NAME     MOBILE PAYMENT_TYPE           CHARGES
---------- ---------- -------------------- ----------
anup        776534580 creditcard                    6
narasimha  8074413313 cod                           1
anand      8500573177 debitcard                     2
sai        9491211999 netbanking                    3
ram        9011233111 upi                           4
abbas      7412368905 paytm                         5
maya       9986435200 mastercard                    7

7 rows selected.
```

CARTISIAN PRODUCT:

```
SQL> select customer.customer_id,payment.payment_id from customer,payment;

CUSTOMER_I PAYMENT_ID
---------- ----------
c6         p1
c6         p2
c6         p3
c6         p4
c6         p5
c6         p6
c6         p7
c1         p1
c1         p2
c1         p3
c1         p4

CUSTOMER_I PAYMENT_ID
---------- ----------
c1         p5
c1         p6
c1         p7
c2         p1
c2         p2
c2         p3
c2         p4
c2         p5
c2         p6
c2         p7
c3         p1

CUSTOMER_I PAYMENT_ID
---------- ----------
c3         p2
c3         p3
c3         p4
c3         p5
c3         p6
c3         p7
c4         p1
c4         p2
c4         p3
c4         p4
c4         p5
```

```
CUSTOMER_I  PAYMENT_ID
----------  ----------
c4          p6
c4          p7
c5          p1
c5          p2
c5          p3
c5          p4
c5          p5
c5          p6
c5          p7
c7          p1
c7          p2

CUSTOMER_I  PAYMENT_ID
----------  ----------
c7          p3
c7          p4
c7          p5
c7          p6
c7          p7

49 rows selected.
```

JOIN:

```
SQL> select first_name,last_name,charges,final_amount,dress_id from customer c,payment p,orders o
  2  where c.customer_id=p.customer_id and p.customer_id=o.customer_id;

FIRST_NAME LAST_NAME     CHARGES FINAL_AMOUNT DRESS_ID
---------- ---------- ---------- ------------ ----------
narasimha  raju                1          101 d1
anand      venkata             2          202 d2
sai        karthik             3          303 d3
ram        sai                 4          404 d4
abbas      mohammad            5          505 d5
anup       kumar               6          606 d6
maya       kumari              7          707 d7

7 rows selected.
```

PLSQL QUIRES EXECUTION

1) Write a PL/SQL program to practice reading the record from a table into local variables using different data types and %TYPE and display the same using locally declared variables

SQL> set serveroutput on;

SQL> declare

c_no customers.customer_id&type;

fname customers.firstname type;

name customers.last name%type;

mob_no customers.mobile%type;

pass customers.passwdt%ype;

begin

select customer_id,first_name, lastname, mobile,passwd into c_no,f_name,1_ name, mob_no,pass from customers where customer_id= 'c1';

dbms_output.put_line('customer_id : ‘ || c_no);

dbms_output.put_line('first_name : ‘ || f_name);

dbms_output.put_line(‘last_name : ‘ || l_name);

dbms_output.put_line(‘mobile : ‘||mob_no);

dbms_output.put_line('passwd: ‘ || pass);

end; /

```
SQL> set serveroutput on;
SQL> declare
  2  c_no customers.customer_id%type;
  3  f_name customers.first_name%type;
  4  l_name customers.last_name%type;
  5  mob_no customers.mobile%type;
  6  pass customers.passwd%type;
  7  begin
  8  select customer_id,first_name,last_name,mobile,passwd into c_no,f_name,l_name,mob_no,pass from customers where customer_id='c1';
  9  dbms_output.put_line('customer_id : ' || c_no);
 10  dbms_output.put_line('first_name : ' || f_name);
 11  dbms_output.put_line('last_name : ' || l_name);
 12  dbms_output.put_line('mobile : ' || mob_no);
 13  dbms_output.put_line('passwd : ' || pass);
 14  end;
 15  /
customer_id : c1
first_name : narasimha
last_name : raju
mobile : 8074413313
passwd : 123

PL/SQL procedure successfully completed.
```

2.Write a PL/SQL program to insert records into any of the tables in your database.

SQL> begin

insert into customers values('c4', '900', 'jagan', 'reddy', '9014979439',’dubai’); insert into customers values(‘c5','333','lokesh', 'babu', 9123456234','america’);

end;

/

```
PL/SQL procedure successfully completed.

SQL> begin
  2  insert into customers values('c4', '900','jagan','reddy','9014970439','dubai');
  3  insert into customers values('c5','333','lokesh','babu','9123456234','america');
  4  end;
  5  /

PL/SQL procedure successfully completed.

SQL> select * from customers;

CUSTOMER_I PASSWD     FIRST_NAME LAST_NAME      MOBILE
---------- ---------- ---------- ---------- ----------
ADDRESS
--------------------------------------------------
           123        narasimha  raju       8074413313
efd

c2         asds       anand      venkata    8500573177
dfgfgh

c3         678        sai        karthik    9491211999
ertytr


CUSTOMER_I PASSWD     FIRST_NAME LAST_NAME      MOBILE
---------- ---------- ---------- ---------- ----------
ADDRESS
--------------------------------------------------
c4         900        jagan      reddy      9014970439
dubai

c5         333        lokesh     babu       9123456234
america
```

3-Create a function customer to find the customer_id in the given address. Use the address name as the input parameter for the function.

```
SQL> create or replace function st_count (address varchar) return number is
CNT number;
Begin
select count(address) into CNT from customers where customers.address= address;
return CNT;
 end;
 /
SQL>declare
address varchar(20);
a_count number;
 begin
address:='dubai';
a_count:=st_count(address);
 dbms_output.put_line ('no of customers from dubai is '||a_count);
end;
/
```

```
SQL> create or replace function st_count(address varchar) return number is
  2  CNT number;
  3  begin
  4  select count(address) into CNT from customers where customers.address=address;
  5  return CNT;
  6  end;
  7  /

Function created.

SQL> declare
  2  address varchar(20);
  3  a_count number;
  4  begin
  5  address:='dubai';
  6  a_count:=st_count(address);
  7  dbms_output.put_line('no of customers from dubai is '||a_count);
  8  end;
  9  /
no of customers from dubai is 5

PL/SQL procedure successfully completed.
```

CURSORS

1) **Write a CURSOR to give 5% additional discount to all payment_id's whose final amount>101**

SQL> set serveroutput on;

SQL> declare

cursor pay is select payment_id, amount from payment;

PID payment.payment_id%type;

amount number;

begin

open pay;

loop

fetch pay into PID, amount;

exit when pay%not found;

if amount>101 then

update payment set discout=discout+5 where payment_id=PID;

end if;

end lo;

close pay;

end;

/

```
SQL> set serveroutput on;
SQL> declare
  2  cursor pay is select payment_id,amount from payment;
  3  PID payment.payment_id%type;
  4  amount number;
  5  begin
  6  open pay;
  7  loop
  8  fetch pay into PID,amount;
  9  exit when pay%notfound;
 10  if amount>101 then
 11  update payment set discout=discout+5 where payment_id=PID;
 12  end if;
 13  end loop;
 14  close pay;
 15  end;
 16  /

PL/SQL procedure successfully completed.

SQL> select * from payment;

PAYMENT_ID    CHARGES FINAL_AMOUNT    DISCOUT     AMOUNT
---------- ---------- ------------ ---------- ----------
p1                 1          101          0        101
p2                 2          202          5        202
p3                 3          303          5        303
p4                 4          404          5        404
```

2-Get the customer_id and name of the customers from tabe using implicit cursor and use ROWCOUNT

SQL> set serveroutput on

SQL> declare

c_id customer.customer_id%type;

c_name customer.last_name%type;

cursor cus_cursor is

select customer_id, last_name

from customer;

begin

open cus_cursor;

fetch cus_cursor into c_id, c_name;

exit when cus_cursor%ROWCOUNT>5 or cus_cursor%NOTFOUND;
dbms_output.put_line(c_id || ':' || c_name);

end loop;

close cus_cursor;

end;

/

```
SQL> set serveroutput on
SQL> declare
  2  c_id customer.customer_id%type;
  3  c_name customer.last_name%type;
  4  cursor cus_cursor is
  5        select customer_id,last_name
  6        from customer;
  7  begin
  8    open cus_cursor;
  9    loop
 10      fetch cus_cursor into c_id,c_name;
 11      exit when cus_cursor%ROWCOUNT>5 or cus_cursor%NOTFOUND;
 12       dbms_output.put_line(c_id || ' : ' || c_name);
 13  end loop;
 14  close cus_cursor;
 15  end;
 16  /
c1 : raju
c2 : venkata
c3 : karthik
c4 : mohan
c5 : babu

PL/SQL procedure successfully completed.
```

FUNCTIONS AND PROCEDURES

1. Write a PL/SQL stored procedure to adjust the payment type of orders to CASH if the payment_id and amount details given as input.

Sql>set serveroutput on

Create or replace procedure payment_udate

(pay_id in payment,payment_id%type,amount in payments.final_amount%type);

Is

Begin

Update payments set payment = ‘cash’ where payment_id=pay_id and final_amount=amount;

End payment_update; /

```
SQL> set serveroutput on
SQL> create or replace procedure payment_update
  2  (pay_id in payments.payment_id%type,amount in payments.final_amount%type)
  3  is
  4  begin
  5  update payments set payment = 'cash' where payment_id=pay_id and final_amount=amount;
  6  end payment_update;
  7  /

Procedure created.

SQL> select * from payments;

PAYMENT_ID FINAL_AMOUNT         PAYMENT
---------- -------------------- --------------------
p1         101                  debitcard
p2         202                  creditcard
p3         303                  upi
```

```
SQL> EXECUTE payment_update('p2','202');

PL/SQL procedure successfully completed.

SQL> select * from payments;

PAYMENT_ID FINAL_AMOUNT         PAYMENT
---------- -------------------- --------------------
p1         101                  debitcard
p2         202                  cash
p3         303                  upi
```

TRIGGERS

1.Write a Trigger to find and fill the age of a customer whenever a customer record is inserted into customer table.

sql>create or replace function get_age(cDOB date) return number is

begin

return floor(months_between(sysdate,cDOB)/12);

end;

/

Sql>create or replace trigger age before insert or update on customer

For each row

Begin

:new.age := get_age(: new.dob);

End;

/

```
SQL> create or replace function get_age(cDOB date) return number is
  2
  3  begin
  4     return floor(months_between(sysdate,cDOB)/12);
  5  end;
  6  /

Function created.

SQL> create or replace trigger age before insert or update on customer
  2  for each row
  3  begin
  4    :new.age := get_age(:new.dob);
  5  end;
  6  /

Trigger created.

SQL> select * from customer;

CUSTOMER_I NAME                 DOB           MOBILE STREET
---------- -------------------- --------- ---------- --------------------
       AGE
----------
c1         anand                20-OCT-01 9012345687 vellore
         0

c2         karthik              18-OCT-02 9012334567 hyd
         0

c3         narasimha            25-JAN-05 9134578497 kadapa
         0
```

```
SQL> insert into customer values('c4','kcr','12-oct-2004','9012348099','sdnr','2');

1 row created.

SQL> select * from customer;

CUSTOMER_I NAME                 DOB           MOBILE STREET
---------- -------------------- --------- ---------- --------------------
       AGE
----------
c1         anand                20-OCT-01 9012345687 vellore
         0

c2         karthik              18-OCT-02 9012334567 hyd
         0

c3         narasimha            25-JAN-05 9134578497 kadapa
         0


CUSTOMER_I NAME                 DOB           MOBILE STREET
---------- -------------------- --------- ---------- --------------------
       AGE
----------
c4         kcr                  12-OCT-04 9012348099 sdnr
        16
```

6-FRONTEND:

```
<?php
session_start();
include("../db.php");
include "sidenav.php";
include "topheader.php";
?>
<!-- End Navbar -->
<div class="content">
<div class="container-fluid">
<div class="panel-body">
<a>
<?php //success message
if(isset($_POST['success'])) {
$success = $_POST["success"];
echo "<h1 style='color:#0C0'>Your Product was added successfully    <span
class='glyphicon glyphicon-ok'></h1></span>";
}
?></a>
</div>
<div class="col-md-14">
<div class="card ">
<div class="card-header card-header-primary">
<h4 class="card-title"> Users List</h4>
</div>
<div class="card-body">
<div class="table-responsive ps">
<table class="table table-hover tablesorter " id="">
<thead class=" text-primary">
<tr><th>ID</th><th>FirstName</th><th>LastName</th><th>Email</th><th>Password</th
><
```

```
th>Contact</th><th>Address</th><th>City</th>
</tr></thead>
<tbody>
<?php
$result=mysqli_query($con,"select * from user_info")or die ("query 1 incorrect.....");
while(list($user_id,$first_name,$last_name,$email,$password,$phone,$address1,$address2)=
m
ysqli_fetch_array($result))
{
echo
"<tr><td>$user_id</td><td>$first_name</td><td>$last_name</td><td>$email</td><td>$pa
ss
word</td><td>$phone</td><td>$address1</td><td>$address2</td>
</tr>";
}
?>
</tbody>
</table>
<div class="ps__rail-x" style="left: 0px; bottom: 0px;"><div class="ps__thumb-x"
tabindex="0" style="left: 0px; width: 0px;"></div></div><div class="ps__rail-y" style="top:
0px; right: 0px;"><div class="ps__thumb-y" tabindex="0" style="top: 0px; height:
0px;"></div></div></div>
</div>
</div>
</div>
<div class="row">
<div class="col-md-6">
<div class="card ">
<div class="card-header card-header-primary">
<h4 class="card-title"> Categories List</h4>
</div>
```

```
<div class="card-body">
<div class="table-responsive ps">
<table class="table table-hover tablesorter " id="">
<thead class=" text-primary">
<tr><th>ID</th><th>Categories</th><th>Count</th>
</tr></thead>
<tbody>
<?php
$result=mysqli_query($con,"select * from categories")or die ("query 1 incorrect.....");
$i=1;
while(list($cat_id,$cat_title)=mysqli_fetch_array($result))
{
$sql = "SELECT COUNT(*) AS count_items FROM products WHERE product_cat=$i";
$query = mysqli_query($con,$sql);
$row = mysqli_fetch_array($query);
$count=$row["count_items"];
$i++;
echo "<tr><td>$cat_id</td><td>$cat_title</td><td>$count</td>
</tr>";
}
?>
</tbody>
</table>
<div class="ps__rail-x" style="left: 0px; bottom: 0px;"><div class="ps__thumb-x"
tabindex="0" style="left: 0px; width: 0px;"></div></div><div class="ps__rail-y" style="top:
0px; right: 0px;"><div class="ps__thumb-y" tabindex="0" style="top: 0px; height:
0px;"></div></div></div>
</div>
</div>
</div>
```

```
<div class="col-md-6">
<div class="card ">
<div class="card-header card-header-primary">
<h4 class="card-title">Brands List</h4>
</div>
<div class="card-body">
<div class="table-responsive ps">
<table class="table table-hover tablesorter " id="">
<thead class=" text-primary">
<tr><th>ID</th><th>Brands</th><th>Count</th>
</tr></thead>
<tbody>
<?php
$result=mysqli_query($con,"select * from brands")or die ("query 1 incorrect.....");
$i=1;
while(list($brand_id,$brand_title)=mysqli_fetch_array($result))
{
$sql = "SELECT COUNT(*) AS count_items FROM products WHERE product_brand=$i";
$query = mysqli_query($con,$sql);
$row = mysqli_fetch_array($query);
$count=$row["count_items"];
$i++;
echo "<tr><td>$brand_id</td><td>$brand_title</td><td>$count</td>
</tr>";
}
?>
</tbody>
</table>
<div class="ps__rail-x" style="left: 0px; bottom: 0px;"><div class="ps__thumb-x"
tabindex="0" style="left: 0px; width: 0px;"></div></div><div class="ps__rail-y" style="top:
```

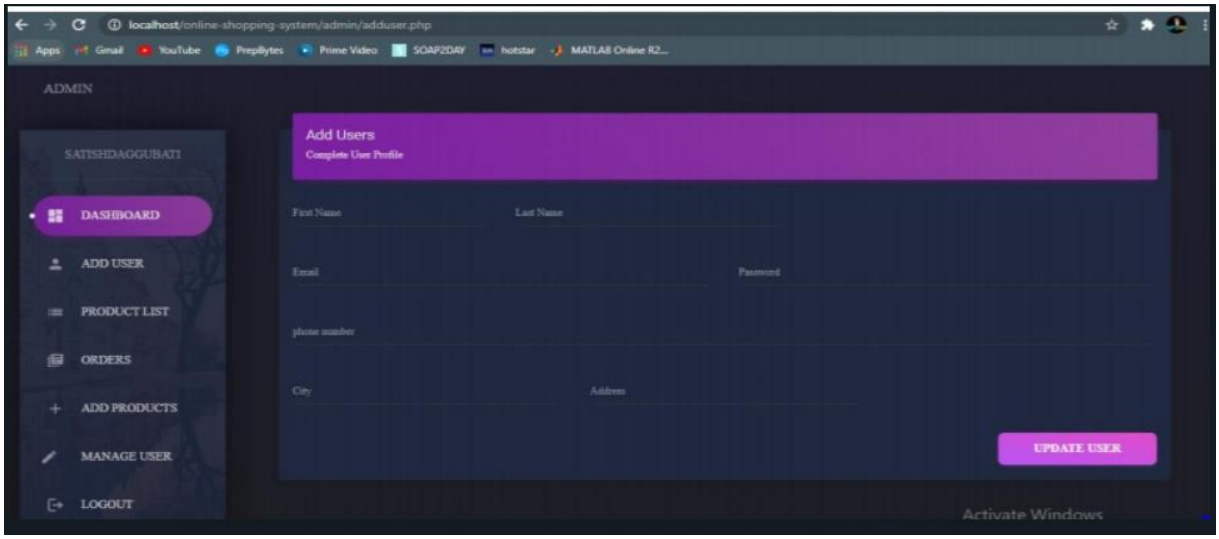
```
0px; right: 0px;"><div class="ps__thumb-y" tabindex="0" style="top: 0px; height:
0px;"></div></div></div>
</div>
</div>
</div>
</div>
<div class="col-md-5">
<div class="card ">
<div class="card-header card-header-primary">
<h4 class="card-title">Subscribers</h4>
</div>
<div class="card-body">
<div class="table-responsive ps">
<table class="table table-hover tablesorter " id="">
<thead class=" text-primary">
<tr><th>ID</th><th>email</th>
</tr></thead>
<tbody>
<?php
$result=mysqli_query($con,"select * from email_info")or die ("query 1 incorrect.....");
while(list($brand_id,$brand_title)=mysqli_fetch_array($result))
{
echo "<tr><td>$brand_id</td><td>$brand_title</td>
</tr>";
}
?>
</tbody>
</table>
<div class="ps__rail-x" style="left: 0px; bottom: 0px;"><div class="ps__thumb-x"
tabindex="0" style="left: 0px; width: 0px;"></div></div><div class="ps__rail-y" style="top:
```

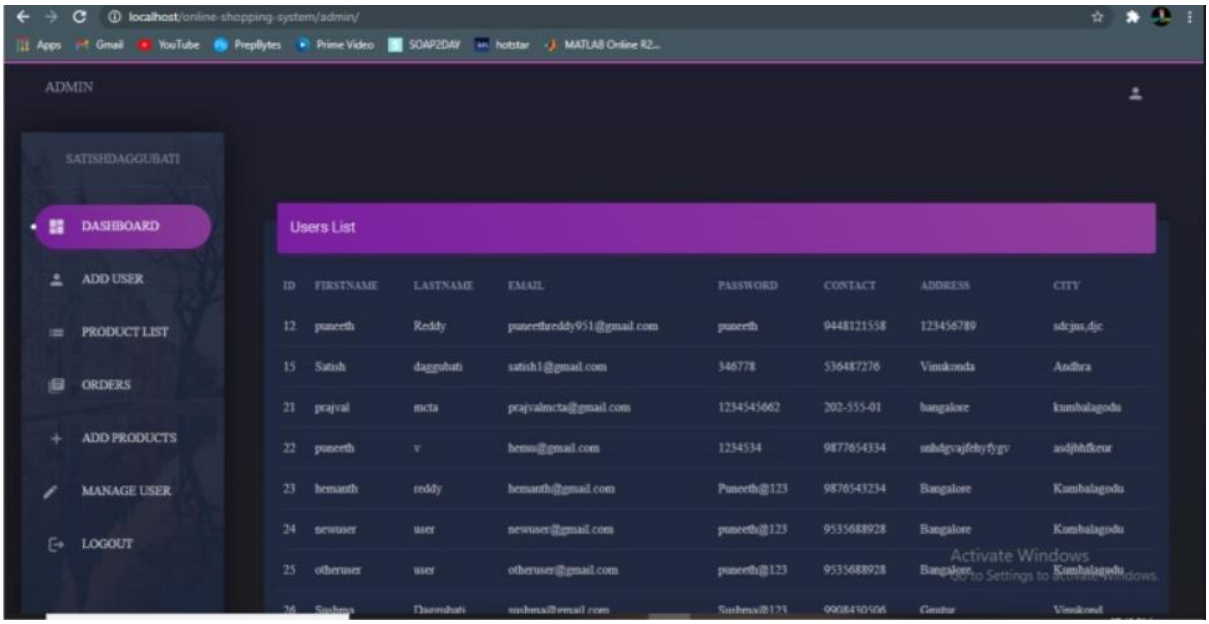
0px; right: 0px;">

<?php

include "footer.php";

?>





7-CONCLUSION AND FUTURE WORK:

A customer can purchase any product of his/her choice through our website by using internet. The customer will also get the information of the shop like its name and address. If the shop is near to the customer, he/she can purchase the desired product offline. If the shop is far from customer, he/she can purchase online.All shops who are registered on our website may become famous and their profit may increase. Since our system is providing name and address of the shop, there are a lot of chances that this system might become famous in the future. Moreover, the shop owners are getting benefit through this system and customers will not need to go here and there in search of the product. They just need to open the website and check that which shop is selling the required product currently. Shop owners just need to update quantity of the products time to time by logging in. Thus, customers can either do purchasing online or offline.

REFERENCES:

1- Aurelia Michaud-Trevinal, Thomas Stenger, “Online shopping experiences: a qualitative research”, Academy of Marketing Science Annual Conference, At New-Orleans, May 2012.

2-Shibin Chittil, Nidheesh Chittil, Rishikese M R, “Online Shopping System”, Computer Science & Engineering school of Engineering Cochin University of Science & Technology Kochi-682022 March 2014