CHAPTER 1
INTRODUCTION

The aim of this project is on the online shopping application which is developed using the front-end tools like HTML and CSS. The application is useful as the customers can buy the products listed by the seller in a very short amount of time and without the need of physically going to the shop to buy the products. The application intends to reduce the workload of the customer as well as the seller. The sale and purchase transaction are
completed electronically and interactively in real- time. The development of this new system contains the following activities, which try to develop online application by keeping the entire process in the view of database integration approach. User uses its email id and password to access their account.

Administrator of Online Shopping System has multiple features such as Add, Delete and Update shopping items.

Some of the salient features of the Online Shopping System are:

- Secure registration and profile management facilities for the customers.
- Browsing through the e-Mall to see the items that are listed under each category of products like Apparel, Kitchen accessories, Bath accessories, Food items etc.
- Creating a Shopping cart so that customer can shortlist the items of and finally checkout with the desired products.

CHAPTER 2
REQUIREMENT SPECIFICATION

2.1 Requirements

2.1.1 Hardware Requirements

Processor	:	Intel Core i3
Hard Disk	:	40 GB
RAM	:	1 GB or more

2.1.2 Software Requirements

Operating System	:	Windows 7 or above / Linux
User Interface	:	HTML, CSS
Back-end	:	PHP
Database	:	Oracle SQL 11g
Server Deployment	:	Xampp

2.2 Entities and their Attributes

The entities used in the database along with their attributes are as follows:

CUSTOMER	CustomerID	CustomerName	Password	Email	State	City	State
ORDER	OrderID	OrderDate	Amount				
ORDER DETAILS	Quantity	Status					
SUPPLIER	Location	Supplier Name	SupplierID				
PRODUCT	Product ID	Product Name	Price				
CATEGORY	CategoryID	CategoryName					

2.3 Relationship Types

The type of relationships used in the design of ER diagram are as follows:

Relationship	Type
PLACES	1: N
HAS	1:1
CONTAINS	N:1
BELONGS TO	N:1
SUPPLIES	M: N

CHAPTER 3
CONSTRAINTS

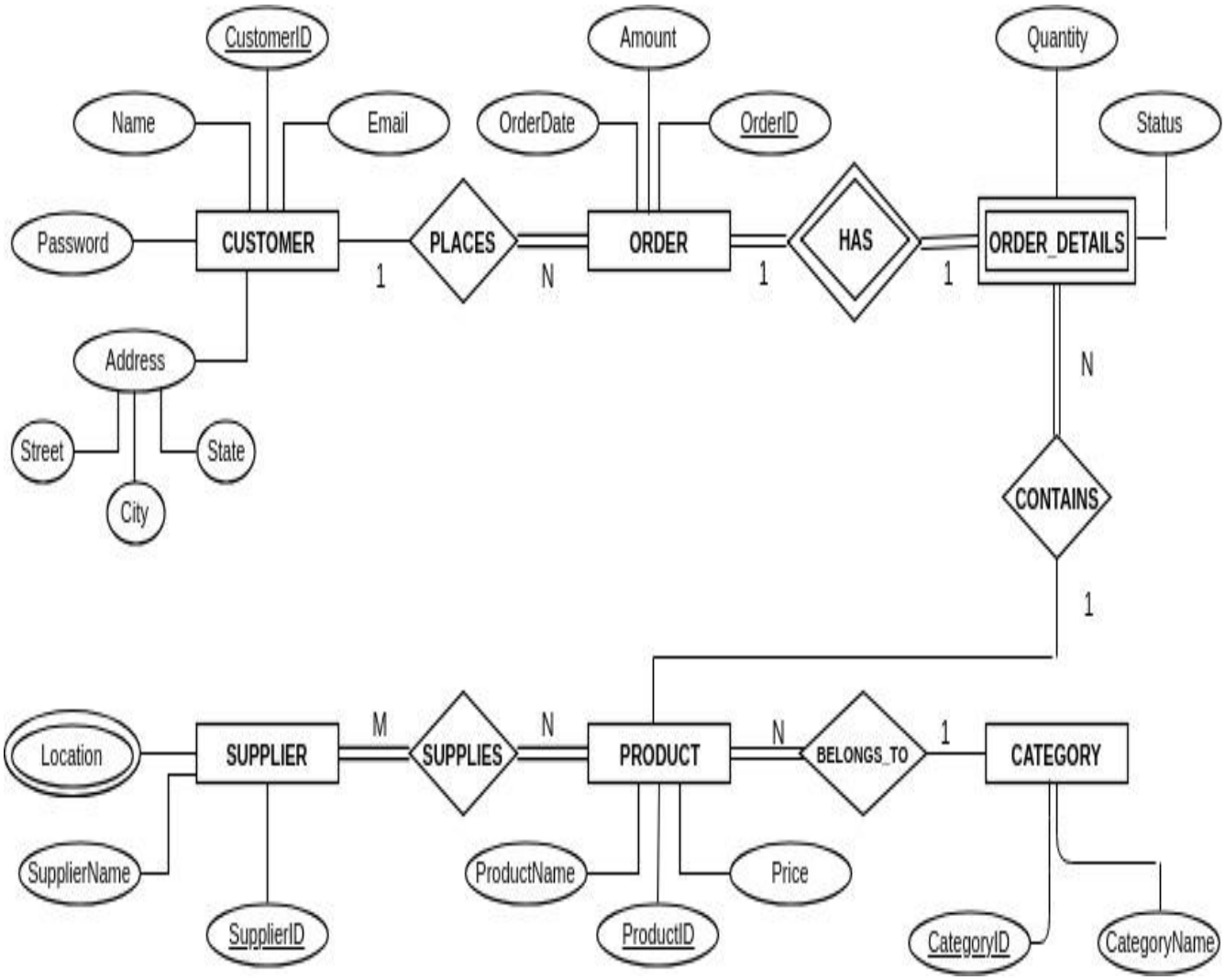
Some of the constraints used in the database are as follows:

1. The Customer details such as Customer Name, Email, Password and Address should not be NULL.
2. Order Date cannot be greater than the system date.
3. Order amount should be a valid positive integer.
4. The number of products under each category should not be greater than 20.
5. Order Quantity should be a valid non-negative integer.
6. Total number of categories should not exceed 10.
7. The Product Name should not be NULL and the Product Price should be a valid positive integer.
8. The Supplier Name should not be NULL.

CHAPTER 4
DESIGN

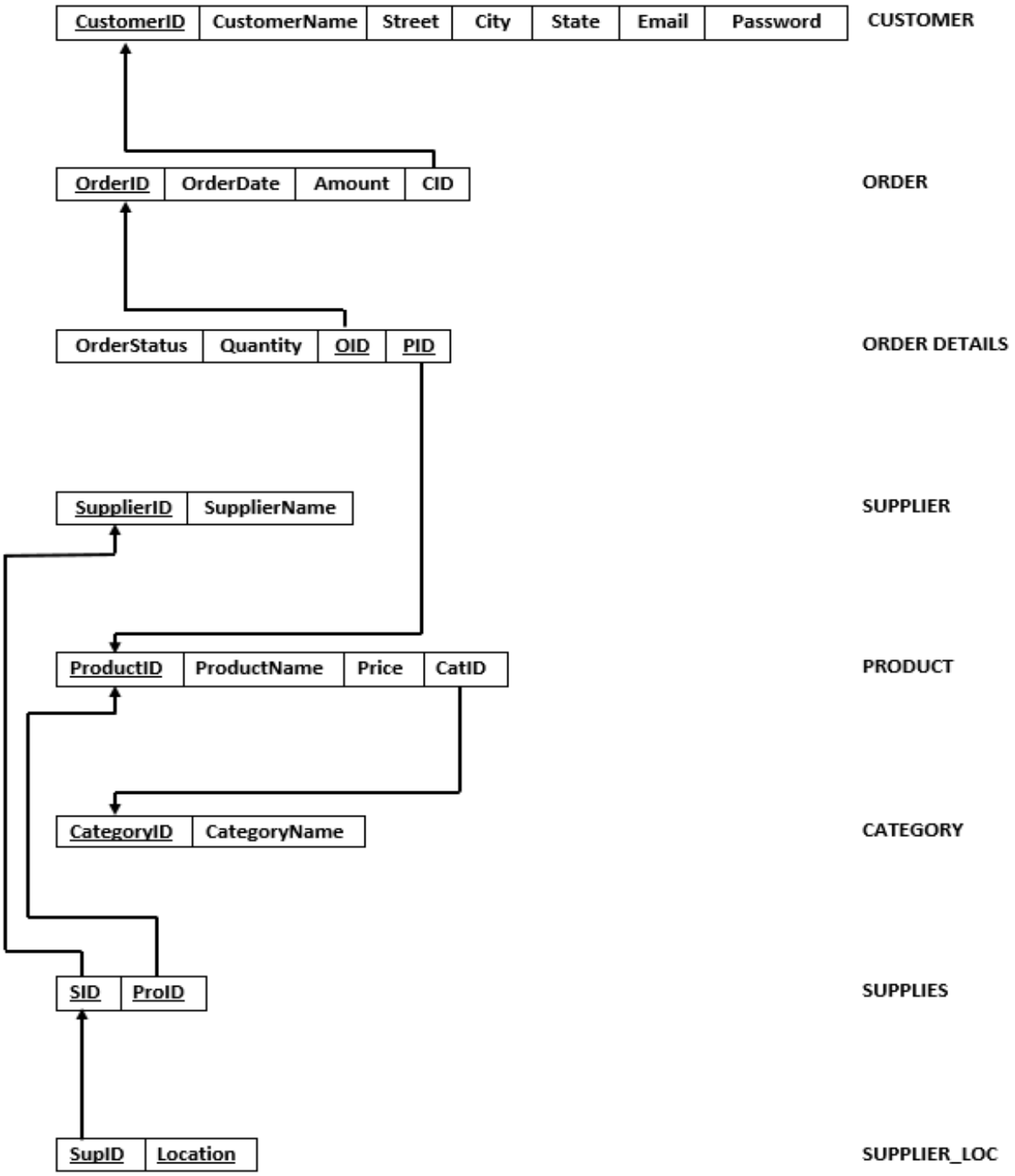
4.1 E-R Diagram

The ER Diagram which represents our database is mentioned below.



4.2 Relation Schema Diagram

The Relational Schema Diagram corresponding to the ER Diagram is mentioned below.



CHAPTER 5
IMPLEMENTATION

5.1 Oracle SQL

SQL Structured Query Language is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). It is particularly useful in handling structured data where there are relations between different entities/variables of the data.

SQL provides statements for a variety of tasks, including:

- Querying data
- Inserting, updating, and deleting rows in a table
- Creating, replacing, altering, and dropping objects
- Controlling access to the database and its objects
- Guaranteeing database consistency and integrity

SQL unifies all of the preceding tasks in one consistent language.

5.2 Create Table Queries

Some of the queries used to create the database are:

```
Select Run SQL Command Line
SQL> connect akshat;
Enter password:
Connected.
SQL> CREATE TABLE customer(
  2    CustomerID int Primary Key,
  3    CustomerName varchar(20) NOT NULL,
  4    Street varchar(20) NOT NULL,
  5    City varchar(15) NOT NULL,
  6    State varchar(15) NOT NULL,
  7    Email varchar(30) NOT NULL,
  8    Password varchar(10) NOT NULL
  9  );

Table created.

SQL>
SQL> CREATE TABLE orders (
  2    OrderID int Primary Key,
  3    OrderDate date NOT NULL,
  4    Amount float NOT NULL,
  5    CID int NOT NULL,
  6    Foreign Key(CID) references customer(CustomerID)
  7  );

Table created.

SQL>
SQL> CREATE TABLE category (
  2    CategoryID int Primary Key,
  3    CategoryName varchar(20) NOT NULL
  4  );

Table created.
```

```
Select Run SQL Command Line
SQL> CREATE TABLE product (
  2    ProductID int Primary Key,
  3    ProductName varchar(30) NOT NULL,
  4    Price float NOT NULL,
  5    CatID int NOT NULL,
  6    Foreign Key(CatID) references category(CategoryID)
  7  );

Table created.

SQL>
SQL> CREATE TABLE orderdetails (
  2  OrderStatus varchar(20),
  3  Quantity int Not Null,
  4    OID int NOT NULL,
  5    PID int NOT NULL,
  6    Foreign Key(OID) references Orders(OrderID),
  7    Foreign Key(PID) references Product(ProductID)
  8  );

Table created.

SQL>
SQL> CREATE TABLE suppliers (
  2    SupplierID int Primary Key,
  3    SupplierName varchar(30) NOT NULL
  4  );

Table created.
```

```
Select Run SQL Command Line
SQL> CREATE TABLE supplies(
  2    SID int,
  3    ProID int,
  4    Primary Key(SID,PROID),
  5    Foreign Key(SID) references Suppliers(SupplierID),
  6    Foreign Key(ProID) references PRODUCT(ProductID)
  7  );

Table created.

SQL>
SQL> CREATE TABLE SupplierLocation (
  2    SupID int,
  3    Location varchar(30),
  4    Primary Key(SupID,Location),
  5    Foreign Key(SupID) references Suppliers(SupplierID)
  6  );

Table created.
```

5.3 Insert Queries

```
Run SQL Command Line

SQL> INSERT INTO customer VALUES (1, 'Ram', 'Chandni chowk', 'NewDelhi', 'Delhi', 'ram@gmail.com', 'ram');

1 row created.

SQL> INSERT INTO customer VALUES (2, 'Rita', 'jump street', 'LA', 'USA', 'rita@gmail.com', 'rita');

1 row created.

SQL> INSERT INTO customer VALUES (3, 'Anannya Uberoi', 'Shalimar', 'Delhi', 'Delhi', 'anannya@gmail.com', 'anannya');

1 row created.

SQL> INSERT INTO customer VALUES (4, 'Sarthika', 'Rohini', 'Delhi', 'Delhi', 'sarthika@gmail.com', 'sarthika');

1 row created.

SQL> INSERT INTO customer VALUES (5, 'Katy', 'Albany', 'NY', 'NY', 'katy@gmail.com', 'katy');

1 row created.

SQL> INSERT INTO customer VALUES (6, 'Logan', 'Mumbai', 'Mumbai', 'Maharashtra', 'logan@gmail.com', 'logan');

1 row created.

SQL>
SQL> INSERT INTO orders VALUES (1, '15-APR-2017', 500, 1);

1 row created.

SQL> INSERT INTO orders VALUES (2, '15-APR-2017', 500, 5);

1 row created.

SQL> INSERT INTO orders VALUES (3, '15-APR-2017', 500, 1);

1 row created.
```

```
Run SQL Command Line
SQL> INSERT INTO category VALUES (111, 'Women');

1 row created.

SQL> INSERT INTO category VALUES (112, 'Men');

1 row created.

SQL> INSERT INTO category VALUES (113, 'Kids');

1 row created.

SQL>
SQL> INSERT INTO product VALUES (11, 'Top', 500, 111);

1 row created.

SQL> INSERT INTO product VALUES (13, 'Frock', 200, 113);

1 row created.

SQL> INSERT INTO product VALUES (14, 'Trouser', 1000, 112);

1 row created.

SQL> INSERT INTO product VALUES (15, 'Baby Suit', 500, 113);

1 row created.

SQL> INSERT INTO product VALUES (16, 'Baby Suit', 200, 113);

1 row created.

SQL> INSERT INTO product VALUES (17, 'Shirt', 780, 112);

1 row created.
```

Run SQL Command Line

```
SQL> INSERT INTO product VALUES (18, 'Shirt', 735, 112);

1 row created.

SQL> INSERT INTO product VALUES (19, 'Jeans', 400, 112);

1 row created.

SQL> INSERT INTO product VALUES (20, 'Skirt', 290, 111);

1 row created.

SQL> INSERT INTO product VALUES (21, 'Skirt', 879, 111);

1 row created.

SQL> INSERT INTO product VALUES (22, 'Top', 700, 111);

1 row created.

SQL> INSERT INTO product VALUES (23, 'Top', 250, 111);

1 row created.

SQL> INSERT INTO product VALUES (24, 'Top', 988, 111);

1 row created.

SQL> INSERT INTO product VALUES (25, 'Baby Suit', 788, 113);

1 row created.

SQL> INSERT INTO product VALUES (26, 'Baby Suit', 877, 113);

1 row created.
```

```
Run SQL Command Line
SQL> INSERT INTO product VALUES (27, 'Baby Suit', 900, 113);

1 row created.

SQL> INSERT INTO product VALUES (28, 'Fancy Clothes', 100, 113);

1 row created.

SQL> INSERT INTO product VALUES (29, 'Fancy Clothes', 800, 113);

1 row created.

SQL> INSERT INTO product VALUES (30, 'Sweater', 1000, 112);

1 row created.

SQL> INSERT INTO product VALUES (31, 'Tuxedo', 2900, 112);

1 row created.

SQL> INSERT INTO product VALUES (32, 'Jeans', 700, 112);

1 row created.

SQL> INSERT INTO product VALUES (33, 'Sweater', 800, 112);

1 row created.

SQL> INSERT INTO product VALUES (34, 'Tee Shirt', 500, 112);

1 row created.

SQL>
SQL> INSERT INTO orderdetails VALUES ('confirmed', 3, 1, 11);

1 row created.
```

Run SQL Command Line

```
SQL> INSERT INTO orderdetails VALUES ('confirmed', 4, 2, 15);

1 row created.

SQL> INSERT INTO orderdetails VALUES ('confirmed', 1, 3, 11);

1 row created.

SQL>
SQL> INSERT INTO suppliers VALUES (11111, 'Dhanraj Textiles');

1 row created.

SQL> INSERT INTO suppliers VALUES (11112, 'Pricely');

1 row created.

SQL> INSERT INTO suppliers VALUES (11113, 'ABCD Suppliers');

1 row created.

SQL> INSERT INTO suppliers VALUES (11114, 'Mango Biz');

1 row created.

SQL> INSERT INTO suppliers VALUES (11115, 'Lilliput Clothing');

1 row created.

SQL>
SQL> INSERT INTO supplies (SID, ProID) VALUES (11111, 14);

1 row created.

SQL> INSERT INTO supplies (SID, ProID) VALUES (11112, 31);

1 row created.
```

```
Run SQL Command Line
SQL> INSERT INTO supplies (SID, ProID) VALUES (11114, 13);

1 row created.

SQL> INSERT INTO supplies (SID, ProID) VALUES (11115, 14);

1 row created.

SQL> INSERT INTO supplies (SID, ProID) VALUES (11111, 20);

1 row created.

SQL> INSERT INTO supplies (SID, ProID) VALUES (11112, 21);

1 row created.

SQL> INSERT INTO supplies (SID, ProID) VALUES (11113, 22);

1 row created.

SQL> INSERT INTO supplies (SID, ProID) VALUES (11114, 23);

1 row created.

SQL> INSERT INTO supplies (SID, ProID) VALUES (11115, 24);

1 row created.

SQL>
SQL> INSERT INTO SupplierLocation (SupID, Location) VALUES (11111, 'Delhi');

1 row created.

SQL> INSERT INTO SupplierLocation (SupID, Location) VALUES (11112, 'Maladi');

1 row created.
```

```
Run SQL Command Line
SQL> INSERT INTO SupplierLocation (SupID, Location) VALUES (11113, 'Juhu');

1 row created.

SQL> INSERT INTO SupplierLocation (SupID, Location) VALUES (11114, 'Green Park');

1 row created.

SQL> INSERT INTO SupplierLocation (SupID, Location) VALUES (11115, 'Bhiwandi');

1 row created.
```

```
Run SQL Command Line
SQL> SELECT ProductName, Price, CategoryName
  2  FROM PRODUCT, CATEGORY
  3  WHERE CategoryName = 'Men' AND CategoryID = CatID ;

PRODUCTNAME                          PRICE CATEGORYNAME
------------------------------ ---------- --------------------
Trouser                              1000 Men
Shirt                                 780 Men
Shirt                                 735 Men
Jeans                                 400 Men
Sweater                              1000 Men
Tuxedo                               2900 Men
Jeans                                 700 Men
Sweater                               800 Men
Tee Shirt                             500 Men

9 rows selected.
```

5.4 Simple Queries

1. Display all the customer details who lives in Delhi**.**

```
Run SQL Command Line
SQL> SELECT *
  2  FROM customer
  3  WHERE state = 'Delhi';

CUSTOMERID CUSTOMERNAME        STREET             CITY           STATE          EMAIL                      PASSWORD
---------- ------------------- ------------------- -------------- -------------- ---------------------------- ----------
         1 Ram                Chandni chowk      NewDelhi       Delhi          ram@gmail.com                ram
         3 Anannya Uberoi     Shalimar           Delhi          Delhi          anannya@gmail.com            anannya
         4 Sarthika           Rohini             Delhi          Delhi          sarthika@gmail.com           sarthika
```

2. Retrieve all the tops having price greater than 200 rupees from the product table.

```
SQL> SELECT *
  2  FROM Product
  3  WHERE ProductName = 'Top' AND Price>200.0;

 PRODUCTID PRODUCTNAME                          PRICE      CATID
---------- ------------------------------ ---------- ----------
        11 Top                                   500        111
        22 Top                                   700        111
        23 Top                                   250        111
        24 Top                                   988        111
```

3. Increase the price of each item by 10%.

```
Run SQL Command Line
SQL> update product set price=price*1.1;

23 rows updated.

SQL> select * from product;

 PRODUCTID PRODUCTNAME                          PRICE      CATID
---------- ------------------------------ ---------- ----------
        11 Top                                   550        111
        13 Frock                                 220        113
        14 Trouser                              1100        112
        15 Baby Suit                             550        113
        16 Baby Suit                             220        113
        17 Shirt                                 858        112
        18 Shirt                                808.5        112
        19 Jeans                                 440        112
```

4. Add check constraint to price in product table, which should allow only positive values.

```
Run SQL Command Line

SQL> alter table product add constraint ck_price
  2  check(price>0);

Table altered.

SQL> insert into product values (3, 'Frock', 0, 111);
insert into product values (3, 'Frock', 0, 111)
*
ERROR at line 1:
ORA-02290: check constraint (AKSHAT.CK_PRICE) violated
```

5. Display the number of products under each category.

```
Run SQL Command Line
SQL> select categoryname, count(*)
  2  from category C, product P
  3  Where P.catID = C.CategoryID
  4  group by C.categoryname;

CATEGORYNAME           COUNT(*)
-------------------- ----------
Women                         6
Men                           9
Kids                          8
```

6. Retrieve Min and Max Order Amount.

```
Select Run SQL Command Line
SQL> select MAX(Amount), MIN(Amount) from orders;

MAX(AMOUNT) MIN(AMOUNT)
----------- -----------
        500         500
```

7. Produce the list of order b/w JAN 2000 to JAN 2019.

```
Run SQL Command Line
SQL> select * from orders where orderdate between '01-JAN-2000' AND '31-JAN-2019';

   ORDERID ORDERDATE     AMOUNT        CID
---------- --------- ---------- ----------
         1 15-APR-17        500          1
         2 15-APR-17        500          5
         3 15-APR-17        500          1
```

8. For all orders in orders table, show their status and quantity.

```
Run SQL Command Line
SQL> select * from orders o left outer join orderdetails d on o.orderid=d.oid;

   ORDERID ORDERDATE     AMOUNT        CID ORDERSTATUS           QUANTITY        OID        PID
---------- --------- ---------- ---------- -------------------- ---------- ---------- ----------
         1 15-APR-17        500          1 confirmed                    3          1         11
         1 15-APR-17        500          1 confirmed                    3          1         11
         2 15-APR-17        500          5 confirmed                    4          2         15
         2 15-APR-17        500          5 confirmed                    4          2         15
         3 15-APR-17        500          1 confirmed                    1          3         11
         3 15-APR-17        500          1 confirmed                    1          3         11

6 rows selected.
```

9. For each category c list all the products that belong to c.

```
Run SQL Command Line

SQL> select * from category c full outer join product d on c.categoryid=d.catid;

CATEGORYID CATEGORYNAME         PRODUCTID PRODUCTNAME                      PRICE      CATID
---------- ------------------- --------- ------------------------------ ---------- ----------
       111 Women                      11 Top                                 550        111
       113 Kids                       13 Frock                               220        113
       112 Men                        14 Trouser                            1100        112
       113 Kids                       15 Baby Suit                           550        113
       113 Kids                       16 Baby Suit                           220        113
       112 Men                        17 Shirt                               858        112
       112 Men                        18 Shirt                             808.5        112
       112 Men                        19 Jeans                               440        112
       111 Women                      20 Skirt                               319        111
```

10. Retrieve all the supplier id who supplies at least one product.

```
Run SQL Command Line
SQL> select supplierID from suppliers
  2  intersect
  3  Select SID from supplies;

SUPPLIERID
----------
     11111
     11112
     11113
     11114
     11115
```

11. Retrieve the supplier ID who do not supply any product.

```
Run SQL Command Line
SQL> select supplierID from suppliers
  2  minus
  3  Select SID from supplies;

no rows selected
```

5.5 Nested Queries

1. Retrieve the supplier name and id number which supplies maximum number of products.

```
Run SQL Command Line
SQL> INSERT INTO supplies (SID, ProID) VALUES (11115, 30);

1 row created.

SQL> INSERT INTO supplies (SID, ProID) VALUES (11115, 25);

1 row created.

SQL>
SQL> select SupplierName, SupplierID, count(s1.SID) as number_of_products
  2  from suppliers s, supplies s1
  3  where s.SupplierID = s1.SID
  4  group by SupplierName, SupplierID
  5  having count(s1.SID) >= all (select count(x.SID)
  6   from supplies x
  7   group by x.SID);

SUPPLIERNAME                   SUPPLIERID NUMBER_OF_PRODUCTS
------------------------------ ---------- ------------------
Lilliput Clothing                   11115                  4
```

2. Retrieve the supplier names and their id number who supplies maximum number of products.

```
Run SQL Command Line
SQL> select SupplierName, SupplierID, count(s1.SID) as number_of_products
  2  from suppliers s, supplies s1
  3  where s.SupplierID = s1.SID
  4  group by SupplierName, SupplierID
  5  having count(s1.SID) <= all (select count(x.SID)
  6   from supplies x
  7   group by x.SID);

SUPPLIERNAME                   SUPPLIERID NUMBER_OF_PRODUCTS
------------------------------ ---------- ------------------
ABCD Suppliers                      11113                  2
Dhanraj Textiles                    11111                  2
Pricely                             11112                  2
Mango Biz                           11114                  2
```

5.6 Triggers

1. Write a trigger to notify back order quantity with suitable message when product quantity in a category crosses 5.

```
Run SQL Command Line
SQL> create or replace trigger max_product
  2  before insert on product
  3  for each row
  4  declare
  5  cnt number;
  6  begin
  7  select count(*) into cnt from product
  8  where CatID=:NEW.CatID;
  9  if(cnt>5) then
 10  raise_application_error(-20009,'MAX Product LIMIT REACHED');
 11  end if;
 12  end;
 13  /

Trigger created.

SQL> INSERT INTO product VALUES (40, 'Sweater', 800, 112);
INSERT INTO product VALUES (40, 'Sweater', 800, 112)
            *
ERROR at line 1:
ORA-20009: MAX Product LIMIT REACHED
ORA-06512: at "AKSHAT.MAX_PRODUCT", line 7
ORA-04088: error during execution of trigger 'AKSHAT.MAX_PRODUCT'
```

2. Write a trigger to notify back with suitable error message when date of order crosses current date.

```
Run SQL Command Line
SQL> create or replace trigger orders_date
  2  before insert on orders
  3  for each row
  4  declare
  5  cur date;
  6  begin
  7  select sysdate into cur from dual;
  8  if(cur<:NEW.OrderDate) then
  9  raise_application_error(-20009,'Incorrect Date! Order Date should be past or a current date.');
 10  end if;
 11  end;
 12  /

Trigger created.

SQL> INSERT INTO orders VALUES (4, '10-APR-2020', 500, 1);
INSERT INTO orders VALUES (4, '10-APR-2020', 500, 1)
            *
ERROR at line 1:
ORA-20009: Incorrect Date! Order Date should be past or a current date.
ORA-06512: at "AKSHAT.ORDERS_DATE", line 6
ORA-04088: error during execution of trigger 'AKSHAT.ORDERS_DATE'
```

5.7 Stored Procedures

1. Write a Stored procedure to display the details of order which are ordered on specific Order-Date.

```
Run SQL Command Line
SQL> create or replace procedure pr_order_date(s date)
  2  is
  3  x orders.OrderID%type;
  4  cursor c is select *
  5  from orders c
  6  where c.OrderDate = s;
  7  begin
  8  sys.dbms_output.put_line('OrderID          OrderDate');
  9  for x in c loop
 10  sys.dbms_output.put_line(x.OrderID||'                 '||x.OrderDate);
 11  end loop;
 12  end;
 13  /

Procedure created.

SQL>
SQL> set serveroutput on;
SQL>
SQL> begin
  2  pr_order_date('15-APR-2017');
  3  end;
  4  /
OrderID          OrderDate
1                 15-APR-17
2                 15-APR-17
3                 15-APR-17

PL/SQL procedure successfully completed.
```

CHAPTER 6
CONCLUSION

The central concept of the application is to allow the customer to shop virtually using the internet and allow customers to buy the items and articles of their desire from the store. The information pertaining to the products are stored in a database at the server side. The server process the customer requirements and the items are shipped to the address submitted by the customer.

The application is designed into two modules. First, for the customers who wish to buy the products and second is for the storekeepers who maintains and updates the information pertaining to the products.

The end user of this product is a departmental store where the application can be hosted on the web and the administrator maintains the database. The details of the items are brought forward from the database for the customer view based on the selection through the menu and database of all the products are updated ate the end of each transaction.

REFERENCES

1. https://www.oracle.com/in/database/
2. https://en.wikipedia.org/wiki/SQL
3. https://www.codeproject.com/