
# American International University- Bangladesh
# Faculty of CSE

## Advance Database Management System

Project Name:Superstore Management System

## Members NAME

| Name |
| --- |
| Md. Didarul Alam Alif |

# Table of Contents

## Introduction

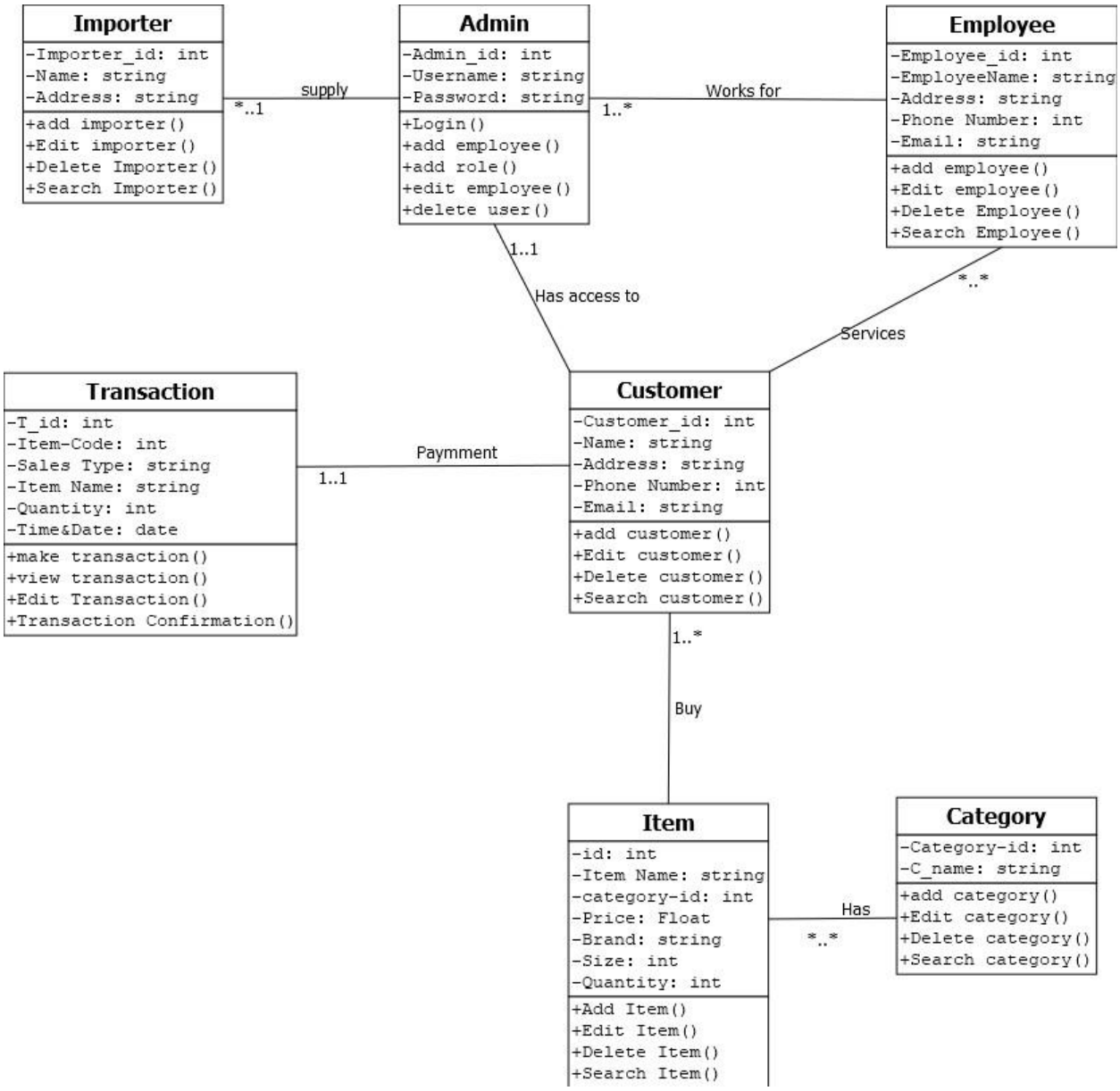
The purpose of database design for Super Store Management System is to assist in the buying of products, billing, and making purchase transactions quick and accurate. This management system supports in the storage and security of sales data. It allows employees and company owners to effortlessly monitor and modify business transactions. It could save time by eliminating the need to write down all of the pertinent facts and data about the business transaction and the customer. This system processes and saves all of the store's data and information. All information on customers, employees, items, sales, importers, and transactions will be stored in the system's database. This management system enables the owner to keep track of the items or sales that the consumer purchases. The user or authorized individual can only log in as a staff administrator on the Cashier system.

## Project Proposal

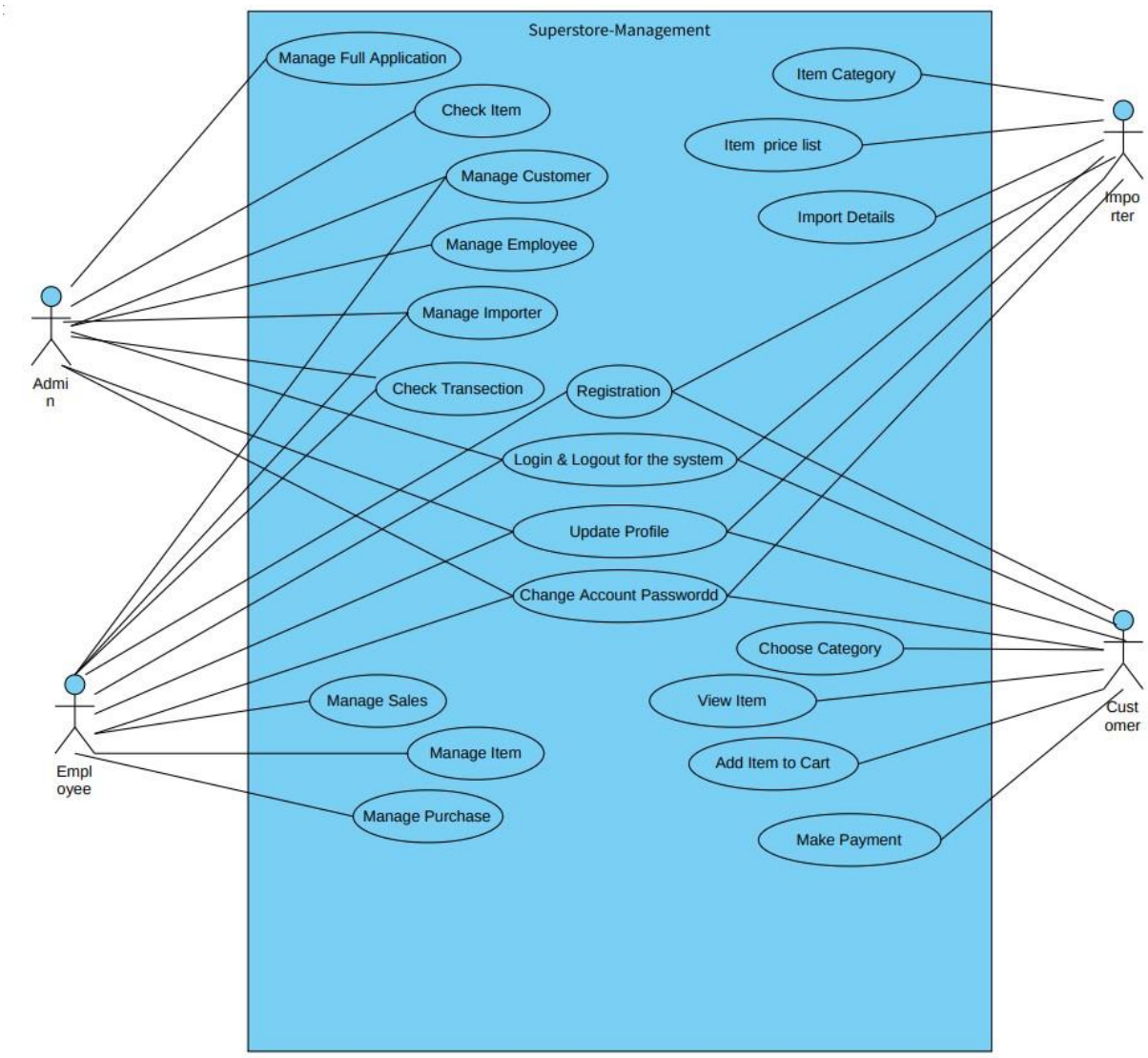
A super store is a large store that stocks many varieties of good in different departments. This is a retail offering a large number of consumer goods belonging to different product catagories. As we know a superstore deal with lots of importers, customers and

stuffs. This superstore management system has realized the transmission and control of large goods, so as to facilitate the management and decision of sales, and reduce a big burden for superstore and superstore managers. It also can help to improve the work efficiency of superstore. Its requirements is to provide the basic information maintenance function of employees, customers and products so that managers can through the function to add, delete, and modify the basic information of employees and the employees can through it to add, modify and delete the basic information of customers and goods. A super store allowing the customers to buy his requirements under one roof. This project 'super store management system ' which is designed to handles all the transaction done in the shop. This system is developed to computerize the store works. When a bundle of products are purchased from the importers, all the details are noted down along with the importers name, id, phone no and email. The quantity purchased is entered in the stock. Initially the name of the product is saved in the database.as like customers are buying their requirement products and all the details are noted along with customers name, id, address, phone no, email and also noted down the payment transection.

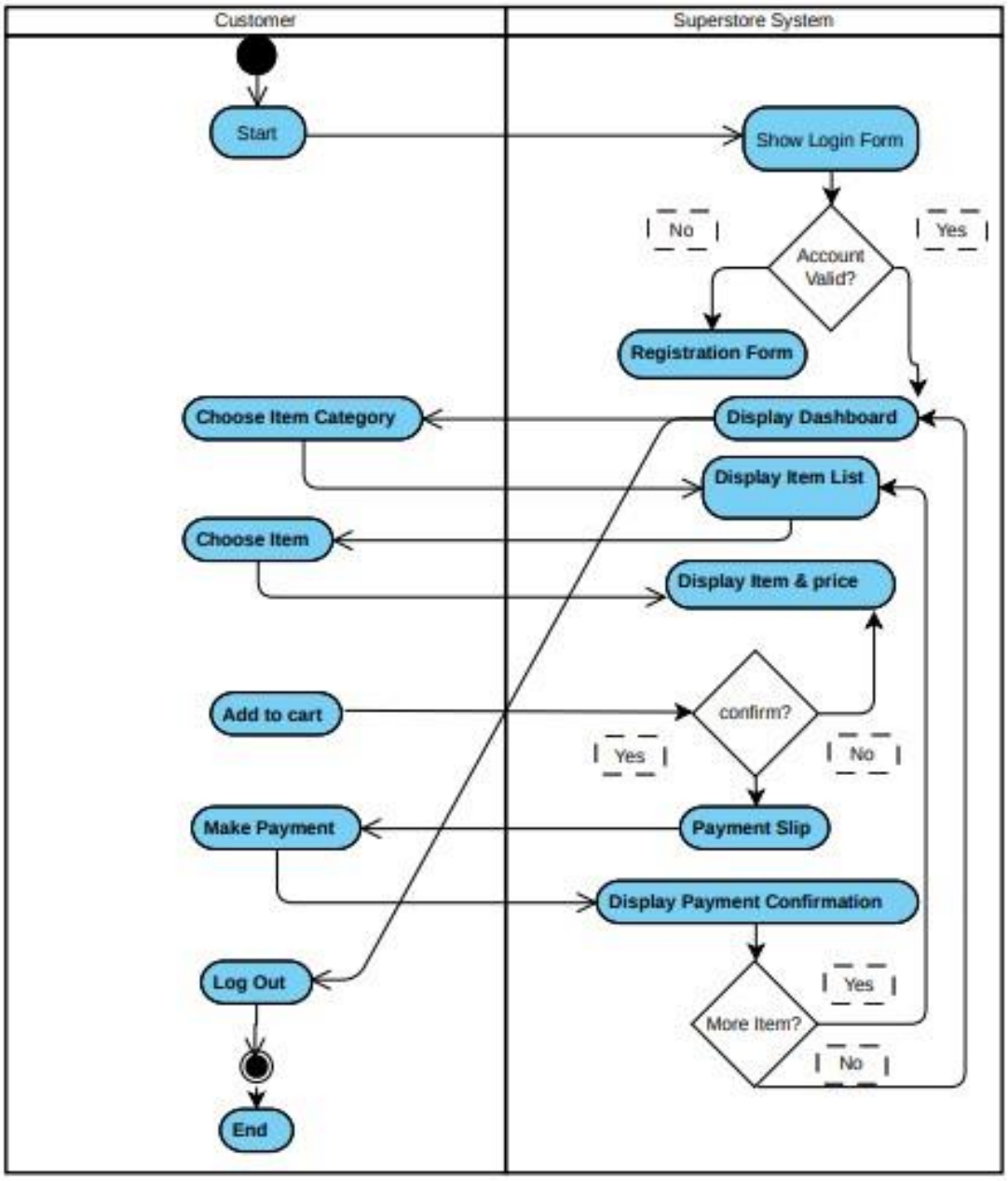
## Class Diagram



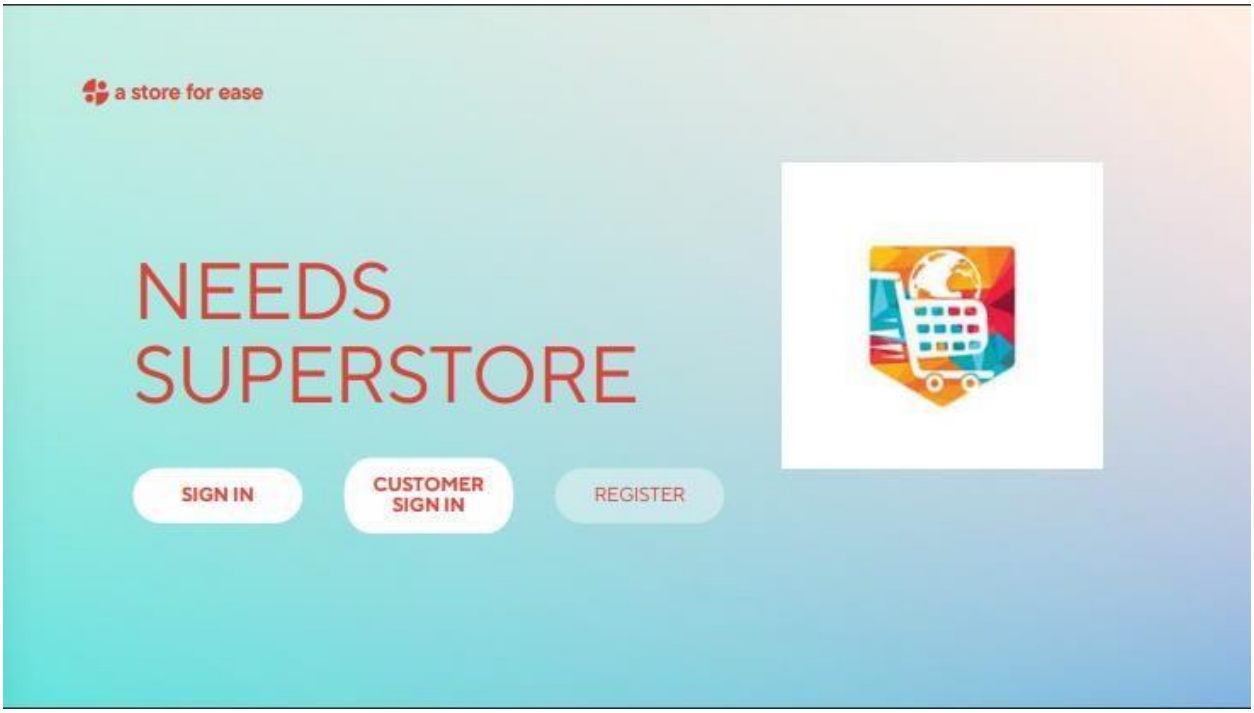

# Use case diagram

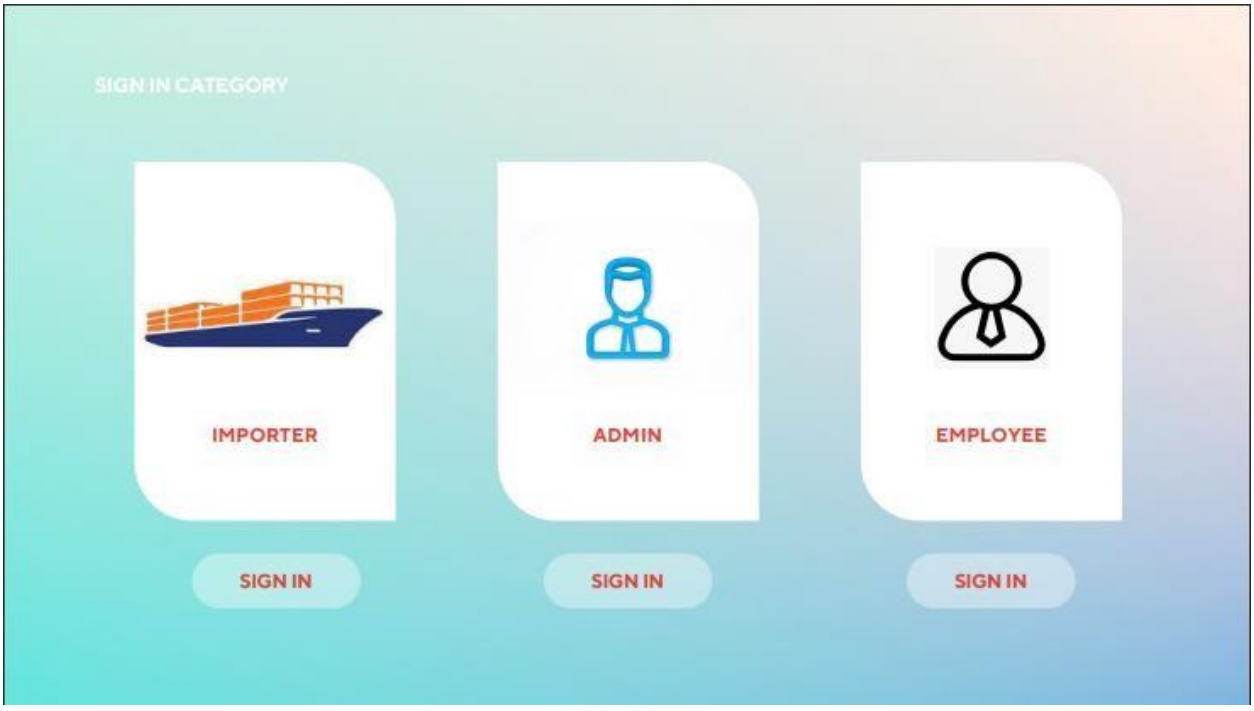


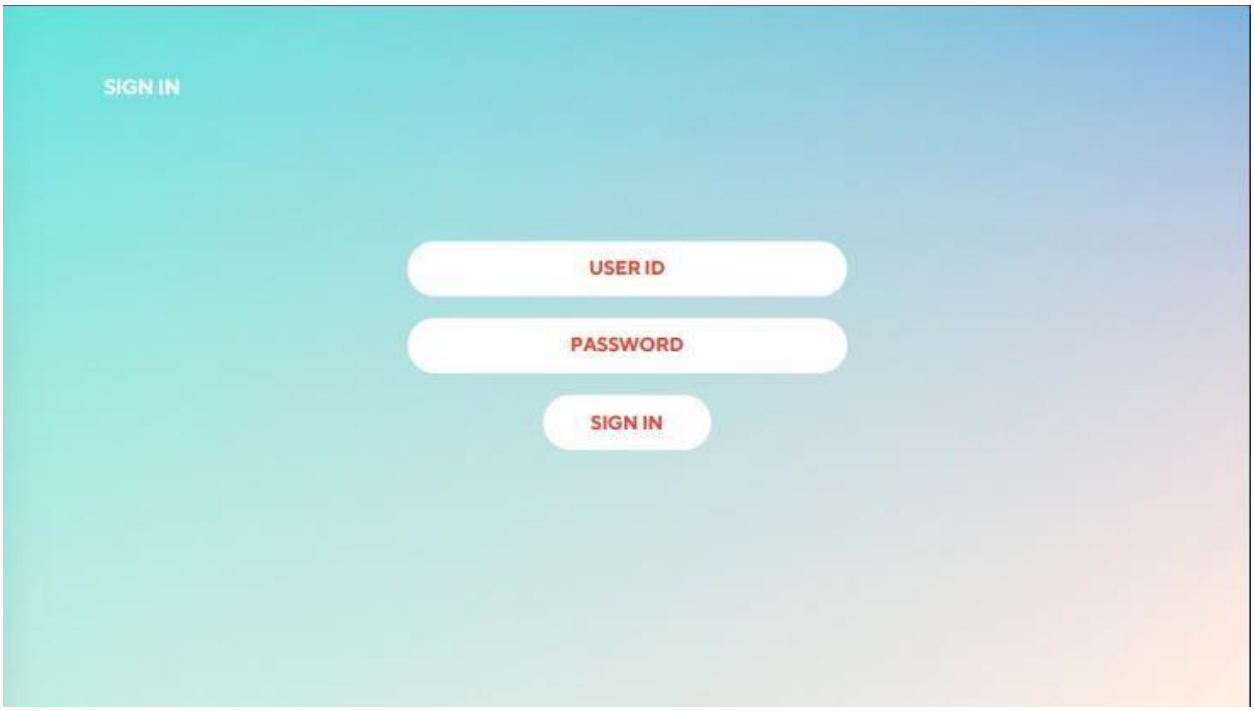

# Activity Diagram

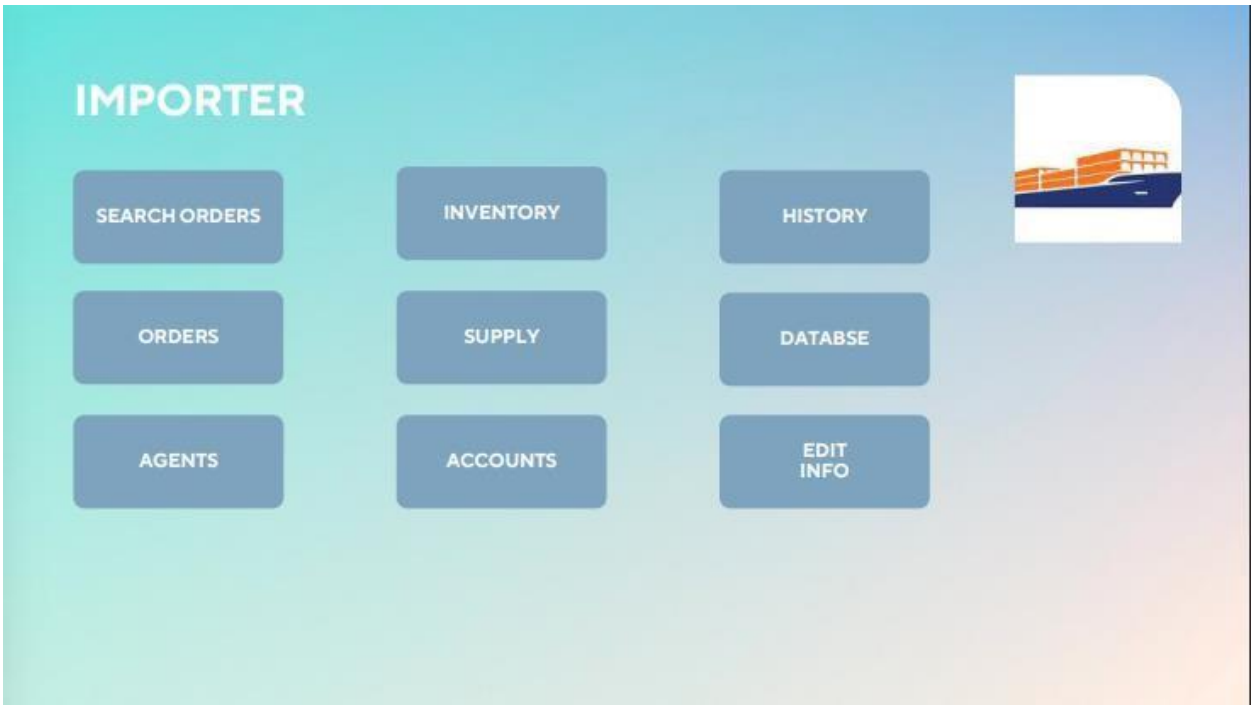

Customer
Superstore System
Start
Show Login Form
No
Account Valid?
Yes
Registration Form
Choose Item Category
Display Dashboard
Display Item List
Choose Item
Display Item & price
Add to cart
confirm?
Yes
No
Make Payment
Payment Slip
Display Payment Confirmation
Log Out
Yes
More Item?
No
End

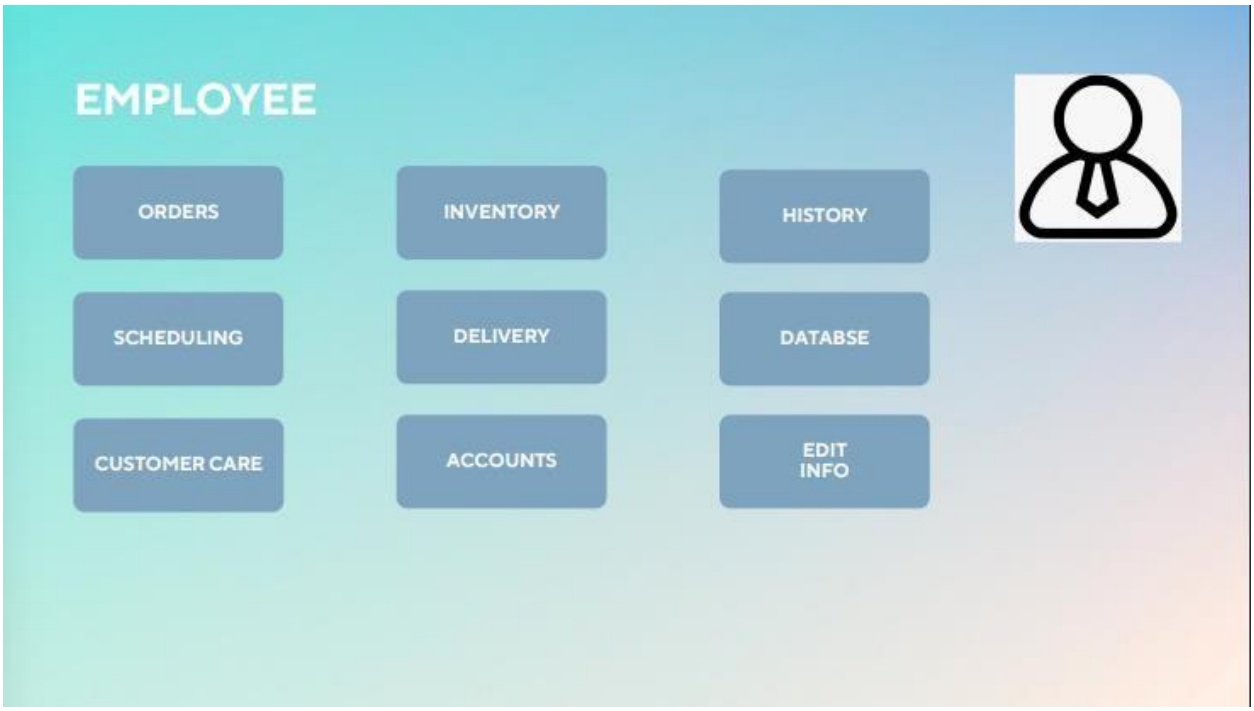
# Interface


a store for ease
NEEDS SUPERSTORE
SIGN IN
CUSTOMER SIGN IN
REGISTER

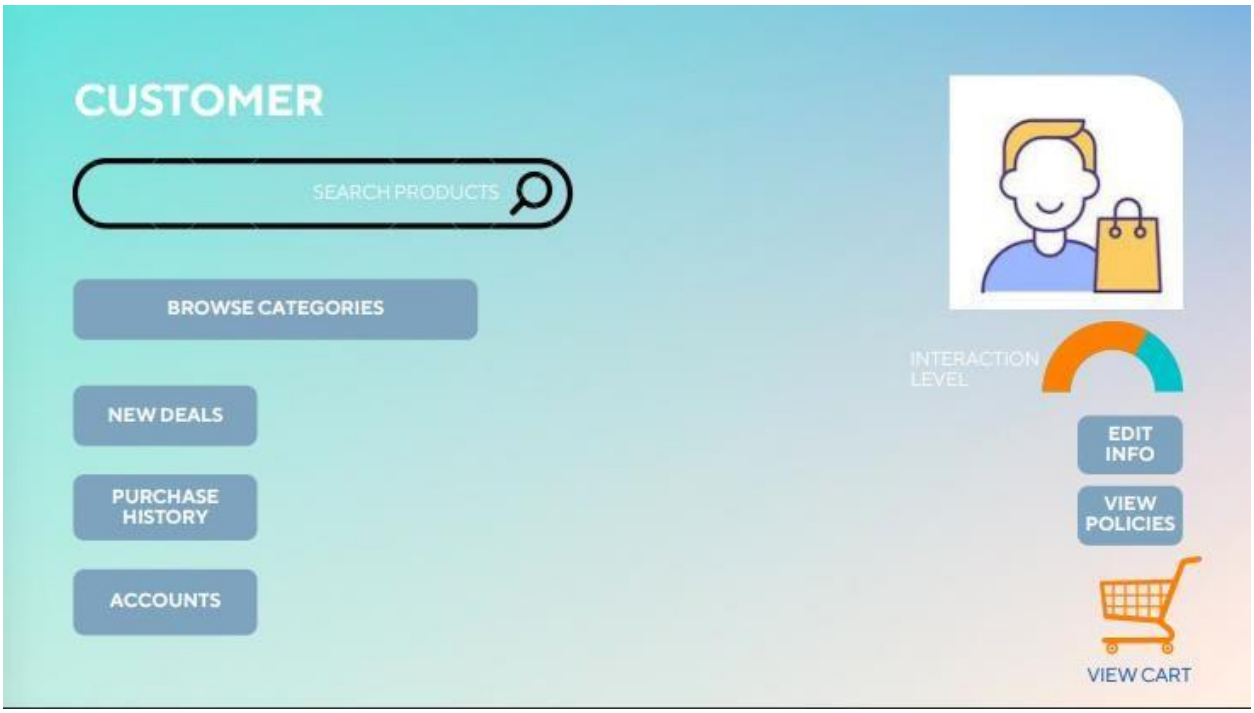

SIGN IN CATEGORY
IMPORTER
ADMIN
EMPLOYEE
SIGN IN
SIGN IN
SIGN IN

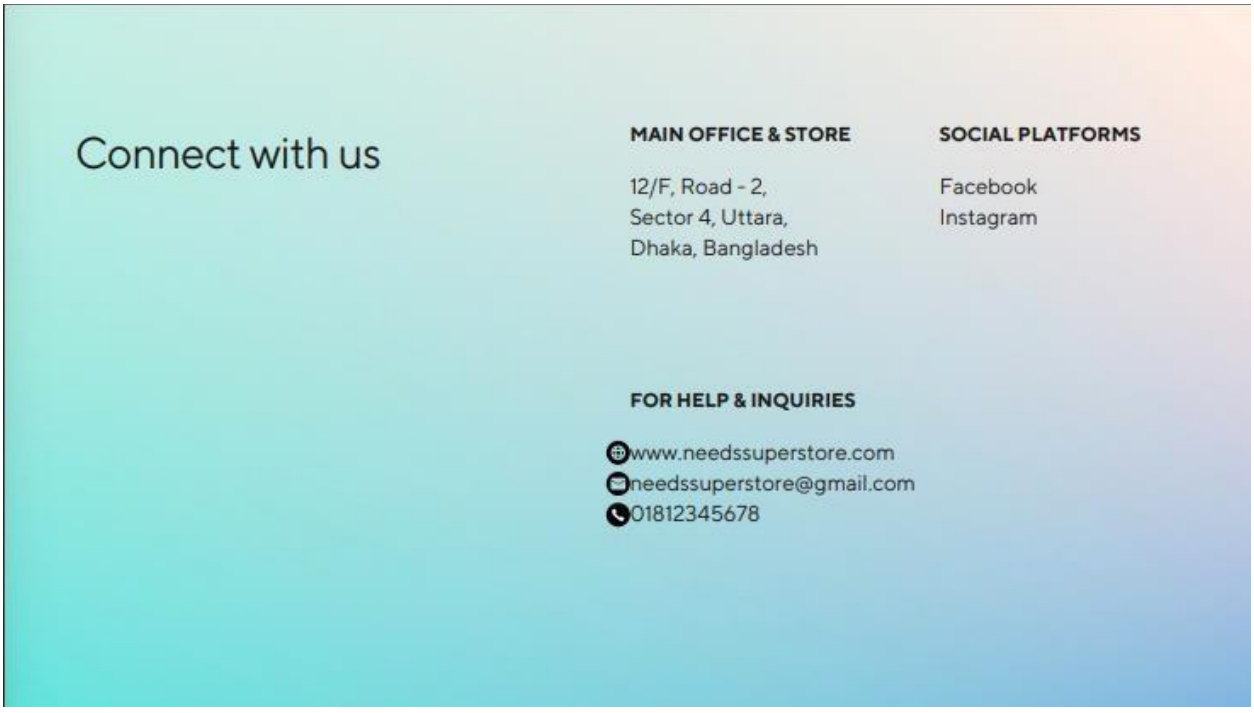

SIGN IN
USER ID
PASSWORD
SIGN IN


IMPORTER
SEARCH ORDERS
INVENTORY
HISTORY
ORDERS
SUPPLY
DATABSE
AGENTS
ACCOUNTS
EDIT INFO


EMPLOYEE
ORDERS
INVENTORY
HISTORY
SCHEDULING
DELIVERY
DATABSE
CUSTOMER CARE
ACCOUNTS
EDIT INFO


ADMIN
CUSTOMERS
INVENTORY
SECURITY
SUPPLIERS
SUPPLY
DATABSE
EMPLOYEES
SALES & PERFORMANCE
EDIT INFO

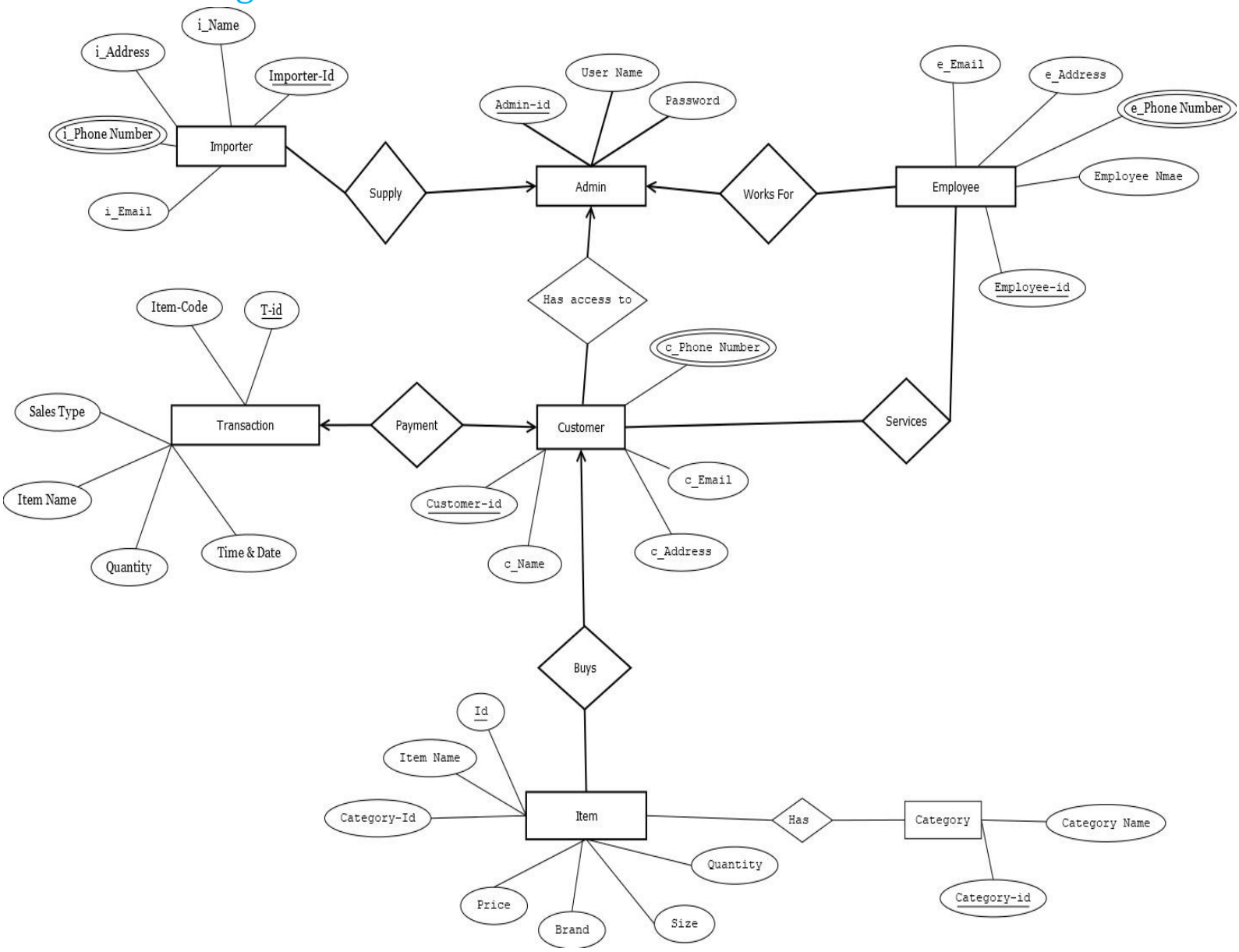

CUSTOMER
SEARCH PRODUCTS
BROWSE CATEGORIES
NEW DEALS
PURCHASE HISTORY
ACCOUNTS
INTERACTION LEVEL
EDIT INFO
VIEW POLICIES
VIEW CART


Connect with us
MAIN OFFICE & STORE
12/F, Road - 2,
Sector 4, Uttara,
Dhaka, Bangladesh
SOCIAL PLATFORMS
Facebook
Instagram
FOR HELP & INQUIRIES
www.needssuperstore.com
needssuperstore@gmail.com
01812345678

## Scenario Description

In a superstore management, a store is conducted by its admin that including (admin id, name and password). A store has many importers. They supply products for the store and they are identified by importer id, i_name,i_address,i_phone number and i_email. A store can get products from many importers and one importer maybe supply to exactly one store. A store has multiple employees. But an employee could be work on exactly one store. An employee is identified by employee name, employee id, e_address, e_phone number and e_email. A store has many customers. One customer can visit one store at a time. Customer is identified by c_name,customer_id,c_phone number, c_email and c_address. A customer can get service from many employees and an employee can give service to many customers. An item is identified by id,item name, category_id, price, brand, size and quantity. An item has many categories and a category is associated with several item. A category can be identified by category name,category-id.A customer has to pay bills for the products. The transaction is identified by t_id, item_code, item_name, quantity, sales type, time & date. A customer can make a transaction at a time and a transaction can be for one customer.

# ER Diagram




## Normalization(Normalize up to 3rd Normal Form): Supply

**UNF :**

Supply ( Admin_id , User Name, Password, Importer_id, i_Name, i_Address,i_Phone Number,i_Email)

**1NF:**

Phone Number is a multivalued attribute

1 . Admin_id , User Name, Password, Importer_id, i_Name, i_Address,i_Phone Number,i_Email

2 **NF :**

1. Admin_id , User Name, Password

2. Importer_id, Name, Address,Phone Number,Email

**3 NF:**

There is no transitive dependency . Relation already in 3NF.

1 . Admin_id , User Name, Password

2. Importer_id, i_Name, i_Address,i_Phone Number,i_Email

**Table creation :**

1 . Admin_id , User Name, Password

2. Importer_id, i_Name, i_Address,i_Phone Number,i_Email, **Admin_id**

**UNF :**

## Works For

Works For ( Admin_id , User Name, Password,Employee_id,Employee Name, e_Address,e_Phone Number,e_Email)

**1NF:**

Phone Number is a multivalued attribute

1 . Admin_id , User Name, Password,Employee_id, Employee Name, e_Address,e_Phone Number,e_Email

2 **NF :**

1. Admin_id , User Name, Password

2. Employee_id, Employee Name, e_Address,e_Phone Number,e_Email

**3 NF:**

There is no transitive dependency . Relation already in 3NF.

1 . Admin_id , User Name, Password

2. Employee_id, Employee Name, e_Address,e_Phone Number,e_Email

**Table creation :**

1 . Admin_id , User Name, Password

2. Employee_id, Employee Name, e_Address,e_Phone Number,e_Email, **Admin_id**

**UNF :**

## Services

Services ( Employee_id,Employee Name, e_Address,e_Phone Number,e_Email,Customer-id , c_Name, c_Address, c_Phone Number,c_Email) **1NF:**

Phone Number is a multivalued attribute

1 . Employee_id,Employee Name, e_Address,e_Phone Number,e_Email,Customer-id , c_Name, c_Address, c_Phone Number,c_Email

2 **NF :**

1. Employee_id, Employee Name, e_Address,e_Phone Number,e_Email
2. Customer-id , c_Name, c_Address, c_Phone Number,c_Email

**3 NF:**

There is no transitive dependency . Relation already in 3NF.

1. Employee_id, Employee Name, e_Address,e_Phone Number,e_Email
2. Customer-id , c_Name, c_Address, c_Phone Number,c_Email

**Table creation :**

1. Employee_id, Employee Name, e_Address,e_Phone Number,e_Email
2. Customer-id , c_Name, c_Address, c_Phone Number,c_Email
3. **Employee_id, Customer-id**

**UNF :**

## Has access to

Holds ( Admin_id , User Name, Password, Customer-id , c_Name, c_Address, c_Phone Number,c_Email

)

**1NF:**

Phone Number is a multivalued attribute

1 . Admin_id , User Name, Password, Customer-id , c_Name, c_Address, c_Phone Number,c_Email

2 **NF :**

1. Admin_id , User Name, Password

2. Customer-id , c_Name, c_Address, c_Phone Number,c_Email

**3 NF:**

There is no transitive dependency . Relation already in 3NF.

1 . Admin_id , User Name, Password

2. Customer-id , c_Name, c_Address, c_Phone Number,c_Email

**Table creation :**

1 . Admin_id , User Name, Password

**UNF :**

2. Customer-id , c_Name, c_Address, c_Phone Number,c_Email, **Admin_id**

## Payment

Payment (Customer-id , c_Name, c_Address, c_Phone Number,c_Email, T-id,Item-Code,Sales Type,Item Name,Quantity,Time & Date)

**1NF:**

Phone Number is a multivalued attribute

1 , Customer-id , c_Name, c_Address, c_Phone Number,c_Email,T-id,Item-Code,Sales Type,Item Name,Quantity,Time & Date)

2 **NF :**

1. Customer-id , c_Name, c_Address, c_Phone Number,c_Email

2. T-id,Item-Code,Sales Type,Item Name,Quantity,Time & Date

**3 NF:**

There is no transitive dependency . Relation already in 3NF.

1. Customer-id , c_Name, c_Address, c_Phone Number,c_Email

2. T-id,Item-Code,Sales Type,Item Name,Quantity,Time & Date

**Table creation :**

1. Customer-id , c_Name, c_Address, c_Phone Number,c_Email

**UNF :**

2. T-id,Item-Code,Sales Type,Item Name,Quantity,Time & Date, **Customer-id**

## Buys

Buys (Customer-id , c_Name, c_Address, c_Phone Number,c_Email, Id, Item Name,CategoryId,Price,Brand,Size,Quantity)

**1NF:**

Phone Number is a multivalued attribute

1 , Customer-id , c_Name, c_Address, c_Phone Number,c_Email, Id, Item Name,CategoryId,Price,Brand,Size,Quantity

2 **NF :**

1. Customer-id , c_Name, c_Address, c_Phone Number,c_Email

2. Id, Item Name,Category-Id,Price,Brand,Size,Quantity

**3 NF:**

There is no transitive dependency . Relation already in 3NF.

1. Customer-id , Name, Address, Phone Number,Email

2. Id, Name,Category-Id,Price,Brand,Size,Quantity

**Table creation :**

1. Customer-id , c_Name, c_Address, c_Phone Number,c_Email

**UNF :**

2. Id, Item Name,Category-Id,Price,Brand,Size,Quantity,**Customer-id**

## Has

Has (Id, Item Name,Category-Id,Price,Brand,Size,Quantity,Category_id,Category Name)

**1NF:**

Phone Number is a multivalued attribute

1 . Id, Item Name,Category-Id,Price,Brand,Size,Quantity,Category_id,Category Name

2 **NF :**

1. Id, Item Name,Category-Id,Price,Brand,Size,Quantity

2. Category_id,Category Name

**3 NF:**

There is no transitive dependency . Relation already in 3NF.

1. Id, Item Name,Category-Id,Price,Brand,Size,Quantity

2. Category_id,Category Name

**Table creation :**

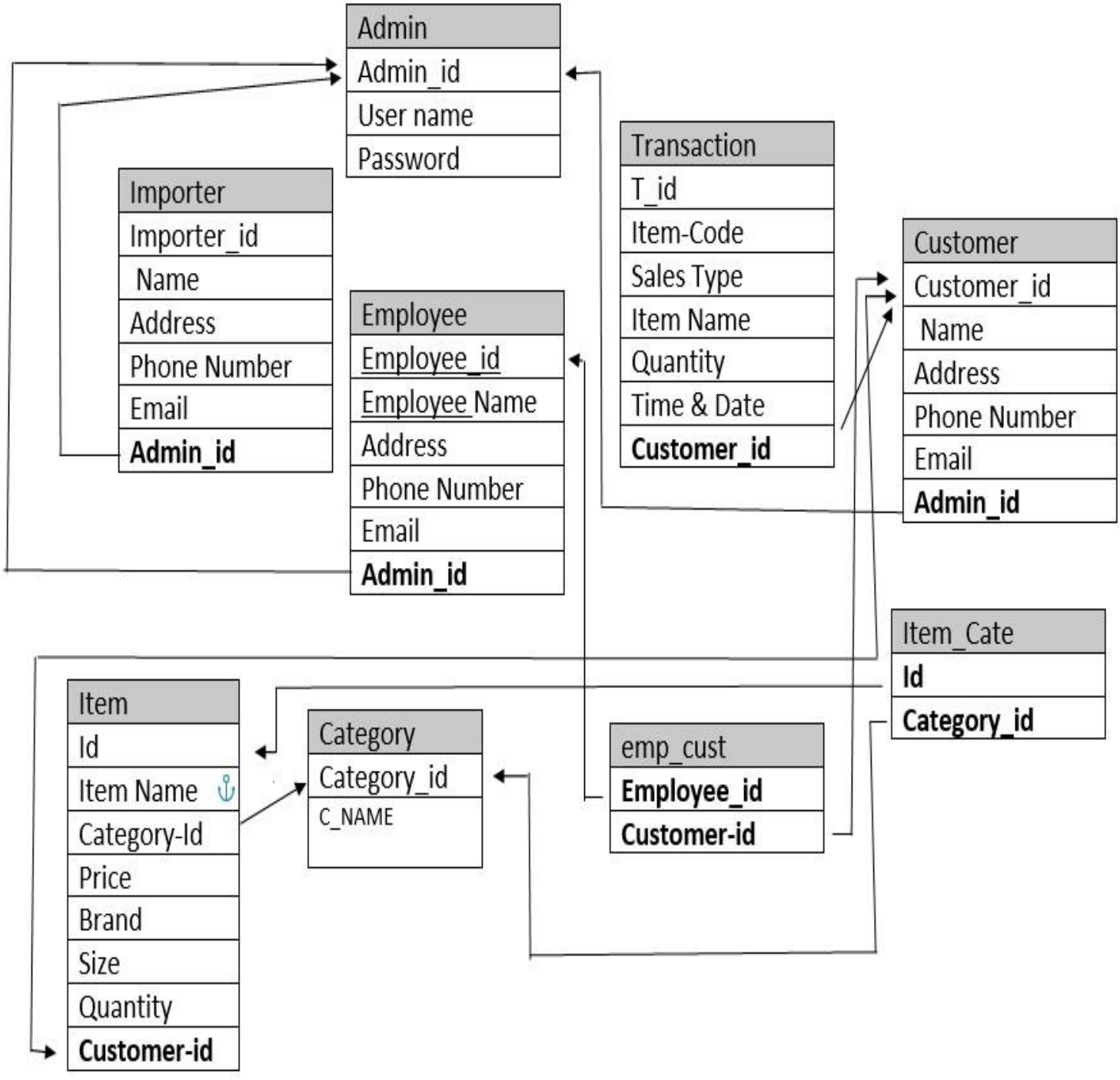
1. Id, Item Name,Category-Id,Price,Brand,Size,Quantity

2. Category_id,Category Name

**UNF :**

3. **Id ,Category_id**

## Schema Diagram:



# Table Creation:

## -After Normalization create tables

## Temporary tables

1. ~~Admin_id , User Name, Password~~
2. ~~Importer_id, i_Name, i_Address,i_Phone Number,i_Email, **Admin_id**~~
3. ~~Admin_id , User Name, Password~~
4. ~~Employee_id,Employee Name, e_Address,e_Phone Number,e_Email, **Admin_id**~~
5. ~~Employee_id,Employee Name,e_Address,e_Phone Number,e_Email,~~
6. ~~Customer-id ,c_Name, c_Address, c_Phone Number,c_Email~~
7. ~~**Employee_id, Customer-id**~~
8. ~~Admin_id , User Name, Password~~
9. ~~Customer-id , c_Name, c_Address, c_Phone Number,c_Email, **Admin_id**~~
10. ~~Customer-id , c_Name,c_Address, c_Phone Number,c_Email~~
11. ~~T-id,Item-Code,Sales Type,Item Name,Quantity,Time & Date, **Customer-id**~~
12. ~~Customer-id , c_Name,c_Address, c_Phone Number,c_Email~~
13. ~~Id, Item Name,Category-Id,Price,Brand,Size,Quantity, **Customer-id**~~
14. ~~Id,Item Name,Category-Id,Price,Brand,Size,Quantity~~
15. ~~Category_id, Category _Name~~
16. ~~**Id ,Category_id**~~

## Final tables

1. Importer_id, i_Name, i_Address,i_Phone Number,i_Email, **Admin_id**
2. Employee_id,Employee Name, e_Address,e_Phone Number,e_Email, **Admin_id**
3. **Employee_id, Customer-id**
4. Admin_id , User Name, Password

5. Customer-id , c_Name, c_Address, c_Phone Number,c_Email, **Admin_id**
6. T-id,Item-Code,Sales Type,Item Name,Quantity,Time & Date, **Customer-id**
7. Id, Item Name,Category-Id,Price,Brand,Size,Quantity, **Customer-id**
8. Category_id, Category_Name
9. **Id ,Category_id**

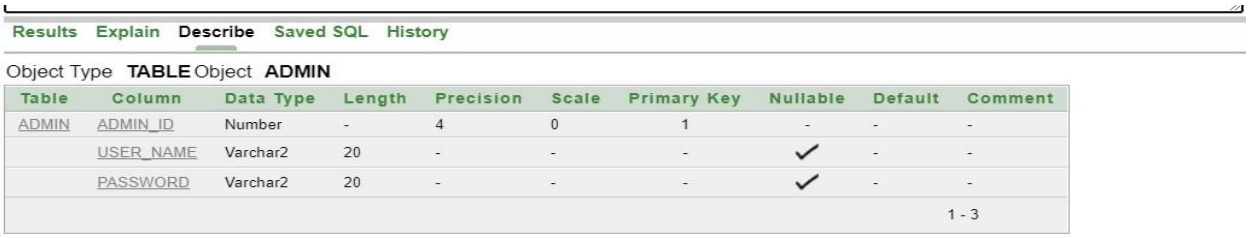
## -Include constraints when creating Tables(Table's name in Dark red color)

# 1. Admin

Create table Admin(

admin_id number(4), user_name varchar2(20),password varchar2(20),CONSTRAINT pk1 PRIMARY KEY (admin_id));

Describe Admin;



Results Explain Describe Saved SQL History

Object Type TABLE Object ADMIN

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| ADMIN | ADMIN_ID | Number | - | 4 | 0 | 1 | - | - | - |
| | USER_NAME | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | PASSWORD | Varchar2 | 20 | - | - | - | ✓ | - | - |

1 - 3

# 2. Importer

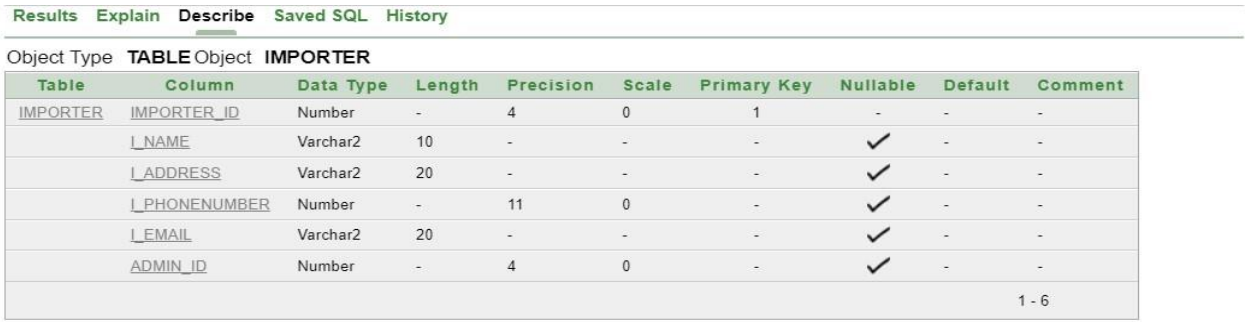
Create table Importer(

Importer_id number(4), i_Name varchar2(10),i_Address varchar2(20),i_PhoneNumber number(11),i_Email varchar2(20),admin_id number(4),

CONSTRAINT pk2 PRIMARY KEY (Importer_id),

CONSTRAINT fK1 FOREIGN KEY (admin_id) REFERENCES Admin (admin_id));

Describe Importer;



Results Explain Describe Saved SQL History

Object Type TABLE Object IMPORTER

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| IMPORTER | IMPORTER_ID | Number | - | 4 | 0 | 1 | - | - | - |
| | I_NAME | Varchar2 | 10 | - | - | - | ✓ | - | - |
| | I_ADDRESS | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | I_PHONENUMBER | Number | - | 11 | 0 | - | ✓ | - | - |
| | I_EMAIL | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | ADMIN_ID | Number | - | 4 | 0 | - | ✓ | - | - |

1 - 6

# 3. Employee

Create table Employee(

Employee_id number(4), Employee_Name varchar2(10),e_Address varchar2(20),

e_PhoneNumber number(11),e_Email varchar2(20),admin_id number(4),

CONSTRAINT pk3 PRIMARY KEY (Employee_id),

CONSTRAINT fK2 FOREIGN KEY (admin_id) REFERENCES Admin (admin_id));

Describe Employee;

Results Explain Describe Saved SQL History

Object Type TABLE Object EMPLOYEE

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| EMPLOYEE | EMPLOYEE_ID | Number | - | 4 | 0 | 1 | - | - | - |
| | EMPLOYEE_NAME | Varchar2 | 10 | - | - | - | ✓ | - | - |
| | E_ADDRESS | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | E_PHONENUMBER | Number | - | 11 | 0 | - | ✓ | - | - |
| | E_EMAIL | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | ADMIN_ID | Number | - | 4 | 0 | - | ✓ | - | - |

1 - 6

# 4. Customer

Create table Customer(

Customer_id number (4), c_Name varchar2 (10), c_Address varchar2 (20), c_PhoneNumber number (11),c_Email varchar2(20), admin_id number (4), CONSTRAINT pk4 PRIMARY KEY (Customer_id),

CONSTRAINT FK3 FOREIGN KEY (admin_id) REFERENCES Admin (admin_id));
Describe Customer;

Results Explain Describe Saved SQL History

Object Type TABLE Object CUSTOMER

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| CUSTOMER | CUSTOMER_ID | Number | - | 4 | 0 | 1 | - | - | - |
| | C_NAME | Varchar2 | 10 | - | - | - | ✓ | - | - |
| | C_ADDRESS | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | C_PHONENUMBER | Number | - | 11 | 0 | - | ✓ | - | - |
| | C_EMAIL | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | ADMIN_ID | Number | - | 4 | 0 | - | ✓ | - | - |

1 - 6

# 5. Transaction

Create table Transaction(

T_id number(4), Sales_Type varchar2(10),Item_name varchar2(20),Quantity number(10),Time_Date Date,Customer_id number(4),

CONSTRAINT pk5 PRIMARY KEY (T_id),

CONSTRAINT fK4 FOREIGN KEY (Customer_id) REFERENCES Customer(Customer_id)

);

Describe Transaction;

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **TRANSACTION**

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| TRANSACTION | T_ID | Number | - | 4 | 0 | 1 | - | - | - |
| | SALES_TYPE | Varchar2 | 10 | - | - | - | ✓ | - | - |
| | ITEM_NAME | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | QUANTITY | Number | - | 10 | 0 | - | ✓ | - | - |
| | TIME_DATE | Date | 7 | - | - | - | ✓ | - | - |
| | CUSTOMER_ID | Number | - | 4 | 0 | - | ✓ | - | - |

1 - 6

# 6. Item

Create table Item(

Id number(4),Item_name varchar2(20),Category_id number(4),Price number(10),Brand varchar2(20),Quantity number(7),"Size" number(5),Customer_id number(4),

CONSTRAINT pk6 PRIMARY KEY (Id),

CONSTRAINT fK5 FOREIGN KEY (Customer_id) REFERENCES Customer (Customer_id));

Describe Item;

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **ITEM**

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| ITEM | ID | Number | - | 4 | 0 | 1 | - | - | - |
| | ITEM_NAME | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | CATEGORY_ID | Number | - | 4 | 0 | - | ✓ | - | - |
| | PRICE | Number | - | 10 | 0 | - | ✓ | - | - |
| | BRAND | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | QUANTITY | Number | - | 7 | 0 | - | ✓ | - | - |
| | Size | Number | - | 5 | 0 | - | ✓ | - | - |
| | CUSTOMER_ID | Number | - | 4 | 0 | - | ✓ | - | - |

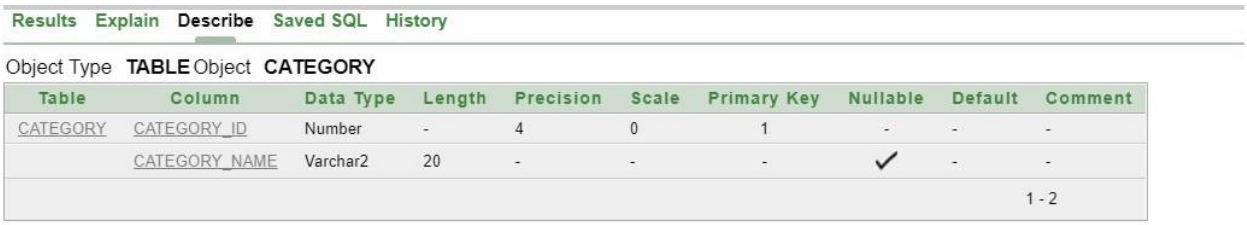
1 - 8

# 7. Category

Create table Category

(Category_id number (4), category_name varchar2(20), CONSTRAINT pk7 PRIMARY KEY (Category_id) );

Describe Category;



Results Explain Describe Saved SQL History

Object Type TABLE Object CATEGORY

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| CATEGORY | CATEGORY_ID | Number | - | 4 | 0 | 1 | - | - | - |
| | CATEGORY_NAME | Varchar2 | 20 | - | - | - | ✓ | - | - |

1 - 2

# 8. Item_Cate

Create table Item_Cate

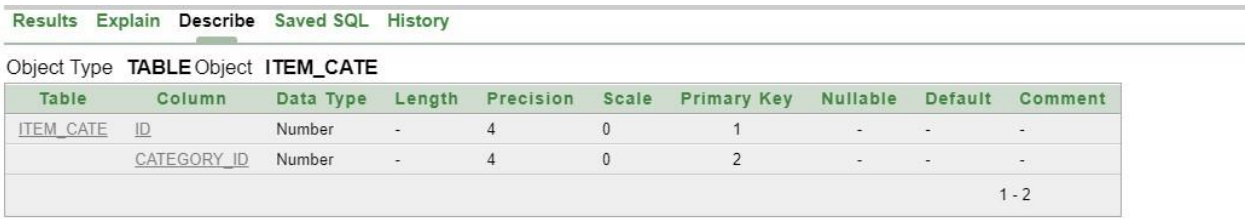
(

Id number(4), Category_id number(4),

CONSTRAINT pk8 PRIMARY KEY (Id,Category_id )

);

Describe Item_Cate;



Results Explain Describe Saved SQL History

Object Type TABLE Object ITEM_CATE

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| ITEM_CATE | ID | Number | - | 4 | 0 | 1 | - | - | - |
| | CATEGORY_ID | Number | - | 4 | 0 | 2 | - | - | - |

1 - 2

# 9. emp_cust

Create table emp_cust

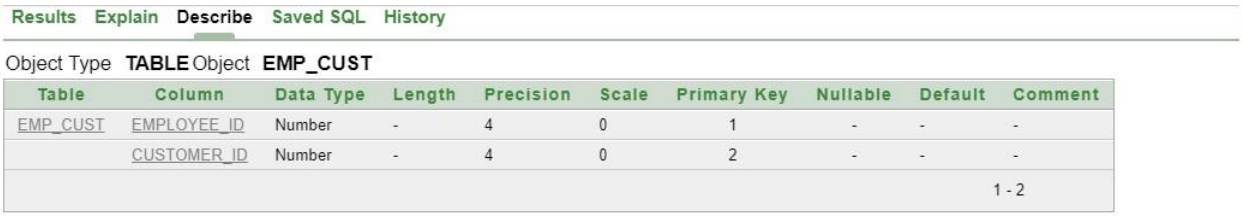
(

Employee_id number(4), Customer_id number(4),

CONSTRAINT pk9 PRIMARY KEY (Employee_id,Customer_id)

);

Describe emp_cust;



Results Explain Describe Saved SQL History

Object Type TABLE Object EMP_CUST

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---|---|---|---|---|---|---|---|---|---|
| EMP_CUST | EMPLOYEE_ID | Number | - | 4 | 0 | 1 | - | - | - |
| | CUSTOMER_ID | Number | - | 4 | 0 | 2 | - | - | - |

1 - 2

## -Include the queries required to create sequence

1.CREATE SEQUENCE Admin_adminid

INCREMENT BY 1

START WITH 111;

2.CREATE SEQUENCE Importer_Importerid

INCREMENT BY 1

START WITH 222;

3.CREATE SEQUENCE Employee_empid

INCREMENT BY 1

START WITH 2234;

4.CREATE SEQUENCE Customer_Customerid

INCREMENT BY 1

START WITH 333;

5.CREATE SEQUENCE Transaction_Tid

INCREMENT BY 1

START WITH 444;

6.CREATE SEQUENCE Item_Id

INCREMENT BY 1

START WITH 6666;

7.CREATE SEQUENCE Category_Categoryid

INCREMENT BY 1

START WITH 777;

8. CREATE SEQUENCE Item_CateId

INCREMENT BY 1

START WITH 6686;

9.CREATE SEQUENCE emp_custid

INCREMENT BY 1

START WITH 2234;

## Data Insertion:

# 1.Admin:

INSERT INTO Admin (admin_id, user_name, password) VALUES (Admin_adminid.NEXTVAL, 'a', 'abcd1234');

INSERT INTO Admin (admin_id, user_name, password) VALUES (Admin_adminid.NEXTVAL, 'b','adcd1254');

INSERT INTO Admin (admin_id, user_name, password) VALUES (Admin_adminid.NEXTVAL, 'c', 'xbzd12649');

INSERT INTO Admin (admin_id, user_name, password) VALUES (Admin_adminid.NEXTVAL, 'd', 'abc412rh4');

INSERT INTO Admin (admin_id, user_name, password) VALUES (Admin_adminid.NEXTVAL, 'e', 'aegcdlyj34');

Select * from Admin;


Results Explain Describe Saved SQL History

| ADMIN_ID | USER_NAME | PASSWORD |
|---|---|---|
| 111 | a | abcd1234 |
| 112 | b | adcd1254 |
| 113 | c | xbzd12649 |
| 114 | d | abc412rh4 |
| 115 | e | aegcd1yj34 |

5 rows returned in 0.00 seconds CSV Export

# 2.Importer:

INSERT INTO Importer(Importer_id, i_name, i_address,i_PhoneNumber,i_Email,admin_id) VALUES (Importer_Importerid.NEXTVAL, 'x', 'Beijing-China','+8616521689','x1@gmail.com','111');

INSERT INTO Importer(Importer_id, i_name, i_address,i_PhoneNumber,i_Email,admin_id) VALUES (Importer_Importerid.NEXTVAL, 'x2', 'Chittagong-BD','01712345678','x2@gmail.com','112');

INSERT INTO Importer(Importer_id, i_name, i_address,i_PhoneNumber,i_Email,admin_id) VALUES (Importer_Importerid.NEXTVAL, 'x3', 'TokyoJapan','+8114541689','x3@gmail.com','113');

INSERT INTO Importer(Importer_id, i_name, i_address,i_PhoneNumber,i_Email,admin_id) VALUES (Importer_Importerid.NEXTVAL, 'x4', 'Rajshahi-Bangladesh','01812345679','x4@gmail.com','114');

INSERT INTO Importer(Importer_id, i_name, i_address,i_PhoneNumber,i_Email,admin_id) VALUES (Importer_Importerid.NEXTVAL, 'x5', 'Macau-China','+8608521657','x5@gmail.com','115');

Select * from Importer;

Results Explain Describe Saved SQL History

| IMPORTER_ID | I_NAME | I_ADDRESS | I_PHONENUMBER | I_EMAIL | ADMIN_ID |
|---|---|---|---|---|---|
| 223 | x | Beijing-China | 8616521689 | x1@gmail.com | 111 |
| 225 | x2 | Chittagong-BD | 1712345678 | x2@gmail.com | 112 |
| 226 | x3 | Tokyo-Japan | 8114541689 | x3@gmail.com | 113 |
| 227 | x4 | Rajshahi-Bangladesh | 1812345679 | x4@gmail.com | 114 |
| 228 | x5 | Macau-China | 8608521657 | x5@gmail.com | 115 |

5 rows returned in 0.00 seconds CSV Export

# 3.Employee:

INSERT INTO

Employee(Employee_id,Employee_Name,e_Address,e_PhoneNumber,e_Email,admin_id) VALUES

(Employee_empid.NEXTVAL, 'd1' ,'Dhaka-BD','01988256435','d1@gmail.com','111')

INSERT INTO Employee(Employee_id,Employee_Name,e_Address,e_PhoneNumber,e_Email,admin_id) VALUES

(Employee_empid.NEXTVAL, 'd2','Kushtia-BD','01988253576','d2@gmail.com','112')

INSERT INTO Employee(Employee_id,Employee_Name,e_Address,e_PhoneNumber,e_Email,admin_id) VALUES

(Employee_empid.NEXTVAL, 'd3' ,'Pabna-BD','01988256777','d3@gmail.com','113')

INSERT INTO

Employee(Employee_id,Employee_Name,e_Address,e_PhoneNumber,e_Email,admin_id)
VALUES

(Employee_empid.NEXTVAL, 'd4' ,'Cumilla-BD','01988256755','d4@gmail.com','114')

INSERT INTO
Employee(Employee_id,Employee_Name,e_Address,e_PhoneNumber,e_Email,admin_id)
VALUES

(Employee_empid.NEXTVAL, 'd5' ,'Chittagong-BD','01988356743','d5@gmail.com','115')

Select * from Employee;

Results Explain Describe Saved SQL History

| EMPLOYEE_ID | EMPLOYEE_NAME | E_ADDRESS | E_PHONENUMBER | E_EMAIL | ADMIN_ID |
|---|---|---|---|---|---|
| 2234 | d1 | Dhaka-BD | 1988256435 | d1@gmail.com | 111 |
| 2235 | d2 | Kushtia-BD | 1988253576 | d2@gmail.com | 112 |
| 2236 | d3 | Pabna-BD | 1988256777 | d3@gmail.com | 113 |
| 2237 | d4 | Cumilla-BD | 1988256755 | d4@gmail.com | 114 |
| 2238 | d5 | Chittagong-BD | 1988356743 | d5@gmail.com | 115 |

5 rows returned in 0.00 seconds CSV Export

# 4.Customer:

INSERT INTO Customer(Customer_id, c_name, c_address,c_PhoneNumber,c_Email,admin_id)
VALUES (Customer_Customerid.NEXTVAL, 'c1', 'Dhaka-
Bd','01758792437','c1@gmail.com','111');

INSERT INTO Customer(Customer_id, c_name, c_address,c_PhoneNumber,c_Email,admin_id)
VALUES (Customer_Customerid.NEXTVAL, 'c2', 'Narayanganj-
Bd','01858729223','c2@gmail.com','112');

INSERT INTO Customer(Customer_id, c_name, c_address,c_PhoneNumber,c_Email,admin_id)
VALUES (Customer_Customerid.NEXTVAL, 'c3', 'Cumilla-
Bd','01958792444','c3@gmail.com','113');

INSERT INTO Customer(Customer_id, c_name, c_address,c_PhoneNumber,c_Email,admin_id)
VALUES (Customer_Customerid.NEXTVAL, 'c4', 'Dhaka-
Bd','01358792476','c4@gmail.com','114');

INSERT INTO Customer(Customer_id, c_name, c_address,c_PhoneNumber,c_Email,admin_id)
VALUES (Customer_Customerid.NEXTVAL, 'c5', 'Chittagong-
Bd','01558792489','c5@gmail.com','115');

Select * from Customer;

Results Explain Describe Saved SQL History

| CUSTOMER_ID | C_NAME | C_ADDRESS | C_PHONENUMBER | C_EMAIL | ADMIN_ID |
|---|---|---|---|---|---|
| 333 | c1 | Dhaka-Bd | 1758792437 | c1@gmail.com | 111 |
| 334 | c2 | Narayanganj-Bd | 1858729223 | c2@gmail.com | 112 |
| 335 | c3 | Cumilla-Bd | 1958792444 | c3@gmail.com | 113 |
| 336 | c4 | Dhaka-Bd | 1358792476 | c4@gmail.com | 114 |
| 337 | c5 | Chittagong-Bd | 1558792489 | c5@gmail.com | 115 |

5 rows returned in 0.00 seconds CSV Export

# 5.Transaction:

INSERT INTO Transaction(T_id, Sales_Type, Item_name,Quantity,Time_Date,Customer_id) VALUES (Transaction_Tid.NEXTVAL, 'Online', 'T_Shirt','3','19-oct-2022','333');

INSERT INTO Transaction(T_id, Sales_Type, Item_name,Quantity,Time_Date,Customer_id) VALUES (Transaction_Tid.NEXTVAL, 'Inside ', 'Coca-Cola', '4','20-oct-2022', '334') ;

INSERT INTO Transaction(T_id, Sales_Type, Item_name,Quantity,Time_Date,Customer_id) VALUES (Transaction_Tid.NEXTVAL, 'Inside ', 'Roast Chicken','2','21-oct-2022','335');

INSERT INTO Transaction(T_id, Sales_Type, Item_name,Quantity,Time_Date,Customer_id) VALUES (Transaction_Tid.NEXTVAL, 'Online', 'Water Melon','3','22-oct-2022','336');

INSERT INTO Transaction(T_id, Sales_Type, Item_name,Quantity,Time_Date,Customer_id) VALUES (Transaction_Tid.NEXTVAL, 'Inside ', 'Pen Set','8', '23-oct-2022','337'); Select

* from Transaction;

Results Explain Describe Saved SQL History

| T_ID | SALES_TYPE | ITEM_NAME | QUANTITY | TIME_DATE | CUSTOMER_ID |
|---|---|---|---|---|---|
| 469 | Online | T_Shirt | 3 | 19-OCT-22 | 333 |
| 470 | Inside | Coca-Cola | 4 | 20-OCT-22 | 334 |
| 471 | Inside | Roast Chicken | 2 | 21-OCT-22 | 335 |
| 472 | Online | Water Melon | 3 | 22-OCT-22 | 336 |
| 473 | Inside | Pen Set | 8 | 23-OCT-22 | 337 |

# 6.Item:

INSERT INTO Item(Id ,Item_name,Category_id,Price,Brand,Quantity,"Size",Customer_id ) VALUES(Item_Id.NEXTVAL, 'Pepsi' ,'777','20','Pepsico','1','1','333');

INSERT INTO Item(Id ,Item_name,Category_id,Price,Brand,Quantity,"Size",Customer_id )

VALUES(Item_Id.NEXTVAL,'Apple','778','195','None','1','1','334');

INSERT INTO Item(Id ,Item_name,Category_id,Price,Brand,Quantity,"Size",Customer_id )
VALUES(Item_Id.NEXTVAL, 'Egg' ,'779','9','Kazi','1','1','335');

INSERT INTO Item(Id ,Item_name,Category_id,Price,Brand,Quantity,"Size",Customer_id )
VALUES(Item_Id.NEXTVAL, 'Rice' ,'780','75','Pran','1','1','336');

INSERT INTO Item(Id ,Item_name,Category_id,Price,Brand,Quantity,"Size",Customer_id )

VALUES(Item_Id.NEXTVAL, 'Orange' ,'781','238','NONE','1','1','337'); Select * from Item;

Results Explain Describe Saved SQL History

| ID | ITEM_NAME | CATEGORY_ID | PRICE | BRAND | QUANTITY | Size | CUSTOMER_ID |
|---|---|---|---|---|---|---|---|
| 6686 | Pepsi | 777 | 20 | Pepsico | 1 | 1 | 333 |
| 6687 | Apple | 778 | 195 | None | 1 | 1 | 334 |
| 6688 | Egg | 779 | 9 | Kazi | 1 | 1 | 335 |
| 6689 | Rice | 780 | 75 | Pran | 1 | 1 | 336 |
| 6690 | Orange | 781 | 238 | NONE | 1 | 1 | 337 |

5 rows returned in 0.00 seconds CSV Export

# 7.Category:

INSERT INTO Category(Category_id, Category_name) VALUES
(Category_Categoryid.NEXTVAL, 'cate1');

INSERT INTO Category(Category_id, Category_name) VALUES
(Category_Categoryid.NEXTVAL, 'cate2');

INSERT INTO Category(Category_id, Category_name) VALUES
(Category_Categoryid.NEXTVAL, 'cate3');

INSERT INTO Category(Category_id, Category_name) VALUES
(Category_Categoryid.NEXTVAL, 'cate4');

INSERT INTO Category(Category_id, Category_name) VALUES
(Category_Categoryid.NEXTVAL, 'cate5');

Select * from Category;

Results Explain Describe Saved SQL History

| CATEGORY_ID | CATEGORY_NAME |
|---|---|
| 777 | cate1 |
| 778 | cate2 |
| 779 | cate3 |
| 780 | cate4 |
| 781 | cate5 |

5 rows returned in 0.00 seconds CSV Export

# 8.Item_Cate:

INSERT INTO Item_Cate(Id,Category_id) VALUES(Item_CateId.NEXTVAL,'777'); INSERT INTO Item_Cate(Id,Category_id) VALUES(Item_CateId.NEXTVAL,'778');

INSERT INTO Item_Cate(Id,Category_id) VALUES(Item_CateId.NEXTVAL,'779');

INSERT INTO Item_Cate(Id,Category_id) VALUES(Item_CateId.NEXTVAL,'780'); INSERT INTO Item_Cate(Id,Category_id) VALUES(Item_CateId.NEXTVAL,'781');

Select* From Item_Cate;

Results Explain Describe Saved SQL History

| ID | CATEGORY_ID |
|---|---|
| 6686 | 777 |
| 6687 | 778 |
| 6688 | 779 |
| 6689 | 780 |
| 6690 | 781 |

5 rows returned in 0.00 seconds CSV Export

# 9.emp_cust:

INSERT INTO emp_cust(Employee_Id,Customer_id) VALUES(emp_custid.NEXTVAL,'333')

INSERT INTO emp_cust(Employee_Id,Customer_id) VALUES(emp_custid.NEXTVAL,'334')

INSERT INTO emp_cust(Employee_Id,Customer_id) VALUES(emp_custid.NEXTVAL,'335')

INSERT INTO emp_cust(Employee_Id,Customer_id) VALUES(emp_custid.NEXTVAL,'336')

INSERT INTO emp_cust(Employee_Id,Customer_id) VALUES(emp_custid.NEXTVAL,'337')

Select * from emp_cust;

Results Explain Describe Saved SQL History

| EMPLOYEE_ID | CUSTOMER_ID |
|---|---|
| 2234 | 333 |
| 2235 | 334 |
| 2236 | 335 |
| 2237 | 336 |
| 2238 | 337 |

5 rows returned in 0.00 seconds CSV Export

# Query Writing:

## -3 single row function

Q1:Use TO_CHAR function with dates to show item enlist date

Ans: SELECT Item_name,

TO_CHAR(Time_Date, 'fmDD Month YYYY') Time_Date

FROM  Transaction;

| ITEM_NAME | TIME_DATE |
|---|---|
| T_Shirt | 19 October 2022 |
| Coca-Cola | 20 October 2022 |
| Roast Chicken | 21 October 2022 |
| Water Melon | 22 October 2022 |
| Pen Set | 23 October 2022 |

Q2:Use CONCAT function to concatenate two string values.

Ans: SELECT CONCAT (Item_name, Brand)

FROM Item

WHERE rownum < 6;

| CONCAT(ITEM_NAME,BRAND) |
|---|
| PepsiPepsico |
| AppleNone |
| EggKazi |
| RicePran |
| OrangeNONE |

Q3: Calculate how many days have passed since one product purchase using the date arithmetic function.

Ans: SELECT T_id, (sysdate - Time_Date) Passed_days

FROM Transaction

WHERE rownum < 5;

| T_ID | PASSED_DAYS |
|---|---|
| 469 | 20.009201388888888888888888888888888889 |
| 470 | 19.009201388888888888888888888888888889 |
| 471 | 18.009201388888888888888888888888888889 |
| 472 | 17.009201388888888888888888888888888889 |

## -3 group function

Q1: Display the Item_name which is more costly than egg and group by its name

Ans: SELECT Item_name, max(Price)

FROM Item

GROUP BY Item_name

HAVING max(Price)>9;

| ITEM_NAME | MAX(PRICE) |
|---|---|
| Apple | 195 |
| Orange | 238 |
| Pepsi | 20 |
| Rice | 75 |

Q2:Display the Item_name which is more costlier than egg and group by its name

Ans: SELECT  Item_name, min(Price)

FROM Item

GROUP BY Item_name

HAVING  min(Price)<238;

| ITEM_NAME | MIN(PRICE) |
|---|---|
| Apple | 195 |
| Egg | 9 |
| Pepsi | 20 |
| Rice | 75 |

Q3:Display the oldest and latest date of transaction

Ans: SELECT MIN (Time_Date) oldest, MAX (Time_Date) latest

FROM Transaction;

| OLDEST | LATEST |
|---|---|
| 19-OCT-22 | 23-OCT-22 |

## -3 subquery

Q1:Display the Item_name which is more less costlier than orange.

Ans:

Select * from Item;

select Item_name from Item where Price> (select Price from Item where Item_name ='Apple')

Results Explain Describe Saved SQL History

| ITEM_NAME |
|---|
| Orange |

1 rows returned in 0.00 seconds CSV Export

Q2: Display the c_name who use email address "c5@gmail.com"

Ans:

Select * from Customer;

select c_name from Customer where c_Email = (select c_Email from Customer where c_Email ='c5@gmail.com');


Q3: Display which items are bought earlier than pen set

Ans: SELECT Item_name,T_Id

FROM Transaction

WHERE Time_Date< (SELECT Time_Date

FROM Transaction

WHERE T_id=473);

| ITEM_NAME | T_ID |
|---|---|
| T_Shirt | 469 |
| Coca-Cola | 470 |
| Roast Chicken | 471 |
| Water Melon | 472 |

## -3 joining

Q1: Display the name of the employee who give services to customer id 334.

Ans: SELECT Employee.Employee_Name from Employee ,emp_cust

where Employee.Employee_id=emp_cust.Employee_id and emp_cust.Customer_id='334';


Q2: Display the name of the Item which one belongs to category id "780"

Ans:

SELECT Item.Item_name

from Item,Item_Cate

where Item.Category_id=Item_Cate.Category_id and Item_Cate.Category_id='780';


Q3: Display the name of the Item which bought by the particular customer

Ans: SELECT Customer.c_name from Customer,Transaction where Customer.Customer_id=Transaction .Customer_id and Transaction .Customer_id =336;


## -3 View

**Q1:** Create a view called CategoryView based on the Category_id from the Category table.

**Ans:**

```
CREATE VIEW CategoryView
 AS SELECT Category_id
 FROM Category
 WHERE Category_name = 'cate5';
```

Select * from CategoryView;

Results Explain Describe Saved SQL History

| CATEGORY_ID |
| --- |
| 781 |

1 rows returned in 0.00 seconds CSV Export

**Q2:** Create a view called ItemView based on the Item_name and Price from the Item table.

**Ans:**

CREATE VIEW ItemView

AS SELECT Item_name,Price

FROM Item

WHERE Id = 6668;

Select * from ItemView;

Results Explain Describe Saved SQL History

| ITEM_NAME | PRICE |
| --- | --- |
| Egg | 9 |

1 rows returned in 0.00 seconds CSV Export

**Q3:** Create a view called ItemInfo that contains item name,price and brand

**Ans:** CREATE VIEW ItemInfo

AS SELECT Item_name , Price,Brand

FROM Item where Id=6669;

Results Explain Describe Saved SQL History

| ITEM_NAME | PRICE | BRAND |
| --- | --- | --- |
| Rice | 75 | Pran |

1 rows returned in 0.00 seconds CSV Export

## -3 synonym

**Q1:** Create a synonym for view called ItemInfo

**Ans:** CREATE SYNONYM ItemI

For ItemInfo;

```
Select * from ItemI;
```

Results Explain Describe Saved S

| ITEM_NAME | PRICE | BRAND |
|---|---|---|
| Rice | 75 | Pran |

1 rows returned in 0.00 seconds

**Q2:** Create a synonym for view called ItemView

**Ans:** CREATE SYNONYM ItemV

For ItemView;

```
Select * from ItemV;
```

Results Explain Describe Sa

| ITEM_NAME | PRICE |
|---|---|
| Egg | 9 |

1 rows returned in 0.00 seconds

**Q3:** Create a synonym for view called CategoryView

**Ans:** CREATE SYNONYM CateV

For CategoryView;

```
Select * from CateV;
```

Results Explain Describe Save

| CATEGORY_ID |
|---|
| 781 |

1 rows returned in 0.03 seconds

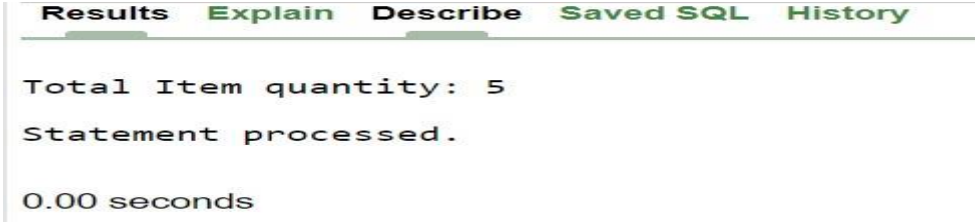
# PL/SQL

## -3 Function

**Q1:** Defining and Invoking a simple PL/SQL function which will compute and return the total number of Item quantity.

**Ans:**

```
CREATE OR REPLACE FUNCTION TotalItem
RETURN number IS
total number(12) := 0;
BEGIN
  SELECT count(*) into total
  FROM Item;
  RETURN total;
END;
/
DECLARE     c
number(12);
BEGIN
  c := TotalItem();
  dbms_output.put_line('Total Item quantity: ' || c);   END;
```




**Q2:** create a function that takes the name as input and returns the welcome message as output. We are going to use anonymous block and select statement to call the function.

**Ans:**

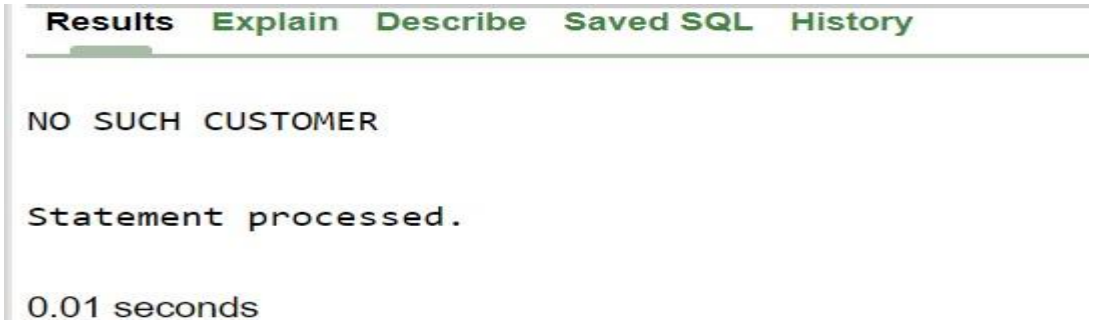
```
CREATE OR REPLACE FUNCTION welcome_msg ( p_name IN VARCHAR2) RETURN VARCHAR2
IS
BEGIN
RETURN ('Welcome '|| p_name);
END;
/
DECLARE
lv_msg VARCHAR2(250);  BEGIN
lv_msg := welcome_msg ('MAhir');  dbms_output.put_line(lv_msg);
END;
```


**Q3:** Create a PL/SQL function get_customer_id that returns the customer's Id given Customer name and also show exception message when data did not found

**Ans:**

```
REATE OR REPLACE FUNCTION get_customer_id(cust_id IN VARCHAR)
RETURN NUMBER
IS
cid VARCHAR2 (50);
BEGIN
   SELECT CUSTOMER_ID INTO cid FROM Customer WHERE C_NAME= 'C8';
RETURN cid;
   EXCEPTION
   WHEN no_data_found THEN
   DBMS_OUTPUT.PUT_LINE('NO SUCH CUSTOMER');
RETURN cid;
END;
/
DECLARE  cid
VARCHAR2 (50);
BEGIN
   cid := get_customer_id (1010);
   DBMS_OUTPUT.PUT_LINE (cid);
```

*END;*




## -3 Procedure

**Q1:** Adjust price of rice using procedure

**Ans:**

CREATE OR REPLACE PROCEDURE adjust_price( in_Price
IN Item.Price%TYPE
)
IS
BEGIN
UPDATE Item
SET Price='80'
WHERE Price= in_Price;
END;

begin
adjust_price('75');
end select * from
Item; rollback

Results Explain Describe Saved SQL History

| ID | ITEM_NAME | CATEGORY_ID | PRICE | BRAND | QUANTITY | Size | CUSTOMER_ID |
|---|---|---|---|---|---|---|---|
| 6666 | Pepsi | 777 | 20 | Pepsico | 1 | 1 | 333 |
| 6667 | Apple | 778 | 195 | None | 1 | 1 | 334 |
| 6668 | Egg | 779 | 9 | Kazi | 1 | 1 | 335 |
| 6669 | Rice | 780 | 80 | Pran | 1 | 1 | 336 |
| 6670 | Orange | 781 | 238 | NONE | 1 | 1 | 337 |

**Q2:** Increase the quantity of selling T-shirts of transaction table using procedure

**Ans:**

```
CREATE OR REPLACE PROCEDURE adjust_Quantity( in_Quantity
IN Transaction.Quantity%TYPE
)
IS
BEGIN
UPDATE Transaction
SET Quantity='5'
WHERE Quantity= in_Quantity ;
END;

begin
adjust_Quantity('3'); end
select * from Transaction;
rollback
```

Results Explain Describe Saved SQL History

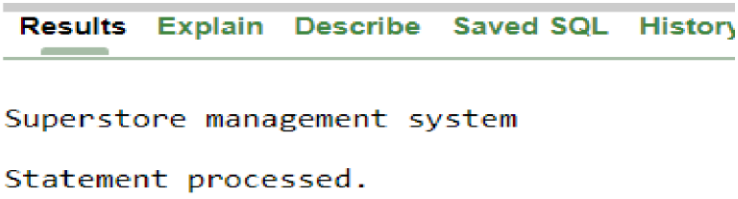
| T_ID | SALES_TYPE | ITEM_NAME | QUANTITY | TIME_DATE | CUSTOMER_ID |
|---|---|---|---|---|---|
| 444 | Online | T_Shirt | 5 | 19-OCT-22 | 333 |
| 445 | Inside | Coca-Cola | 4 | 20-OCT-22 | 334 |
| 446 | Inside | Roast Chicken | 2 | 21-OCT-22 | 335 |
| 447 | Online | Water Melon | 5 | 22-OCT-22 | 336 |
| 448 | Inside | Pen Set | 8 | 23-OCT-22 | 337 |

**Q3:** Display your project title(…) using procedure

**Ans:**

```
CREATE OR REPLACE PROCEDURE project_title
AS
BEGIN
  dbms_output.put_line('Superstore management system');
END;
/
```

```
BEGIN
  project_title;
END;
/
```




## -3 Record

**Q1:** Display a customer email using record (one row)

**Ans:** declare

Customer_rec Customer%rowtype; begin

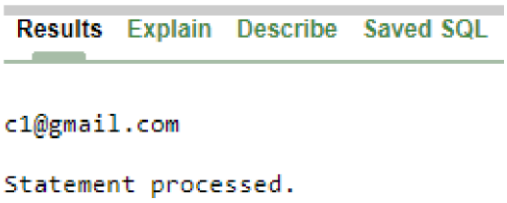
select * into Customer_rec from Customer

where c_name='c1';

dbms_output.put_line(Customer_rec.c_Email);

end

/




**Q2:** Display employee name and their addresses using record (multiple row)

**Ans:** declare

Employee_rec Employee%rowtype; begin for Employee_rec in(select * from Employee) loop dbms_output.put_line(Employee_rec.Employee_Name||' '||Employee_rec.e_Address); end loop; end

/


```
d1 Dhaka-BD
d2 Kushtia-BD
d3 Pabna-BD
d4 Cumilla-BD
d5 Chittagong-BD

Statement processed.
```

**Q3:** Display the transaction details for one transaction id using record (table-based)

**Ans:**

```
DECLARE
  Transaction_rec Transaction%rowtype;
BEGIN
  SELECT * into Transaction_rec
  FROM Transaction
WHERE T_id = 447;
  dbms_output.put_line('Transaction ID: ' || Transaction_rec.T_id);
dbms_output.put_line('Transaction sales type: ' || Transaction_rec.Sales_Type);
dbms_output.put_line('Transaction item name: ' || Transaction_rec.Item_name);
dbms_output.put_line('Transaction qantity: ' || Transaction_rec.Quantity);
dbms_output.put_line('Transaction time and date: ' || Transaction_rec.Time_Date);
dbms_output.put_line('Transaction cstomer id: ' || Transaction_rec.Customer_id);  END;
/
```

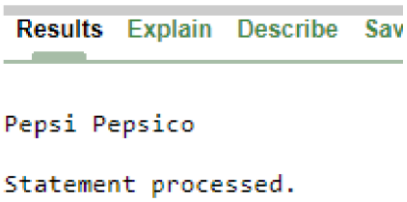

```
Transaction ID: 447
Transaction sales type: Online
Transaction item name: Water Melon
Transaction qantity: 3
Transaction time and date: 22-OCT-22
Transaction cstomer id: 336

Statement processed.
```

## -3 Cursor

**Q1:** Display an item name and its brand using cursor (one row)

**Ans:** declare

```
I_name Item.Item_name%type; I_brand
Item.Brand%type; cursor c_Item is select
Item_name,Brand from Item; begin open
c_Item; fetch c_Item into I_name ,I_brand
; dbms_output.put_line(I_name||' '||I_brand
); close c_Item; end
/
```




**Q2:** Display all importer name and their address using cursor(multiple row)

**Ans:** declare

```
importer_name Importer.i_name%type; Importer_address
Importer.i_address%type; cursor c_Importer is select
i_name,i_address from Importer; begin open c_Importer;
loop fetch c_Importer into
importer_name,Importer_address; exit when
c_Importer%notfound;
dbms_output.put_line(importer_name||'
'||Importer_address); end loop; close c_Importer; end
/
```


```
x Beijing@China
x2 Chittagong@BD
x3 Tokyo@Japan
x4 Rajshahi@Bangladesh
x5 Macau@China

Statement processed.
```

**Q3:** Update item price by 10 taka using cursor (implicit cursor attributes)

**Ans:**

```
DECLARE  var_rows number(5);
BEGIN
  UPDATE Item
  SET price = price + 10;
  IF SQL%NOTFOUND THEN
    dbms_output.put_line('None of the price were updated');
ELSIF SQL%FOUND THEN     var_rows := SQL%ROWCOUNT;
    dbms_output.put_line('Price for ' || var_rows || 'Items are updated');
  END IF;
END;
```


```
Price for 5Items are updated

1 row(s) updated.
```

## -3 Trigger

**Q1:** Create the Category trigger and update it

**Ans:**

```
CREATE OR REPLACE TRIGGER Category_added after
INSERT ON Category
FOR EACH ROW  BEGIN
```

```
dbms_output.put_line('New Category Added');  END;
select * from Category; insert into
Category values ('782','cate6'); rollback
```


```
Trigger created.
```

**Q2:** Create the Item_Cate trigger and use (BEFORE UPDATE)

**Ans:**

```
CREATE or REPLACE TRIGGER Item_Cate_Update
BEFORE UPDATE ON Item_Cate
FOR EACH ROW
Begin
dbms_output.put_line('Id updated');
END;
/
```


```
Trigger created.
```

**Q3:** Create a row level trigger for the Transaction table that would fire for DELETE operations performed on the Transaction table.

**Ans:**

```
CREATE OR REPLACE TRIGGER display_Quantity_changes
BEFORE DELETE ON Transaction
FOR EACH ROW   DECLARE
  quantity_diff number;   BEGIN
  quantity_diff := :NEW.Quantity- :OLD.Quantity;
dbms_output.put_line('Old Quantity: ' || :OLD.Quantity);
```

```
dbms_output.put_line('New Quantity: ' || :NEW.Quantity);
dbms_output.put_line('Quantity difference: ' || quantity_diff );
END;
/
```

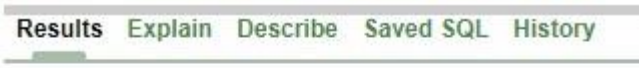

```
Trigger created.
```

## -3 Package

**Q1:** Create a package, which can display item name

**Ans:**

```
CREATE PACKAGE Item_pack AS
PROCEDURE display_name(i_Id
Item.Id%type);
END Item_pack ;
/
```

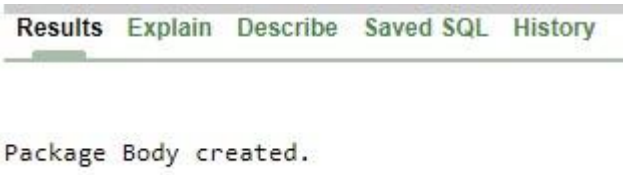



```
Package created.
```

**Q2:** Create a package body, which can display item name **Ans:**

```
CREATE or Replace PACKAGE BODY Item_pack AS
PROCEDURE display_name(I_id Item.Id%TYPE) IS
I_name Item.Item_name%TYPE;
BEGIN
SELECT Item_name INTO I_name
FROM Item
```

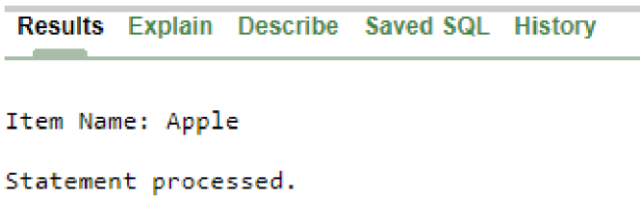
```
WHERE Id= I_id ;  dbms_output.put_line('Item

Name: '|| I_name );  END display_name;


END Item_pack ;

/
```




**Q3:** Display item name for a item id using package

**Ans:**

```
begin

Item_pack.display_name('6667');

End
```




# Conclusion:

This is a primary scenario of a superstore management system. The main lacking of this application is we cannot add all the sectors of a superstore. In reality the superstore management system is too big and the stored data is huge. But it might be helpful for them who want to build a superstore. Owner can implement the management of a superstore by taking the idea