DATABASE MANAGEMENT SYSTEM

LAB PROJECT SUBMISSION

PROJECT NAME : **INVENTORY MANAGEMENT SYSTEM**

SUBMITTED BY :

ANANYA GAUR

102216114

INDEX

INDEX OF THE PROJECT

INTRODUCTION

An inventory management system is the combination of technology (hardware and software) and processes and procedures that oversee the monitoring and maintenance of stocked products, whether those products are company assets, raw materials and supplies, or finished products ready to be sent to vendors or end consumers. This system can widely be used by normal shops, departmental stores or MNCs for keeping a proper track of the stock. It also consists of information like customer details etc. An Inventory Management System (IMS) is a crucial component of any business involved in buying, storing, and selling physical goods. It serves as the backbone for efficiently managing inventory levels, tracking stock movements, and optimising supply chain operations.

Scope:

- This will help us in maintain the exact count of any product.
- To generate an alert when the product stock falls below the minimum stock level, ensuring that stocks are replenished to avoid stockouts.
- Can reduce duplicate entries.

Goals of proposed system:

1. Planned approach towards working: - The working in the organization will be well planned and organized. The data will be stored properly in data stores, which will help in retrieval of information as well as its storage.
2. Accuracy: - The level of accuracy in the proposed system will be higher. All operation would be done correctly and it ensures that whatever information is coming from the center is accurate.
3. Reliability: - The reliability of the proposed system will be high due to the above stated reasons. The reason for the increased reliability of the system is that now there would be proper storage of information.
4. No Redundancy: - In the proposed system utmost care would be that no information is repeated anywhere, in storage or otherwise. This would assure economic use of storage space and consistency in the data stored.
5. Immediate retrieval of information: - The main objective of proposed system is to provide for a quick and efficient retrieval of information

Requirement Analysis

Stakeholder Identification: Identify all stakeholders involved in the inventory management process, including warehouse managers, inventory clerks, purchasing agents, sales representatives, and finance department personnel.

Gather Requirements: Engage with stakeholders through interviews, surveys, and workshops to gather their requirements. Ask open ended questions to allow them to express their needs and expectations. Record their responses and categorize them into functional and non-functional requirements.

Define Functional Requirements: Define the functional requirements of the system , which describe what the system should do.
For Example:-
Inventory tracking: Ability to add, edit, and delete inventory items.

Define Non-Functional Requirements: Define Non-functional requirements of the system, which describe how the system should perform.
For example:-
Performance: Response times for inventory queries and transactions.
Scalability: Ability to handle increasing volumes of inventory and transactions.

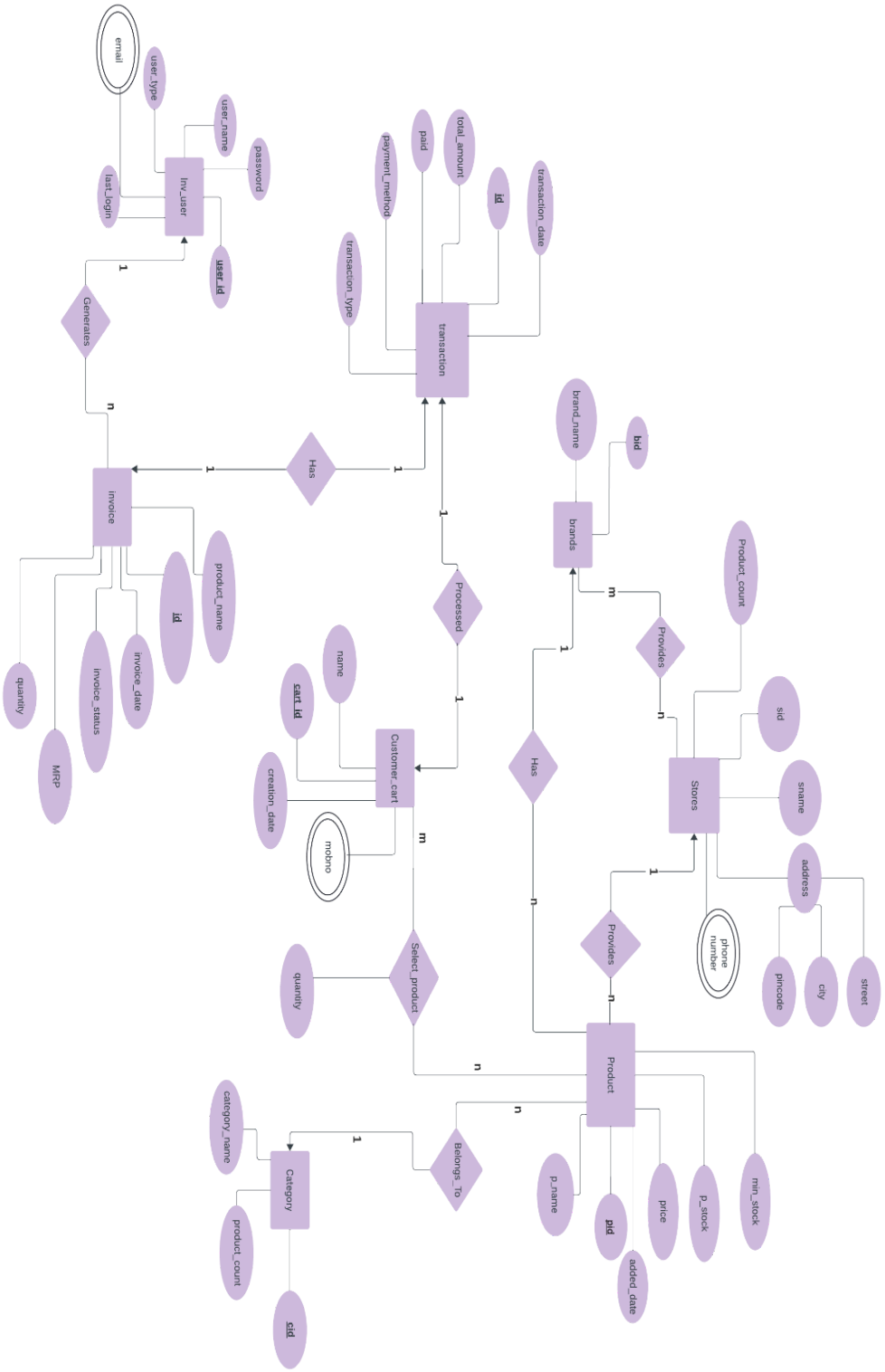
Reliability:The system should be reliable and available for use during business hours. It should have built-in mechanisms for error detection and recovery to minimize downtime and data loss.

Prioritize Requirements: Prioritize the requirements based on their importance and urgency. This will help the development team to focus on the most critical features first.

Document Requirements: Create a detailed requirements document that clearly outlines each requirement, including its description, priority, acceptance criteria, and any dependencies.

Review and Validation: Review the requirements document with stakeholders to ensure that all needs are accurately captured and understood. Validate the requirements against real-world scenarios and potential edge cases.

ER Diagrams


email
user_type
user_name
password
last_login
Inv_user
user_id
Generates
1
n
transaction
paid
total_amount
payment_method
id
transaction_date
transaction_type
Has
invoice
product_name
id
quantity
MRP
invoice_status
invoice_date
Processed
Customer_cart
cart_id
name
creation_date
mobno
m
Select_product
quantity
brand_name
bid
brands
m
Provides
Product_count
sid
Stores
sname
address
phone number
pincode
city
street
Has
Provides
Product
p_name
pid
price
p_stock
min_stock
added_date
Belongs_To
Category
category_name
product_count
cid

TABLES

On this sequence query language we created 10 tables named:

1. Brands
2. inv_user
3. Categories
4. Products
5. Stores
6. Provides
7. Customer_cart
8. Select_product
9. Transaction
10. Invoice

ER TO TABLES

Relational schema for an inventory management system:

1. Brands(bid , bname)
2. inv_user(user_id, user_name, password, last_login, user_type, email)
3. categories(cid, category name , product_count)
4. product(cid, bid, sid , pid, p_name, p_stock, min_stock, price, added_date)
   Foreign Keys : cid (references Categories Table)
   sid (references Stores Table)
   bid (references brands Table)

5. stores(sid , sname , address, phoneNo, product_count)
6. provides(bid,sid)
   Composite Primary Key: (bid,sid)
   Foreign Keys: sid (references Stores Table)
   bid (references brands Table)

7. customer_cart(cart_id, cust_id , name , mobno, creation_date)
   Foreign Keys: cust_id (references Inv_Users Table)
8. select_product(cart_id,pid,quantity)
   Composite Primary Key: cart_id,pid)
   Foreign Keys: cart_id (references customer_cart Table)
   pid (references product Table)
9. transaction(id, total amount ,paid ,payment method,cart_id, transaction_date, transaction_type)
   Foreign Keys: cart_id (references customer_cart Table)
10. invoice(id,product_name, net_price, quantity, user_id, transaction_id, invoice_date, invoice_status)
    Foreign Keys: user_id (references Inv_Users Table)
    transaction_id (references transaction Table)

NORMALIZATION

Categories table:

1. **First Normal Form (1NF):**

Ensure each field contains a single value. In this case, it does.

2. **Second Normal Form (2NF):**

In 2NF, all non-key attributes must depend entirely on the primary key. Here, all non-key attributes are determined by primary key.
So, table is in 2NF

3. **Third Normal Form (3NF):**

Ensure that the table is in 2NF.

Remove transitive dependencies by making sure that non-key attributes depend only on the primary key and not on other non-key attributes.

There is no transitive dependency. Each attribute is directly related to the primary key `cid`.

4. **BCNF:**

Functional Dependencies (FD):

- Cid -> Cname
- Cid ->

product_count

Therefore,table is in

BCNF

Brand name:

1. **First Normal Form (1NF):**

- Ensures that each column contains atomic (indivisible) values, with each row having a unique primary key.
- current table definition already meets 1NF.

2. **Second Normal Form (2NF):**

- Requires 1NF and that all non-key attributes are fully dependent on the primary key.
- table with `bid` as the primary key and `bname` does not violate this norm. It's entirely dependent on `bid`.
- thus table is in 2NF

3. **Third Normal Form (3NF):**

- Requires 2NF and that no transitive dependencies exist among attributes.
-current `Brands` table, there is no transitive dependency. Each attribute is directly related

to the primary key `bid`.

In summary, the provided `Brands` table already meets the requirements for 1NF, 2NF, and 3NF. No additional normalization steps are required for this table, as it does not contain redundancies or transitive dependencies.

4. **BCNF:**

Functional Dependencies (FD):

- Bid -> Bname

Therefore,table is in

BCNF

Inv_user table:

First Normal Form (1NF):

- Current table does not meets 1NF because it contains multivalued attribute email
- Therefore table must be normalized into:
- **Inv_User :** (user_id,name,password,last_login)
- **Inv_email :** (user_id,email)

Second Normal Form (2NF):

- Both tables are in 1NF, and every non-key attribute should be fully dependent on the primary key.
- The current structure has a clear relationship with the primary key, `user_id`, ensuring 2NF.
- Both tables are in 2NF.

Third Normal Form (3NF):

This normal form ensures there's no transitive dependency, meaning that all non-key attributes are dependent solely on the primary key.

table has a unique primary key (`user_id`), and all attributes directly relate to it, meeting the requirements for 1NF, 2NF, and 3NF.

BCNF:

- **Inv_User**

Functional Dependencies (FD):

- user_id -> name
- user_id -> password
- user_id ->

last_login Therefore,table

is in BCNF

- **Inv_email**

Functional Dependencies (FD):

- user_id,email -> no other attribute

Therefore,table is in BCNF

Product table:

1. First Normal Form (1NF):
    - Each column should contain atomic values.
    - There should be no repeating groups or arrays of data.
    - All entries in each column should be of the same kind.

The provided table appears to satisfy 1NF as each column contains atomic values without any repeating groups.

2. Second Normal Form (2NF):
    - It should satisfy 1NF.
    - All non-key attributes should be fully functionally dependent on the primary key.

The primary key is `pid`. Let's analyze the dependencies:

- `pid` -> `cid`, `bid`, `sid`, `pname`, `p_stock`, `price`, `added_date`, `minStock`

All non-key attributes seem to be fully functionally dependent on the primary key. Thus, it satisfies 2NF.

3. Third Normal Form (3NF):
    - It should satisfy 2NF.
    - There should be no transitive dependencies.

In this table, we have the following dependencies:

- `pid` -> `cid`, `bid`, `sid`, `pname`, `p_stock`, `price`, `added_date`, `minStock`

There don't seem to be any transitive dependencies in this table. All attributes depend directly on the primary key.

Therefore, the `Product` table appears to be normalized up to the third normal form (3NF)

BCNF:

- `pid` -> `cid`, `bid`, `sid`, `pname`, `p_stock`, `price`, `added_date`, `minStock`

Therefore table is in BCNF

Stores table:

1. **First Normal Form (1NF):**
   - Ensure that each column contains atomic values. This means that each cell in the table should contain only one value, and there should be no repeating groups or arrays.
   - The stores table already does not satisfy the requirements of 1NF as address is a composite attribute and phoneNo is a multivalues attribute.
   - In order to normalize in 1NF, this will be broken down into 2 tables:

Stores(sid,sname,Street,City,Pin code,product_count) : Primary key = sid

SPhone(sid,phoneNo) : Composite Primary key = (sid,phoneNo)
Both tables are in 1NF.

2. **Second Normal Form (2NF):**
   - Ensure that the table is in 1NF.
   - Remove partial dependencies by making sure that non-key attributes depend fully on the primary key.
   - ·Since now the above table is in 1NF
     - **Stores** table : sid is primary key and rest all keys are fully functionally dependent on the bid, then the table is already in 2NF.
     - **SPhone** table : (sid,phoneNo) is a composite primary key and there is no partial functional dependency, then the **table is already in 2NF.**
3. **Third Normal Form (3NF):**
   - · Ensure that the table is in 2NF.
   - · Remove transitive dependencies by making sure that non-key attributes depend only on the primary key and not on other non-key attributes.

Stores(sid,sname,Street,City,Pincode,product_count) : Primary key = sid

- There is a transitive dependency between pincode->(Street,City)
- Therefore, this table is not in 3NF
- To make it in 3NF, we need to normalize table into 2

tables: **Stores**(sid,sname,Street,City,product_count) : Primary

key =sid **SPin**(Sid,Pincode) : Primary key=sid

Now, there are no apparent transitive dependencies

- Therefore, **appears to be in 3NF**

SPhone(sid,phoneNo) : Composite Primary key = (sid,phoneNo)

- There are no apparent transitive dependencies, as there are no other non-key attributes.
- Therefore, appears to be in 3NF

4. **BCNF:**
   - A table is in BCNF if, for every non-trivial functional dependency X -> Y, where X is a superkey, Y is a candidate key. In simpler terms, a table is in BCNF if every determinant is a candidate key.

Stores table (sid, sname, Street, City, product_count) with the primary key (sid):

1. **Functional Dependencies (FD):**
   - sid -> sname (Each store ID uniquely determines the store name)
   - sid -> Street (Each store ID uniquely determines the street)
   - sid -> City (Each store ID uniquely determines the city)
   - sid -> product_count (Each store ID uniquely determines the product count)

Since the primary key (sid) determines all other attributes individually, and each non-key attribute depends only on the whole primary key, there are no non-trivial functional dependencies where the determinant is not a superkey.

Therefore, all non-trivial functional dependencies satisfy the criteria for BCNF, and the Stores table (sid, sname, Street, City, product_count) with the primary key (sid) is in Boyce-Codd Normal Form (BCNF).

SPin(Sid,Pincode) : Primary key=sid

Functional Dependencies (FD):

- Sid -> Pincode (Each store ID uniquely determines the pin code)

In this case, the functional dependency Sid -> Pincode implies that the pin code depends on the store ID. Since the primary key (Sid) determines the pin code, there are no non-trivial functional dependencies where the determinant is not a superkey.

Therefore, all non-trivial functional dependencies satisfy the criteria for BCNF, and the table SPin (Sid, Pincode) with the primary key Sid is in Boyce-Codd Normal Form (BCNF).

SPhone(sid,phoneNo) : Composite Primary key = (sid,phoneNo)

Functional Dependencies (FD):

- (sid, phoneNo) -> None, as both sid and phoneNo together uniquely identify each row.

In this case, the composite primary key (sid, phoneNo) uniquely identifies each row in the table. Since there are no non-trivial functional dependencies where the determinant is not a superkey, the table satisfies the criteria for BCNF.

Therefore, the table SPhone (sid, phoneNo) with the composite primary key (sid, phoneNo) is in Boyce-Codd Normal Form (BCNF).

All tables are in BCNF

Provides table:

5. **First Normal Form (1NF):**

- Ensures that each column contains atomic (indivisible) values, with each row having a unique primary key.
- current table definition already meets 1NF.

6. **Second Normal Form (2NF):**

- Requires 1NF and that all non-key attributes are fully dependent on the primary key.
- table with `bid` and 'sid' as the composite primary key -Thus table is in 2NF

7. **Third Normal Form (3NF):**

- Requires 2NF and that no transitive dependencies exist among attributes. Table is in 3NF

8. **BCNF:**

Functional Dependencies (FD):

- bid,sid -> no other attribute Therefore,table is in BCNF

Customer_cart table:

First Normal Form (1NF):

- Current table does not meets 1NF because it contains multivalued attribute mobno
- Therefore table must be normalized into:
- **Cart_User :** (cart_id,name,creation_date,cust_id)
- **Cart_mob :** (cart_id,mobno)

Second Normal Form (2NF):

- Both tables are in 1NF, and every non-key attribute should be fully dependent on the primary key.
- The current structure has a clear relationship with the primary key, `cart_id`, ensuring 2NF.

- Both tables are in 2NF.

Third Normal Form (3NF):

This normal form ensures there's no transitive dependency, meaning that all non-key attributes are dependent solely on the primary key.

table has a unique primary key (`cart_id`), and all attributes directly relate to it, meeting the requirements for 1NF, 2NF, and 3NF.

BCNF:

- **Cart_User :**
  cart_id -> name

  cart_id -> creation_date

  cart_id -> cust_id

  Therefore,table is in

  BCNF

- **Cart_mob:**

Functional Dependencies (FD):

  - cart_id,mobno -> no other

  attribute Therefore,table is in BCNF

Select_product table:

1. First Normal Form (1NF):
   - Each column should contain atomic values.
   - There should be no repeating groups or arrays of data.
   - All entries in each column should be of the same kind.

The table seems to satisfy 1NF as each column contains atomic values and there are no repeating groups.

2. Second Normal Form (2NF):

- It should satisfy 1NF.
  - All non-key attributes should be fully functionally dependent on the primary key.

The table has composite primary key candidates: `(cust_id, pid)`.

Functional dependencies:

- `cust_id, pid` **->** `quantity`

All non-key attributes (`quantity`) seem to be fully functionally dependent on the primary key `(cust_id, pid)`, which includes the foreign keys. Thus, it satisfies 2NF.

3. Third Normal Form (3NF):
    - It should satisfy 2NF.
    - There should be no transitive dependencies.

In this table, there are no transitive dependencies, as each non-key attribute depends directly on the primary key.

The table seems to be normalized up to the third normal form (3NF).

BCNF:

- `cust_id, pid` **->** `quantity`

Therefore table is in BCNF

Transaction Table:

1. First Normal Form (1NF):
    - Each column should contain atomic values.
    - There should be no repeating groups or arrays of data.
    - All entries in each column should be of the same kind.

The provided table appears to meet the requirements of 1NF as each column contains atomic values and there are no repeating groups.

2. Second Normal Form (2NF):
    - It should satisfy 1NF.
    - All non-key attributes should be fully functionally dependent on the primary key.

The primary key is `id`**.**

Functional

dependencies:

- id -> total_amount, paid, payment_method, transaction_date, transaction_type, cart_id

All non-key attributes seem to be fully functionally dependent on the primary key. However, `cart_id` is a foreign key, which suggests there might be another table involved, namely `customer_cart`. This foreign key relationship suggests that `cart_id` may be a candidate for removal to achieve 2NF.

3. Third Normal Form (3NF):
    - It should satisfy 2NF.
    - There should be no transitive dependencies.

4. BCNF:

- `id` -> `total_amount`, `paid`, `payment_method`, `transaction_date`, `transaction_type`, `cart_id`

Therefore table is in BCNF

Invoice table:

1. First Normal Form (1NF):
    - Each column should contain atomic values.
    - There should be no repeating groups or arrays of data.
    - All entries in each column should be of the same kind.

The provided table seems to satisfy 1NF. Each column contains atomic values, and there are no repeating groups.

2. Second Normal Form (2NF):
    - It should satisfy 1NF.
    - All non-key attributes should be fully functionally dependent on the primary key.

The primary key here is `id`. Let's examine the dependencies:

- `id` -> `product_name`, `quantity`, `MRP`, `invoice_date`, `invoice_status`, `user_id`, `transaction_id`
- `user_id` -> `inv_user(user_id)`
- `transaction_id` -> `transaction(id)`

All non-key attributes (`product_name`, `quantity`, `MRP`, `invoice_date`, `invoice_status`, `user_id`, `transaction_id`) seem to be functionally dependent on the primary key. However, `user_id` and `transaction_id` have foreign key dependencies.

3. Third Normal Form (3NF):
    - It should satisfy 2NF.

- There should be no transitive dependencies.

Therefore,table is in 3NF

4. BCNF:

- `id` -> `product_name`, `quantity`, `MRP`, `invoice_date`, `invoice_status`, `user_id`, `transaction_id`

Therefore table is in BCNF

SQL CODE IMPLEMENTATION

1. Brands table

```
create table Brands(
    bid Number(5) primary key,
    bname varchar(20)
    );

Table created.
```

```
insert into Brands Values(11,'Samsung');
insert into Brands Values(12,'Pepperfry');
insert into Brands Values(13,'LG');
insert into Brands Values(14,'Philips');
insert into Brands Values(15,'Apple');
insert into Brands Values(16,'Hasbro');
insert into Brands Values(17,'Classmate');
insert into Brands Values(18,'H&M');
insert into Brands Values(19,'HP');
insert into Brands Values(20,'Nilkamal');
insert into Brands Values(21,'Pigeon');
insert into Brands Values(22,'Nike');
insert into Brands Values(23,'Nivea');
```

BID	BNAME
11	Samsung
12	Pepperfry
13	LG
14	Philips
15	Apple
16	Hasbro
17	Classmate
18	H&M
19	HP
20	Nilkamal

2. Inv_user

```
CREATE TABLE Inv_user(
 user_id varchar(20) primary key,
 name varchar(20),
 password varchar(20),
 last_login timestamp,
 user_type varchar(10) CHECK (user_type IN('Administrator','Manager','Clerk','Customer')),
 email varchar(100)
 );
```

Table created.

```
INSERT INTO Inv_user(user_id, name, password, last_login, user_type,email)
VALUES('1', 'Aman Gupta', 'pass123', TIMESTAMP '2024-04-12 10:54:23', 'Administrator','aman@gmail.com');
INSERT INTO Inv_user(user_id, name, password, last_login, user_type,email)
VALUES(2, 'Shreya ', 'abcs1234', TIMESTAMP '2024-04-11 11:54:13.000' , 'Manager','shreya@gmail.com');
INSERT INTO Inv_user(user_id, name, password, last_login, user_type,email)
VALUES(3, 'Gagan Singh ', 'p4ofg',TIMESTAMP '2024-03-15 09:34:23.000' , 'Clerk','gagan@gmail.com');
INSERT INTO Inv_user(user_id, name, password, last_login, user_type,email)
VALUES(4, 'Neha Gupta ', 'dripws',TIMESTAMP '2024-02-12 07:50:23.000' , 'Customer','neha@gmail.com');
INSERT INTO Inv_user(user_id, name, password, last_login, user_type,email)
VALUES(5, 'Ram', 'arrwws',TIMESTAMP '2024-02-12 07:50:23.000' , 'Customer','ram@gmail.com');
-- SELECT * FROM user_tables;
INSERT INTO Inv_user(user_id, name, password, last_login, user_type,email)
VALUES(6, 'Naman ', 'strril',TIMESTAMP '2024-02-12 07:50:23.000' , 'Customer','naman@gmail.com');
INSERT INTO Inv_user(user_id, name, password, last_login, user_type,email)
VALUES(7, 'Raman', 'diwws',TIMESTAMP '2024-02-12 07:50:23.000' , 'Customer','raman@gmail.com');
INSERT INTO Inv_user(user_id, name, password, last_login, user_type,email)
VALUES(8, 'Naira ', '1233s',TIMESTAMP '2024-02-12 07:50:23.000' , 'Customer','neha@gmail.com');
INSERT INTO Inv_user(user_id, name, password, last_login, user_type,email)
VALUES(9, 'Arushi Talwar', '23sedd',TIMESTAMP '2024-02-12 07:50:23.000' , 'Customer','arushi@gmail.com');
INSERT INTO Inv_user(user_id, name, password, last_login, user_type,email)
VALUES(10, 'Aditi Vasudeva', 'ueysw',TIMESTAMP '2024-02-12 07:50:23.000' , 'Customer','aditi@gmail.com');
```

USER_ID	NAME	PASSWORD	LAST_LOGIN	USER_TYPE	EMAIL
2	Shreya	abcs1234	11-APR-24 11.54.13.000000 AM	Manager	shreya@gmail.com
3	Gagan Singh	p4ofg	15-MAR-24 09.34.23.000000 AM	Clerk	gagan@gmail.com
4	Neha Gupta	dripws	12-FEB-24 07.50.23.000000 AM	Customer	neha@gmail.com
5	Ram	arrwws	12-FEB-24 07.50.23.000000 AM	Customer	ram@gmail.com
6	Naman	strril	12-FEB-24 07.50.23.000000 AM	Customer	naman@gmail.com
7	Raman	diwws	12-FEB-24 07.50.23.000000 AM	Customer	raman@gmail.com
8	Naira	1233s	12-FEB-24 07.50.23.000000 AM	Customer	neha@gmail.com
9	Arushi Talwar	23sedd	12-FEB-24 07.50.23.000000 AM	Customer	arushi@gmail.com
10	Aditi Vasudeva	ueysw	12-FEB-24 07.50.23.000000 AM	Customer	aditi@gmail.com
1	Aman Gupta	pass123	12-APR-24 10.54.23.000000 AM	Administrator	aman@gmail.com

3. Categories Table

```
CREATE TABLE Categories(
 cid number(5) primary key,
 category_name varchar(20),
 product_count number(6)
 );
```

```
Table created.
```

CID	CATEGORY_NAME	PRODUCT_COUNT
1	Electronics	0
2	Grocery	0
3	Appliances	0
4	HealthCare	0
5	Cosmetics	0
6	Furniture	0
7	Home Furnishings	0
8	Clothing	0
9	Toys&Games	0
10	Stationery	0

```
INSERT INTO Categories(cid, category_name,product_count)
VALUES(1,'Electronics',0) ;
INSERT INTO Categories(cid, category_name,product_count)
VALUES(2,'Grocery',0) ;
INSERT INTO Categories(cid, category_name,product_count)
VALUES(3,'Appliances',0) ;
INSERT INTO Categories(cid, category_name,product_count)
VALUES(4,'HealthCare',0) ;
INSERT INTO Categories(cid, category_name,product_count)
VALUES(5,'Cosmetics',0) ;
INSERT INTO Categories(cid, category_name,product_count)
VALUES(6,'Furniture',0) ;
INSERT INTO Categories(cid, category_name,product_count)
VALUES(7,'Home Furnishings',0) ;
INSERT INTO Categories(cid, category_name,product_count)
VALUES(8,'Clothing',0) ;
INSERT INTO Categories(cid, category_name,product_count)
VALUES(9,'Toys&Games',0) ;
INSERT INTO Categories(cid, category_name,product_count)
VALUES(10,'Stationery',0) ;
```

4. Product table

```
CREATE TABLE Product(
 pid number(5) primary key,
 cid number(5) references categories(cid),
 bid number(5) references brands(bid),
 sid number(5) references stores(sid),
 pname varchar(20),
 p_stock number(5),
 price number(5),
 added_date date,
 minStock number(3)
);
```

```
Table created.
```

```
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock, bid, sid )
VALUES(101, 1, 'Mobile Phone S21', 25, 65000, TO_DATE('2024-02-18', 'YYYY-MM-DD'), 10,11,3);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock, bid, sid  )
VALUES(102, 6, 'Sofa', 15, 45000, TO_DATE('2024-03-08', 'YYYY-MM-DD'), 5,12,5);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock , bid, sid )
VALUES(103, 3, 'Washing Machine', 25, 45000, TO_DATE('2024-01-10', 'YYYY-MM-DD'), 10,13,3);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock, bid, sid  )
VALUES(104, 3, 'Fan', 21, 2000, TO_DATE('2024-02-18', 'YYYY-MM-DD'), 8,14,3);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock , bid, sid )
VALUES(105, 1, 'Mobile iPhone 15', 25, 95000, TO_DATE('2024-02-14', 'YYYY-MM-DD'), 10,15,3);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock , bid, sid )
VALUES(106, 9, 'Jenga', 25, 400, TO_DATE('2024-03-03', 'YYYY-MM-DD'), 10,16,2);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock , bid, sid )
VALUES(107, 10, 'Single-line Notebook', 110, 150, TO_DATE('2024-04-25', 'YYYY-MM-DD'), 40,17,2);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock , bid, sid )
VALUES(108, 8, 'Skirt', 45, 450, TO_DATE('2024-01-16', 'YYYY-MM-DD'), 20,18,1);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock , bid, sid )
VALUES(109, 1, 'Laptop', 25, 65000, TO_DATE('2024-02-18', 'YYYY-MM-DD'), 10,19,3);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock , bid, sid )
VALUES(110, 10, 'Pencil Box', 15, 150, TO DATE('2024-03-08', 'YYYY-MM-DD'), 5,17,2);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock , bid, sid )
VALUES(111, 5, 'Moisturizer', 25, 250, TO_DATE('2024-01-10', 'YYYY-MM-DD'), 10, 23, 4);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock , bid, sid )
VALUES(112, 7, 'Table', 21, 2000, TO_DATE('2024-02-18', 'YYYY-MM-DD'), 8,20,5);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock, bid, sid  )
VALUES(113, 7, 'Bed', 25, 75000, TO_DATE('2024-02-14', 'YYYY-MM-DD'), 10,20,5);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock, bid, sid  )
VALUES(114, 3, 'Tubelight', 25, 2000, TO_DATE('2024-03-03', 'YYYY-MM-DD'), 10,14,3);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock , bid, sid )
VALUES(115, 3, 'Electric Kettle', 50, 3000, TO_DATE('2024-04-25', 'YYYY-MM-DD'), 30,21,3);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock, bid, sid  )
VALUES(116, 8, 'Shoes', 85, 7000, TO_DATE('2024-01-16', 'YYYY-MM-DD'), 20,22,1);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock  )
VALUES(117, 8, 'Shoes', 85, 7000, TO_DATE('2024-01-16', 'YYYY-MM-DD'), 20);
INSERT INTO Product(pid, cid, pname, p_stock, price, added_date, minStock, bid)
VALUES(118, 8, 'Shoes', 85, 7000, TO_DATE('2024-01-16', 'YYYY-MM-DD'), 20,22);
```

PID	PNAME	P_STOCK	PRICE	ADDED_DATE	MINSTOCK	CID	BID	SID
103	Washing Machine	25	45000	10-JAN-24	10	3	13	3
104	Fan	21	2000	18-FEB-24	8	3	14	3
105	Mobile iPhone 15	25	95000	14-FEB-24	10	1	15	3
106	Jenga	25	400	03-MAR-24	10	9	16	2
107	Single-line Notebook	110	150	25-APR-24	40	10	17	2
108	Skirt	45	450	16-JAN-24	20	8	18	1
109	Laptop	25	65000	18-FEB-24	10	1	19	3
110	Pencil Box	15	150	08-MAR-24	5	10	17	2
111	Moisturizer	25	250	10-JAN-24	10	5	23	4
112	Table	21	2000	18-FEB-24	8	7	20	5
113	Bed	25	75000	14-FEB-24	10	7	20	5
114	Tubelight	25	2000	03-MAR-24	10	3	14	3
115	Electric Kettle	50	3000	25-APR-24	30	3	21	3
116	Shoes	85	7000	16-JAN-24	20	8	22	1
117	Shoes	85	7000	16-JAN-24	20	8	-	-
102	Sofa	15	45000	08-MAR-24	5	6	12	5
101	Mobile Phone S21	25	65000	18-FEB-24	10	1	11	3

5. Stores table

```
create table stores(
sid number(5) PRIMARY KEY ,
sname varchar(20),
address varchar(20),
phoneno number(10) ,
product_count number(4)
)
```

```
Table created.
```

```
insert into stores(sid, sname, address, phoneNo,product_count)
values(1,'Shoppers Stop','Katpadi vellore',9456781234,0);
insert into stores(sid, sname, address, phoneNo,product_count)
values(2,'Spar Hypermarket','Katpadi vellore',8457811111,0);
insert into stores(sid, sname, address, phoneNo,product_count)
values(3,'Tata Croma','Katpadi vellore',7456123456,0);
insert into stores(sid, sname, address, phoneNo,product_count)
values(4,'Health Buddy','Katpadi vellore',1256799234,0);
insert into stores values(5,'Fabindia','Katpadi vellore',9999781234,0);
```

SID	SNAME	ADDRESS	PHONENO	PRODUCT_COUNT
1	Shoppers Stop	Katpadi vellore	9456781234	130
2	Spar Hypermarket	Katpadi vellore	8457811111	150
3	Tata Croma	Katpadi vellore	7456123456	196
4	Health Buddy	Katpadi vellore	1256799234	25
5	Fabindia	Katpadi vellore	9999781234	61

6. select_product

```
create table select_product(
    ct_id INT references customer_cart(cart_id),
    pid number(5)references product(pid),
    quantity number(4),
    PRIMARY KEY(ct_id, pid)
    );
```

```
Table created.
```

```
insert into select_product Values(2,103,50);
insert into select_product Values(5,101,60);
insert into select_product Values(8,115,30);
insert into select_product Values(6,102,45);
insert into select_product Values(4,113,24);
insert into select_product Values(5,106,62);
```

ct_id	pid	quantity
2	103	50
5	101	60
8	115	30
6	102	45
4	113	24
5	106	62

7. Provides Table

```
create table provides(
    bid number(5) references brands(bid),
    sid number(5) references stores(sid),
    PRIMARY KEY(sid, bid)
    );
```

```
Table created.
```

```
INSERT INTO PROVIDES
    VALUES(13,3);
INSERT INTO PROVIDES
    VALUES(14,3);
INSERT INTO PROVIDES
    VALUES(15,3);
INSERT INTO PROVIDES
    VALUES(16,2);
INSERT INTO PROVIDES
    VALUES(17,2);
INSERT INTO PROVIDES
    VALUES(18,1);
INSERT INTO PROVIDES
    VALUES(19,3);
INSERT INTO PROVIDES
    VALUES(20,5);
INSERT INTO PROVIDES
    VALUES(21,2);
INSERT INTO PROVIDES
    VALUES(22,1);
INSERT INTO PROVIDES
```

BID	SID
18	1
22	1
16	2
17	2
21	2
11	3
13	3
14	3
15	3
19	3
23	4
12	5
20	5

8. Customer_Cart Table:

```
create table customer_cart(
 cart_id INT primary key,
 name varchar(20),
 mobno number(10) ,
creation_date DATE,
cust_id number(5) references inv_user(user_id)
 );
```

Table created.

```
INSERT INTO Customer_Cart (cart_id, cust_id, name, mobno, creation_date) VALUES
(2, 5, 'Ram', '8580761225', DATE '2023-06-19');

INSERT INTO Customer_Cart (cart_id, cust_id, name, mobno, creation_date) VALUES
(3, 6, 'Naman', '7475113524', DATE '2023-04-03');

INSERT INTO Customer_Cart (cart_id, cust_id, name, mobno, creation_date) VALUES
(4, 7, 'Raman', '6569342108', DATE '2024-01-15');

INSERT INTO Customer_Cart (cart_id, cust_id, name, mobno, creation_date) VALUES
(5, 8, 'Naira', '8580761225', DATE '2024-02-19');

INSERT INTO Customer_Cart (cart_id, cust_id, name, mobno, creation_date) VALUES
(6, 9, 'Arushi Talwar', '6569342108', DATE '2024-03-20');

INSERT INTO Customer_Cart (cart_id, cust_id, name, mobno, creation_date) VALUES
(7, 10, 'Aditi Vasudeva', '9415632149', DATE '2024-02-08');

INSERT INTO Customer_Cart (cart_id, cust_id, name, mobno, creation_date) VALUES
(8, 5, 'Ram', '8580761225', DATE '2023-06-19');
```

```
1 row(s) inserted.

1 row(s) inserted.
```

CART_ID	NAME	MOBNO	CREATION_DATE	CUST_ID
2	Ram	8580761225	29-APR-24	5
3	Naman	7475113524	29-APR-24	6
4	Raman	6569342108	29-APR-24	7
5	Naira	8580761225	29-APR-24	8
6	Arushi Talwar	6569342108	29-APR-24	9
7	Aditi Vasudeva	9415632149	29-APR-24	10
8	Ram	8580761225	29-APR-24	5

9. Invoice Table

```
create table invoice(
 id INT primary key,
 product_name varchar(20),
 quantity number(5),
 MRP decimal(5),
invoice_date DATE,
 invoice_status varchar(50) DEFAULT 'PAID',
user_id varchar(20) references inv_user(user_id),
transaction_id INT references transaction(id)
 )
```

Table created.

```
INSERT INTO Invoice (id, transaction_id,product_name,quantity, user_id, invoice_date, invoice_status) VALUES
(1, 1,'Mobile iPhone 15',1, 1 ,Date'2023-11-27', 'Paid');
    INSERT INTO Invoice (id, transaction_id,product_name,  quantity, user_id, invoice_date, invoice_status) VALUES
(2,2, 'skirt', 2, 2,Date '2023-06-19 ', 'Paid');
    INSERT INTO Invoice (id, transaction_id,product_name,  quantity, user_id, invoice_date, invoice_status) VALUES
(3,3, 'sofa',  1,3, Date'2023-04-03 ', 'Paid');
    INSERT INTO Invoice (id, transaction_id,product_name,  quantity, user_id, invoice_date, invoice_status) VALUES
(4,4, 'washing machine',1, 4,Date '2024-01-15 ', 'Paid');
    INSERT INTO Invoice (id, transaction_id,product_name,  quantity, user_id, invoice_date, invoice_status) VALUES
(5,5, 'Jenga', 2, 2, Date'2024-02-19 ', 'Paid');
    INSERT INTO Invoice (id, transaction_id,product_name,  quantity, user_id, invoice_date, invoice_status) VALUES
(6,6, 'Fan', 2, 4, Date'2024-03-20 ', 'Paid');
    INSERT INTO Invoice (id, transaction_id,product_name,  quantity, user_id, invoice_date, invoice_status) VALUES
(7,7, 'single-line Notebook',  4, 1,Date '2024-02-08  ', 'Paid');
    INSERT INTO Invoice (id, transaction_id,product_name,  quantity, user_id, invoice_date, invoice_status) VALUES
(8,8, 'Mobile Phone S21', 1, 3, Date'2023-05-11 ', 'Paid');
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

ID	PRODUCT_NAME	QUANTITY	MRP	INVOICE_DATE	INVOICE_STATUS	USER_ID	TRANSACTION_ID
2	skirt	2	225	19-JUN-23	Paid	2	2
4	washing machine	1	45000	15-JAN-24	Paid	4	4
5	Jenga	2	400	19-FEB-24	Paid	2	5
6	Fan	2	2000	20-MAR-24	Paid	4	6
7	single-line Notebook	4	150	08-FEB-24	Paid	1	7
8	Mobile Phone S21	1	65000	11-MAY-23	Paid	3	8

10. Transaction Table

```
create table transaction(
 id number(5) primary key,
 total_amount decimal(10,2),
 paid decimal(10,2),
 payment_method varchar(50),
 transaction_date DATE,
transaction_type varchar(50) default 'purchase',
 cart_id number(5) references customer_cart(cart_id)
 )
```

```
Table created.
```

```
INSERT INTO Transaction (id, total_amount, paid,  payment_method, cart_id, transaction_date, transaction_type) VALUES
(1, 95000.00, 95000.00,  'Credit Card', 1, DATE '2023-11-27', 'Purchase');
    INSERT INTO Transaction (id, total_amount, paid, payment_method, cart_id, transaction_date, transaction_type) VALUES
(2, 450.00, 450.00,  'UPI', 2 ,DATE'2023-06-19 ', 'Purchase');
    INSERT INTO Transaction (id, total_amount, paid,  payment_method, cart_id, transaction_date, transaction_type) VALUES
(3, 45000.00, 45000.00,  'Online Banking', 3,DATE '2023-04-03 ', 'Purchase');
    INSERT INTO Transaction (id, total_amount, paid,  payment_method, cart_id, transaction_date, transaction_type) VALUES
(4, 45000.00, 45000.00, 'Cash', 4,DATE '2024-01-15', 'Purchase');
    INSERT INTO Transaction (id, total_amount, paid,  payment_method, cart_id, transaction_date, transaction_type) VALUES
(5, 800.00, 800.00, 'Online Banking', 5,DATE '2024-02-19 ', 'Purchase');
    INSERT INTO Transaction (id, total_amount, paid,  payment_method, cart_id, transaction_date, transaction_type) VALUES
(6, 4000.00, 4000.00,  'Debit Card', 6,DATE'2024-03-20', 'Purchase');
    INSERT INTO Transaction (id, total_amount, paid, payment_method, cart_id, transaction_date, transaction_type) VALUES
(7, 600.00, 600.00, 'UPI', 7,DATE'2024-02-08 ', 'Purchase');
    INSERT INTO Transaction (id, total_amount, paid,  payment_method, cart_id, transaction_date, transaction_type) VALUES
(8, 65000.00, 65000.00, 'Cash', 8, DATE'2024-01-11  ','Purchase');
INSERT INTO Transaction (id, total_amount, paid,  payment_method, cart_id, transaction_date, transaction_type) VALUES
(9, 250.00, 250.00, 'Cash', 9, DATE'2024-01-11  ','Purchase');
```

```
1 row(s) inserted.

1 row(s) inserted.
```

ID	TOTAL_AMOUNT	PAID	PAYMENT_METHOD	TRANSACTION_DATE	TRANSACTION_TYPE	CART_ID
2	450	450	UPI	19-JUN-23	Purchase	2
3	45000	45000	Online Banking	03-APR-23	Purchase	3
4	45000	45000	Cash	15-JAN-24	Purchase	4
5	800	800	Online Banking	19-FEB-24	Purchase	5
6	4000	4000	Debit Card	20-MAR-24	Purchase	6
7	600	600	UPI	08-FEB-24	Purchase	7
8	65000	65000	Cash	11-MAY-23	Purchase	8

PL/SQL CODES

TRIGGERS:

1. Trigger to update the stock of each product's category after a product is added

```
CREATE OR REPLACE TRIGGER update_category_count_on_insert
AFTER INSERT ON Product
FOR EACH ROW
BEGIN
  UPDATE Categories
  SET product_count = product_count + :NEW.p_stock
  WHERE cid = :NEW.cid;
END;
```

2. Trigger to check if product stock < min stock to generate alert to reorder stock

```
    CREATE OR REPLACE TRIGGER check_stock_threshold
AFTER INSERT OR UPDATE ON select_product
FOR EACH ROW
DECLARE
    v_p_stock product.p_stock%TYPE;
    v_minStock product.minStock%TYPE;
BEGIN
    SELECT p_stock, minStock INTO v_p_stock, v_minStock
    FROM product
    WHERE pid = :NEW.pid;
    IF v_p_stock < v_minStock THEN
        DBMS_OUTPUT.PUT_LINE('Product stock is below the minimum stock threshold.');
    END IF;
END;
```

3. Trigger to keep track of cart creation date as soon as any product is added in cart

```
CREATE OR REPLACE TRIGGER update_creation_date
BEFORE INSERT ON Customer_Cart
FOR EACH ROW
BEGIN
    :NEW.creation_date := SYSDATE;
END;
```

4. Trigger to update product count as soon as new product comes in store

```
CREATE OR REPLACE TRIGGER update_product_count
AFTER INSERT ON product
FOR EACH ROW
DECLARE
    store_id NUMBER;
BEGIN
    store_id := :NEW.sid;

    UPDATE stores
    SET product_count = product_count + :NEW.p_stock
    WHERE sid = store_id;
END;
```

5. Trigger to update stock in Products table as soon as any user adds product in cart

```
CREATE OR REPLACE TRIGGER update_stock_trigger
BEFORE INSERT ON select_product
FOR EACH ROW
BEGIN
  UPDATE Product
  SET p_stock = p_stock - :NEW.quantity
  WHERE pid = :NEW.pid;
END;
```

FUNCTIONS

1. Function to get product stock by giving id as input parameter

```
CREATE OR REPLACE FUNCTION get_product_stock(id NUMBER)
RETURN NUMBER IS
    stock_qty NUMBER;
BEGIN
    SELECT p.p_stock
    INTO stock_qty
    FROM Product p
    WHERE p.pid = id;

    RETURN stock_qty;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL; -- Return NULL if product ID is not found
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20001, 'An error occurred while retrieving product stock.');
END get_product_stock;
```

```
DECLARE
    stock_qty NUMBER;
BEGIN
    stock_qty := get_product_stock(101);
    IF stock_qty IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('Stock Quantity: ' || stock_qty);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Product not found.');
    END IF;
END;
```

```
Statement processed.
Stock Quantity: 25
```

2. Function to get brand name by giving brand id as input parameter

```
CREATE OR REPLACE FUNCTION get_brand_info(brand_id IN NUMBER) RETURN VARCHAR2 IS
    brand_name VARCHAR2(100);
BEGIN
    SELECT brand_name INTO brand_name
    FROM brandS
    WHERE brand_id = bid;

    RETURN brand_name;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 'Brand Not Found';
END;
```

```
DECLARE
    brand_name VARCHAR2(20);
BEGIN
    brand_name := get_brand_info(11);
    DBMS_OUTPUT.PUT_LINE('Brand Name: ' || brand_name);
END;
/
```

```
Function created.


Statement processed.
Brand Name: Samsung
```

3. Function to calculate total price for an invoice id

```
CREATE OR REPLACE FUNCTION calculate_invoice_total(
   p_price DECIMAL,
   p_quantity INT
)
RETURN DECIMAL
IS
   v_total_price DECIMAL(10,2);
BEGIN
   v_total_price := p_price * p_quantity;
   RETURN v_total_price;
END;
```

```
      DECLARE
        v_invoice_id INT;
        v_customer_name VARCHAR2(50);
        v_product_name VARCHAR2(50);
        v_quantity NUMBER(5);
        v_mrp DECIMAL(5);
        v_total_price DECIMAL(10,2);
      BEGIN

        v_invoice_id := 123;


        SELECT customer_name, product_name, quantity, mrp
        INTO v_customer_name, v_product_name, v_quantity, v_mrp
        FROM invoice

FROM invoice
WHERE id = v_invoice_id;


v_total_price := calculate_invoice_total(v_mrp, v_quantity);


DBMS_OUTPUT.PUT_LINE('Invoice ID: ' || v_invoice_id);
DBMS_OUTPUT.PUT_LINE('Customer Name: ' || v_customer_name);
DBMS_OUTPUT.PUT_LINE('Product Name: ' || v_product_name);
DBMS_OUTPUT.PUT_LINE('Quantity: ' || v_quantity);
DBMS_OUTPUT.PUT_LINE('MRP: ' || v_mrp);
DBMS_OUTPUT.PUT_LINE('Total Price: ' || v_total_price);

END;
```

CURSORS & PROCEDURES

1. Cursor to get entire brand info

```
DECLARE
    CURSOR brand_cursor IS
        SELECT bid, bname
        FROM brands;
BEGIN
    FOR brand_rec IN brand_cursor LOOP
        DBMS_OUTPUT.PUT_LINE('Brand ID: ' || brand_rec.bid || ', Brand Name: ' || brand_rec.bname);
    END LOOP;
END;
```

```
Statement processed.
Brand ID: 11, Brand Name: Samsung
Brand ID: 12, Brand Name: Pepperfry
Brand ID: 13, Brand Name: LG
Brand ID: 14, Brand Name: Philips
Brand ID: 15, Brand Name: Apple
Brand ID: 16, Brand Name: Hasbro
Brand ID: 17, Brand Name: Classmate
Brand ID: 18, Brand Name: H&M
Brand ID: 19, Brand Name: HP
Brand ID: 20, Brand Name: Nilkamal
Brand ID: 21, Brand Name: Pigeon
Brand ID: 22, Brand Name: Nike
Brand ID: 23, Brand Name: Nivea
```

2. Procedure to update user info by taking id as input parameter

```
CREATE OR REPLACE PROCEDURE update_user_info(
id IN VARCHAR , new_password IN VARCHAR
    ) IS
BEGIN
  UPDATE Inv_user
  SET password = new_password
  WHERE user_id = id;
END update_user_info;

DECLARE
  user_id NUMBER := 1;
  new_password VARCHAR2(20) := 'newpass21';
BEGIN
  update_user_info(user_id, new_password);
  DBMS_OUTPUT.PUT_LINE('User password updated successfully!');

END;
```

```
Statement processed.
User password updated successfully!
```

3. Procedure to Add a New Brand:

```
CREATE OR REPLACE PROCEDURE add_brand(
    brand_id IN NUMBER,
    brand_name IN VARCHAR2

) IS
BEGIN
    INSERT INTO brands (bid, bname)
    VALUES (brand_id, brand_name);
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Brand added successfully.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error adding brand: ' || SQLERRM);
        ROLLBACK;
END;
```

4. Cursor to fetch all product details

```
    CURSOR product_cursor IS
        SELECT pid, cid, bid, sid, pname, p_stock, price, added_date, minStock
        FROM Product;

BEGIN
    -- Open cursor
    OPEN product_cursor;

    -- Fetch and display each row from the cursor
    LOOP
        FETCH product_cursor INTO v_pid, v_cid, v_bid, v_sid, v_pname, v_p_stock, v_price, v_added_date, v_minStock;
        EXIT WHEN product_cursor%NOTFOUND;

        -- Display product details (You can replace this with any action you want to perform with the retrieved data)
        DBMS_OUTPUT.PUT_LINE('Product ID: ' || v_pid);
        DBMS_OUTPUT.PUT_LINE('Category ID: ' || v_cid);
        DBMS_OUTPUT.PUT_LINE('Brand ID: ' || v_bid);
        DBMS_OUTPUT.PUT_LINE('Store ID: ' || v_sid);
        DBMS_OUTPUT.PUT_LINE('Product Name: ' || v_pname);
        DBMS_OUTPUT.PUT_LINE('Stock: ' || v_p_stock);
        DBMS_OUTPUT.PUT_LINE('Price: ' || v_price);
        DBMS_OUTPUT.PUT_LINE('Added Date: ' || TO_CHAR(v_added_date, 'YYYY-MM-DD'));
        DBMS_OUTPUT.PUT_LINE('Minimum Stock: ' || v_minStock);
    END LOOP;

    -- Close cursor
    CLOSE product_cursor;
END;
```

```
Statement processed.
Product ID: 103
Category ID: 3
Brand ID: 13
Store ID: 3
Product Name: Washing Machine
Stock: 25
Price: 45000
Added Date: 2024-01-10
Minimum Stock: 10
Product ID: 104
Category ID: 3
Brand ID: 14
Store ID: 3
Product Name: Fan
Stock: 21
Price: 2000
Added Date: 2024-02-18
Minimum Stock: 8
Product ID: 105
Category ID: 1
Brand ID: 15
Store ID: 3
Product Name: Mobile iPhone 15
Stock: 25
Price: 95000
Added Date: 2024-02-14
Minimum Stock: 10
Product ID: 106
Category ID: 9
Brand ID: 16
Store ID: 2
Product Name: Jenga
Stock: 25
Price: 400
Added Date: 2024-03-03
Minimum Stock: 10
Product ID: 107
Category ID: 10
Brand ID: 17
Store ID: 2
Product Name: Single-line Notebook
Stock: 110
Price: 150
Added Date: 2024-04-25
Minimum Stock: 40
Product ID: 108
Category ID: 8
Brand ID: 18
Store ID: 1
Product Name: Skirt
Stock: 45
Price: 450
Added Date: 2024-01-16
Minimum Stock: 20
Brand ID: 20
Store ID: 5
Product Name: Table
Stock: 21
Price: 2000
Added Date: 2024-02-18
Minimum Stock: 8
Product ID: 109
Category ID: 1
Brand ID: 19
Store ID: 3
Product Name: Laptop
Stock: 25
Price: 65000
Added Date: 2024-02-18
Minimum Stock: 10
Product ID: 110
Category ID: 10
Brand ID: 17
Store ID: 2
Product Name: Pencil Box
Stock: 15
Price: 150
Added Date: 2024-03-08
Minimum Stock: 5
Product ID: 111
Category ID: 5
Brand ID: 23
Store ID: 4
Product Name: Moisturizer
Stock: 25
Price: 250
Added Date: 2024-01-10
Minimum Stock: 10
Product ID: 112
Category ID: 7
Brand ID: 20
Store ID: 5
Product Name: Bed
Stock: 25
Price: 75000
Added Date: 2024-02-14
Minimum Stock: 10
Product ID: 114
Category ID: 3
Brand ID: 14
Store ID: 3
Product Name: Tubelight
Stock: 25
Price: 2000
Added Date: 2024-03-03
Minimum Stock: 10
Product ID: 115
Category ID: 3
Brand ID: 21
Store ID: 3
```

```
Product Name: Electric Kettle
Stock: 50
Price: 3000
Added Date: 2024-04-25
Minimum Stock: 30
Product ID: 116
Category ID: 8
Brand ID: 22
Store ID: 1
Product Name: Shoes
Stock: 85
Price: 7000
Added Date: 2024-01-16
Minimum Stock: 20
Product ID: 117
Category ID: 8
Brand ID:
Store ID:
Product Name: Shoes
Stock: 85
Price: 7000
Added Date: 2024-01-16
Minimum Stock: 20
Product ID: 102
Category ID: 6
Brand ID: 12
Store ID: 5
Product Name: Sofa
Stock: 15
Price: 45000
```

ieve Store Information:

```
address, phoneNo
Added Date: 2024-03-08
Minimum Stock: 5
Product ID: 101
Category ID: 1
Brand ID: 11
Store ID: 3
Product Name: Mobile Phone S21
Stock: 25
Price: 65000
Added Date: 2024-02-18
Minimum Stock: 10
```

```
| store_rec.sname
);
```

```
, Phone No: 9456781234
ore, Phone No: 8457811111
one No: 7456123456
 Phone No: 1256799234
e No: 9999781234
```

6. Cursor To Fetch Transaction Details

```
    DECLARE
    CURSOR trans_cursor IS
        SELECT id, total_amount, payment_method FROM Transaction;
    v_transaction_id Transaction.id%TYPE;
    v_total_amount Transaction.total_amount%TYPE;
    v_payment_method Transaction.payment_method%TYPE;
BEGIN
    OPEN trans_cursor;
    LOOP
        FETCH trans_cursor INTO v_transaction_id, v_total_amount, v_payment_method;
        EXIT WHEN trans_cursor%NOTFOUND;
        -- Process the retrieved data
        DBMS_OUTPUT.PUT_LINE('Transaction ID: ' || v_transaction_id || ', Total Amount: '
            || v_total_amount || ', Payment Method: ' || v_payment_method);
    END LOOP;
    CLOSE trans_cursor;
END;
```

```
Statement processed.
Transaction ID: 2, Total Amount: 450, Payment Method: UPI
Transaction ID: 3, Total Amount: 45000, Payment Method: Online Banking
Transaction ID: 4, Total Amount: 45000, Payment Method: Cash
Transaction ID: 5, Total Amount: 800, Payment Method: Online Banking
Transaction ID: 6, Total Amount: 4000, Payment Method: Debit Card
Transaction ID: 7, Total Amount: 600, Payment Method: UPI
Transaction ID: 8, Total Amount: 65000, Payment Method: Cash
```

CONCLUSION

In this project we developed a complete back end software in which we can update the stock, modify stock, we can forecast the stock, generate invoice. From this application we can get an update that if a particular
inventory or stock is less than some pre-fixed quantity then it'll be easy for the manager/owner to reorder the product from the supplier to overcome the "Out of Stock" stage. We can have complete customer details which
can help us to retrieve the order details of regular customers. From this program we can also keep a track of transactions performed by different customers/clients. We can also get an idea of how much fund we received from different payment methodologies.

REFERENCES

- W3schools.com
- tutorialspoint.com
- YouTube.com