Inventory management System

Team: SQL HUSKIES
Karan Thakkar(002644239)
Aayush Soni(002208039)
Janvi Chitroda(002474422)
Harsh Agarwal(002839620)

Business Rules

- **Category**: A product must belong to only one category
- **Product order**: A customer cannot place bulk orders, he is entitled to order a product up to a specific quantity.Order can contain multiple products.
- **Supplier**: Suppliers will always have products in stock.
- **Warehouse**: Warehouse will store all the products and their quantity received from the supplier.
- **Customer**: Customers will have at least one address. Customers can place multiple orders.
- **Reviews**: Each review must be associated with one customer. Reviews cannot be anonymous.
- **Product**: Whenever a product goes out of stock, an order will be placed with the supplier.
- **Address**: Each Customer can have multiple addresses for the product to be delivered

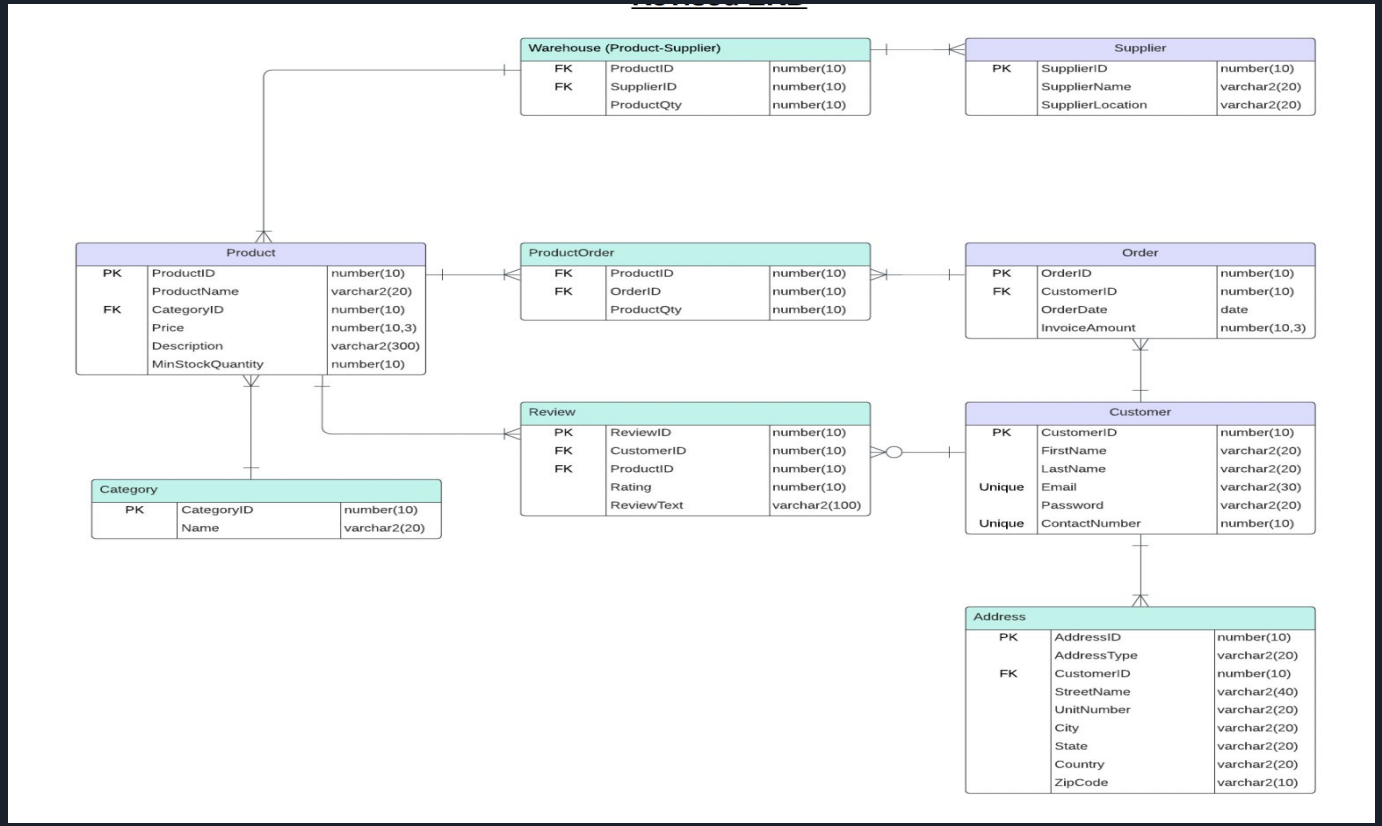
Introduction

Retailers grapple with challenges in inventory management, including overstocking, understocking, and data inconsistencies.

Our project aims to create a comprehensive, efficient, and user-friendly inventory management system to meet the evolving needs of modern retailers, ultimately leading to increased customer satisfaction and business profitability.

- **Data Integrity and Accuracy**: Ensure accurate, consistent, and reliable data through the implementation of constraints such as primary keys and foreign keys.
- **Efficient Inventory Tracking**: Track product quantities, locations, and movements within the warehouse to optimize stock levels and trigger alerts.
- **Customer Management:** Maintain customer information for order processing and customer service, enabling efficient retrieval of customer data.
- **Order Processing and Fulfillment:** Handle customer orders, update inventory quantities upon order placement and fulfillment.
- **Product Categorization:** Organize products into categories for easy search and filtering.
- **Review and Feedback:** Enable customers to provide feedback and ratings for products, enhancing customer engagement.

Entity Relationship Diagram



Views

- Product View: This view displays all the product details including name, description, price, etc. It can be used by customers for purchasing
- Order View : This view shows all the order details including customer information, product ordered, quantity, and order status. It is primarily used by customers to place orders and by the warehouse staff to process orders.
- Warehouse View : This view provides information about the stock in the warehouse. It is used by the warehouse staff to manage inventory and by suppliers to update stock.
- Supplier View : This view shows all the supplier details and the products they supply. It is used by the warehouse staff for restocking and by the management for supplier relations.
- Customer View: This view displays all the customer details. It is used by customers to manage their profile
- Customer Order History View: This view displays the history of orders placed by a particular customer. It is used by admin, auditor and customers to view their previous orders.
- Reviews View: This view shows all the reviews given by customers for different products. It is used by customers to make purchasing decisions and by suppliers to improve their products.

View For Inventory Manager

INVENTORYADMIN | InventoryManager | LOGISTICSMANAGER | SALESANALYST | CUSTOMERSERVICEMANAGER

0.50599998 seconds

Worksheet | Query Builder

```
SELECT * FROM INVENTORYADMIN.V_Product;
SELECT * FROM INVENTORYADMIN.V_ProductSales;
```

Script Output

Task completed in 0.506 seconds

PRODUCTID	PRODUCTNAME	CATEGORY	PRICE	DESCRIPTION
21	Iphone 15	1	999.999	Latest iPhone model
22	Samsung Galaxy S22	1	899.999	Latest Galaxy model
23	Hp Probook	2	599.999	Reliable business laptop
24	Macbook Air	2	999.999	Lightweight and powerful
25	Nikon D3500	3	449.999	Great for beginners
26	Canon Eos Rebel	3	499.999	Versatile DSLR
27	Ipad Pro	4	799.999	Powerful performance
28	Samsung Tab S7	4	649.999	High-resolution screen
29	Boat Earbuds 110	5	49.999	Clear sound, noise cancellation
30	Sony Wh-1000xm4	5	349.999	Industry-leading noise cancellation

10 rows selected.

PRODUCTNAME	TOTALQUANTITY	TOTALINVOICEAMOUNT
Iphone 15	1	100
Canon Eos Rebel	3	1000
Samsung Galaxy S22	3	700
Macbook Air	4	200
Nikon D3500	5	900

View For Sales Manager

INVENTORYADMIN | InventoryManager | LOGISTICSMANAGER | SALESANALYST | CUSTOMERSERVICEMANAGER

0.36300001 seconds

Worksheet | Query Builder

```
SELECT * FROM INVENTORYADMIN.V_Order;
SELECT * FROM INVENTORYADMIN.V_ProductSales;
SELECT * FROM INVENTORYADMIN.V_CustomerOrderFrequency;
```

Script Output

Task completed in 0.363 seconds

ORDERID	CUSTOMERID	ORDERDATE	INVOICEAMOUNT	EMAIL	PRODUCTNAME	PRODUCTQTY
11	15	13-APR-24	100	johndoe@email.com	Iphone 15	1
12	16	13-APR-24	700	janesmith@email.com	Samsung Galaxy S22	3
13	17	13-APR-24	200	michaelbrown@email.com	Macbook Air	4
14	19	13-APR-24	900	williamwilson@email.com	Nikon D3500	5
15	19	13-APR-24	1000	williamwilson@email.com	Canon Eos Rebel	3

PRODUCTNAME	TOTALQUANTITY	TOTALINVOICEAMOUNT
Iphone 15	1	100
Canon Eos Rebel	3	1000
Samsung Galaxy S22	3	700
Macbook Air	4	200
Nikon D3500	5	900

FIRSTNAME	LASTNAME	EMAIL	ORDERCOUNT
William	Wilson	williamwilson@email.com	2
Jane	Smith	janesmith@email.com	1
John	Doe	johndoe@email.com	1
Michael	Brown	michaelbrown@email.com	1

Table Creation

DDL Code snippet

```sql
-- Create tables
CREATE TABLE Product (
    ProductID NUMBER(10) NOT NULL PRIMARY KEY,
    ProductName VARCHAR2(20) NOT NULL UNIQUE,
    Category NUMBER(10) NOT NULL ,
    Price NUMBER(10,3) NOT NULL ,
    Description VARCHAR2(300) NOT NULL ,
    MinStockQuantity NUMBER(10) NOT NULL
);

CREATE TABLE Category (
    CategoryID NUMBER(10) NOT NULL  PRIMARY KEY,
    Name VARCHAR2(20) NOT NULL
);

CREATE TABLE Warehouse (
    ProductID NUMBER(10) NOT NULL ,
    SupplierID NUMBER(10) NOT NULL ,
    ProductQty NUMBER(10) NOT NULL
);

CREATE TABLE ProductOrder (
    ProductID NUMBER(10) NOT NULL ,
    OrderID NUMBER(10) NOT NULL ,
    ProductQty NUMBER(10) NOT NULL
);

CREATE TABLE Customer (
    CustomerID NUMBER(10) NOT NULL  PRIMARY KEY,
    FirstName VARCHAR2(20) NOT NULL ,
    LastName VARCHAR2(20) NOT NULL ,
    Email VARCHAR2(30)  NOT NULL UNIQUE,
    Password VARCHAR2(20) NOT NULL ,
    ContactNumber NUMBER(10)  NOT NULL UNIQUE
);

CREATE TABLE Address (
    AddressID NUMBER(10)  NOT NULL PRIMARY KEY,
    AddressType VARCHAR2(20) NOT NULL CHECK (AddressType IN ('Home', 'Work', 'Alternate')),
    CustomerID NUMBER(10) NOT NULL ,
    StreetName VARCHAR2(40) NOT NULL ,
    UnitNumber VARCHAR2(20) NOT NULL ,
    City VARCHAR2(20) NOT NULL ,
    State VARCHAR2(20) NOT NULL ,
    Country VARCHAR2(20) NOT NULL ,
    ZipCode VARCHAR2(10) NOT NULL
);

CREATE TABLE Review (
    ReviewID NUMBER(10) NOT NULL PRIMARY KEY,
    CustomerID NUMBER(10) NOT NULL,
    ProductID NUMBER(10) NOT NULL,
    Rating NUMBER(10) NOT NULL CHECK (Rating IN (1, 2, 3, 4, 5)),
    ReviewText VARCHAR2(100) NOT NULL
);

CREATE TABLE Supplier (
    SupplierID NUMBER(10) NOT NULL  PRIMARY KEY,
```

DDL Output

```
Script Output
Task completed in 3.092 seconds

Table PRODUCT created.

Table CATEGORY created.

Table WAREHOUSE created.

Table PRODUCTORDER created.

Table CUSTOMER created.

Table ADDRESS created.

Table REVIEW created.

Table SUPPLIER created.

Table CUSTOMERORDER created.
```

Packages

User Management Package: Contains Create User Procedure

INVENTORYADMIN | InventoryManager | LOGISTICSMANAGER | SALESANALYST | CUSTOMERSERVICEMANAGER

0.227 seconds

Worksheet | Query Builder

```
CREATE OR REPLACE PACKAGE UserManagementPackage AS
  PROCEDURE CreateUser(p_username IN VARCHAR2, p_password IN VARCHAR2);
END UserManagementPackage;
/
```

Script Output

Task completed in 0.227 seconds

```
Package USERMANAGEMENTPACKAGE compiled

Package Body USERMANAGEMENTPACKAGE compiled

PL/SQL procedure successfully completed.
```

Procedure To Add Customer

Code Snippet

```oracle_plsql
CREATE OR REPLACE PROCEDURE ADD_CUSTOMER (
    PI_FIRST_NAME Customer.FirstName%TYPE,
    PI_LAST_NAME Customer.LastName%TYPE,
    PI_EMAIL Customer.Email%TYPE,
    PI_PASSWORD Customer.Password%TYPE,
    PI_CONTACT_NUMBER Customer.ContactNumber%TYPE
) AS
    E_PASSWORD_VALID EXCEPTION;
    E_EMAIL_EXISTS EXCEPTION;
    E_CONTACT_NUMBER_EXISTS EXCEPTION;
    E_NUMERIC_NAME EXCEPTION; -- Exception for numeric values in the name

    V_EMAIL_COUNT NUMBER;
    V_CONTACT_NUMBER_COUNT NUMBER;
BEGIN
    -- Check if the password meets the minimum length requirement
    IF LENGTH(PI_PASSWORD) < 8 THEN
        RAISE E_PASSWORD_VALID;
    END IF;

    -- Check if the first name contains numeric values
    IF REGEXP_LIKE(PI_FIRST_NAME, '\d') THEN
        RAISE E_NUMERIC_NAME;
    END IF;

    -- Check if the last name contains numeric values
    IF REGEXP_LIKE(PI_LAST_NAME, '\d') THEN
        RAISE E_NUMERIC_NAME;
    END IF;

    -- Check if the email already exists
    SELECT COUNT(*) INTO V_EMAIL_COUNT FROM Customer WHERE Email = LOWER(PI_EMAIL);
    IF V_EMAIL_COUNT > 0 THEN
        RAISE E_EMAIL_EXISTS;
    END IF;

    -- Check if the contact number already exists
    SELECT COUNT(*) INTO V_CONTACT_NUMBER_COUNT FROM Customer WHERE ContactNumber = PI_CONTACT_NUMBER;
    IF V_CONTACT_NUMBER_COUNT > 0 THEN
        RAISE E_CONTACT_NUMBER_EXISTS;
    END IF;

    -- Insert the customer record
    INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Password, ContactNumber) VALUES (
        CustomerSeq.NEXTVAL,
        INITCAP(PI_FIRST_NAME),
        INITCAP(PI_LAST_NAME),
        LOWER(PI_EMAIL),
        PI_PASSWORD,
        PI_CONTACT_NUMBER
    );

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Customer added successfully');
EXCEPTION
    WHEN E_PASSWORD_VALID THEN
        DBMS_OUTPUT.PUT_LINE('Password should have at least 8 characters');
```

Execution And output

```oracle_plsql
-- Execute the ADD_CUSTOMER stored procedure for each customer record
EXEC ADD_CUSTOMER('John', 'Doe', 'johndoe@email.com', 'pass1234', 1234567890);
EXEC ADD_CUSTOMER('Jane', 'Smith', 'janesmith@email.com', 'pass1234', 2345678901);
EXEC ADD_CUSTOMER('Michael', 'Brown', 'michaelbrown@email.com', 'pass1234', 3456789012);
EXEC ADD_CUSTOMER('Emily', 'Davis', 'emilydavis@email.com', 'pass1234', 4567890123);
EXEC ADD_CUSTOMER('William', 'Wilson', 'williamwilson@email.com', 'pass1234', 5678901234);
EXEC ADD_CUSTOMER('Emma', 'Martinez', 'emmamartinez@email.com', 'pass1234', 6789012345);
EXEC ADD_CUSTOMER('Oliver', 'Taylor', 'olivertaylor@email.com', 'pass1234', 7890123456);
```

t Output

Task completed in 7.492 seconds

ws selected.

ERID	FIRSTNAME	LASTNAME	EMAIL	PASSWORD	CONTACTNUMBER
15	John	Doe	johndoe@email.com	pass1234	1234567890
16	Jane	Smith	janesmith@email.com	pass1234	2345678901
17	Michael	Brown	michaelbrown@email.com	pass1234	3456789012
18	Emily	Davis	emilydavis@email.com	pass1234	4567890123
19	William	Wilson	williamwilson@email.com	pass1234	5678901234
20	Emma	Martinez	emmamartinez@email.com	pass1234	6789012345
21	Oliver	Taylor	olivertaylor@email.com	pass1234	7890123456

selected.

Procedure To Add Supplier

Code Snippet

```oracle_plsql
---------- Stored procedure to add suppliers
CREATE OR REPLACE PROCEDURE ADD_SUPPLIER (
    PI_SUPPLIER_NAME Supplier.SupplierName%TYPE,
    PI_SUPPLIER_LOCATION Supplier.SupplierLocation%TYPE
) AS
    E_SUPPLIER_NAME_EXISTS EXCEPTION;
    E_INVALID_NAME EXCEPTION;
    E_INVALID_LOCATION EXCEPTION;
    E_NAME_TOO_LONG EXCEPTION;
    E_LOCATION_TOO_LONG EXCEPTION;

    -- Constants for maximum lengths based on your table definition
    CONST_MAX_NAME_LENGTH CONSTANT INTEGER := 20;
    CONST_MAX_LOCATION_LENGTH CONSTANT INTEGER := 20;
BEGIN
    -- Check for NULL inputs
    IF PI_SUPPLIER_NAME IS NULL THEN
        RAISE E_INVALID_NAME;
    END IF;

    IF PI_SUPPLIER_LOCATION IS NULL THEN
        RAISE E_INVALID_LOCATION;
    END IF;

    -- Check for input length
    IF LENGTH(PI_SUPPLIER_NAME) > CONST_MAX_NAME_LENGTH THEN
        RAISE E_NAME_TOO_LONG;
    END IF;

    IF LENGTH(PI_SUPPLIER_LOCATION) > CONST_MAX_LOCATION_LENGTH THEN
        RAISE E_LOCATION_TOO_LONG;
    END IF;

    -- Check for existing supplier name to ensure uniqueness
    FOR C IN (SELECT 1 FROM Supplier WHERE UPPER(SupplierName) = UPPER(PI_SUPPLIER_NAME)) LOOP
        RAISE E_SUPPLIER_NAME_EXISTS;
    END LOOP;

    -- Insert the new supplier with an ID generated from the sequence
    INSERT INTO Supplier (SupplierID, SupplierName, SupplierLocation) VALUES (
        SupplierSeq.NEXTVAL,
        INITCAP(PI_SUPPLIER_NAME),
        INITCAP(PI_SUPPLIER_LOCATION)
    );

    COMMIT; -- Commit the transaction to make sure the changes are saved

    -- Success message
    DBMS_OUTPUT.PUT_LINE('Supplier added successfully: ' || INITCAP(PI_SUPPLIER_NAME));

EXCEPTION
    WHEN E_SUPPLIER_NAME_EXISTS THEN
        DBMS_OUTPUT.PUT_LINE('Error: The supplier name "' || PI_SUPPLIER_NAME || '" already exists.');
    WHEN E_INVALID_NAME THEN
        DBMS_OUTPUT.PUT_LINE('Error: Supplier name cannot be null.');
    WHEN E_INVALID_LOCATION THEN
        DBMS_OUTPUT.PUT_LINE('Error: Supplier location cannot be null.');
    WHEN E_NAME_TOO_LONG THEN
```

Execution And Output

```oracle_plsql
-- Execute the ADD_SUPPLIER stored procedure for each supplier record
EXEC ADD_SUPPLIER('Apple', 'Cupertino');
EXEC ADD_SUPPLIER('HP', 'Palo Alto');
EXEC ADD_SUPPLIER('Boat', 'Mumbai');
EXEC ADD_SUPPLIER('Sony', 'Tokyo');
EXEC ADD_SUPPLIER('Nikon', 'Boston');
```

Script Output

Task completed in 7.492 seconds

SUPPLIERID	SUPPLIERNAME	SUPPLIERLOCATION
11	Apple	Cupertino
12	Hp	Palo Alto
13	Boat	Mumbai
14	Sony	Tokyo
15	Nikon	Boston

User Defined Function

Calculate Invoice Output

```oracle_plsql
/
CREATE OR REPLACE FUNCTION CalculateInvoiceAmount(
    p_productid PRODUCT.PRODUCTID%TYPE,
    p_qty NUMBER
) RETURN NUMBER IS
    v_price PRODUCT.PRICE%TYPE;
BEGIN
    -- Get the product price
    SELECT PRICE INTO v_price FROM PRODUCT WHERE PRODUCTID = p_productid;

    -- Calculate the invoice amount
    RETURN v_price * p_qty;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20002, 'Product not found');
    WHEN OTHERS THEN
        RAISE;
END CalculateInvoiceAmount;
/
SELECT CalculateInvoiceAmount(21, 10) AS InvoiceAmount FROM dual;
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 1 in 0.064 seconds

	INVOICEAMOUNT
1	9999.99

Contributions

Karan Thakkar: User Creation (InventoryManager, LogisticsManager, SalesAnalyst, CusomerServiceManager), Views (V_ProductOrderDetail), Stored Procedure(AddCustomerOrder, AddAddress, UpdateCustomer, UpdateAddress)

Janvi Chitroda: DDL, DML, V_ProductOrderDetail,View (V_RestockReport),Stored Procedure(AddCategory, AddSupplier, AddProduct, AddProductWarehouse, AddCustomer, UpdateSupplier, UpdateProduct, AddQuantityWarehouse),

Harsh Agarwal: UserPackage, Views (V_Product, V_Order, v_customeraddress, V_ProductReview , v_customerreview), Business Report, Documentation, UserDefined_Function

Aayush Soni: UserCreation(inventoryAdmin), View(V_ProductSales, V_CustomerOrderFrequency), Trigger

Collaborated Effort: Business Problem & Solution, ER Diagram , Business rules/constraints ,Logical Diagrams ,Relational Diagrams ,Data Flow Diagrams