INSTITUTE OF ENGINEERING

ADVANCE COLLEGE OF ENGINEERING AND MANAGEMENT

KALANKI, KATHMANDU

(AFFILATED TO TRIBHUVAN UNIVERSITY)

ADVANCED COLLEGE
OF ENGINEERING & MANAGEMENT

PROJECT REPORT

ON

Online Shopping Management System

<u>SUBMITTED BY:</u>

NAME: Manoj Gaire

ROLL NO: ACE080BCT036

SUBMITTED TO:

Shailesh Singh

Lecturer, Department Of Electronics and Computer Engineering

SUBMISSION DATE:2082/11/15

INTRODUCTION

➔ The Online Shopping Management System is a database-based application designed to manage the core activities of an online shopping platform. With the rapid growth of e-commerce, managing users, items, categories, and orders efficiently has become very important. This system helps in storing, organizing, and retrieving data related to users, products, and customer orders in a structured manner.

The project is developed using database concepts such as entities, relationships, primary keys, foreign keys, and SQL queries. It demonstrates how real-world online shopping processes can be represented using a relational database. The system is simple, realistic, and easy to understand, making it suitable for a DBMS mini project.

OBJECTIVES

➔ **The main objective of this project are:**

- To design a relational database for an online shopping system.
- To understand database transactions using COMMIT and ROLLBACK.
- To identify entities and relationships involved in online shopping.
- To implement tables using primary keys and foreign keys.

ENTITY LIST

➔**The database consists of five Entities, as required:**

1□Users

➔**Stores information about users who place orders.**

Attributes:

- **user_id (Primary Key)**
- **name**
- **email**
- **phone**

2□Categoryss

➔**Stores different item categories.**

Attributes:

- **category_id (Primary Key)**
- **category_name**

3□Items

➔**Stores details of products/items available for sale.**

Attributes:

- **item_id (Primary Key)**
- **item_name**
- **price**
- **category_id (Foreign Key)**

4□CustomerOrders

➔Stores information about orders placed by users.

Attributes:

- customerorder_id (Primary Key)
- order_date
- user_id (Foreign Key)

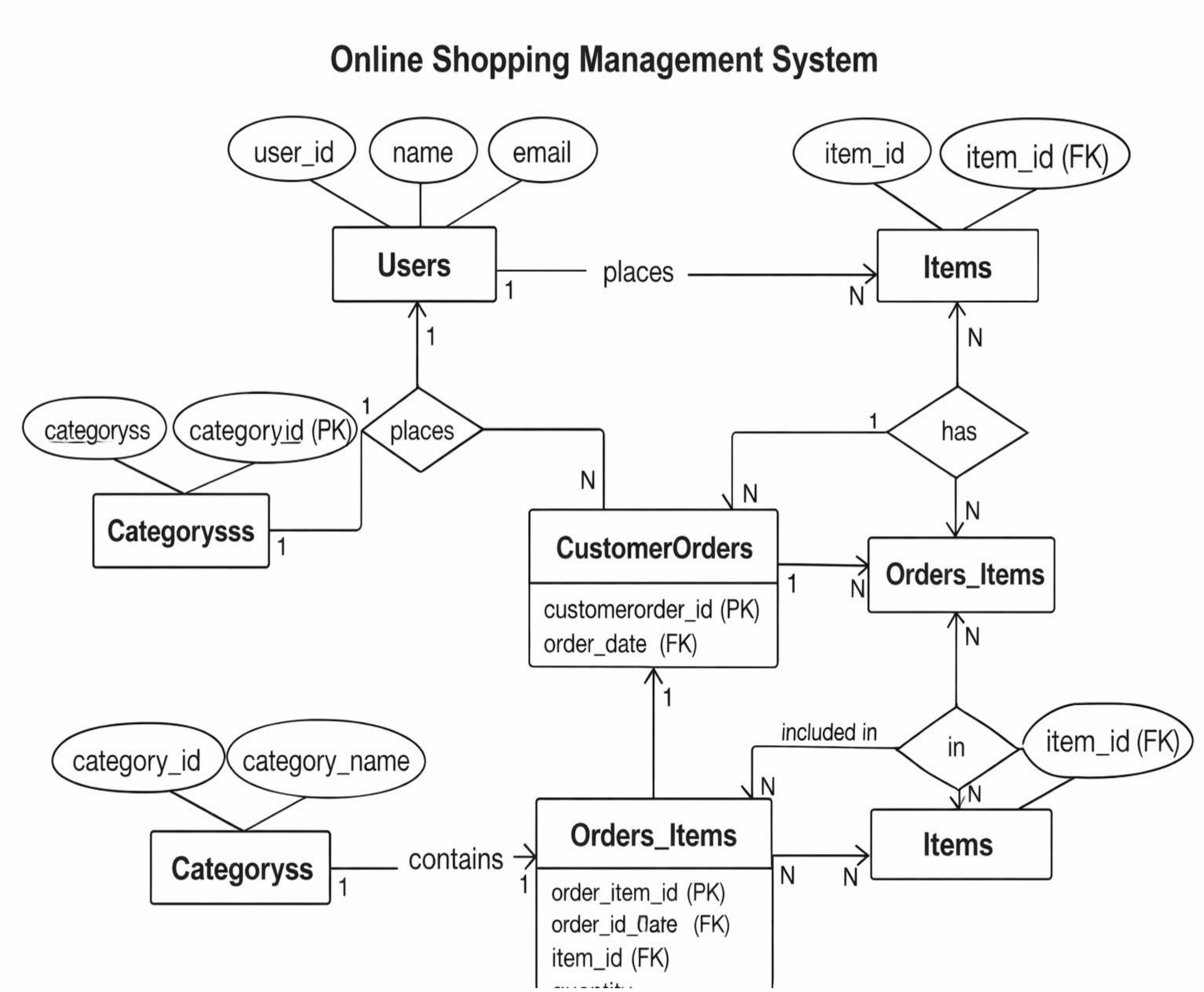
5□Orders_Items

➔Stores details of items included in each order.

Attributes:

- order_item_id (Primary Key)
- customerorder_id (Foreign Key)
- item_id (Foreign Key)
- quantity

ER DIAGRAM


Online Shopping Management System
user_id
name
email
item_id
item_id (FK)
Users
places
Items
1
N
1
N
categoryss
category.id (PK)
1
places
has
1
Categorysss
1
N
N
N
CustomerOrders
Orders_Items
1
N
customerorder_id (PK)
order_date (FK)
N
1
included in
in
item_id (FK)
category_id
category_name
N
N
Orders_Items
Items
Categoryss
contains
1
1
N
N
order_item_id (PK)
order_id_Date (FK)
item_id (FK)
quantity

APPENDIX:SOURCE CODE

Below is the Complete SQL script designed for this Project

```
CREATE TABLE Users (
   customer_id INTEGER PRIMARY KEY,
   name TEXT NOT NULL,
   email TEXT UNIQUE,
   phone TEXT
);

CREATE TABLE Categoryss (
   category_id INTEGER PRIMARY KEY,
   category_name TEXT NOT NULL
);

CREATE TABLE Items (
   product_id INTEGER PRIMARY KEY,
   product_name TEXT NOT NULL,
   price REAL NOT NULL,
   category_id INTEGER,
   FOREIGN KEY (category_id) REFERENCES Category(category_id)
);

CREATE TABLE CustomerOrders (
   order_id INTEGER PRIMARY KEY,
   order_date TEXT,
   customer_id INTEGER,
   FOREIGN KEY (customer_id) REFERENCES Users(customer_id)
);

CREATE TABLE Orders_Items (
   order_item_id INTEGER PRIMARY KEY,
   order_id INTEGER,
   product_id INTEGER,
   quantity INTEGER,
   FOREIGN KEY (order_id) REFERENCES CustomerOrders(order_id),
   FOREIGN KEY (product_id) REFERENCES Items(product_id)
);
DELETE from Users;
INSERT INTO Users VALUES
(1,'Manoj Gaire','manoj@gmail.com','98003535001'),
(2,'Sita Lamichane','sita@gmail.com','9804000002'),
(3,'Hari Thapa','hari@gmail.com','98000005678'),
(4,'Ram Sharma','ram@gmail.com','98241555161'),
(5,'Virat Kohli','virat@gmail.com','98422455717'),
(6,'Subham Bidari','shubham@gmail.com','985678432'),
(7,'Hamza Khan','hamza@gmail.com','97864245579');

INSERT INTO Categoryss VALUES
(1,'Electronics'),
(2,'Clothing'),
(3,'Home&Kitchen'),
(4,'Beauty & Personal Care'),
(5,'Books & Stationery');

INSERT INTO Items VALUES
(1,'Laptop',80000,1),
(2,'Tshirt',30000,2),
(3,'Cooker',1500,3),
(4,'FaceWash',4000,4),
(5,'Cricket Bat',7000,5),
(6,'Mobile Phones',35000,1),
(7,'Hoodie',6700,2);

INSERT INTO CustomerOrders VALUES
(1,'2026-03-01',1),
(2,'2026-03-02',2),
(3,'2026-03-03',3),
(4,'2026-02-29',4),
(5,'2026-03-04',5),
(6,'2026-03-15',6),
(7,'2026-03-21',7);

INSERT INTO Orders_Items VALUES
(1,1,1,2),
(2,2,2,1),
(3,3,3,2),
(4,4,4,1),
(5,5,5,4);

//Inner Join
SELECT U.name, O.order_id, O.order_date
FROM Users U
INNER JOIN CustomerOrders O
ON U.customer_id = O.customer_id;

//Left Join
SELECT I.product_name, OI.order_id
FROM  Items I
LEFT JOIN Orders_Items OI
```

```
ON I.product_id = OI.product_id;

//Aggregate(Count+Group By)
SELECT customer_id, COUNT(order_id) AS total_orders
FROM CustomerOrders
GROUP BY customer_id;

//Aggregate(Sum/Avg)
SELECT AVG(price) AS average_price FROM Items;

//SubQuery
SELECT name
FROM Users
WHERE customer_id IN (
   SELECT customer_id
   FROM CustomerOrders
   GROUP BY customer_id
   HAVING COUNT(order_id) > 1
);

//VIEW
CREATE VIEW Order_Summary AS
SELECT U.name, I.product_name, OI.quantity
FROM Users U
JOIN Orders O ON u.customer_id = O.customer_id
JOIN Orders_Items OI ON O.order_id = OI.order_id
JOIN Items I ON I.product_id = OI.product_id;

//Transaction(Commit/RollBack)
BEGIN TRANSACTION;

INSERT INTO Orders VALUES (4,'2026-03-04',3);
INSERT INTO Order_Items VALUES (5,4,2,1);

COMMIT;
```

Discussion and Conclusion:

➔ The Online Shopping Management System database was successfully designed and implemented using SQLite. All required entities, relationships, and constraints were properly defined. SQL operations such as joins, aggregate functions, subqueries, views, and transactions were executed correctly. This project helped in understanding ER modeling, database design, and practical SQL implementation. Overall, the system efficiently manages online shopping data in a simple and reliable manner.