Online Bookstore Management System

Project submitted to the
SRM University – AP, Andhra Pradesh
for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology
in
Computer Science and Engineering
School of Engineering and Sciences

Submitted by

Dandamudi Gowreesh Raja – AP22110010081
B Hemanth – AP22110010074
Munagala Shanmukha Krishna Chaitanya – AP22110010089
Jetti Kundan Kumar – AP22110010116

SRM University–AP
Neerukonda, Mangalagiri, Guntur
Andhra Pradesh – 522 240
[May, 2024]

Contents

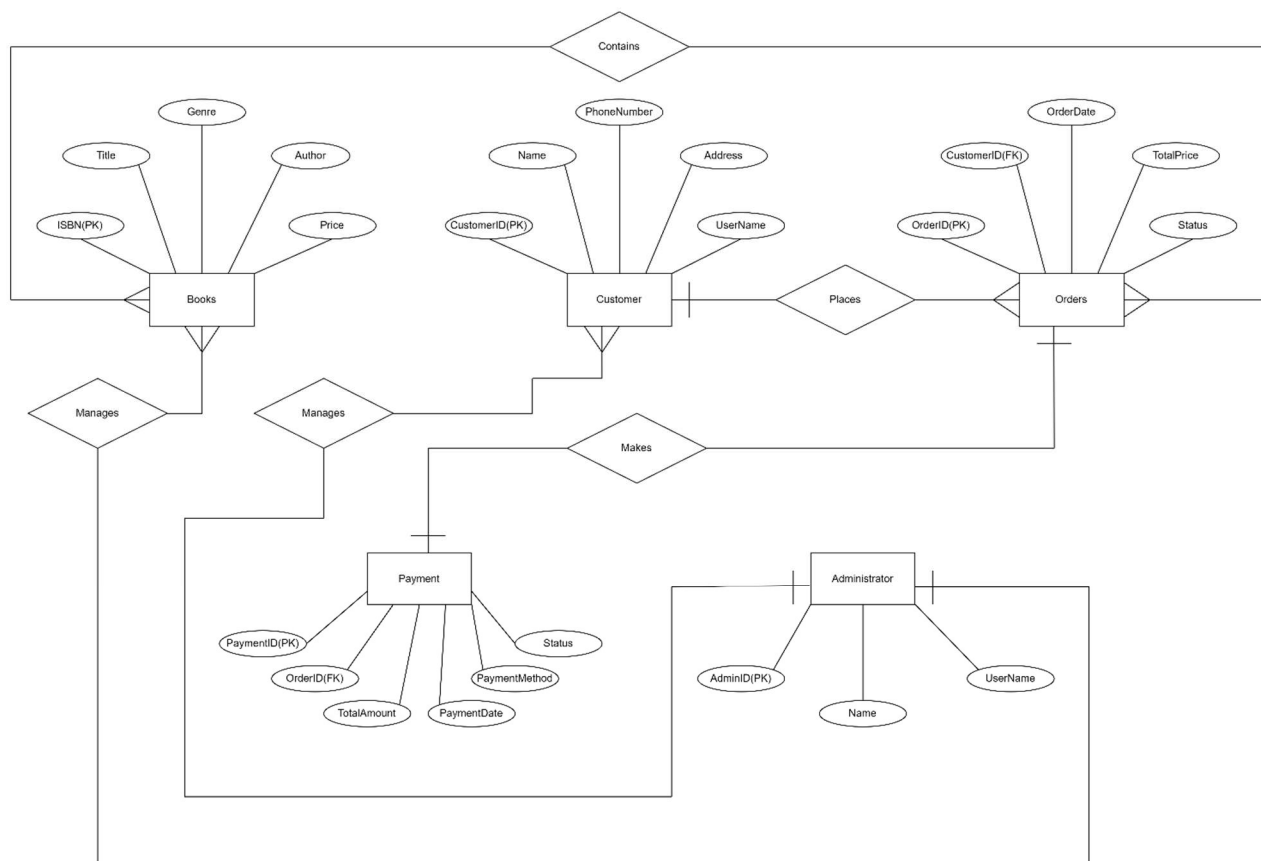**Project Title:** Online Bookstore Management System

Project Background

The Online Bookstore Management System project aims to develop a comprehensive database management solution for efficiently managing the operations of an online bookstore. In today's digital age, online bookstores play a vital role in providing customers with convenient access to a wide range of books from various genres and authors. However, managing an online bookstore effectively requires database systems for handling inventory, customer orders, and administrative tasks.

Project Description

The Online Bookstore Management System is a database management project aimed at designing and implementing a database system for managing an online bookstore's operations solely within a database management system (DBMS) environment. The project will focus on creating Entity-Relationship (ER) model, defining the database schema using Data Definition Language (DDL), and implementing queries to perform various tasks related to managing the bookstore's inventory and customer orders.

ER Diagram

Description of ER Diagram

The Entity-Relationship (ER) diagram for the Online Bookstore Management System depicts the entities, attributes, and relationships involved in the system's database design. The ER diagram provides a visual representation of the conceptual model, illustrating how different entities are related to each other and how they store and interact with data.

Entities:

The above ER Diagram includes the following entities:

Books: Represents individual books available in the bookstore, with attributes such as ISBN, Title, Genre, Author, and Price.

Customer: Represents customers who interact with the bookstore, with attributes including CustomerID, Name, PhoneNumber, Address, and UserName.

Orders: Represents individual orders placed by customers, with attributes such as OrderID, Order Date, Total Price, and Status.

Payment: Represents payments made for orders, with attributes including PaymentID, TotalAmount, PaymentDate, PaymentMethod and Status.

Administrator: Represents administrators who manage the bookstore's operations, with attributes such as AdminID, Name, and UserName.

Relationships:

The ER diagram illustrates the following relationships between entities:

Customer-Order Relationship: Represents the fact that each customer can place multiple orders, and each order is placed by one customer (one-to-many relationship).

Order-Book Relationship: Represents the fact that each order can contain multiple books, and each book can be included in multiple orders (many-to-many relationship).

Payment-Order Relationship: Represents the fact that each payment is associated with one order, and each order can have one payment (one-to-one relationship).

Administrator-Book Relationship: Represents the fact that each administrator can manage multiple books, and each book is managed by one administrator (one-to-many relationship).

Administrator-Customer Relationship: Represents the fact that each administrator can manage multiple customers, and each customer is managed by one administrator (one-to-many relationship).

Attributes and Keys:

Each entity in the ER diagram includes attributes that describe the properties of the entity.

Primary keys are indicated for each entity, denoting the unique identifier for each record in the table.

ER Diagram in the form of Tables
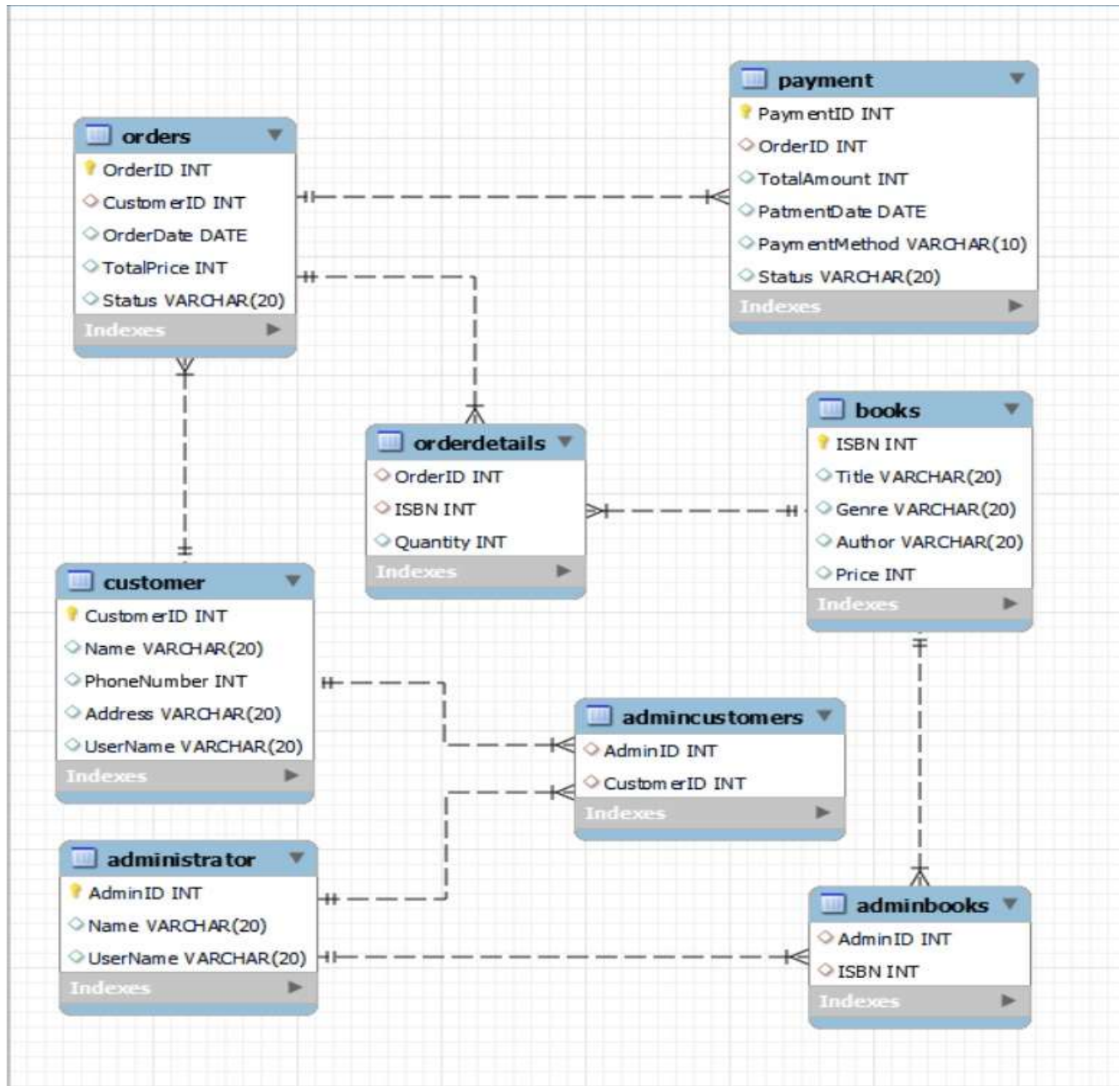
Description of Tables (Relations)

Books: The Books table stores information about individual books available in the bookstore. It includes attributes such as ISBN, Title, Genre, Author, and Price. Each record in the Books table represents a unique book entry in the bookstore's inventory.

Customer: The Customers table stores information about customers who interact with the bookstore. It includes attributes such as CustomerID, Name, PhoneNumber, Address, and UserName. Each record in the Customers table represents a unique customer registered with the bookstore.

Orders: The Orders table stores information about individual orders placed by customers. It includes attributes such as OrderID, Order Date, Total Price, Status and CustomerID (Foreign Key referencing Customer table). Each record in the Orders table represents a unique order placed by a customer.

Payment: The Payment table stores information about payments made for orders. It includes attributes such as PaymentID, TotalAmount, PaymentDate, PaymentMethod, Status and OrderID (Foreign Key referencing Orders table). Each record in the Payment table represents a unique payment transaction associated with an order.

Administrator: The Administrator table stores information about administrators who manage the bookstore's operations. It includes attributes such as AdminID, Name, and UserName. Each record in the Administrator table represents a unique administrator responsible for overseeing bookstore activities.

OrderDetails: The OrderDetails table serves as an intermediate table to represent the many-to-many relationship between orders and books. It stores information about the books included in each order, along with the quantity of each book. It includes attributes such as OrderID (Foreign Key referencing Orders table), ISBN (Foreign Key referencing Books table), and Quantity.

AdminBooks: The AdminBooks table serves as an intermediate table to represent the one-to-many relationship between administrators and books. It stores information about the books managed by each administrator. It includes attributes such as AdminID (Foreign Key referencing Administrator table) and ISBN (Foreign Key referencing Books table).

AdminCustomers: The AdminCustomers table serves as an intermediate table to represent the one-to-many relationship between administrators and customers. It stores information about the customers managed by each administrator. It includes attributes such as AdminID (Foreign Key referencing Administrator table) and CustomerID (Foreign Key referencing Customers table).

Normalization

In database management systems (DBMS), normal forms are a series of guidelines that help to ensure that the design of a database is efficient, organized, and free from data anomalies. There are several levels of normalization, each with its own set of guidelines, known as normal forms they are 1NF, 2NF, 3NF, BCNF, 4NF, and 5NF.

First Normal Form (1NF): This is the most basic level of normalization. In 1NF, each table cell should contain only a single value, and each column should have a unique name. The first normal form helps to eliminate duplicate data and simplify queries.

Second Normal Form (2NF): 2NF eliminates redundant data by requiring that each non-key attribute be dependent on the primary key. This means that each column should be directly related to the primary key, and not to other columns.

Third Normal Form (3NF): 3NF builds on 2NF by requiring that all non-key attributes are independent of each other. This means that each column should be directly related to the primary key, and not to any other columns in the same table.

Boyce-Codd Normal Form (BCNF): BCNF is a stricter form of 3NF that ensures that each determinant in a table is a candidate key. In other words, BCNF ensures that each non-key attribute is dependent only on the candidate key.

Fourth Normal Form (4NF): 4NF is a further refinement of BCNF that ensures that a table does not contain any multi-valued dependencies.

Fifth Normal Form (5NF): 5NF is the highest level of normalization and involves decomposing a table into smaller tables to remove data redundancy and improve data integrity.

Normal forms help to reduce data redundancy, increase data consistency, and improve database performance. However, higher levels of normalization can lead to more complex database designs and queries. It is important to strike a balance between normalization and practicality when designing a database.

Normalization of tables up to 3NF

In our Online Bookstore Management System, the database schema has been structured in such a way that it inherently satisfies the requirements of the Third Normal Form (3NF) without the need for explicit normalization efforts. The initial design of the schema ensures that each table is organized to avoid data redundancy and dependency, thereby preserving data integrity and minimizing anomalies. By acknowledging the principles of 3NF from the outset, our database schema supports efficient data storage, manipulation, and retrieval, contributing to the overall robustness and reliability of the system.

Creation of Data in Tables

Books:

```
create table Books(
ISBN int primary key,
Title varchar(20),
Genre varchar(20),
Author varchar(20),
Price int
);
```

```
insert into Books values(123,"Harry Potter","Fiction","J.K. Rowling",450);
insert into Books values(234,"Make Epic Money","Non-Fiction","Ankur Warikoo",500);
insert into Books values(345,"Wings On Fire","Biography","Arun Tiwari",550);
insert into Books values(456,"Book Of Nature","Poetry","Arjit Trivedi",600);
insert into Books values(567,"The Visitor","Thriller","KL Slater",650);
```

```
select * from Books;
```

Output:

ISBN	Title	Genre	Author	Price
123	Harry Potter	Fiction	J.K. Rowling	450
234	Make Epic Money	Non-Fiction	Ankur Warikoo	500
345	Wings On Fire	Biography	Arun Tiwari	550
456	Book Of Nature	Poetry	Arjit Trivedi	600
567	The Visitor	Thriller	KL Slater	650
NULL	NULL	NULL	NULL	NULL

Customers:

```
create table Customer(
CustomerID int primary key,
Name varchar(20),
PhoneNumber int,
Address varchar(20),
UserName varchar(20)
);

INSERT INTO Customer VALUES(1, 'John Doe', 12345678, '123 Main St',
'johndoe123');
INSERT INTO Customer VALUES(2, 'Jane Smith', 98765432, '456 Elm St',
'janesmith456');
INSERT INTO Customer VALUES(3, 'Michael Johnson', 55512345, '789 Oak St',
'michaelj');
INSERT INTO Customer VALUES(4, 'Emily Davis', 11122233, '321 Pine St', 'emilyd');
INSERT INTO Customer VALUES(5, 'Chris Brown', 99988877, '654 Birch St', 'chrisb');

select * from Customer;
```

Output:

CustomerID	Name	PhoneNumber	Address	UserName
1	John Doe	12345678	123 Main St	johndoe123
2	Jane Smith	98765432	456 Elm St	janesmith456
3	Michael Johnson	55512345	789 Oak St	michaelj
4	Emily Davis	11122233	321 Pine St	emilyd
5	Chris Brown	99988877	654 Birch St	chrisb
NULL	NULL	NULL	NULL	NULL

Orders:

```
create table Orders(
OrderID int primary key,
CustomerID int,
OrderDate date,
TotalPrice int,
Status varchar(20),
foreign key(CustomerID) references Customer(CustomerID)
);

INSERT INTO Orders VALUES(1, 1, '2024-04-23', 100, 'Pending');
INSERT INTO Orders VALUES(2, 2, '2024-04-22', 150, 'Completed');
INSERT INTO Orders VALUES(3, 3, '2024-04-21', 200, 'Pending');
INSERT INTO Orders VALUES(4, 1, '2024-04-20', 120, 'Completed');

select * from Orders;
```

Output:

	OrderID	CustomerID	OrderDate	TotalPrice	Status
►	2	2	2024-04-22	150	Completed
	3	3	2024-04-21	200	Pending
	4	1	2024-04-20	120	Completed
	5	4	2024-04-19	90	Pending
＊	NULL	NULL	NULL	NULL	NULL

Payment:

```sql
create table Payment(
PaymentID int primary key,
OrderID int,
TotalAmount int,
PatmentDate date,
PaymentMethod varchar(10),
Status varchar(20),
foreign key(OrderID) references Orders(OrderID)
);

INSERT INTO Payment VALUES(1, 1, 100, '2024-04-23', 'Credit', 'Completed');
INSERT INTO Payment VALUES(2, 2, 150, '2024-04-22', 'Debit', 'Completed');
INSERT INTO Payment VALUES(3, 3, 200, '2024-04-21', 'Cash', 'Pending');
INSERT INTO Payment VALUES(4, 4, 120, '2024-04-20', 'Credit', 'Completed');
INSERT INTO Payment VALUES(5, 5, 90, '2024-04-19', 'Debit', 'Pending');

select * from Payment;
```

Output:

PaymentID	OrderID	TotalAmount	PatmentDate	PaymentMethod	Status
2	2	150	2024-04-22	Debit	Completed
3	3	200	2024-04-21	Cash	Pending
4	4	120	2024-04-20	Credit	Completed
5	5	90	2024-04-19	Debit	Pending
NULL	NULL	NULL	NULL	NULL	NULL

Administrator:

```
create table Administrator(
AdminID int primary key,
Name varchar(20),
UserName varchar(20)
);

INSERT INTO Administrator VALUES(1, 'John Doe', 'johndoe_admin');
INSERT INTO Administrator VALUES(2, 'Jane Smith', 'janesmith_admin');
INSERT INTO Administrator VALUES(3, 'Michael Johnson', 'michaelj_admin');
INSERT INTO Administrator VALUES(4, 'Emily Davis', 'emilyd_admin');
INSERT INTO Administrator VALUES(5, 'David Brown', 'davidb_admin');

select * from Administrator;
```

Output:

AdminID	Name	UserName
1	John Doe	johndoe_admin
2	Jane Smith	janesmith_admin
3	Michael Johnson	michaelj_admin
4	Emily Davis	emilyd_admin
5	David Brown	davidb_admin
NULL	NULL	NULL

OrderDetails:

```
create table OrderDetails(
OrderID int,
ISBN int,
Quantity int,
foreign key(OrderID) references Orders(OrderID),
foreign key(ISBN) references Books(ISBN)
);
```

AdminBooks:

create table AdminBooks(

AdminID int,

ISBN int,

foreign key(AdminID) references Administrator(AdminID),

foreign key(ISBN) references Books(ISBN)

);

AdminCustomers:

create table AdminCustomers(

AdminID int,

CustomerID int,

foreign key(AdminID) references Administrator(AdminID),

foreign key(CustomerID) references Customer(CustomerID)

);

SQL Queries on the created tables

Problem Statement:

Write a Query to Update the Genre as "Auto Biography" where ISBN is 345.

Query:

update Book set Genre="Auto Biography" where ISBN=345;

Output:

	ISBN	Title	Genre	Author	Price
▶	123	Harry Potter	Fiction	J.K. Rowling	450
	234	Make Epic Money	Non-Fiction	Ankur Warikoo	500
	345	Wings On Fire	Auto Biography	Arun Tiwari	550
	456	Book Of Nature	Poetry	Arjit Trivedi	600
	567	The Visitor	Thriller	KL Slater	650
*	NULL	NULL	NULL	NULL	NULL

Problem Statement:

Write a query to delete data from the table where ISBN is 123 or Genre is "Thriller"

Query:

delete from Book where ISBN=123 or Genre = "Thriller";

Output:

ISBN	Title	Genre	Author	Price
234	Make Epic Money	Non-Fiction	Ankur Warikoo	500
345	Wings On Fire	Auto Biography	Arun Tiwari	550
456	Book Of Nature	Poetry	Arjit Trivedi	600
NULL	NULL	NULL	NULL	NULL

Problem Statement:

Write a query to insert a new data row in the table

Where ISBN is 789,

Title is "The Destiny",

Genre is "Non-Fiction",

Author is "DK Gupta",

and Price is 700.

Query:

insert into Book values(789,"The Destiny","Non-Fiction","DK Gupta",700);

Output:

ISBN	Title	Genre	Author	Price
234	Make Epic Money	Non-Fiction	Ankur Warikoo	500
345	Wings On Fire	Auto Biography	Arun Tiwari	550
456	Book Of Nature	Poetry	Arjit Trivedi	600
789	The Destiny	Non-Fiction	DK Gupta	700
NULL	NULL	NULL	NULL	NULL

14

Problem Statement:

Write a query to add a new column as "Year" of int datatype to the table.

Query:

alter table Book add Year int;

Output:

ISBN	Title	Genre	Author	Price	Year
234	Make Epic Money	Non-Fiction	Ankur Warikoo	500	NULL
345	Wings On Fire	Auto Biography	Arun Tiwari	550	NULL
456	Book Of Nature	Poetry	Arjit Trivedi	600	NULL
789	The Destiny	Non-Fiction	DK Gupta	700	NULL
NULL	NULL	NULL	NULL	NULL	NULL

Problem Statement:

Write a query to Display the sum of prices in the table as "TotalCost".

Query:

select sum(Price) as TotalCost from Books;

Output:

TotalCost
2350

Creation of Views using Tables

Problem Statement:
Write a Query to create a View named as BooksView.

Query:

CREATE VIEW BooksView AS

SELECT ISBN, Title, Genre, Author, Price

FROM Books;

SELECT * FROM BooksView;

Output:

	ISBN	Title	Genre	Author	Price
▶	123	Harry Potter	Fiction	J.K. Rowling	450
	234	Make Epic Money	Non-Fiction	Ankur Warikoo	500
	345	Wings On Fire	Biography	Arun Tiwari	550
	456	Book Of Nature	Poetry	Arjit Trivedi	600
	567	The Visitor	Thriller	KL Slater	650

Problem Statement:
Write a Query to create a View named as CustomersOrdersView.

Query:

CREATE VIEW CustomerOrdersView AS

SELECT o.OrderID, o.CustomerID, c.Name AS CustomerName, o.OrderDate, o.TotalPrice, o.Status

FROM Orders o

JOIN Customer c ON o.CustomerID = c.CustomerID;

SELECT * FROM CustomerOrdersView;

Output:

	OrderID	CustomerID	CustomerName	OrderDate	TotalPrice	Status
▶	4	1	John Doe	2024-04-20	120	Completed
	2	2	Jane Smith	2024-04-22	150	Completed
	3	3	Michael Johnson	2024-04-21	200	Pending
	5	4	Emily Davis	2024-04-19	90	Pending

Problem Statement:
Write a Query to create a View named as PaymentDetailsView.

Query:

CREATE VIEW PaymentDetailsView AS

SELECT p.PaymentID, p.OrderID, o.CustomerID, c.Name AS CustomerName,
p.TotalAmount, p.PatmentDate, p.PaymentMethod, p.Status

FROM Payment p

JOIN Orders o ON p.OrderID = o.OrderID

JOIN Customer c ON o.CustomerID = c.CustomerID;

select * from PaymentDetailsView;

Output:

PaymentID	OrderID	CustomerID	CustomerName	TotalAmount	PatmentDate	PaymentMethod	Status
2	2	2	Jane Smith	150	2024-04-22	Debit	Completed
3	3	3	Michael Johnson	200	2024-04-21	Cash	Pending
4	4	1	John Doe	120	2024-04-20	Credit	Completed
5	5	4	Emily Davis	90	2024-04-19	Debit	Pending

Problem Statement:
Write a Query to create a View named as AdministratorBooksView.

Query:

CREATE VIEW AdministratorBooksView AS

SELECT ab.AdminID, a.Name AS AdminName, ab.ISBN, b.Title, b.Author, b.Price

FROM AdminBooks ab

JOIN Administrator a ON ab.AdminID = a.AdminID

LEFT JOIN Books b ON ab.ISBN = b.ISBN;

select * from AdministratorBooksView;

Output:

AdminID	AdminName	ISBN	Title	Author	Price
1	John Doe	345	Wings On Fire	Arun Tiwari	550
2	Jane Smith	456	Book Of Nature	Arjit Trivedi	600

Problem Statement:

Write a Query to create a View named as AdminCustomersView.

Query:

CREATE VIEW AdminCustomersView AS

SELECT ac.AdminID, a.Name AS AdminName, c.CustomerID, c.Name AS
CustomerName, c.PhoneNumber, c.Address, c.UserName

FROM AdminCustomers ac

JOIN Administrator a ON ac.AdminID = a.AdminID

JOIN Customer c ON ac.CustomerID = c.CustomerID;

Output:

AdminID	AdminName	CustomerID	CustomerName	PhoneNumber	Address	UserName
1	John Doe	1	John Doe	12345678	123 Main St	johndoe123
1	John Doe	2	Jane Smith	98765432	456 Elm St	janesmith456
2	Jane Smith	3	Michael Johnson	55512345	789 Oak St	michaelj
2	Jane Smith	4	Emily Davis	11122233	321 Pine St	emilyd
2	Jane Smith	5	Chris Brown	99988877	654 Birch St	chrisb

----- Thank You -----