AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

Summer 2024-25

Mid Term Project Report

BOOK STORE MANAGEMENT SYSTEM

NAME	ID	CONTRIBUTION	TOPIC
Mohammad Rafsan Yeasir	22-46065-1	30	UI Implementation, ER Diagram, Normalization.
MD.Fardin	21-45603-3	20	Introduction, SQL Query
Asikqur Rahman	22-48971-3	20	Scenario,Schema Diagram, Contents, Merging.
Md. Asaduzzaman Rony	22-49057-3	30	Project Proposal, Table Creation, Table contents Insertion

PROJECT REPORT

SECTION : A

Contents:

Bookshop Management System 4
1. Project Title 4
2. Introduction 4
3. Project Proposal 4
4. User Interface Planning 8
5. Scenerio Description 19
6. ER Diagram 20
7. Normalization 21
8. Schema Diagram 27
9. Table Creation 28
10. Data Insertion 36
11. Sql Query 47

Introduction:

Bookstores today must manage a wide variety of operations, including maintaining book inventories, handling supplier relationships, processing customer orders, managing employees, and recording payments. Manual methods or fragmented systems often cause stock inaccuracies, delays in order fulfillment, and difficulties in tracking financial transactions. These inefficiencies reduce both operational performance and customer satisfaction.

The Bookstore Management System addresses these challenges by integrating all core entities—Books, Categories, Publishers, Suppliers, Customers, Employees, Orders, and Payments—into a centralized database-driven platform. The system automates book cataloging, supplier coordination, sales processing, and payment management while ensuring real-time updates on inventory and order status. It also streamlines employee tasks and generates valuable reports for managers. By digitizing and unifying these operations, the system enhances accuracy, improves service quality, and provides greater reliability for decision-making and overall bookstore management.

Project Proposal:

Bookshop Management System

1. Project Title

Bookshop Management System

2. Introduction

Managing a bookshop involves handling various operations such as maintaining book records, tracking inventory, managing customers, recording sales, and processing payments. Manual handling of these operations often leads to inefficiencies, errors, and data redundancy.

This project proposes the development of a Bookshop Management System that utilizes an advanced relational database to automate and streamline bookshop operations. The system will focus on data integrity, efficiency, and reporting using advanced database concepts like triggers, stored procedures, transactions, indexing, and views.

3. Objectives

- To design and implement a normalized database for managing bookshop operations.
- To enable efficient management of books, categories, publishers, suppliers, employees, customers, and orders.
- To automate processes such as stock updates after sales, payment handling, and report generation.
- To ensure data accuracy and consistency using constraints and relationships.
- To generate useful reports (sales analysis, stock status, customer history).

4. Database Design Overview (Tables & Relationships)

Main Entities:

1. Books – Stores details of each book including title, author, category, publisher, supplier, price, and stock.
2. Category – Categorizes books (e.g., Fiction, Science, Technology).
3. Publisher – Maintains publisher information (name, address, contact).
4. Supplier – Tracks book suppliers and their details.
5. Customer – Holds customer records for purchases and order tracking.
6. Employee – Stores employee details including roles.
7. Orders – Manages sales transactions, linking customers and employees.
8. Payment – Records payment details for each order.
9. OrderItem – Calculates individual book quantity and subtotal for orders.

Key Relationships:

- A Book belongs to one Category, one Publisher, and one Supplier.
- A Customer can place many Orders.
- An Employee processes many Orders.
- Each Order has exactly one Payment.

5. Advanced Database Features to be Used

- Constraints (PK, FK, Unique, Check).
- Triggers for stock update and logging.
- Stored Procedures & Functions for order processing and calculations.
- Views for reports and purchase history.
- Transactions ensuring ACID properties.
- Indexing for faster search.

6. Methodology

1. Requirement Analysis – Finalize system requirements and business rules.
2. Database Design – Develop ER Diagram and relational schema.
3. Implementation – Create normalized tables with constraints.
4. Integration – Implement stored procedures, triggers, views.
5. Testing – Validate constraints, run test queries, simulate sales.

6. Documentation – Write final report with schema, SQL code, and screenshots.

7. Scope of the Project (Modules)

- Book Management
- Category & Publisher Management
- Supplier Management
- Customer & Employee Management
- Order & Payment Management
- Reporting & Analytics

8. Expected Outcomes

- A fully functional Bookshop Management Database with advanced features.
- Automatic stock management after sales.
- Real-time insights into sales trends, customer behavior, and inventory levels.
- Efficient and secure data handling.

9. Tools & Technologies

- Database: Oracle / MySQL / SQL Server
- Design Tools: Draw.io / Lucidchart
- Frontend (optional): C#, Java, or Python
- SQL Features: Triggers, Stored Procedures, Views, Transactions

10. Project Timeline

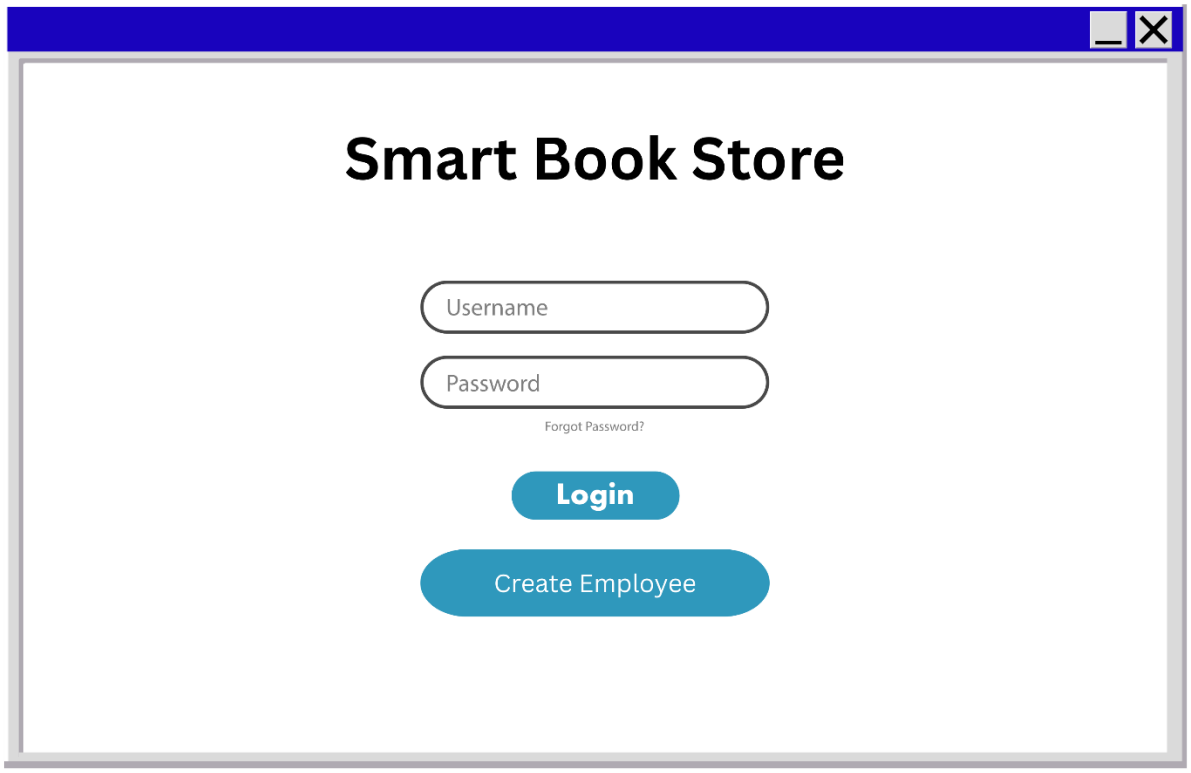
Requirement Analysis – 1 week
Database Design – 2 weeks
Implementation – 3 weeks
Advanced Features Setup – 2 weeks
Testing & Debugging – 1 week
Documentation & Demo – 1 week

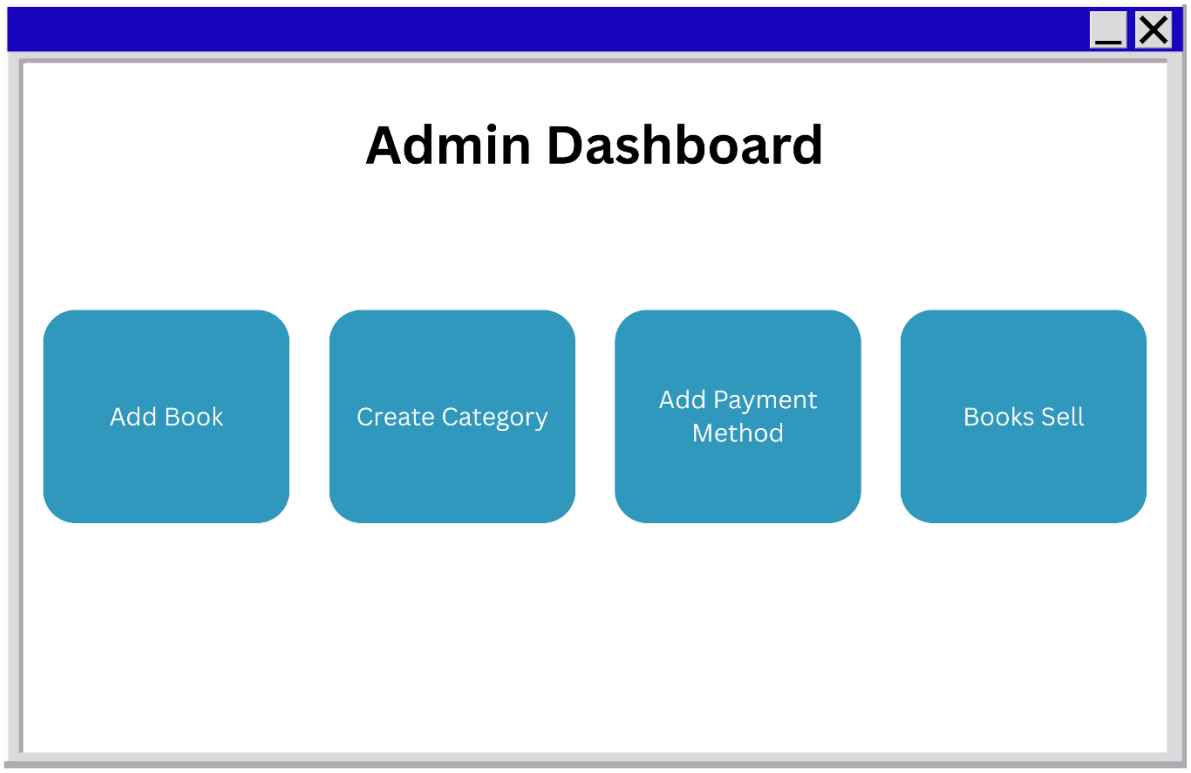
11. Conclusion

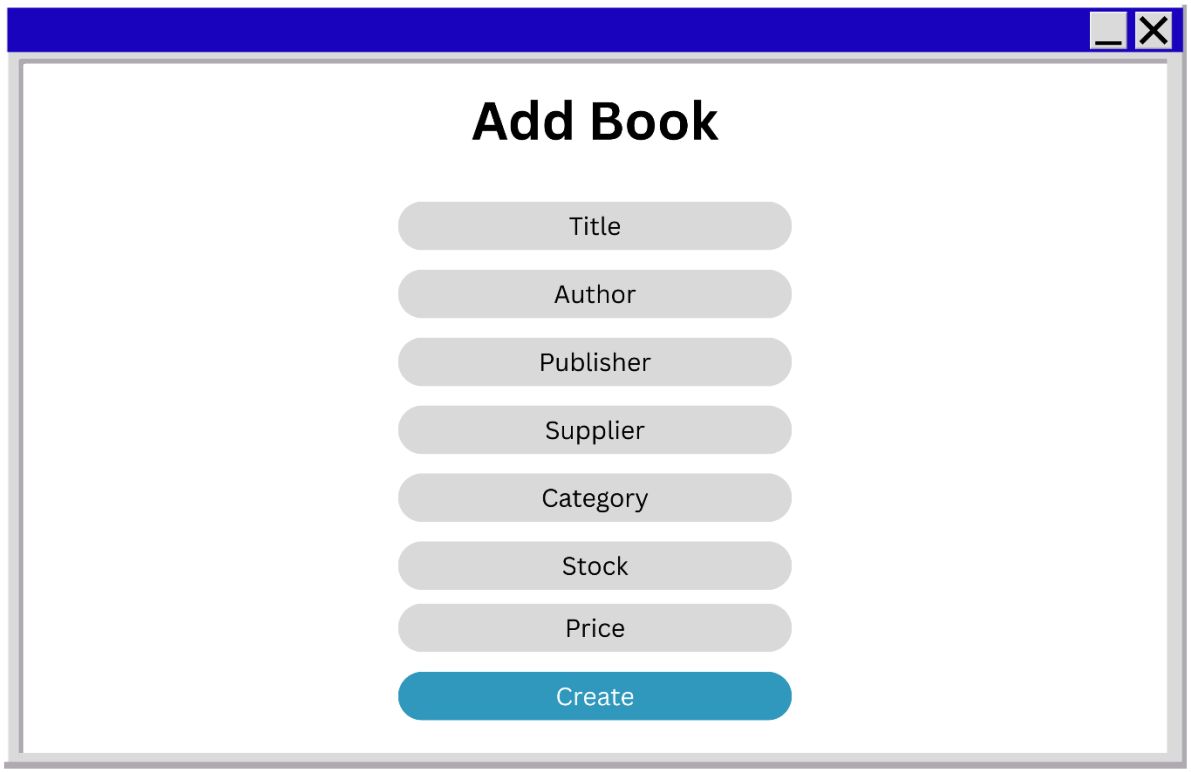
The proposed Bookshop Management System will simplify bookshop operations by automating book, order, customer, and payment management.

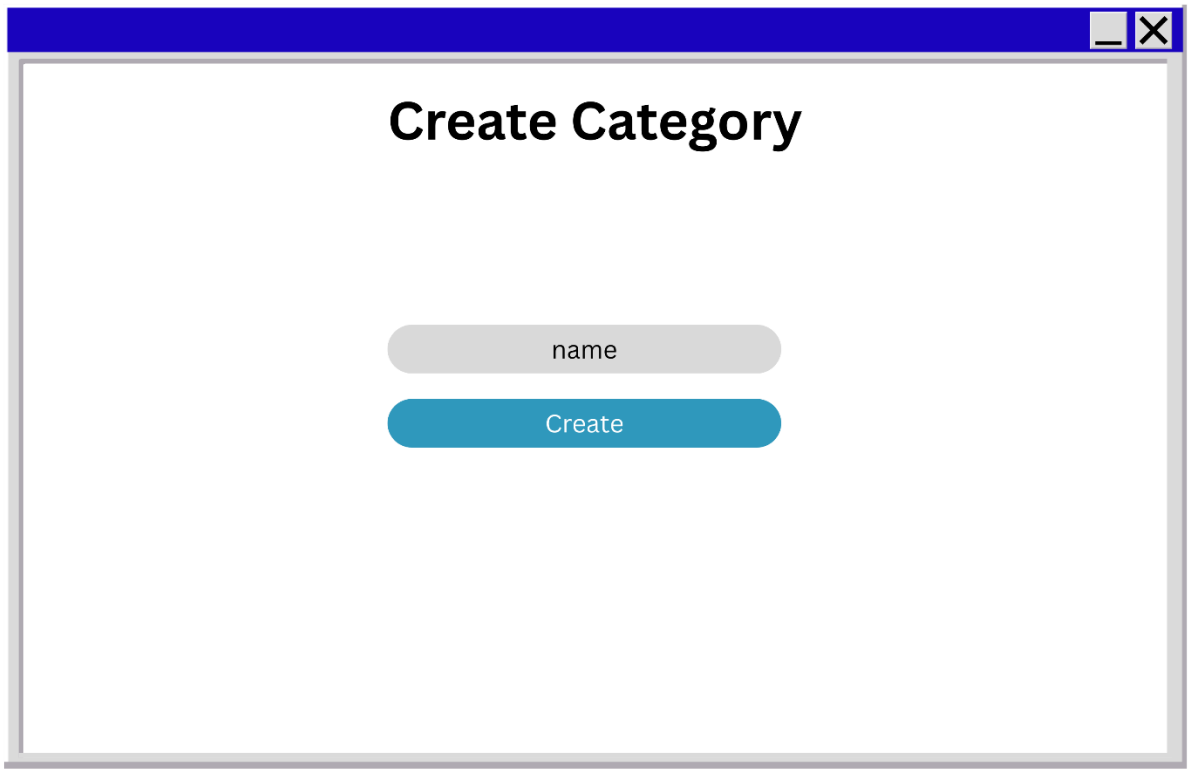
The project will apply advanced database techniques to ensure data integrity, security, and efficiency, while also supporting decision-making with detailed reports.

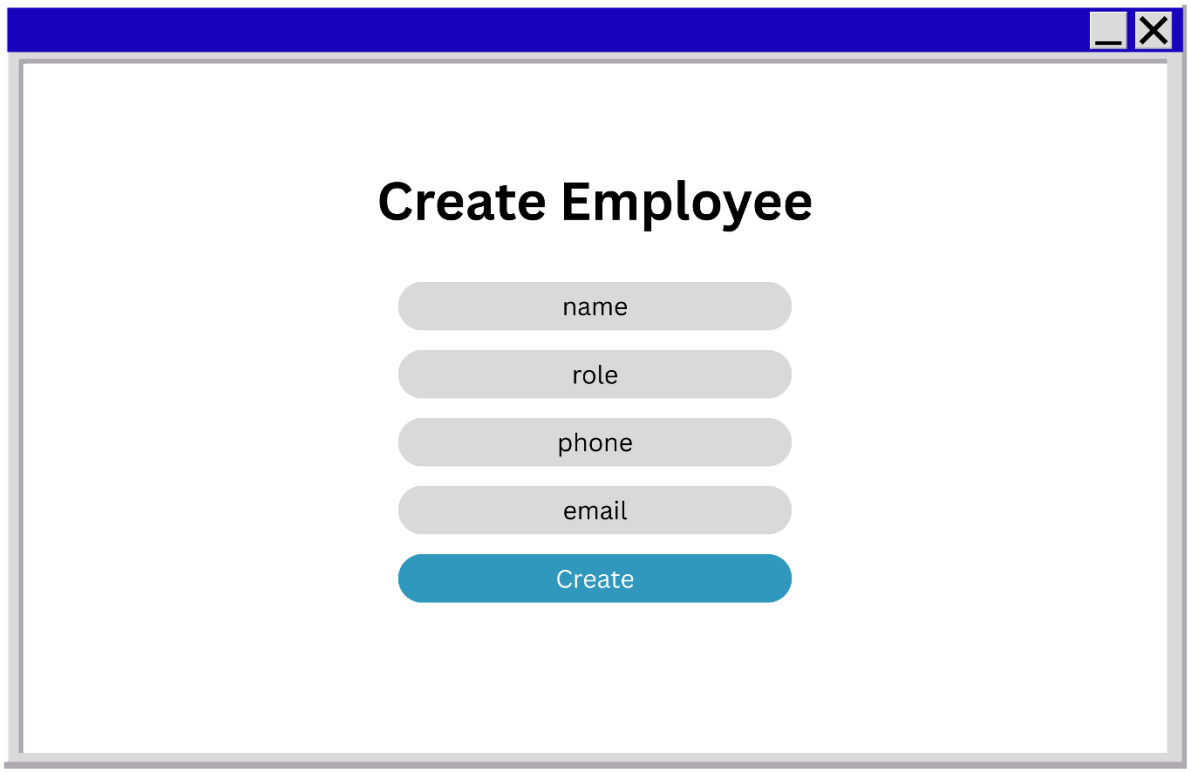
User Interface Planning:

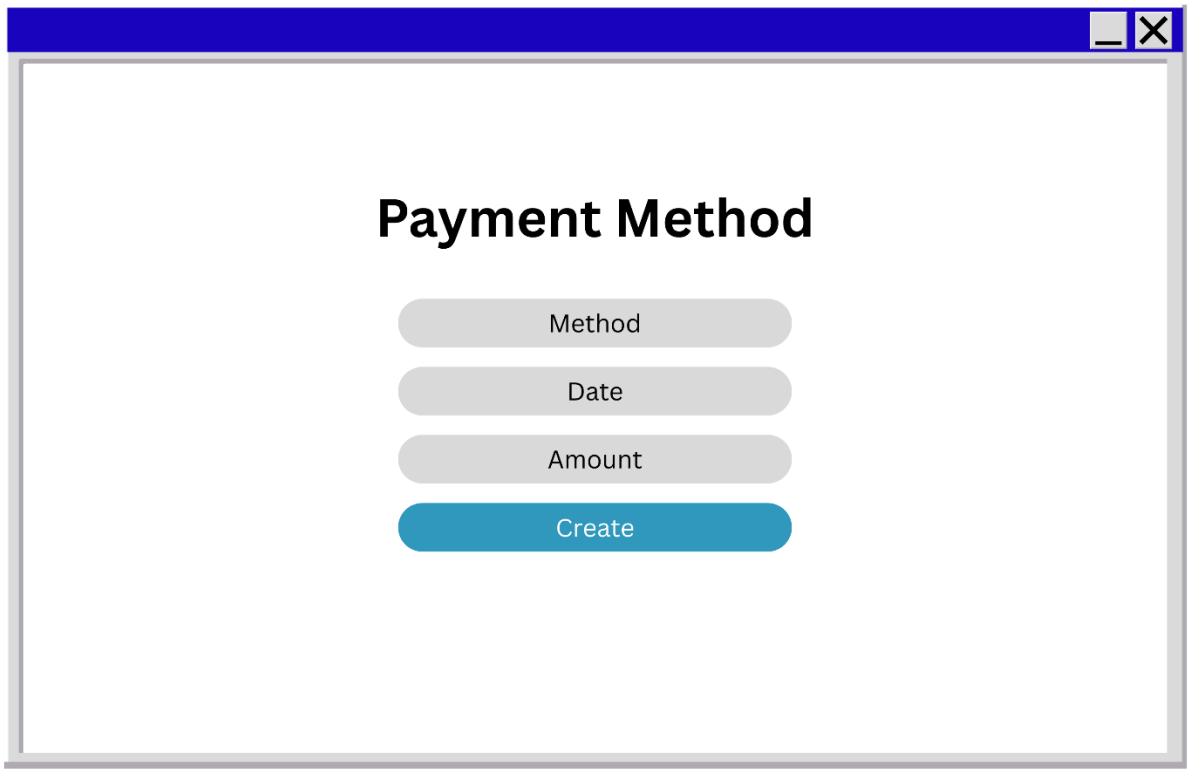

Smart Book Store
Username
Password
Forgot Password?
Login
Create Employee

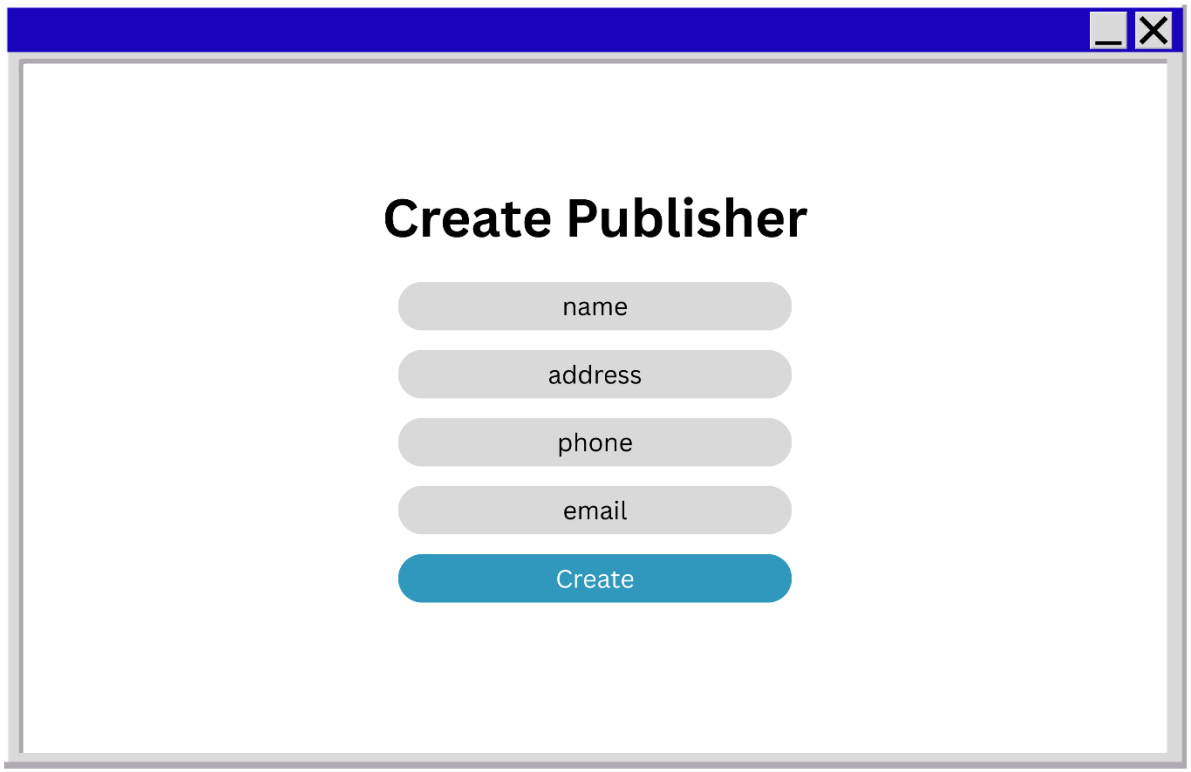

Admin Dashboard
Add Book
Create Category
Add Payment Method
Books Sell

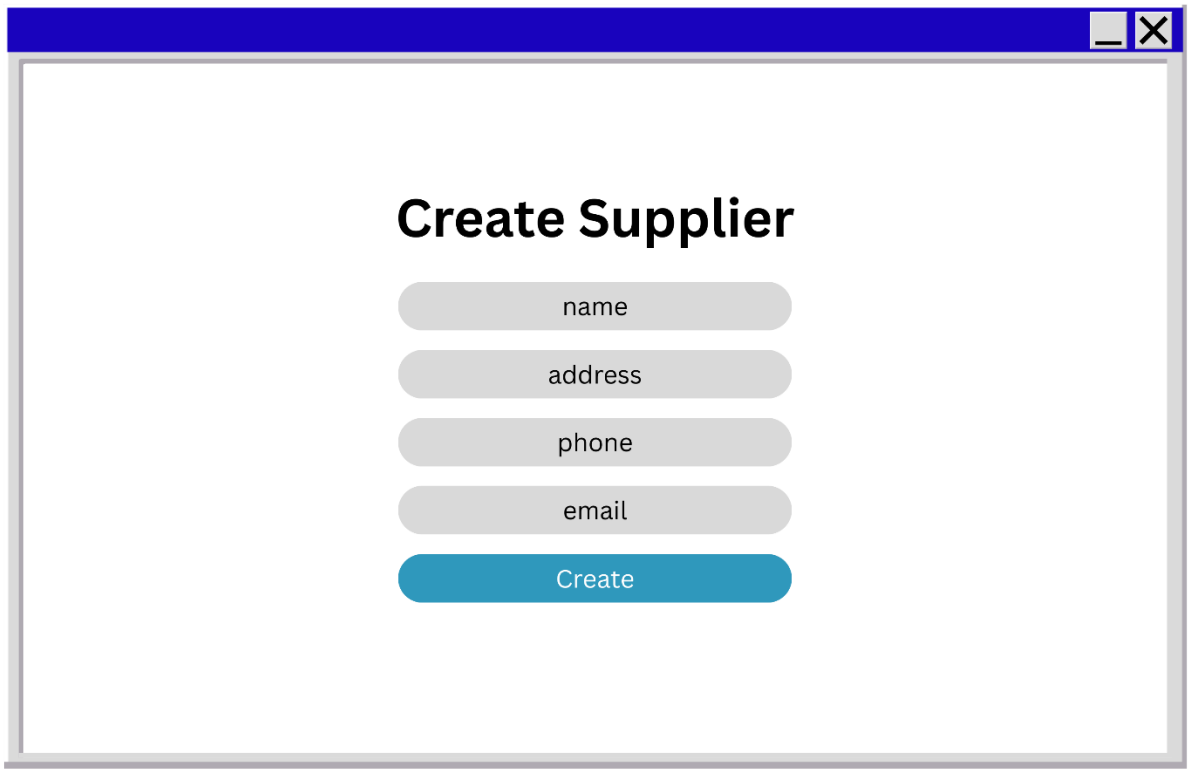

Add Book
Title
Author
Publisher
Supplier
Category
Stock
Price
Create


Create Category
name
Create

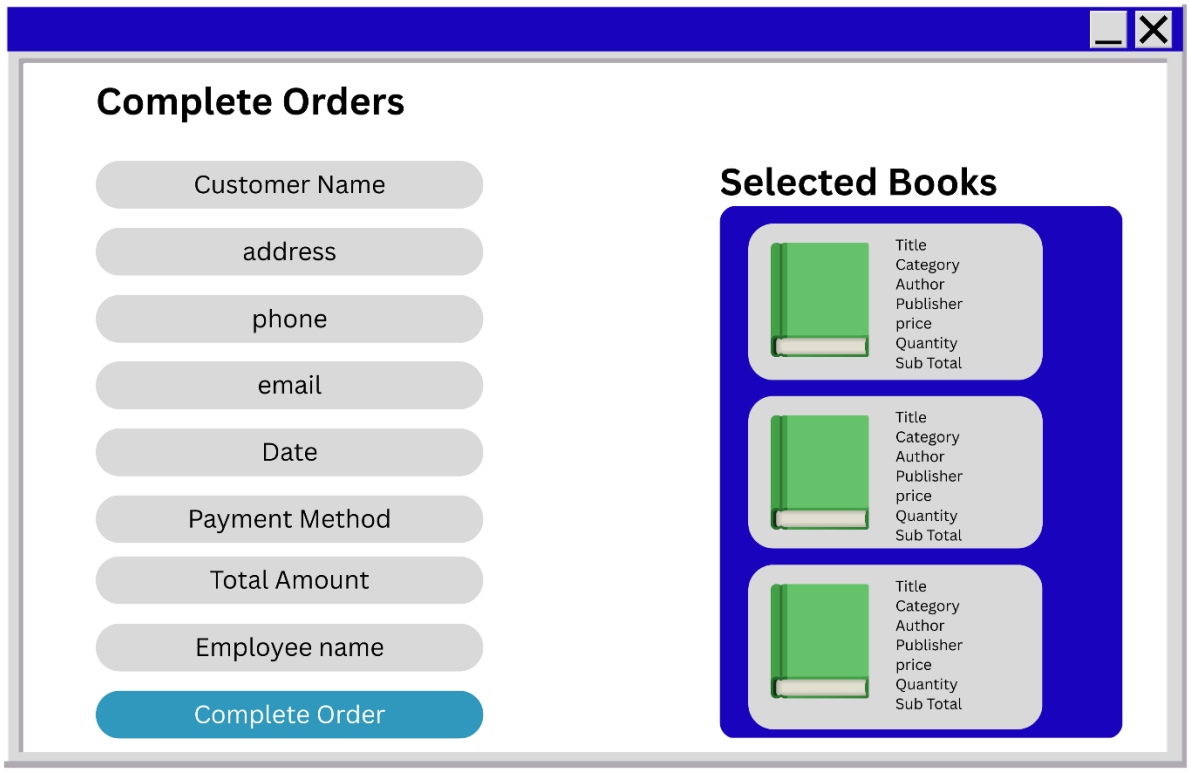

Create Employee
name
role
phone
email
Create


Payment Method
Method
Date
Amount
Create


Create Publisher
name
address
phone
email
Create


Create Supplier
name
address
phone
email
Create

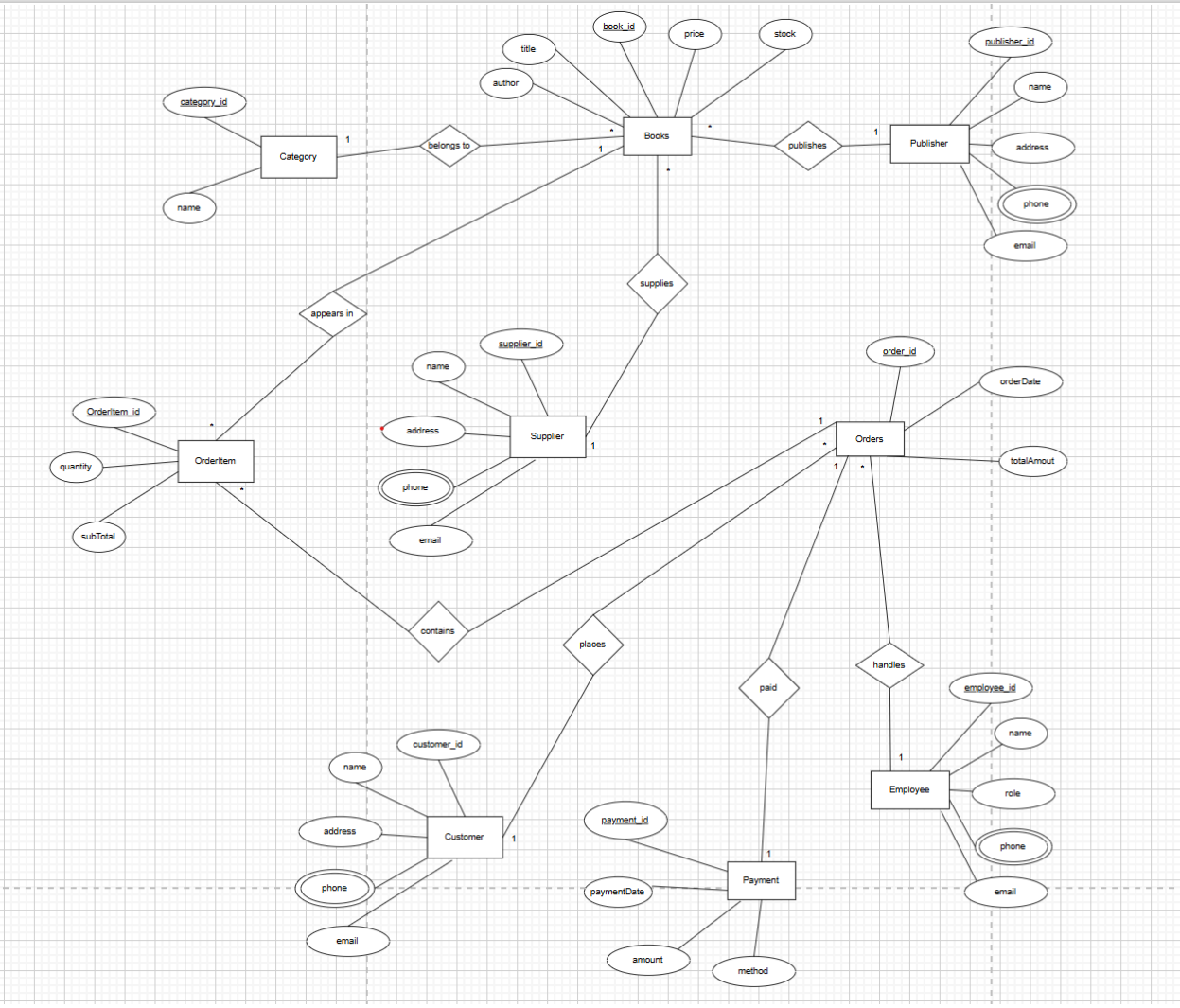

Select Books to Buy
Title
Category
Author
Publisher
price
Add to cart
Title
Category
Author
Publisher
price
Add to cart
Title
Category
Author
Publisher
price
Add to cart
Title
Category
Author
Publisher
price
Add to cart


Complete Orders
Customer Name
address
phone
email
Date
Payment Method
Total Amount
Employee name
Complete Order
Selected Books
Title
Category
Author
Publisher
price
Quantity
Sub Total
Title
Category
Author
Publisher
price
Quantity
Sub Total
Title
Category
Author
Publisher
price
Quantity
Sub Total

Scenario:

In a bookstore management system, books are stored with details such as title, author, price, stock, and are linked to their respective categories, publishers, and suppliers. Categories help organize books by subject, while publishers provide information like name, contact, and address. Suppliers record details of external parties from whom books are purchased, ensuring stock availability.Customers are registered with their name, email, phone, and address so they can place orders, which are processed by employees responsible for handling sales. Each order records the customer, the employee managing it, the order date, and total amount. To capture the details of which books are included in an order, an OrderItem entity is introduced. Each OrderItem links an order to a book and includes details such as quantity and subtotal. An order may have many OrderItems, but each OrderItem belongs to exactly one order and refers to exactly one book. Once an order is placed, payment information is captured, including payment method, amount, and date, linked uniquely to the order. This setup allows the bookstore to manage books, track suppliers and publishers, organize sales handled by employees, serve customers effectively, and maintain accurate records of payments, transactions, and order details.

ER Diagram:


book_id
price
stock
title
author
publisher_id
name
category_id
Category
belongs to
Books
publishes
Publisher
address
phone
email
name
supplies
appears in
supplier_id
name
order_id
orderDate
OrderItem_id
address
Supplier
Orders
quantity
OrderItem
totalAmout
phone
subTotal
email
contains
places
paid
handles
employee_id
customer_id
name
name
role
Employee
address
Customer
payment_id
phone
phone
Payment
email
paymentDate
email
amount
method

Normalization:

Belongs to:

UNF: BookID, Title, Author, Price, Stock, CategoryID, CategoryName

1NF: BookID, Title, Author, Price, Stock, CategoryID, PublisherID, SupplierID, CategoryName

2NF:
i) BookID, Title, Author, Price, Stock
ii) CategoryID, CategoryName

3NF:
i) BookID, Title, Author, Price, Stock
ii) CategoryID, CategoryName

Table Creation:

i) Books(BookID, Title, Author, Price, Stock)
ii) Category(CategoryID, Name)

Publishes:

UNF: BookID, Title, Author, Price, Stock, PublisherID, PublisherName, Address, Phone, Email

1NF: BookID, Title, Author, Price, Stock, PublisherID, PublisherName, Address, Phone, Email

2NF:
i) BookID, Title, Author, Price, Stock
ii) PublisherID, PublisherName, Address, Phone, Email

3NF:
i) BookID, Title, Author, Price, Stock
ii) PublisherID, PublisherName, Address, Phone, Email

Table Creation:

i)Books(BookID, Title, Author, Price, Stock)

ii) Publisher(PublisherID, PublisherName, Address, Phone, Email)

Supplies:

UNF: BookID, Title, Author, Price, Stock, SupplierID, Name, ContactPerson, Phone, Email, Address

1NF: BookID, Title, Author, Price, Stock, SupplierID, Name, ContactPerson, Phone, Email, Address

2NF:

i) BookID, Title, Author, Price, Stock

ii) SupplierID, Name, ContactPerson, Phone, Email, Address

3NF:

i) BookID, Title, Author, Price, Stock

ii) SupplierID, Name, ContactPerson, Phone, Email, Address

Table Creation:

i) Books(BookID, Title, Author, Price, Stock)

ii) Supplier(SupplierID, Name, ContactPerson, Phone, Email, Address)

Appears in:

UNF: BookID, Title, Author, Price, Stock, OrderItemID, Quantity, SubTotal

1NF: BookID, Title, Author, Price, Stock, OrderItemID, Quantity, SubTotal

2NF:

i) BookID, Title, Author, Price, Stock

ii) OrderItemID, Quantity, SubTotal

3NF:

i) BookID, Title, Author, Price, Stock

ii) OrderItemID, Quantity, SubTotal

Table Creation:

i) Books(BookID, Title, Author, Price, Stock)

ii) OrderItem(OrderItemID, Quantity, SubTotal)

Contains:

UNF: OrderID, OrderDate, TotalAmount, OrderItemID, Quantity, SubTotal

1NF: OrderID, OrderDate, TotalAmount, OrderItemID, Quantity, SubTotal

2NF:

i) OrderID, OrderDate, TotalAmount

ii) OrderItemID, Quantity, SubTotal

3NF:

i) OrderID, OrderDate, TotalAmount

ii) OrderItemID, Quantity, SubTotal

Table Creation:

i) Orders(OrderID, OrderDate, TotalAmount)

ii) OrderItem(OrderItemID, Quantity, SubTotal)

Handles:

UNF: OrderID, OrderDate, TotalAmount, EmployeeID, Name, Role, Email, Phone

1NF: OrderID, OrderDate, TotalAmount, EmployeeID, Name, Role, Email, Phone

2NF:

i) OrderID, OrderDate, TotalAmount

ii) EmployeeID, Name, Role, Email, Phone

3NF:

i) OrderID, OrderDate, TotalAmount

ii) EmployeeID, Name, Role, Email, Phone

Table Creation:

i) Orders(OrderID, OrderDate, TotalAmount)

ii) Employees(EmployeeID, Name, Role, Email, Phone)

Places:

UNF: OrderID, OrderDate, TotalAmount, CustomerID, Name, Email, Phone, Address

1NF: OrderID, OrderDate, TotalAmount, CustomerID, Name, Email, Phone, Address

2NF:

i) OrderID, OrderDate, TotalAmount

ii) CustomerID, Name, Email, Phone, Address

3NF:

i) OrderID, OrderDate, TotalAmount

ii) CustomerID, Name, Email, Phone, Address

Table Creation:

i) Orders(OrderID, OrderDate, TotalAmount)

ii) Customer(CustomerID, Name, Email, Phone, Address)

Paid:

UNF: OrderID, OrderDate, TotalAmount, PaymentID, Amount, PaymentDate, Method

1NF: OrderID, OrderDate, TotalAmount, PaymentID, Amount, PaymentDate, Method

2NF:

i) OrderID, OrderDate, TotalAmount

ii) PaymentID, Amount, PaymentDate, Method

3NF:

i) OrderID, OrderDate, TotalAmount

ii) PaymentID, Amount, PaymentDate, Method

Table Creation:

i) Orders(OrderID, OrderDate, TotalAmount)

ii) Payment(PaymentID, Amount, PaymentDate, Method)

General table:

1. Books(BookID, Title, Author, Price, Stock)
2. Category(CategoryID, Name)
3. ~~Books(BookID, Title, Author, Price, Stock)~~

4. Publisher(PublisherID, PublisherName, Address, Phone, Email)
5. ~~Books(BookID, Title, Author, Price, Stock)~~
6. Supplier(SupplierID, Name, ContactPerson, Phone, Email, Address)
7. ~~Books(BookID, Title, Author, Price, Stock)~~
8. OrderItem(OrderItemID, Quantity, SubTotal)
9. Orders(OrderID, OrderDate, TotalAmount)
10. Payment(PaymentID, Amount, PaymentDate, Method)
11. ~~Orders(OrderID, OrderDate, TotalAmount)~~
12. Customer(CustomerID, Name, Email, Phone, Address)
13. ~~Orders(OrderID, OrderDate, TotalAmount)~~
14. Employees(EmployeeID, Name, Role, Email, Phone)
15. ~~Orders(OrderID, OrderDate, TotalAmount)~~
16. ~~OrderItem(OrderItemID, Quantity, SubTotal)~~

Final table:

1. **Books**(BookID, Title, Author, Price, Stock, CategoryID **FK**, PublisherID **FK**, SupplierID **FK**)
2. **Category**(CategoryID, Name)
3. **Publisher**(PublisherID, PublisherName, Address, Phone, Email)
4. **Supplier**(SupplierID, Name, ContactPerson, Phone, Email, Address)
5. **OrderItem**(OrderItemID, OrderID **FK**, BookID **FK**, Quantity, SubTotal)
6. **Orders**(OrderID, OrderDate, TotalAmount, CustomerID **FK**, EmployeeID **FK**)
7. **Payment**(PaymentID, OrderID **FK**, Amount, PaymentDate, Method)
8. **Customer**(CustomerID, Name, Email, Phone, Address)
9. **Employees**(EmployeeID, Name, Role, Email, Phone)

Schema Diagram:

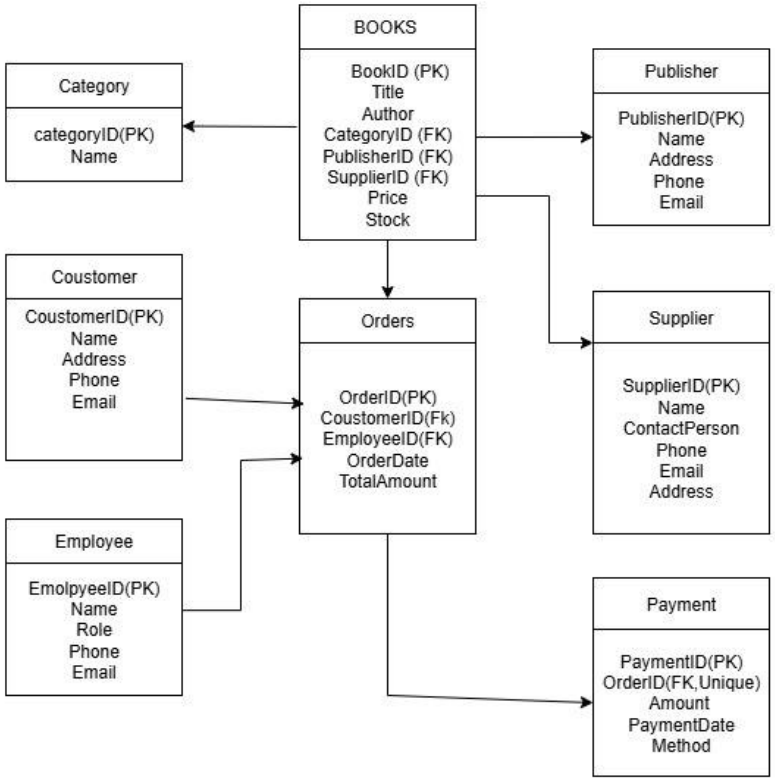
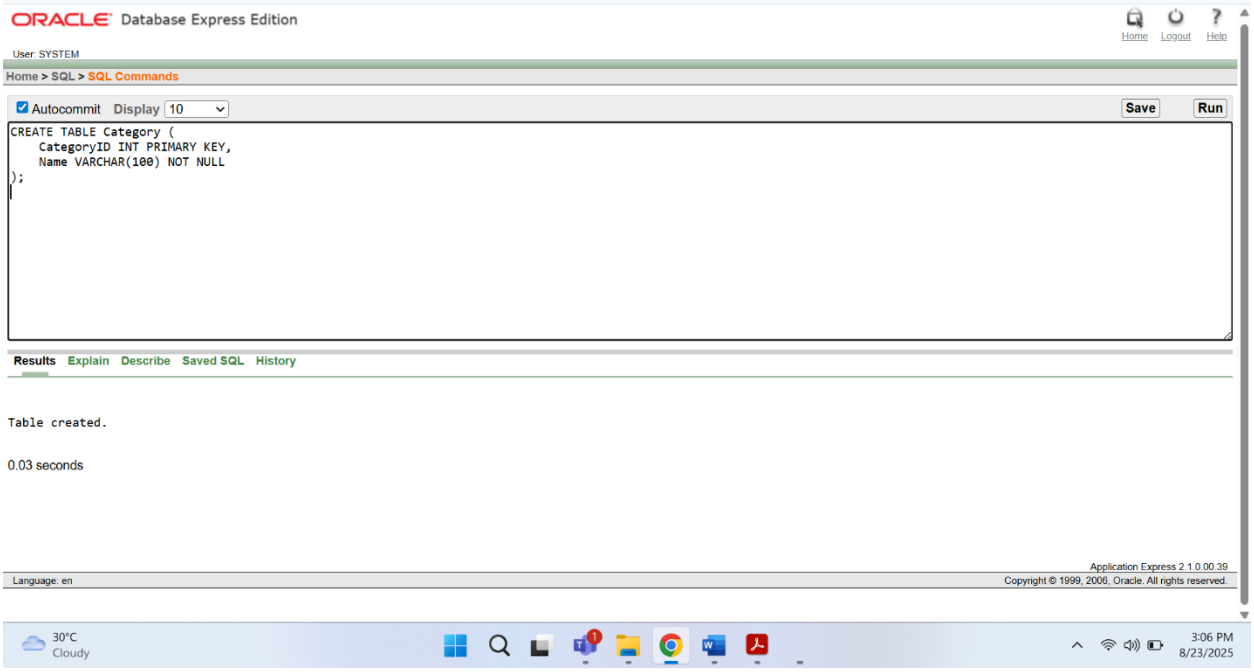
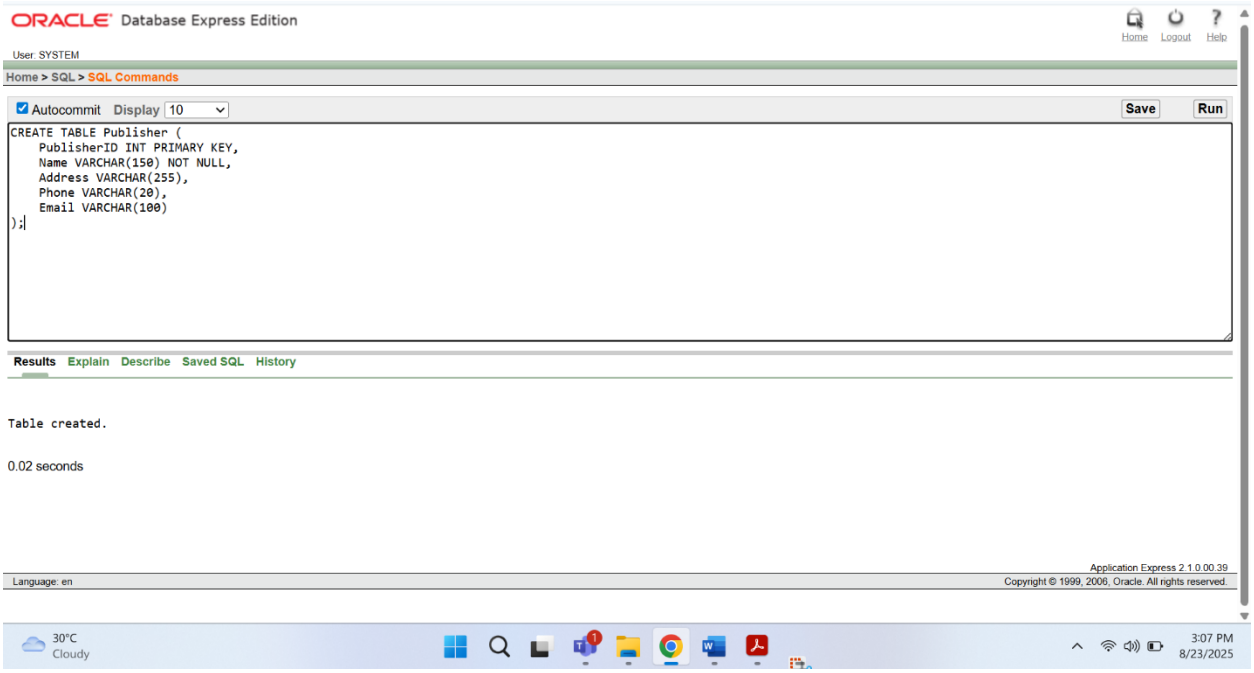


Table Creation:

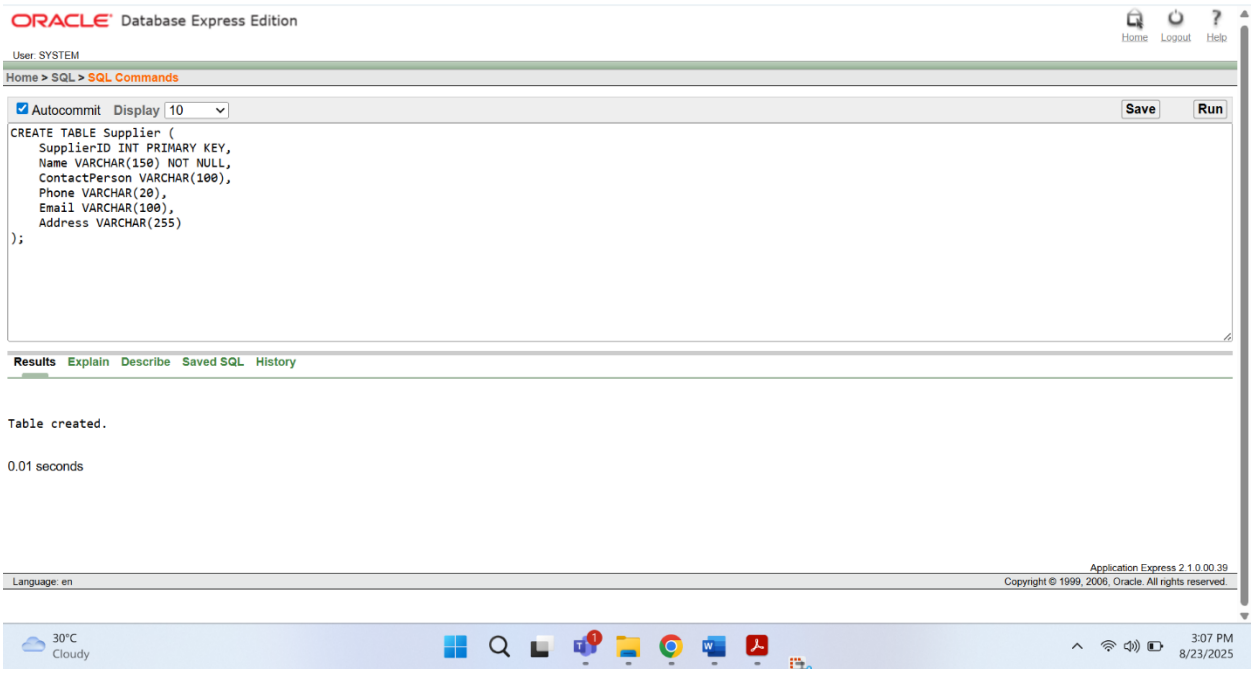
```
-- ========================
-- Bookshop Management System Database
-- ========================

-- Drop tables if already exist (for clean setup)
DROP TABLE Payment CASCADE CONSTRAINTS;
DROP TABLE Orders CASCADE CONSTRAINTS;
DROP TABLE Employee CASCADE CONSTRAINTS;
DROP TABLE Customer CASCADE CONSTRAINTS;
DROP TABLE Books CASCADE CONSTRAINTS;
DROP TABLE Supplier CASCADE CONSTRAINTS;
DROP TABLE Publisher CASCADE CONSTRAINTS;
DROP TABLE Category CASCADE CONSTRAINTS;

-- ========================
-- 1. Category Table
-- ========================
CREATE TABLE Category (
  CategoryID INT PRIMARY KEY,
  Name VARCHAR(100) NOT NULL
);
```




```
-- ============================
-- 2. Publisher Table
-- ============================
CREATE TABLE Publisher (
   PublisherID INT PRIMARY KEY,
   Name VARCHAR(150) NOT NULL,
   Address VARCHAR(255),
   Phone VARCHAR(20),
   Email VARCHAR(100)
);
```



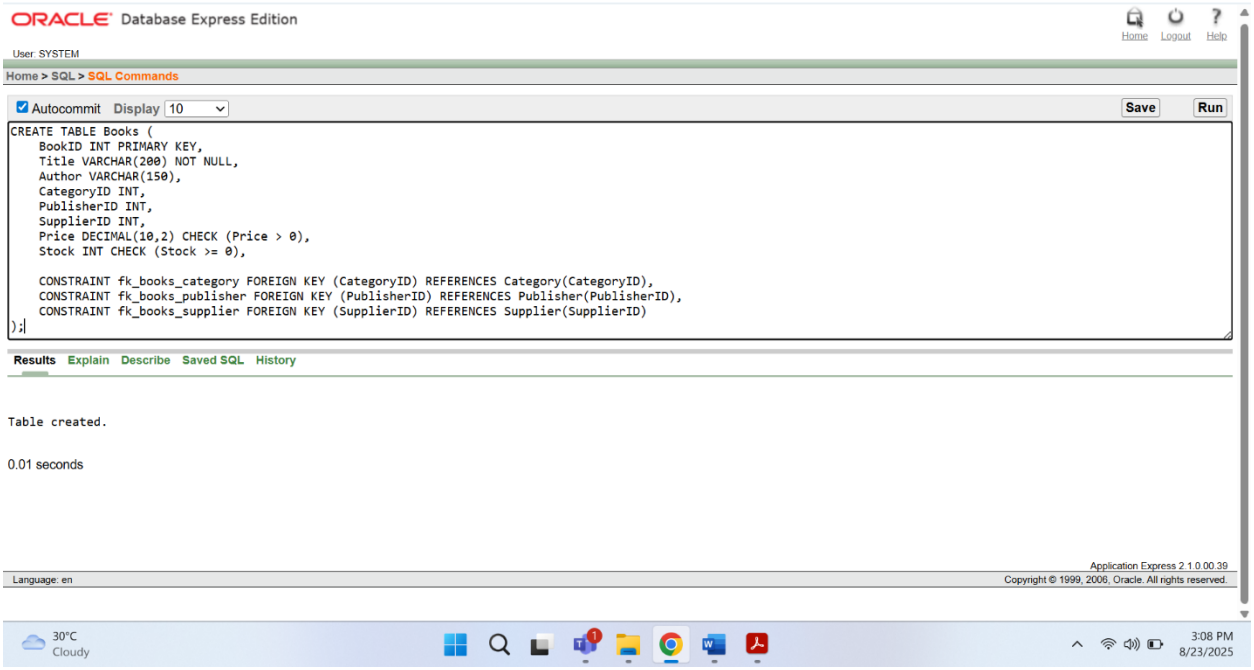

```
-- ============================
-- 3. Supplier Table
-- ============================
CREATE TABLE Supplier (
   SupplierID INT PRIMARY KEY,
   Name VARCHAR(150) NOT NULL,
   ContactPerson VARCHAR(100),
   Phone VARCHAR(20),
   Email VARCHAR(100),
   Address VARCHAR(255)
);
```




```
-- ==========================
-- 4. Books Table
-- ==========================
CREATE TABLE Books (
   BookID INT PRIMARY KEY,
   Title VARCHAR(200) NOT NULL,
   Author VARCHAR(150),
   CategoryID INT,
   PublisherID INT,
   SupplierID INT,
   Price DECIMAL(10,2) CHECK (Price > 0),
   Stock INT CHECK (Stock >= 0),


   CONSTRAINT fk_books_category FOREIGN KEY (CategoryID) REFERENCES
Category(CategoryID),
   CONSTRAINT fk_books_publisher FOREIGN KEY (PublisherID) REFERENCES
Publisher(PublisherID),
   CONSTRAINT fk_books_supplier FOREIGN KEY (SupplierID) REFERENCES
```

Supplier(SupplierID)

);



-- ==========

-- 5. Customer Table

-- ==========

CREATE TABLE Customer (

CustomerID INT PRIMARY KEY,

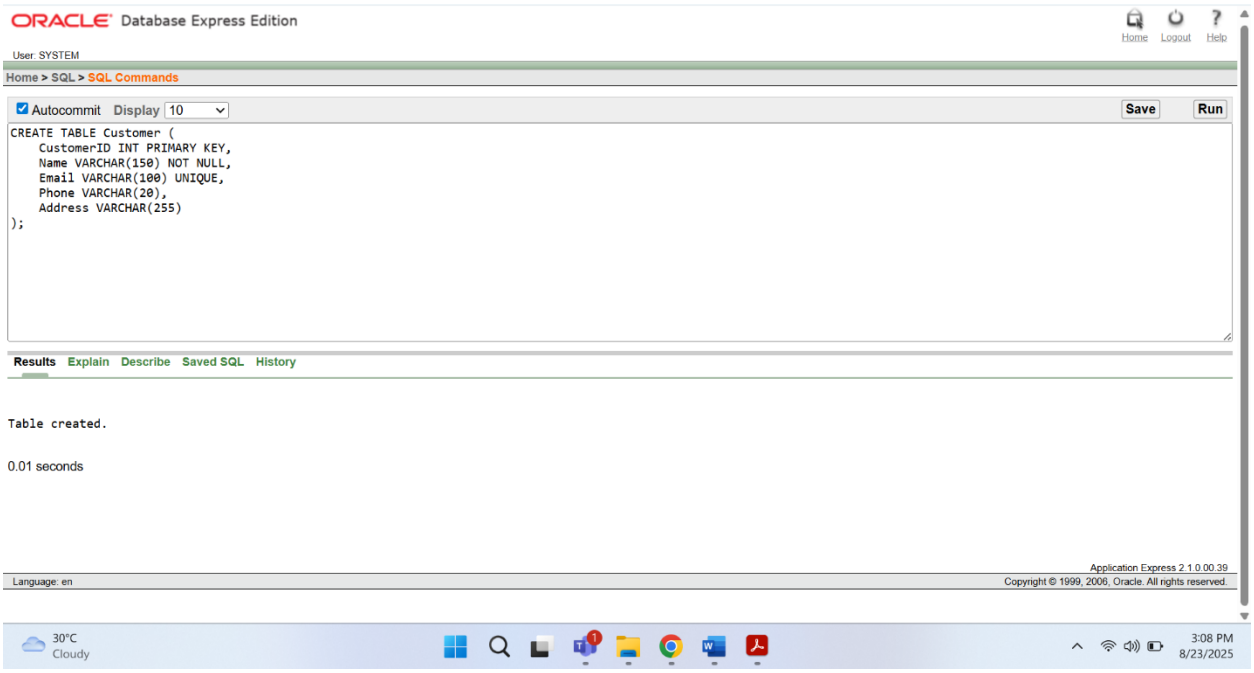
Name VARCHAR(150) NOT NULL,

Email VARCHAR(100) UNIQUE,

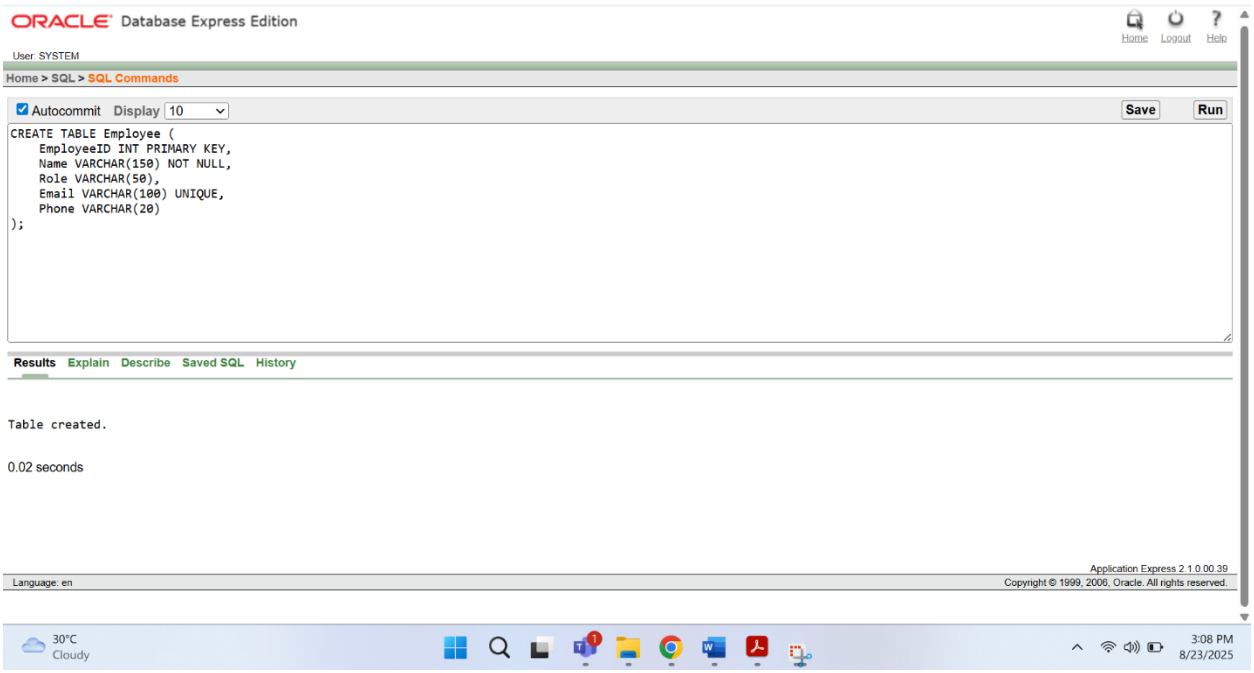
Phone VARCHAR(20),

Address VARCHAR(255)

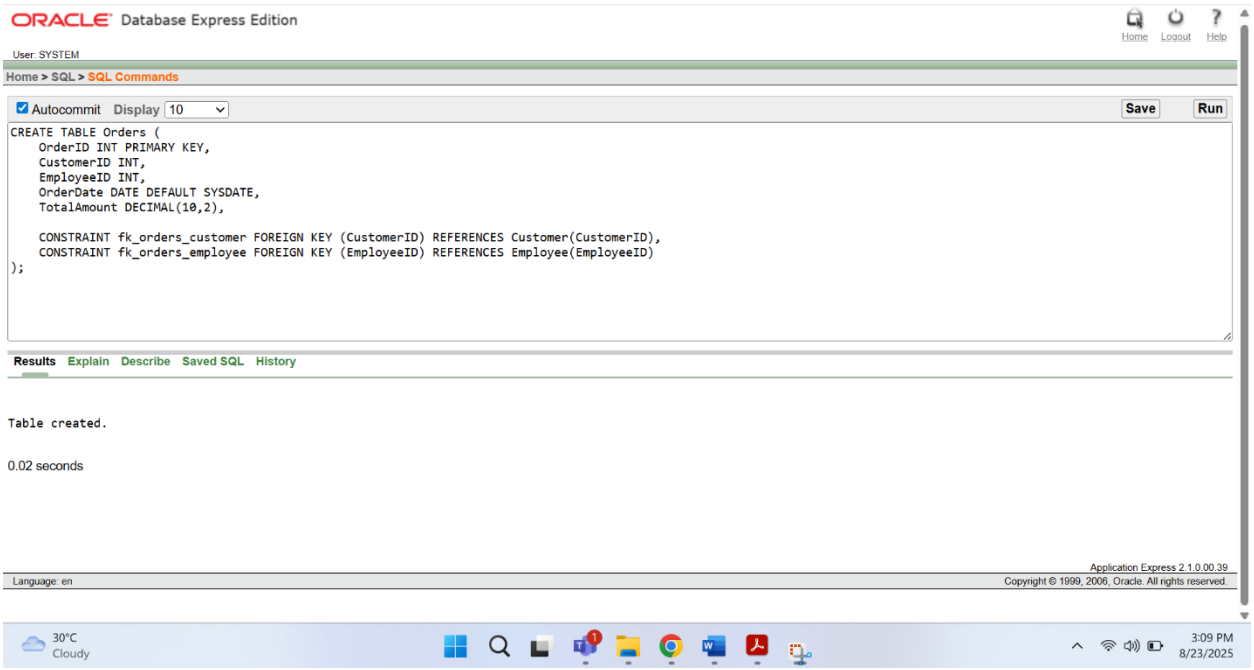
);



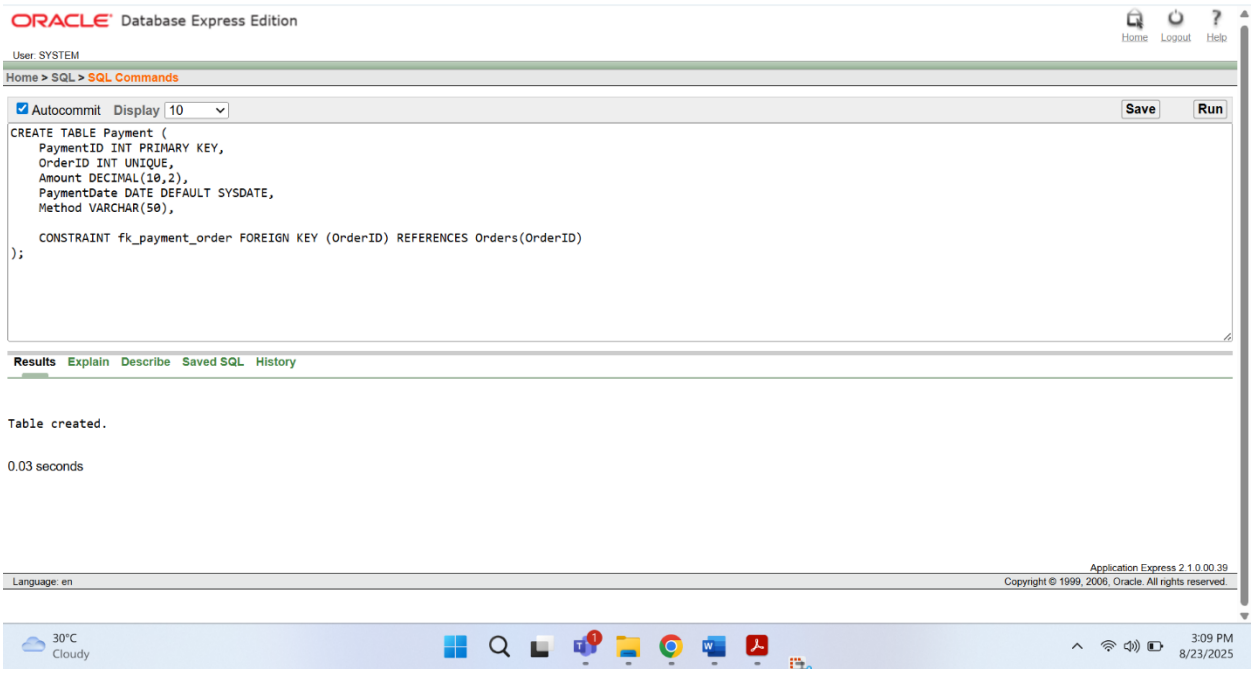

```
-- ============================
-- 6. Employee Table
-- ============================
CREATE TABLE Employee (
   EmployeeID INT PRIMARY KEY,
   Name VARCHAR(150) NOT NULL,
   Role VARCHAR(50),
   Email VARCHAR(100) UNIQUE,
   Phone VARCHAR(20)
);
```




```
-- ============================
-- 7. Orders Table
-- ============================
CREATE TABLE Orders (
   OrderID INT PRIMARY KEY,
   CustomerID INT,
   EmployeeID INT,
   OrderDate DATE DEFAULT SYSDATE,
   TotalAmount DECIMAL(10,2),

   CONSTRAINT fk_orders_customer FOREIGN KEY (CustomerID)
REFERENCES Customer(CustomerID),
   CONSTRAINT fk_orders_employee FOREIGN KEY (EmployeeID)
REFERENCES Employee(EmployeeID)
);
```




```
-- ============================
-- 8. Payment Table
-- ============================
CREATE TABLE Payment (
   PaymentID INT PRIMARY KEY,
   OrderID INT UNIQUE,
   Amount DECIMAL(10,2),
   PaymentDate DATE DEFAULT SYSDATE,
   Method VARCHAR(50),

   CONSTRAINT fk_payment_order FOREIGN KEY (OrderID) REFERENCES
Orders(OrderID)
);
```



Data Insertion:

-- ==========

-- INSERT SAMPLE DATA

-- ==========

-- Category

INSERT INTO Category VALUES (1, 'Programming');

```
INSERT INTO Category VALUES (2, 'Science');
INSERT INTO Category VALUES (3, 'Technology');
INSERT INTO Category VALUES (4, 'Database');

-- Publisher
INSERT INTO Publisher VALUES (1, 'Pearson', ' Dhaka', '01711111111',
'pearson@gmail.com');
INSERT INTO Publisher VALUES (2, 'Lecture', ' Dhaka', '01722222222',
'lecture@gmail.com');

-- Supplier
INSERT INTO Supplier VALUES (1, 'Global Books', 'Mr. Karim', '01811111111',
'global@gmail.com', 'Banani, Dhaka');
INSERT INTO Supplier VALUES (2, 'TechSource', 'Mr. Rahman', '01822222222',
'techsource@gmail.com', 'Uttara, Dhaka');

-- Books
INSERT INTO Books VALUES (1, 'Database Systems', 'C.J. Date', 3, 1, 2,
1200.00, 50);
INSERT INTO Books VALUES (2, 'Modern Physics', 'Kenneth Krane', 2, 1, 1,
950.00, 30);
INSERT INTO Books VALUES (3, 'The Great Gatsby', 'F. Scott Fitzgerald', 1, 2, 1,
500.00, 20);

-- Customer
INSERT INTO Customer VALUES (1, 'Ashik', 'ashik@gmail.com', '01911111111',
'Dhanmondi, Dhaka');
INSERT INTO Customer VALUES (2, 'Rony', 'rony@gmail.com', '01922222222',
'Bashundhara, Dhaka');
INSERT INTO Customer VALUES (2, 'Rafsan', 'rafsan@gmail.com',
'01922002222', 'Kuril, Dhaka');
```

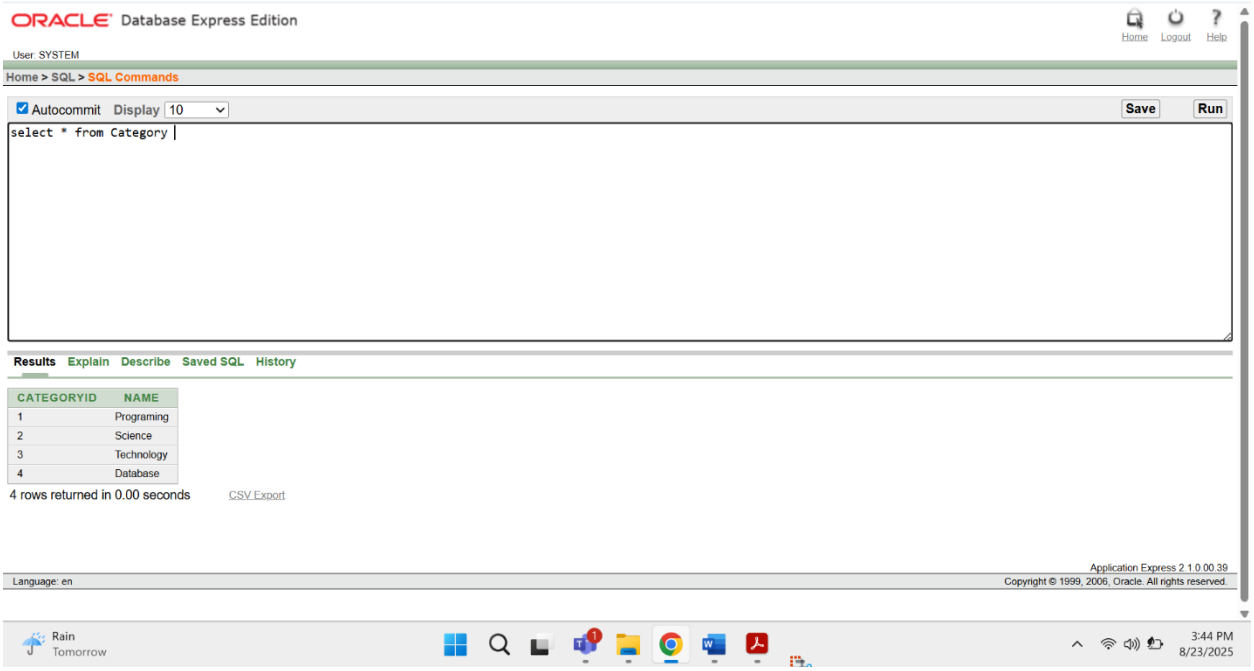
```
-- Employee
INSERT INTO Employee VALUES (1, 'Mr. Ashik', 'Cashier', 'ashik@gmail.com',
'01611111111');
INSERT INTO Employee VALUES (2, 'Ms. Rony', 'Manager', 'rony@gmail.com',
'01622222222');
INSERT INTO Employee VALUES (2, 'Ms. Rafsan', 'Manager',
'rafsan@gmail.com', '01622200222');

-- Orders
INSERT INTO Orders VALUES (1, 1, 1, SYSDATE, 1200.00);
INSERT INTO Orders VALUES (2, 2, 2, SYSDATE, 1450.00);

-- Payment
INSERT INTO Payment VALUES (1, 1, 1200.00, SYSDATE, 'Cash');
INSERT INTO Payment VALUES (2, 2, 1450.00, SYSDATE, 'Bkash');


-- Category (5 rows)

-- ========================

INSERT INTO Category VALUES (1, 'Programming');

INSERT INTO Category VALUES (2, 'Science');

INSERT INTO Category VALUES (3, 'Technology');

INSERT INTO Category VALUES (4, 'Database');

INSERT INTO Category VALUES (5, 'Literature');


-- ========================

-- Publisher (5 rows)

-- ========================
```

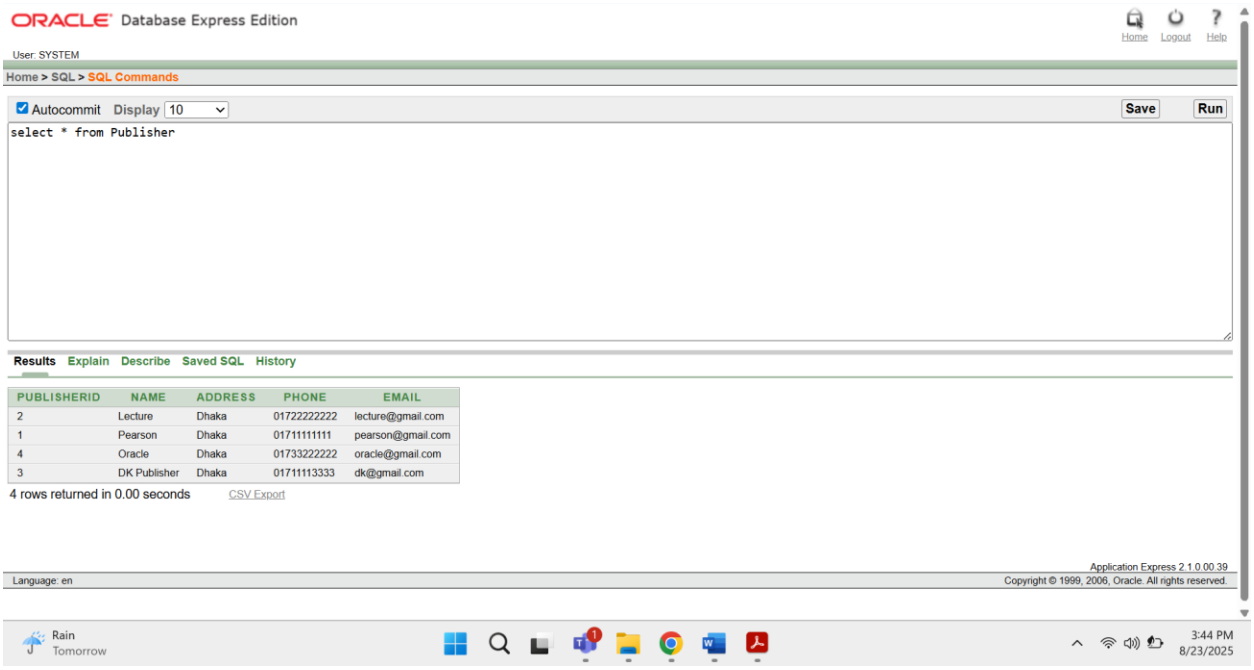
```
INSERT INTO Publisher VALUES (1, 'Pearson', 'Dhaka', '01711111111',
'pearson@gmail.com');

INSERT INTO Publisher VALUES (2, 'Lecture', 'Dhaka', '01722222222',
'lecture@gmail.com');

INSERT INTO Publisher VALUES (5, 'TechWorld', 'Sylhet', '01755555555',
'techworld@gmail.com');

INSERT INTO Publisher VALUES (6, 'Ten Minute', 'Dhaka', '01733333333',
'tenminute@gmail.com');



-- ========================

-- Supplier (5 rows)

-- ========================

INSERT INTO Supplier VALUES (1, 'Global Books', 'Mr. Karim', '01811111111',
'global@gmail.com', 'Banani, Dhaka');

INSERT INTO Supplier VALUES (2, 'TechSource', 'Mr. Rahman', '01822222222',
'techsource@gmail.com', 'Uttara, Dhaka');

INSERT INTO Supplier VALUES (3, 'BookZone', 'Mr. Fardin', '01833333333',
'bookzone@gmail.com', 'Mirpur, Dhaka');

INSERT INTO Supplier VALUES (4, 'KnowledgeHub', 'Mr. Noman',
'01844444444', 'knowledgehub@gmail.com', 'Dhanmondi, Dhaka');

INSERT INTO Supplier VALUES (5, 'EduMart', 'Mr. Sourov', '01855555555',
'edumart@gmail.com', 'Gulshan, Dhaka');



-- ========================

-- Books (5 rows)
```

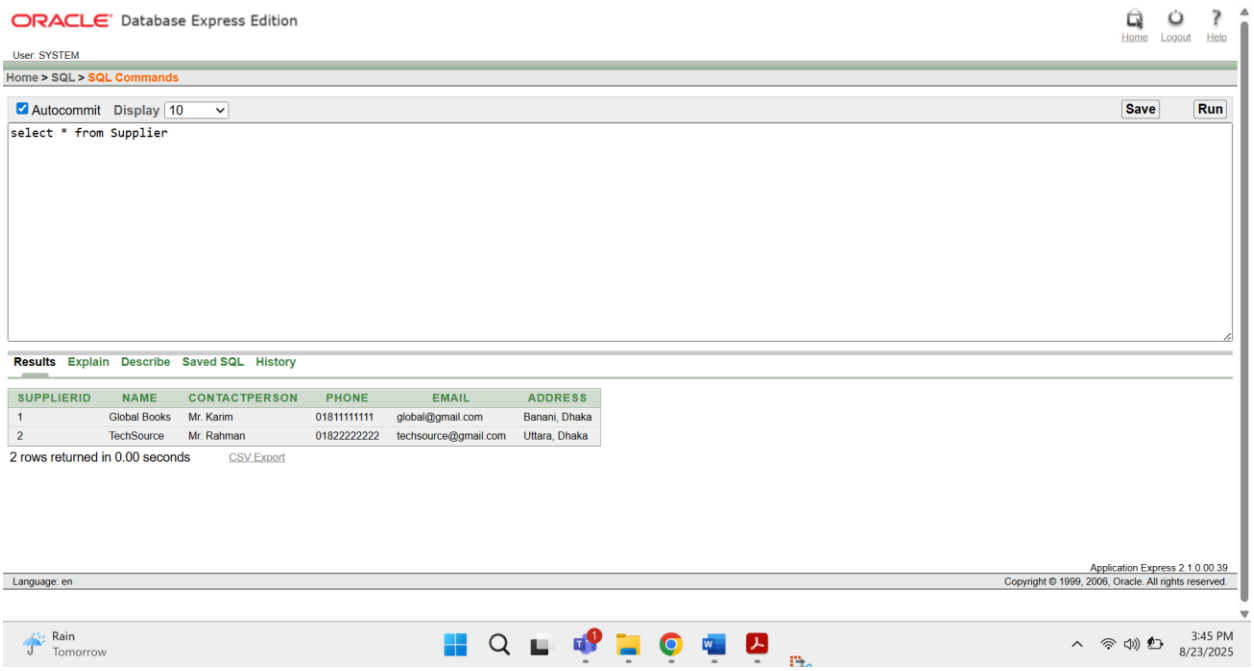
```
-- ========================

INSERT INTO Books VALUES (1, 'Database Systems', 'C.J. Date', 4, 1, 2, 1200.00, 50);

INSERT INTO Books VALUES (2, 'Modern Physics', 'Kenneth Krane', 2, 1, 1, 950.00, 30);

INSERT INTO Books VALUES (3, 'The Great Gatsby', 'F. Scott Fitzgerald', 5, 2, 1, 500.00, 20);

INSERT INTO Books VALUES (4, 'Learning Python', 'Mark Lutz', 1, 3, 3, 1500.00, 40);

INSERT INTO Books VALUES (5, 'AI Revolution', 'Stuart Russell', 3, 4, 4, 1800.00, 25);


-- ========================

-- Customer (5 rows)

-- ========================

INSERT INTO Customer VALUES (1, 'Ashik', 'ashik@gmail.com', '01911111111', 'Dhanmondi, Dhaka');

INSERT INTO Customer VALUES (2, 'Rony', 'rony@gmail.com', '01922222222', 'Bashundhara, Dhaka');

INSERT INTO Customer VALUES (3, 'Rafsan', 'rafsan@gmail.com', '01933333333', 'Kuril, Dhaka');

INSERT INTO Customer VALUES (4, 'Fardin', 'fardin@gmail.com', '01944444444', 'Banani, Dhaka');

INSERT INTO Customer VALUES (5, 'Noman', 'noman@gmail.com', '01955555555', 'Mirpur, Dhaka');
```

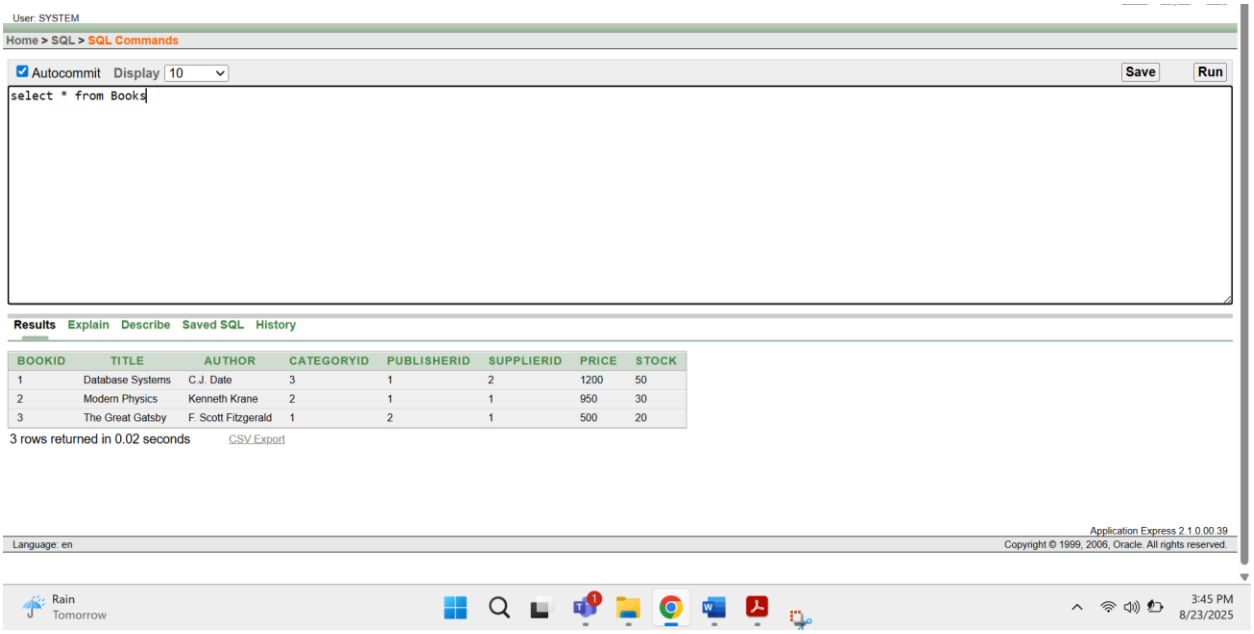
```
-- ========================

-- Employee (5 rows)

-- ========================

INSERT INTO Employee VALUES (1, 'Mr. Ashik', 'Cashier', 'ashik@gmail.com', '01611111111');

INSERT INTO Employee VALUES (2, 'Ms. Rony', 'Manager', 'rony@gmail.com', '01622222222');

INSERT INTO Employee VALUES (3, 'Mr. Rafsan', 'Sales', 'rafsan@gmail.com', '01633333333');

INSERT INTO Employee VALUES (4, 'Mr. Sourov', 'Delivery', 'sourov@gmail.com', '01644444444');

INSERT INTO Employee VALUES (5, 'Mr. Akash', 'Accountant', 'akash@gmail.com', '01655555555');


-- ========================

-- Orders (5 rows)

-- ========================

INSERT INTO Orders VALUES (1, 1, 1, SYSDATE, 1200.00);

INSERT INTO Orders VALUES (2, 2, 2, SYSDATE, 1450.00);

INSERT INTO Orders VALUES (3, 3, 3, SYSDATE, 500.00);

INSERT INTO Orders VALUES (4, 4, 4, SYSDATE, 1500.00);

INSERT INTO Orders VALUES (5, 5, 5, SYSDATE, 1800.00);


-- ========================

-- Payment (5 rows)
```

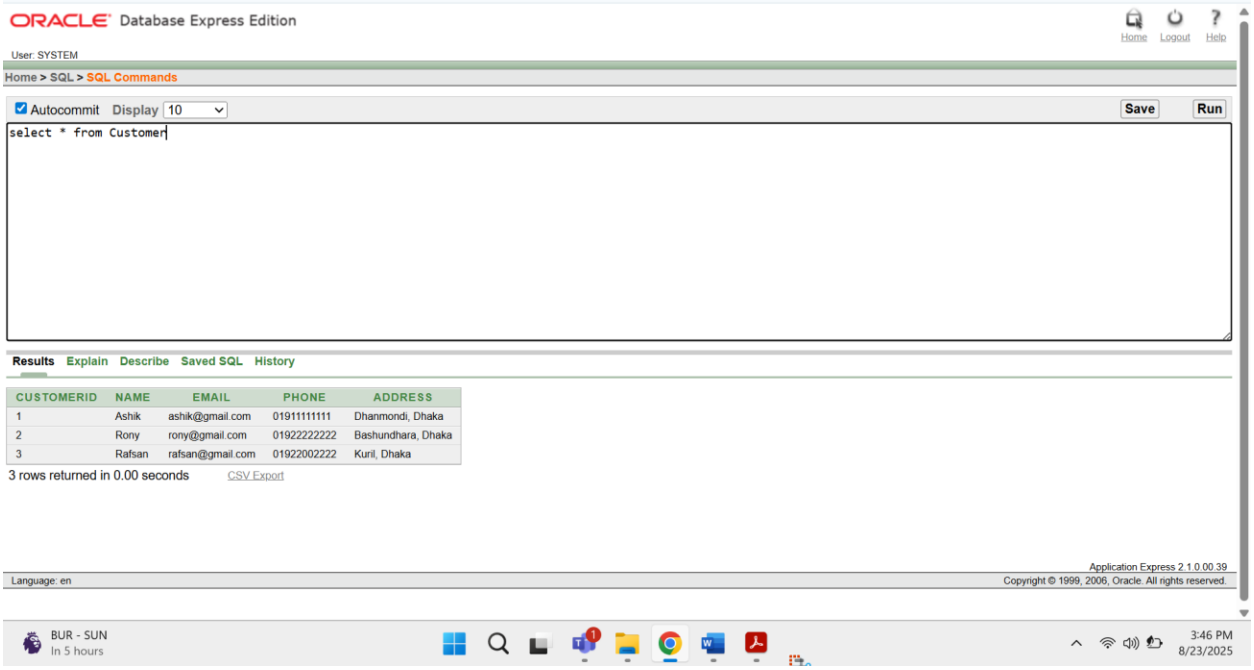
```
-- ========================

INSERT INTO Payment VALUES (1, 1, 1200.00, SYSDATE, 'Cash');

INSERT INTO Payment VALUES (2, 2, 1450.00, SYSDATE, 'Bkash');

INSERT INTO Payment VALUES (3, 3, 500.00, SYSDATE, 'Card');

INSERT INTO Payment VALUES (4, 4, 1500.00, SYSDATE, 'Cash');

INSERT INTO Payment VALUES (5, 5, 1800.00, SYSDATE, 'Bkash');
```

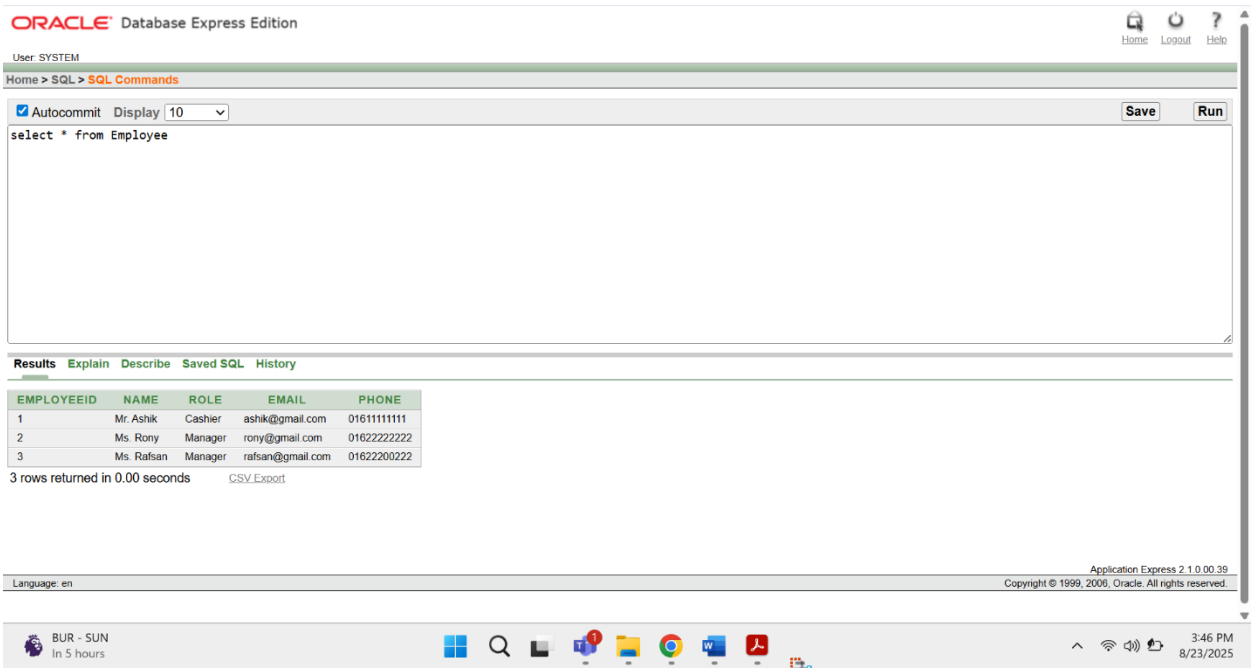

ORACLE Database Express Edition
Home
Logout
Help
User: SYSTEM
Home > SQL > SQL Commands
Autocommit
Display 10
Save
Run
select * from Category
Results
Explain
Describe
Saved SQL
History
CATEGORYID NAME
1 Programing
2 Science
3 Technology
4 Database
4 rows returned in 0.00 seconds
CSV Export
Application Express 2.1.0.00.39
Language: en
Copyright © 1999, 2006, Oracle. All rights reserved.
Rain
Tomorrow
3:44 PM
8/23/2025

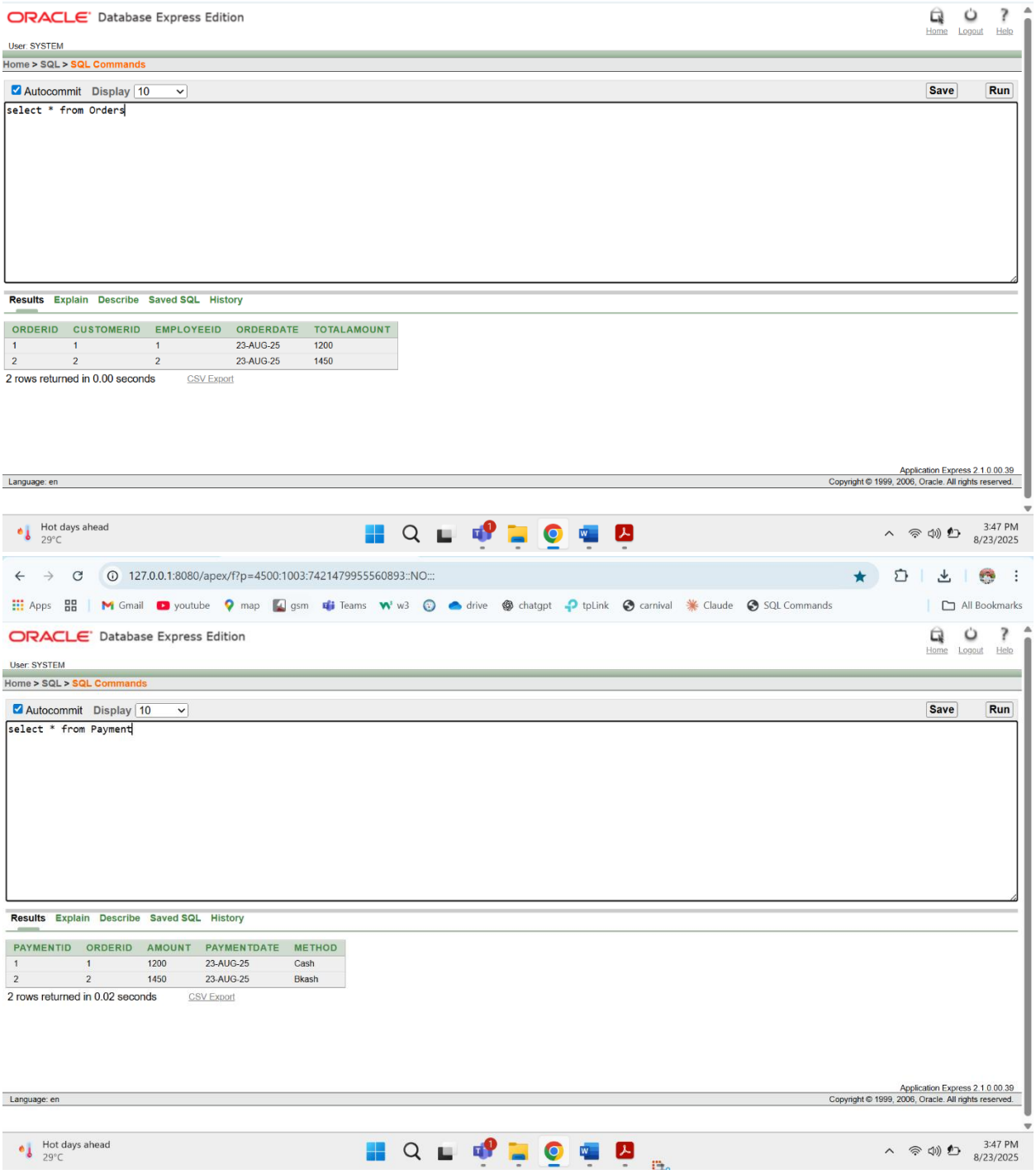

ORACLE Database Express Edition
Home
Logout
Help
User: SYSTEM
Home > SQL > SQL Commands
Autocommit
Display 10
Save
Run
select * from Publisher
Results
Explain
Describe
Saved SQL
History
PUBLISHERID NAME ADDRESS PHONE EMAIL
2 Lecture Dhaka 01722222222 lecture@gmail.com
1 Pearson Dhaka 01711111111 pearson@gmail.com
4 Oracle Dhaka 01733222222 oracle@gmail.com
3 DK Publisher Dhaka 01711113333 dk@gmail.com
4 rows returned in 0.00 seconds
CSV Export
Application Express 2.1.0.00.39
Language: en
Copyright © 1999, 2006, Oracle. All rights reserved.
Rain
Tomorrow
3:44 PM
8/23/2025


ORACLE Database Express Edition
Home
Logout
Help
User: SYSTEM
Home > SQL > SQL Commands
Autocommit
Display 10
Save
Run
select * from Supplier
Results
Explain
Describe
Saved SQL
History
SUPPLIERID NAME CONTACTPERSON PHONE EMAIL ADDRESS
1 Global Books Mr. Karim 01811111111 global@gmail.com Banani, Dhaka
2 TechSource Mr. Rahman 01822222222 techsource@gmail.com Uttara, Dhaka
2 rows returned in 0.00 seconds
CSV Export
Application Express 2.1.0.00.39
Language: en
Copyright © 1999, 2006, Oracle. All rights reserved.
Rain
Tomorrow
3:45 PM
8/23/2025


User: SYSTEM
Home > SQL > SQL Commands
Autocommit
Display 10
Save
Run
select * from Books
Results
Explain
Describe
Saved SQL
History
BOOKID TITLE AUTHOR CATEGORYID PUBLISHERID SUPPLIERID PRICE STOCK
1 Database Systems C.J. Date 3 1 2 1200 50
2 Modern Physics Kenneth Krane 2 1 1 950 30
3 The Great Gatsby F. Scott Fitzgerald 1 2 1 500 20
3 rows returned in 0.02 seconds
CSV Export
Application Express 2.1.0.00.39
Language: en
Copyright © 1999, 2006, Oracle. All rights reserved.
Rain
Tomorrow
3:45 PM
8/23/2025


ORACLE Database Express Edition
Home
Logout
Help
User: SYSTEM
Home > SQL > SQL Commands
Autocommit
Display 10
Save
Run
select * from Customer
Results
Explain
Describe
Saved SQL
History
CUSTOMERID NAME EMAIL PHONE ADDRESS
1 Ashik ashik@gmail.com 01911111111 Dhanmondi, Dhaka
2 Rony rony@gmail.com 01922222222 Bashundhara, Dhaka
3 Rafsan rafsan@gmail.com 01922002222 Kuril, Dhaka
3 rows returned in 0.00 seconds
CSV Export
Application Express 2.1.0.00.39
Language: en
Copyright © 1999, 2006, Oracle. All rights reserved.
BUR - SUN
In 5 hours
3:46 PM
8/23/2025


ORACLE Database Express Edition
Home
Logout
Help
User: SYSTEM
Home > SQL > SQL Commands
Autocommit
Display 10
Save
Run
select * from Employee
Results
Explain
Describe
Saved SQL
History
EMPLOYEEID NAME ROLE EMAIL PHONE
1 Mr. Ashik Cashier ashik@gmail.com 01611111111
2 Ms. Rony Manager rony@gmail.com 01622222222
3 Ms. Rafsan Manager rafsan@gmail.com 01622200222
3 rows returned in 0.00 seconds
CSV Export
Application Express 2.1.0.00.39
Language: en
Copyright © 1999, 2006, Oracle. All rights reserved.
BUR - SUN
In 5 hours
3:46 PM
8/23/2025


ORACLE Database Express Edition
User: SYSTEM
Home > SQL > SQL Commands
Autocommit Display 10
Save Run
select * from Orders
Results Explain Describe Saved SQL History
ORDERID CUSTOMERID EMPLOYEEID ORDERDATE TOTALAMOUNT
1 1 1 23-AUG-25 1200
2 2 2 23-AUG-25 1450
2 rows returned in 0.00 seconds CSV Export
Application Express 2.1.0.00.39
Language: en
Copyright © 1999, 2006, Oracle. All rights reserved.
Hot days ahead
29°C
3:47 PM
8/23/2025
127.0.0.1:8080/apex/f?p=4500:1003:7421479955560893::NO:::
Apps Gmail youtube map gsm Teams w3 drive chatgpt tpLink carnival Claude SQL Commands All Bookmarks
ORACLE Database Express Edition
User: SYSTEM
Home > SQL > SQL Commands
Autocommit Display 10
Save Run
select * from Payment
Results Explain Describe Saved SQL History
PAYMENTID ORDERID AMOUNT PAYMENTDATE METHOD
1 1 1200 23-AUG-25 Cash
2 2 1450 23-AUG-25 Bkash
2 rows returned in 0.02 seconds CSV Export
Application Express 2.1.0.00.39
Language: en
Copyright © 1999, 2006, Oracle. All rights reserved.
Hot days ahead
29°C
3:47 PM
8/23/2025

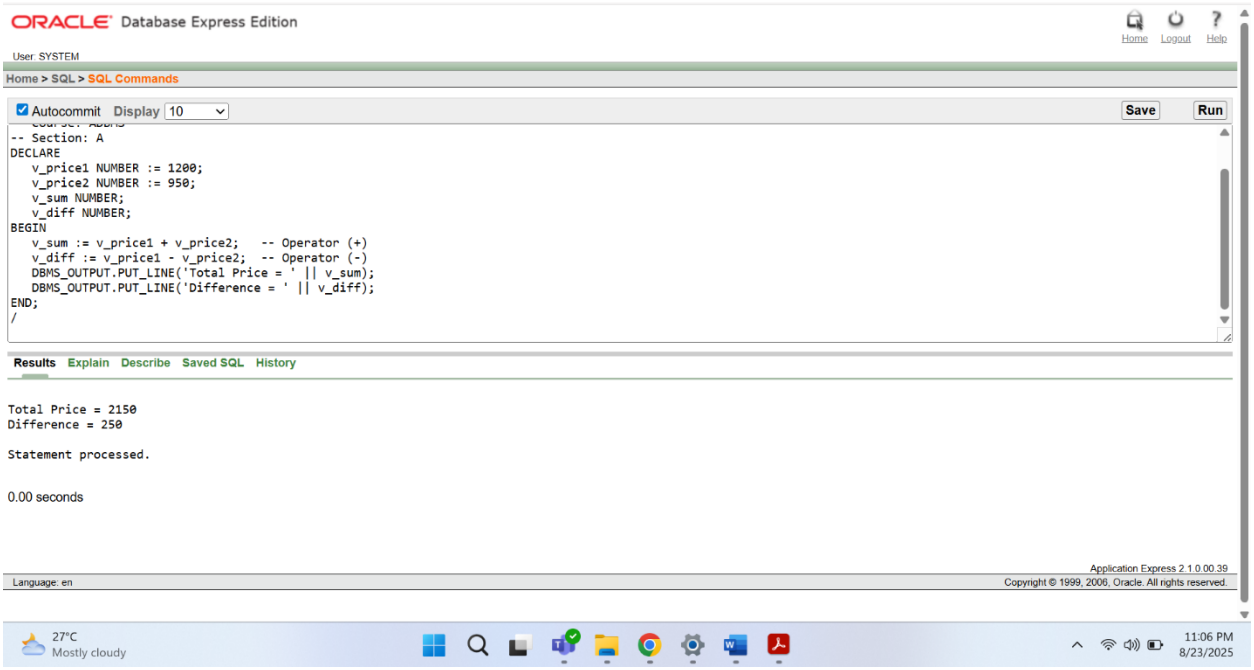
SQL Query:

Basic PL/SQL

1. variables

```
-- Project: Bookshop Management System
-- Semester: Summer 2025
-- Course: ADBMS
-- Section: A
DECLARE
  v_price1 NUMBER := 1200;
  v_price2 NUMBER := 950;
  v_sum NUMBER;
  v_diff NUMBER;
BEGIN
  v_sum := v_price1 + v_price2;   -- Operator (+)
  v_diff := v_price1 - v_price2;  -- Operator (-)
  DBMS_OUTPUT.PUT_LINE('Total Price = ' || v_sum);
  DBMS_OUTPUT.PUT_LINE('Difference = ' || v_diff);
END;
```

/



2.variables + single row

-- Project: Bookshop Management System

-- Semester: Summer 2025

-- Course: ADBMS

-- Section: A

DECLARE

v_name VARCHAR2(50) := 'fardin';

v_date DATE := SYSDATE;

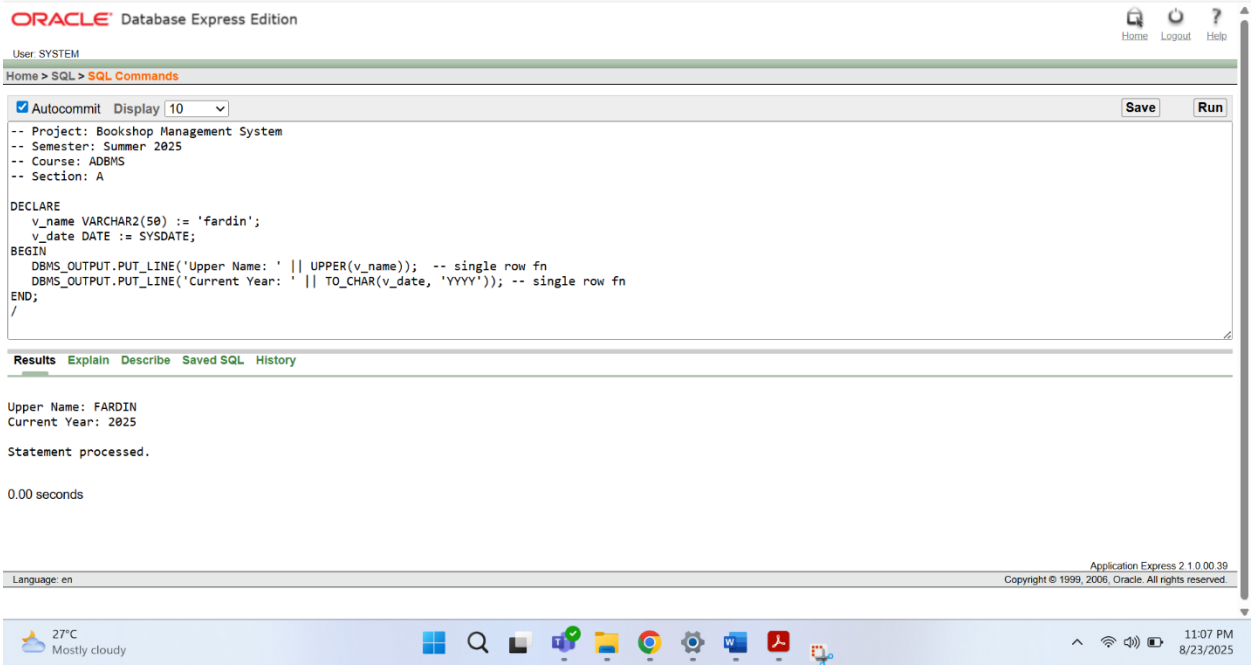
BEGIN

DBMS_OUTPUT.PUT_LINE('Upper Name: ' || UPPER(v_name)); -- single row fn

DBMS_OUTPUT.PUT_LINE('Current Year: ' || TO_CHAR(v_date, 'YYYY')); -- single row fn

END;

/



3. operators

-- Project: Bookshop Management System

-- Semester: Summer 2025

-- Course: ADBMS

-- Section: A

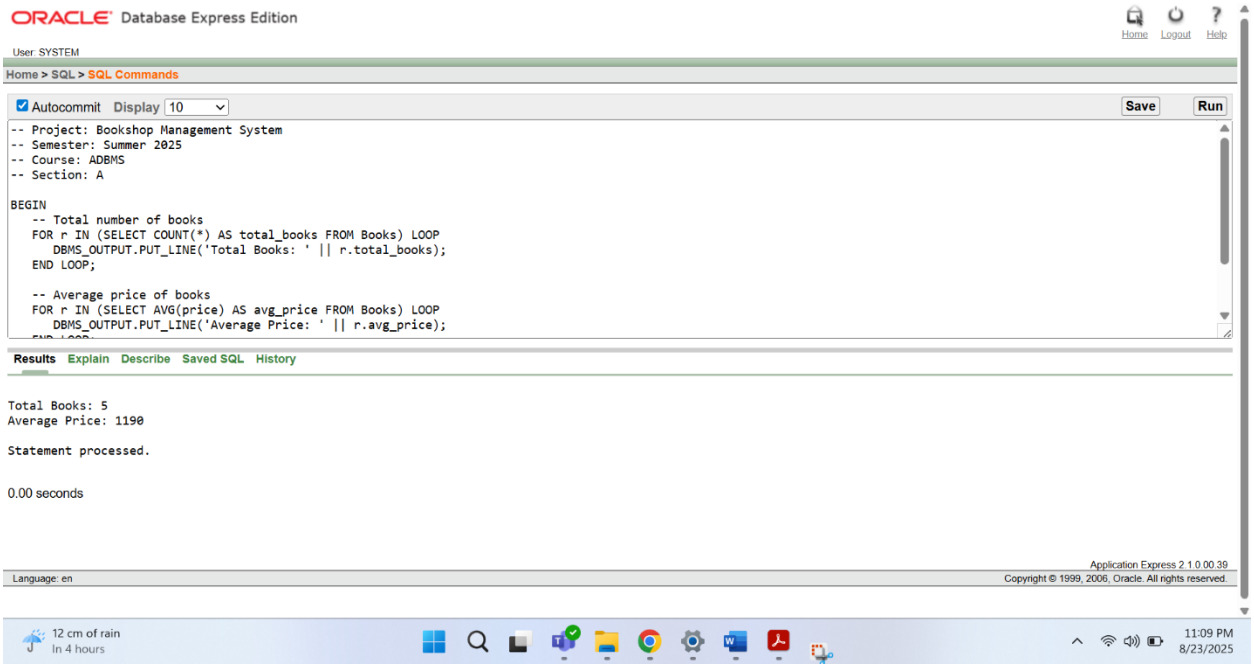
BEGIN

-- Total number of books

FOR r IN (SELECT COUNT(*) AS total_books FROM Books) LOOP

DBMS_OUTPUT.PUT_LINE('Total Books: ' || r.total_books);

```
  END LOOP;

  -- Average price of books
  FOR r IN (SELECT AVG(price) AS avg_price FROM Books) LOOP
    DBMS_OUTPUT.PUT_LINE('Average Price: ' || r.avg_price);
  END LOOP;
END;
/
```

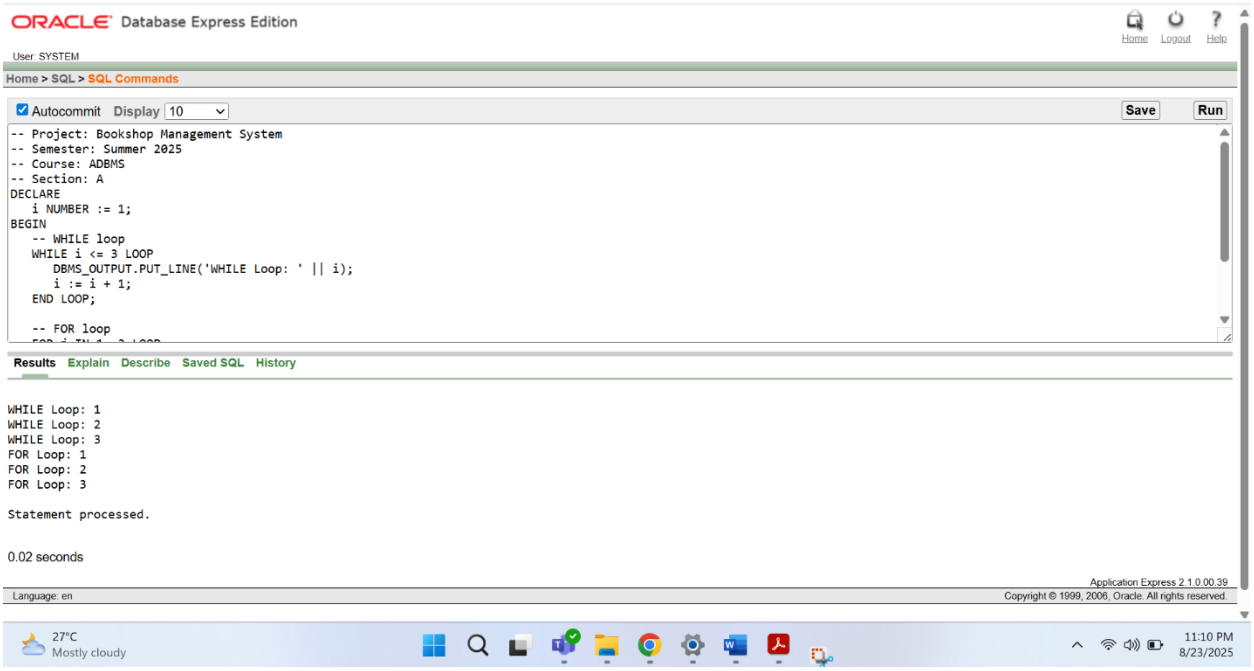


4.Loop

```
-- Project: Bookshop Management System
-- Semester: Summer 2025
-- Course: ADBMS
-- Section: A
```

```
DECLARE
  i NUMBER := 1;
BEGIN
  -- WHILE loop
  WHILE i <= 3 LOOP
    DBMS_OUTPUT.PUT_LINE('WHILE Loop: ' || i);
    i := i + 1;
  END LOOP;

  -- FOR loop
  FOR j IN 1..3 LOOP
    DBMS_OUTPUT.PUT_LINE('FOR Loop: ' || j);
  END LOOP;
END;
```

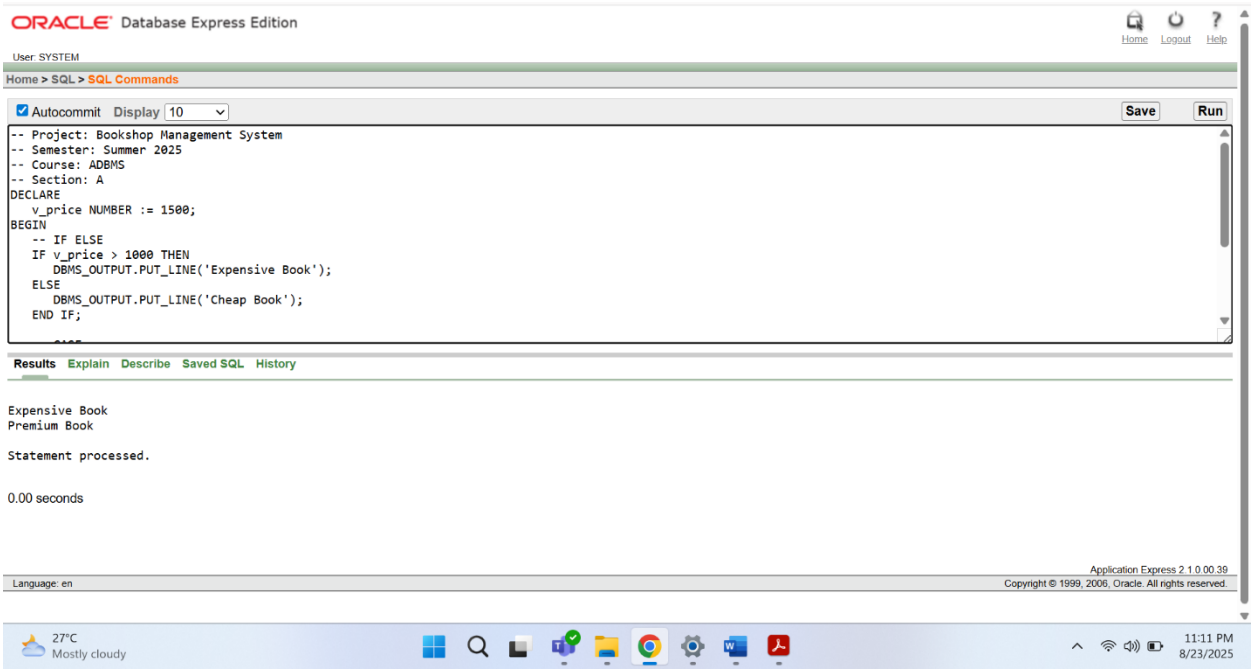
/



5. *Conditional statements*

```
-- Project: Bookshop Management System

-- Semester: Summer 2025

-- Course: ADBMS

-- Section: A

DECLARE

  v_price NUMBER := 1500;

BEGIN

  -- IF ELSE

  IF v_price > 1000 THEN

    DBMS_OUTPUT.PUT_LINE('Expensive Book');

  ELSE
```

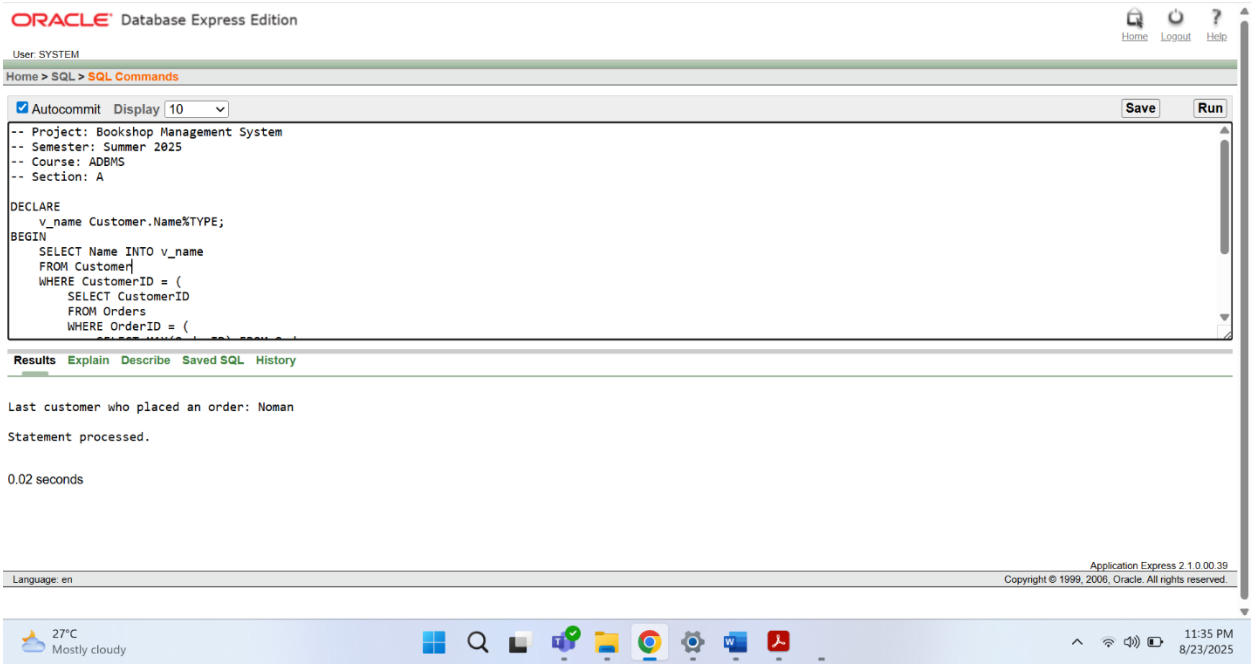
```
    DBMS_OUTPUT.PUT_LINE('Cheap Book');
  END IF;


  -- CASE
  CASE
    WHEN v_price = 500 THEN DBMS_OUTPUT.PUT_LINE('Low Price');
    WHEN v_price = 1500 THEN DBMS_OUTPUT.PUT_LINE('Premium Book');
    ELSE DBMS_OUTPUT.PUT_LINE('Other Price Range');
  END CASE;
END;
/
```



6. Subquery

```
-- Project: Bookshop Management System
```

```
-- Semester: Summer 2025
-- Course: ADBMS
-- Section: A

DECLARE
  v_name Customer.Name%TYPE;
BEGIN
  SELECT Name INTO v_name
  FROM Customer
  WHERE CustomerID = (
    SELECT CustomerID
    FROM Orders
    WHERE OrderID = (
      SELECT MAX(OrderID) FROM Orders
    )
  );

  DBMS_OUTPUT.PUT_LINE('Last customer who placed an order: ' || v_name);
END;
/
```



7.Joining

-- Project: Bookshop Management System

-- Semester: Summer 2025

-- Course: ADBMS

-- Section: A

-- Project: Bookshop Management System

-- Semester: Summer 2025

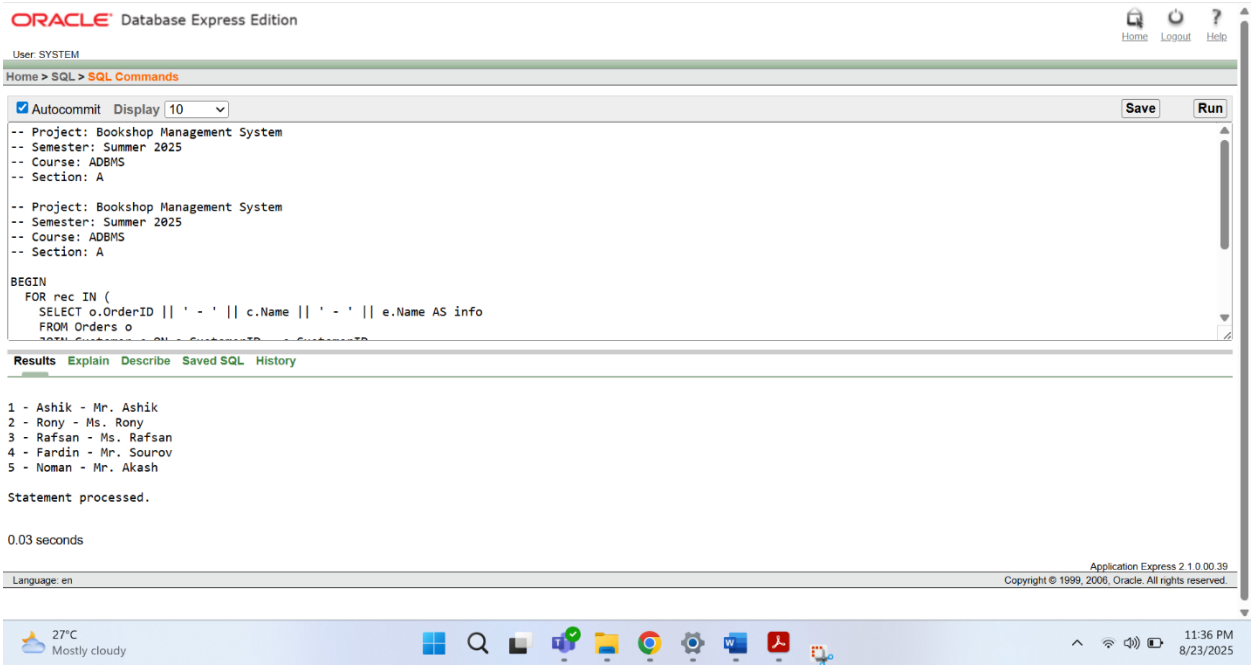
-- Course: ADBMS

-- Section: A

BEGIN

FOR rec IN (

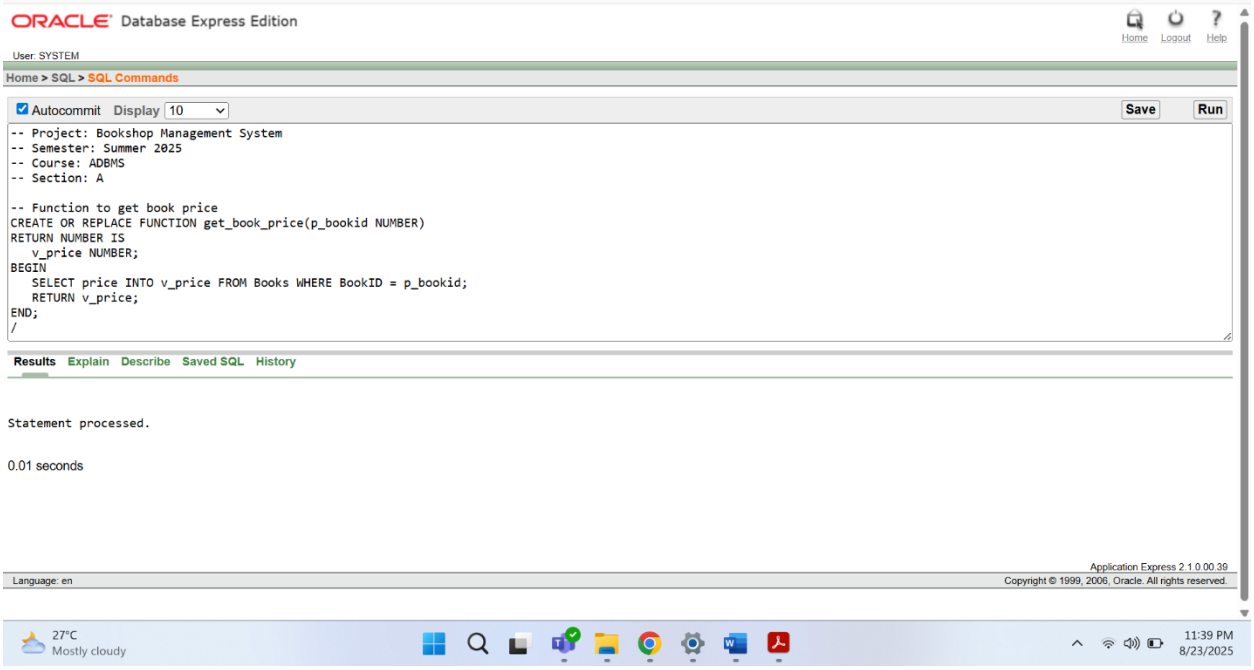
```
  SELECT o.OrderID || ' - ' || c.Name || ' - ' || e.Name AS info
  FROM Orders o
  JOIN Customer c ON o.CustomerID = c.CustomerID
  JOIN Employee e ON o.EmployeeID = e.EmployeeID
 ) LOOP
  DBMS_OUTPUT.PUT_LINE(rec.info);
 END LOOP;
END;
/
```



Advance PL/SQL

1.Stored Function

```
-- Project: Bookshop Management System
-- Semester: Summer 2025
-- Course: ADBMS
-- Section: A


-- Function to get book price
CREATE OR REPLACE FUNCTION get_book_price(p_bookid NUMBER)
RETURN NUMBER IS
  v_price NUMBER;
BEGIN
  SELECT price INTO v_price FROM Books WHERE BookID = p_bookid;
  RETURN v_price;
END;
/
```



2.Stored Function

-- Project: Bookshop Management System

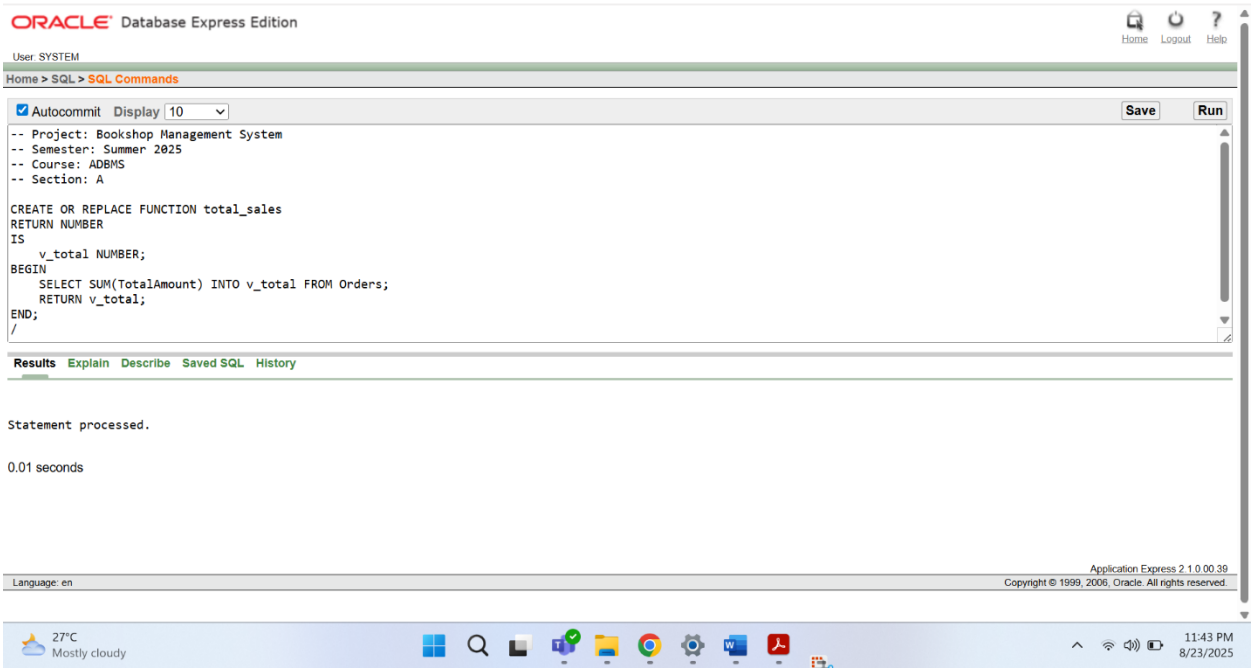
-- Semester: Summer 2025

-- Course: ADBMS

-- Section: A

```
CREATE OR REPLACE FUNCTION total_sales
RETURN NUMBER
IS
   v_total NUMBER;
BEGIN
   SELECT SUM(TotalAmount) INTO v_total FROM Orders;
```

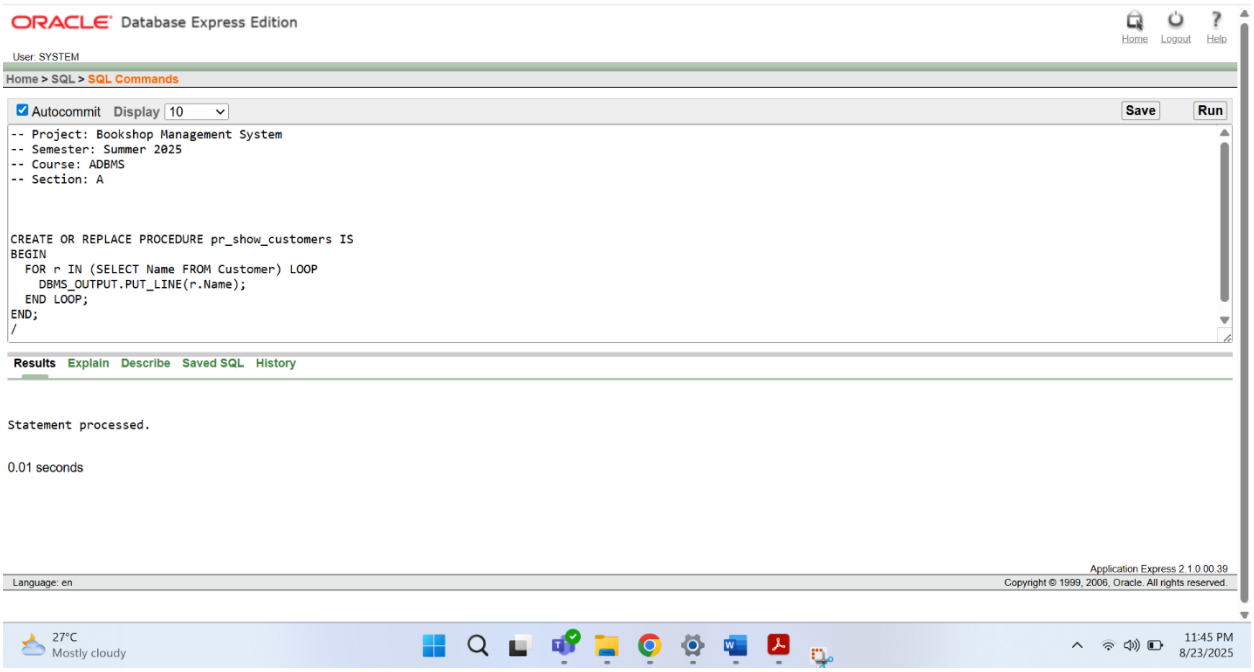
```
  RETURN v_total;
END;
/
```



3. Stored procedure

```
-- Project: Bookshop Management System
-- Semester: Summer 2025
-- Course: ADBMS
-- Section: A


CREATE OR REPLACE PROCEDURE pr_show_customers IS
BEGIN
  FOR r IN (SELECT Name FROM Customer) LOOP
```

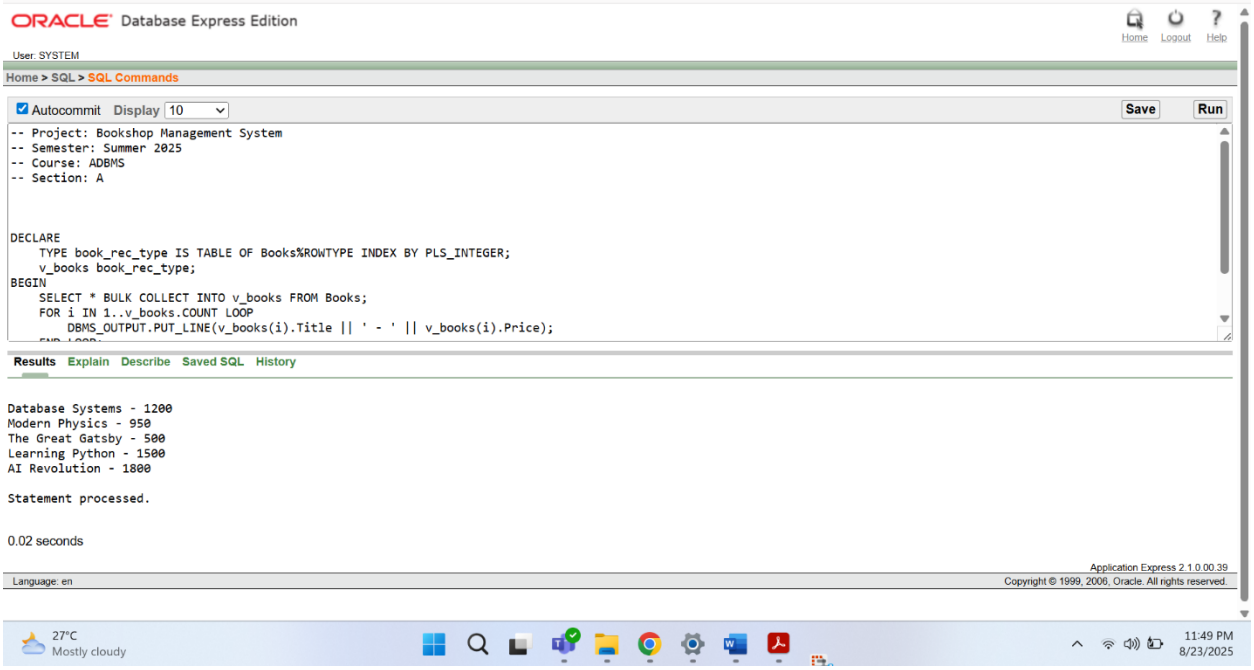
```
  DBMS_OUTPUT.PUT_LINE(r.Name);
 END LOOP;
END;
/
```



3. Table-based record

```
-- Project: Bookshop Management System
-- Semester: Summer 2025
-- Course: ADBMS
-- Section: A

DECLARE
  TYPE book_rec_type IS TABLE OF Books%ROWTYPE INDEX BY
PLS_INTEGER;
```

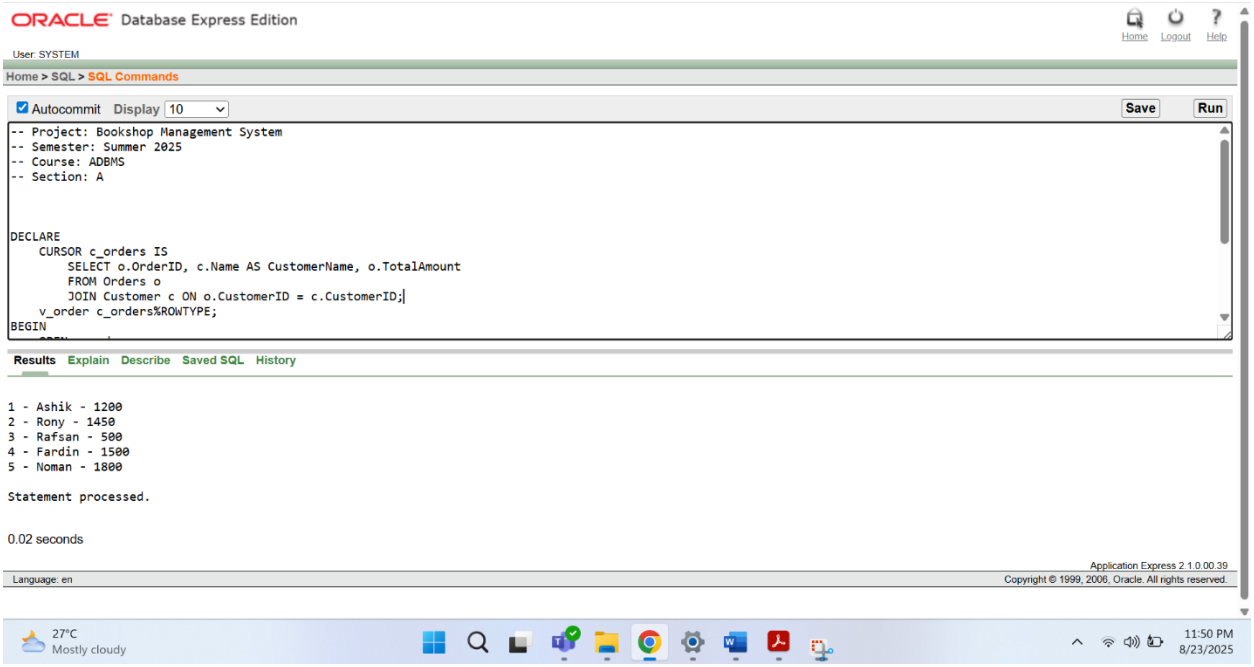
```
  v_books book_rec_type;
BEGIN
  SELECT * BULK COLLECT INTO v_books FROM Books;
  FOR i IN 1..v_books.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE(v_books(i).Title || ' - ' || v_books(i).Price);
  END LOOP;
END;
/
```



4. Explicit cursor & Cursor-based record

```
-- Project: Bookshop Management System
-- Semester: Summer 2025
-- Course: ADBMS
-- Section: A
```

```
DECLARE
  CURSOR c_orders IS
    SELECT o.OrderID, c.Name AS CustomerName, o.TotalAmount
    FROM Orders o
    JOIN Customer c ON o.CustomerID = c.CustomerID;
  v_order c_orders%ROWTYPE;
BEGIN
  OPEN c_orders;
  LOOP
    FETCH c_orders INTO v_order;
    EXIT WHEN c_orders%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_order.OrderID || ' - ' || v_order.CustomerName
|| ' - ' || v_order.TotalAmount);
  END LOOP;
  CLOSE c_orders;
END;
/
```



5. Explicit cursor & Cursor-based record

-- Project: Bookshop Management System

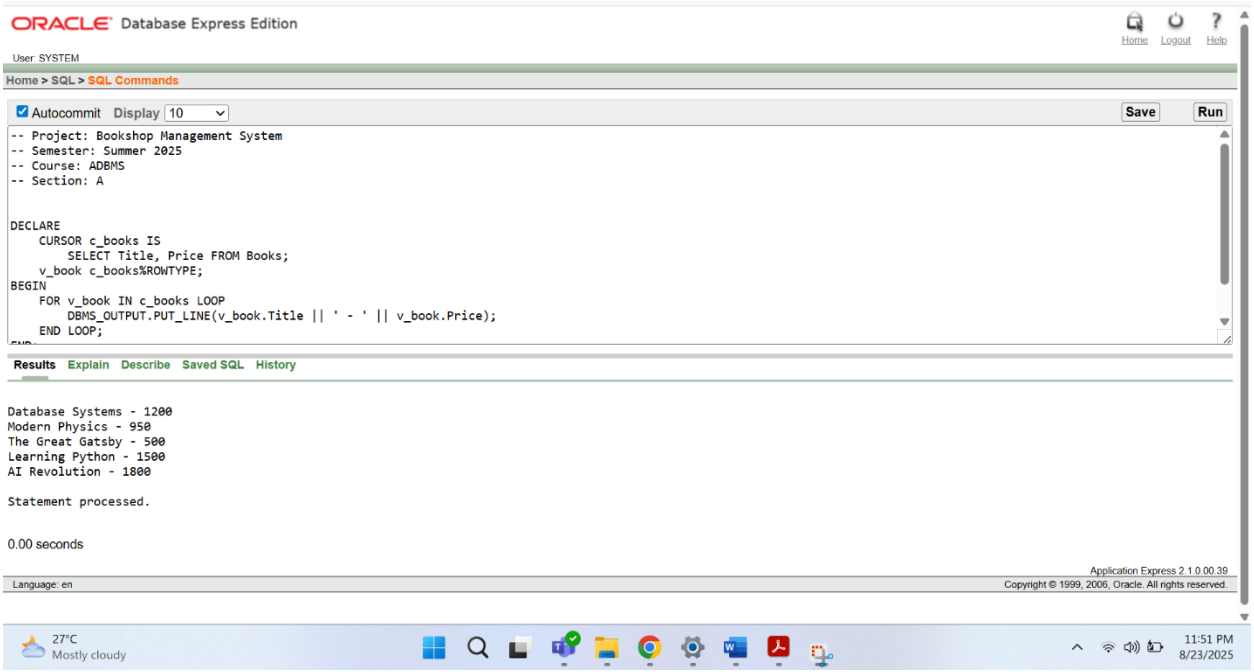
-- Semester: Summer 2025

-- Course: ADBMS

-- Section: A

```
DECLARE
   CURSOR c_books IS
      SELECT Title, Price FROM Books;
   v_book c_books%ROWTYPE;
BEGIN
   FOR v_book IN c_books LOOP
```

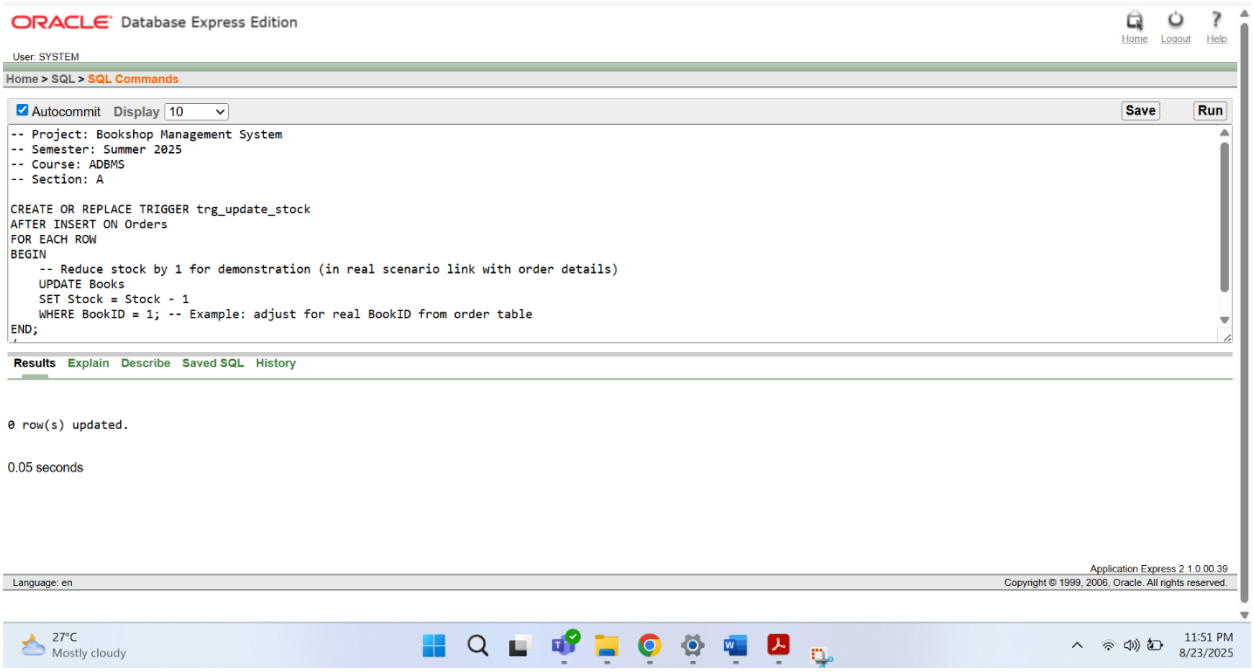
```
    DBMS_OUTPUT.PUT_LINE(v_book.Title || ' - ' || v_book.Price);
  END LOOP;
END;
/
```



6. Row level trigger

```
-- Project: Bookshop Management System
-- Semester: Summer 2025
-- Course: ADBMS
-- Section: A


CREATE OR REPLACE TRIGGER trg_update_stock
AFTER INSERT ON Orders
FOR EACH ROW
```

```
BEGIN
   -- Reduce stock by 1 for demonstration (in real scenario link with order details)
   UPDATE Books
   SET Stock = Stock - 1
   WHERE BookID = 1; -- Example: adjust for real BookID from order table
END;
/
```

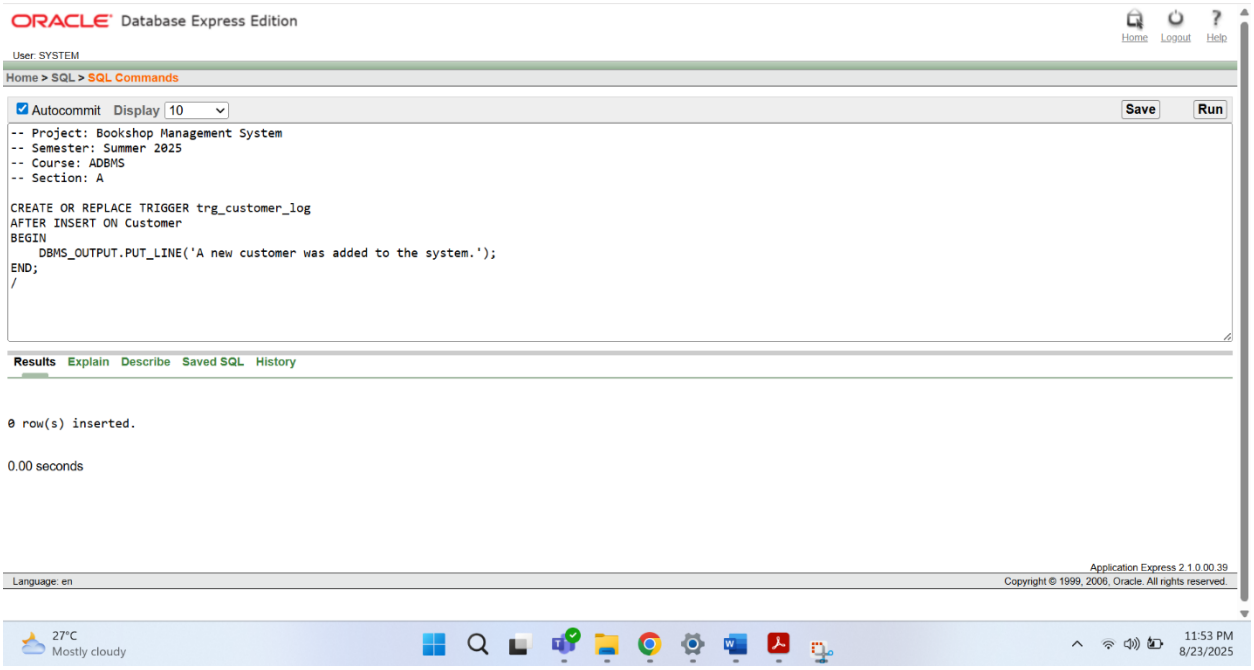


7. Statement level trigger

```
-- Project: Bookshop Management System
-- Semester: Summer 2025
-- Course: ADBMS
```

-- Section: A

```
CREATE OR REPLACE TRIGGER trg_customer_log
AFTER INSERT ON Customer
BEGIN
   DBMS_OUTPUT.PUT_LINE('A new customer was added to the system.');
END;
/
```



8. Package

```
-- Project: Bookshop Management System
-- Semester: Summer 2025
-- Course: ADBMS
```

```
-- Section: A

CREATE OR REPLACE PACKAGE sales_pkg IS
   FUNCTION total_orders RETURN NUMBER;
   PROCEDURE show_customers;
END sales_pkg;
/

CREATE OR REPLACE PACKAGE BODY sales_pkg IS

   FUNCTION total_orders RETURN NUMBER IS
      v_count NUMBER;
   BEGIN
      SELECT COUNT(*) INTO v_count FROM Orders;
      RETURN v_count;                    -- Return total orders
   END;

   PROCEDURE show_customers IS
   BEGIN
      FOR rec IN (SELECT Name, Email FROM Customer) LOOP
         DBMS_OUTPUT.PUT_LINE(rec.Name || ' - ' || rec.Email);
      END LOOP;                          -- Loop through all customers
   END;
```

END sales_pkg;

/