



AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

Summer 2024-25

**Final Term Project Documentation**

**BOOK STORE MANAGEMENT SYSTEM**

PROJECT REPORT  
Advanced Database Management System  
SECTION : A

NAME	ID	CONTRIBUTION
Mohammad Rafsan Yeasir	22-46065-1	30
MD.Fardin	21-45603-3	20
Asikqur Rahman	22-48971-3	20
Md. Asaduzzaman Rony	22-49057-3	30

# **Contents:**

Mid term feedback Solution

Senario Update .....	3
Normalization Update .....	3-8
Schema Diagram: (updated) .....	8
Advanced PL/SQL with Exception Handling .....	9-26
Relational Algebra Question and Answer .....	26-27

# Based on Mid Term Feedback solution

## Scenario updated

A bookstore deals with multiple operations, including book inventory, customer purchases, employee management, and payment handling. Without a proper system, manual processes cause stock mismatches, difficulties in tracking sales, and delays in customer service.

The **Bookstore Management System** solves these issues by storing and managing data about:

- **Books** (title, author, price, stock, category, publisher, supplier)
- **Customers** (personal details and order history)
- **Employees** (who handle orders)
- **Orders and OrderItems** (which track purchased books, quantity, subtotal)
- **Payments** (amount, method, date linked with orders)

When a customer places an order, it records the order date, books purchased, employee handling the sale, and total price. Stock levels are automatically updated. Payments are then recorded with method and date. This ensures smooth inventory management, accurate order handling, and reliable reporting for decision-making.

## Corrected Normalization (Updated)

---

### Belongs to (Books – Category)

**UNF:**

(BookID, Title, Author, Price, Stock, CategoryID, CategoryName)

**1NF:**

(BookID, Title, Author, Price, Stock, CategoryID, CategoryName)

**2NF:**

1. (BookID, Title, Author, Price, Stock, CategoryID)
2. (CategoryID, CategoryName)

**3NF:**

- Already in 3NF (no transitive dependency).

**Final Tables:**

1. Books(BookID, Title, Author, Price, Stock, CategoryID)
  2. Category(CategoryID, CategoryName)
- 

**Publishes (Books – Publisher)****UNF:**

(BookID, Title, Author, Price, Stock, PublisherID, PublisherName, Address, Phone, Email)

**1NF:**

(BookID, Title, Author, Price, Stock, PublisherID, PublisherName, Address, Phone, Email)

**2NF:**

1. (BookID, Title, Author, Price, Stock, PublisherID)
2. (PublisherID, PublisherName, Address, Phone, Email)

**3NF:**

- No transitive dependency, already in 3NF.

**Final Tables:**

1. Books(BookID, Title, Author, Price, Stock, PublisherID)
  2. Publisher(PublisherID, PublisherName, Address, Phone, Email)
- 

**Supplies (Books – Supplier)****UNF:**

(BookID, Title, Author, Price, Stock, SupplierID, SupplierName, ContactPerson, Phone, Email, Address)

**1NF:**

(BookID, Title, Author, Price, Stock, SupplierID, SupplierName, ContactPerson, Phone, Email, Address)

**2NF:**

1. (BookID, Title, Author, Price, Stock, SupplierID)

2. (SupplierID, SupplierName, ContactPerson, Phone, Email, Address)

**3NF:**

- No transitive dependency.

**Final Tables:**

1. Books(BookID, Title, Author, Price, Stock, SupplierID)

2. Supplier(SupplierID, SupplierName, ContactPerson, Phone, Email, Address)

---

**Appears in (Books – OrderItem)**

**UNF:**

(BookID, Title, Author, Price, Stock, OrderItemID, Quantity, SubTotal)

**1NF:**

(BookID, Title, Author, Price, Stock, OrderItemID, Quantity, SubTotal)

**2NF:**

1. (OrderItemID, BookID, Quantity, SubTotal)

2. (BookID, Title, Author, Price, Stock)

**3NF:**

- Book details moved to Books table, Order details remain separate.

**Final Tables:**

1. Books(BookID, Title, Author, Price, Stock)

2. OrderItem(OrderItemID, BookID, Quantity, SubTotal)

---

**Contains (Orders – OrderItem)**

**UNF:**

(OrderID, OrderDate, TotalAmount, OrderItemID, Quantity, SubTotal)

**1NF:**

(OrderID, OrderDate, TotalAmount, OrderItemID, Quantity, SubTotal)

**2NF:**

1. (OrderID, OrderDate, TotalAmount)

2. (OrderItemID, Quantity, SubTotal)

**3NF:**

- Already in 3NF.

**Final Tables:**

1. Orders(OrderID, OrderDate, TotalAmount)
  2. OrderItem(OrderItemID, OrderID, Quantity, SubTotal)
- 

**Handles (Orders – Employee)**

**UNF:**

(OrderID, OrderDate, TotalAmount, EmployeeID, EmployeeName, Role, Email, Phone)

**1NF:**

(OrderID, OrderDate, TotalAmount, EmployeeID, EmployeeName, Role, Email, Phone)

**2NF:**

1. (OrderID, OrderDate, TotalAmount, EmployeeID)
2. (EmployeeID, EmployeeName, Role, Email, Phone)

**3NF:**

- Already in 3NF.

**Final Tables:**

1. Orders(OrderID, OrderDate, TotalAmount, EmployeeID)
  2. Employee(EmployeeID, EmployeeName, Role, Email, Phone)
- 

**Places (Orders – Customer)**

**UNF:**

(OrderID, OrderDate, TotalAmount, CustomerID, CustomerName, Email, Phone, Address)

**1NF:**

(OrderID, OrderDate, TotalAmount, CustomerID, CustomerName, Email, Phone, Address)

**2NF:**

1. (OrderID, OrderDate, TotalAmount, CustomerID)

2. (CustomerID, CustomerName, Email, Phone, Address)

**3NF:**

- Already in 3NF.

**Final Tables:**

1. Orders(OrderID, OrderDate, TotalAmount, CustomerID)
  2. Customer(CustomerID, CustomerName, Email, Phone, Address)
- 

### **Paid (Orders – Payment)**

**UNF:**

(OrderID, OrderDate, TotalAmount, PaymentID, Amount, PaymentDate, Method)

**1NF:**

(OrderID, OrderDate, TotalAmount, PaymentID, Amount, PaymentDate, Method)

**2NF:**

1. (OrderID, OrderDate, TotalAmount)
2. (PaymentID, Amount, PaymentDate, Method, OrderID)

**3NF:**

- Already in 3NF.

**Final Tables:**

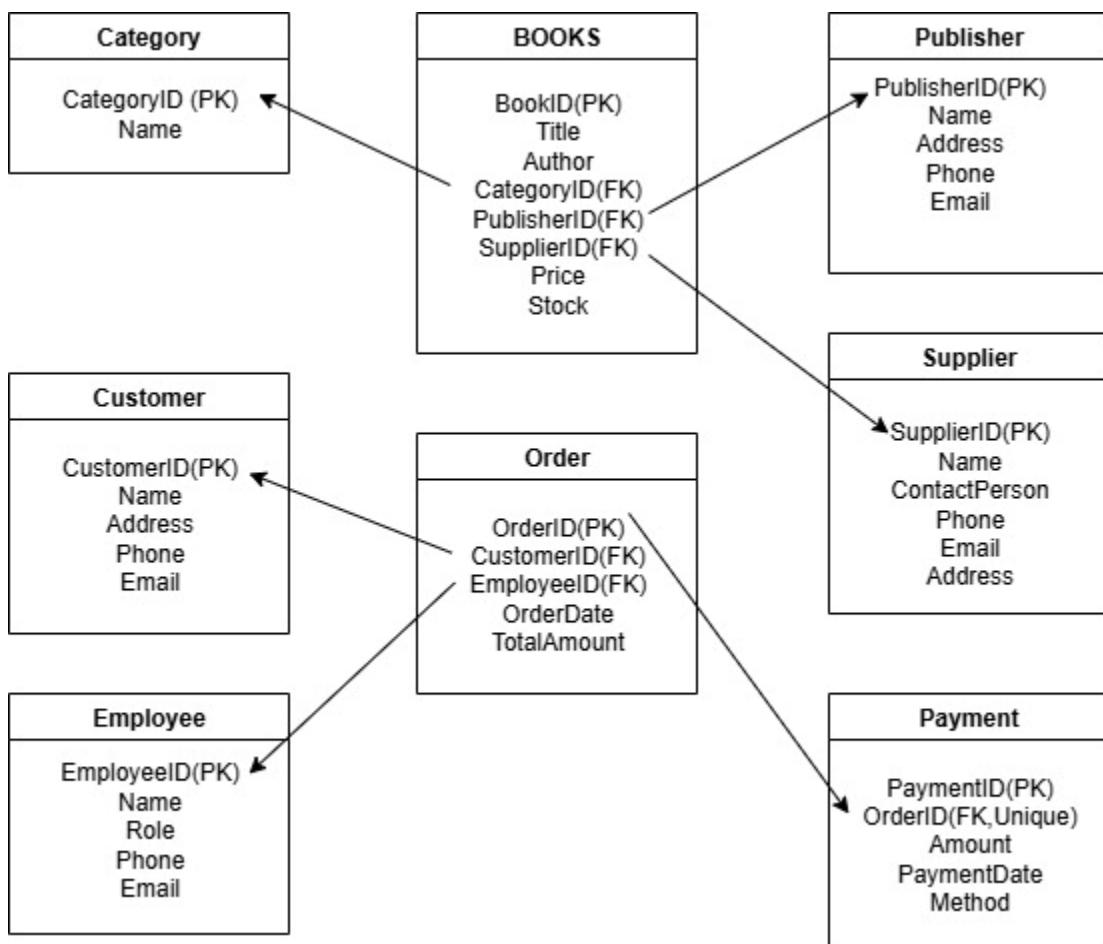
1. Orders(OrderID, OrderDate, TotalAmount)
  2. Payment(PaymentID, OrderID, Amount, PaymentDate, Method)
- 

### **Final Normalized Tables (Consolidated)**

1. Books(BookID, Title, Author, Price, Stock, CategoryID FK, PublisherID FK, SupplierID FK)
2. Category(CategoryID, CategoryName)
3. Publisher(PublisherID, PublisherName, Address, Phone, Email)
4. Supplier(SupplierID, SupplierName, 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, CustomerName, Email, Phone, Address)
9. Employee(EmployeeID, EmployeeName, Role, Email, Phone)

## Schema Diagram: (updated)



## Advance PL/SQL With Exception Handling

### Stored function:

-- Project: Bookshop Management System

-- Semester: Summer 2025

-- Course: ADBMS

-- Section: A

-- Stored Function: Get Book Price with Exception Handling

```
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;
```

```
EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
```

```
        DBMS_OUTPUT.PUT_LINE('No book found with ID: ' || p_bookid);
```

```
        RETURN NULL; -- or you could RETURN -1 as an indicator
```

```
    WHEN TOO_MANY_ROWS THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Multiple books found with ID: ' || p_bookid);
```

```
        RETURN NULL;
```

```
    WHEN OTHERS THEN
```

```

DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);

RETURN NULL;

END;

/

```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands tab, the following code is executed:

```

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

-- Stored Function: Get Book Price with Exception Handling
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;

```

The results show "Statement processed." and a time of "0.05 seconds". The bottom status bar indicates "Language: en" and "Application Express 21.0.0.39 Copyright © 1999, 2006, Oracle. All rights reserved." The system tray shows "1 cm of rain Today" and the date/time "12:05 AM 9/16/2025".

### Stored function:

-- Project: Bookshop Management System

-- Semester: Summer 2025

-- Course: ADBMS

-- Section: A

-- Stored Function: Get Book Stock with Exception Handling

```
CREATE OR REPLACE FUNCTION get_book_stock(p_bookid NUMBER)
```

RETURN NUMBER IS

v\_stock NUMBER;

BEGIN

SELECT Stock

```
INTO v_stock
FROM Books
WHERE BookID = p_bookid;

RETURN v_stock;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No book found with ID: ' || p_bookid);
    RETURN -1; -- return -1 to indicate missing book
  WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE('Data error: multiple books found with ID: ' || p_bookid);
    RETURN -2; -- return -2 to indicate data issue
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
    RETURN -99; -- generic error code
END;
/
```

ORACLE Database Express Edition

User SYSTEM

Home > SQL > SQL Commands

```

 Autocommit Display 10
Save Run
-- Project: Bookshop Management System
-- Semester: Summer 2025
-- Course: ADBMS
-- Section: A

-- Stored Function: Get Book Stock with Exception Handling
CREATE OR REPLACE FUNCTION get_book_stock(p_bookid NUMBER)
RETURN NUMBER IS
    v_stock NUMBER;
BEGIN
    SELECT Stock
    INTO v_stock
    FROM Books
    WHERE BookID = p_bookid;

```

Results Explain Describe Saved SQL History

Statement processed.

0.01 seconds

Language: en Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

Rain coming In about 1.5 hours 12:07 AM 9/16/2025

### **stored procedure :**

```
-- Stored Procedure: Calculate Total Sales
-- Project: Bookshop Management System
-- Semester: Summer 2025
-- Course: ADBMS
-- Section: A
```

```
CREATE OR REPLACE PROCEDURE get_total_sales(p_total OUT NUMBER)
IS
BEGIN
    SELECT NVL(SUM(TotalAmount), 0)
    INTO p_total
    FROM Orders;
```

EXCEPTION

```

WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No sales data found.');
    p_total := 0;
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
    p_total := -1; -- error indicator
END;
/

```

ORACLE Database Express Edition

User SYSTEM

Home > SQL > SQL Commands

Autocommit

-- Stored Procedure: Calculate Total Sales

```

CREATE OR REPLACE PROCEDURE get_total_sales(p_total OUT NUMBER)
IS
BEGIN
    SELECT NVL(SUM(TotalAmount), 0)
    INTO p_total
    FROM Orders;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No sales data found.');
        p_total := 0;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);

```

Results Explain Describe Saved SQL History

Statement processed.

0.02 seconds

Language: en Application Express 2.1.0.0.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

26°C Mostly cloudy 12:13 AM 9/16/2025

### Table-based record :

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

```

DECLARE
    -- Define a table-based record (PL/SQL collection)
    TYPE t_customer_rec IS TABLE OF Customer%ROWTYPE;
    v_customers t_customer_rec;

BEGIN
    -- Fetch all customers into the table-based record
    SELECT * BULK COLLECT INTO v_customers FROM Customer;

    -- Loop through each record
    FOR i IN 1 .. v_customers.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE('Customer ID: ' || v_customers(i).CustomerID ||
            ', Name: ' || v_customers(i).Name);
    END LOOP;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No customers found.');
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('More rows than expected.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/

```

ORACLE Database Express Edition

User SYSTEM

Home > SQL > **SQL Commands**

Autocommit Display 10

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

DECLARE
    -- Define a table-based record (PL/SQL collection)
    TYPE t_customer_rec IS TABLE OF Customer%ROWTYPE;
    v_customers t_customer_rec;
BEGIN
    -- Fetch all customers into the table-based record
    SELECT * BULK COLLECT INTO v_customers FROM Customer;

    -- Loop through each record
    FOR i IN 1..v_customers.COUNT LOOP
        dbms_output.put_line(v_customers(i));
    END LOOP;
END;
/

```

**Results Explain Describe Saved SQL History**

```
Customer ID: 1, Name: Ashik
Customer ID: 2, Name: Rony
Customer ID: 3, Name: Rafsan
Customer ID: 4, Name: Fardin
Customer ID: 5, Name: Noman

Statement processed.
```

0.02 seconds

Language: en Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

26°C Mostly cloudy 12:18 AM 9/16/2025

## Explicit Cursor :

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

```
DECLARE
```

```
-- Step 1: Declare the explicit cursor
```

```
CURSOR c_customer IS
```

```
SELECT CustomerID, Name, Email FROM Customer;
```

```
-- Step 2: Declare a record to hold each row fetched
```

```
v_customer c_customer%ROWTYPE;
```

```
BEGIN
```

```

-- Step 3: Open the cursor
OPEN c_customer;

-- Step 4: Fetch rows in a loop
LOOP
  FETCH c_customer INTO v_customer;
  EXIT WHEN c_customer%NOTFOUND; -- Exit loop when no more rows

-- Step 5: Process each row
  DBMS_OUTPUT.PUT_LINE('Customer ID: ' || v_customer.CustomerID ||
    ', Name: ' || v_customer.Name ||
    ', Email: ' || v_customer.Email);
END LOOP;

-- Step 6: Close the cursor
CLOSE c_customer;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No customers found.');
  WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE('Too many rows returned.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/

```

ORACLE Database Express Edition

User SYSTEM

Home > SQL > SQL Commands

```

 Autocommit Display 10
Save Run

-- Step 6: Close the cursor
CLOSE c_customer;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No customers found.');
  WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE('Too many rows returned.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/

```

**Results** Explain Describe Saved SQL History

```

Customer ID: 1, Name: Ashik, Email: ashik@gmail.com
Customer ID: 2, Name: Rony, Email: rony@gmail.com
Customer ID: 3, Name: Rafsan, Email: rafsan@gmail.com
Customer ID: 4, Name: Fardin, Email: fardin@gmail.com
Customer ID: 5, Name: Noman, Email: noman@gmail.com

Statement processed.

0.02 seconds

```

Language: en Application Express 2.1.0.00.39  
Copyright © 1999; 2006, Oracle. All rights reserved.

Rainy days ahead 26°C 12:20 AM 9/16/2025

## Cursor-based record :

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

DECLARE

- Step 1: Declare an explicit cursor

```
CURSOR c_customer IS
SELECT CustomerID, Name, Email FROM Customer;
```

- Step 2: Declare a record to hold each row from the cursor

```
v_customer c_customer%ROWTYPE;
```

BEGIN

```

-- Step 3: Open the cursor
OPEN c_customer;

-- Step 4: Fetch each row into the record and process
LOOP
  FETCH c_customer INTO v_customer;
  EXIT WHEN c_customer%NOTFOUND; -- Exit loop when no more rows

-- Step 5: Display the row data
  DBMS_OUTPUT.PUT_LINE('Customer ID: ' || v_customer.CustomerID ||
    ', Name: ' || v_customer.Name ||
    ', Email: ' || v_customer.Email);
END LOOP;

-- Step 6: Close the cursor
CLOSE c_customer;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No customers found.');
  WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE('Too many rows returned.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
/

```

The screenshot shows the Oracle Database Express Edition SQL Commands interface. The SQL code in the editor window is:

```

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

DECLARE
    -- Step 1: Declare an explicit cursor
    CURSOR c_customer IS
        SELECT CustomerID, Name, Email FROM Customer;
    -- Step 2: Declare a record to hold each row from the cursor
    v_customer c_customer%ROWTYPE;
BEGIN
    -- Step 3: Open the cursor
    OPEN c_customer;
    -- Fetch the first row
    FETCH c_customer INTO v_customer;
    -- Process the fetched row
    -- ...
END;

```

The results pane below shows the output of the query:

```

Customer ID: 1, Name: Ashik, Email: ashik@gmail.com
Customer ID: 2, Name: Rony, Email: rony@gmail.com
Customer ID: 3, Name: Rafsan, Email: rafsan@gmail.com
Customer ID: 4, Name: Fardin, Email: fardin@gmail.com
Customer ID: 5, Name: Noman, Email: noman@gmail.com
Statement processed.

0.00 seconds

```

At the bottom, the status bar indicates "Application Express 2.1 00.39" and "Copyright © 1999, 2006, Oracle. All rights reserved."

## Row Level Trigger:

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

```
CREATE OR REPLACE TRIGGER trg_before_insert_customer
```

```
BEFORE INSERT ON Customer
```

```
FOR EACH ROW -- Row-level trigger
```

```
DECLARE
```

```
v_dummy VARCHAR2(1); -- Just for demonstration
```

```
BEGIN
```

```
-- Example check: Email should not be null
```

```
IF :NEW.Email IS NULL THEN
```

```
RAISE_APPLICATION_ERROR(-20001, 'Email cannot be null.');
```

```
END IF;
```

```
-- Example: Assign a default value if Name is null
```

```
IF :NEW.Name IS NULL THEN  
    :NEW.Name := 'Unknown';  
END IF;
```

## EXCEPTION

```
WHEN OTHERS THEN
```

```
-- Handle any unexpected error  
DBMS_OUTPUT.PUT_LINE('Error in trigger: ' || SQLERRM);  
-- Optionally, re-raise the error to stop the DML  
RAISE;  
END;
```

```
/
```

The screenshot shows the Oracle Database Express Edition interface. The top navigation bar includes 'Home', 'Logout', and 'Help' buttons. The main window has a toolbar with various icons. The SQL command window displays the following code:

```
ORACLE Database Express Edition  
User: SYSTEM  
Home > SQL > SQL Commands  
Autocommit Display 10 Save Run  
-- Project: Bookshop Management System  
-- Semester: Summer 2025  
-- Course: ADBMS  
-- Section: A  
CREATE OR REPLACE TRIGGER trg_before_insert_customer  
BEFORE INSERT ON Customer  
FOR EACH ROW -- Row-level trigger  
DECLARE  
    v_dummy VARCHAR2(1); -- Just for demonstration  
BEGIN  
    -- Example check: Email should not be null  
    IF :NEW.Email IS NULL THEN  
        RAISE_APPLICATION_ERROR(-20001, 'Email cannot be null.');
```

The results pane below shows the output of the command:

```
Results Explain Describe Saved SQL History  
Unable to bind ":NEW"  
0 row(s) inserted.  
0.05 seconds
```

At the bottom, there is a footer with language information and copyright details:

```
Language: en Application Express 2.0.0.39  
Copyright © 1999, 2000, Oracle. All rights reserved.
```

**Statement level trigger:**

-- Project: Bookshop Management System

-- Semester: Summer 2025

-- Course: ADBMS

-- Section: A

-- Package to store counter

```
CREATE OR REPLACE PACKAGE customer_pkg IS
```

```
    cnt NUMBER := 0;
```

```
END;
```

```
/
```

-- Row-level trigger to increment counter

```
CREATE OR REPLACE TRIGGER trg_before_insert_customer
```

```
BEFORE INSERT ON Customer
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    customer_pkg.cnt := customer_pkg.cnt + 1;
```

```
END;
```

```
/
```

-- Statement-level trigger to log count

```
CREATE OR REPLACE TRIGGER trg_after_insert_customer
```

```
AFTER INSERT ON Customer
```

```
DECLARE
```

```
BEGIN
```

```
    INSERT INTO Customer_Audit(Audit_Date, Total_Customers)
```

```

VALUES (SYSDATE, customer_pkg.cnt);

-- Reset counter

customer_pkg.cnt := 0;

EXCEPTION

WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('Error in statement-level trigger: ' || SQLERRM);

RAISE;

END;
/

```

The screenshot shows the Oracle Database Express Edition interface. In the SQL Commands tab, the following PL/SQL code is run:

```

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

-- Package to store counter
CREATE OR REPLACE PACKAGE customer_pkg IS
    cnt NUMBER := 0;
END;
/

-- Row-level trigger to increment counter
CREATE OR REPLACE TRIGGER trg_before_insert_customer
BEFORE INSERT ON Customer
FOR EACH ROW
BEGIN
    NEW.cnt := cnt + 1;
    cnt := cnt + 1;
END;
/

```

The results pane shows:

- 0 row(s) inserted.
- 0.04 seconds

The status bar at the bottom right indicates Application Express 2.1.0.00.39, Copyright © 1999, 2006, Oracle. All rights reserved.

## Package:

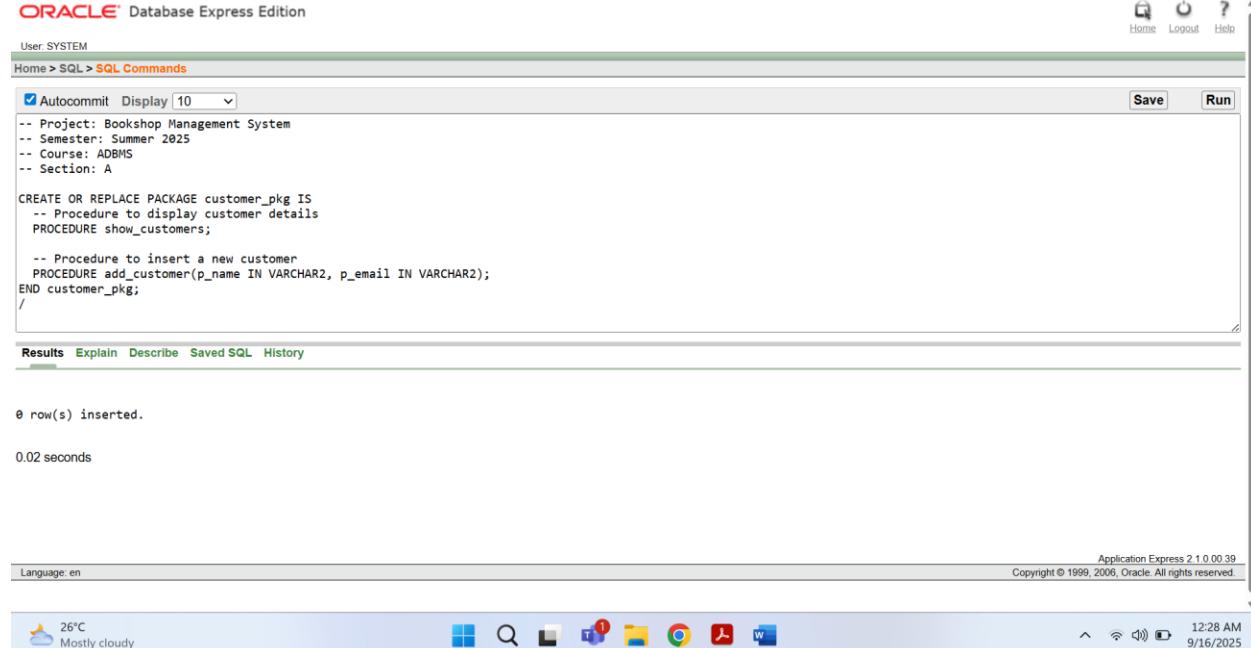
### --Package Specification

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

```
-- Section: A
```

```
CREATE OR REPLACE PACKAGE customer_pkg IS  
    -- Procedure to display customer details  
    PROCEDURE show_customers;  
  
    -- Procedure to insert a new customer  
    PROCEDURE add_customer(p_name IN VARCHAR2, p_email IN VARCHAR2);  
END customer_pkg;
```

```
/
```



```
ORACLE Database Express Edition  
User: SYSTEM  
Home > SQL > SQL Commands  
Save Run  
-- Project: Bookshop Management System  
-- Semester: Summer 2025  
-- Course: ADBMS  
-- Section: A  
CREATE OR REPLACE PACKAGE customer_pkg IS  
    -- Procedure to display customer details  
    PROCEDURE show_customers;  
  
    -- Procedure to insert a new customer  
    PROCEDURE add_customer(p_name IN VARCHAR2, p_email IN VARCHAR2);  
END customer_pkg;  
/  
  
Results Explain Describe Saved SQL History  
  
0 row(s) inserted.  
0.02 seconds  
  
Language: en Application Express 21.0.0.39  
Copyright © 1999, 2006, Oracle. All rights reserved.  
26°C Mostly cloudy 12:28 AM 9/16/2025
```

```
--BODY
```

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

```

CREATE OR REPLACE PACKAGE BODY customer_pkg IS

    -- Procedure to display all customers
    PROCEDURE show_customers IS
        CURSOR c_customer IS
            SELECT CustomerID, Name, Email FROM Customer;
            v_customer c_customer%ROWTYPE;
    BEGIN
        OPEN c_customer;
        LOOP
            FETCH c_customer INTO v_customer;
            EXIT WHEN c_customer%NOTFOUND;

            DBMS_OUTPUT.PUT_LINE('Customer ID: ' || v_customer.CustomerID ||
                ', Name: ' || v_customer.Name ||
                ', Email: ' || v_customer.Email);
        END LOOP;
        CLOSE c_customer;

        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                DBMS_OUTPUT.PUT_LINE('No customers found.');
            WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('Error in show_customers: ' || SQLERRM);
    END show_customers;

```

```
-- Procedure to add a new customer

PROCEDURE add_customer(p_name IN VARCHAR2, p_email IN VARCHAR2) IS
BEGIN
    INSERT INTO Customer(Name, Email)
    VALUES (p_name, p_email);

    DBMS_OUTPUT.PUT_LINE('Customer added successfully!');

EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Error: Customer with this email already exists.');
    WHEN VALUE_ERROR THEN
        DBMS_OUTPUT.PUT_LINE('Error: Invalid data provided.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error in add_customer: ' || SQLERRM);
END add_customer;

END customer_pkg;
/
```

ORACLE Database Express Edition

User SYSTEM

Home > SQL > SQL Commands

```

 Autocommit Display 10
-- Project: Bookshop Management System
-- Semester: Summer 2025
-- Course: ADBMS
-- Section: A

CREATE OR REPLACE PACKAGE BODY customer_pkg IS
    -- Procedure to display all customers
    PROCEDURE show_customers IS
        CURSOR c_customer IS
            SELECT CustomerID, Name, Email FROM Customer;
        v_customer c_customer%ROWTYPE;
    BEGIN
        OPEN c_customer;
        LOOP
            ...
        END LOOP;
        CLOSE c_customer;
    END;
END;

```

**Save** **Run**

Results Explain Describe Saved SQL History

0 row(s) inserted.

0.02 seconds

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

Language: en

26°C Mostly cloudy      12:28 AM 9/16/2025

## Relational Algebra Questions and Answers:

### 1. Question:

Find the names of all customers who have purchased a book priced over 500.

#### Relations:

- Customer(CustID, Name)
- Purchase(CustID, BookID)
- Book(BookID, Title, Price)

#### Answer:

$$\pi_{\text{Name}} (\text{Customer} \bowtie (\sigma_{\text{Price} > 500} \text{Book} \bowtie \text{Purchase}))$$

### 2. Question:

List the titles of books that have not been purchased by any customer.

#### Relations:

- Book(BookID, Title)
- Purchase(CustID, BookID)

**Answer:**

$$\pi_{\text{Title}}(\text{Book}) - \pi_{\text{Title}}(\text{Book} \bowtie \text{Purchase})$$
**3. Question:**

Find the IDs of customers who purchased all books in the category “Fiction”.

**Relations:**

- Customer(CustID, Name)
- Purchase(CustID, BookID)
- Book(BookID, Title, Category)

**Answer:**

$$\pi_{\text{CustID}}(\text{Customer}) \div \pi_{\text{BookID}}(\sigma_{\text{Category}=\text{'Fiction'}} \text{ Book})$$
**4. Question:**

Retrieve the names of customers who purchased both book with BookID = 101 and BookID = 102.

**Relations:**

- Customer(CustID, Name)
- Purchase(CustID, BookID)

**Answer:**

$$\pi_{\text{Name}} (\text{Customer} \bowtie (\pi_{\text{CustID}}(\sigma_{\text{BookID}=101} \text{ Purchase}) \cap \pi_{\text{CustID}}(\sigma_{\text{BookID}=102} \text{ Purchase})))$$
**5. Question:**

Find the titles and prices of books purchased by customer named “Alice”.

**Relations:**

- Customer(CustID, Name)
- Purchase(CustID, BookID)
- Book(BookID, Title, Price)

**Answer:**

$$\pi_{\text{Title}, \text{Price}} (\sigma_{\text{Name}=\text{'Alice'}} \text{ Customer} \bowtie \text{Purchase} \bowtie \text{Book})$$

