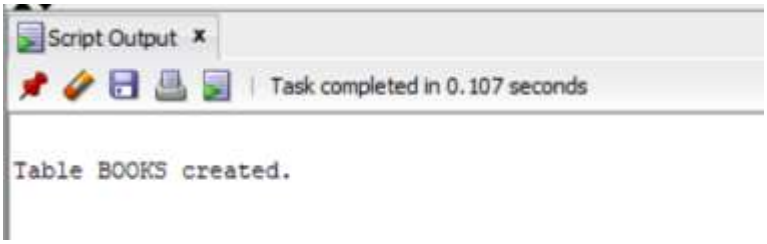
1. Creation of the Books table

Code

```
CREATE TABLE Books (
   BookID INT PRIMARY KEY,
   Title VARCHAR(100) NOT NULL,
   Author VARCHAR(50) NOT NULL,
   Price DECIMAL(10, 2) NOT NULL,
   ISBN VARCHAR(13) NOT NULL,
   PublishDate DATE NOT NULL);
```

Output

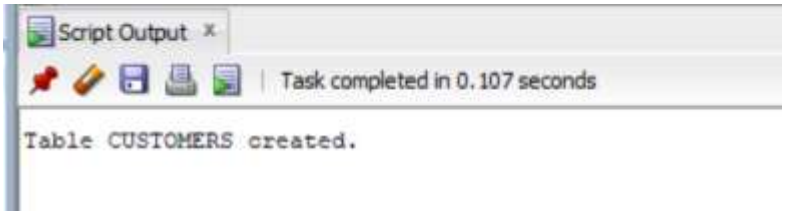


Creation of the Customers table

Code

```
CREATE TABLE Customers (
   CustomerID INT PRIMARY KEY,
   FirstName VARCHAR(20) NOT NULL,
   LastName VARCHAR(20) NOT NULL,
   Email VARCHAR(255) NOT NULL,
   JoinDate DATE NOT NULL);
```

Output

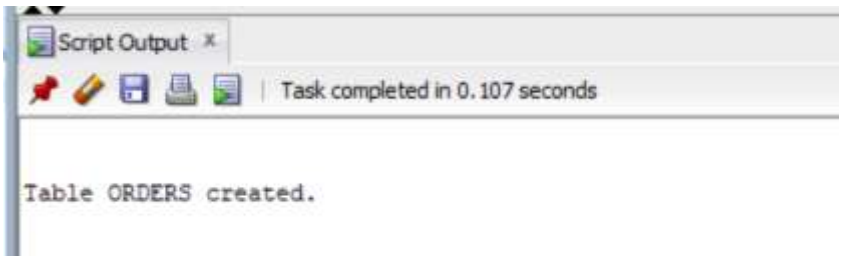


Creation of the Orders table

Code

```
CREATE TABLE Orders (
   OrderID INT PRIMARY KEY,
   CustomerID INT NOT NULL,
   BookID INT NOT NULL,
   Quantity INT NOT NULL,
   OrderDate DATE NOT NULL,
   FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
   FOREIGN KEY (BookID) REFERENCES Books(BookID));
```

Output

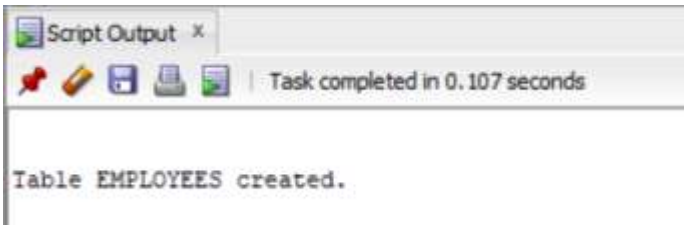


Creation of the Employees table

Code

```
CREATE TABLE Employees (
   EmployeeID INT PRIMARY KEY,
   FirstName VARCHAR(25) NOT NULL,
   LastName VARCHAR(25) NOT NULL,
   HireDate DATE NOT NULL,
   Email VARCHAR(255) NOT NULL
);
```

Output



2. Inserting 5 rows of data in the Books Table

Code

INSERT INTO Books (BookID, Title, Author, Price, ISBN, PublishDate) VALUES

(101, 'To Kill a Mockingbird', 'Harper Lee', 9.99, '9780060935467', TO_DATE('1960-07-11', 'YYYY-MM-DD'));

INSERT INTO Books (BookID, Title, Author, Price, ISBN, PublishDate) VALUES

(102, '1984', 'George Orwell', 8.99, '9780451524935', TO_DATE('1949-06-08', 'YYYY-MM-DD'));

INSERT INTO Books (BookID, Title, Author, Price, ISBN, PublishDate) VALUES

(103, 'The Great Gatsby', 'F. Scott Fitzgerald', 10.99, '9780743273565', TO_DATE('1925-04-10', 'YYYY-MM-DD'));

INSERT INTO Books (BookID, Title, Author, Price, ISBN, PublishDate) VALUES

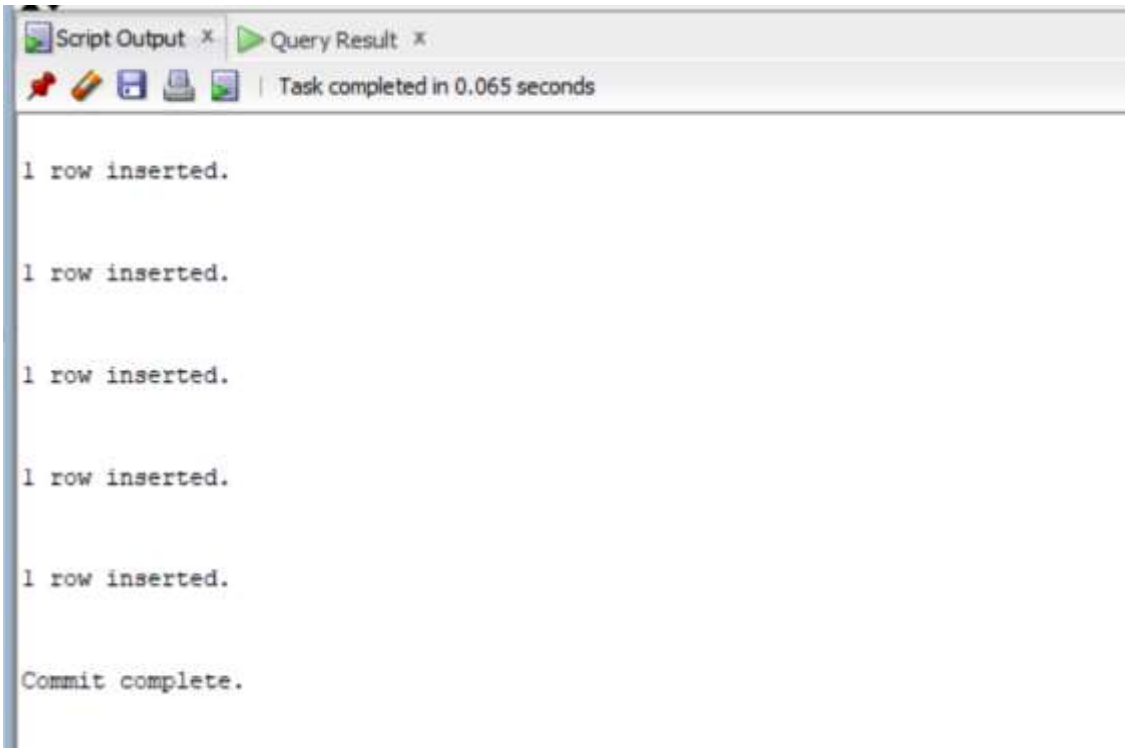
(104, 'Pride and Prejudice', 'Jane Austen', 6.99, '9781503290563', TO_DATE('1813-01-28', 'YYYY-MM-DD'));

INSERT INTO Books (BookID, Title, Author, Price, ISBN, PublishDate) VALUES

(105, 'The Catcher in the Rye', 'J.D. Salinger', 7.99, '9780316769488', TO_DATE('1951-07-16', 'YYYY-MM-DD'));

commit;

Output



Displaying the Books Table

Code

select * from Books;

Output

Script Output × Query Result ×

SQL | All Rows Fetched: 5 in 0.091 seconds

	BOOKID	TITLE	AUTHOR	PRICE	ISBN	PUBLISHDATE
1	101	To Kill a Mockingbird	Harper Lee	9.99	9780060935467	11-JUL-60
2	102	1984	George Orwell	8.99	9780451524935	08-JUN-49
3	103	The Great Gatsby	F. Scott Fitzgerald	10.99	9780743273565	10-APR-25
4	104	Pride and Prejudice	Jane Austen	6.99	9781503290563	28-JAN-13
5	105	The Catcher in the Rye	J.D. Salinger	7.99	9780316769488	16-JUL-51

Inserting 5 rows of data in the Customers Table

Code

INSERT INTO Customers (CustomerID, FirstName, LastName, Email, JoinDate) VALUES

(201, 'John', 'Doe', 'john.doe@example.com', TO_DATE('2022-01-15', 'YYYY-MM-DD'));

INSERT INTO Customers (CustomerID, FirstName, LastName, Email, JoinDate) VALUES

(202, 'Jane', 'Smith', 'jane.smith@example.com', TO_DATE('2022-02-20', 'YYYY-MM-DD'));

INSERT INTO Customers (CustomerID, FirstName, LastName, Email, JoinDate) VALUES

(203, 'Robert', 'Brown', 'robert.brown@example.com', TO_DATE('2022-03-25', 'YYYY-MM-DD'));

INSERT INTO Customers (CustomerID, FirstName, LastName, Email, JoinDate) VALUES

(204, 'Emily', 'Davis', 'emily.davis@example.com', TO_DATE('2022-04-30', 'YYYY-MM-DD'));

INSERT INTO Customers (CustomerID, FirstName, LastName, Email, JoinDate) VALUES

(205, 'Michael', 'Wilson', 'michael.wilson@example.com', TO_DATE('2022-05-05', 'YYYY-MM-DD'));

commit;

Output

```
Script Output x   Query Result x
Task completed in 0.06 seconds

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Commit complete.
```

Displaying the Customers Table

Code

select * from Customers;

Output

Script Output x Query Result x

All Rows Fetched: 5 in 0.004 seconds

	CUSTOMERID	FIRSTNAME	LASTNAME	EMAIL	JOINDATE
1	201	John	Doe	john.doe@example.com	15-JAN-22
2	202	Jane	Smith	jane.smith@example.com	20-FEB-22
3	203	Robert	Brown	robert.brown@example.com	25-MAR-22
4	204	Emily	Davis	emily.davis@example.com	30-APR-22
5	205	Michael	Wilson	michael.wilson@example.com	05-MAY-22

Inserting 5 rows of data into the Orders table

Code

INSERT INTO Orders (OrderID, CustomerID, BookID, Quantity, OrderDate) VALUES

(301, 201, 101, 2, TO_DATE('2023-01-10', 'YYYY-MM-DD'));

INSERT INTO Orders (OrderID, CustomerID, BookID, Quantity, OrderDate) VALUES

(302, 202, 103, 1, TO_DATE('2023-02-15', 'YYYY-MM-DD'));

INSERT INTO Orders (OrderID, CustomerID, BookID, Quantity, OrderDate) VALUES

(303, 203, 102, 3, TO_DATE('2023-03-20', 'YYYY-MM-DD'));

INSERT INTO Orders (OrderID, CustomerID, BookID, Quantity, OrderDate) VALUES

(304, 204, 104, 1, TO_DATE('2023-04-25', 'YYYY-MM-DD'));

INSERT INTO Orders (OrderID, CustomerID, BookID, Quantity, OrderDate) VALUES

(305, 205, 105, 2, TO_DATE('2023-05-30', 'YYYY-MM-DD'));

commit;

Output

Displaying the Orders table

Code

select * from Orders;

Output

Script Output × Query Result ×

SQL | All Rows Fetched: 5 in 0.006 seconds

	ORDERID	CUSTOMERID	BOOKID	QUANTITY	ORDERDATE
1	301	201	101	2	10-JAN-23
2	302	202	103	1	15-FEB-23
3	303	203	102	3	20-MAR-23
4	304	204	104	1	25-APR-23
5	305	205	105	2	30-MAY-23

Inserting 5 rows of data into the Employees table

Code

```
INSERT INTO Employees (EmployeeID, FirstName, LastName, HireDate, Email) VALUES
(401, 'Alice', 'Johnson', TO_DATE('2021-01-01', 'YYYY-MM-DD'),'alice.johnson@example.com');
INSERT INTO Employees (EmployeeID, FirstName, LastName, HireDate, Email) VALUES
(402, 'Bob', 'Williams', TO_DATE('2021-02-01', 'YYYY-MM-DD'),'bob.williams@example.com');
INSERT INTO Employees (EmployeeID, FirstName, LastName, HireDate, Email) VALUES
(403, 'Charlie', 'Jones', TO_DATE('2021-03-01', 'YYYY-MM-DD'),'charlie.jones@example.com');
INSERT INTO Employees (EmployeeID, FirstName, LastName, HireDate, Email) VALUES
(404, 'Diana', 'Miller', TO_DATE('2021-04-01', 'YYYY-MM-DD'),'diana.miller@example.com');
INSERT INTO Employees (EmployeeID, FirstName, LastName, HireDate, Email) VALUES
(405, 'Edward', 'Garcia', TO_DATE('2021-05-01', 'YYYY-MM-DD'),'edward.garcia@example.com');
commit;
```

Output

Script Output × Query Result ×

Task completed in 0.046 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Commit complete.
```

Displaying the Employees table

Code

select * from Employees;

Output

Script Output × Query Result ×

SQL | All Rows Fetched: 5 in 0.005 seconds

	EMPLOYEEID	FIRSTNAME	LASTNAME	HIREDATE	EMAIL
1	401	Alice	Johnson	01-JAN-21	alice.johnson@example.com
2	402	Bob	Williams	01-FEB-21	bob.williams@example.com
3	403	Charlie	Jones	01-MAR-21	charlie.jones@example.com
4	404	Diana	Miller	01-APR-21	diana.miller@example.com
5	405	Edward	Garcia	01-MAY-21	edward.garcia@example.com

3.Stored Procedures

a) Creation of the stored procedure “AddNewOrder” and Execution of the procedure to display output

Code

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE AddNewOrder (
   p_OrderID IN NUMBER,
   p_CustomerID IN NUMBER,
   p_BookID IN NUMBER,
   p_Quantity IN NUMBER
)
AS
BEGIN
   INSERT INTO Orders (OrderID,CustomerID, BookID, Quantity, OrderDate)
   VALUES (p_OrderID,p_CustomerID, p_BookID, p_Quantity, SYSDATE);
   COMMIT;
   DBMS_OUTPUT.PUT_LINE('Order placed successfully.');
END;
```

```
BEGIN
   AddNewOrder(306,204,102,5);
END;
```

Output

```
Script Output x   Query Result x
Task completed in 0.115 seconds

Procedure ADDNEWORDER compiled

Order placed successfully.


PL/SQL procedure successfully completed.
```

Confirming the new order details added for Order ID 306

Code

```
select * from Orders;
```

Output

Script Output x Query Result x

SQL | All Rows Fetched: 6 in 0.003 seconds

	ORDERID	CUSTOMERID	BOOKID	QUANTITY	ORDERDATE
1	301	201	101	2	10-JAN-23
2	302	202	103	1	15-FEB-23
3	303	203	102	3	20-MAR-23
4	304	204	104	1	25-APR-23
5	305	205	105	2	30-MAY-23
6	306	204	102	5	07-AUG-24

b) Creation of the stored procedure “UpdateBookPrice” and Execution of the procedure to display output

Code

```
set serveroutput on;
CREATE OR REPLACE PROCEDURE UpdateBookPrice (
   p_BookID IN NUMBER,
   p_NewPrice IN NUMBER )
```

```
AS
BEGIN
   UPDATE Books
   SET Price = p_NewPrice
   WHERE BookID = p_BookID;
   COMMIT;
   DBMS_OUTPUT.PUT_LINE('Book price updated successfully.');
END;
BEGIN
   UpdateBookPrice(103,20.5);
END;
```

Output

Script Output × | Query Result ×

Task completed in 0.116 seconds

```
Procedure UPDATEBOOKPRICE compiled

Book price updated successfully.


PL/SQL procedure successfully completed.
```

Display of the Books table to Confirm the updated price for book ID 103

Code

```
select * from Books;
```

Output

Script Output × | Query Result ×

SQL | All Rows Fetched: 5 in 0.003 seconds

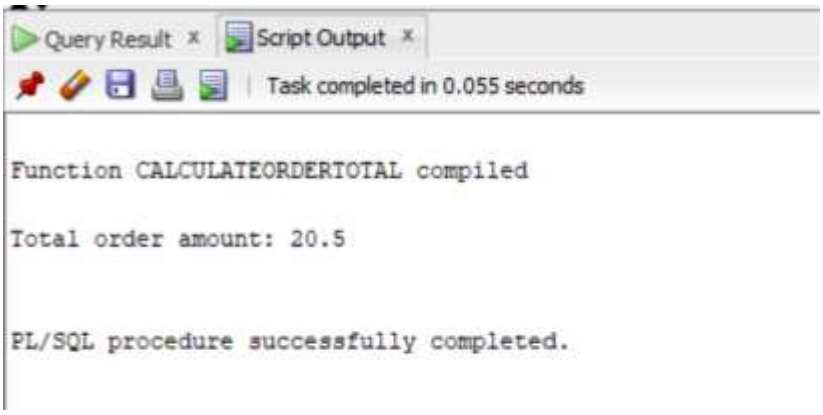
	BOOKID	TITLE	AUTHOR	PRICE	ISBN	PUBLISHDATE
1	101	To Kill a Mockingbird	Harper Lee	9.99	9780060935467	11-JUL-60
2	102	1984	George Orwell	8.99	9780451524935	08-JUN-49
3	103	The Great Gatsby	F. Scott Fitzgerald	20.5	9780743273565	10-APR-25
4	104	Pride and Prejudice	Jane Austen	6.99	9781503290563	28-JAN-13
5	105	The Catcher in the Rye	J.D. Salinger	7.99	9780316769488	16-JUL-51

4 a) Creation and Compiling a Function “CalculateOrderTotal” and displaying the Total amount for Order ID 302

Code

```
set serveroutput on;

CREATE OR REPLACE FUNCTION CalculateOrderTotal (

   p_OrderID IN NUMBER

)

RETURN NUMBER

IS

   total NUMBER;

BEGIN

   SELECT SUM(b.Price * o.Quantity)

   INTO total

   FROM Orders o

   JOIN Books b ON o.BookID = b.BookID

   WHERE o.OrderID = p_OrderID;

   RETURN total;

END;

DECLARE

   total_amount NUMBER;

BEGIN

   total_amount := CalculateOrderTotal(302);

   DBMS_OUTPUT.PUT_LINE('Total order amount: ' || total_amount);

END;
```

Output



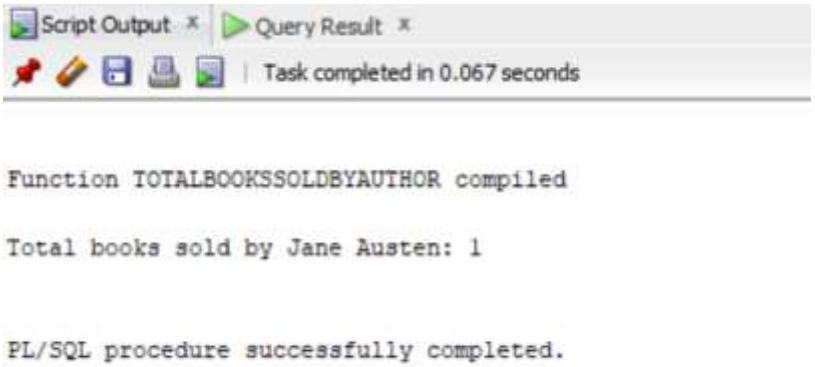
b) Creating and Compiling a Function “TotalBooksSoldByAuthor” and displaying the number of books sold by an Author “Jane Austen”

Code

```
CREATE OR REPLACE FUNCTION TotalBooksSoldByAuthor (
   p_AuthorName IN VARCHAR2
)
RETURN NUMBER
IS
   total NUMBER;
BEGIN
   SELECT SUM(o.Quantity)
   INTO total
   FROM Orders o
   JOIN Books b ON o.BookID = b.BookID
   WHERE b.Author = p_AuthorName;
   RETURN total;
END;

DECLARE
   total_sold NUMBER;
BEGIN
```

```
    total_sold := TotalBooksSoldByAuthor('Jane Austen');
    DBMS_OUTPUT.PUT_LINE('Total books sold by Jane Austen: ' || total_sold);
END;
```

Output



5. Queries

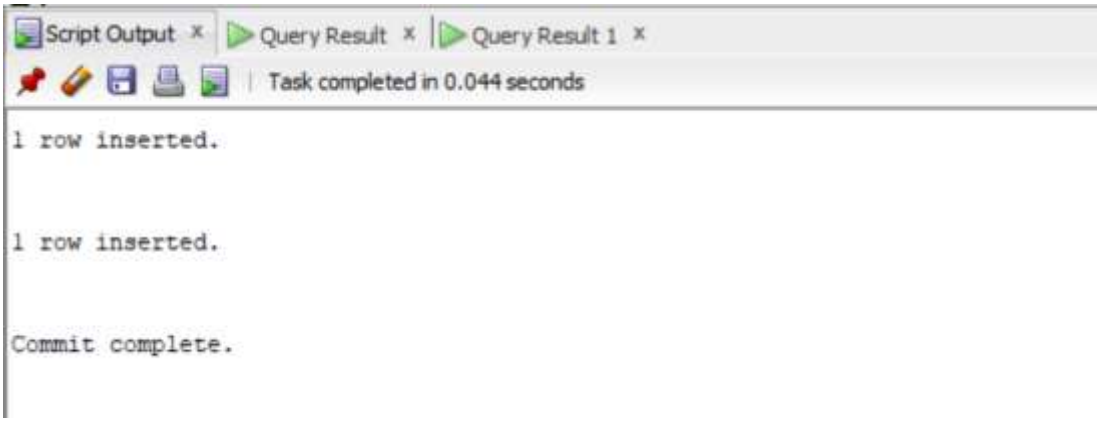
i) SQL Query to Retrieve all books published after 2020.

First I Inserted 2 more rows in the Books Table

Code

```
INSERT INTO Books (BookID, Title, Author, Price, ISBN, PublishDate)
VALUES (106, 'The Midnight Library', 'Matt Haig', 14.99, '9781786892737', TO_DATE('2020-09-29', 'YYYY-MM-DD'));
INSERT INTO Books (BookID, Title, Author, Price, ISBN, PublishDate)
VALUES (107, 'Klara and the Sun', 'Kazuo Ishiguro', 16.99, '9780571364886', TO_DATE('2021-03-02', 'YYYY-MM-DD'));
commit;
```

Output



Then I executed an SQL query to display the book details that are published after 2020

Code

```
SELECT * FROM Books

WHERE PublishDate > TO_DATE('2020-01-01', 'YYYY-MM-DD');
```

Output

All Rows Fetched: 2 in 0.002 seconds

	BOOKID	TITLE	AUTHOR	PRICE	ISBN	PUBLISHDATE
1	106	The Midnight Library	Matt Haig	14.99	9781786892737	29-SEP-20
2	107	Klara and the Sun	Kazuo Ishiguro	16.99	9780571364886	02-MAR-21

ii)Query to List customers who have placed more than 5 orders

First I Inserted 6 more rows in the Orders table

Code

```
INSERT INTO Orders (OrderID, CustomerID, BookID, Quantity, OrderDate) VALUES

(307, 204, 103, 8, TO_DATE('2020-01-15', 'YYYY-MM-DD'));

INSERT INTO Orders (OrderID, CustomerID, BookID, Quantity, OrderDate) VALUES

(308, 204, 105, 3, TO_DATE('2013-02-20', 'YYYY-MM-DD'));

INSERT INTO Orders (OrderID, CustomerID, BookID, Quantity, OrderDate) VALUES

(309, 204, 106, 5, TO_DATE('2022-06-15', 'YYYY-MM-DD'));

INSERT INTO Orders (OrderID, CustomerID, BookID, Quantity, OrderDate) VALUES

(310, 204, 101, 10, TO_DATE('2019-04-04', 'YYYY-MM-DD'));

INSERT INTO Orders (OrderID, CustomerID, BookID, Quantity, OrderDate) VALUES

(311, 204, 102, 1, TO_DATE('2023-03-21', 'YYYY-MM-DD'));

INSERT INTO Orders (OrderID, CustomerID, BookID, Quantity, OrderDate) VALUES

(312, 204, 107, 4, TO_DATE('2023-08-30', 'YYYY-MM-DD'));

commit;
```

Output

Script Output × | Query Result ×

Task completed in 0.046 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Commit complete.
```

Then I displayed the Orders table to confirm the inserted rows

Code

```
SELECT c.CustomerID, c.FirstName, c.LastName, COUNT(o.OrderID) AS OrderCount
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
GROUP BY c.CustomerID, c.FirstName, c.LastName
HAVING COUNT(o.OrderID) > 1;
```

Output

All Rows Fetched: 12 in 0.002 seconds

	ORDERID	CUSTOMERID	BOOKID	QUANTITY	ORDERDATE
1	301	201	101	2	10-JAN-23
2	302	202	103	1	15-FEB-23
3	303	203	102	3	20-MAR-23
4	304	204	104	1	25-APR-23
5	305	205	105	2	30-MAY-23
6	306	204	102	5	07-AUG-24
7	307	204	103	8	15-JAN-20
8	308	204	105	3	20-FEB-13
9	309	204	106	5	15-JUN-22
10	310	204	101	10	04-APR-19
11	311	204	102	1	21-MAR-23
12	312	204	107	4	30-AUG-23

Executed the Query to List the customers with more than 5 orders

Code

```
SELECT c.CustomerID, c.FirstName, c.LastName, COUNT(o.OrderID) AS OrderCount
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
GROUP BY c.CustomerID, c.FirstName, c.LastName
HAVING COUNT(o.OrderID) > 5;
```

Output

All Rows Fetched: 1 in 0.005 seconds

	CUSTOMERID	FIRSTNAME	LASTNAME	ORDERCOUNT
1	204	Emily	Davis	8

iii)Query to Find the employee who has been with the company the longest.

Executed Query to display who has been with the company the longest

Code

```
SELECT *
FROM Employees
WHERE HireDate = (SELECT MIN(HireDate) FROM Employees);
```

Output

All Rows Fetched: 1 in 0.011 seconds

	EMPLOYEEID	FIRSTNAME	LASTNAME	HIREDATE	EMAIL
1	401	Alice	Johnson	01-JAN-21	alice.johnson@example.com

7. Dynamic SQL Implementation by using the "SearchBooksDynamic" procedure

Code

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE SearchBooksDynamic (
   p_SearchType IN VARCHAR2,
   p_SearchValue IN VARCHAR2
)
AS
   v_sql VARCHAR2(4000);
   v_BookID Books.BookID%TYPE;
   v_Title Books.Title%TYPE;
   v_Author Books.Author%TYPE;
   v_Price Books.Price%TYPE;
   v_ISBN Books.ISBN%TYPE;
   v_PublishDate Books.PublishDate%TYPE;
   CURSOR c IS
      SELECT BookID, Title, Author, Price, ISBN, PublishDate
      FROM Books
      WHERE CASE p_SearchType
            WHEN 'Author' THEN Author
            WHEN 'Title' THEN Title
            WHEN 'PublishYear' THEN TO_CHAR(PublishDate, 'YYYY')
            END = p_SearchValue;
BEGIN
   OPEN c;
   LOOP
      FETCH c INTO v_BookID, v_Title, v_Author, v_Price, v_ISBN, v_PublishDate;
      EXIT WHEN c%NOTFOUND;

      DBMS_OUTPUT.PUT_LINE('BookID: ' || v_BookID);
```

```
    DBMS_OUTPUT.PUT_LINE('Title: ' || v_Title);
    DBMS_OUTPUT.PUT_LINE('Author: ' || v_Author);
    DBMS_OUTPUT.PUT_LINE('Price: ' || v_Price);
    DBMS_OUTPUT.PUT_LINE('ISBN: ' || v_ISBN);
    DBMS_OUTPUT.PUT_LINE('Publish Date: ' || TO_CHAR(v_PublishDate, 'YYYY-MM-DD'));
    DBMS_OUTPUT.PUT_LINE('-----------------------------------');
  END LOOP;
  CLOSE c;
END;
```

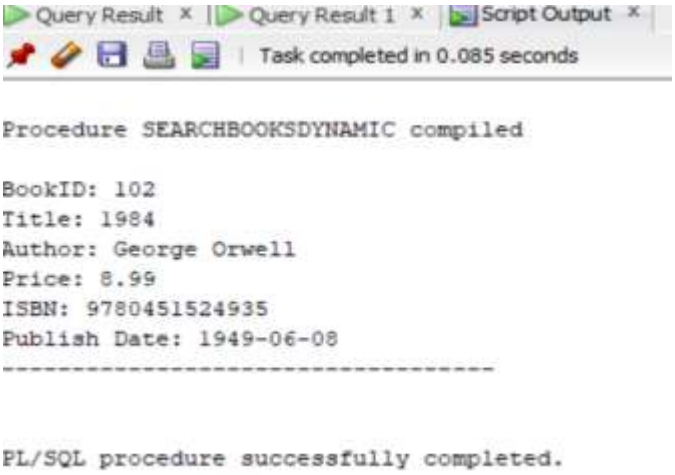
Output for Creating the procedure “SearchBooksDynamic” and compiling it to output the Book details basing on

i) Search Type - Author

Code

```
BEGIN
   SearchBooksDynamic('Author','George Orwell');

END;
```

Output



ii) Search Type - Title

Code

```
BEGIN
   SearchBooksDynamic('Title','The Midnight Library');

END;
```

Output