****

**Book Store Database Using Oracle PL/SQL**

**End Module Group Assessment**

**Higher National Diploma in Software Engineering-23.1**

**Advanced Database Management System**

**Group Members:**

**COHDSE231F-005 M.D.S.Probodya**

**COHDSE231F-006 K.D.R.Dulaksha**

**COHDSE231F-007 J.J.Sooriarachchi**

**COHDSE231F-025 B.A.D.I.S.Samaranayke**

**COHDSE231F-056 P.A.P. Premalal**

**National Institute of Business Management – School of Computing**

**Introduction to Real-World Business Organization**

With the evolution of society many businesses have moved mostly towards efficient data management methods. When selecting a real-world business scenario, we have chosen to create an oracle database for a bookstore. This makes it much easier for the shop owner as all the details are stored in a well-structured manner. Also, this method is useful when the demand is increasing.

The database is the backbone of the business as it stores all the needed information of the business. So, we have done a case study and identified the needs of the user. Mainly we have identified the bookstore’s need to manage the inventory, provide the needed information to the customers, and keep track of sales and orders. According to those needs we clarified the functions that we should fulfill through database. The main functions we identified after analyzing the business scenario are as below.

* Managing the information about books.
* Managing the information about authors.
* Managing the information about customer orders.

As we proceeded through the scenario, we created needed tables, inserted matching records to the tables and performed different queries to retrieve different tasks. For the implementation of the following queries, we have used Oracle PL/SQL blocks. As the software we have used Oracle SQL developer.

1. **Create tables with primary key and foreign key constraints having auto-increment sequences for one of the tables.**

According to our scenario we created three tables as “Book”, “Order”, and “Author”.

**Book Table**

|  |  |
| --- | --- |
| **Column Name** | **Variable Type** |
| BookID (Primary Key) | Varchar2(5) |
| Title | Varchar2(30) |
| AuthorID (Foreign Key) | Varchar2(5) |
| Price | Number(5) |

**Customer\_Order Table**

|  |  |
| --- | --- |
| Column Name | Variable Type |
| OrderID (Primary Key) | Varchar2(5) |
| CustomerName | Varchar2(30) |
| BookID(Foreign Key) | Varchar2(5) |
| PurchaseDate | Date |

**Author Table**

|  |  |
| --- | --- |
| **Column Name** | **Variable Type** |
| AuthorID (Primary Key) | Varchar2(5) |
| AuthorName | Varchar2(30) |

All the tables are relational through the foreign keys. And auto incrementing has been used in author table, AuthorID.

**Code**

create table Author(AuthorID varchar2(5) primary key, AuthorName varchar2(30) not null);

create table Book(BookID varchar2(5) primary key, Title varchar2(30) not null,AuthorID varchar(5) References Author(AuthorID) ,Price number(5) not null);

create table Customer\_Order(OrderID varchar2(5) primary key, CustomerName varchar2(30) not null, BookID varchar2(5) references Book(BookID), PurchaseDate date not null);

create sequence author\_seq;

select author\_seq.NEXTVAL FROM dual;

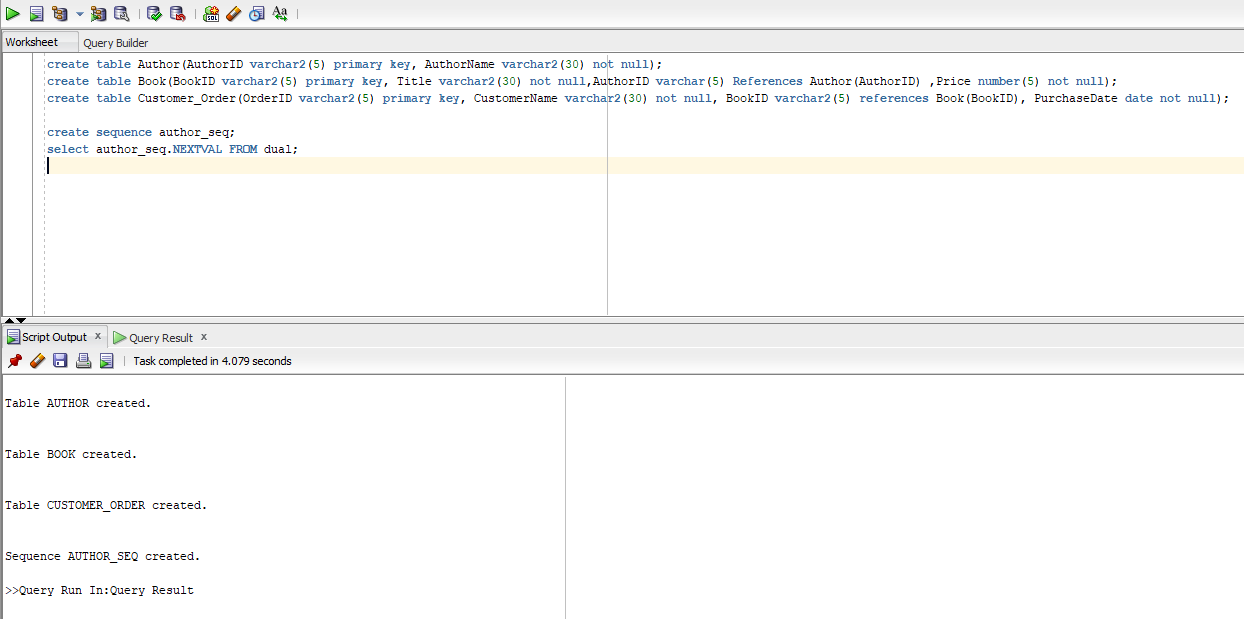
**Output**

Figure 1- Creating tables and sequences.

1. **Insert a set of matching records for the above tables.**

Firstly, we entered data into the author table as given below.

**Code**

insert into Author values (author\_seq.NEXTVAL, 'Jeff Kinney');

insert into Author values (author\_seq.NEXTVAL, 'Mark Twain');

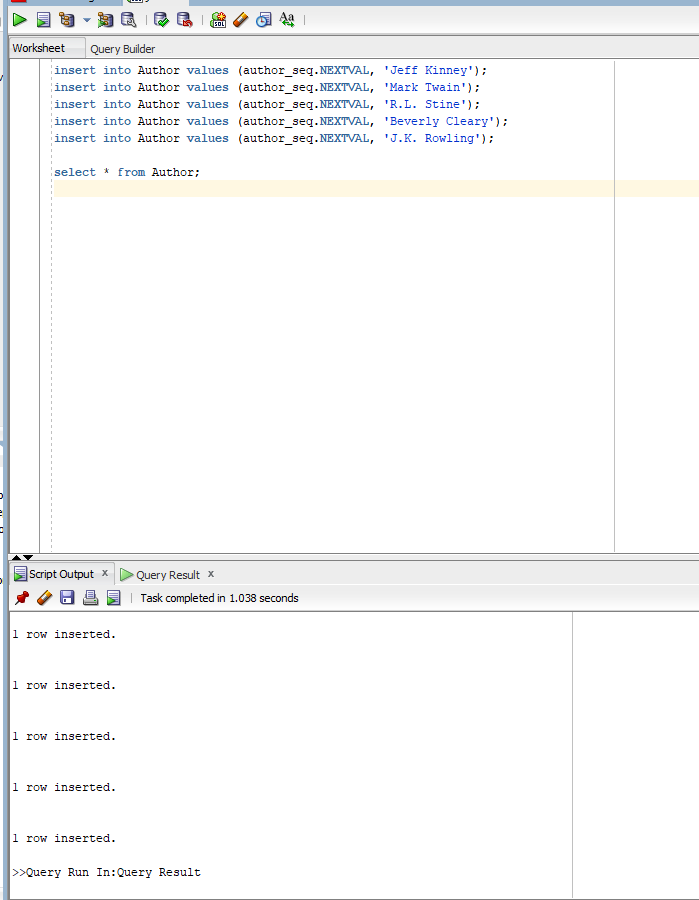
insert into Author values (author\_seq.NEXTVAL, 'R.L. Stine');

insert into Author values (author\_seq.NEXTVAL, 'Beverly Cleary');

insert into Author values (author\_seq.NEXTVAL, 'J.K. Rowling');

select \* from Author;

All the data inserted to the Author table through the insert queries. Select query is used to view the entered details of the Author table.



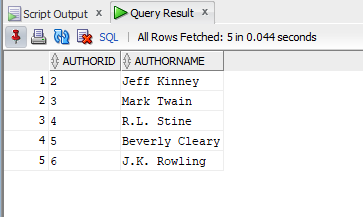


Figure 2 - Author table

Figure 3 - Inserting data to Author table

Data insertion to the ‘Book’ table

**Code**

insert into Book values (476, 'Ramona and the Pest','5',700);

insert into Book values (005, 'Toy Terror - Goosebumps','4',750);

insert into Book values (890, 'The Prince and the Pauper','3',300);

insert into Book values (145, 'DWK - Big Shot','2',950);

insert into Book values (009, 'Harry Potter Cursed Child','6',1550);

insert into Book values (897, 'Ramona the Brave','5',750);

insert into Book values (458, 'Beezus and Ramona','5',700);

select \* from Book;

**Output**

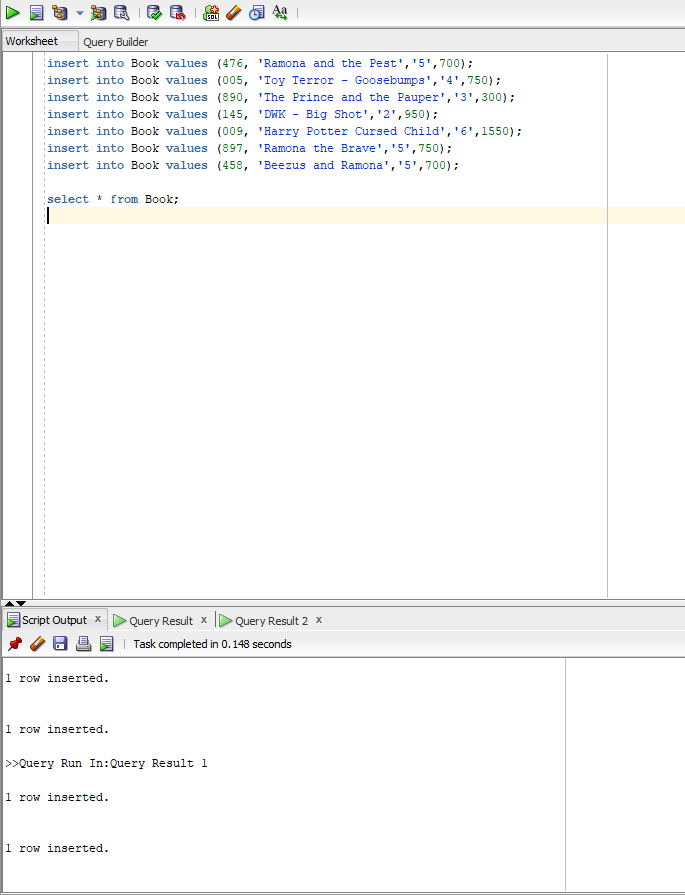


Figure 4 - Inserting data into Book table

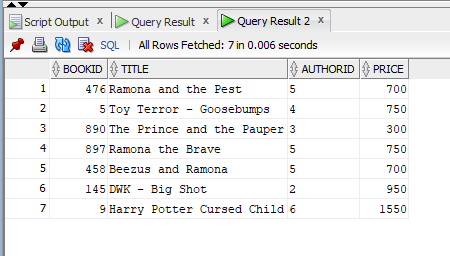


Figure 5 - Book table

Data insertion to the ‘Customer\_Order’ table

**Code**

insert into Customer\_order values (1, 'Nethmi','476','02-Feb-2023');

insert into Customer\_order values (2, 'Amal','476','15-Jan-2023');

insert into Customer\_order values (3, 'Perera','476','26-Jun-2023');

insert into Customer\_order values (4, 'Nimal','897','01-Mar-2023');

insert into Customer\_order values (5, 'Thilini','9','02-Feb-2023');

insert into Customer\_order values (6, 'Kaveesha','458','28-Feb-2023');

select \* from Customer\_Order;

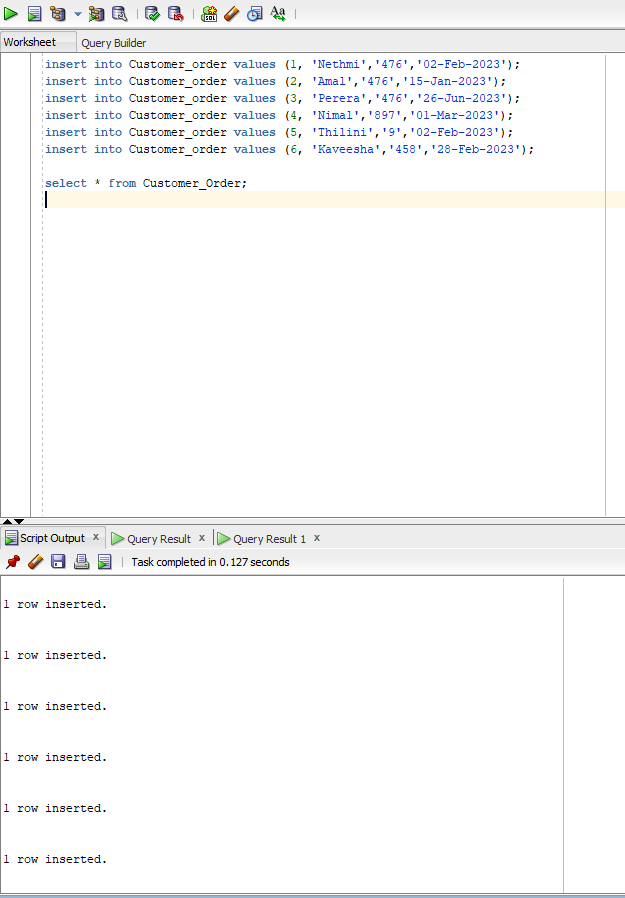
**Output**

Figure 6 - Inserting data into Customer\_Order table

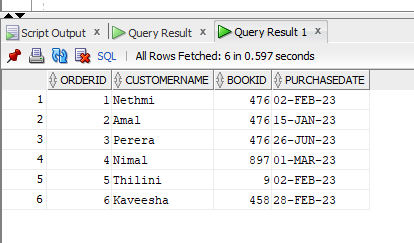


Figure 7 - Customer\_Order table

1. **Write any select queries each using where, group by, having, and order by.**

When we want to display the purchase count of the books in the Customer\_Order table except the book with id 458 we can use these types of queries. And the results are shown according to the book purchase count in the table. Also, it displays in descending order.

**Code**

Select BookID, Count(BookID) as Purchase\_Count

from Customer\_Order

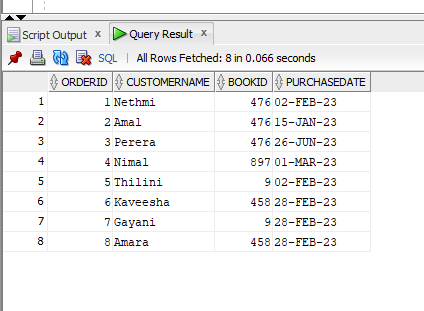
where BookID != 458

group by BookID

having Count(BookID)> 1

order by BookID DESC;

**Output**

A screenshot of a computer

Description automatically generated

Figure 8 - Question 3 output

Figure 9 - Customer\_Order table

1. **Write a single-row and multiple-row subquery using the above tables.**

**Single-row subquery**

This selects the book which is more expensive than the book with the id 476 and author id should be 2. The results which satisfied both conditions are as below.

**Code**

select bookid, title, price

from book

where price > (select price from book where bookid='476')

AND bookid = (select bookid from book where authorid='2');

A screenshot of a computer

Description automatically generated**Output**

Figure 10 - Book table

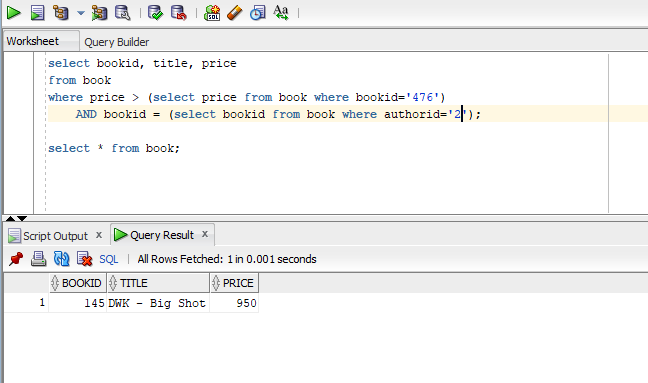


Figure 11 – Single-row subquery

**Multiple-row subquery**

Below query finds the books which are more expensive than the book reference to book id 476. And the author id should not equal to 5. ‘ANY’ keyword is used to represent the multiple-row subquery.

**Code**

select bookid, title, price

from book

where price > ANY (select price from book where bookid='476')

AND authorid <> 5;

A screenshot of a computer

Description automatically generated**Output**

Figure 12 - Book table

**A screenshot of a computer

Description automatically generated**

Figure 13 - Multiple-row subquery

1. **Write queries using left, right, and full outer joins. (There should be suitable records inserted to get the required outputs)**

Consider Book table as the right table and Author table as the left table.

**Left Join**

The LEFT JOIN keyword returns all records from the Author table, even if there are no matches in the Book table.

**Code**

select b.bookid, b.title, b.price, b.authorid, a.authorname

from Author a

left join Book b

on a.AuthorID=b.AuthorID;

**Output**

A screenshot of a computer

Description automatically generated

Figure 14 - Left join

**Right Join**

Right join returns all the records from the book table, and the matched records from the author table. The result is NULL from the left side when there is no match.

**Code**

select b.bookid, b.title, b.price, b.authorid, a.authorname

from Author a

right join Book b

on a.AuthorID=b.AuthorID;

**A screenshot of a computer

Description automatically generatedOutput**

Figure 15 - Right join

**Full Outer Join**

For this instance, we have considered the customer\_order table and the book table.

**Code**

select b.BookID, b.Title, b.Price, c.CustomerName, c.PurchaseDate, c.OrderID

from Customer\_Order c

full outer join Book b

on c.BookID=b.BookID;

**Output**

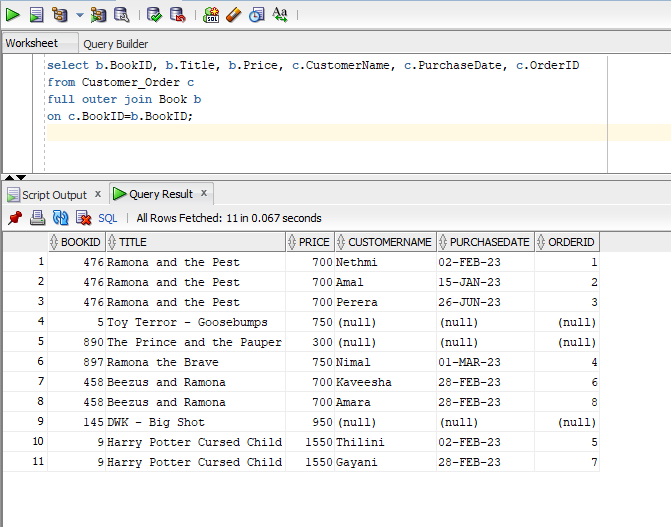


Figure 16 - Full Outer Join

1. **Create a view using one of the tables created.**

Book table is used to create the view and select query is used to see the columns and data.

**Code**

CREATE VIEW Book\_View AS

SELECT Title, Price

FROM Book;

SELECT \* from Book\_View;

**Output**

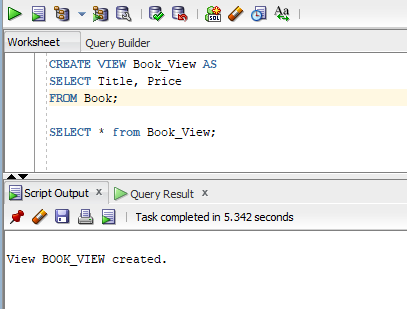


Figure 17 - Create view

A screenshot of a computer

Description automatically generated

Figure 18 - Book\_View table

1. **Write a PL/ SQL block to retrieve a record for specific input.**

Retrieve title and price of a specific book in book table.

**Code**

SET SERVEROUTPUT ON;

DECLARE

v\_book\_title Book.Title%TYPE;

v\_book\_price Book.Price%TYPE;

BEGIN

SELECT Title, Price

INTO v\_book\_title, v\_book\_price

FROM Book

WHERE BookID=9;

DBMS\_OUTPUT.PUT\_LINE ('Book title is :'|| v\_book\_title);

DBMS\_OUTPUT.PUT\_LINE ('Price is :'|| v\_book\_price);

END;

/

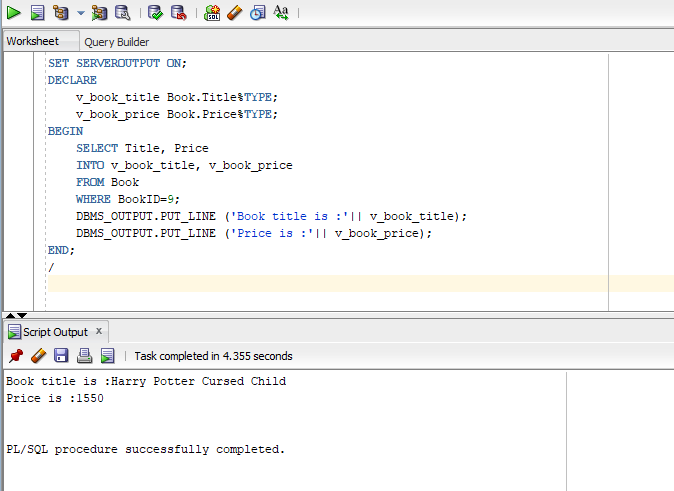
**Output**

Figure 19 - Retrieve record

1. **Write a PL/ SQL block to update a record for specific input.**

The price of a book has updated in the book table.

**Code**

SET SERVEROUTPUT ON;

DECLARE

price\_increase Book.price%TYPE:=100;

updated\_price Book.price%TYPE;

BEGIN

UPDATE Book

SET price=price+price\_increase

WHERE BookID=9

RETURNING price INTO updated\_price;

DBMS\_OUTPUT.PUT\_LINE ('Price updated');

DBMS\_OUTPUT.PUT\_LINE ('Price is :'|| updated\_price);

END;

/

**Output**

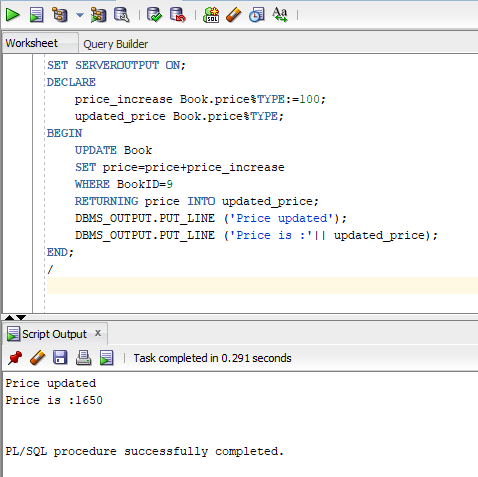


Figure 20 - Update record

1. **Write a PL/ SQL block to delete a record for specific input.**

A record is deleted from the customer\_order table.

**Code**

SET SERVEROUTPUT ON;

DECLARE

v\_order\_id Customer\_Order.OrderID%TYPE:=3;

BEGIN

DELETE FROM Customer\_Order

Where OrderID=v\_order\_id;

DBMS\_OUTPUT.PUT\_LINE ('Record Deleted');

END;

/

**Output**

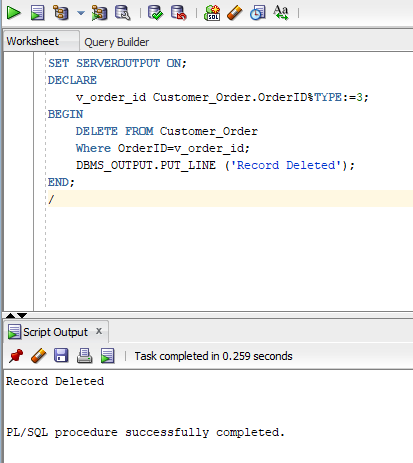
****

Figure 21 - Delete Record

1. **Modify the above query to display the number of rows deleted.**

Above code updated for this question also.

**Code**

SET SERVEROUTPUT ON;

DECLARE

v\_order\_id Customer\_Order.OrderID%TYPE:=4;

v\_rows\_deleted NUMBER;

BEGIN

DELETE FROM Customer\_Order

Where OrderID=v\_order\_id;

v\_rows\_deleted := SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE('Record Deleted. Rows Deleted: ' || v\_rows\_deleted);

END;

/

**Output**

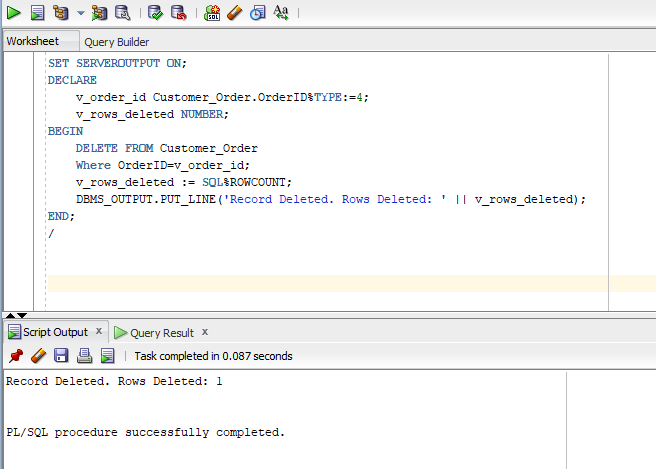
****

Figure 22 - Question number 10 output