GROUP PROJECT   
  
RELATIONAL DATABASE PROG 8590  
  
SECTION - 1

*TEAM MEMBERS:*

CHUNXI ZHANG (8971818)

LYDIA ASHLYN NAVEDHA (8998609)

SATBIR KAUR (8852075)

VETRI VELAN (8979936)

***Question 1:***

***Create a type record named order\_obj\_t that contains all of the attributes of the***

***OEHR\_ORDERS table plus customer’s first and last name, credit limit for this customer as well as***

***sales person’s last and first name (if any).***

***Write an anonymous block that creates a variable using the order\_obj\_t type for a specific order***

***ID. Include code for the type and the block.***

***Run the block with order ID 2458, 2355 and 1456. Include generated output. Sample output:***

***ORDER #:***

***2381***

***ORDER DATE: June 22, 2019***

***CUSTOMER: Matthias Hannah CREDIT LIMIT:***

***$100***

***ORDER STATUS: 3***

***ORDER TOTAL:$23,034.60***

***SALES PERSON: Janette King PROMOTION ID:***

**CODING:  
DECLARE**

**--Define record type**

**TYPE order\_obj\_t IS RECORD(**

**order\_id OEHR\_ORDERS.ORDER\_ID%TYPE,**

**order\_date OEHR\_ORDERS.ORDER\_DATE%TYPE,**

**order\_mode OEHR\_ORDERS.ORDER\_MODE%TYPE,**

**customer\_id OEHR\_ORDERS.CUSTOMER\_ID%TYPE,**

**customer\_fname OEHR\_CUSTOMERS.CUST\_FIRST\_NAME%TYPE,**

**customer\_lname OEHR\_CUSTOMERS.CUST\_LAST\_NAME%TYPE,**

**credit\_limit OEHR\_CUSTOMERS.CREDIT\_LIMIT%TYPE,**

**order\_status OEHR\_ORDERS.ORDER\_STATUS%TYPE,**

**order\_total OEHR\_ORDERS.ORDER\_TOTAL%TYPE,**

**sales\_rep\_id OEHR\_ORDERS.SALES\_REP\_ID%TYPE,**

**sales\_rep\_fname OEHR\_EMPLOYEES.FIRST\_NAME%TYPE,**

**sales\_rep\_lname OEHR\_EMPLOYEES.LAST\_NAME%TYPE,**

**promotion\_id OEHR\_ORDERS.PROMOTION\_ID%TYPE**

**);**

**-- Define the variable**

**order\_detail order\_obj\_t;**

**v\_order\_id NUMBER;**

**--Define the cursor**

**CURSOR order\_cursor IS**

**SELECT OEHR\_ORDERS.ORDER\_ID,**

**OEHR\_ORDERS.ORDER\_DATE,**

**OEHR\_ORDERS.ORDER\_MODE,**

**OEHR\_ORDERS.CUSTOMER\_ID,**

**OEHR\_CUSTOMERS.CUST\_FIRST\_NAME,**

**OEHR\_CUSTOMERS.CUST\_LAST\_NAME,**

**OEHR\_CUSTOMERS.CREDIT\_LIMIT,**

**OEHR\_ORDERS.ORDER\_STATUS,**

**OEHR\_ORDERS.ORDER\_TOTAL,**

**OEHR\_ORDERS.SALES\_REP\_ID,**

**OEHR\_EMPLOYEES.FIRST\_NAME,**

**OEHR\_EMPLOYEES.LAST\_NAME,**

**OEHR\_ORDERS.PROMOTION\_ID**

**FROM OEHR\_ORDERS**

**JOIN OEHR\_CUSTOMERS**

**ON OEHR\_ORDERS.CUSTOMER\_ID = OEHR\_CUSTOMERS.CUSTOMER\_ID**

**LEFT JOIN OEHR\_EMPLOYEES**

**ON OEHR\_ORDERS.SALES\_REP\_ID = OEHR\_EMPLOYEES.EMPLOYEE\_ID**

**WHERE OEHR\_ORDERS.ORDER\_ID IN (2458, 2355, 1456);**

**BEGIN**

**--Opening the cursor**

**OPEN order\_cursor;**

**LOOP**

**--Load data and fill the variable**

**FETCH order\_cursor INTO order\_detail;**

**IF order\_cursor%NOTFOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('ORDER #: '||order\_detail.order\_id||' not found.');**

**EXIT;**

**END IF;**

**--Display**

**DBMS\_OUTPUT.put\_line('ORDER #: '||order\_detail.order\_id);**

**DBMS\_OUTPUT.put\_line('ORDER DATE: '||order\_detail.order\_date);**

**DBMS\_OUTPUT.put\_line('CUSTOMER: '||order\_detail.customer\_fname||' '||order\_detail.customer\_lname);**

**DBMS\_OUTPUT.put\_line('CREDIT LIMIT: $'||order\_detail.credit\_limit);**

**DBMS\_OUTPUT.put\_line('ORDER STATUS: '||order\_detail.order\_status);**

**DBMS\_OUTPUT.put\_line('ORDER TOTAL: $'||order\_detail.order\_total);**

**DBMS\_OUTPUT.put\_line('SALES PERSON:: '||order\_detail.sales\_rep\_fname||' '||order\_detail.sales\_rep\_lname);**

**DBMS\_OUTPUT.put\_line('PROMOTION ID: '||order\_detail.promotion\_id);**

**DBMS\_OUTPUT.put\_line('-----------------------------------------');**

**END LOOP;**

**--Close the cursor**

**CLOSE order\_cursor;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.put\_line('Order not found.');**

**WHEN OTHERS THEN**

**DBMS\_OUTPUT.put\_line('An error occurred: ' || SQLERRM);**

**END;**

**TESTING:**

**A screenshot of a computer

Description automatically generated**

***Question 2: [15 pts]***

***1. Write a PL/SQL program to define a record type rt\_employee that include partial columns from the table oehr\_employees. Make sure to use the same datatype definitions and Include***

***only the following fields: JOB\_ID, SALARY, MANAGER\_ID and Department\_ID.***

***2. Define two records r\_employee1 and r\_employee2 as type rt\_employee.***

***3. Fill r\_employee1 by the data from the table oehr\_employees for the employee 101. Use a cursor for that.***

***4. Fill r\_employee2 by the data from the table oehr\_employees for the employee 102. Use a cursor for that.***

***5. Compare r\_employee1 to r\_employee2 and display a message for the user about the result: equivalent or distinct.***

***6. Create a procedure print\_Employee that display the JOB\_ID, SALARY, MANAGER\_ID and***

***Department\_ID of a variable of the record rt\_employee. Call the procedure for r\_employee1 and r\_employee2.***

**CODING:**

**-- Define the record and cursor**

**DECLARE**

**TYPE rt\_employee IS RECORD(**

**job\_id OEHR\_EMPLOYEES.JOB\_ID%TYPE,**

**salary OEHR\_EMPLOYEES.SALARY%TYPE,**

**manager\_id OEHR\_EMPLOYEES.MANAGER\_ID%TYPE,**

**department\_id OEHR\_EMPLOYEES.DEPARTMENT\_ID%TYPE**

**);**

**r\_employee1 rt\_employee;**

**r\_employee2 rt\_employee;**

**CURSOR employee101 IS**

**SELECT JOB\_ID,**

**SALARY,**

**MANAGER\_ID,**

**DEPARTMENT\_ID**

**FROM OEHR\_EMPLOYEES**

**WHERE EMPLOYEE\_ID = 101;**

**CURSOR employee102 IS**

**SELECT JOB\_ID,**

**SALARY,**

**MANAGER\_ID,**

**DEPARTMENT\_ID**

**FROM OEHR\_EMPLOYEES**

**WHERE EMPLOYEE\_ID = 102;**

**PROCEDURE print\_Employee (emp\_id IN NUMBER, emp\_rec IN rt\_employee)**

**IS**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE('Employee ' || emp\_id || ':');**

**DBMS\_OUTPUT.PUT\_LINE(' Job ID: ' || emp\_rec.job\_id);**

**DBMS\_OUTPUT.PUT\_LINE(' Salary: ' || emp\_rec.salary);**

**DBMS\_OUTPUT.PUT\_LINE(' Manager ID: ' || emp\_rec.manager\_id);**

**DBMS\_OUTPUT.PUT\_LINE(' Department ID: ' || emp\_rec.department\_id);**

**DBMS\_OUTPUT.PUT\_LINE('------------------------------');**

**END print\_Employee;**

**BEGIN**

**OPEN employee101;**

**FETCH employee101**

**INTO r\_employee1;**

**CLOSE employee101;**

**OPEN employee102;**

**FETCH employee102**

**INTO r\_employee2;**

**CLOSE employee102;**

**IF r\_employee1.job\_id = r\_employee2.job\_id**

**AND r\_employee1.salary = r\_employee2.salary**

**AND r\_employee1.manager\_id = r\_employee2.manager\_id**

**AND r\_employee1.department\_id = r\_employee2.department\_id**

**THEN**

**DBMS\_OUTPUT.put\_line('The two employees are equivalent.');**

**ELSE**

**DBMS\_OUTPUT.put\_line('The two employees are distinct.');**

**END IF;**

**print\_Employee(101, r\_employee1);**

**print\_Employee(102 ,r\_employee2);**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('No data found.');**

**WHEN OTHERS THEN**

**DBMS\_OUTPUT.PUT\_LINE('Something wrong: ' || SQLERRM);**

**END;**

**TESTING:**

**A screenshot of a computer

Description automatically generated**

***Q3: Write a PL/SQL function to determine the number of employees for a given department. The program will search using the department's ID and determine how many employees exists in it.***

***Your program should store the result into a variable.***

***Run the program hardcoding in a search for the department id 60.***

***\*\*Hint: Use OEHR\_EMPLOYEES table.***

**CODE:**

**CREATE OR REPLACE FUNCTION get\_employee\_count(p\_department\_id NUMBER)**

**RETURN NUMBER**

**IS**

**v\_employee\_count NUMBER;**

**BEGIN**

**-- Count the number of employees in the specified department**

**SELECT COUNT(\*)**

**INTO v\_employee\_count**

**FROM OEHR\_EMPLOYEES**

**WHERE DEPARTMENT\_ID = p\_department\_id;**

**-- Return the count**

**RETURN v\_employee\_count;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**-- If no employees are found, return 0**

**RETURN 0;**

**WHEN OTHERS THEN**

**-- Handle other unexpected errors**

**DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);**

**RETURN -1;**

**END;**

**OUTPUT:**

**A white rectangular object with a gray border

Description automatically generated**

**CODE:**

**DECLARE**

**v\_department\_id NUMBER := 60;**

**v\_result NUMBER;**

**BEGIN**

**-- Call the function with department ID 60**

**v\_result := get\_employee\_count(v\_department\_id);**

**-- Display the result**

**DBMS\_OUTPUT.PUT\_LINE('Number of employees in department ' || v\_department\_id || ': ' || v\_result);**

**END;**

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**

***4. Write a PL/SQL function to determine the number of employees for a given department. The program will search using the department's ID and determine how many employees exists in it. Your program should store the result into a variable. Run the program hardcoding in a search for the department id 60. \*\*Hint: Use OEHR\_EMPLOYEES table.***

**Solution:**

CREATE OR REPLACE FUNCTION get\_emp\_count(dept\_id IN NUMBER) RETURN NUMBER AS

emp\_count NUMBER; -- Variable to store the number of employees

BEGIN

-- Query to count the number of employees in the dptmt

SELECT COUNT(\*)

INTO emp\_count

FROM oehr\_employees

WHERE department\_id = dept\_id;

-- Returning the result

RETURN emp\_count;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- if no employees exist in the department

RETURN 0;

WHEN OTHERS THEN

-- any unexpected error

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

RETURN -1; -- Indicate an error with a return value

END;

A screenshot of a computer

Description automatically generated

--\*\*\*\*\*\*\*\*\*\*\*Anonymous block\*\*\*\*\*\*\*\*\*

DECLARE

dept\_id NUMBER := 60; -- Hardcoding department ID

num\_emps NUMBER;

BEGIN

num\_emps := get\_emp\_count(dept\_id);

-- Display res

DBMS\_OUTPUT.PUT\_LINE('Department ID: ' || dept\_id || ', Number of Employees: ' || num\_emps);

END;

A screen shot of a computer

Description automatically generated

***5. Modify the written program in above Question to determine the status of a department. If the department has 30 or more employees, display a message telling a ‘crowded department’. If the department has less than 30 employees, display ‘normal department. If the department has only one employee, display ‘New department. Run the program 3 times hardcoding in a search for department id 30, 40 and 50. \*\*Hint: You only need to submit one version of your code but include three outputs.***

**Solution**

CREATE OR REPLACE FUNCTION get\_dept\_status(dept\_id IN NUMBER) RETURN VARCHAR2 AS

emp\_count NUMBER; -- for number of employees

status VARCHAR2(50); -- to store status

BEGIN

-- to count employees

SELECT COUNT(\*)

INTO emp\_count

FROM oehr\_employees

WHERE department\_id = dept\_id;

-- checking the status of dpt

IF emp\_count >= 30 THEN

status := 'Crowded department';

ELSIF emp\_count = 1 THEN

status := 'New department';

ELSE

status := 'Normal department';

END IF;

-- Return the status

RETURN status;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 'No info of such department exists'; -- if the department does not exist

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

RETURN 'Error getting department status';

END;

A screenshot of a computer

Description automatically generated

--\*\*\*\*88\*\*\*\*\*Anonymous block\*\*\*\*\*\*\*

DECLARE

dept\_id NUMBER;

dept\_status VARCHAR2(50);

BEGIN

-- when Department ID is 30

dept\_id := 30;

dept\_status := get\_dept\_status(dept\_id);

DBMS\_OUTPUT.PUT\_LINE('Department ID: ' || dept\_id || ' - ' || dept\_status);

END;

3 outputs according to three dept ids:

A computer screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

***6. Write a program in PL/SQL to create a single explicit cursor. You are asked to display the following: PRODUCT\_ID, PRODUCT\_NAME and LIST\_PRICE from OEHR\_PRODUCT\_INFORMATION QUANTITY\_ON\_HAND from the table OEHR\_INVENTORIES WAREHOUSE\_NAME from the table OEHR\_WAREHOUSES Filter your result set to include only records for the WAREHOUSE\_ID 5.***

**Solution:**

DECLARE

prod\_id OEHR\_PRODUCT\_INFORMATION.product\_id%TYPE;

prod\_name OEHR\_PRODUCT\_INFORMATION.product\_name%TYPE;

list\_price OEHR\_PRODUCT\_INFORMATION.list\_price%TYPE;

quantity\_on\_hand OEHR\_INVENTORIES.quantity\_on\_hand%TYPE;

warehouse\_name OEHR\_WAREHOUSES.warehouse\_name%TYPE;

-- Explicit cursor

CURSOR prod\_cursor IS

SELECT

p.product\_id,

p.product\_name,

p.list\_price,

i.quantity\_on\_hand,

w.warehouse\_name

FROM

OEHR\_PRODUCT\_INFORMATION p

JOIN

OEHR\_INVENTORIES i ON p.product\_id = i.product\_id

JOIN

OEHR\_WAREHOUSES w ON i.warehouse\_id = w.warehouse\_id

WHERE

i.warehouse\_id = 5;

BEGIN

-- Open and fetch from the cursor

OPEN prod\_cursor;

LOOP

FETCH prod\_cursor

INTO prod\_id, prod\_name, list\_price, quantity\_on\_hand, warehouse\_name;

EXIT WHEN prod\_cursor%NOTFOUND;

-- Display

DBMS\_OUTPUT.PUT\_LINE('Product ID: ' || prod\_id || ', Product Name: ' || prod\_name || ', List Price: ' || list\_price || ', Quantity on Hand: ' || quantity\_on\_hand || ', Warehouse Name: ' || warehouse\_name);

END LOOP;

CLOSE prod\_cursor;

END;

A screen shot of a computer

Description automatically generated

# ***7: Table: OEHR\_ORDERS***

1. ***Write a PL/SQL function which finds the highest total order and returns this value as a result from the function***

***Code:***

CREATE OR REPLACE FUNCTION Get\_Highest\_Order\_Total

RETURN NUMBER IS

highest\_total NUMBER;

BEGIN

SELECT MAX(order\_total)

INTO highest\_total

FROM OEHR\_ORDERS;

RETURN highest\_total;

END;

***screenshot:***

A screenshot of a computer

Description automatically generated

1. ***Write an anonymous block which calls your function created in Question 7a and prints to the DBMS output (where # is the value returned from the function) - The highest order total is: #***

***Code:***

DECLARE

highest\_order\_total NUMBER;

BEGIN

highest\_order\_total := Get\_Highest\_Order\_Total;

DBMS\_OUTPUT.PUT\_LINE('The highest order total is: '|| highest\_order\_total);

END;

***screenshot:***

A screenshot of a computer

Description automatically generated

# *Question 8 :*

*Table: OEHR\_JOB\_HISTORY*

*a) Write a PL/SQL procedure named Remove\_History that remove a row from the table OEHR\_JOB\_HISTORY :*

* *The row should be identified by the customer\_id that is passed as an INPUT parameter.*
* *Include an exception handler in your procedure in case no customer is found.*
* *Include a message to the user in case of successful deletion.*

*Write an anonymous block which calls the procedure created in for the customer id 200.*

***Code:***

CREATE OR REPLACE PROCEDURE Remove\_History (customer\_id IN NUMBER) IS

BEGIN

DELETE FROM OEHR\_JOB\_HISTORY

WHERE customer\_id = customer\_id;

IF SQL%ROWCOUNT = 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'No customer found with the provided ID.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Row successfully deleted for customer ID: ' || customer\_id);

END IF;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

***screenshot:***

A screenshot of a computer

Description automatically generated

***Code:***

BEGIN

Remove\_History(200);

END;

***screenshot:***

A screenshot of a computer

Description automatically generated

***Question 9***

***Table:*** *OEHR\_ORDERS*

***a)****Define PL/SQL record named* ***order\_rec*** *that holds the following columns of the tableOEHR\_ORDERS: ORDER\_ID, ORDER\_DATE, ORDER\_MODE, CUSTOMER\_ID andPROMOTION\_ID.  
Use a cursor to fetch all data into a variable of type order\_rec and to display them.*

*Answer:*

DECLARE

-- Define the record structure with required columns from OEHR\_ORDERS table

TYPE order\_rec IS RECORD (

ORDER\_ID OEHR\_ORDERS.ORDER\_ID%TYPE,

ORDER\_DATE OEHR\_ORDERS.ORDER\_DATE%TYPE,

ORDER\_MODE OEHR\_ORDERS.ORDER\_MODE%TYPE,

CUSTOMER\_ID OEHR\_ORDERS.CUSTOMER\_ID%TYPE,

PROMOTION\_ID OEHR\_ORDERS.PROMOTION\_ID%TYPE

);

-- Declare a cursor to fetch data from OEHR\_ORDERS table

CURSOR order\_cursor IS

SELECT ORDER\_ID, ORDER\_DATE, ORDER\_MODE, CUSTOMER\_ID, PROMOTION\_ID

FROM OEHR\_ORDERS;

-- Declare a variable of type order\_rec to hold the fetched row data

order\_data order\_rec;

BEGIN

-- Open the cursor and fetch each row into the record

OPEN order\_cursor;

LOOP

FETCH order\_cursor INTO order\_data;

EXIT WHEN order\_cursor%NOTFOUND;

-- Display the fetched data

DBMS\_OUTPUT.PUT\_LINE('Order ID: ' || order\_data.ORDER\_ID);

DBMS\_OUTPUT.PUT\_LINE('Order Date: ' || order\_data.ORDER\_DATE);

DBMS\_OUTPUT.PUT\_LINE('Order Mode: ' || order\_data.ORDER\_MODE);

DBMS\_OUTPUT.PUT\_LINE('Customer ID: ' || order\_data.CUSTOMER\_ID);

DBMS\_OUTPUT.PUT\_LINE('Promotion ID: ' || order\_data.PROMOTION\_ID);

DBMS\_OUTPUT.PUT\_LINE('-------------------------');

END LOOP;

-- Close the cursor

CLOSE order\_cursor;

END;

A screenshot of a computer

Description automatically generated

***b)*** *Save this output into your output text file.*

No privilege to save a file in apex oracle. Checked all possible ways but could not do it.

***c)*** *Modify your code in Question 9a to* ***include all*** *the following conditions:*

*-Only orders with no promotions are considered,*

*-Order mode: online,*

*-Order total less than 1000,*

*-Sort your result the most recent order first.*

Code:

DECLARE

-- Define the record structure for order data

TYPE order\_rec IS RECORD (

ORDER\_ID OEHR\_ORDERS.ORDER\_ID%TYPE,

ORDER\_DATE OEHR\_ORDERS.ORDER\_DATE%TYPE,

ORDER\_MODE OEHR\_ORDERS.ORDER\_MODE%TYPE,

CUSTOMER\_ID OEHR\_ORDERS.CUSTOMER\_ID%TYPE,

PROMOTION\_ID OEHR\_ORDERS.PROMOTION\_ID%TYPE

);

-- Define the cursor to fetch data with the specified conditions

CURSOR order\_cursor IS

SELECT ORDER\_ID, ORDER\_DATE, ORDER\_MODE, CUSTOMER\_ID, PROMOTION\_ID

FROM OEHR\_ORDERS

WHERE PROMOTION\_ID IS NULL

AND ORDER\_MODE = 'online'

AND ORDER\_TOTAL < 1000

ORDER BY ORDER\_DATE DESC;

-- Declare a variable of type order\_rec to hold the fetched data

order\_data order\_rec;

BEGIN

-- Open the cursor and fetch rows based on conditions

OPEN order\_cursor;

LOOP

FETCH order\_cursor INTO order\_data;

EXIT WHEN order\_cursor%NOTFOUND;

-- Display the fetched values

DBMS\_OUTPUT.PUT\_LINE('Order ID: ' || order\_data.ORDER\_ID);

DBMS\_OUTPUT.PUT\_LINE('Order Date: ' || order\_data.ORDER\_DATE);

DBMS\_OUTPUT.PUT\_LINE('Order Mode: ' || order\_data.ORDER\_MODE);

DBMS\_OUTPUT.PUT\_LINE('Customer ID: ' || order\_data.CUSTOMER\_ID);

DBMS\_OUTPUT.PUT\_LINE('Promotion ID: ' || order\_data.PROMOTION\_ID);

DBMS\_OUTPUT.PUT\_LINE('-------------------------');

END LOOP;

-- Close the cursor after fetching all rows

CLOSE order\_cursor;

END;

Output:

A screenshot of a computer

Description automatically generated

**d)**Save this output into your output text file.

No privilege to save a file in apex oracle. Checked all possible ways but could not do it.

***Question 10 :***

***Table:*** *OEHR\_ORDERS*

***a)****Define PL/SQL trigger that fire for every row before an insert on the table OEHR\_ORDERS asfollows:a.Your trigger should be executed* ***only for online orders*** *and,*

b.Your trigger should display the new inserted values for the column **Order\_Date**.

Code:

CREATE OR REPLACE TRIGGER trg\_online\_order\_insert

BEFORE INSERT ON OEHR\_ORDERS

FOR EACH ROW

BEGIN

-- Check if the order is online

IF :NEW.ORDER\_MODE = 'online' THEN

-- Display the new inserted values for the ORDER\_DATE

DBMS\_OUTPUT.PUT\_LINE('New Order Date: ' || :NEW.ORDER\_DATE);

END IF;

END;

Output:

A screenshot of a computer

Description automatically generated

**b)** Create an insert statement to test the trigger with the following values:

ORDER\_ID:1

ORDER\_DATE: The system date

ORDER\_MODE: online

Customer\_id:101

Code:

INSERT INTO OEHR\_ORDERS (ORDER\_ID, ORDER\_DATE, ORDER\_MODE, CUSTOMER\_ID)

VALUES (1, SYSDATE, 'online', 101);

Output:

A screenshot of a computer

Description automatically generated

1. Save this output into your output text file.

No privilege to save a file in apex oracle. Checked all possible ways but could not do it.

***Question 11 : [10 pts]***

*Write an anonymous block using PL/SQL that contains Varray with the element type of a customer created record. The record must contain the first name, last name and summary of orders for each customer.*

*Sample output:*

|  |  |
| --- | --- |
| Array has 99 elements First Name: Eugene  Last Name: Taylashev | |
| Orders total: | $48.00 |
| First Name: Justin Last Name: Trudeau Orders total: | $220.00 |

Code:   
  
DECLARE

-- Define a record for customer details

TYPE customer\_rec IS RECORD (

first\_name VARCHAR2(50),

last\_name VARCHAR2(50),

order\_total NUMBER

);

-- Define a VARRAY to hold the customer records

TYPE customer\_array IS VARRAY(100) OF customer\_rec;

-- Declare a variable of the VARRAY type

v\_customers customer\_array;

BEGIN

-- Initialize the VARRAY with a maximum of 100 customers

v\_customers := customer\_array();

-- Loop through each customer and populate the array

FOR customer\_data IN (

SELECT c.CUST\_FIRST\_NAME, c.CUST\_LAST\_NAME, COALESCE(SUM(o.ORDER\_TOTAL), 0) AS order\_total

FROM OEHR\_ORDERS o

JOIN OEHR\_CUSTOMERS c ON o.CUSTOMER\_ID = c.CUSTOMER\_ID

WHERE o.ORDER\_MODE = 'online'

GROUP BY c.CUST\_FIRST\_NAME, c.CUST\_LAST\_NAME

ORDER BY c.CUST\_FIRST\_NAME, c.CUST\_LAST\_NAME

) LOOP

-- Add each customer to the array

v\_customers.EXTEND;

v\_customers(v\_customers.LAST) := customer\_rec(customer\_data.CUST\_FIRST\_NAME, customer\_data.CUST\_LAST\_NAME, customer\_data.order\_total);

END LOOP;

-- Output the total number of customers

DBMS\_OUTPUT.PUT\_LINE('Array has ' || v\_customers.COUNT || ' elements');

-- Output the details for each customer in the array

FOR i IN 1 .. v\_customers.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE('First Name: ' || v\_customers(i).first\_name);

DBMS\_OUTPUT.PUT\_LINE('Last Name: ' || v\_customers(i).last\_name);

-- Handle NULL or zero values and format the number output properly

IF v\_customers(i).order\_total IS NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Orders total: $NULL');

ELSIF v\_customers(i).order\_total = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Orders total: $0.00');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Orders total: $' || TO\_CHAR(v\_customers(i).order\_total, 'FM999,999,999.00'));

END IF;

DBMS\_OUTPUT.PUT\_LINE('----------------------------');

END LOOP;

END;

Output:

A white background with black text

Description automatically generated

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