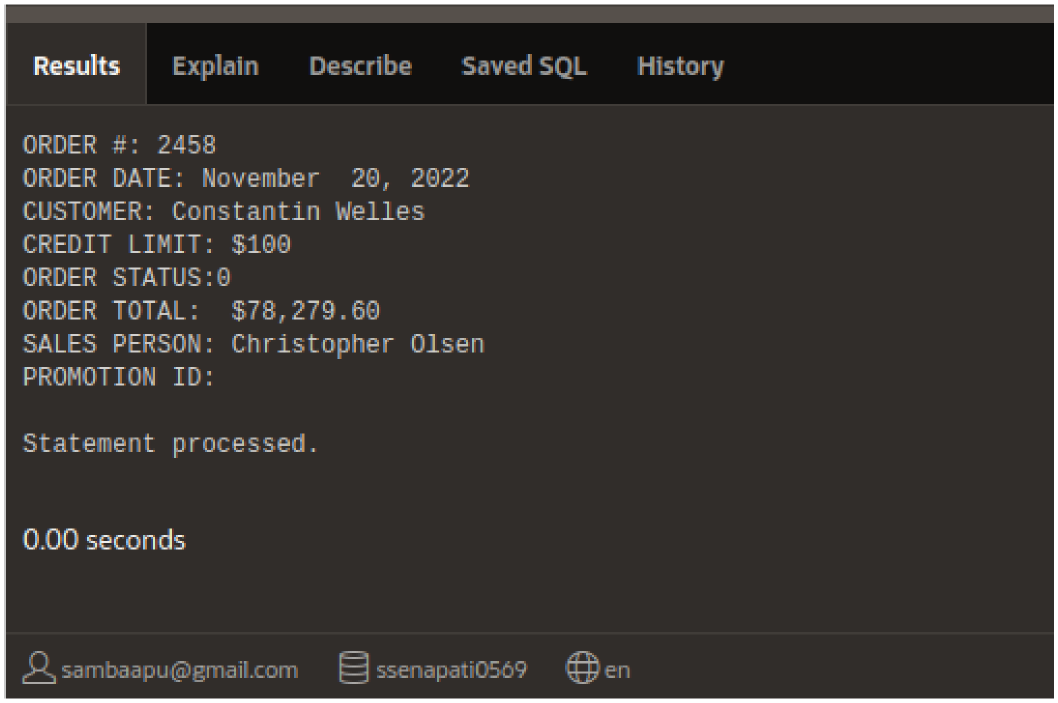
Question 1: Create an object type record named order_obj_t that contains all of the attributes of the OEHR_ORDERS table plus customer's first and last name, credit limit for this customer as well as sales person's last and first name (if any). Write an anonymous block that creates an object using the order_obj_t type for a specific order ID. Include code for the object type and the block. Run the block with order ID 2458, 2355 and 1456. Include generated output.

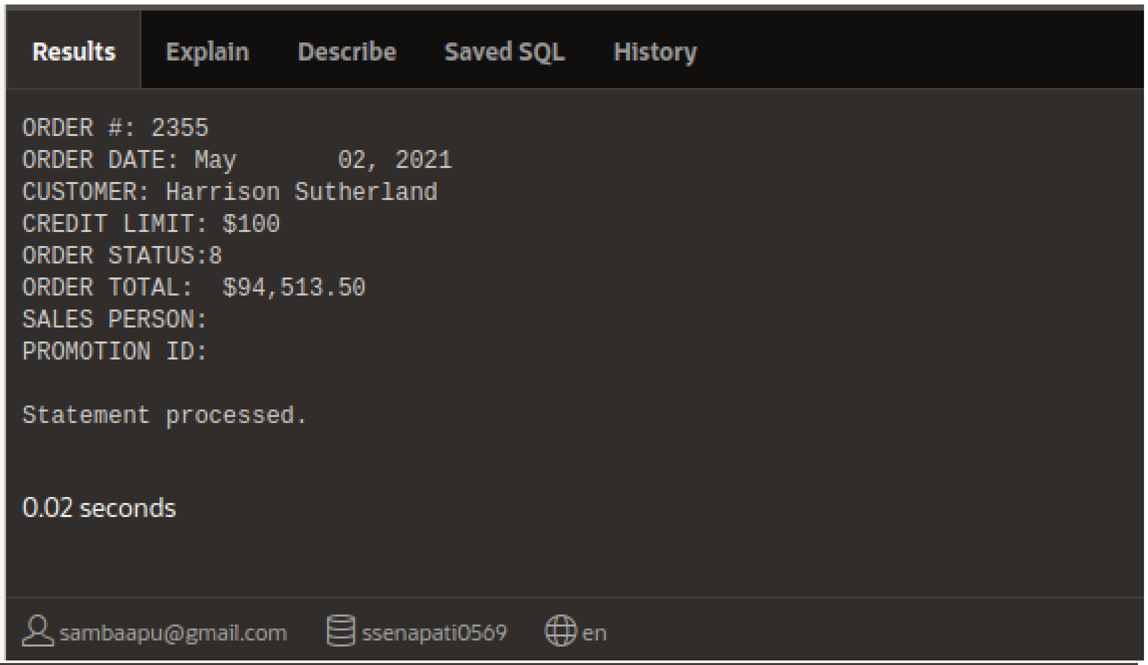
```
DECLARE
-- Define Record
TYPE order_obj_t IS RECORD (
    order_id                OEHR_ORDERS.ORDER_ID%TYPE,
    order_date              OEHR_ORDERS.ORDER_DATE%TYPE,
    order_mode              OEHR_ORDERS.ORDER_MODE%TYPE,
    customer_id             OEHR_ORDERS.CUSTOMER_ID%TYPE,
    order_status            OEHR_ORDERS.ORDER_STATUS%TYPE,
    order_total             OEHR_ORDERS.ORDER_TOTAL%TYPE,
    sales_rep_id            OEHR_ORDERS.SALES_REP_ID%TYPE,
    promotion_id            OEHR_ORDERS.PROMOTION_ID%TYPE,
    customer_first_name     OEHR_CUSTOMERS.CUST_FIRST_NAME%TYPE,
    customer_last_name      OEHR_CUSTOMERS.CUST_FIRST_NAME%TYPE,
    credit_limit            OEHR_CUSTOMERS.CREDIT_LIMIT%TYPE,
    salesperson_first_name  OEHR_EMPLOYEES.FIRST_NAME%TYPE,
    salesperson_last_name   OEHR_EMPLOYEES.LAST_NAME%TYPE
);

v_order_obj order_obj_t;
o_id NUMBER := 2355;

BEGIN
    SELECT
        o.ORDER_ID,
        o.ORDER_DATE,
        o.ORDER_MODE,
        o.CUSTOMER_ID,
        o.order_status,
        o.order_total,
        o.sales_rep_id,
        o.promotion_id,
        c.CUST_FIRST_NAME AS customer_first_name,
        c.CUST_LAST_NAME AS customer_last_name,
        c.credit_limit,
        s.first_name AS salesperson_first_name,
        s.last_name AS salesperson_last_name
    INTO v_order_obj
    FROM oehr_orders o
    LEFT JOIN oehr_customers c ON o.customer_id = c.customer_id
    LEFT JOIN oehr_employees s ON o.sales_rep_id = s.EMPLOYEE_ID
    WHERE o.order_id = o_id;

    -- Displaying Output
    DBMS_OUTPUT.PUT_LINE('ORDER #: ' || v_order_obj.order_id);
    DBMS_OUTPUT.PUT_LINE('ORDER DATE: ' || TO_CHAR(v_order_obj.order_date, 
'Month DD, YYYY'));
    DBMS_OUTPUT.PUT_LINE('CUSTOMER: ' || v_order_obj.customer_first_name || '
' || v_order_obj.customer_last_name );
    DBMS_OUTPUT.PUT_LINE('CREDIT LIMIT: $' || v_order_obj.credit_limit);
    DBMS_OUTPUT.PUT_LINE('ORDER STATUS:' || v_order_obj.order_status);
    DBMS_OUTPUT.PUT_LINE('ORDER TOTAL:' || TO_CHAR(v_order_obj.order_total, 
'$999,999.99'));
    -- DBMS_OUTPUT.PUT_LINE('ORDER TOTAL:$' || v_order_obj.order_total);
    DBMS_OUTPUT.PUT_LINE('SALES PERSON: ' || 
v_order_obj.salesperson_first_name || ' ' || 
v_order_obj.salesperson_last_name);
    DBMS_OUTPUT.PUT_LINE('PROMOTION ID: ' || v_order_obj.promotion_id);
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No order found with order_id: ' || o_id);
END;
```

Output:



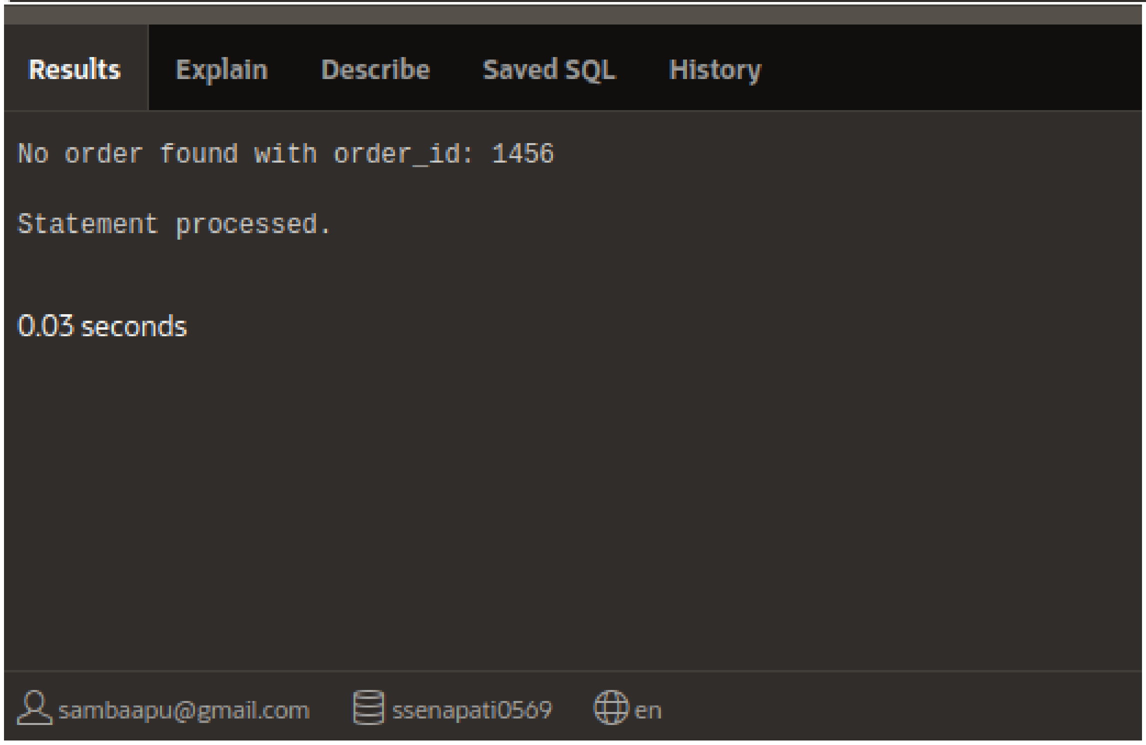
```
ORDER #: 2458
ORDER DATE: November  20, 2022
CUSTOMER: Constantin Welles
CREDIT LIMIT: $100
ORDER STATUS:0
ORDER TOTAL:  $78,279.60
SALES PERSON: Christopher Olsen
PROMOTION ID:

Statement processed.
```

0.00 seconds



```
ORDER #: 2355
ORDER DATE: May       02, 2021
CUSTOMER: Harrison Sutherland
CREDIT LIMIT: $100
ORDER STATUS:8
ORDER TOTAL:  $94,513.50
SALES PERSON:
PROMOTION ID:

Statement processed.
```

0.02 seconds



```
No order found with order_id: 1456

Statement processed.
```

0.03 seconds

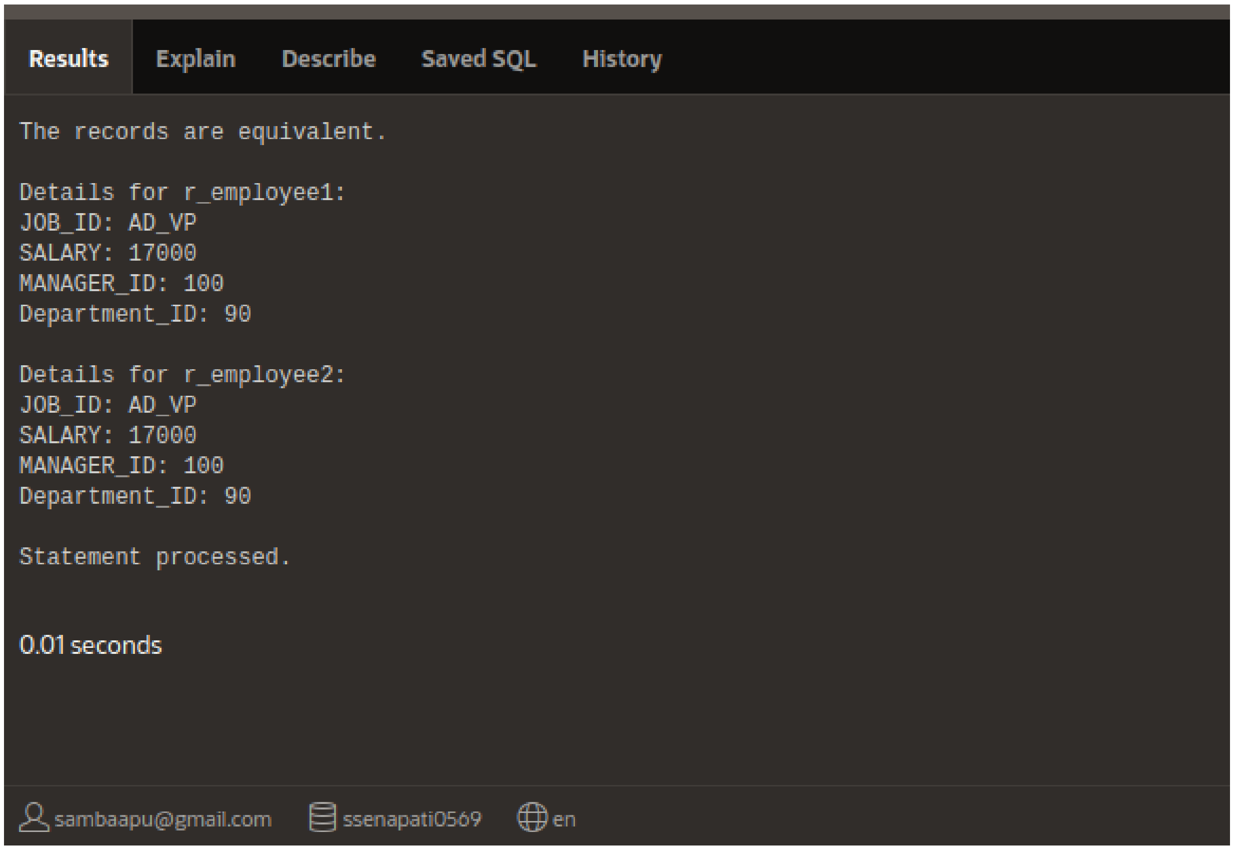
Question 2:

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.**

```
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
    );

-- Define two records r_employee1 and r_employee2 as type rt_employee
    r_employee1 rt_employee;
    r_employee2 rt_employee;

-- Define cursors to put data into records
    CURSOR c_employee1 IS
    SELECT job_id, salary, manager_id, department_id
    FROM oehr_employees
    WHERE employee_id = 101;

    CURSOR c_employee2 IS
    SELECT job_id, salary, manager_id, department_id
    FROM oehr_employees
    WHERE employee_id = 102;

-- Define Function to compare two rt_employee record
    FUNCTION compare_employee (rec1 IN rt_employee, rec2 IN rt_employee)
RETURN BOOLEAN
    IS
    BEGIN
        RETURN (rec1.JOB_ID = rec2.JOB_ID OR (rec1.JOB_ID IS NULL AND
rec2.JOB_ID IS NULL))
```

```
        AND (rec1.SALARY = rec2.SALARY OR (rec1.SALARY IS NULL AND rec2.SALARY
IS NULL))
        AND (rec1.MANAGER_ID = rec2.MANAGER_ID OR (rec1.MANAGER_ID IS NULL AND
rec2.MANAGER_ID IS NULL))
        AND (rec1.Department_ID = rec2.Department_ID OR (rec1.Department_ID IS
NULL AND rec2.Department_ID IS NULL));
    END;
-- Define Procedure to print record
    PROCEDURE print_Employee(emp rt_employee)
    IS
    BEGIN
        DBMS_OUTPUT.PUT_LINE('JOB_ID: ' || emp.JOB_ID);
        DBMS_OUTPUT.PUT_LINE('SALARY: ' || emp.SALARY);
        DBMS_OUTPUT.PUT_LINE('MANAGER_ID: ' || emp.MANAGER_ID);
        DBMS_OUTPUT.PUT_LINE('Department_ID: ' || emp.Department_ID);
    END;

BEGIN
    OPEN c_employee1;
    FETCH c_employee1 INTO r_employee1;
    CLOSE c_employee1;

    OPEN c_employee2;
    FETCH c_employee2 INTO r_employee2;
    CLOSE c_employee2;

    IF compare_employee(r_employee1, r_employee2) THEN
        DBMS_OUTPUT.PUT_LINE('The records are equivalent.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('The records are distinct.');
    END IF;

    DBMS_OUTPUT.PUT_LINE('');

    DBMS_OUTPUT.PUT_LINE('Details for r_employee1:');
    print_Employee(r_employee1);

    DBMS_OUTPUT.PUT_LINE('');

    DBMS_OUTPUT.PUT_LINE('Details for r_employee2:');
    print_Employee(r_employee2);
END;
```

Output:



```
The records are equivalent.

Details for r_employee1:
JOB_ID: AD_VP
SALARY: 17000
MANAGER_ID: 100
Department_ID: 90

Details for r_employee2:
JOB_ID: AD_VP
SALARY: 17000
MANAGER_ID: 100
Department_ID: 90

Statement processed.
```

0.01 seconds

sambaapu@gmail.com ssenapati0569 en

Question 3: 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.

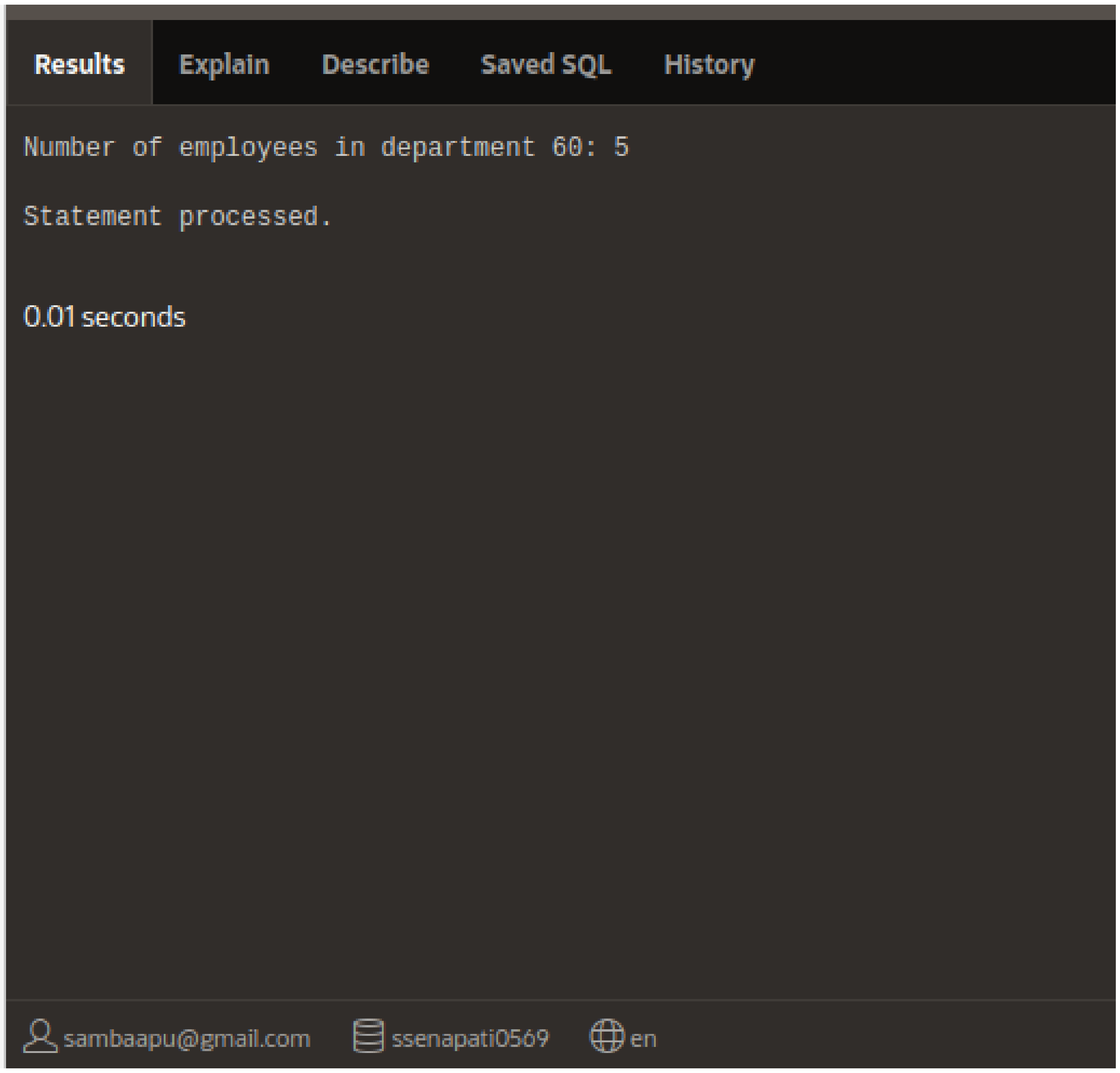
Function:

```
CREATE OR REPLACE FUNCTION get_employee_count(dept_id NUMBER) RETURN NUMBER
IS
    emp_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO emp_count
    FROM OEHR_EMPLOYEES
    WHERE DEPARTMENT_ID = dept_id;

    RETURN emp_count;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 0;

END;
```

Anonymous Block:

```
DECLARE
  res NUMBER;
  did NUMBER := 60;
BEGIN
  res := get_employee_count(did);
  DBMS_OUTPUT.PUT_LINE('Number of employees in department '|| did || ': ' ||
res);

END;
```

Output:



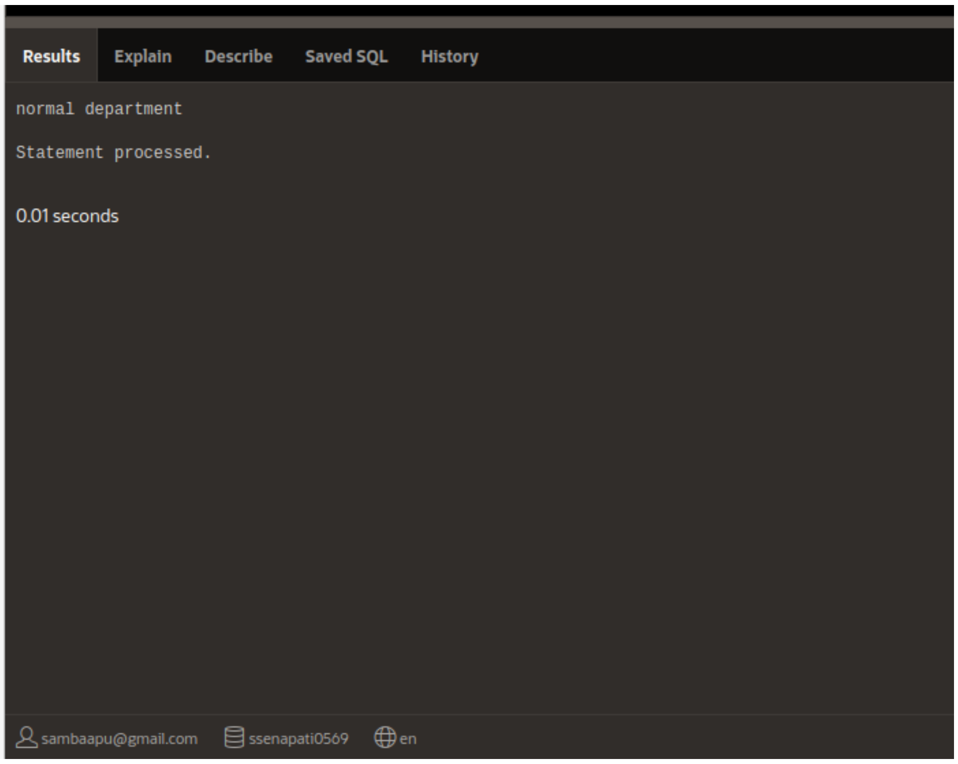
```
Number of employees in department 60: 5

Statement processed.
```

0.01 seconds

Question 4: Modify the written program in Question3 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.

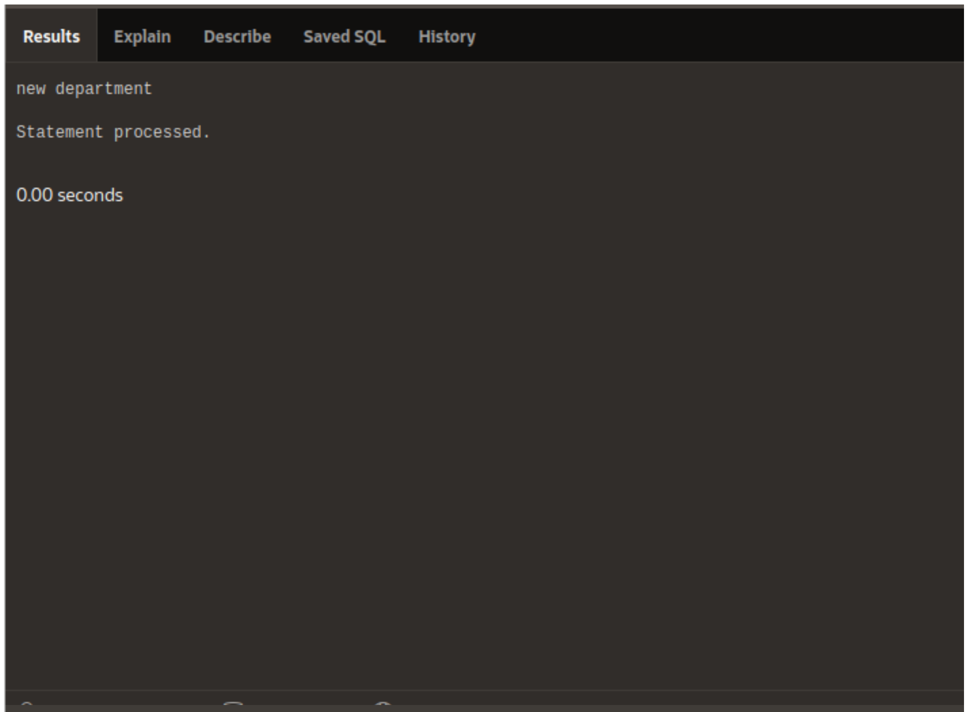
```
DECLARE
  res NUMBER;
  did NUMBER := 30;
BEGIN
  res := get_employee_count(did);
  IF res = 1 THEN
    DBMS_OUTPUT.PUT_LINE('new department');
  ELSIF res < 30 THEN
    DBMS_OUTPUT.PUT_LINE('normal department');
  ELSIF res >= 30 THEN
    DBMS_OUTPUT.PUT_LINE('crowded department');
  END IF;

END;
```

Output:



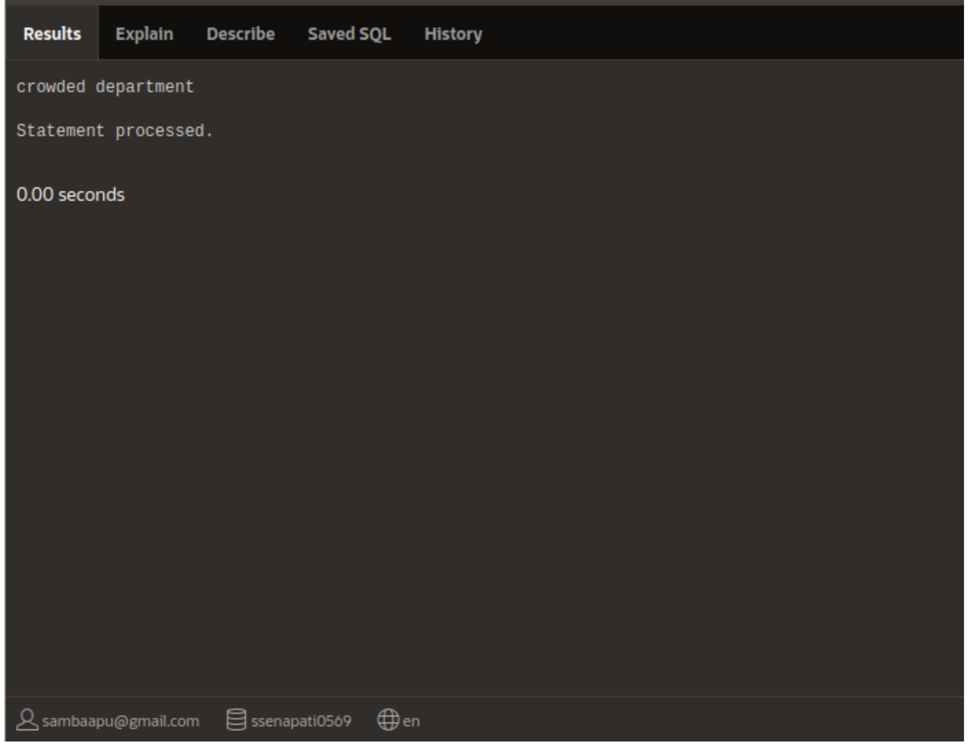
```
normal department

Statement processed.
```

0.01 seconds



```
new department

Statement processed.
```

0.00 seconds



```
crowded department

Statement processed.
```

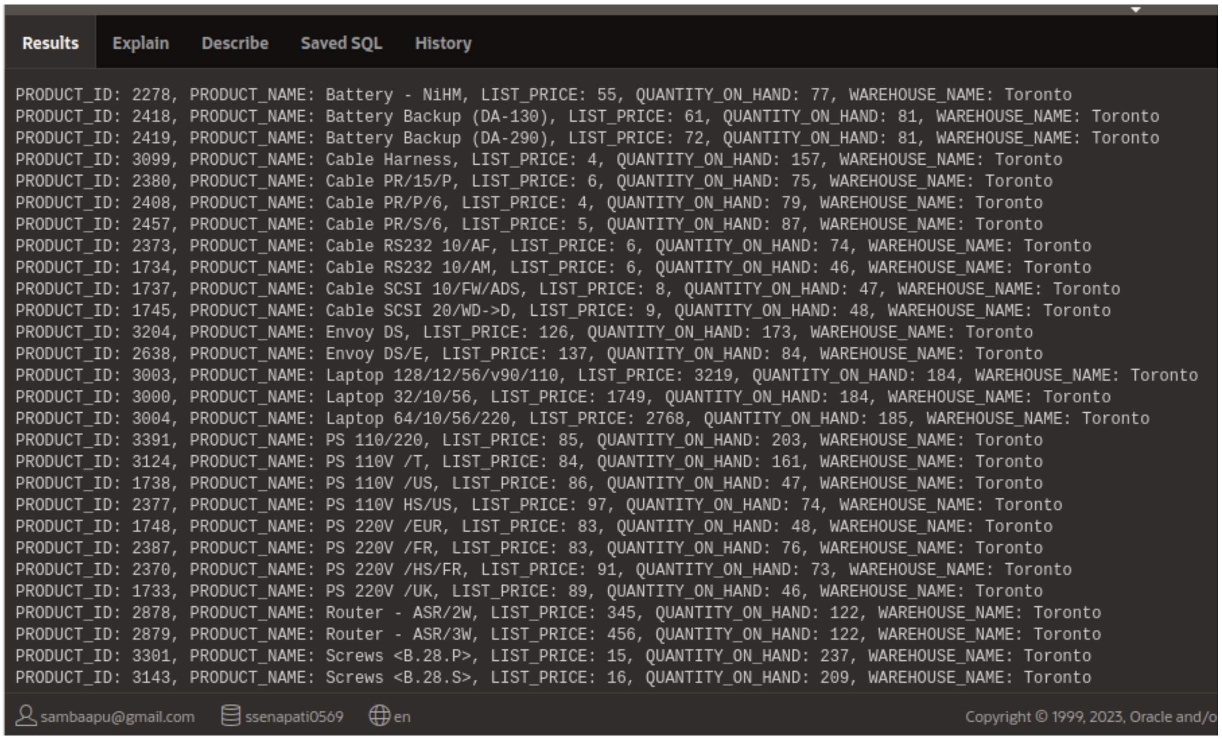
0.00 seconds

Question 5: 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.

```
DECLARE
    product_id        OEHR_PRODUCT_INFORMATION.PRODUCT_ID%TYPE;
    product_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;

    wid NUMBER := 5;

    CURSOR product_cursor IS
        SELECT
            pi.PRODUCT_ID,
            pi.PRODUCT_NAME,
            pi.LIST_PRICE,
            inv.QUANTITY_ON_HAND,
            wh.WAREHOUSE_NAME
        FROM
            OEHR_PRODUCT_INFORMATION pi
            JOIN OEHR_INVENTORIES inv ON pi.PRODUCT_ID = inv.PRODUCT_ID
            JOIN OEHR_WAREHOUSES wh ON inv.WAREHOUSE_ID = wh.WAREHOUSE_ID
        WHERE inv.WAREHOUSE_ID = wid;
BEGIN
    OPEN product_cursor;
    LOOP
        FETCH product_cursor INTO
            product_id,
            product_name,
            list_price,
            quantity_on_hand,
            warehouse_name;
        EXIT WHEN product_cursor%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE(
            'PRODUCT_ID: ' || product_id ||
            ', PRODUCT_NAME: ' || product_name ||
            ', LIST_PRICE: ' || list_price ||
            ', QUANTITY_ON_HAND: ' || quantity_on_hand ||
            ', WAREHOUSE_NAME: ' || warehouse_name
        );
    END LOOP;
    CLOSE product_cursor;
```

```
END;
```

<u>Output</u>

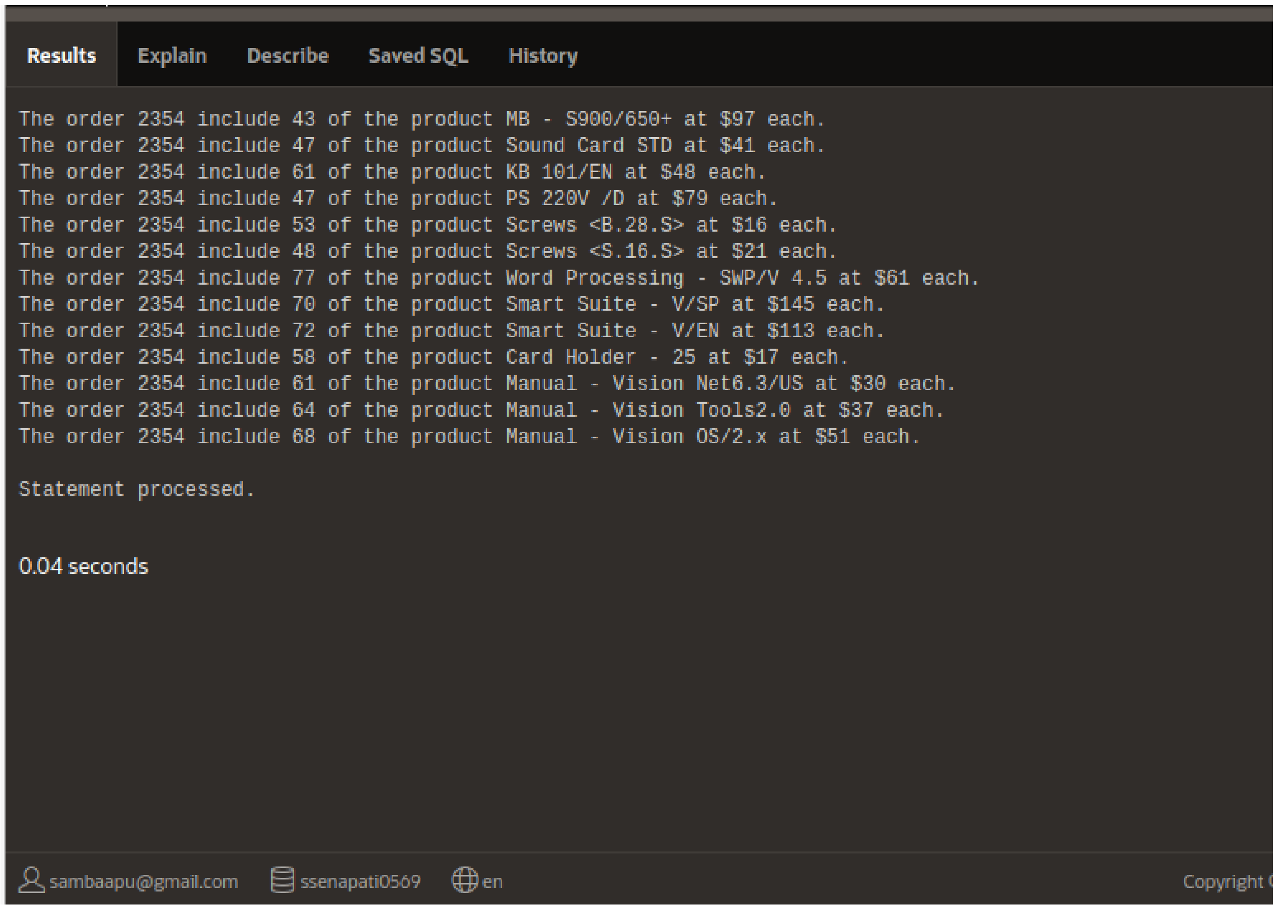


```
PRODUCT_ID: 2278, PRODUCT_NAME: Battery - NiHM, LIST_PRICE: 55, QUANTITY_ON_HAND: 77, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 2418, PRODUCT_NAME: Battery Backup (DA-130), LIST_PRICE: 61, QUANTITY_ON_HAND: 81, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 2419, PRODUCT_NAME: Battery Backup (DA-290), LIST_PRICE: 72, QUANTITY_ON_HAND: 81, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 3099, PRODUCT_NAME: Cable Harness, LIST_PRICE: 4, QUANTITY_ON_HAND: 157, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 2380, PRODUCT_NAME: Cable PR/15/P, LIST_PRICE: 6, QUANTITY_ON_HAND: 75, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 2408, PRODUCT_NAME: Cable PR/P/6, LIST_PRICE: 4, QUANTITY_ON_HAND: 79, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 2457, PRODUCT_NAME: Cable PR/S/6, LIST_PRICE: 5, QUANTITY_ON_HAND: 87, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 2373, PRODUCT_NAME: Cable RS232 10/AF, LIST_PRICE: 6, QUANTITY_ON_HAND: 74, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 1734, PRODUCT_NAME: Cable RS232 10/AM, LIST_PRICE: 6, QUANTITY_ON_HAND: 46, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 1737, PRODUCT_NAME: Cable SCSI 10/FW/ADS, LIST_PRICE: 8, QUANTITY_ON_HAND: 47, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 1745, PRODUCT_NAME: Cable SCSI 20/WD->D, LIST_PRICE: 9, QUANTITY_ON_HAND: 48, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 3204, PRODUCT_NAME: Envoy DS, LIST_PRICE: 126, QUANTITY_ON_HAND: 173, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 2638, PRODUCT_NAME: Envoy DS/E, LIST_PRICE: 137, QUANTITY_ON_HAND: 84, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 3003, PRODUCT_NAME: Laptop 128/12/56/v90/110, LIST_PRICE: 3219, QUANTITY_ON_HAND: 184, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 3000, PRODUCT_NAME: Laptop 32/10/56, LIST_PRICE: 1749, QUANTITY_ON_HAND: 184, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 3004, PRODUCT_NAME: Laptop 64/10/56/220, LIST_PRICE: 2768, QUANTITY_ON_HAND: 185, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 3391, PRODUCT_NAME: PS 110/220, LIST_PRICE: 85, QUANTITY_ON_HAND: 203, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 3124, PRODUCT_NAME: PS 110V /T, LIST_PRICE: 84, QUANTITY_ON_HAND: 161, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 1738, PRODUCT_NAME: PS 110V /US, LIST_PRICE: 86, QUANTITY_ON_HAND: 47, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 2377, PRODUCT_NAME: PS 110V HS/US, LIST_PRICE: 97, QUANTITY_ON_HAND: 74, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 1748, PRODUCT_NAME: PS 220V /EUR, LIST_PRICE: 83, QUANTITY_ON_HAND: 48, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 2387, PRODUCT_NAME: PS 220V /FR, LIST_PRICE: 83, QUANTITY_ON_HAND: 76, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 2370, PRODUCT_NAME: PS 220V /HS/FR, LIST_PRICE: 91, QUANTITY_ON_HAND: 73, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 1733, PRODUCT_NAME: PS 220V /UK, LIST_PRICE: 89, QUANTITY_ON_HAND: 46, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 2878, PRODUCT_NAME: Router - ASR/2W, LIST_PRICE: 345, QUANTITY_ON_HAND: 122, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 2879, PRODUCT_NAME: Router - ASR/3W, LIST_PRICE: 456, QUANTITY_ON_HAND: 122, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 3301, PRODUCT_NAME: Screws <B.28.P>, LIST_PRICE: 15, QUANTITY_ON_HAND: 237, WAREHOUSE_NAME: Toronto
PRODUCT_ID: 3143, PRODUCT_NAME: Screws <B.28.S>, LIST_PRICE: 16, QUANTITY_ON_HAND: 209, WAREHOUSE_NAME: Toronto
```

Question 6: Write an anonymous block which uses an order id value and displays DBMS output in the following format : be (where #1 is the quantity and #2 is the product name and #3 is the unit price.) The order 2354 include #1 of the product#2 at $#3 each. Create a variable to hold the order id value in the declare section of your anonymous block, and hard code in the value for the variable: 2354

```
DECLARE
    my_order_id OEHR_ORDER_ITEMS.ORDER_ID%TYPE := 2354;
    my_unit_price OEHR_ORDER_ITEMS.ORDER_ID%TYPE;
    qty OEHR_ORDER_ITEMS.ORDER_ID%TYPE;
    prod_name OEHR_PRODUCT_INFORMATION.PRODUCT_NAME%TYPE;

    CURSOR o_cursor IS
        SELECT UNIT_PRICE, QUANTITY, PRODUCT_NAME
        FROM OEHR_ORDER_ITEMS oi
        JOIN OEHR_PRODUCT_INFORMATION pi
        ON oi.PRODUCT_ID = pi.PRODUCT_ID
        WHERE ORDER_ID = my_order_id;
BEGIN
    OPEN o_cursor;
    LOOP
        FETCH o_cursor
            INTO my_unit_price, qty, prod_name;
        EXIT WHEN o_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('The order ' || my_order_id || ' include ' || qty
|| ' of the product ' || prod_name ||' at $' || my_unit_price || ' each.');
    END LOOP;

END;
```

Output



```
The order 2354 include 43 of the product MB - S900/650+ at $97 each.
The order 2354 include 47 of the product Sound Card STD at $41 each.
The order 2354 include 61 of the product KB 101/EN at $48 each.
The order 2354 include 47 of the product PS 220V /D at $79 each.
The order 2354 include 53 of the product Screws <B.28.S> at $16 each.
The order 2354 include 48 of the product Screws <S.16.S> at $21 each.
The order 2354 include 77 of the product Word Processing - SWP/V 4.5 at $61 each.
The order 2354 include 70 of the product Smart Suite - V/SP at $145 each.
The order 2354 include 72 of the product Smart Suite - V/EN at $113 each.
The order 2354 include 58 of the product Card Holder - 25 at $17 each.
The order 2354 include 61 of the product Manual - Vision Net6.3/US at $30 each.
The order 2354 include 64 of the product Manual - Vision Tools2.0 at $37 each.
The order 2354 include 68 of the product Manual - Vision OS/2.x at $51 each.

Statement processed.
```

0.04 seconds

Question 7:

A) Write a PL/SQL function which finds the highest total order and returns this value as a result from the function

b) 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)

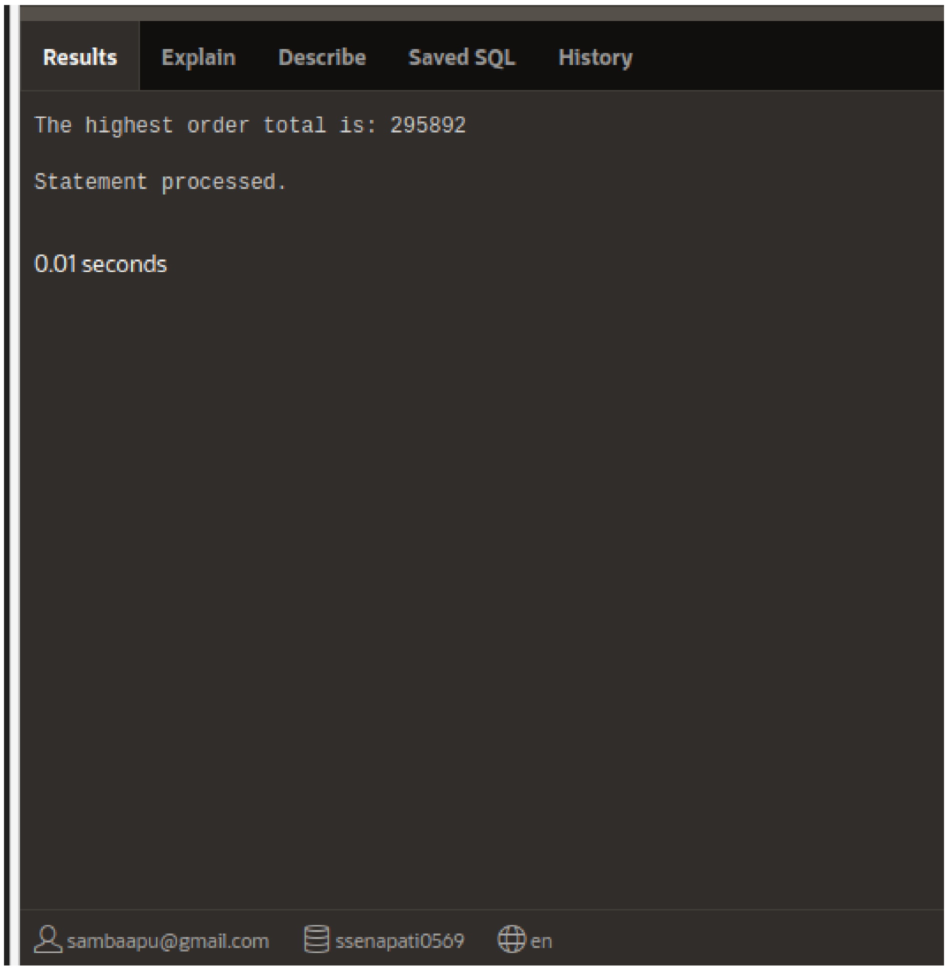
Function

```
CREATE OR REPLACE FUNCTION highest_total_order RETURN NUMBER
IS
max_total NUMBER;
BEGIN
    SELECT MAX(ORDER_TOTAL) INTO max_total FROM OEHR_ORDERS;
    RETURN max_total;

END;
```

Anonymous Block

```
DECLARE
  highest_total NUMBER;
BEGIN
  highest_total := highest_total_order;
  DBMS_OUTPUT.PUT_LINE('The highest order total is: ' || highest_total);

END;
```

Output



```
The highest order total is: 295892

Statement processed.
```

0.01 seconds

Question 8:

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.

Procedure

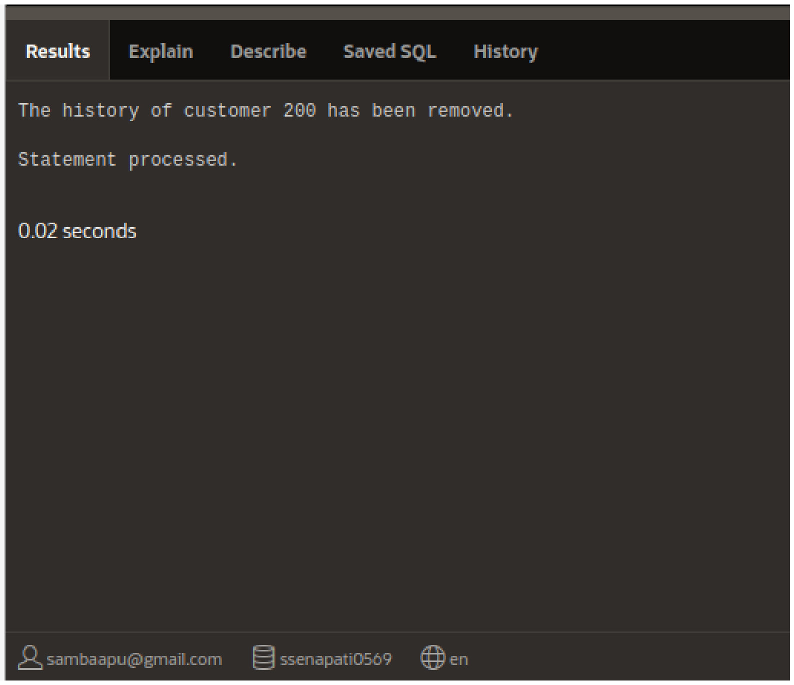
```
CREATE OR REPLACE PROCEDURE Remove_History (cust_id NUMBER)
IS
  no_rows_deleted NUMBER;
BEGIN
  DELETE FROM OEHR_JOB_HISTORY
  WHERE EMPLOYEE_ID = cust_id;

  no_rows_deleted := SQL%ROWCOUNT;

  IF no_rows_deleted > 0 THEN
    DBMS_OUTPUT.PUT_LINE('The history of customer ' || cust_id || ' has been
removed.');
  ELSE
    RAISE NO_DATA_FOUND;
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No history found for customer ' || cust_id || '.');
END;
```

Anonymous Block

```
DECLARE
  cid NUMBER := 200;
BEGIN
  Remove_History(cid);

END;
```

Output

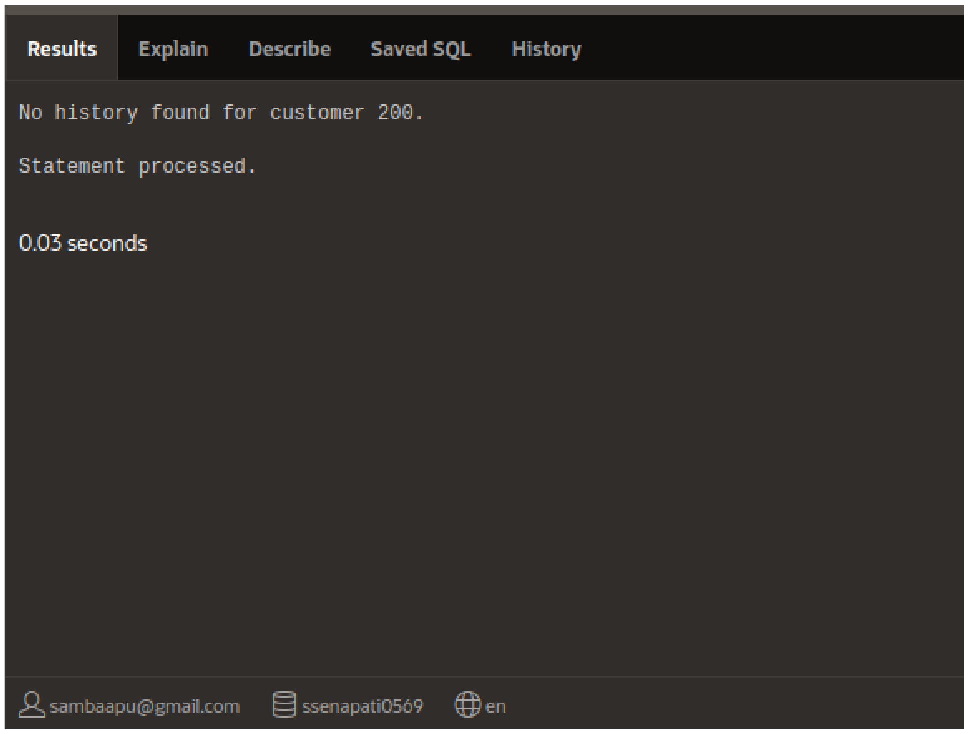
1st Run



```
The history of customer 200 has been removed.

Statement processed.
```

0.02 seconds

2nd Run



```
No history found for customer 200.

Statement processed.
```

0.03 seconds

Question 9 (a):

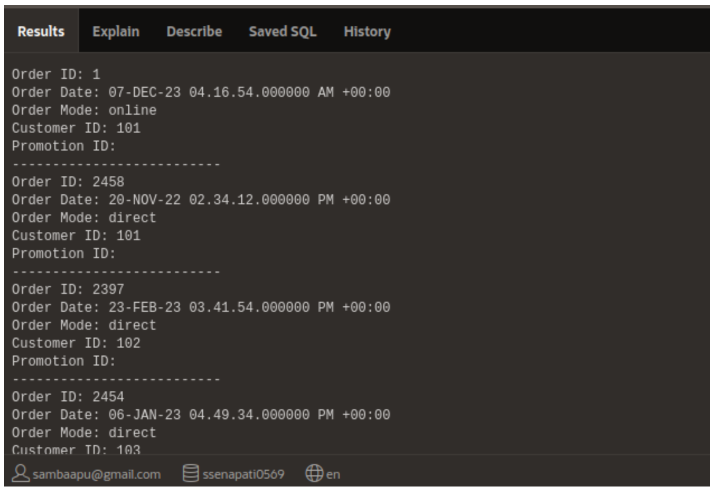
Define PL/SQL record named order_rec that holds the following columns of the table OEHR_ORDERS: ORDER_ID, ORDER_DATE, ORDER_MODE, CUSTOMER_ID and PROMOTION_ID.

Use a cursor to fetch all data into a variable of type order_rec and to display them.

CODE

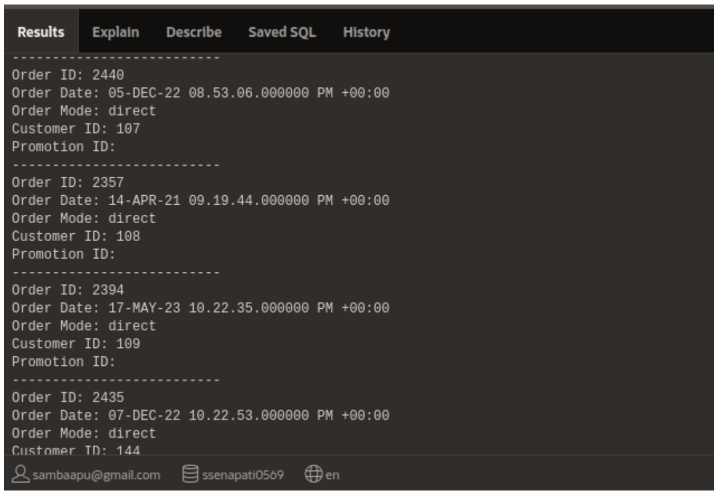
```
DECLARE
  TYPE order_rec IS RECORD (
    ord_id OEHR_ORDERS.ORDER_ID%TYPE,
    ord_date OEHR_ORDERS.ORDER_DATE%TYPE,
    ord_mode OEHR_ORDERS.ORDER_MODE%TYPE,
    cust_id OEHR_ORDERS.CUSTOMER_ID%TYPE,
    prom_id OEHR_ORDERS.PROMOTION_ID%TYPE
  );
  v_order order_rec;
  CURSOR c_order IS
    SELECT ORDER_ID, ORDER_DATE, ORDER_MODE, CUSTOMER_ID, PROMOTION_ID
    FROM OEHR_ORDERS;
BEGIN
  OPEN c_order;
  LOOP
    FETCH c_order INTO v_order;
    EXIT WHEN c_order%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Order ID: ' || v_order.ord_id);
    DBMS_OUTPUT.PUT_LINE('Order Date: ' || v_order.ord_date);
    DBMS_OUTPUT.PUT_LINE('Order Mode: ' || v_order.ord_mode);
    DBMS_OUTPUT.PUT_LINE('Customer ID: ' || v_order.cust_id);
    DBMS_OUTPUT.PUT_LINE('Promotion ID: ' || v_order.prom_id);
    DBMS_OUTPUT.PUT_LINE('--------------------------');
  END LOOP;
  CLOSE c_order;

END;
```

Output


Results Explain Describe Saved SQL History

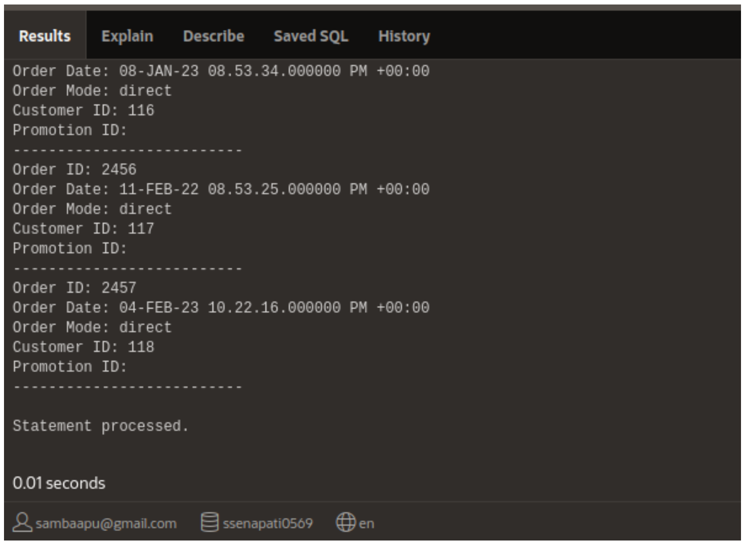
```
Order ID: 1
Order Date: 07-DEC-23 04.16.54.000000 AM +00:00
Order Mode: online
Customer ID: 101
Promotion ID:
-------------------------
Order ID: 2458
Order Date: 20-NOV-22 02.34.12.000000 PM +00:00
Order Mode: direct
Customer ID: 101
Promotion ID:
-------------------------
Order ID: 2397
Order Date: 23-FEB-23 03.41.54.000000 PM +00:00
Order Mode: direct
Customer ID: 102
Promotion ID:
-------------------------
Order ID: 2454
Order Date: 06-JAN-23 04.49.34.000000 PM +00:00
Order Mode: direct
Customer ID: 103
```

sambaapu@gmail.com ssenapati0569 en


Results Explain Describe Saved SQL History

```
-------------------------
Order ID: 2440
Order Date: 05-DEC-22 08.53.06.000000 PM +00:00
Order Mode: direct
Customer ID: 107
Promotion ID:
-------------------------
Order ID: 2357
Order Date: 14-APR-21 09.19.44.000000 PM +00:00
Order Mode: direct
Customer ID: 108
Promotion ID:
-------------------------
Order ID: 2394
Order Date: 17-MAY-23 10.22.35.000000 PM +00:00
Order Mode: direct
Customer ID: 109
Promotion ID:
-------------------------
Order ID: 2435
Order Date: 07-DEC-22 10.22.53.000000 PM +00:00
Order Mode: direct
Customer ID: 144
```

sambaapu@gmail.com ssenapati0569 en


Results Explain Describe Saved SQL History

```
Order Date: 08-JAN-23 08.53.34.000000 PM +00:00
Order Mode: direct
Customer ID: 116
Promotion ID:
-------------------------
Order ID: 2456
Order Date: 11-FEB-22 08.53.25.000000 PM +00:00
Order Mode: direct
Customer ID: 117
Promotion ID:
-------------------------
Order ID: 2457
Order Date: 04-FEB-23 10.22.16.000000 PM +00:00
Order Mode: direct
Customer ID: 118
Promotion ID:
-------------------------

Statement processed.
```

0.01 seconds

sambaapu@gmail.com ssenapati0569 en

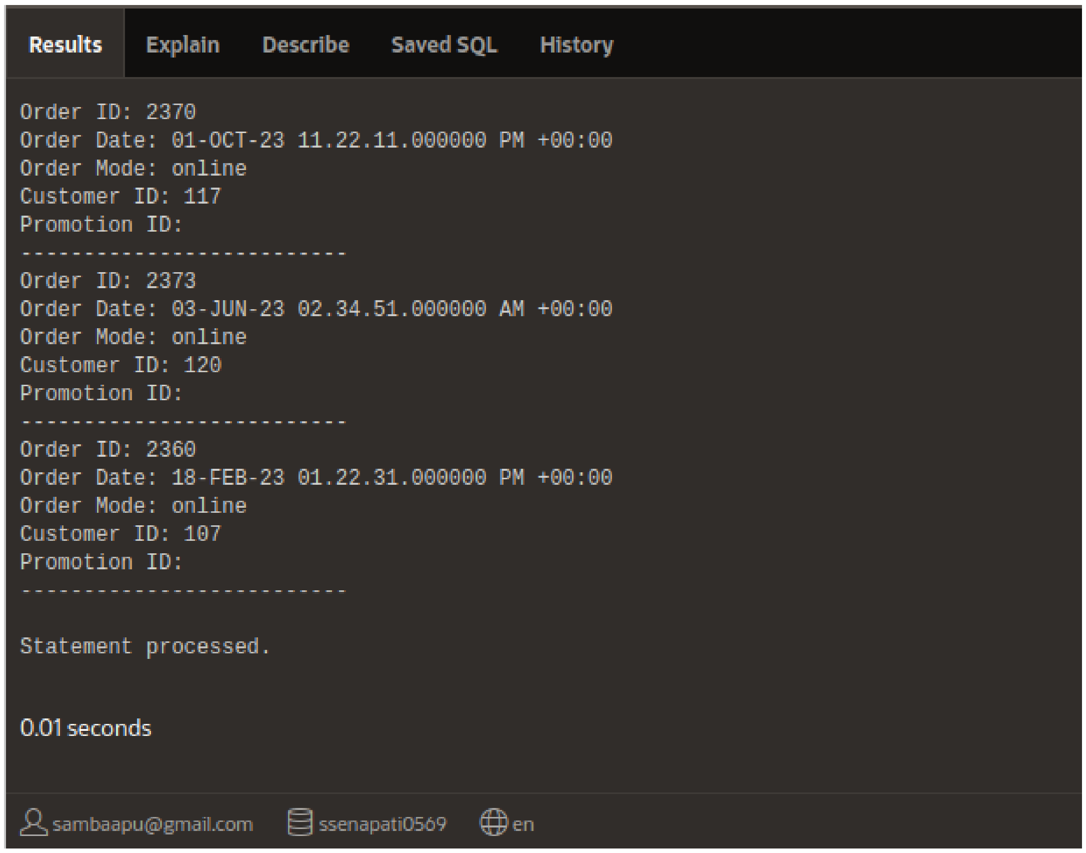
Question 9 (b): Modify your code in Question 9a to include all the following conditions:
- Only orders with no promotions are considered,

- Order mode: online,
- Order total less than 1000,
- Sort your result the most recent order first.

CODE

```
DECLARE
  TYPE order_rec IS RECORD (
    ord_id OEHR_ORDERS.ORDER_ID%TYPE,
    ord_date OEHR_ORDERS.ORDER_DATE%TYPE,
    ord_mode OEHR_ORDERS.ORDER_MODE%TYPE,
    cust_id OEHR_ORDERS.CUSTOMER_ID%TYPE,
    prom_id OEHR_ORDERS.PROMOTION_ID%TYPE
  );
  v_order order_rec;
  CURSOR c_order IS
    SELECT ORDER_ID, ORDER_DATE, ORDER_MODE, CUSTOMER_ID, PROMOTION_ID
    FROM OEHR_ORDERS
    WHERE promotion_id IS NULL -- Only orders with no promotions are considered
    AND order_mode = 'online' -- Order mode: online
    AND order_total < 1000 -- Order total less than 1000
    ORDER BY order_date DESC; -- most recent orders first
BEGIN
  OPEN c_order;
  LOOP
    FETCH c_order INTO v_order;
    EXIT WHEN c_order%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Order ID: ' || v_order.ord_id);
    DBMS_OUTPUT.PUT_LINE('Order Date: ' || v_order.ord_date);
    DBMS_OUTPUT.PUT_LINE('Order Mode: ' || v_order.ord_mode);
    DBMS_OUTPUT.PUT_LINE('Customer ID: ' || v_order.cust_id);
    DBMS_OUTPUT.PUT_LINE('Promotion ID: ' || v_order.prom_id);
    DBMS_OUTPUT.PUT_LINE('-------------------------');
  END LOOP;
  CLOSE c_order;
END;
```

Output



```
Order ID: 2370
Order Date: 01-OCT-23 11.22.11.000000 PM +00:00
Order Mode: online
Customer ID: 117
Promotion ID:
--------------------------
Order ID: 2373
Order Date: 03-JUN-23 02.34.51.000000 AM +00:00
Order Mode: online
Customer ID: 120
Promotion ID:
--------------------------
Order ID: 2360
Order Date: 18-FEB-23 01.22.31.000000 PM +00:00
Order Mode: online
Customer ID: 107
Promotion ID:
--------------------------

Statement processed.
```

0.01 seconds

Question 10:
a) Define PL/SQL trigger that fire for every row before an insert on the table OEHR_ORDERS as follows:

> **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.**

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

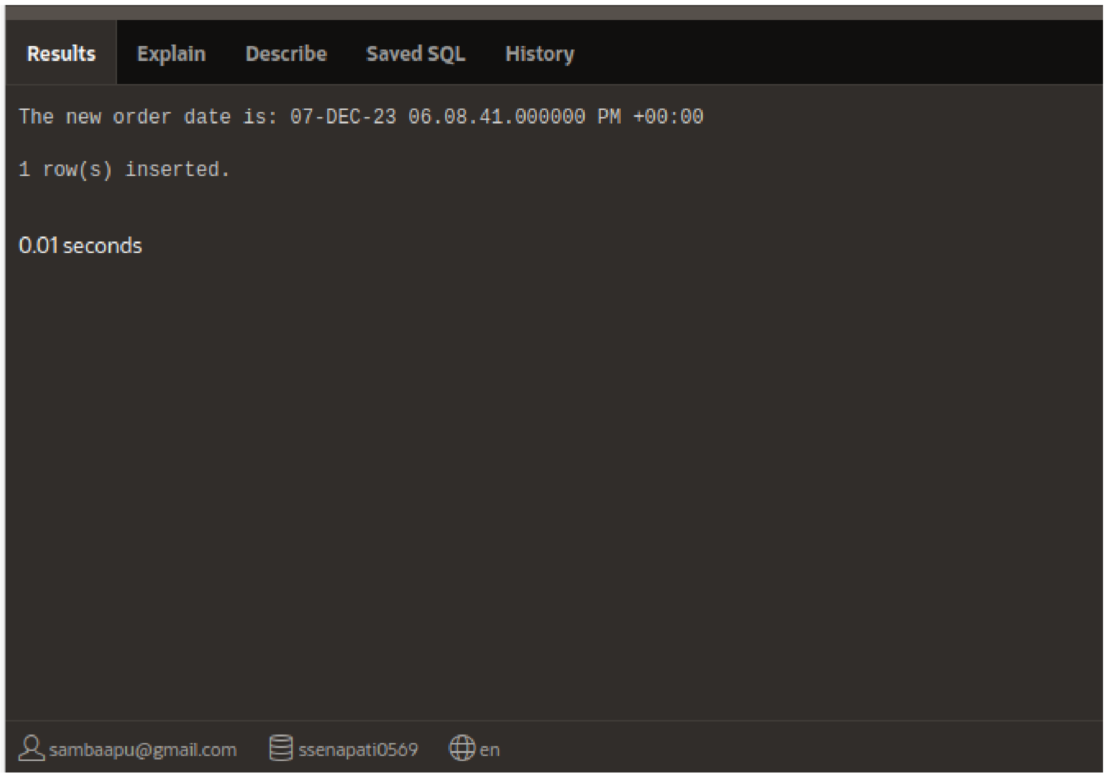
TRIGGER

```
CREATE OR REPLACE TRIGGER before_insert_order
BEFORE INSERT ON OEHR_ORDERS
FOR EACH ROW
WHEN (NEW.order_mode = 'online')
BEGIN
  DBMS_OUTPUT.PUT_LINE('The new order date is: ' || :NEW.order_date);
END;
```

Insert Code

```
INSERT INTO OEHR_ORDERS (order_id, order_date, order_mode, customer_id)
VALUES (1, SYSDATE, 'online', 101);
```

Output



```
The new order date is: 07-DEC-23 06.08.41.000000 PM +00:00

1 row(s) inserted.
```

0.01 seconds

Question 11: 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
```

<u>Helper Function</u>

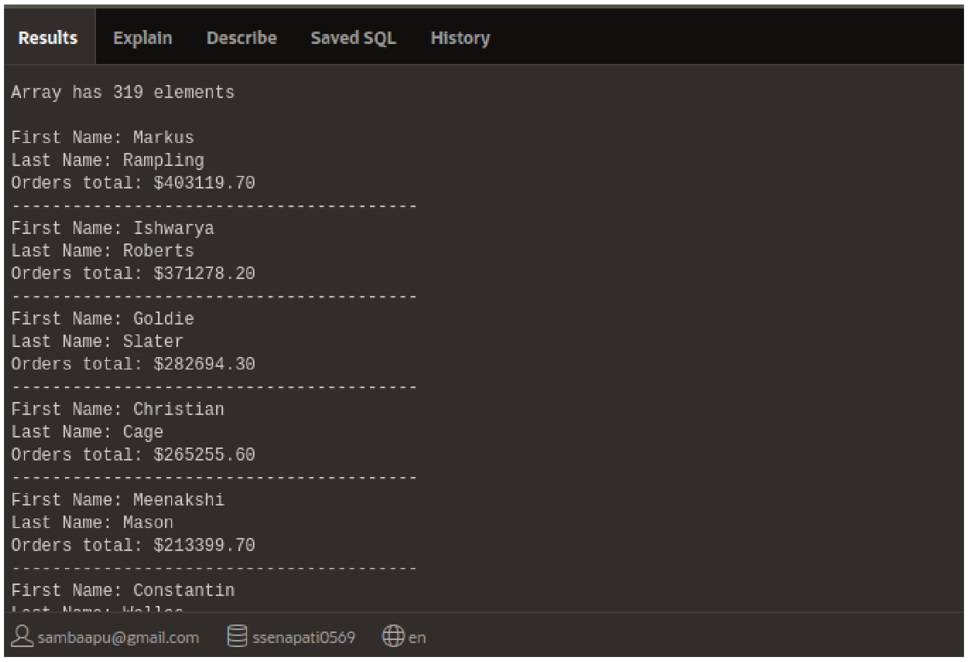
```
CREATE OR REPLACE FUNCTION calc_order_total(cust_id NUMBER) RETURN NUMBER IS
    total NUMBER := 0;
  BEGIN
    SELECT NVL(SUM(ORDER_TOTAL), 0)
    INTO total
    FROM OEHR_ORDERS
    WHERE CUSTOMER_ID = cust_id;

    RETURN total;

  END calc_order_total;
```

<u>Anonymous block</u>

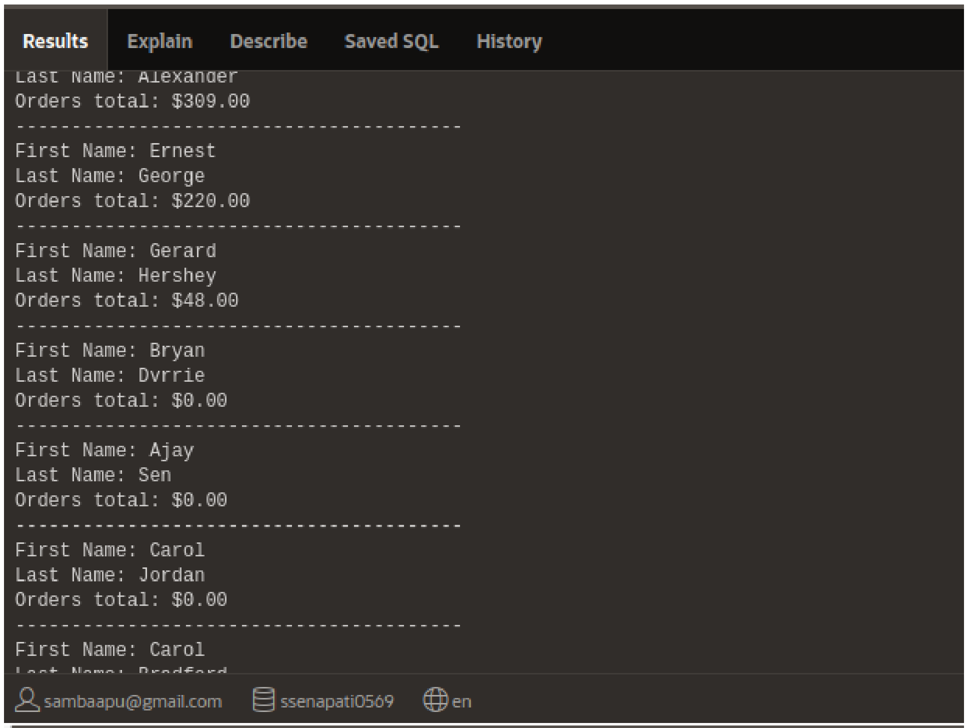
```
DECLARE
  TYPE cust_rec IS RECORD (
    fname VARCHAR2(50),
    lname VARCHAR2(50),
    ord_total NUMBER
  );

  TYPE cust_array IS VARRAY(500) OF cust_rec;

  v_customers cust_array := cust_array();

  CURSOR c_customers
  IS
    SELECT CUST_FIRST_NAME, CUST_LAST_NAME, calc_order_total(CUSTOMER_ID) AS total
    FROM OEHR_CUSTOMERS
    ORDER BY total DESC;
BEGIN
```

```
   FOR cust IN c_customers
   LOOP
      -- ADD  an element
      v_customers.EXTEND;
      -- ADD a value to the element (Assign values)
      v_customers(v_customers.LAST).fname := cust.CUST_FIRST_NAME;
      v_customers(v_customers.LAST).lname := cust.CUST_LAST_NAME;
      v_customers(v_customers.LAST).ord_total := cust.total;
   END LOOP;

   DBMS_OUTPUT.PUT_LINE('Array has ' || v_customers.COUNT || ' elements');
   DBMS_OUTPUT.PUT_LINE('');
   FOR idx IN v_customers.FIRST..v_customers.LAST
   LOOP
      DBMS_OUTPUT.PUT_LINE('First Name: ' || v_customers(idx).fname);
      DBMS_OUTPUT.PUT_LINE('Last Name: ' || v_customers(idx).lname);
      DBMS_OUTPUT.PUT_LINE('Orders total: $' || TO_CHAR(v_customers(idx).ord_total,
'fm99999990.00'));
      DBMS_OUTPUT.PUT_LINE('----------------------------------------');
   END LOOP;

END;
```

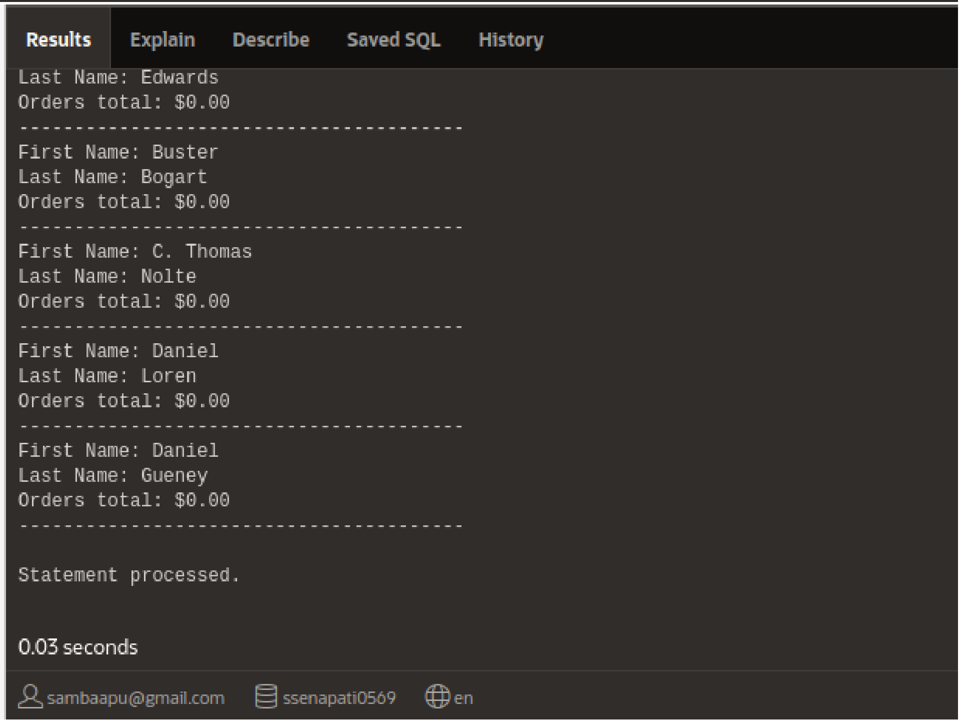
OUTPUT



```
Array has 319 elements

First Name: Markus
Last Name: Rampling
Orders total: $403119.70
----------------------------------------
First Name: Ishwarya
Last Name: Roberts
Orders total: $371278.20
----------------------------------------
First Name: Goldie
Last Name: Slater
Orders total: $282694.30
----------------------------------------
First Name: Christian
Last Name: Cage
Orders total: $265255.60
----------------------------------------
First Name: Meenakshi
Last Name: Mason
Orders total: $213399.70
----------------------------------------
First Name: Constantin
```



```
Last Name: Alexander
Orders total: $309.00
----------------------------------------
First Name: Ernest
Last Name: George
Orders total: $220.00
----------------------------------------
First Name: Gerard
Last Name: Hershey
Orders total: $48.00
----------------------------------------
First Name: Bryan
Last Name: Dvrrie
Orders total: $0.00
----------------------------------------
First Name: Ajay
Last Name: Sen
Orders total: $0.00
----------------------------------------
First Name: Carol
Last Name: Jordan
Orders total: $0.00
----------------------------------------
First Name: Carol
Last Name: Bradford
```



```
Last Name: Edwards
Orders total: $0.00
----------------------------------------
First Name: Buster
Last Name: Bogart
Orders total: $0.00
----------------------------------------
First Name: C. Thomas
Last Name: Nolte
Orders total: $0.00
----------------------------------------
First Name: Daniel
Last Name: Loren
Orders total: $0.00
----------------------------------------
First Name: Daniel
Last Name: Gueney
Orders total: $0.00
----------------------------------------

Statement processed.
```

0.03 seconds