**Individual Assessment**

**Member 3: Loh Wei Jia (2207414)**

|  |  |
| --- | --- |
| **Two Queries** | **1.Display the name of the menu item and price by Restaurant\_ID.**  SELECT name, price  FROM menuitem m, restaurant r  WHERE r.restaurant\_ID = m.restaurant\_ID  AND r.restaurant\_ID = 'R001';    This query selects the column name and price from the 'menuitem' table, specified as 'm', and joins it with the 'restaurant' table, specified as 'r', linking menu items to their respective restaurants. It then filters records where the restaurant\_id matches 'R001'.  **2. Display the employee which their age is equal or below 30.**  SELECT employee\_ID,(FirstName||''||LastName)AS"Employee Name", ROUND((SYSDATE-dob)/365.25) AS "Age"  FROM employee  WHERE (SYSDATE-dob)/365.25 <= 30;    This query selects the employee\_id from the 'employee' table and concatenates the firstname and lastname columns into 'employee\_name'. It calculates the age of each employee in years and filters the results to include only those aged 30 or younger. |
| **Two Stored Procedure** | **1.This procedure is to change the number of inventories.**  CREATE OR REPLACE PROCEDURE updateInventory  (  id in CHAR,  numInv in NUMBER  )  IS  BEGIN  UPDATE Inventory  SET quantity = numInv  WHERE inventory\_ID = id;  COMMIT;  END;  **/**  execute updateInventory('I005','10');  SELECT \*  FROM inventory  WHERE inventory\_ID='I005';      This procedure is to change the number of inventories. It takes two parameters: id and numInv. Inside the procedure, it updates the quantity column in the Inventory table where the inventory\_ID matches the provided ID. After updating, it commits the transaction. To execute the procedure, we use the EXECUTE command followed by the procedure name and its parameters which is I005 and 10. Finally, we verify the update by querying the inventory table for the item with the specified ID.  **2.Change Order date by using OrderID.**  CREATE OR REPLACE PROCEDURE change\_date  (  current\_OrderID IN CHAR,  current\_ODate IN TIMESTAMP  )  IS  BEGIN  UPDATE Orders  SET Order\_Time= current\_ODate  WHERE Order\_ID = current\_OrderID;  COMMIT;  END;  /  EXECUTE change\_date('OD0001', TO\_DATE('16-04-2024 12:11:33', 'DD-MM-YYYY HH24:MI:SS'));  SELECT\*  FROM Orders  WHERE Order\_ID='OD0001';    This procedure is to change Order date. It takes two parameters: current\_OrderID and current\_ODate, representing the order ID and the new order time, respectively. Inside the procedure, it updates the Order\_Time column in the Orders table where the Order\_ID matches the provided order ID. After updating, it commits the transaction to make the changes permanent. To execute the procedure, we use the EXECUTE command followed by the procedure name and its parameters. Finally, we verify the update by querying the Orders table for the order with the specified ID. |
| **Two Functions** | **1.To show the address of employee by using employeeID.**  CREATE OR REPLACE FUNCTION showEmpAddress  (  E\_ID CHAR  )  RETURN VARCHAR2  IS  e\_address VARCHAR2(100);  BEGIN  SELECT Address  INTO e\_address  FROM Employee  WHERE Employee\_ID = E\_ID;  RETURN E\_address;  END;  /  SET SERVEROUTPUT ON;  DECLARE  Emp\_ID CHAR(5);  Emp\_address VARCHAR2(100);  BEGIN  Emp\_ID := '&Emp\_ID';  Emp\_address := showEmpAddress(Emp\_ID);  DBMS\_OUTPUT.PUT\_LINE('The address of '||Emp\_ID||' is '||Emp\_address||'.');  END;  /    This script shows the address of employee by using employeeID. Inside the function, it selects the address based on the provided employee ID and assigns it to the e\_address variable before returning it. After defining the function, the script prompts the user to enter an employee ID (Emp\_ID). When we enter the example id which is E4165, the system calls the showEmpAddress function with the employee id and retrieve the address. It prints out the id along with their address using the DBMS\_OUTPUT.PUT\_LINE function. From the situation above, it will display the message “The address of E4165 is 52000 Taman Boleh, Kuala Lumpur.” after entering the ID.  2.**This function enable to count the total number of members.**  CREATE OR REPLACE FUNCTION countMember  RETURN NUMBER  IS  totalMember NUMBER;  BEGIN  SELECT count(\*) INTO totalMember  FROM Member;  RETURN totalMember;  END;  /  DECLARE  total\_Member NUMBER;  BEGIN  total\_Member := countMember;  DBMS\_OUTPUT.PUT\_LINE('The total number of the member is '||total\_Member||'.');  END;  /    This function enables to count the total number of members1. Inside the function, it counts the number of rows in the Member table using the COUNT (\*) function and assigns the result to the totalMember variable before returning it. After defining the function, the script declares a variable named total\_Member and assigns the result of calling the countMember function to it. Then, it prints out the total number of members using the DBMS\_OUTPUT.PUT\_LINE function. After running the function, it displays the message “The total number of the member is 8.” |