ST10225619

Mubasher Hussain

ADDB6311

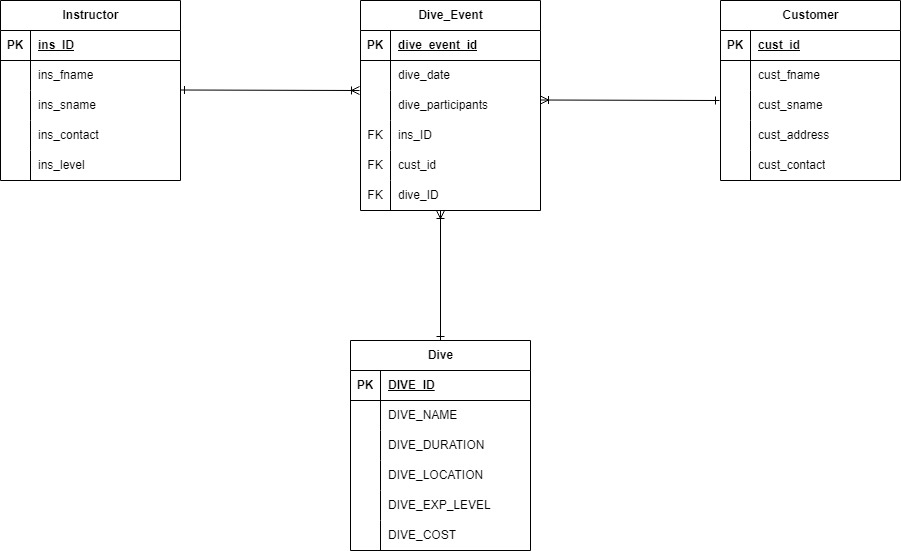
Practicum

24/06/2024

----------------------------------------------------------------------------------------------------------

GitHub Link to repository:

https://github.com/mubasherhu/ADDB6311-Practicum-Exam.git

Question 1   


CREATE TABLE INSTRUCTOR (

ins\_ID VARCHAR2(10) NOT NULL,

ins\_fname VARCHAR2(50) NOT NULL,

ins\_sname VARCHAR2(50) NOT NULL,

ins\_contact VARCHAR2(15) NOT NULL,

ins\_level NUMBER NOT NULL,

CONSTRAINT INSTRUCTOR\_PK PRIMARY KEY (ins\_ID)

);

CREATE TABLE CUSTOMER (

cust\_id VARCHAR2(10) NOT NULL,

cust\_fname VARCHAR2(50) NOT NULL,

cust\_sname VARCHAR2(50) NOT NULL,

cust\_address VARCHAR2(100) NOT NULL,

cust\_contact VARCHAR2(15) NOT NULL,

CONSTRAINT CUSTOMER\_PK PRIMARY KEY (cust\_id)

);

CREATE TABLE DIVE (

DIVE\_ID NUMBER NOT NULL,

DIVE\_NAME VARCHAR2(50) NOT NULL,

DIVE\_DURATION VARCHAR2(20),

DIVE\_LOCATION VARCHAR2(50),

DIVE\_EXP\_LEVEL NUMBER,

DIVE\_COST NUMBER,

CONSTRAINT DIVE\_PK PRIMARY KEY (DIVE\_ID)

);

CREATE TABLE DIVE\_EVENT (

dive\_event\_id VARCHAR2(10) NOT NULL,

dive\_date DATE NOT NULL,

dive\_participants NUMBER NOT NULL,

ins\_ID VARCHAR2(10) NOT NULL,

cust\_id VARCHAR2(10) NOT NULL,

dive\_ID NUMBER NOT NULL,

CONSTRAINT DIVE\_EVENT\_PK PRIMARY KEY (dive\_event\_id),

CONSTRAINT DIVE\_EVENT\_FK\_INS\_ID FOREIGN KEY (ins\_ID) REFERENCES INSTRUCTOR(ins\_ID),

CONSTRAINT DIVE\_EVENT\_FK\_CUST\_ID FOREIGN KEY (cust\_id) REFERENCES CUSTOMER(cust\_id),

CONSTRAINT DIVE\_EVENT\_FK\_DIVE\_ID FOREIGN KEY (dive\_ID) REFERENCES DIVE(dive\_ID)

);

INSERT INTO INSTRUCTOR (ins\_ID, ins\_fname, ins\_sname, ins\_contact, ins\_level) VALUES

('101', 'James', 'Willis', '0843569851', 7);

INSERT INTO INSTRUCTOR (ins\_ID, ins\_fname, ins\_sname, ins\_contact, ins\_level) VALUES

('102', 'Sam', 'Wait', '0763698521', 2);

INSERT INTO INSTRUCTOR (ins\_ID, ins\_fname, ins\_sname, ins\_contact, ins\_level) VALUES

('103', 'Sally', 'Gumede', '0786598521', 8);

INSERT INTO INSTRUCTOR (ins\_ID, ins\_fname, ins\_sname, ins\_contact, ins\_level) VALUES

('104', 'Bob', 'Du Preez', '0796369857', 3);

INSERT INTO INSTRUCTOR (ins\_ID, ins\_fname, ins\_sname, ins\_contact, ins\_level) VALUES

('105', 'Simon', 'Jones', '0826598741', 9);

INSERT INTO CUSTOMER (cust\_id, cust\_fname, cust\_sname, cust\_address, cust\_contact) VALUES

('C115', 'Heinrich', 'Willis', '3 Main Road', '0821253659');

INSERT INTO CUSTOMER (cust\_id, cust\_fname, cust\_sname, cust\_address, cust\_contact) VALUES

('C116', 'David', 'Watson', '13 Cape Road', '0769658547');

INSERT INTO CUSTOMER (cust\_id, cust\_fname, cust\_sname, cust\_address, cust\_contact) VALUES

('C117', 'Waldo', 'Smith', '3 Mountain Road', '0863256574');

INSERT INTO CUSTOMER (cust\_id, cust\_fname, cust\_sname, cust\_address, cust\_contact) VALUES

('C118', 'Alex', 'Hanson', '8 Circle Road', '0762356587');

INSERT INTO CUSTOMER (cust\_id, cust\_fname, cust\_sname, cust\_address, cust\_contact) VALUES

('C119', 'Kuhle', 'Bitterhout', '15 Main Road', '0821235258');

INSERT INTO DIVE (DIVE\_ID, DIVE\_NAME, DIVE\_DURATION, DIVE\_LOCATION, DIVE\_EXP\_LEVEL, DIVE\_COST) VALUES

(550, 'Shark Dive', '3 hours', 'Shark Point', 8, 500);

INSERT INTO DIVE (DIVE\_ID, DIVE\_NAME, DIVE\_DURATION, DIVE\_LOCATION, DIVE\_EXP\_LEVEL, DIVE\_COST) VALUES

(551, 'Coral Dive', '1 hour', 'Break Point', 7, 300);

INSERT INTO DIVE (DIVE\_ID, DIVE\_NAME, DIVE\_DURATION, DIVE\_LOCATION, DIVE\_EXP\_LEVEL, DIVE\_COST) VALUES

(552, 'Wave Cresect', '2 hours', 'Ship wreck ally', 3, 800);

INSERT INTO DIVE (DIVE\_ID, DIVE\_NAME, DIVE\_DURATION, DIVE\_LOCATION, DIVE\_EXP\_LEVEL, DIVE\_COST) VALUES

(553, 'Underwater Adventure', '1 hour', 'Coral ally', 2, 250);

INSERT INTO DIVE (DIVE\_ID, DIVE\_NAME, DIVE\_DURATION, DIVE\_LOCATION, DIVE\_EXP\_LEVEL, DIVE\_COST) VALUES

(554, 'Underwater Adventure', '3 hours', 'Sandy Beach', 3, 750);

INSERT INTO DIVE (DIVE\_ID, DIVE\_NAME, DIVE\_DURATION, DIVE\_LOCATION, DIVE\_EXP\_LEVEL, DIVE\_COST) VALUES

(555, 'Deep Blue Ocean', '30 Minutes', 'Lazy Waves', 2, 120);

INSERT INTO DIVE (DIVE\_ID, DIVE\_NAME, DIVE\_DURATION, DIVE\_LOCATION, DIVE\_EXP\_LEVEL, DIVE\_COST) VALUES

(556, 'Rough Seas', '1 hour', 'Pipe', 9, 700);

INSERT INTO DIVE (DIVE\_ID, DIVE\_NAME, DIVE\_DURATION, DIVE\_LOCATION, DIVE\_EXP\_LEVEL, DIVE\_COST) VALUES

(557, 'White Water', '2 hours', 'Drifts', 5, 200);

INSERT INTO DIVE (DIVE\_ID, DIVE\_NAME, DIVE\_DURATION, DIVE\_LOCATION, DIVE\_EXP\_LEVEL, DIVE\_COST) VALUES

(558, 'Current Adventure', '2 hours', 'Rock Lands', 3, 150);

-- Inserting into DIVE\_EVENT

INSERT INTO DIVE\_EVENT (dive\_event\_id, dive\_date, dive\_participants, ins\_ID, cust\_id, dive\_ID)

VALUES ('de\_101', TO\_DATE('2017-07-15', 'YYYY-MM-DD'), 5, '103', 'C115', 558);

INSERT INTO DIVE\_EVENT (dive\_event\_id, dive\_date, dive\_participants, ins\_ID, cust\_id, dive\_ID)

VALUES ('de\_102', TO\_DATE('2017-07-16', 'YYYY-MM-DD'), 7, '102', 'C117', 555);

INSERT INTO DIVE\_EVENT (dive\_event\_id, dive\_date, dive\_participants, ins\_ID, cust\_id, dive\_ID)

VALUES ('de\_103', TO\_DATE('2017-07-18', 'YYYY-MM-DD'), 8, '104', 'C118', 552);

INSERT INTO DIVE\_EVENT (dive\_event\_id, dive\_date, dive\_participants, ins\_ID, cust\_id, dive\_ID)

VALUES ('de\_104', TO\_DATE('2017-07-19', 'YYYY-MM-DD'), 3, '101', 'C119', 551);

INSERT INTO DIVE\_EVENT (dive\_event\_id, dive\_date, dive\_participants, ins\_ID, cust\_id, dive\_ID)

VALUES ('de\_105', TO\_DATE('2017-07-21', 'YYYY-MM-DD'), 5, '104', 'C115', 558);

INSERT INTO DIVE\_EVENT (dive\_event\_id, dive\_date, dive\_participants, ins\_ID, cust\_id, dive\_ID)

VALUES ('de\_106', TO\_DATE('2017-07-22', 'YYYY-MM-DD'), 8, '105', 'C116', 556);

INSERT INTO DIVE\_EVENT (dive\_event\_id, dive\_date, dive\_participants, ins\_ID, cust\_id, dive\_ID)

VALUES ('de\_107', TO\_DATE('2017-07-25', 'YYYY-MM-DD'), 10, '105', 'C115', 554);

INSERT INTO DIVE\_EVENT (dive\_event\_id, dive\_date, dive\_participants, ins\_ID, cust\_id, dive\_ID)

VALUES ('de\_108', TO\_DATE('2017-07-27', 'YYYY-MM-DD'), 5, '101', 'C118', 552);

INSERT INTO DIVE\_EVENT (dive\_event\_id, dive\_date, dive\_participants, ins\_ID, cust\_id, dive\_ID)

VALUES ('de\_109', TO\_DATE('2017-07-28', 'YYYY-MM-DD'), 3, '102', 'C119', 553);

Question 2

--Question 2

--step 1 Craeting Roles

-- Create Administrator Role

CREATE ROLE C##admin\_role;

-- Create General User Role

CREATE ROLE C##user\_role;

-- Step 2: Assign Privileges

-- Grant Administrator Privileges

-- Administrator has full privileges on all tables

GRANT ALL PRIVILEGES ON INSTRUCTOR TO C##admin\_role;

GRANT ALL PRIVILEGES ON CUSTOMER TO C##admin\_role;

GRANT ALL PRIVILEGES ON DIVE TO C##admin\_role;

GRANT ALL PRIVILEGES ON DIVE\_EVENT TO C##admin\_role;

-- General User Privileges

-- General user has SELECT privileges and can perform INSERT operations

GRANT SELECT, INSERT ON INSTRUCTOR TO C##user\_role;

GRANT SELECT, INSERT ON CUSTOMER TO C##user\_role;

GRANT SELECT, INSERT ON DIVE TO C##user\_role;

GRANT SELECT, INSERT ON DIVE\_EVENT TO C##user\_role;

-- Step 3: Create Users and Assign Roles

-- Create Administrator User

CREATE USER C##admin\_user IDENTIFIED BY admin\_password;

GRANT C##admin\_role TO C##admin\_user;

-- Create General User

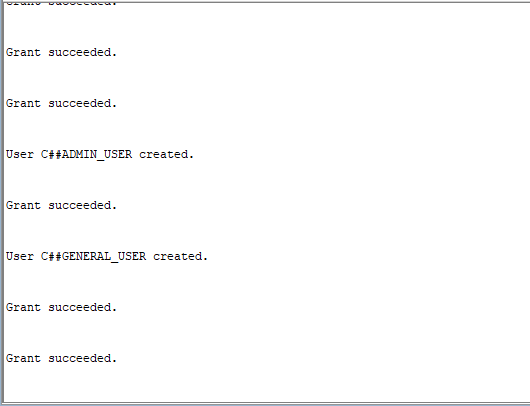
CREATE USER C##general\_user IDENTIFIED BY user\_password;

GRANT C##user\_role TO C##general\_user;

-- Optional: Grant Connect Privilege (allows the user to connect to the database)

GRANT CONNECT TO C##admin\_user;

GRANT CONNECT TO C##general\_user;



Question 3

--Question 3

SELECT

i.ins\_fname || ' ' || i.ins\_sname AS instructor\_name,

c.cust\_fname || ' ' || c.cust\_sname AS customer\_full\_name,

d.DIVE\_LOCATION,

de.dive\_participants

FROM

DIVE\_EVENT de

JOIN

INSTRUCTOR i ON de.ins\_ID = i.ins\_ID

JOIN

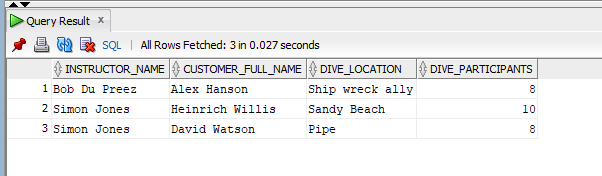
CUSTOMER c ON de.cust\_id = c.cust\_id

JOIN

DIVE d ON de.dive\_ID = d.DIVE\_ID

WHERE

de.dive\_participants BETWEEN 8 AND 10;



Question 4

SET SERVEROUTPUT ON;

DECLARE

-- Cursor to fetch dive name, dive date, and number of participants for events with 10 or more participants

CURSOR dive\_event\_cursor IS

SELECT

d.DIVE\_NAME,

de.dive\_date,

de.dive\_participants

FROM

DIVE\_EVENT de

JOIN

DIVE d ON de.dive\_ID = d.DIVE\_ID

WHERE

de.dive\_participants >= 10;

-- Record type to hold the cursor results

dive\_event\_record dive\_event\_cursor%ROWTYPE;

BEGIN

-- Open the cursor

OPEN dive\_event\_cursor;

-- Loop through the cursor and display the results

LOOP

FETCH dive\_event\_cursor INTO dive\_event\_record;

EXIT WHEN dive\_event\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Dive Name: ' || dive\_event\_record.DIVE\_NAME);

DBMS\_OUTPUT.PUT\_LINE('Allocated Date: ' || TO\_CHAR(dive\_event\_record.dive\_date, 'YYYY-MM-DD'));

DBMS\_OUTPUT.PUT\_LINE('Number of Participants: ' || dive\_event\_record.dive\_participants);

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

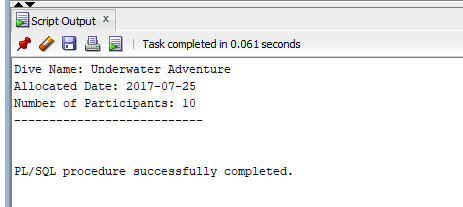
END LOOP;

-- Close the cursor

CLOSE dive\_event\_cursor;

END;

/



QUESTION 5

SET SERVEROUTPUT ON;

DECLARE

-- Cursor to fetch customer full name, dive event name, dive participants, and dive cost for events costing over R500

CURSOR dive\_event\_cursor IS

SELECT

c.cust\_fname || ' ' || c.cust\_sname AS customer\_full\_name,

d.DIVE\_NAME,

de.dive\_participants,

d.DIVE\_COST

FROM

DIVE\_EVENT de

JOIN

CUSTOMER c ON de.cust\_id = c.cust\_id

JOIN

DIVE d ON de.dive\_ID = d.DIVE\_ID

WHERE

d.DIVE\_COST > 500;

-- Record type to hold the cursor results

dive\_event\_record dive\_event\_cursor%ROWTYPE;

-- Variable to store the number of instructors required

num\_instructors NUMBER;

BEGIN

-- Open the cursor

OPEN dive\_event\_cursor;

-- Loop through the cursor and display the results

LOOP

FETCH dive\_event\_cursor INTO dive\_event\_record;

EXIT WHEN dive\_event\_cursor%NOTFOUND;

-- Determine the number of instructors required based on the number of participants

IF dive\_event\_record.dive\_participants <= 4 THEN

num\_instructors := 1;

ELSIF dive\_event\_record.dive\_participants BETWEEN 5 AND 7 THEN

num\_instructors := 2;

ELSE

num\_instructors := 3;

END IF;

-- Display the result

DBMS\_OUTPUT.PUT\_LINE('Customer Full Name: ' || dive\_event\_record.customer\_full\_name);

DBMS\_OUTPUT.PUT\_LINE('Dive Event Name: ' || dive\_event\_record.DIVE\_NAME);

DBMS\_OUTPUT.PUT\_LINE('Dive Participants: ' || dive\_event\_record.dive\_participants);

DBMS\_OUTPUT.PUT\_LINE('Number of Instructors Required: ' || num\_instructors);

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

END LOOP;

-- Display cursor attributes

DBMS\_OUTPUT.PUT\_LINE('Total Rows Processed: ' || dive\_event\_cursor%ROWCOUNT);

IF dive\_event\_cursor%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Last Row Fetched: ' || dive\_event\_cursor%ROWCOUNT);

ELSE

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

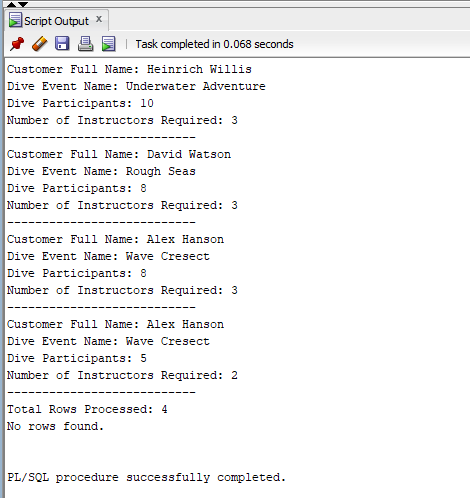
END IF;

-- Close the cursor

CLOSE dive\_event\_cursor;

END;

/



Question 6

SET SERVEROUTPUT ON

CREATE OR REPLACE VIEW Vw\_Dive\_Evnt AS

SELECT

de.ins\_ID,

de.cust\_id,

c.cust\_address,

d.DIVE\_DURATION

FROM

DIVE\_EVENT de

JOIN

CUSTOMER c ON de.cust\_id = c.cust\_id

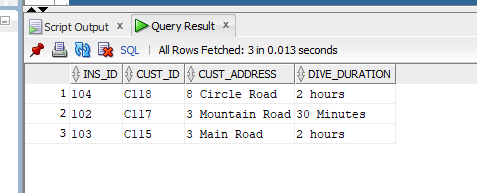
JOIN

DIVE d ON de.dive\_ID = d.DIVE\_ID

WHERE

de.dive\_date < TO\_DATE('2017-07-19', 'YYYY-MM-DD');

SELECT \* FROM Vw\_Dive\_Evnt;



Question 7

CREATE OR REPLACE TRIGGER New\_Dive\_Even

BEFORE INSERT OR UPDATE ON DIVE\_EVENT

FOR EACH ROW

BEGIN

IF :NEW.dive\_participants <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Number of participants must be greater than 0.');

ELSIF :NEW.dive\_participants > 20 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Number of participants must be 20 or less.');

END IF;

END;

/

-- Testing with 0 participants (should raise an error)

BEGIN

INSERT INTO DIVE\_EVENT (dive\_event\_id, dive\_date, dive\_participants, ins\_ID, cust\_id, dive\_ID)

VALUES ('test\_001', TO\_DATE('2023-06-01', 'YYYY-MM-DD'), 0, '101', 'C115', 550);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/

-- Testing with less than 0 participants (should raise an error)

BEGIN

INSERT INTO DIVE\_EVENT (dive\_event\_id, dive\_date, dive\_participants, ins\_ID, cust\_id, dive\_ID)

VALUES ('test\_002', TO\_DATE('2023-06-01', 'YYYY-MM-DD'), -5, '102', 'C116', 551);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/

-- Testing with more than 20 participants (should raise an error)

BEGIN

INSERT INTO DIVE\_EVENT (dive\_event\_id, dive\_date, dive\_participants, ins\_ID, cust\_id, dive\_ID)

VALUES ('test\_003', TO\_DATE('2023-06-01', 'YYYY-MM-DD'), 25, '103', 'C117', 552);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/

-- Testing with valid number of participants (should succeed)

BEGIN

INSERT INTO DIVE\_EVENT (dive\_event\_id, dive\_date, dive\_participants, ins\_ID, cust\_id, dive\_ID)

VALUES ('test\_004', TO\_DATE('2023-06-01', 'YYYY-MM-DD'), 10, '104', 'C118', 553);

DBMS\_OUTPUT.PUT\_LINE('Insert successful');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/

-- Testing with valid number of participants (should succeed)

BEGIN

INSERT INTO DIVE\_EVENT (dive\_event\_id, dive\_date, dive\_participants, ins\_ID, cust\_id, dive\_ID)

VALUES ('test\_004', TO\_DATE('2023-06-01', 'YYYY-MM-DD'), 10, '104', 'C118', 553);

DBMS\_OUTPUT.PUT\_LINE('Insert successful');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/

SET SERVEROUTPUT ON;

test updates

BEGIN

UPDATE DIVE\_EVENT

SET dive\_participants = 0

WHERE dive\_event\_id = 'test\_004';

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/

BEGIN

UPDATE DIVE\_EVENT

SET dive\_participants = 25

WHERE dive\_event\_id = 'test\_004';

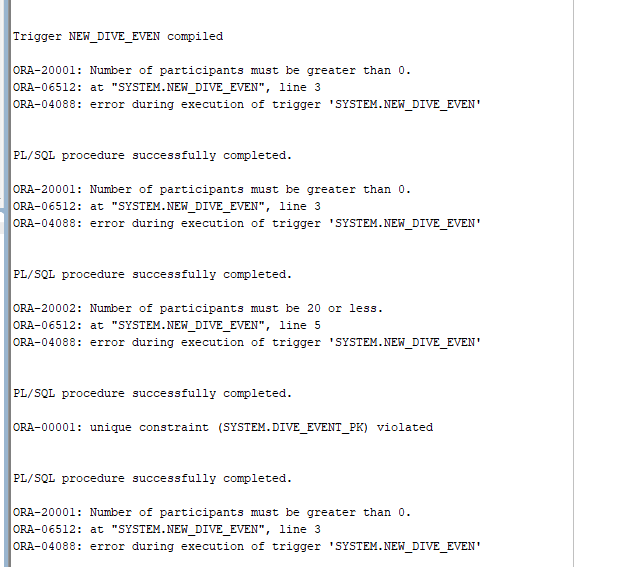
EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/



Question 8

CREATE OR REPLACE PROCEDURE sp\_Customer\_Details (

p\_cust\_id IN VARCHAR2,

p\_dive\_date IN DATE

)

IS

-- Variables to hold customer and dive names

v\_cust\_name VARCHAR2(100);

v\_dive\_name VARCHAR2(50);

BEGIN

-- Select the customer and dive names for the given customer ID and dive date

SELECT c.cust\_fname || ' ' || c.cust\_sname, d.DIVE\_NAME

INTO v\_cust\_name, v\_dive\_name

FROM DIVE\_EVENT de

JOIN CUSTOMER c ON de.cust\_id = c.cust\_id

JOIN DIVE d ON de.dive\_ID = d.DIVE\_ID

WHERE de.cust\_id = p\_cust\_id AND de.dive\_date = p\_dive\_date;

-- Display the results

DBMS\_OUTPUT.PUT\_LINE('Customer Name: ' || v\_cust\_name);

DBMS\_OUTPUT.PUT\_LINE('Dive Name: ' || v\_dive\_name);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

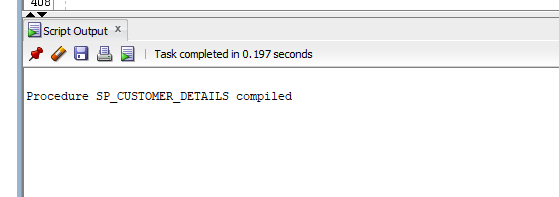
DBMS\_OUTPUT.PUT\_LINE('No customer found for the given ID and dive date.');

WHEN OTHERS THEN

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

END;

/



SET SERVEROUTPUT ON;

-- Execute the procedure with a valid customer ID and dive date

BEGIN

sp\_Customer\_Details('C115', TO\_DATE('2017-07-15', 'YYYY-MM-DD'));

END;

/

-- Execute the procedure with an invalid customer ID

BEGIN

sp\_Customer\_Details('INVALID\_ID', TO\_DATE('2017-07-15', 'YYYY-MM-DD'));

END;

/

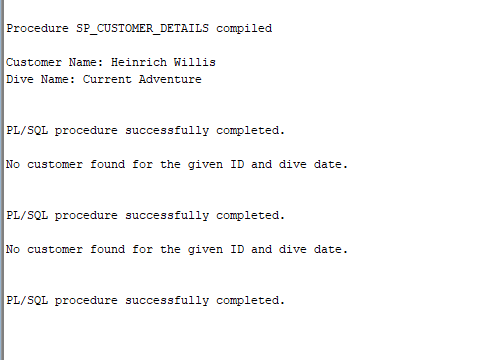
-- Execute the procedure with a valid customer ID but invalid dive date

BEGIN

sp\_Customer\_Details('C115', TO\_DATE('2019-01-01', 'YYYY-MM-DD'));

END;

/



Question 9

CREATE OR REPLACE FUNCTION calculate\_dive\_cost (

p\_dive\_event\_id IN VARCHAR2

) RETURN NUMBER

IS

-- Declare variables to hold the number of participants and cost per participant

v\_participants NUMBER;

v\_cost\_per\_participant NUMBER;

v\_total\_cost NUMBER;

BEGIN

-- Objective: Fetch the number of participants and cost per participant for the given dive event ID

-- Fetch the number of participants from the DIVE\_EVENT table

SELECT dive\_participants

INTO v\_participants

FROM DIVE\_EVENT

WHERE dive\_event\_id = p\_dive\_event\_id;

-- Fetch the cost per participant from the DIVE table via the DIVE\_EVENT table

SELECT d.DIVE\_COST

INTO v\_cost\_per\_participant

FROM DIVE d

JOIN DIVE\_EVENT de ON d.DIVE\_ID = de.dive\_ID

WHERE de.dive\_event\_id = p\_dive\_event\_id;

-- Perform the calculation: total cost = number of participants \* cost per participant

v\_total\_cost := v\_participants \* v\_cost\_per\_participant;

-- Return the total cost

RETURN v\_total\_cost;

EXCEPTION

-- Handle exceptions

-- If no data is found for the given dive event ID, return 0 and display an error message

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No data found for the given dive event ID: ' || p\_dive\_event\_id);

RETURN 0;

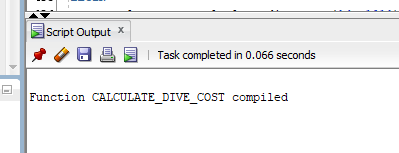
-- Handle any other exceptions

WHEN OTHERS THEN

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

RETURN 0;

END calculate\_dive\_cost;

/  


-- Ensure server output is enabled to see the DBMS\_OUTPUT.PUT\_LINE messages

SET SERVEROUTPUT ON;

-- Test the function with a valid dive event ID

DECLARE

v\_total\_cost NUMBER;

BEGIN

v\_total\_cost := calculate\_dive\_cost('de\_101');

DBMS\_OUTPUT.PUT\_LINE('Total Cost for Dive Event de\_101: ' || v\_total\_cost);

END;

/

-- Test the function with an invalid dive event ID

DECLARE

v\_total\_cost NUMBER;

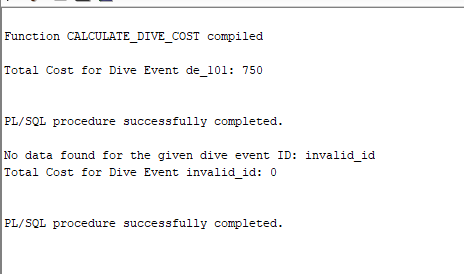
BEGIN

v\_total\_cost := calculate\_dive\_cost('invalid\_id');

DBMS\_OUTPUT.PUT\_LINE('Total Cost for Dive Event invalid\_id: ' || v\_total\_cost);

END;

/



Question 10

Gui

I decided to make use of C# in Visual Studio design the console Gui

Console.WriteLine("----------------------------------------------\n");

Console.WriteLine( "IT Gear Dealer\n");

Console.WriteLine("----------------------------------------------\n");

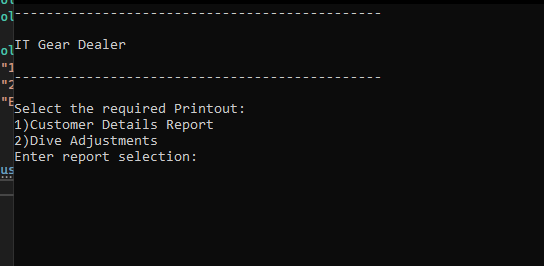
Console.WriteLine("Select the required Printout:\n" +

"1)Customer Details Report\n" +

"2)Dive Adjustments\n" +

"Enter report selection:\n");

int userChoice = int.Parse( Console.ReadLine());



Question 11

