NAME: TSHEPANG JIMMY MOTSHEGOE

STUDENT NUMBER: ST10193697

GROUP 2: DIS3  
  
ADVANCED DATABASES (ADDB6311)

Github link: https://github.com/ParadiseFx/ADDB6311-PRACTICUM.git

--Question 1

-- Creating Instructor Table

CREATE TABLE Instructor (

INS\_ID INT PRIMARY KEY,

INS\_FNAME VARCHAR2(100) NOT NULL,

INS\_SNAME VARCHAR2(100) NOT NULL,

INS\_CONTACT VARCHAR2(15) NOT NULL,

INS\_LEVEL INT NOT NULL

);

-- Creating Customer Table

CREATE TABLE Customer (

CUST\_ID VARCHAR2(10) PRIMARY KEY,

CUST\_FNAME VARCHAR2(100) NOT NULL,

CUST\_SNAME VARCHAR2(100) NOT NULL,

CUST\_ADDRESS VARCHAR2(50) NOT NULL,

CUST\_CONTACT VARCHAR2(15) NOT NULL

);

-- Creating Dive Table

CREATE TABLE Dive (

DIVE\_ID INT PRIMARY KEY,

DIVE\_NAME VARCHAR2(100) NOT NULL,

DIVE\_DURATION VARCHAR2(50) NOT NULL,

DIVE\_LOCATION VARCHAR2(100) NOT NULL,

DIVE\_EXP\_LEVEL VARCHAR2(5) NOT NULL,

DIVE\_COST NUMBER(10,2) NOT NULL

);

-- Creating Dive\_Event Table

CREATE TABLE Dive\_Event (

DIVE\_EVENT\_ID VARCHAR2(10) PRIMARY KEY,

DIVE\_DATE DATE NOT NULL,

DIVE\_PARTICIPANTS INT NOT NULL,

INS\_ID INT NOT NULL,

CUST\_ID VARCHAR2(10) NOT NULL,

DIVE\_ID INT NOT NULL,

FOREIGN KEY (INS\_ID) REFERENCES Instructor(INS\_ID),

FOREIGN KEY (CUST\_ID) REFERENCES Customer(CUST\_ID),

FOREIGN KEY (DIVE\_ID) REFERENCES Dive(DIVE\_ID)

);

-- Inserting data into Instructor Table

INSERT INTO Instructor (INS\_ID, INS\_FNAME, INS\_SNAME, INS\_CONTACT, INS\_LEVEL) VALUES

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

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

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

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

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

-- Inserting data into Customer Table

INSERT INTO Customer (CUST\_ID, CUST\_FNAME, CUST\_SNAME, CUST\_ADDRESS, CUST\_CONTACT) VALUES

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

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

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

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

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

('C120', 'Thando', 'Zolani', '88 Summer Road', '0847541254'),

('C121', 'Philip', 'Jackson', '3 Long Road', '0745556658'),

('C122', 'Sarah', 'Jones', '7 Sea Road', '0814745745'),

('C123', 'Catherine', 'Howard', '31 Lake Side Road', '0822232521');

-- Inserting data into Dive Table

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

(550, 'Shark Dive', '3 hours', 'Shark Point', 'A', 500),

(551, 'Coral Dive', '1 hour', 'Break Point', '300', 800),

(552, 'Wave Crescent', '2 hours', 'Ship wreck ally', '250', 553),

(553, 'Underwater Exploration', '1 hour', 'Coral ally', '2', 750),

(554, 'Underwater Adventure', '3 hours', 'Sandy Beach', '1', 555),

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

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

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

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

-- Inserting data into Dive\_Event Table

INSERT INTO Dive\_Event (DIVE\_EVENT\_ID, DIVE\_DATE, DIVE\_PARTICIPANTS, INS\_ID, CUST\_ID, DIVE\_ID) VALUES

('de101', TO\_DATE('15/JUL/17', 'DD/MON/RR'), 5, 103, 'C115', 558),

('de102', TO\_DATE('16/JUL/17', 'DD/MON/RR'), 7, 102, 'C117', 555),

('de103', TO\_DATE('18/JUL/17', 'DD/MON/RR'), 8, 104, 'C118', 552),

('de104', TO\_DATE('19/JUL/17', 'DD/MON/RR'), 3, 101, 'C119', 551),

('de105', TO\_DATE('21/JUL/17', 'DD/MON/RR'), 5, 104, 'C121', 558),

('de106', TO\_DATE('22/JUL/17', 'DD/MON/RR'), 8, 105, 'C120', 556),

('de107', TO\_DATE('25/JUL/17', 'DD/MON/RR'), 10, 105, 'C115', 554),

('de108', TO\_DATE('27/JUL/17', 'DD/MON/RR'), 5, 101, 'C122', 552),

('de109', TO\_DATE('28/JUL/17', 'DD/MON/RR'), 3, 102, 'C123', 553);

--Q1.2

A screenshot of a computer

Description automatically generated

--Question 2

-- Creating the administrator role

CREATE ROLE admin\_role;

-- Creating the general user role

CREATE ROLE general\_user\_role;

-- Granting privileges to the admin\_role

-- Admin can create, alter, and drop tables and other database objects

GRANT CREATE ANY TABLE TO admin\_role;

GRANT DROP ANY TABLE TO admin\_role;

GRANT ALTER ANY TABLE TO admin\_role;

GRANT SELECT ANY TABLE TO admin\_role;

GRANT INSERT ANY TABLE TO admin\_role;

GRANT UPDATE ANY TABLE TO admin\_role;

GRANT DELETE ANY TABLE TO admin\_role;

-- Admin can manage users and roles

GRANT CREATE USER TO admin\_role;

GRANT DROP USER TO admin\_role;

GRANT ALTER USER TO admin\_role;

GRANT GRANT ANY ROLE TO admin\_role;

-- Granting privileges to the general\_user\_role

-- General users can only select, insert, update, and delete data from the tables

GRANT SELECT ON Instructor TO general\_user\_role;

GRANT SELECT ON Customer TO general\_user\_role;

GRANT SELECT ON Dive TO general\_user\_role;

GRANT SELECT ON Dive\_Event TO general\_user\_role;

GRANT INSERT ON Dive\_Event TO general\_user\_role;

GRANT UPDATE ON Dive\_Event TO general\_user\_role;

GRANT DELETE ON Dive\_Event TO general\_user\_role;

-- Creating users and assign roles

CREATE USER admin\_user IDENTIFIED BY password;

CREATE USER general\_user IDENTIFIED BY password;

-- Assigning roles to users

GRANT admin\_role TO admin\_user;

GRANT general\_user\_role TO general\_user;

--Question 3

SELECT

i.INS\_FNAME || ', ' || i.INS\_SNAME AS INSTRUCTOR,

c.CUST\_FNAME || ', ' || c.CUST\_SNAME AS CUSTOMER,

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;

BEGIN

-- Declaring a cursor to fetch the necessary information

FOR rec IN (

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

)

LOOP

-- Output the results in the specified format

DBMS\_OUTPUT.PUT\_LINE('DIVE NAME: ' || rec.DIVE\_NAME);

DBMS\_OUTPUT.PUT\_LINE('DIVE DATE: ' || TO\_CHAR(rec.DIVE\_DATE, 'DD/MON/YYYY'));

DBMS\_OUTPUT.PUT\_LINE('PARTICIPANTS: ' || rec.DIVE\_PARTICIPANTS);

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

END LOOP;

END;

/

--Question 5

SET SERVEROUTPUT ON;

BEGIN

-- Declaring a cursor to fetch the necessary information

CURSOR dive\_cursor IS

SELECT

c.CUST\_FNAME || ', ' || c.CUST\_SNAME AS CUSTOMER,

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;

-- Variables to hold cursor attributes

v\_customer VARCHAR2(100);

v\_dive\_name VARCHAR2(100);

v\_participants NUMBER;

v\_instructors\_needed NUMBER;

BEGIN

OPEN dive\_cursor;

LOOP

FETCH dive\_cursor INTO v\_customer, v\_dive\_name, v\_participants;

EXIT WHEN dive\_cursor%NOTFOUND;

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

IF v\_participants <= 4 THEN

v\_instructors\_needed := 1;

ELSIF v\_participants BETWEEN 5 AND 7 THEN

v\_instructors\_needed := 2;

ELSE

v\_instructors\_needed := 3;

END IF;

-- Output of the results in the specified format

DBMS\_OUTPUT.PUT\_LINE('CUSTOMER: ' || v\_customer);

DBMS\_OUTPUT.PUT\_LINE('DIVE NAME: ' || v\_dive\_name);

DBMS\_OUTPUT.PUT\_LINE('PARTICIPANTS: ' || v\_participants);

DBMS\_OUTPUT.PUT\_LINE('STATUS: ' || v\_instructors\_needed || ' instructors required.');

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

END LOOP;

CLOSE dive\_cursor;

END;

/

--Question 6

CREATE OR REPLACE VIEW Vw\_Dive\_Event AS

SELECT

de.INS\_ID,

de.CUST\_ID,

c.CUST\_ADDRESS,

d.DIVE\_DURATION,

de.DIVE\_DATE

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('19-JUL-2017', 'DD-MON-YYYY');

​

SELECT \* FROM Vw\_Dive\_Event;

--Question 7

CREATE OR REPLACE TRIGGER New\_Dive\_Event

BEFORE INSERT OR UPDATE ON Dive\_Event

FOR EACH ROW

BEGIN

IF :NEW.DIVE\_PARTICIPANTS <= 0 OR :NEW.DIVE\_PARTICIPANTS > 20 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Number of participants must be between 1 and 20.');

END IF;

END;

/

--Question 8

​

-- Enable server output

SET SERVEROUTPUT ON;

-- Creating the procedure

CREATE OR REPLACE PROCEDURE sp\_Customer\_Details (

p\_cust\_id IN VARCHAR2,

p\_dive\_date IN DATE

) AS

v\_cust\_name VARCHAR2(100);

v\_dive\_name VARCHAR2(100);

BEGIN

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;

DBMS\_OUTPUT.PUT\_LINE('CUSTOMER DETAILS: ' || v\_cust\_name || ' booked for the ' || v\_dive\_name || ' on ' || TO\_CHAR(p\_dive\_date, 'DD/MON/YYYY'));

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No booking found for customer ID ' || p\_cust\_id || ' on ' || TO\_CHAR(p\_dive\_date, 'DD/MON/YYYY'));

END;

/

-- Testing the procedure with an existing customer and date

BEGIN

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

END;

/

-- Testing the procedure with a non-existing customer or date

BEGIN

sp\_Customer\_Details('C999', TO\_DATE('15-JUL-2017', 'DD-MON-YYYY'));

END;

/

--Question 9

-- Enable server output

SET SERVEROUTPUT ON;

-- Creating the function

CREATE OR REPLACE FUNCTION fn\_Calculate\_Discount (

p\_cust\_id IN VARCHAR2

) RETURN NUMBER IS

v\_event\_count NUMBER;

v\_discount NUMBER;

BEGIN

-- Query to count the number of dive events a customer has participated in

SELECT COUNT(\*)

INTO v\_event\_count

FROM Dive\_Event

WHERE CUST\_ID = p\_cust\_id;

-- Determining discount based on the number of events

IF v\_event\_count > 15 THEN

v\_discount := 0.30; -- 30% discount

ELSIF v\_event\_count > 10 THEN

v\_discount := 0.20; -- 20% discount

ELSIF v\_event\_count > 5 THEN

v\_discount := 0.10; -- 10% discount

ELSE

v\_discount := 0.00; -- No discount

END IF;

RETURN v\_discount;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No dive events found for customer ID ' || p\_cust\_id);

RETURN 0.00; -- No discount if no data found

WHEN OTHERS THEN

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

RETURN 0.00; -- No discount in case of an error

END;

/

-- Testing the function with an existing customer

DECLARE

v\_discount NUMBER;

BEGIN

v\_discount := fn\_Calculate\_Discount('C115');

DBMS\_OUTPUT.PUT\_LINE('Discount for customer C115: ' || (v\_discount \* 100) || '%');

END;

/

-- Testing the function with a non-existing customer

DECLARE

v\_discount NUMBER;

BEGIN

v\_discount := fn\_Calculate\_Discount('C999');

DBMS\_OUTPUT.PUT\_LINE('Discount for customer C999: ' || (v\_discount \* 100) || '%');

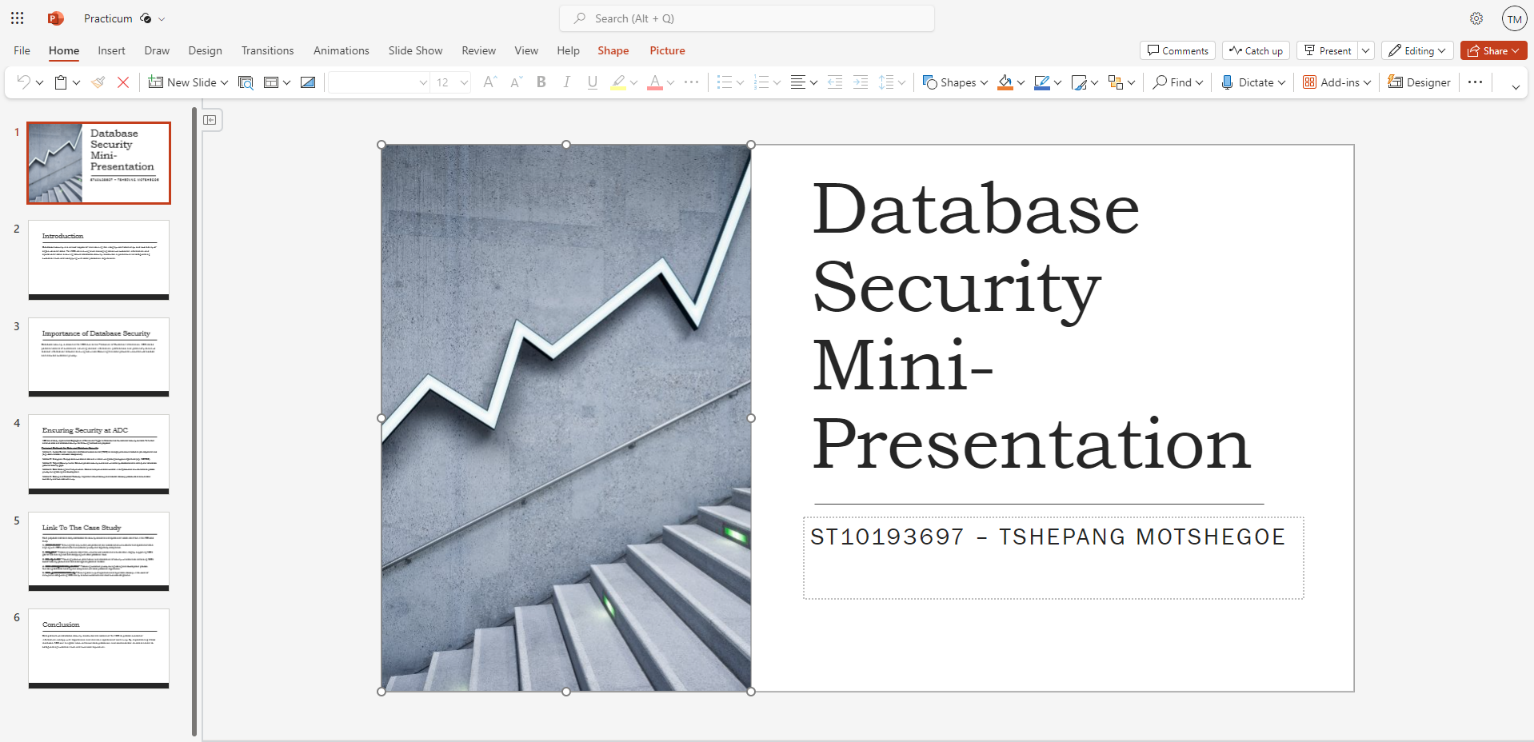
END;

/

--Question 10

--Question 11

(Powerpoint Presentation Screenshots)



A screenshot of a computer

Description automatically generated

A close-up of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A black and white text on a white background

Description automatically generated