**ADVANCED DATABASE EXAM**

**DIS GROUP1**

**STUDENT NUMBR10107698**

**KUTLWANO LUCKY NTHUTANG**

2024

**Table of Contents**

[**QUESTION 1 2**](#_Toc170128943)

[**QUESTION 2 3**](#_Toc170128944)

[**QUESTION 3 4**](#_Toc170128945)

[**QUESTION 4 5**](#_Toc170128946)

[**QUESTION 5 6**](#_Toc170128947)

[**QUESTION 6 7**](#_Toc170128948)

[**QUESTION 7 8**](#_Toc170128949)

[**QUESTION8 9**](#_Toc170128950)

[**QUESTION 9 10**](#_Toc170128951)

[**QUESTION 10 11**](#_Toc170128952)

## **QUESTION 1**

A black and white diagram

Description automatically generated

## **QUESTION 2**

-- Creating Roles

CREATE ROLE admin\_role;

CREATE ROLE general\_role;

-- Granting Privileges to the Admin Role

-- Admins can create, alter, and drop objects in the database, and manage users

GRANT CREATE SESSION TO admin\_role;

GRANT CREATE TABLE TO admin\_role;

GRANT CREATE VIEW TO admin\_role;

GRANT CREATE PROCEDURE TO admin\_role;

GRANT CREATE TRIGGER TO admin\_role;

GRANT ALTER ANY TABLE TO admin\_role;

GRANT DROP ANY TABLE TO admin\_role;

GRANT CREATE USER TO admin\_role;

GRANT DROP USER TO admin\_role;

GRANT GRANT ANY PRIVILEGE TO admin\_role;

-- Granting Privileges to the General Role

-- General users can only select data from the tables and views

GRANT CREATE SESSION TO general\_role;

GRANT SELECT ON CUSTOMER TO general\_role;

GRANT SELECT ON INSTRUCTOR TO general\_role;

GRANT SELECT ON DIVE TO general\_role;

GRANT SELECT ON DIVE\_EVENT TO general\_role;

-- Creating Users and Assigning Roles

-- Create an admin user

CREATE USER admin\_user IDENTIFIED BY admin\_password;

GRANT admin\_role TO admin\_user;

-- Create a general user

CREATE USER general\_user IDENTIFIED BY general\_password;

GRANT general\_role TO general\_user;

## **QUESTION 3**

SELECT

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

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

d.DIVE\_LOCATION,

de.DIVE\_PARTICIPANTS

FROM

DIVE\_EVENT de

JOIN

INSTRUCTOR i

ON de.INS\_ID = i.INS\_ID

JOIN

CUSTOMERRR 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;

***OUTPUT:***

***A screenshot of a computer

Description automatically generated***

## **QUESTION 4**

SET SERVEROUTPUT ON;

DECLARE

v\_DIVE\_NAME VARCHAR2(100);

v\_DIVE\_DATE DATE;

v\_DIVE\_PARTICIPANTS NUMBER;

BEGIN

FOR rec IN (SELECT d.dive\_name, de.dive\_date, de.dive\_participants

FROM DIVE d

JOIN DIVE\_EVENT de

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

WHERE de.dive\_participants >= 10)

LOOP

v\_DIVE\_NAME := rec.dive\_name;

v\_DIVE\_DATE := rec.dive\_date;

v\_DIVE\_PARTICIPANTS := rec.dive\_participants;

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

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

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

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

END LOOP;

END;

/

***OUTPUT:***

A screenshot of a computer

Description automatically generated

## **QUESTION 5**

SET SERVEROUTPUT ON;

DECLARE

-- Declare cursor for the required data

CURSOR dive\_cursor IS

SELECT

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

d.DIVE\_NAME,

de.DIVE\_PARTICIPANTS,

d.DIVE\_COST

FROM

CUSTOMERRR c

JOIN DIVE\_EVENT de

ON c.CUST\_ID = de.CUST\_ID

JOIN DIVE d

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

WHERE

d.DIVE\_COST > 500;

-- Variables to hold the fetched data

v\_customer\_name VARCHAR2(100);

v\_dive\_name VARCHAR2(100);

v\_dive\_participants NUMBER;

v\_cost NUMBER;

v\_instructors\_needed NUMBER;

BEGIN

OPEN dive\_cursor;

-- Check if the cursor has any rows

IF dive\_cursor%ISOPEN THEN

-- Fetch each row and process it

LOOP

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

EXIT WHEN dive\_cursor%NOTFOUND;

-- Determine the number of instructors needed based on dive participants

IF v\_dive\_participants <= 4 THEN

v\_instructors\_needed := 1;

ELSIF v\_dive\_participants BETWEEN 5 AND 7 THEN

v\_instructors\_needed := 2;

ELSE

v\_instructors\_needed := 3;

END IF;

-- Output the results

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

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

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

DBMS\_OUTPUT.PUT\_LINE('INSTRUCTORS REQUIRED: ' || v\_instructors\_needed ||' INSTRUCTORS REQUIRED.');

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

END LOOP;

END IF;

-- Close the cursor

CLOSE dive\_cursor;

END;

/

***OUTPUT:***

A screenshot of a computer program

Description automatically generated

## **QUESTION 6**

CREATE VIEW Vw\_Dive\_Event AS

SELECT

de.INS\_ID,

de.CUST\_ID,

c.CUST\_ADDRESS,

d.DIVE\_DURATION

FROM

DIVE\_EVENT de

JOIN CUSROMERRR 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;

/

A computer screen shot of a message

Description automatically generated

-- Test case: 0 participants (should fail)

BEGIN

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

VALUES ('de\_110', TO\_DATE('29-Jul-2017', 'DD-Mon-YYYY'), 0, 103, 'C115', 554);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/

A screenshot of a computer error message

Description automatically generated

-- Test case: 21 participants (should fail)

BEGIN

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

VALUES ('de\_111', TO\_DATE('30-Jul-2017', 'DD-Mon-YYYY'), 21, 104, 'C116', 555);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/

A screenshot of a computer

Description automatically generated

-- Test case: 10 participants (should succeed)

BEGIN

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

VALUES ('de\_112', TO\_DATE('31-Jul-2017', 'DD-Mon-YYYY'), 10, 105, 'C117', 556);

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

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

END;

/

A screenshot of a computer

Description automatically generated

**QUESTION8**

CREATE OR REPLACE PROCEDURE sp\_Customer\_Details (

p\_cust\_id IN CUSTOMERRR.CUST\_ID%TYPE,

p\_dive\_date IN DATE

)

IS

v\_cust\_fname CUSTOMERRR.CUST\_FNAME%TYPE;

v\_cust\_sname CUSTOMERRR.CUST\_SNAME%TYPE;

v\_dive\_name DIVE.DIVE\_NAME%TYPE;

BEGIN

-- Retrieve customer details

SELECT CUST\_FNAME, CUST\_SNAME

INTO v\_cust\_fname, v\_cust\_sname

FROM CUSTOMERRR

WHERE CUST\_ID = p\_cust\_id;

-- Retrieve dive name booked for the given date

SELECT d.DIVE\_NAME

INTO v\_dive\_name

FROM DIVE\_EVENT de

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 results

DBMS\_OUTPUT.PUT\_LINE('Customer ID: ' || p\_cust\_id);

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

DBMS\_OUTPUT.PUT\_LINE('Dive Name booked on ' || TO\_CHAR(p\_dive\_date, 'DD-Mon-YYYY') || ': ' || v\_dive\_name);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No data found for customer ' || p\_cust\_id || ' on ' || TO\_CHAR(p\_dive\_date, 'DD-Mon-YYYY'));

WHEN OTHERS THEN

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

END sp\_Customer\_Details;

/

--

BEGIN

sp\_Customer\_Details('C115', TO\_DATE('25-Jul-2017', 'DD-Mon-YYYY'));

END;

***OUTPUT:***

A screenshot of a computer error

Description automatically generated

BEGIN

sp\_Customer\_Details('C115', TO\_DATE('25-Jul-2017', 'DD-Mon-YYYY'));

END;

A screenshot of a computer

Description automatically generated

## **QUESTION 9**

CREATE OR REPLACE FUNCTION calculate\_total\_participants (

p\_ins\_id INSTRUCTOR.INS\_ID%TYPE,

p\_start\_date IN DATE,

p\_end\_date IN DATE

)

RETURN NUMBER

IS

v\_total\_participants NUMBER := 0;

BEGIN

-- Calculate total participants instructed by the instructor within the date range

SELECT SUM(de.DIVE\_PARTICIPANTS)

INTO v\_total\_participants

FROM DIVE\_EVENT de

WHERE de.INS\_ID = p\_ins\_id

AND de.DIVE\_DATE

BETWEEN p\_start\_date

AND p\_end\_date;

-- Handle case where no data is found (no events for the instructor in the date range)

IF v\_total\_participants IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'No dive events found for instructor ' || p\_ins\_id || ' between ' || TO\_CHAR(p\_start\_date, 'DD-Mon-YYYY') || ' and ' || TO\_CHAR(p\_end\_date, 'DD-Mon-YYYY'));

END IF;

RETURN v\_total\_participants;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20002, 'No dive events found for instructor ' || p\_ins\_id || ' between ' || TO\_CHAR(p\_start\_date, 'DD-Mon-YYYY') || ' and ' || TO\_CHAR(p\_end\_date, 'DD-Mon-YYYY'));

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Error: ' || SQLCODE || ' - ' || SQLERRM);

END calculate\_total\_participants;

/

***OUTPUT:***

A screenshot of a computer error

Description automatically generated

-- Example execution of the function

DECLARE

v\_total\_participants NUMBER;

BEGIN

-- Call the function with sample input parameters

v\_total\_participants := calculate\_total\_participants(103, TO\_DATE('01-Jul-2017', 'DD-Mon-YYYY'), TO\_DATE('31-Jul-2017', 'DD-Mon-YYYY'));

-- Display the result

DBMS\_OUTPUT.PUT\_LINE('Total participants instructed: ' || v\_total\_participants);

EXCEPTION

WHEN OTHERS THEN

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

END;

/

A screenshot of a computer

Description automatically generated

## **QUESTION 10**

import tkinter as tk

from tkinter import ttk

class GearDealerGUI:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("IT Gear Dealer Management GUI")

# Create tabs for different reports

self.tabControl = ttk.Notebook(root)

# Tab for sp\_customer\_details

self.tab1 = ttk.Frame(self.tabControl)

self.tabControl.add(self.tab1, text='Customer Details')

# Tab for fn\_dive\_adjustments

self.tab2 = ttk.Frame(self.tabControl)

self.tabControl.add(self.tab2, text='Dive Adjustments')

self.tabControl.pack(expand=1, fill="both")

# Widgets for sp\_customer\_details tab

tk.Label(self.tab1, text="Customer ID:").grid(row=0, column=0, padx=10, pady=5)

self.cust\_id\_entry = tk.Entry(self.tab1)

self.cust\_id\_entry.grid(row=0, column=1, padx=10, pady=5)

tk.Label(self.tab1, text="Dive Date (DD-Mon-YYYY):").grid(row=1, column=0, padx=10, pady=5)

self.dive\_date\_entry = tk.Entry(self.tab1)

self.dive\_date\_entry.grid(row=1, column=1, padx=10, pady=5)

self.customer\_details\_button = tk.Button(self.tab1, text="Get Customer Details", command=self.get\_customer\_details)

self.customer\_details\_button.grid(row=2, column=0, columnspan=2, pady=10)

self.customer\_details\_output = tk.Text(self.tab1, height=10, width=50)

self.customer\_details\_output.grid(row=3, column=0, columnspan=2, padx=10, pady=10)

# Widgets for fn\_dive\_adjustments tab

tk.Label(self.tab2, text="Instructor ID:").grid(row=0, column=0, padx=10, pady=5)

self.ins\_id\_entry = tk.Entry(self.tab2)

self.ins\_id\_entry.grid(row=0, column=1, padx=10, pady=5)

tk.Label(self.tab2, text="Start Date (DD-Mon-YYYY):").grid(row=1, column=0, padx=10, pady=5)

self.start\_date\_entry = tk.Entry(self.tab2)

self.start\_date\_entry.grid(row=1, column=1, padx=10, pady=5)

tk.Label(self.tab2, text="End Date (DD-Mon-YYYY):").grid(row=2, column=0, padx=10, pady=5)

self.end\_date\_entry = tk.Entry(self.tab2)

self.end\_date\_entry.grid(row=2, column=1, padx=10, pady=5)

self.dive\_adjustments\_button = tk.Button(self.tab2, text="Get Dive Adjustments", command=self.get\_dive\_adjustments)

self.dive\_adjustments\_button.grid(row=3, column=0, columnspan=2, pady=10)

self.dive\_adjustments\_output = tk.Text(self.tab2, height=10, width=50)

self.dive\_adjustments\_output.grid(row=4, column=0, columnspan=2, padx=10, pady=10)

def get\_customer\_details(self):

cust\_id = self.cust\_id\_entry.get()

dive\_date = self.dive\_date\_entry.get()

# Placeholder for actual function call to database and result processing

result = f"Customer Details for {cust\_id} on {dive\_date}:\n\n"

# Simulated result for display

result += "Customer Name: Heinrich Willis\n"

result += "Dive booked: Current Adventure\n"

self.customer\_details\_output.delete(1.0, tk.END) # Clear previous output

self.customer\_details\_output.insert(tk.END, result)

def get\_dive\_adjustments(self):

ins\_id = self.ins\_id\_entry.get()

start\_date = self.start\_date\_entry.get()

end\_date = self.end\_date\_entry.get()

# Placeholder for actual function call to database and result processing

result = f"Dive Adjustments for Instructor {ins\_id} between {start\_date} and {end\_date}:\n\n"

# Simulated result for display

result += "Total dive adjustments: 150\n"

self.dive\_adjustments\_output.delete(1.0, tk.END) # Clear previous output

self.dive\_adjustments\_output.insert(tk.END, result)

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = GearDealerGUI(root)

root.mainloop()