ADDB6311 PRACTICUM

ADVANCED DATABASES

Muhammed Talha Shaikh

2024

# ***QUESTION 1***

CREATE TABLE INSTRUCTOR (

    ins\_ID INT PRIMARY KEY,

    ins\_fname VARCHAR(50),

    ins\_sname VARCHAR(50),

    ins\_contact VARCHAR(20),

    ins\_level INT

);

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);

CREATE TABLE CUSTOMER (

    cust\_id VARCHAR(4) PRIMARY KEY,

    cust\_fname VARCHAR(50),

    cust\_sname VARCHAR(50),

    cust\_address VARCHAR(100),

    cust\_contact VARCHAR(20)

);

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 CUSTOMER (cust\_id, cust\_fname, cust\_sname, cust\_address, cust\_contact)

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

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

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

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

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

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

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

CREATE TABLE DIVE (

    DIVE\_ID INT PRIMARY KEY,

    DIVE\_NAME VARCHAR(50),

    DIVE\_DURATION VARCHAR(20),

    DIVE\_LOCATION VARCHAR(50),

    DIVE\_EXP\_LEVEL INT,

    DIVE\_COST INT

);

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', '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 Exploration', '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 Adventure', '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 Adventure', '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);

CREATE TABLE DIVE\_EVENT (

    dive\_event\_id VARCHAR(6) PRIMARY KEY,

    dive\_date DATE,

    dive\_participants INT,

    ins\_ID INT,

    cust\_id VARCHAR(4),

    dive\_ID INT,

    FOREIGN KEY (ins\_ID) REFERENCES INSTRUCTOR(ins\_ID),

    FOREIGN KEY (cust\_id) REFERENCES CUSTOMER(cust\_id),

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

);

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, 'C121', 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, 'C120', 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, 'C122', 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, 'C123', 553);

SELECT \* FROM DIVE\_EVENT;

# ***QUESTION 2***

-- Create a General user role

CREATE ROLE general\_user;

-- Grant privileges to the General user role

-- Example privileges: read-only access to specific tables and execute stored procedures

GRANT SELECT, EXECUTE ON specific\_table TO general\_user;

-- Create a General user and assign the role

CREATE USER general\_user1 IDENTIFIED BY password;

GRANT general\_user TO general\_user1;

-- Create the Administrator user with all privileges

CREATE USER 'admin\_user'@'localhost' IDENTIFIED BY 'password';

GRANT ALL PRIVILEGES ON \*.\* TO 'admin\_user'@'localhost' WITH GRANT OPTION;

-- Create the General user with limited privileges

CREATE USER 'general\_user'@'localhost' IDENTIFIED BY 'password';

GRANT SELECT, INSERT, UPDATE, DELETE ON database\_name.\* TO 'general\_user'@'localhost';

# ***QUESTION 3***

SELECT

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

    c.cust\_fname || ' ' || c.cust\_sname AS customer\_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

    AND i.ins\_ID = 104;

A screenshot of a computer

Description automatically generated

# ***QUESTION 4***

SET SERVEROUTPUT ON;

DECLARE

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

    v\_dive\_date DIVE\_EVENT.dive\_date%TYPE;

    v\_dive\_part DIVE\_EVENT.dive\_participants%TYPE;

BEGIN

    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

        v\_dive\_name := rec.DIVE\_NAME;

        v\_dive\_date := rec.dive\_date;

        v\_dive\_part := REC.dive\_participants;

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

        DBMS\_OUTPUT.PUT\_LINE( 'Date: ' || TO\_CHAR(v\_dive\_date, 'DD-Mon-YYYY') );

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

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

    END LOOP;

END;

A screenshot of a computer

Description automatically generated

# ***QUESTION 5***

SET SERVEROUTPUT ON;

DECLARE

    v\_customer\_name VARCHAR2(100);

    v\_dive\_event\_name VARCHAR2(100);

    v\_participants INT;

    CURSOR dive\_events\_cur IS

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

               d.dive\_name AS dive\_event\_name,

               de.dive\_participants AS participants

        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;

BEGIN

    OPEN dive\_events\_cur;

    LOOP

        FETCH dive\_events\_cur INTO v\_customer\_name, v\_dive\_event\_name, v\_participants;

        EXIT WHEN dive\_events\_cur%NOTFOUND;

        -- Print or process each row (for demonstration, printing to console)

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

        DBMS\_OUTPUT.PUT\_LINE( 'Dive Event: ' || v\_dive\_event\_namE);

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

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

    END LOOP;

    CLOSE dive\_events\_cur;

END;

A screenshot of a computer

Description automatically generated

# ***QUESTION 6***

CREATE VIEW Vw\_Dive\_Event AS

SELECT

    de.ins\_id AS INS\_ID,

    de.cust\_id AS CUST\_ID,

    c.cust\_address AS CUST\_ADDRESS,

    d.dive\_duration AS DIVE\_DURATION,

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

A screenshot of a computer

Description automatically generated

# ***QUESTION 7***

CREATE OR REPLACE TRIGGER New\_Dive\_Event

BEFORE INSERT ON dive\_event

FOR EACH ROW

BEGIN

    IF :NEW.dive\_participants <= 0 OR :NEW.dive\_participants > 20 THEN

        RAISE\_APPLICATION\_ERROR(-20001, 'Invalid number of participants. Must be between 1 and 20.');

    END IF;

END;

A screenshot of a computer

Description automatically generated

# ***QUESTION 8***

-- Create a stored procedure to retrieve customer details for a specific dive event

CREATE OR REPLACE PROCEDURE sp\_Customer\_Details (

    p\_cust\_id IN VARCHAR2,

    p\_dive\_date IN DATE,

    p\_customer\_details OUT VARCHAR2

)

IS

    v\_customer\_name VARCHAR2(100);

    v\_dive\_name VARCHAR2(100);

BEGIN

    SELECT

        c.cust\_fname || ' ' || c.cust\_sname,

        d.dive\_name

    INTO

        v\_customer\_name,

        v\_dive\_name

    FROM

        customer c

    JOIN

        dive\_event de ON c.cust\_id = de.cust\_id

    JOIN

        dive d ON de.dive\_id = d.dive\_id

    WHERE

        c.cust\_id = p\_cust\_id

        AND de.dive\_date = p\_dive\_date;

    p\_customer\_details := v\_customer\_name || ' booked for the ' || v\_dive\_name || ' on ' || TO\_CHAR(p\_dive\_date, 'DD/MON/YY');

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        p\_customer\_details := 'Customer not found for the specified dive event.';

END;

A screenshot of a computer

Description automatically generated

# ***QUESTION 9***

-- Create a function to calculate the total cost of a dive event based on dive ID

CREATE OR REPLACE FUNCTION fn\_Dive\_Cost (

    p\_dive\_id IN NUMBER

)

RETURN NUMBER

IS

    v\_dive\_cost NUMBER;

BEGIN

    SELECT dive\_cost INTO v\_dive\_cost

    FROM dive

    WHERE dive\_id = p\_dive\_id;

    RETURN v\_dive\_cost;

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        RETURN NULL; -- or handle as appropriate

END;

A screenshot of a computer error

Description automatically generated

# ***QUESTION 10***

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace AdventureDiveClubGUI

{

   public partial class MainForm : Form

   {

       public MainForm()

       {

           InitializeComponent();

       }

       // Event handler for button click to execute stored procedure

       private void btnExecute\_Click(object sender, EventArgs e)

       {

           string customerId = txtCustomerId.Text.Trim(); // Example: Get customer ID from textbox

           DateTime diveDate = dateTimePickerDiveDate.Value; // Example: Get dive date from date picker

           try

           {

               // Replace "your\_connection\_string\_here" with your actual connection string

               string connectionString = "2024databaseExam";

               using (SqlConnection connection = new SqlConnection(connectionString))

               {

                   connection.Open();

                   SqlCommand cmd = new SqlCommand("sp\_Customer\_Details", connection);

                   cmd.CommandType = CommandType.StoredProcedure;

                   // Add parameters to the stored procedure

                   cmd.Parameters.AddWithValue("@customer\_id", customerId);

                   cmd.Parameters.AddWithValue("@dive\_date", diveDate);

                   SqlDataReader reader = cmd.ExecuteReader();

                   if (reader.HasRows)

                   {

                       while (reader.Read())

                       {

                           // Example: Display customer details in a label

                           lblCustomerDetails.Text = $"Customer Details: {reader["CustomerDetails"]}";

                       }

                   }

                   else

                   {

                       MessageBox.Show("No data found.");

                   }

               }

           }

           catch (Exception ex)

           {

               MessageBox.Show("Error: " + ex.Message);

           }

       }

   }

}

# ***QUESTION 11***

Based on the case study and the implemented measures, here are five additional ways to ensure data and database security for ADC Management, along with their benefits:

### 1. Encryption of Sensitive Data

\*\*Motivation:\*\*

Encryption transforms data into an unreadable format using encryption algorithms, ensuring that even if unauthorized individuals gain access to the database, they cannot view sensitive information without the decryption key. This is crucial for protecting customer details, financial records, and any other confidential information stored by ADC Management.

\*\*Benefits:\*\*

- \*\*Data Confidentiality:\*\* Ensures that sensitive information remains confidential and protected from unauthorized access.

- \*\*Compliance:\*\* Helps meet regulatory requirements (e.g., GDPR, HIPAA) related to data protection and privacy.

- \*\*Reputation Management:\*\* Enhances customer trust by demonstrating commitment to protecting their sensitive information.

### 2. Regular Security Audits and Vulnerability Assessments

\*\*Motivation:\*\*

Regular security audits and vulnerability assessments involve systematic reviews of database security policies, configurations, and access controls. They help identify and mitigate security gaps or vulnerabilities before they can be exploited by malicious actors.

\*\*Benefits:\*\*

- \*\*Proactive Security Measures:\*\* Identifies and addresses security weaknesses before they can lead to data breaches or system compromises.

- \*\*Compliance Assurance:\*\* Demonstrates compliance with industry standards and regulations through documented audit trails.

- \*\*Risk Reduction:\*\* Minimizes the risk of financial losses, operational disruptions, and reputational damage associated with data breaches.

### 3. Implementing Database Activity Monitoring (DAM)

\*\*Motivation:\*\*

Database Activity Monitoring (DAM) involves real-time monitoring and logging of database activities, such as queries, transactions, and user access. It provides visibility into who accesses what data and under what circumstances, enabling swift detection and response to suspicious activities.

\*\*Benefits:\*\*

- \*\*Early Threat Detection:\*\* Detects unauthorized access attempts, unusual patterns of activity, or insider threats in real-time.

- \*\*Forensic Analysis:\*\* Facilitates forensic investigations by providing detailed audit logs of database activities.

- \*\*Compliance and Reporting:\*\* Supports regulatory compliance by documenting and reporting on data access and usage.

### 4. Implementing Database Firewall

\*\*Motivation:\*\*

A database firewall acts as a barrier between the database and incoming traffic, inspecting and filtering database queries based on predefined security rules. It prevents unauthorized access and SQL injection attacks that attempt to exploit vulnerabilities in application code.

\*\*Benefits:\*\*

- \*\*Granular Access Control: \*\* Filters incoming queries and commands to ensure that only authorized and valid requests are processed.

- \*\*Intrusion Prevention: \*\* Blocks malicious SQL injection attacks and unauthorized attempts to manipulate or extract data.

- \*\*Performance Optimization: \*\* Improves database performance by reducing the impact of malicious traffic and ensuring efficient query handling.

### 5. Disaster Recovery and Backup Plans

\*\*Motivation: \*\*

Disaster recovery and backup plans involve creating and regularly updating copies of critical data and database configurations. These backups are stored securely and can be used to restore data in case of accidental deletion, corruption, or a catastrophic event like a natural disaster or cyber-attack.

\*\*Benefits: \*\*

- \*\*Business Continuity: \*\* Ensures uninterrupted access to data and critical systems during and after a disaster.

- \*\*Risk Management:\*\* Mitigates the impact of data loss or system downtime on business operations.

- \*\*Regulatory Compliance: \*\* Supports compliance with data retention and recovery requirements by maintaining up-to-date backups and recovery plans.