INSY7213

Ruan Cupido

VARSITY COLLEGE | ST10450618

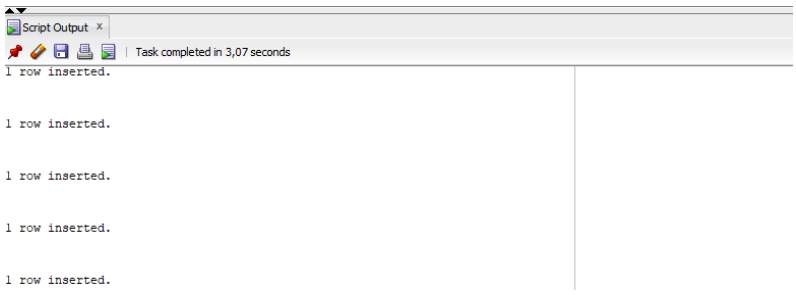
Table of Contents

Question 1

```
-----------------------------------
--QUESTION 1
-----------------------------------
-- Create the DONOR table
CREATE TABLE DONOR (
    DONOR_ID VARCHAR2(10) PRIMARY KEY,
    DONOR_NAME VARCHAR2(50) NOT NULL,
    CONTACT_NO VARCHAR2(15),
    EMAIL VARCHAR2(50)
);

-- Create the BIKE table
CREATE TABLE BIKE (
    BIKE_ID VARCHAR2(10) PRIMARY KEY,
    DESCRIPTION VARCHAR2(50) NOT NULL,
    BIKE_TYPE VARCHAR2(30) NOT NULL,
    MANUFACTURER VARCHAR2(30) NOT NULL
);

-- Create the VOLUNTEER table
CREATE TABLE VOLUNTEER (
    VOLUNTEER_ID VARCHAR2(10) PRIMARY KEY,
    VOL_NAME VARCHAR2(50) NOT NULL,
    ROLE VARCHAR2(30)
);

-- Create the DONATION table
CREATE TABLE DONATION (
    DONATION_ID NUMBER PRIMARY KEY,
    DONOR_ID VARCHAR2(10) NOT NULL,
    BIKE_ID VARCHAR2(10) NOT NULL,
    VALUE NUMBER(10,2) CHECK (VALUE > 0),
    VOLUNTEER_ID VARCHAR2(10) NOT NULL,
    DONATION_DATE DATE NOT NULL,
    CONSTRAINT fk_donor FOREIGN KEY (DONOR_ID) REFERENCES DONOR(DONOR_ID),
    CONSTRAINT fk_bike FOREIGN KEY (BIKE_ID) REFERENCES BIKE(BIKE_ID),
    CONSTRAINT fk_volunteer FOREIGN KEY (VOLUNTEER_ID) REFERENCES VOLUNTEER(VOLUNTEER_ID)
);
```


```
Table DONOR created.


Table BIKE created.


Table VOLUNTEER created.


Table DONATION created.
```

```sql
-- Insert BIKE data
INSERT INTO BIKE VALUES ('B001', 'BMX AX1', 'Road Bike', 'BMX');
INSERT INTO BIKE VALUES ('B002', 'Giant Domain 1', 'Road Bike', 'Giant');
INSERT INTO BIKE VALUES ('B003', 'Ascent 26In', 'Mountain Bike', 'Raleigh');
INSERT INTO BIKE VALUES ('B004', 'Canyon 6X', 'Kids Bike', 'Canyon');
INSERT INTO BIKE VALUES ('B005', 'Marvel', 'Gravel Road Bike', 'BMX');
INSERT INTO BIKE VALUES ('B006', 'Mountain 21 Speed', 'Mountain Bike', 'BMX');
INSERT INTO BIKE VALUES ('B007', 'Canyon Roadster', 'Road Bike', 'Canyon');
INSERT INTO BIKE VALUES ('B008', 'Legion 101', 'Hybrid Bike', 'BMX');
INSERT INTO BIKE VALUES ('B009', 'Madonna 9', 'Road Bike', 'Trek');
INSERT INTO BIKE VALUES ('B010', 'Comp 2022', 'Mountain Bike', 'Trek');
INSERT INTO BIKE VALUES ('B011', 'BMX AX15', 'Road Bike', 'BMX');

-- Insert DONOR data
INSERT INTO DONOR VALUES ('DID11', 'Jeff Wanya', '0827172250', 'wanyajeff@ymail.com');
INSERT INTO DONOR VALUES ('DID12', 'Sthembeni Pisho', '0837865670', 'sthepisho@ymail.com');
INSERT INTO DONOR VALUES ('DID13', 'James Temba', '0878978650', 'jimmy@ymail.com');
INSERT INTO DONOR VALUES ('DID14', 'Luramo Misi', '0826575650', 'luramom@ymail.com');
INSERT INTO DONOR VALUES ('DID15', 'Abraham Xolani', '0797656430', 'axolani@ymail.com');
INSERT INTO DONOR VALUES ('DID16', 'Rumi Jones', '0668899221', 'rjones@true.com');
INSERT INTO DONOR VALUES ('DID17', 'Xolani Redo', '0614553389', 'xredo@ymail.com');
INSERT INTO DONOR VALUES ('DID18', 'Tenny Stars', '0824228870', 'tenstars@true.com');
INSERT INTO DONOR VALUES ('DID19', 'Tiny Rambo', '0715554333', 'trambo@ymail.com');
INSERT INTO DONOR VALUES ('DID20', 'Yannick Leons', '0615554323', 'yleons@true.com');

-- Insert VOLUNTEER data
INSERT INTO VOLUNTEER VALUES ('vol101', 'Kenny Temba', 'Manager');
INSERT INTO VOLUNTEER VALUES ('vol102', 'Mamelodi Marks', 'Coordinator');
INSERT INTO VOLUNTEER VALUES ('vol103', 'Ada Andrews', 'Coordinator');
INSERT INTO VOLUNTEER VALUES ('vol104', 'Evans Tambala', 'Assistant');
INSERT INTO VOLUNTEER VALUES ('vol105', 'Xolani Samson', 'Supervisor');

-- Insert DONATION data
INSERT INTO DONATION VALUES (1,  'DID11', 'B001', 1500, 'vol101', TO_DATE('01-May-2023','DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (2,  'DID12', 'B002', 2500, 'vol101', TO_DATE('03-May-2023','DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (3,  'DID13', 'B003', 1000, 'vol103', TO_DATE('03-May-2023','DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (4,  'DID14', 'B004', 1750, 'vol105', TO_DATE('05-May-2023','DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (5,  'DID15', 'B006', 2000, 'vol101', TO_DATE('07-May-2023','DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (6,  'DID16', 'B007', 1800, 'vol105', TO_DATE('09-May-2023','DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (7,  'DID17', 'B008', 1500, 'vol101', TO_DATE('15-May-2023','DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (8,  'DID18', 'B009', 1500, 'vol103', TO_DATE('19-May-2023','DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (9,  'DID12', 'B010', 2500, 'vol103', TO_DATE('25-May-2023','DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (10, 'DID20', 'B005', 3500, 'vol105', TO_DATE('05-May-2023','DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (11, 'DID19', 'B011', 2500, 'vol103', TO_DATE('30-May-2023','DD-Mon-YYYY'));
```



BIKE:

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	BIKE_ID	VARCHAR2(10 BYTE)	No	(null)	1	(null)
2	DESCRIPTION	VARCHAR2(50 BYTE)	No	(null)	2	(null)
3	BIKE_TYPE	VARCHAR2(30 BYTE)	No	(null)	3	(null)
4	MANUFACTURER	VARCHAR2(30 BYTE)	No	(null)	4	(null)

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
1	SYS_C008374	Check	"DESCRIPTION" IS NOT NULL
2	SYS_C008375	Check	"BIKE_TYPE" IS NOT NULL
3	SYS_C008376	Check	"MANUFACTURER" IS NOT NULL
4	SYS_C008377	Primary_Key	(null)

Donor:

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	DONOR_ID	VARCHAR2(10 BYTE)	No	(null)	1	(null)
2	DONOR_NAME	VARCHAR2(50 BYTE)	No	(null)	2	(null)
3	CONTACT_NO	VARCHAR2(15 BYTE)	Yes	(null)	3	(null)
4	EMAIL	VARCHAR2(50 BYTE)	Yes	(null)	4	(null)

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
1	SYS_C008372	Check	"DONOR_NAME" IS NOT NULL
2	SYS_C008373	Primary_Key	(null)

Donation:

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	DONATION_ID	NUMBER	No	(null)	1	(null)
2	DONOR_ID	VARCHAR2(10 BYTE)	No	(null)	2	(null)
3	BIKE_ID	VARCHAR2(10 BYTE)	No	(null)	3	(null)
4	VALUE	NUMBER(10,2)	Yes	(null)	4	(null)
5	VOLUNTEER_ID	VARCHAR2(10 BYTE)	No	(null)	5	(null)
6	DONATION_DATE	DATE	No	(null)	6	(null)

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
1	FK_BIKE	Foreign_Key	(null)
2	FK_DONOR	Foreign_Key	(null)
3	FK_VOLUNTEER	Foreign_Key	(null)
4	SYS_C008380	Check	"DONOR_ID" IS NOT NULL
5	SYS_C008381	Check	"BIKE_ID" IS NOT NULL
6	SYS_C008382	Check	"VOLUNTEER_ID" IS NOT NULL
7	SYS_C008383	Check	"DONATION_DATE" IS NOT NULL
8	SYS_C008384	Check	VALUE > 0
9	SYS_C008385	Primary_Key	(null)

Volunteer:

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	VOLUNTEER_ID	VARCHAR2(10 BYTE)	No	(null)	1	(null)
2	VOL_NAME	VARCHAR2(50 BYTE)	No	(null)	2	(null)
3	ROLE	VARCHAR2(30 BYTE)	Yes	(null)	3	(null)

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
1	SYS_C008378	Check	"VOL_NAME" IS NOT NULL
2	SYS_C008379	Primary_Key	(null)

Question 2

```
-------------------------------------
--QUESTION 2
-------------------------------------
SELECT
    d.DONOR_ID AS "Donor ID",
    b.BIKE_TYPE AS "Bike Type",
    b.DESCRIPTION AS "Bike Description",
    'R ' || TO_CHAR(n.VALUE, '9990') AS "Bike Value"
FROM
    DONATION n
    JOIN DONOR d ON n.DONOR_ID = d.DONOR_ID
    JOIN BIKE b ON n.BIKE_ID = b.BIKE_ID
WHERE
    n.VALUE > 1500;
```

Script Output x | Query Result x

SQL | All Rows Fetched: 7 in 0,215 seconds

	Donor ID	Bike Type	Bike Description	Bike Value
1	DID12	Road Bike	Giant Domain 1	R 2500
2	DID14	Kids Bike	Canyon 6X	R 1750
3	DID20	Gravel Road Bike	Marvel	R 3500
4	DID15	Mountain Bike	Mountain 21 Speed	R 2000
5	DID16	Road Bike	Canyon Roadster	R 1800
6	DID12	Mountain Bike	Comp 2022	R 2500
7	DID19	Road Bike	BMX AX15	R 2500

Question 3

```
-------------------------------------
--QUESTION 3
-------------------------------------
SET SERVEROUTPUT ON;

DECLARE
    -- Constant declaration
    v_vat_rate CONSTANT NUMBER := 0.15;

    -- Variable declarations
    v_description  BIKE.DESCRIPTION%TYPE;
    v_manufacturer BIKE.MANUFACTURER%TYPE;
    v_type         BIKE.BIKE_TYPE%TYPE;
    v_value        DONATION.VALUE%TYPE;
    v_vat_amount   NUMBER;
    v_total        NUMBER;

    -- Cursor for Road Bikes only
    CURSOR road_bikes_cur IS
        SELECT b.DESCRIPTION, b.MANUFACTURER, b.BIKE_TYPE, n.VALUE
        FROM BIKE b
        JOIN DONATION n ON b.BIKE_ID = n.BIKE_ID
        WHERE b.BIKE_TYPE = 'Road Bike';

BEGIN
    -- Loop through cursor results
    FOR bike_rec IN road_bikes_cur LOOP
        -- Assign values
        v_description  := bike_rec.DESCRIPTION;
        v_manufacturer := bike_rec.MANUFACTURER;
        v_type         := bike_rec.BIKE_TYPE;
        v_value        := bike_rec.VALUE;

        -- Calculate VAT and total
        v_vat_amount := v_value * v_vat_rate;
        v_total      := v_value + v_vat_amount;

        -- Output results
        DBMS_OUTPUT.PUT_LINE('-------------------------------------');
        DBMS_OUTPUT.PUT_LINE('BIKE DESCRIPTION: ' || v_description);
        DBMS_OUTPUT.PUT_LINE('BIKE MANUFACTURER: ' || v_manufacturer);
        DBMS_OUTPUT.PUT_LINE('BIKE TYPE: ' || v_type);
        DBMS_OUTPUT.PUT_LINE('VALUE: R' || TO_CHAR(v_value, '9G999'));
        DBMS_OUTPUT.PUT_LINE('VAT: R' || TO_CHAR(v_vat_amount, '9G999'));
        DBMS_OUTPUT.PUT_LINE('TOTAL AMNT: R' || TO_CHAR(v_total, '9G999'));
        DBMS_OUTPUT.PUT_LINE('-------------------------------------');
    END LOOP;
END;
/
```

```
-----------------------------------
BIKE DESCRIPTION: BMX AX1
BIKE MANUFACTURER: BMX
BIKE TYPE: Road Bike
VALUE: R 1,500
VAT: R   225
TOTAL AMNT: R 1,725
-----------------------------------
-----------------------------------
BIKE DESCRIPTION: Giant Domain 1
BIKE MANUFACTURER: Giant
BIKE TYPE: Road Bike
VALUE: R 2,500
VAT: R   375
TOTAL AMNT: R 2,875
-----------------------------------
-----------------------------------
BIKE DESCRIPTION: Canyon Roadster
BIKE MANUFACTURER: Canyon
BIKE TYPE: Road Bike
VALUE: R 1,800
VAT: R   270
TOTAL AMNT: R 2,070
-----------------------------------
-----------------------------------
BIKE DESCRIPTION: Madonna 9
BIKE MANUFACTURER: Trek
BIKE TYPE: Road Bike
VALUE: R 1,500
VAT: R   225
TOTAL AMNT: R 1,725
-----------------------------------
-----------------------------------
BIKE DESCRIPTION: BMX AX15
BIKE MANUFACTURER: BMX
BIKE TYPE: Road Bike
VALUE: R 2,500
VAT: R   375
TOTAL AMNT: R 2,875
-----------------------------------


PL/SQL procedure successfully completed.
```

Question 4

```
-----------------------------------
--QUESTION 4
-----------------------------------
CREATE OR REPLACE VIEW vwBikeRUs AS
SELECT
    d.DONOR_NAME AS DONOR_NAME,
    d.CONTACT_NO AS CONTACT_NO,
    b.BIKE_TYPE AS BIKE_TYPE,
    TO_CHAR(n.DONATION_DATE, 'DD/Mon/YY') AS DONATION_DATE
FROM
    DONATION n
    JOIN DONOR d ON n.DONOR_ID = d.DONOR_ID
    JOIN BIKE b ON n.BIKE_ID = b.BIKE_ID
WHERE
    n.VOLUNTEER_ID = 'vol105';

SELECT * FROM vwBikeRUs;

-- ============================================================
-- Justification for using a VIEW
-- ============================================================
/*
Justification - Benefits of Using a View:
1. **Data Security:**
   Views allow limited access to specific columns and rows (e.g., only donors assisted by vol105)
   without giving users full access to the underlying tables. This helps protect sensitive donor information.

2. **Simplified Querying:**
   The view combines data from multiple tables (DONOR, BIKE, DONATION) into one logical table,
   making it easier for users to retrieve frequently needed information with a simple SELECT statement.

These benefits improve data management, consistency, and security for BikeRUs staff.
*/
```

```
-----------------------------------
--QUESTION 4
-----------------------------------
CREATE OR REPLACE VIEW vwBikeRUs AS
SELECT
    d.DONOR_NAME AS DONOR_NAME,
    d.CONTACT_NO AS CONTACT_NO,
    b.BIKE_TYPE AS BIKE_TYPE,
    TO_CHAR(n.DONATION_DATE, 'DD/Mon/YY') AS DONATION_DATE
FROM
    DONATION n
    JOIN DONOR d ON n.DONOR_ID = d.DONOR_ID
    JOIN BIKE b ON n.BIKE_ID = b.BIKE_ID
WHERE
    n.VOLUNTEER_ID = 'vol105';

SELECT * FROM vwBikeRUs;
```

ript Output | Query Result

SQL | All Rows Fetched: 3 in 0,582 seconds

	DONOR_NAME	CONTACT_NO	BIKE_TYPE	DONATION_DATE
1	Luramo Misi	0826575650	Kids Bike	05/May/23
2	Yannick Leons	0615554323	Gravel Road Bike	05/May/23
3	Rumi Jones	0668899221	Road Bike	09/May/23

Question 5

```
-------------------------------------
--QUESTION 5
-------------------------------------
--Create the procedure
CREATE OR REPLACE PROCEDURE spDonorDetails(p_bike_id IN BIKE.BIKE_ID%TYPE)
IS
    -- Variable declarations
    v_donor_name     DONOR.DONOR_NAME%TYPE;
    v_contact_no     DONOR.CONTACT_NO%TYPE;
    v_volunteer_name VOLUNTEER.VOL_NAME%TYPE;
    v_bike_desc      BIKE.DESCRIPTION%TYPE;
    v_donation_date  DONATION.DONATION_DATE%TYPE;

-- Exception for missing data
BEGIN
    -- Select donor, volunteer, and bike details based on input bike ID
    SELECT
        d.DONOR_NAME,
        d.CONTACT_NO,
        v.VOL_NAME,
        b.DESCRIPTION,
        n.DONATION_DATE
    INTO
        v_donor_name,
        v_contact_no,
        v_volunteer_name,
        v_bike_desc,
        v_donation_date
    FROM
        DONATION n
        JOIN DONOR d ON n.DONOR_ID = d.DONOR_ID
        JOIN VOLUNTEER v ON n.VOLUNTEER_ID = v.VOLUNTEER_ID
        JOIN BIKE b ON n.BIKE_ID = b.BIKE_ID
    WHERE
        n.BIKE_ID = p_bike_id;

    -- Display formatted output
    DBMS_OUTPUT.PUT_LINE('ATTENTION: ' || v_donor_name ||
                         ' assisted by: ' || v_volunteer_name ||
                         ', donated the ' || v_bike_desc ||
                         ' on the ' || TO_CHAR(v_donation_date, 'DD/Mon/YY'));

EXCEPTION
    -- Handle case where no record matches the input Bike ID
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No donor record found for Bike ID: ' || p_bike_id);

    -- Handle any other unexpected errors
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
```

```
END spDonorDetails;
/

-- Enable output display
SET SERVEROUTPUT ON;

-- Execute the procedure for Bike ID 'B004'
BEGIN
    spDonorDetails('B004');
END;
/
```

```
-------------------------------------
--QUESTION 5
-------------------------------------
--Create the procedure
CREATE OR REPLACE PROCEDURE spDonorDetails(p_bike_id IN BIKE.BIKE_ID%TYPE)
IS
    -- Variable declarations
    v_donor_name     DONOR.DONOR_NAME%TYPE;
    v_contact_no     DONOR.CONTACT_NO%TYPE;
    v_volunteer_name VOLUNTEER.VOL_NAME%TYPE;
    v_bike_desc      BIKE.DESCRIPTION%TYPE;
    v_donation_date  DONATION.DONATION_DATE%TYPE;

-- Exception for missing data
BEGIN
    -- Select donor, volunteer, and bike details based on input bike ID
    SELECT
        d.DONOR_NAME,
```

Script Output

Task completed in 1,693 seconds

```
Procedure SPDONORDETAILS compiled

ATTENTION: Luramo Misi assisted by: Xolani Samson, donated the Canyon 6X on the 05/May/23


PL/SQL procedure successfully completed.
```

Question 6

```
-------------------------------------
--QUESTION 6
-------------------------------------
SET SERVEROUTPUT ON;

CREATE OR REPLACE FUNCTION fn_TotalDonationValue(p_donor_id IN DONOR.DONOR_ID%TYPE)
RETURN NUMBER
IS
    v_total_value  NUMBER := 0;  -- Variable to store calculated total
BEGIN
    -- Calculate the total value of donations for the given donor
    SELECT SUM(VALUE)
    INTO v_total_value
    FROM DONATION
    WHERE DONOR_ID = p_donor_id;

    -- Handle case where donor has no donations
    IF v_total_value IS NULL THEN
        v_total_value := 0;
    END IF;

    RETURN v_total_value;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        -- No record found for given donor
        DBMS_OUTPUT.PUT_LINE('No donations found for Donor ID: ' || p_donor_id);
        RETURN 0;

    WHEN OTHERS THEN
        -- Generic error handling
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
        RETURN 0;
END;
/

DECLARE
    v_donor_id      DONOR.DONOR_ID%TYPE := 'D002';  -- Example donor ID
    v_total_value   NUMBER;
    v_donor_name    DONOR.DONOR_NAME%TYPE;
BEGIN
    -- Retrieve donor name for display
    SELECT DONOR_NAME INTO v_donor_name
    FROM DONOR
    WHERE DONOR_ID = v_donor_id;

    -- Call the function to get total donation value
    v_total_value := fn_TotalDonationValue(v_donor_id);

    -- Display results in formatted message
    DBMS_OUTPUT.PUT_LINE('-----------------------------------------------');
    DBMS_OUTPUT.PUT_LINE('Donor Name: ' || v_donor_name);
    DBMS_OUTPUT.PUT_LINE('Total Value of Donations: R' || TO_CHAR(v_total_value, '9G999'));
    DBMS_OUTPUT.PUT_LINE('-----------------------------------------------');

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: Donor not found in the database.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Unexpected error occurred: ' || SQLERRM);
END;
/
```

```
------------------------------------------------
Donor Name: Sthembeni Pisho
Total Value of Donations: R 5,000
------------------------------------------------


PL/SQL procedure successfully completed.
```

Question 7

```
--QUESTION 7
-------------------------------------
SET SERVEROUTPUT ON;

DECLARE
    -- Variable declarations
    v_bike_id       BIKE.BIKE_ID%TYPE;
    v_bike_type     BIKE.BIKE_TYPE%TYPE;
    v_manufacturer  BIKE.MANUFACTURER%TYPE;
    v_value         DONATION.VALUE%TYPE;
    v_status        VARCHAR2(10);

    -- Cursor to get all bikes and their donation values
    CURSOR bike_cur IS
        SELECT
            b.BIKE_ID,
            b.BIKE_TYPE,
            b.MANUFACTURER,
            n.VALUE
        FROM
            BIKE b
            JOIN DONATION n ON b.BIKE_ID = n.BIKE_ID;

BEGIN
    -- Loop through all bikes
    FOR bike_rec IN bike_cur LOOP
        v_bike_id := bike_rec.BIKE_ID;
        v_bike_type := bike_rec.BIKE_TYPE;
        v_manufacturer := bike_rec.MANUFACTURER;
        v_value := bike_rec.VALUE;

        -- IF...ELSIF...ELSE to determine status
        IF v_value <= 1500 THEN
            v_status := '*'; -- 1-star
        ELSIF v_value > 1500 AND v_value <= 3000 THEN
            v_status := '**'; -- 2-star
        ELSE
            v_status := '***'; -- 3-star
        END IF;

        -- Display output in required format
        DBMS_OUTPUT.PUT_LINE('--------------------------------');
        DBMS_OUTPUT.PUT_LINE('BIKE ID: ' || v_bike_id);
        DBMS_OUTPUT.PUT_LINE('BIKE TYPE: ' || v_bike_type);
        DBMS_OUTPUT.PUT_LINE('BIKE MANUFACTURER: ' || v_manufacturer);
        DBMS_OUTPUT.PUT_LINE('BIKE VALUE: R' || TO_CHAR(v_value, '9G999'));
        DBMS_OUTPUT.PUT_LINE('STATUS: ' || v_status);
        DBMS_OUTPUT.PUT_LINE('--------------------------------');
    END LOOP;
END;
/
```

```
--------------------------------
BIKE ID: B001
BIKE TYPE: Road Bike
BIKE MANUFACTURER: BMX
BIKE VALUE: R 1,500
STATUS: *
--------------------------------
--------------------------------
BIKE ID: B002
BIKE TYPE: Road Bike
BIKE MANUFACTURER: Giant
BIKE VALUE: R 2,500
STATUS: **
--------------------------------
--------------------------------
BIKE ID: B003
BIKE TYPE: Mountain Bike
BIKE MANUFACTURER: Raleigh
BIKE VALUE: R 1,000
STATUS: *
--------------------------------
--------------------------------
BIKE ID: B004
BIKE TYPE: Kids Bike
BIKE MANUFACTURER: Canyon
BIKE VALUE: R 1,750
STATUS: **
--------------------------------
--------------------------------
BIKE ID: B006
BIKE TYPE: Mountain Bike
BIKE MANUFACTURER: BMX
BIKE VALUE: R 2,000
STATUS: **
--------------------------------
--------------------------------
BIKE ID: B007
BIKE TYPE: Road Bike
BIKE MANUFACTURER: Canyon
BIKE VALUE: R 1,800
STATUS: **
--------------------------------
```

```
--------------------------------
BIKE ID: B008
BIKE TYPE: Hybrid Bike
BIKE MANUFACTURER: BMX
BIKE VALUE: R 1,500
STATUS: *
--------------------------------
--------------------------------
BIKE ID: B009
BIKE TYPE: Road Bike
BIKE MANUFACTURER: Trek
BIKE VALUE: R 1,500
STATUS: *
--------------------------------
--------------------------------
BIKE ID: B010
BIKE TYPE: Mountain Bike
BIKE MANUFACTURER: Trek
BIKE VALUE: R 2,500
STATUS: **
--------------------------------
--------------------------------
BIKE ID: B005
BIKE TYPE: Gravel Road Bike
BIKE MANUFACTURER: BMX
BIKE VALUE: R 3,500
STATUS: ***
--------------------------------
--------------------------------
BIKE ID: B011
BIKE TYPE: Road Bike
BIKE MANUFACTURER: BMX
BIKE VALUE: R 2,500
STATUS: **
--------------------------------
```

Question 8

```
-----------------------------------
--QUESTION 8
-----------------------------------
SELECT
    BIKE_ID,
    BIKE_TYPE,
    MANUFACTURER,
    VALUE AS BIKE_VALUE,
    CASE
        WHEN VALUE <= 1500 THEN '*'
        WHEN VALUE > 1500 AND VALUE <= 3000 THEN '**'
        WHEN VALUE > 3000 THEN '***'
        ELSE 'No Rating'
    END AS STATUS
FROM
    BIKE
ORDER BY
    BIKE_ID;
```

ery Result ×

SQL | All Rows Fetched: 11 in 1,345 seconds

	BIKE ID	BIKE TYPE	MANUFACTURER	BIKE VALUE	STATUS
1	B001	Road Bike	BMX	1500	*
2	B002	Road Bike	Giant	2500	**
3	B003	Mountain Bike	Raleigh	1000	*
4	B004	Kids Bike	Canyon	1750	**
5	B005	Gravel Road Bike	BMX	3500	***
6	B006	Mountain Bike	BMX	2000	**
7	B007	Road Bike	Canyon	1800	**
8	B008	Hybrid Bike	BMX	1500	*
9	B009	Road Bike	Trek	1500	*
10	B010	Mountain Bike	Trek	2500	**
11	B011	Road Bike	BMX	2500	**

Question 9

I used LiveSQL to generate and test my SQL code for q9 since because my SQL Developer user was under SYS, it could not create a trigger so I used LiveSQL to demonstrate that my code is working.

Q 9.1

```
-------------------------------------
--QUESTION 9
-------------------------------------
--Q 9.1
-------------------------------------
CREATE OR REPLACE TRIGGER trg_prevent_donation_delete
BEFORE DELETE ON DONATION
FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20001, 'ERROR: Deletion of donation records is not allowed!');
END;
/

-- Attempt to delete a record
DELETE FROM DONATION WHERE DONATION_ID = 1;
```

```
SQL> CREATE OR REPLACE TRIGGER trg_prevent_donation_delete
     BEFORE DELETE ON DONATION
     FOR EACH ROW
     BEGIN...
Show more...



Trigger TRG_PREVENT_DONATION_DELETE compiled

No errors.
Elapsed: 00:00:00.018
```

Q 9.2

```
-------------------------------------
--Q 9.2
-------------------------------------
CREATE OR REPLACE TRIGGER trg_validate_donation_value
BEFORE UPDATE OF VALUE ON DONATION
FOR EACH ROW
BEGIN
    IF :NEW.VALUE <= 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'ERROR: Donation value must be greater than 0!');
    END IF;
END;
/

-- Test with an invalid value (should fail)
UPDATE DONATION
SET VALUE = 0
WHERE DONATION_ID = 2;

-- Test with a valid value (should succeed)
UPDATE DONATION
SET VALUE = 2500
WHERE DONATION_ID = 2;
```

Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.007 seconds

	BIKE ID	BIKE TYPE	MANUFACTURER	BIKE VALUE	STATUS
1	B001	Road Bike	BMX	1500	*
2	B002	Road Bike	Giant	2500	**
3	B003	Mountain Bike	Raleigh	1000	*
4	B004	Kids Bike	Canyon	1750	**
5	B005	Gravel Road Bike	BMX	3500	***
6	B006	Mountain Bike	BMX	2000	**
7	B007	Road Bike	Canyon	1800	**
8	B008	Hybrid Bike	BMX	1500	*
9	B009	Road Bike	Trek	1500	*
10	B010	Mountain Bike	Trek	2500	**
11	B011	Road Bike	BMX	2500	**

Question 10

Technical Report on Ensuring Confidentiality, Integrity, and Availability of Data at BikesRUs

1. Introduction

In the current digital environment, safeguarding organizational information has become an essential aspect of sustainable business practices. For BikesRUs, a nonprofit organization that oversees donations, bicycles, contributors, and volunteers, securing data is vital for sustaining trust, ensuring compliance, and maintaining operational continuity. This document presents a comprehensive technical strategy to uphold the confidentiality, integrity, and availability (CIA) of the organization's data. It elaborates on how these principles can be applied through a mix of secure technologies, database management methods, and administrative controls, backed by relevant tools and sample code for BikesRUs' Oracle database context.

2. Ensuring Confidentiality

Maintaining confidentiality is essential for preventing illicit access to sensitive information like donor information, volunteer records, and details of donations (Kidd, 2024). Leaking such information can lead to breaches of privacy and harm to one's reputation. To uphold confidentiality, it is necessary to implement various technical and procedural protections (Kidd, 2024).

2.1. Database Access Control

Establishing role-based access control (RBAC) in Oracle guarantees that only permitted users can carry out particular tasks (Anna, 2024).

For instance:

```
-- Create roles for different system users
CREATE ROLE donor_clerk;
CREATE ROLE admin_user;
-- Assign permissions according to job responsibilities.
GRANT SELECT, INSERT ON DONATION TO donor_clerk;
GRANT ALL PRIVILEGES ON DONATION TO admin_user;
```

This guarantees that clerks can input donation information but cannot erase or change current records, whereas administrators have complete access (Anna, 2024).

2.2. Data Encryption

Encryption safeguards data whether it is stored or being transmitted (Microsoft, 2025). Oracle Transparent Data Encryption (TDE) is capable of encrypting sensitive information, including donor contact numbers and email addresses (Microsoft, 2025).

```
ALTER TABLE DONOR MODIFY (CONTACT_NO ENCRYPT USING 'AES256');
ALTER TABLE DONOR MODIFY (EMAIL ENCRYPT USING 'AES256');
```

This guarantees that even if the database files are directly accessed, the data stays encrypted (Microsoft, 2025). For securing data in transit, BikesRUs must mandate SSL/TLS connections for accessing the database remotely, safeguarding against the interception of credentials or queries (Microsoft, 2025).

2.3. Secure Authentication and Password Policies

Robust authentication methods are essential (AI2sql, 2025). Oracle provides password profiles that mandate rules regarding length, complexity, and expiration (AI2sql, 2025):

```
CREATE PROFILE secure_profile LIMIT
    PASSWORD_LIFE_TIME 60
    PASSWORD_REUSE_TIME 365
    PASSWORD_VERIFY_FUNCTION ora12c_verify_function;
ALTER USER BIKERUS PROFILE secure_profile;
```

Moreover, enforcing Multi-Factor Authentication (MFA) for administrators, such as through the integration with Oracle Identity Management (OIM) or Azure Active Directory, provides an additional level of security (AI2sql, 2025).

2.4. Data Masking

To safeguard personal information in testing or training settings, BikesRUs ought to utilize Oracle Data Masking and Subsetting (Microsoft, 2025). This guarantees that sensitive data like donor names or contact information is anonymized while preserving realistic formats (Microsoft, 2025).

3. Ensuring Integrity

Integrity involves preserving the accuracy and consistency of data at all stages of its lifecycle. Any corruption or unauthorized changes to data could distort the values of donations or donor records, compromising trust and accountability.

3.1. Constraints and Triggers

Integrity starts with the design of the database schema (w3schools, 2025). Implementing primary keys, foreign keys, and check constraints guarantees referential integrity (w3schools, 2025).

For example:

```
ALTER TABLE DONATION
ADD CONSTRAINT chk_donation_value CHECK (VALUE > 0);
```

Furthermore, triggers can stop invalid updates or deletions, as shown in previous questions, making sure that no donation records are deleted or wrongly changed (w3schools, 2025).

3.2. Auditing and Logging

Oracle Database Auditing (DBA_AUDIT_TRAIL) logs all modifications made to the database (Toorpu, 2025). Activating auditing assists in identifying unauthorized alterations to data (Toorpu, 2025).

```
AUDIT INSERT, UPDATE, DELETE ON DONATION BY ACCESS;
```

Administrators have the ability to regularly examine audit logs to maintain accountability (Toorpu, 2025).

3.3. Backup Validation and Hashing

To ensure the integrity of data that has been transferred or backed up, cryptographic hashing methods (such as SHA-256) can be employed for verification (Gupta, 2019).

Here's a straightforward example in Python to check the integrity of a backup:

```
import hashlib
def hash_file(filename):
    h = hashlib.sha256()
    with open(filename, 'rb') as f:
        h.update(f.read())
    return h.hexdigest()
original = hash_file("donation_backup.sql")
restored = hash_file("donation_restored.sql")
if original == restored:
    print("Backup integrity verified.")
else:
    print("Data corruption detected.")
```

This guarantees that the backup data corresponds to the original source without any alterations (Gupta, 2019).

3.4. Transaction Management

Implementing COMMIT and ROLLBACK commands in Oracle guarantees data integrity during transaction processes (geeksforgeeks, 2025). In the event that an error arises while processing a donation, the transaction can be reverted to uphold precision (geeksforgeeks, 2025).

```
BEGIN
    INSERT INTO DONATION VALUES (12, 'DID21', 'B012', 2000, 'vol103', SYSDATE);
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Transaction failed and was rolled back.');
END;
```

4. Ensuring Availability

Uptime ensures that authorized users can consistently access data whenever necessary. BikesRUs needs to prepare for hardware malfunctions, network interruptions, and system congestion.

4.1. Backup and Recovery

Consistent backups of the database with Oracle Recovery Manager (RMAN) guarantee that data can be recovered in case of damage or loss (w3schools, 2025).

Sample command:

```
rman target /
BACKUP DATABASE PLUS ARCHIVELOG;
```

Cloud-based backups can be stored on either Oracle Cloud Infrastructure (OCI) or Azure Blob Storage for added redundancy (w3schools, 2025).

4.2. High Availability Solutions

BikesRUs has the option to utilize Oracle Data Guard or Real Application Clusters (RAC) to ensure continuous database availability (PaperCut, 2025). These solutions copy the database to backup servers and automatically switch over during system outages (PaperCut, 2025).

4.3. Monitoring and Maintenance

Solutions like Oracle Enterprise Manager or open-source options such as Zabbix are capable of overseeing performance, storage, and system health in real-time, notifying administrators ahead of potential failures (AdvancedTech, 2025).

5. Conclusion

The CIA triad, Confidentiality, Integrity, and Availability, serves as the cornerstone of data security for BikesRUs. By utilizing encryption, implementing role-based access control, and enforcing password policies, confidentiality can be maintained. Data integrity is strengthened through the use of database constraints, auditing processes, and transaction management, while availability is ensured by comprehensive backup plans and failover solutions. Collectively, these strategies create a strong security framework that fosters donor trust, adheres to data protection regulations, and supports the organization's long-term reliability.

With effective planning and ongoing monitoring, BikesRUs can reliably protect its important data assets from both accidental and intentional threats.

References

AdvancedTech, 2025. *What is Condition-Based Monitoring and Maintenance?.* [Online]
Available at: https://www.advancedtech.com/blog/condition-based-monitoring-and-maintenance/
[Accessed 2025].

AI2sql, 2025. *Secure SQL Authentication — Examples & 2025 Guide.* [Online]
Available at: https://ai2sql.io/learn/secure-sql-authentication
[Accessed 2025].

Anna, 2024. *Access Controls for Users and Roles in SQL.* [Online]
Available at: https://dev.to/anna_p_s/access-controls-for-users-and-roles-in-sql-44b6
[Accessed 2025].

geeksforgeeks, 2025. *SQL TRANSACTIONS.* [Online]
Available at: https://www.geeksforgeeks.org/sql/sql-transactions/
[Accessed 2025].

Gupta, R., 2019. *Validate backups with SQL restore database operations using DBATools.* [Online]
Available at: https://www.sqlshack.com/validate-backups-with-sql-restore-database-operations-using-dbatools/#:~:text=Example%204:%20Verify%20backups%20without%20database%20restoration%0A%0AIn,using%20a%20parameter%20%2DVerifyOnly%20in%20Test%2DDbaLastBackup%20command.
[Accessed 2025].

Kidd, C., 2024. *What's The CIA Triad? Confidentiality, Integrity, & Availability, Explained.* [Online]
Available at: https://www.splunk.com/en_us/blog/learn/cia-triad-confidentiality-integrity-availability.html
[Accessed 2025].

Microsoft, 2025. *Dynamic data masking.* [Online]
Available at: https://learn.microsoft.com/en-us/sql/relational-databases/security/dynamic-data-masking?view=sql-server-ver17
[Accessed 2025].

Microsoft, 2025. *Transparent data encryption (TDE).* [Online]
Available at: https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/transparent-data-encryption?view=sql-server-ver17
[Accessed 2025].

PaperCut, 2025. *The ultimate guide to High Availability methods for Microsoft SQL Server.* [Online]
Available at: https://www.papercut.com/discover/best-practices/high-availability-methods-for-microsoft-sql-server/
[Accessed 2025].

Toorpu, A., 2025. *Auditing SQL Server Database Users, Logins, and Activity: A Comprehensive Guide.* [Online]
Available at: https://dev.to/arvind_toorpu/auditing-sql-server-database-users-logins-and-activity-a-comprehensive-guide-4dk4#:~:text=1.,Create%20a%20Session:
[Accessed 2025].

w3schools, 2025. *SQL BACKUP DATABASE for SQL Server.* [Online]
Available at: https://www.w3schools.com/sql/sql_backup_db.asp
[Accessed 2025].

w3schools, 2025. *SQL Constraints.* [Online]
Available at: https://www.w3schools.com/sql/sql_constraints.asp
[Accessed 2025].

Disclosure of AI Usage in my Assessment

Section(s) within the assessment in which generative AI was used:

Question 4 and 6

Name of AI tool(s) used:

ChatGPT

Purpose/Intention:

To understand certain topics/concepts and use that understanding to answer the questions.

Date(s) in which generative AI was used:

10 Nov 2025

Conversation Links:

https://chatgpt.com/share/691218b0-3ab4-8011-a18b-77679035446d