

Muhammad Rahim

ST10043611

ADDB7311

Summative Project

Table of Contents

Question 1	3
Proof Question 1 Tables Created (Please Zoom in to see image clearer)	7
Proof Question 1 Volunteer Table Created and inserted data (Please Zoom in to see image clearer)	7
Proof Question 1 Donor Table Created and inserted data (Please Zoom in to see image clearer)	8
Proof Question 1 Donation Table Created and inserted data (Please Zoom in to see image clearer)	8
Proof Question 1 Bike Table Created and inserted data (Please Zoom in to see image clearer)	9
Question 2	10
Proof of Question 2 (Please Zoom in to see image clearer)	10
Question 3	11
Proof of Question 3 (Please Zoom in to see image clearer)	12
Question 4	13
Proof of Question 4 (Please Zoom in to see image clearer)	14
Question 5	15
Proof of Question 5 Without Error Message Handled (Please Zoom in to see image clearer)	18
Proof of Question 5 With Error Message Handled (Please Zoom in to see image clearer)	19
Question 6	20
Proof of Question 6 Without Error Message Handled (Please Zoom in to see image clearer)	23
Proof of Question 6 With Error Message Handled (Please Zoom in to see image clearer)	23
Question 7	24
Proof of Question 7 Part 1 (Please Zoom in to see image clearer)	25
Proof of Question 7 Part 2 (Please Zoom in to see image clearer)	26
Question 8	27
Proof of Question 8 (Please Zoom in to see image clearer)	28
Question 9	29
Proof of Error I am getting with regards to Question 9 (Please Zoom in to see image clearer)	31
Reason for this error:	31
Question 10	32
1. Confidentiality	32
2. Integrity	33
3. Accessibility	34
4. Additional Measures for Security and Performance	34
References Used in this Summative Assessment	36
References	36

Question 1

-- Create VOLUNTEER table

```
CREATE TABLE VOLUNTEER (  
    VOL_ID VARCHAR2(10) PRIMARY KEY,  
    VOL_FNAME VARCHAR2(50),  
    VOL_SNAME VARCHAR2(50),  
    CONTACT VARCHAR2(15),  
    ADDRESS VARCHAR2(100),  
    EMAIL VARCHAR2(100)  
);
```

-- Create DONOR table

```
CREATE TABLE DONOR (  
    DONOR_ID VARCHAR2(10) PRIMARY KEY,  
    DONOR_FNAME VARCHAR2(50),  
    DONOR_LNAME VARCHAR2(50),  
    CONTACT_NO VARCHAR2(15),  
    EMAIL VARCHAR2(100)  
);
```

-- Create BIKE table

```
CREATE TABLE BIKE (  
    BIKE_ID VARCHAR2(10) PRIMARY KEY,  
    DESCRIPTION VARCHAR2(100),  
    BIKE_TYPE VARCHAR2(50),  
    MANUFACTURER VARCHAR2(50)  
);
```

-- Create DONATION table

```
CREATE TABLE DONATION (  
    DONATION_ID NUMBER PRIMARY KEY,  
    DONOR_ID VARCHAR2(10),  
    BIKE_ID VARCHAR2(10),  
    VALUE NUMBER(10, 2),  
    VOLUNTEER_ID VARCHAR2(10),  
    DONATION_DATE DATE,  
    FOREIGN KEY (DONOR_ID) REFERENCES DONOR(DONOR_ID),  
    FOREIGN KEY (BIKE_ID) REFERENCES BIKE(BIKE_ID),  
    FOREIGN KEY (VOLUNTEER_ID) REFERENCES VOLUNTEER(VOL_ID)  
);
```

-- Insert data into VOLUNTEER table

```
INSERT INTO VOLUNTEER VALUES ('vol101', 'Kenny', 'Temba', '0677277521', '10 Sands Road',  
'kennyt@bikerus.com');
```

```
INSERT INTO VOLUNTEER VALUES ('vol102', 'Mamelodi', 'Marks', '0737377522', '20 Langes  
Street', 'mamelodim@bikerus.com');
```

```
INSERT INTO VOLUNTEER VALUES ('vol103', 'Ada', 'Andrews', '0817117523', '31 Williams Street',  
'adanyaa@bikerus.com');
```

```
INSERT INTO VOLUNTEER VALUES ('vol104', 'Evans', 'Tambala', '0697215244', '1 Free Road',  
'evanst@bikerus.com');
```

```
INSERT INTO VOLUNTEER VALUES ('vol105', 'Xolani', 'Samson', '0727122255', '12 main road',  
'xolanis@bikerus.com');
```

-- Insert data into DONOR table

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 data into BIKE table

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 data into DONATION table

```
INSERT INTO DONATION VALUES (1, 'DID11', 'B001', 1500, 'vol101', TO_DATE('01-MAY-23', 'DD-MON-YY'));
```

```
INSERT INTO DONATION VALUES (2, 'DID12', 'B002', 2500, 'vol101', TO_DATE('03-MAY-23', 'DD-MON-YY'));
```

```
INSERT INTO DONATION VALUES (3, 'DID13', 'B003', 1000, 'vol103', TO_DATE('03-MAY-23', 'DD-MON-YY'));
```

```
INSERT INTO DONATION VALUES (4, 'DID14', 'B004', 1750, 'vol105', TO_DATE('05-MAY-23', 'DD-MON-YY'));
```

```
INSERT INTO DONATION VALUES (5, 'DID15', 'B006', 2000, 'vol101', TO_DATE('07-MAY-23', 'DD-MON-YY'));
```

```
INSERT INTO DONATION VALUES (6, 'DID16', 'B007', 1800, 'vol105', TO_DATE('09-MAY-23', 'DD-MON-YY'));
```

```
INSERT INTO DONATION VALUES (7, 'DID17', 'B008', 1500, 'vol101', TO_DATE('15-MAY-23', 'DD-MON-YY'));
```

```
INSERT INTO DONATION VALUES (8, 'DID18', 'B009', 1500, 'vol103', TO_DATE('19-MAY-23', 'DD-MON-YY'));
```

```
INSERT INTO DONATION VALUES (9, 'DID12', 'B010', 2500, 'vol103', TO_DATE('25-MAY-23', 'DD-MON-YY'));
```

```
INSERT INTO DONATION VALUES (10, 'DID20', 'B005', 3500, 'vol105', TO_DATE('05-MAY-23', 'DD-MON-YY'));
```

```
INSERT INTO DONATION VALUES (11, 'DID19', 'B011', 2500, 'vol103', TO_DATE('30-MAY-23', 'DD-MON-YY'));
```

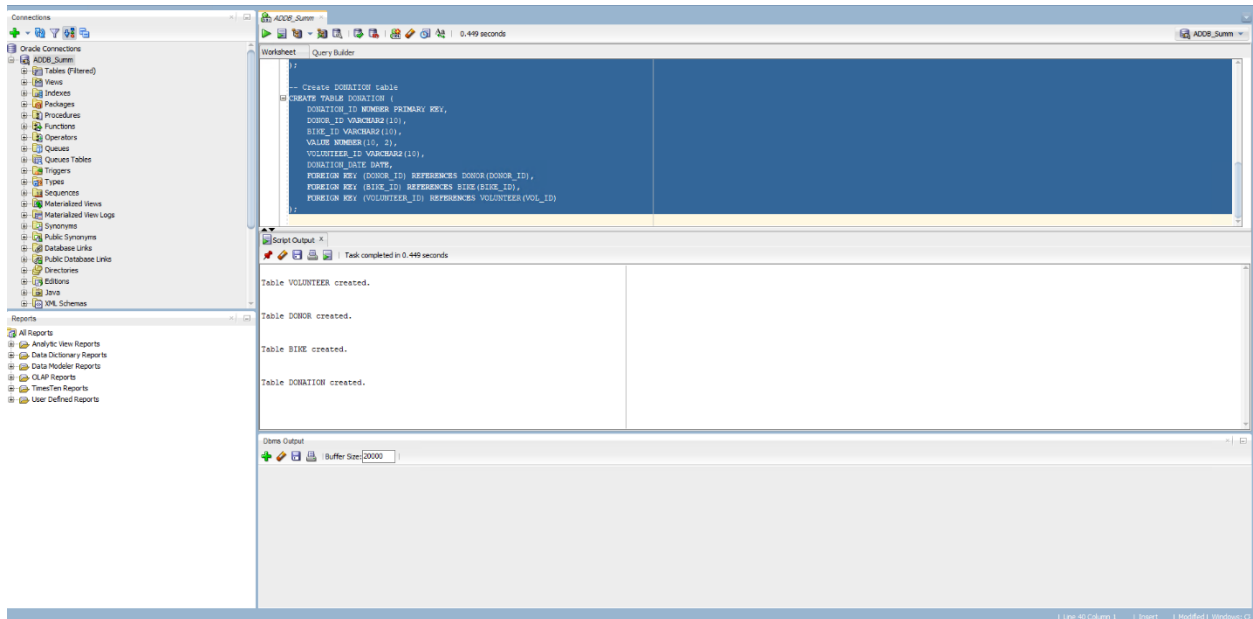
```
SELECT * FROM VOLUNTEER;
```

```
SELECT * FROM DONOR;
```

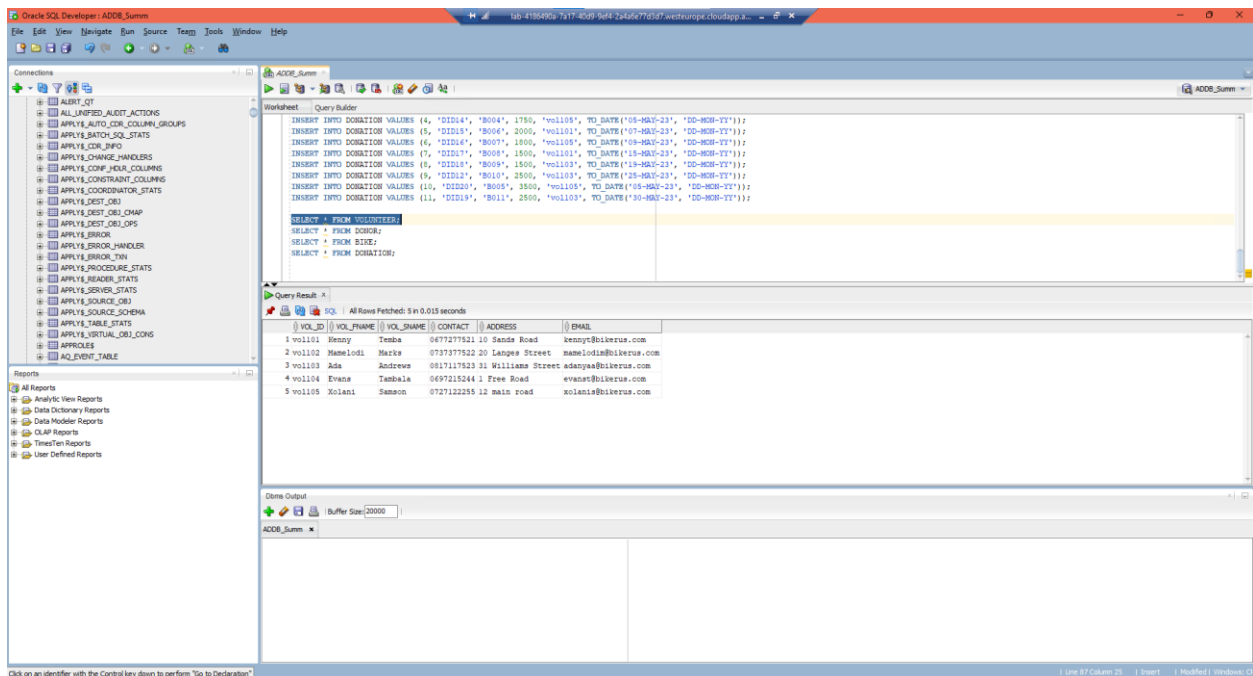
```
SELECT * FROM BIKE;
```

```
SELECT * FROM DONATION;
```

Proof Question 1 Tables Created (Please Zoom in to see image clearer)



Proof Question 1 Volunteer Table Created and inserted data (Please Zoom in to see image clearer)



Oracle SQL Developer - ACDSS_Summ

File Edit View Navigate Run Source Tools Window Help

Connections

- ACDSS_Summ

Schema Tree:

- ALL_RT
- ALL_UNPISD_AUDIT_ACTIONS
- APPLY_AUTO_CDM_COLUMN_GROUPS
- APPLY_BATCH_SQL_STATS
- APPLY_CDR_INFO
- APPLY_CHANGE_HANDLERS
- APPLY_CDM_VIEW_COLUMNS
- APPLY_CONSTRAINT_COLUMNS
- APPLY_COORDINATOR_STATS
- APPLY_DEST_OBI
- APPLY_DEST_OBI_CHAP
- APPLY_DEST_OBI_OPS
- APPLY_ERROR
- APPLY_ERROR_HANDLER
- APPLY_ERROR_TN
- APPLY_PROCEDURE_STATS
- APPLY_READER_STATS
- APPLY_SERVER_STATS
- APPLY_SOURCE_OBI
- APPLY_SOURCE_SCHEMA
- APPLY_TABLE_STATS
- APPLY_VIRTUAL_OBI_COLS
- APPLYVIEW
- AQ_EVENT_TABLE

Worksheet - Query Builder

```

INSERT INTO DONATION VALUES (4, 'D1014', 'B004', 1750, '20110515', TO_DATE('10-MAY-23', 'DD-MON-YY'));
INSERT INTO DONATION VALUES (5, 'D1015', 'B006', 2000, '20110515', TO_DATE('07-MAY-23', 'DD-MON-YY'));
INSERT INTO DONATION VALUES (6, 'D1016', 'B007', 1800, '20110515', TO_DATE('09-MAY-23', 'DD-MON-YY'));
INSERT INTO DONATION VALUES (7, 'D1017', 'B008', 1500, '20110515', TO_DATE('15-MAY-23', 'DD-MON-YY'));
INSERT INTO DONATION VALUES (8, 'D1018', 'B009', 1500, '20110515', TO_DATE('18-MAY-23', 'DD-MON-YY'));
INSERT INTO DONATION VALUES (9, 'D1019', 'B010', 2500, '20110515', TO_DATE('25-MAY-23', 'DD-MON-YY'));
INSERT INTO DONATION VALUES (10, 'D1020', 'B006', 3500, '20110515', TO_DATE('10-MAY-23', 'DD-MON-YY'));
INSERT INTO DONATION VALUES (11, 'D1019', 'B011', 2500, '20110515', TO_DATE('30-MAY-23', 'DD-MON-YY'));

SELECT * FROM VOLUNTEER;
SELECT * FROM DONGRO;
SELECT * FROM SITE;
SELECT * FROM DONATION;

```

Query Result

10 Rows Fetched: 10 in 0.012 seconds

	DONOR_ID	DONOR_NAME	DONOR_PHONE	CONTACT_ID	EMAIL
1	D1011	Jeff Wanya	0827172250	wanya.jeff@gmail.com	
2	D1012	Schenbeni Fisho	0037654570	sthepfisho@gmail.com	
3	D1013	James Temba	0019787650	jtemba@gmail.com	
4	D1014	Lucama Misi	0024575650	lucama@gmail.com	
5	D1015	Abraham Xolani	0797454430	axolani@gmail.com	
6	D1016	Rumi Jones	0668595021	rjones@true.com	
7	D1017	Xolani Redo	0414533339	xredo@gmail.com	
8	D1018	Tenny Sears	004228870	tennsears@true.com	
9	D1019	Tilly Rambo	0715554333	trambo@gmail.com	
10	D1020	Yannick Leons	0615554323	yleons@true.com	

Data Output

Buffer Size: 20000

ACDSS_Summ

The screenshot displays the Oracle SQL Developer interface. The top menu bar includes File, Edit, View, Navigate, Run, Source, Tools, Window, and Help. The main workspace is divided into three panes:

- Connections:** Shows a tree view of database connections under the 'ACD\$ Summ' connection.
- Schema Browser:** Displays the 'ACD\$ Summ' schema structure, listing various tables such as ALL_AUDIT_ACTIONS, APPLYS_AUTO_CDR_COLUMNS_GROUPS, APPLYS_BATCH_STATS, APPLYS_CDR_INFO, APPLYS_CHANGE_HANDLERS, APPLYS_COST_ACH_COLUMNS, APPLYS_CONSTRAINT_COLUMNS, APPLYS_COORDINATOR_STATS, APPLYS_DEST_OBI, APPLYS_DEST_OBI_CHAP, APPLYS_DEST_OBI_LOPS, APPLYS_ERROR, APPLYS_ERROR_HANDLER, APPLYS_ERROR_TN, APPLYS_PROCEDURE_STATS, APPLYS_READER_STATS, APPLYS_SERVER_STATS, APPLYS_SOURCE_OBI, APPLYS_SOURCE_SCHEMA, APPLYS_TABLE_STATS, APPLYS_VIRTUAL_OBI_CONG, APPLOCKER, and AQ_EVENT_TABLE.
- Query Editor:** Contains a SQL script for inserting data into the DONATION table. The script uses a sequence of INSERT INTO statements with values for DONOR_ID, EDE_ID, VALUE, VOLUNTEER_ID, and DONATION_DATE. Below the script, the 'Query Result' pane shows 11 rows fetched in 0.01 seconds, displaying columns DONATION_ID, DONOR_ID, EDE_ID, VALUE, VOLUNTEER_ID, and DONATION_DATE.

DONATION_ID	DONOR_ID	EDE_ID	VALUE	VOLUNTEER_ID	DONATION_DATE
1	1 DED11	B001	1500	voll101	01-MAY-23
2	2 DED12	B002	2500	voll101	03-MAY-23
3	3 DED13	B003	1000	voll103	03-MAY-23
4	4 DED14	B004	1750	voll105	05-MAY-23
5	5 DED15	B006	2000	voll101	07-MAY-23
6	6 DED16	B007	1800	voll105	09-MAY-23
7	7 DED17	B008	1500	voll101	15-MAY-23
8	8 DED18	B009	1500	voll103	19-MAY-23
9	9 DED12	B010	2500	voll103	25-MAY-23
10	10 DED20	B005	3500	voll105	05-JUN-23
11	11 DED19	B011	2500	voll103	30-MAY-23

Proof Question 1 Bike Table Created and inserted data (Please Zoom in to see image clearer)

The screenshot displays the Oracle SQL Developer interface. On the left, the 'Connections' pane shows the 'ACOE_Summit' connection. The 'Query Builder' pane in the center shows the following SQL script:

```
INSERT INTO DONATION VALUES (4, 'DID14', 'B004', 1750, 'vol105', TO_DATE('05-MAY-23', 'DD-MON-YY'));  
INSERT INTO DONATION VALUES (5, 'DID15', 'B004', 2000, 'vol101', TO_DATE('07-MAY-23', 'DD-MON-YY'));  
INSERT INTO DONATION VALUES (6, 'DID16', 'B007', 1500, 'vol105', TO_DATE('08-MAY-23', 'DD-MON-YY'));  
INSERT INTO DONATION VALUES (7, 'DID17', 'B008', 1500, 'vol101', TO_DATE('15-MAY-23', 'DD-MON-YY'));  
INSERT INTO DONATION VALUES (8, 'DID18', 'B009', 1500, 'vol103', TO_DATE('19-MAY-23', 'DD-MON-YY'));  
INSERT INTO DONATION VALUES (9, 'DID19', 'B010', 2500, 'vol103', TO_DATE('25-MAY-23', 'DD-MON-YY'));  
INSERT INTO DONATION VALUES (10, 'DID20', 'B008', 3500, 'vol105', TO_DATE('18-JUN-23', 'DD-MON-YY'));  
INSERT INTO DONATION VALUES (11, 'DID19', 'B011', 2500, 'vol103', TO_DATE('30-JUN-23', 'DD-MON-YY'));
```

Below the script, the 'Query Results' pane shows the following data:

Bike_ID	DESCRIPTION	Bike_Type	MANUFACTURER
1 B001	Bike AX1	Road Bike	BMS
2 B002	Giant Domain 1	Road Bike	Giant
3 B003	Avant 2421	Mountain Bike	Raleigh
4 B004	Canyon 6X	Ride Bike	Canyon
5 B005	Marvel Gevel	Road Bike	BMS
6 B006	Mountain 21 Speed	Mountain Bike	BMS
7 B007	Canyon Roadster	Road Bike	Canyon
8 B008	Canyon 101	Hybrid Bike	BMS
9 B009	Madone 9	Road Bike	Trek
10 B010	Comp 2022	Mountain Bike	Trek
11 B011	Bike AX15	Road Bike	BMS

The 'Data Output' pane at the bottom shows the table structure for 'ACOE_Summit'.

Question 2

SELECT

D.DONOR_ID,

B.BIKE_TYPE,

B.DESCRPTION AS BIKE_DESCRIPTION,

DON.VALUE AS BIKE_VALUE

FROM

DONATION DON

JOIN

BIKE B ON DON.BIKE_ID = B.BIKE_ID

JOIN

DONOR D ON DON.DONOR_ID = D.DONOR_ID

WHERE

DON.VALUE > 1500;

Proof of Question 2 (Please Zoom in to see image clearer)

The screenshot shows the Oracle SQL Developer interface. The 'Query Builder' window displays the following SQL query:

```
SELECT
  D.DONOR_ID,
  B.BIKE_TYPE,
  B.DESCRPTION AS BIKE_DESCRIPTION,
  DON.VALUE AS BIKE_VALUE
FROM
  DONATION DON
JOIN
  BIKE B ON DON.BIKE_ID = B.BIKE_ID
JOIN
  DONOR D ON DON.DONOR_ID = D.DONOR_ID
WHERE
  DON.VALUE > 1500;
```

The 'Script Output' window shows the query results:

DONOR_ID	BIKE_TYPE	BIKE_DESCRIPTION	BIKE_VALUE
1 DID12	Road Bike	Giant Domain 1	2500
2 DID14	Ride Bike	Canyon EL	1750
3 DID20	Road Bike	Marvél Gravel	3500
4 DID15	Mountain Bike	Mountain 21 Speed	2000
5 DID16	Road Bike	Canyon Roadster	1800
6 DID12	Mountain Bike	Comp 2022	2500
7 DID19	Road Bike	BMX AX15	2500

Question 3

```
-- Define VAT constant

DECLARE

    VAT_RATE CONSTANT NUMBER := 0.15; -- 15% VAT

BEGIN

    -- Query to fetch bike details along with VAT and total amount

    FOR rec IN (

        SELECT

            b.DESCRPTION AS bike_description,

            b.MANUFACTURER AS bike_manufacturer,

            b.BIKE_TYPE AS bike_type,

            d.VALUE AS value,

            ROUND(d.VALUE * VAT_RATE, 2) AS vat,

            ROUND(d.VALUE + (d.VALUE * VAT_RATE), 2) AS total_amount

        FROM

            BIKE b

        JOIN

            DONATION d ON b.BIKE_ID = d.BIKE_ID

        WHERE

            b.BIKE_TYPE = 'Road Bike'

    ) LOOP

        -- Output the formatted result

        DBMS_OUTPUT.PUT_LINE('BIKE DESCRIPTION: ' || rec.bike_description);

        DBMS_OUTPUT.PUT_LINE('BIKE MANUFACTURER: ' || rec.bike_manufacturer);

        DBMS_OUTPUT.PUT_LINE('BIKE TYPE: ' || rec.bike_type);

        DBMS_OUTPUT.PUT_LINE('VALUE: R' || TO_CHAR(rec.value, 'FM9999999.00'));

        DBMS_OUTPUT.PUT_LINE('VAT: R' || TO_CHAR(rec.vat, 'FM9999999.00'));

        DBMS_OUTPUT.PUT_LINE('TOTAL AMNT: R' || TO_CHAR(rec.total_amount, 'FM9999999.00'));
```

```

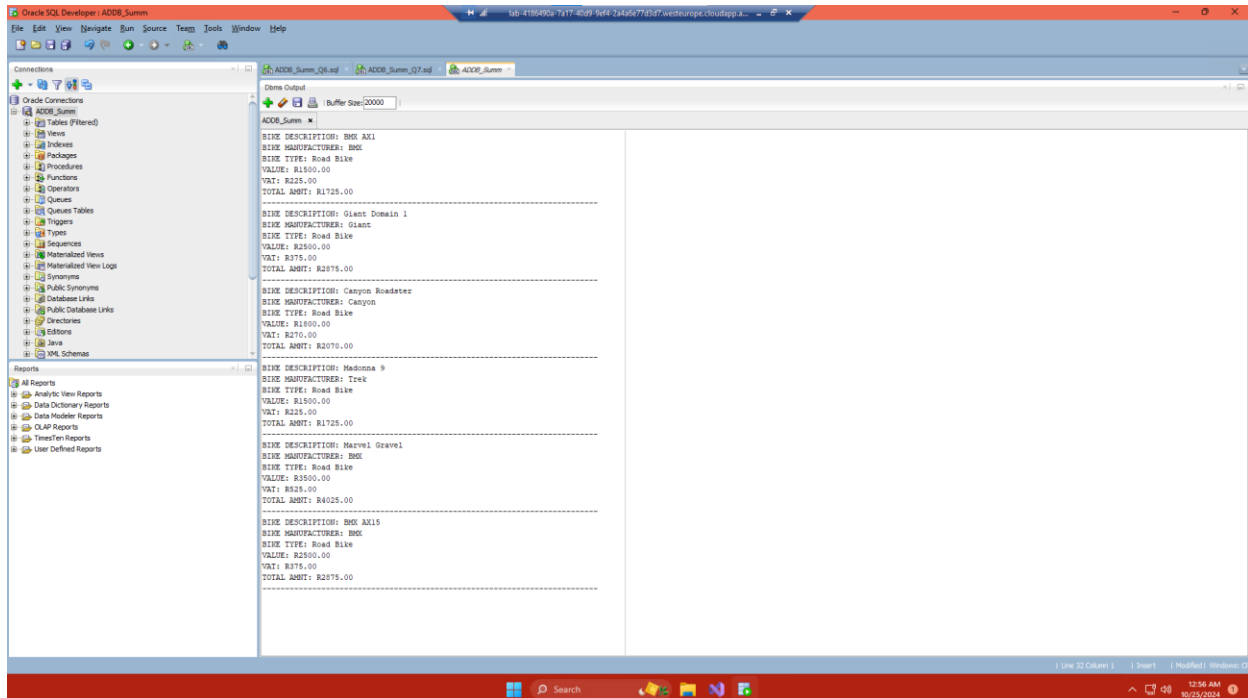
        DBMS_OUTPUT.PUT_LINE('-----');
    END LOOP;

END;

/

```

Proof of Question 3 (Please Zoom in to see image clearer)



Question 4

-- Create a view for donor's name, contact, bike type, and donation date for vol105

CREATE OR REPLACE VIEW vwBikeRUs AS

SELECT

D.DONOR_FNAME || ' ' || D.DONOR_LNAME AS DONOR_NAME,

D.CONTACT_NO AS DONOR_CONTACT,

B.BIKE_TYPE,

DON.DONATION_DATE

FROM

DONATION DON

JOIN

DONOR D ON DON.DONOR_ID = D.DONOR_ID

JOIN

BIKE B ON DON.BIKE_ID = B.BIKE_ID

WHERE

DON.VOLUNTEER_ID = 'vol105';

-- Query to run the view and retrieve the data

SELECT * FROM vwBikeRUs;

-- Justification for using the view:

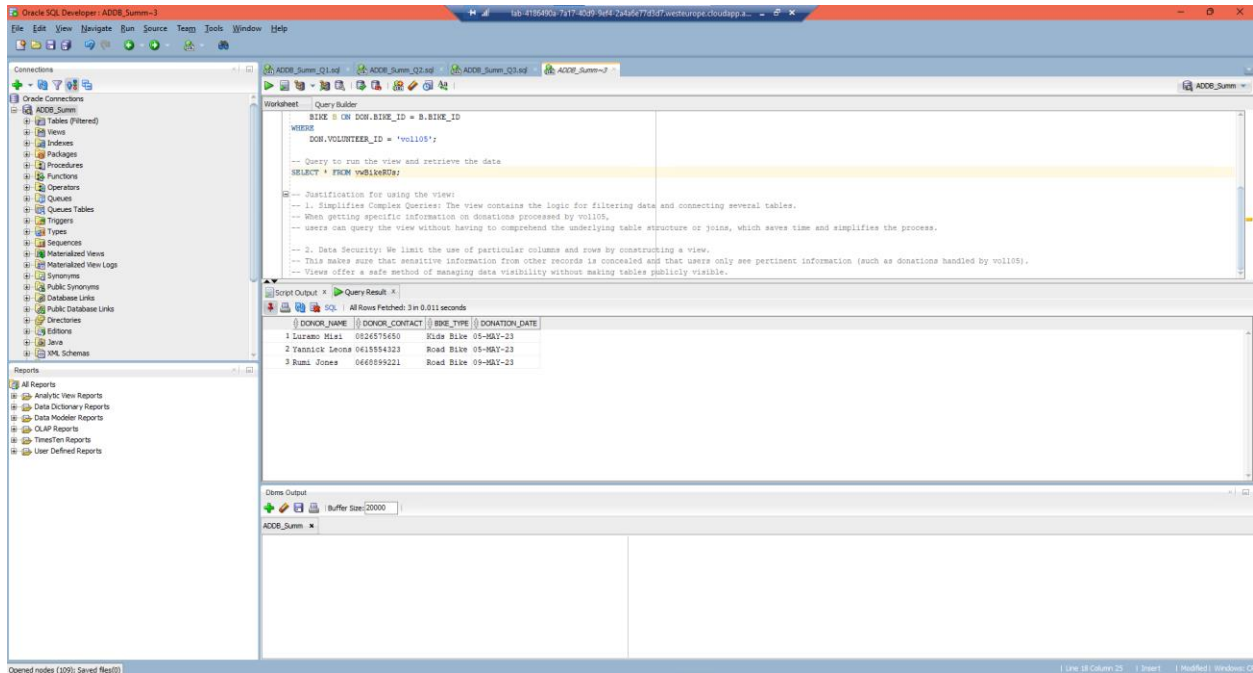
-- 1. Simplifies Complex Queries: The view contains the logic for filtering data and connecting several tables.

-- When getting specific information on donations processed by vol105,

-- users can query the view without having to comprehend the underlying table structure or joins, which saves time and simplifies the process.

- 2. Data Security: We limit the use of columns and rows by constructing a view.
- This makes sure that sensitive information from other records is concealed and that users only see pertinent information (such as donations handled by vol105).
- Views offer a safe method of managing data visibility without making tables publicly visible.

Proof of Question 4 (Please Zoom in to see image clearer)



Question 5

```
-- Create procedure to fetch donor details based on the provided Bike ID

CREATE OR REPLACE PROCEDURE spDonorDetails (
    p_bike_id IN BIKE.BIKE_ID%TYPE
)
IS
    -- Declare variables to store fetched data
    v_donor_name VARCHAR2(100);
    v_donor_contact VARCHAR2(15);
    v_volunteer_fname VARCHAR2(50);
    v_donation_date DATE;

    -- Custom exception
    no_data_found EXCEPTION;

BEGIN
    -- Fetch the donor, volunteer, and donation details for the provided Bike ID
    SELECT
        D.DONOR_FNAME || ' ' || D.DONOR_LNAME AS donor_name,
        D.CONTACT_NO,
        V.VOL_FNAME,
        DON.DONATION_DATE
    INTO
        v_donor_name, v_donor_contact, v_volunteer_fname, v_donation_date
    FROM
        DONATION DON
    JOIN
```

```

        DONOR D ON DON.DONOR_ID = D.DONOR_ID
JOIN
        VOLUNTEER V ON DON.VOLUNTEER_ID = V.VOL_ID
WHERE
        DON.BIKE_ID = p_bike_id;

-- Output the fetched data
DBMS_OUTPUT.PUT_LINE('DONOR NAME: ' || v_donor_name);
DBMS_OUTPUT.PUT_LINE('DONOR CONTACT: ' || v_donor_contact);
DBMS_OUTPUT.PUT_LINE('VOLUNTEER NAME: ' || v_volunteer_fname);
DBMS_OUTPUT.PUT_LINE('DONATION DATE: ' || TO_CHAR(v_donation_date, 'DD-MON-
YYYY'));

EXCEPTION

-- Handle case where no data is found
WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: No data found for the provided Bike ID.');
```

-- Handle any other errors

```

WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);

-- Comments for Exception Handling:
-- 1. When the SELECT statement is unable to locate a matching Bike ID, NO_DATA_FOUND is
utilized to catch the problem.
-- Rather than generating an error, we present a clear response.
-- 2. To aid with troubleshooting, OTHERS records any additional unexpected exceptions and
shows the error message.
```



```
END spDonorDetails;
```

```
/
```

```
-- Executing the Procedure for Bike ID 'B004' without a error
```

```
-- Set server output to display the results for without error
```

```
SET SERVEROUTPUT ON;
```

```
-- Execute the procedure for Bike ID 'B004'
```

```
BEGIN
```

```
    spDonorDetails('B004');
```

```
END;
```

```
/
```

```
-- Executing the Procedure for Bike ID 'B0004' with the error
```

```
-- Set server output to display the results for with the error
```

```
SET SERVEROUTPUT ON;
```

```
-- Execute the procedure for Bike ID 'B0004'
```

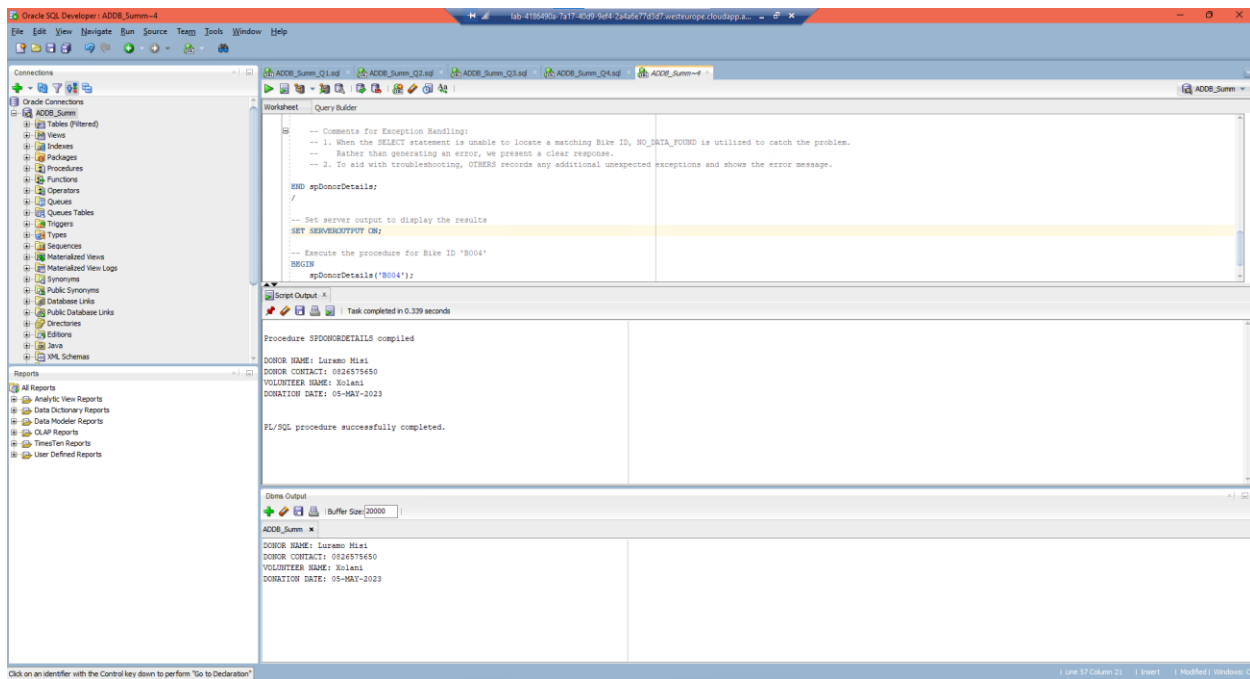
```
BEGIN
```

```
    spDonorDetails('B0004');
```

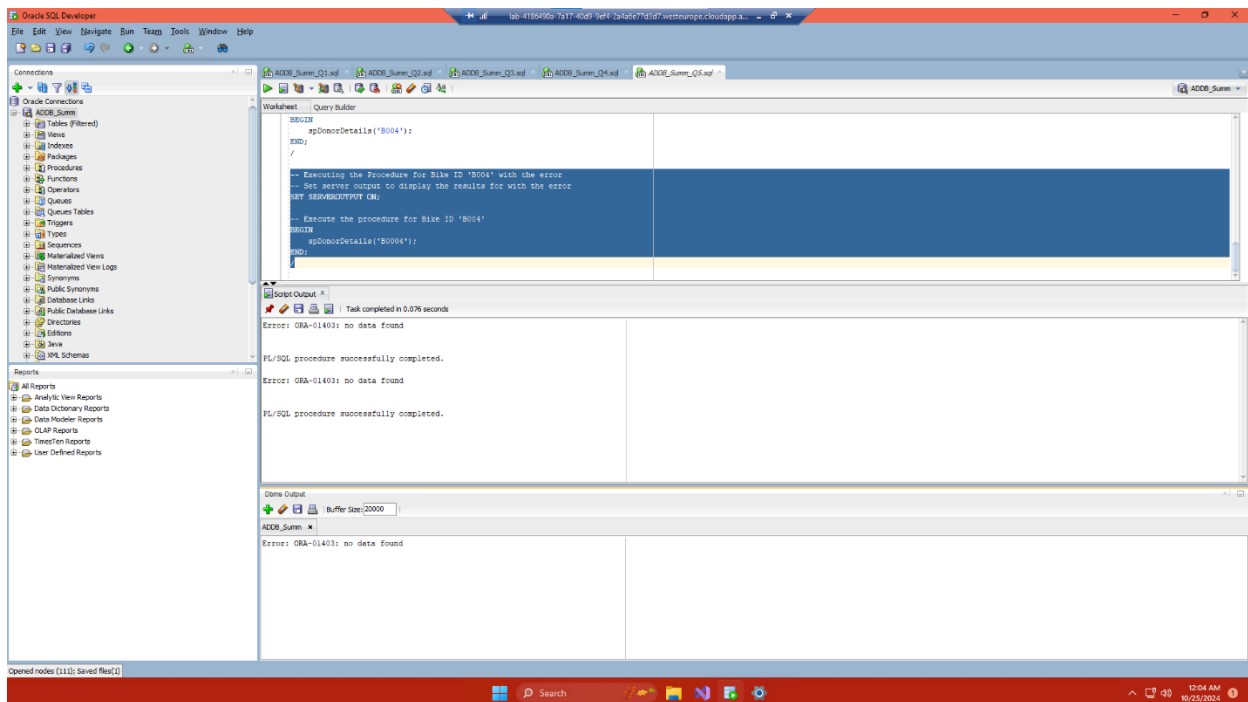
```
END;
```

```
/
```

Proof of Question 5 Without Error Message Handled (Please Zoom in to see image clearer)



Proof of Question 5 With Error Message Handled (Please Zoom in to see image clearer)



Question 6

```
-- Create function to calculate bike status based on value

CREATE FUNCTION fnBikeStatus (
    p_bike_id IN BIKE.BIKE_ID%TYPE
)
RETURN VARCHAR2
IS
    -- Declare a variable to store the bike's value
    v_bike_value DONATION.VALUE%TYPE;
    -- Declare a variable to store the bike's status (1-star, 2-star, 3-star)
    v_bike_status VARCHAR2(10);

BEGIN
    -- Fetch the bike value for the provided Bike ID
    SELECT DON.VALUE
    INTO v_bike_value
    FROM DONATION DON
    WHERE DON.BIKE_ID = p_bike_id;

    -- Determine the status based on the bike's value
    IF v_bike_value BETWEEN 0 AND 1500 THEN
        v_bike_status := '*'; -- 1-star status
    ELSIF v_bike_value > 1500 AND v_bike_value <= 3000 THEN
        v_bike_status := '**'; -- 2-star status
    ELSIF v_bike_value > 3000 THEN
        v_bike_status := '***'; -- 3-star status
    END IF;
```

```

-- Return the bike's status
RETURN v_bike_status;

EXCEPTION

-- Handle the case where the bike ID is not found
WHEN NO_DATA_FOUND THEN
    RETURN 'Error: Bike not found';

-- Handle any other unexpected errors
WHEN OTHERS THEN
    RETURN 'Error: ' || SQLERRM;

-- Comments for Exception Handling:
-- 1. NO_DATA_FOUND: Prevents an unhandled error by catching instances in which the input
    Bike ID is not present in the database.
-- 2. OTHERS: Records any other unexpected errors and provides a detailed generic error
    notice.

END fnBikeStatus;

/

```

-- Executing the Function without an error

-- Set server output to display the results

SET SERVEROUTPUT ON;

-- Execute the function for a sample bike (e.g., B004)

BEGIN

DBMS_OUTPUT.PUT_LINE('Bike Status: ' || fnBikeStatus('B004'));

END;

/

-- Executing the Function with an error

-- Set server output to display the results with the error

SET SERVEROUTPUT ON;

-- Execute the function for a sample bike (e.g., B0004)

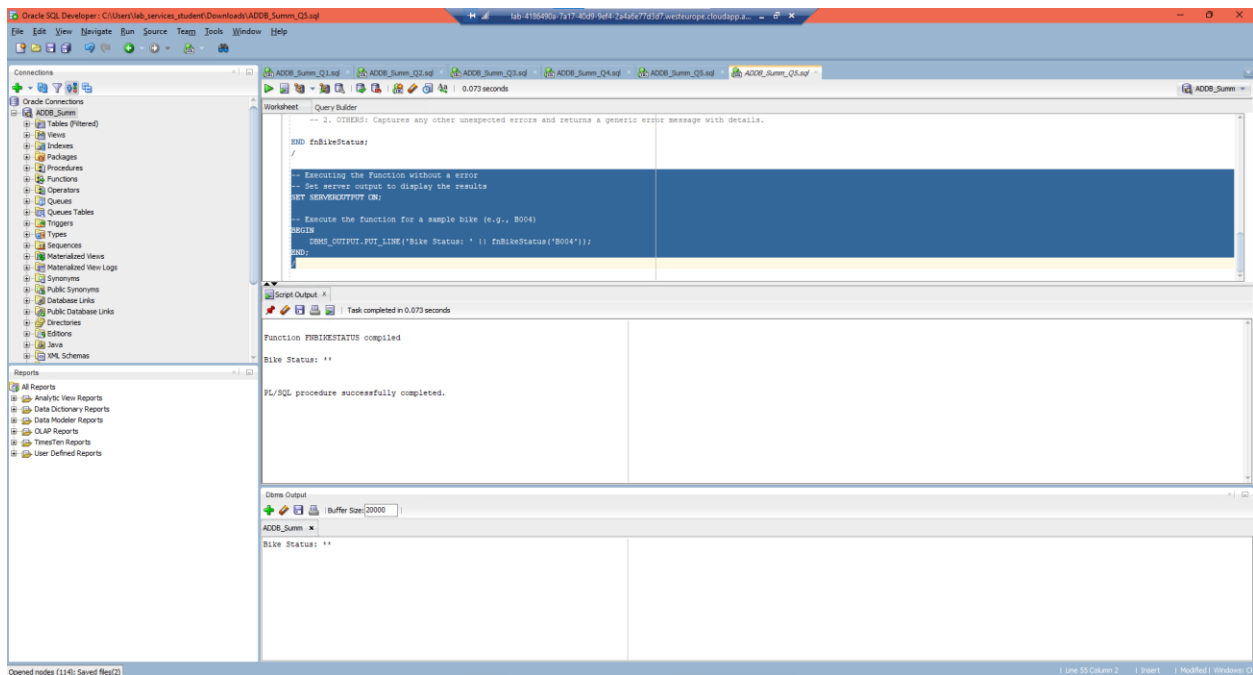
BEGIN

DBMS_OUTPUT.PUT_LINE('Bike Status: ' || fnBikeStatus('B0004'));

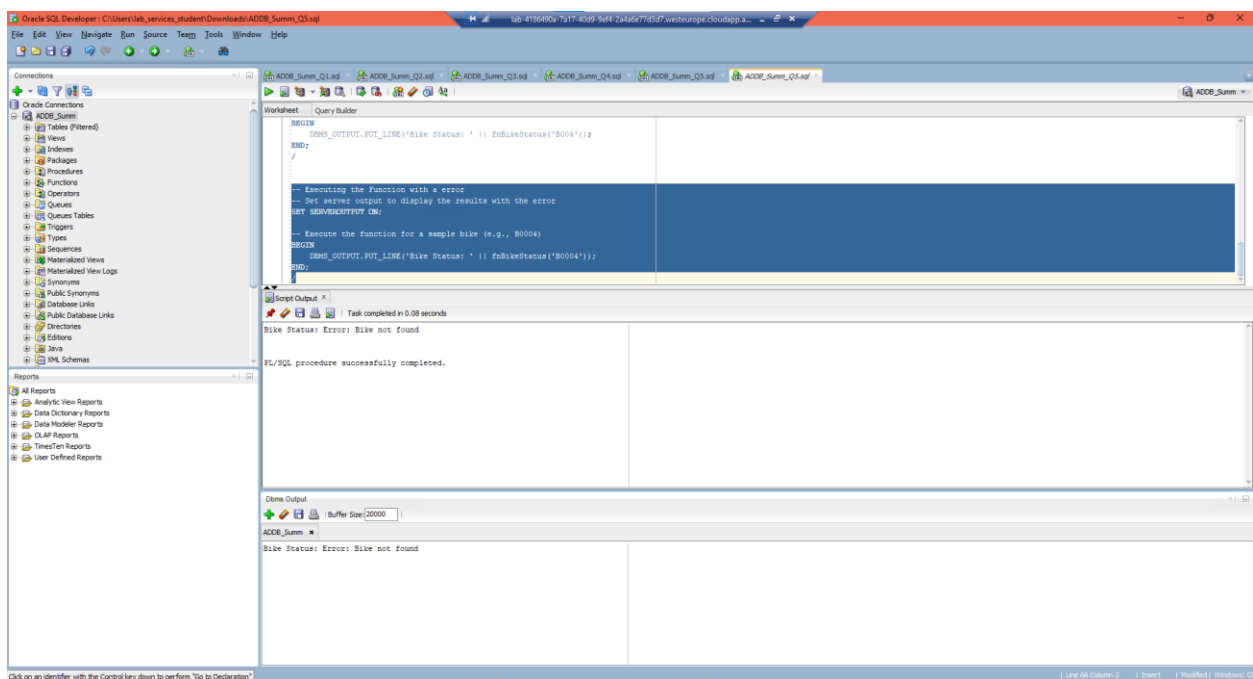
END;

/

Proof of Question 6 Without Error Message Handled (Please Zoom in to see image clearer)



Proof of Question 6 With Error Message Handled (Please Zoom in to see image clearer)



Question 7

-- Enable output to display the results

SET SERVEROUTPUT ON;

-- PL/SQL block to generate the bike report using IF statements for status

BEGIN

-- Loop through all the bikes and print details along with their status

FOR rec IN (

 SELECT B.BIKE_ID, B.BIKE_TYPE, B.MANUFACTURER, DON.VALUE

 FROM DONATION DON

 JOIN BIKE B ON DON.BIKE_ID = B.BIKE_ID

)

LOOP

-- Display bike details

DBMS_OUTPUT.PUT_LINE('BIKE ID: ' || rec.BIKE_ID);

DBMS_OUTPUT.PUT_LINE('BIKE TYPE: ' || rec.BIKE_TYPE);

DBMS_OUTPUT.PUT_LINE('MANUFACTURER: ' || rec.MANUFACTURER);

DBMS_OUTPUT.PUT_LINE('VALUE: R' || TO_CHAR(rec.VALUE, '999G999D00'));

-- Determine bike status using IF...ELSIF

IF rec.VALUE BETWEEN 0 AND 1500 THEN

 DBMS_OUTPUT.PUT_LINE('STATUS: *'); -- 1-star

ELSIF rec.VALUE > 1500 AND rec.VALUE <= 3000 THEN

 DBMS_OUTPUT.PUT_LINE('STATUS: **'); -- 2-stars

ELSIF rec.VALUE > 3000 THEN

 DBMS_OUTPUT.PUT_LINE('STATUS: ***'); -- 3-stars

END IF;

-- Add a separator for clarity

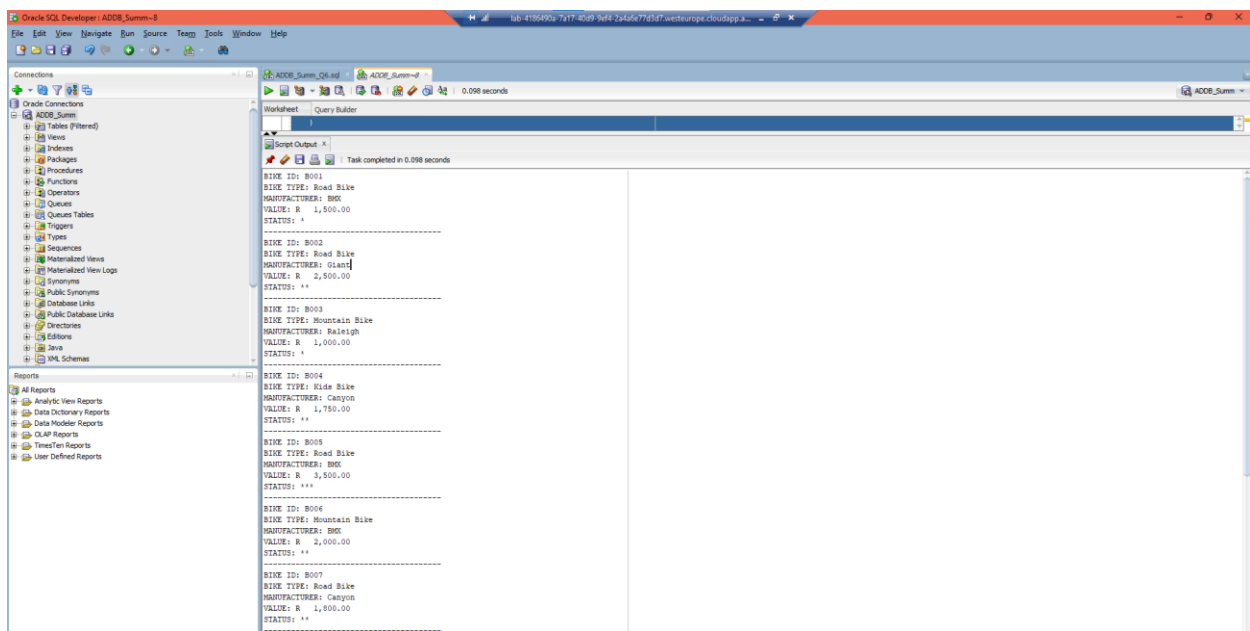
DBMS_OUTPUT.PUT_LINE('-----');

END LOOP;

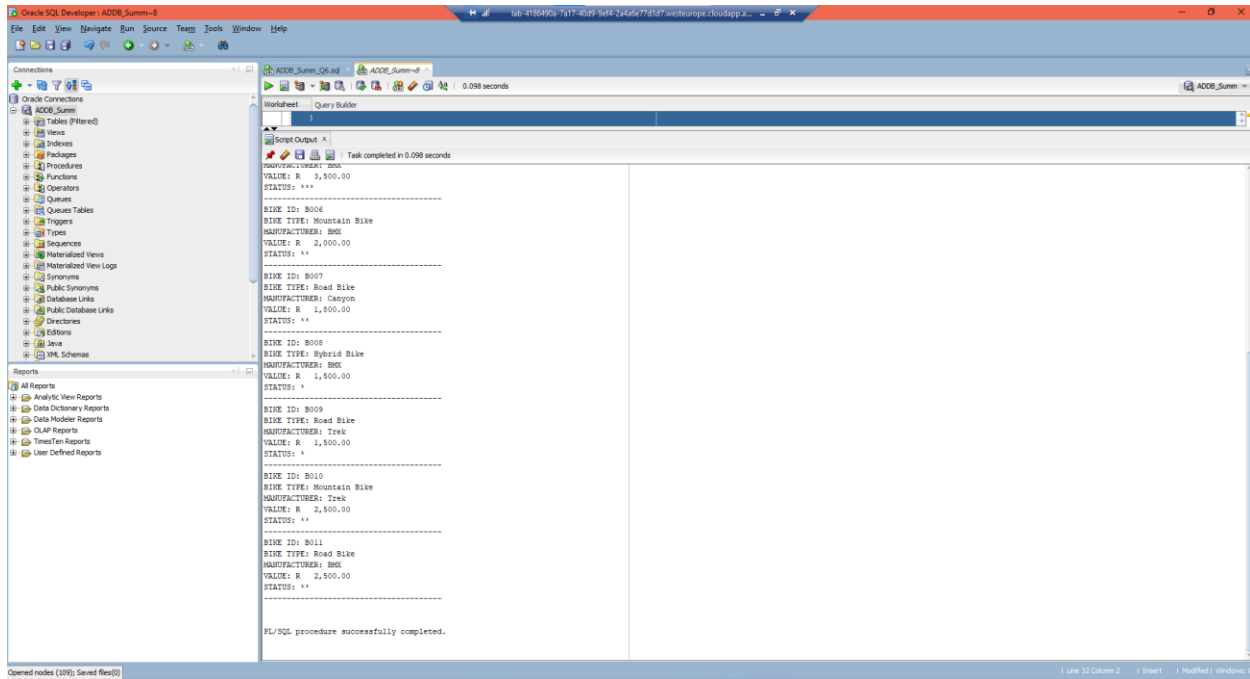
END;

/

Proof of Question 7 Part 1 (Please Zoom in to see image clearer)



Proof of Question 7 Part 2 (Please Zoom in to see image clearer)



Question 8

-- Generate a report using a CASE statement to assign the bike status

SELECT

B.BIKE_ID,

B.BIKE_TYPE,

B.MANUFACTURER,

'R' || TO_CHAR(DON.VALUE, '999G999D00') AS BIKE_VALUE,

CASE

-- 1-star status for bikes valued between 0 and 1500

WHEN DON.VALUE BETWEEN 0 AND 1500 THEN '*'

-- 2-star status for bikes valued between 1501 and 3000

WHEN DON.VALUE > 1500 AND DON.VALUE <= 3000 THEN '**'

-- 3-star status for bikes valued above 3000

WHEN DON.VALUE > 3000 THEN '***'

END AS STATUS

FROM

DONATION DON

JOIN

BIKE B ON DON.BIKE_ID = B.BIKE_ID;

Proof of Question 8 (Please Zoom in to see image clearer)

The screenshot displays the Oracle SQL Developer interface. The main window shows a SQL query in the Query Builder, which is designed to generate a report using a CASE statement to assign a bike status based on its DOH value. The query is as follows:

```
-- Generate a report using a CASE statement to assign the bike status
SELECT
  B.BIKE_ID,
  B.BIKE_TYPE,
  B.MANUFACTURER,
  "R" || TO_CHAR(DOH.VALUE, '9999999000') AS BIKE_VALUE,
  CASE
    -- 1-star status for bikes valued between 0 and 1500
    WHEN DOH.VALUE BETWEEN 0 AND 1500 THEN ***
    -- 2-star status for bikes valued between 1501 and 3000
    WHEN DOH.VALUE > 1500 AND DOH.VALUE <= 3000 THEN ***
    -- 3-star status for bikes valued above 3000
    WHEN DOH.VALUE > 3000 THEN ****
  END AS STATUS
FROM
  BIKES
  DOHATION DOH
  BIKE B ON DOH.BIKE_ID = B.BIKE_ID;
```

The Query Results pane shows the output of the query, displaying 11 rows of data. The columns are BIKE_ID, BIKE_TYPE, MANUFACTURER, BIKE_VALUE, and STATUS. The data is as follows:

BIKE_ID	BIKE_TYPE	MANUFACTURER	BIKE_VALUE	STATUS
1 B001	Road Bike	BMC	R 1,500.00 *	
2 B002	Road Bike	Giant	R 2,500.00 **	
3 B003	Mountain Bike	Raleigh	R 1,000.00 *	
4 B004	Kids Bike	Canyon	R 1,750.00 **	
5 B005	Road Bike	BMC	R 3,500.00 ***	
6 B006	Mountain Bike	BMC	R 2,000.00 **	
7 B007	Road Bike	Canyon	R 1,800.00 **	
8 B008	Hybrid Bike	BMC	R 1,900.00 *	
9 B009	Road Bike	Trek	R 1,500.00 *	
10 B010	Mountain Bike	Trek	R 2,800.00 **	
11 B011	Road Bike	BMC	R 2,500.00 **	

Question 9

-- Task 9.1: Trigger to prevent deletions from the DONATION table

```
CREATE OR REPLACE TRIGGER trg_prevent_donation_delete
BEFORE DELETE ON DONATION
FOR EACH ROW
BEGIN
    -- Raise an error if there is an attempt to delete a record from the DONATION table
    RAISE_APPLICATION_ERROR(-20001, 'Deletion from the DONATION table is not allowed.');
```

END trg_prevent_donation_delete;

/

-- Test for Trigger trg_prevent_donation_delete

-- This DELETE statement should fail and raise an error.

```
BEGIN
    DELETE FROM DONATION WHERE DONATION_ID = 1;
END;
```

/

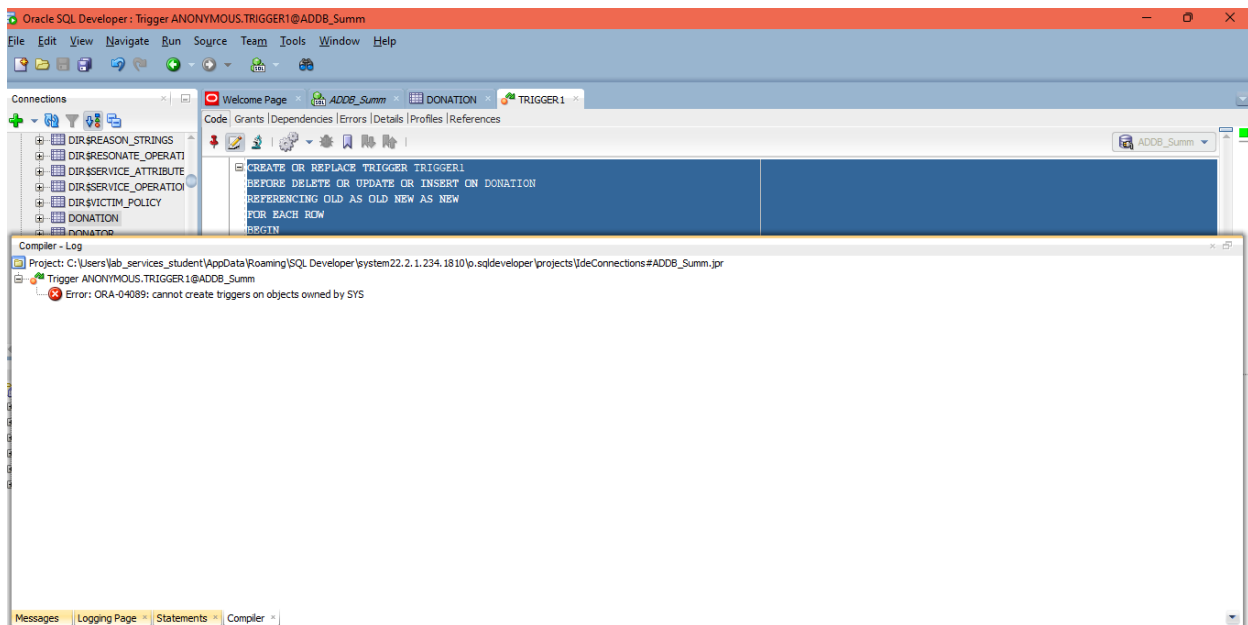
-- Task 9.2: Trigger to ensure a valid bike value on update in the DONATION table

```
CREATE OR REPLACE TRIGGER trg_validate_donation_value
BEFORE UPDATE OF VALUE ON DONATION
FOR EACH ROW
BEGIN
    -- Check if the new bike value is greater than 0
    IF :NEW.VALUE <= 0 THEN
        -- Raise an error if the new value is invalid (0 or negative)
        RAISE_APPLICATION_ERROR(-20002, 'Bike value must be greater than 0.');
```

```
END IF;
END trg_validate_donation_value;
/

-- Test for Trigger trg_validate_donation_value
-- This UPDATE statement should fail and raise an error if the new value is 0 or negative.
BEGIN
    UPDATE DONATION SET VALUE = -100 WHERE DONATION_ID = 1;
END;
/
```

Proof of Error I am getting with regards to Question 9 (Please Zoom in to see image clearer)



Reason for this error:

Oracle prevents users from creating triggers on tables that belong to the SYS schema, which is why I'm getting the problem ORA-04089: cannot construct triggers on objects owned by SYS. Oracle's default administration schema, known as the SYS schema, includes system-related tables and objects that are necessary for the database to function. So therefore, I am receiving this error as I cant create triggers that belong to a SYS Schema.

Question 10

1. Confidentiality

Only those with permission can access sensitive data thanks to confidentiality. The BikesRUs database can be secured as follows:

a. Roles and Privileges of Users

- Limit access to only the information required for each user's function in accordance with the principle of least privilege (PoLP). Volunteers should, for example, only view the tables pertaining to donations and not financial information.

- **Example SQL Code:**

```
-- Grant specific privileges to volunteers
```

```
GRANT SELECT, INSERT ON Donation TO volunteer_role;
```

```
REVOKE UPDATE, DELETE ON Donation FROM volunteer_role;
```

b. Encryption of Data

- Encryption in Transit: Use SSL/TLS to encrypt all network communications to guard against illegal parties intercepting them. Oracle Advanced Security offers configuration options for these.
- Resting Encryption: For sensitive data columns like donor contact details, enable Transparent Data Encryption (TDE).

- **Example Command for TDE:**

```
-- Encrypt sensitive column
```

```
ALTER TABLE Donation MODIFY (Value ENCRYPT USING 'AES256');
```

c. Masking Data

- When working in non-production settings, apply data masking techniques. This guarantees that development personnel and volunteers will not have access to actual donor data in test databases.
- Tool: For testing or development, the Oracle Data Masking and Subsetting tool can create masked duplicates of production data.

2. Integrity

reserving correct and precise data throughout its existence is the core goal of integrity.

a. Constraints and Triggers

- To preserve referential integrity, apply primary and foreign key constraints to every relational table. For instance, there should be a foreign key constraint connecting Donor_ID in the Donation field to the Donor table.
- As seen in Question 9, when we enforce a positive value for donations, use triggers to evaluate data on update or insert.

b. Audit Trails

- Allow auditing to document activities on sensitive tables, such Volunteer and Donation. This guarantees responsibility and makes it possible to track improvements.
- Example SQL Code:
-- Enable auditing on the Donation table for INSERT and DELETE operations
AUDIT INSERT, DELETE ON Donation;
- Tool: Database Firewall and Oracle Audit Vault offer centralized audits and can notify administrators of anomalous activity.

c. Data Validation and Checks

- To avoid SQL injection and other types of corruption, make sure that all user input is verified.
- For instance, use a trigger in Question 9 to enforce checks on the bike value column to make sure it is positive.

3. Accessibility

Availability guarantees that authorized users may access data as needed.

a. Recovery and Backup

- Safeguarding against data loss requires routinely planned backups and recovery processes.
- Tool: To automate backups and guarantee quick recovery, use Oracle Recovery Manager (RMAN).

b. Failover and Redundancy

- Oracle Data Guard can be used to replicate the database to a standby server and create a high-availability environment. In the event of a hardware breakdown, this permits a smooth failover.
- **Example:**
-- Configure Data Guard environment
`ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=standby_db';`

c. Management of Resources

- To avoid resource monopolization by any one user, restrict resource consumption per user.
- Tool: Oracle Resource Manager can assist in efficiently allocating CPU resources so that lower-priority workloads do not impact high-priority operations.

4. Additional Measures for Security and Performance

a. Use of Views

- Implement views to restrict users from accessing entire tables. A view can provide only the necessary columns without exposing sensitive data.
- Example View:
`CREATE VIEW volunteer_view AS
SELECT DONATION_ID, BIKE_ID, DONATION_DATE
FROM Donation
WHERE VOLUNTEER_ID = 'vol105';`

b. Frequent updates and patching

- Updating the database software is essential for preventing risks. Plan to update frequently and keep an eye on Oracle Security Alerts.

In conclusion, a mix of user access control, encryption, data integrity procedures, frequent backups, and availability measures is needed to safeguard BikesRUs's data. We can guarantee that the database is safe, dependable, and accessible by utilizing Oracle's built-in tools and security procedures, which will support BikesRUs's purpose while protecting its data assets.

References Used in this Summative Assessment

(Feuerstein, 2020) (GeekforGeek, 2023) (MSFT, et al., 2024) (W3Schools, n.d.) (To, et al., 2024)
 (Programiz, n.d.) (Wolhuter, 2024) (Wolhuter, 2024) (Wolhuter, 2024) (Wolhuter, 2024) (Wolhuter,
 2024) (Wolhuter, 2024) (Wolhuter, 2024) (Wolhuter, 2024) (Wolhuter, 2024) (Wolhuter, 2024)
 (Wolhuter, 2024) (Wolhuter, 2024) (Wolhuter, 2024) (Wolhuter, 2024) (Wolhuter, 2024) (Wolhuter,
 2024) (ChatGPT, 2024) (ChatGPT, 2024) (ChatGPT, 2024) (satori, n.d.) (Loshin, 2022) (Charles Sturt
 University, n.d.) (Secoda, 2024) (IBM, 2023) (Microsoft, n.d.) (Microsoft, n.d.)

References

Charles Sturt University, n.d. *ASC202 Research Skills Guide: Use Journal Databases*. [Online]

Available at:

[https://libguides.csu.edu.au/c.php?g=220102&p=6596671#:~:text=A%20journal%20database%20is%20an,of%20publication%2C%20title%2C%20etc\)](https://libguides.csu.edu.au/c.php?g=220102&p=6596671#:~:text=A%20journal%20database%20is%20an,of%20publication%2C%20title%2C%20etc))

[Accessed 28 October 2024].

ChatGPT, 2024. *BikeRUs donation table error*. [Online]

Available at: <https://chatgpt.com/c/671ad018-e334-8012-8fe4-4f84a5d2018b5>

[Accessed 29 October 2024].

ChatGPT, 2024. *How to drop Tables/ View Tables/ Table insert error*. [Online]

Available at: <https://chatgpt.com/c/671839c0-30e8-8012-b2dd-a4203a3fa07e>

[Accessed 29 October 2024].

ChatGPT, 2024. *Oracle Trigger Creation Error*. [Online]

Available at: <https://chatgpt.com/c/6720c626-11c0-8012-839f-5f5f2c4b6244>

[Accessed 29 October 2024].

Feuerstein, S., 2020. *Building with blocks in PL/SQL*. [Online]

Available at: <https://blogs.oracle.com/connect/post/building-with-blocks>

[Accessed 29 October 2024].

GeekforGeek, 2023. *Blocks in PL/SQL*. [Online]

Available at: <https://www.geeksforgeeks.org/blocks-in-pl-sql/>

[Accessed 29 October 2024].

IBM, 2023. *12 Data Integrity Examples: Types, Industry Usage, and Risks*. [Online]

Available at: <https://www.ibm.com/think/topics/data->

integrity#:~:text=The%20main%20types%20of%20data,across%20data%20relationships%20in%20databases.

[Accessed 29 October 2024].

Loshin, P., 2022. *Open Web Application Security Project (OWASP)*. [Online]

Available at: <https://www.techtarget.com/searchsoftwarequality/definition/OWASP>

[Accessed 28 October 2024].

Microsoft, n.d. *Learn the structure of an Access database*. [Online]
Available at: <https://support.microsoft.com/en-us/office/learn-the-structure-of-an-access-database-001a5c05-3fea-48f1-90a0-cccaa57ba4af>
[Accessed 29 October 2024].

Microsoft, n.d. *What is database security?*. [Online]
Available at: <https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-database-security>
[Accessed 29 October 2024].

MSFT, R. W., Buck, A. & Roth, J., 2024. *BEGIN...END (Transact-SQL)*. [Online]
Available at: <https://learn.microsoft.com/en-us/sql/t-sql/language-elements/begin-end-transact-sql?view=sql-server-ver16>
[Accessed 29 October 2024].

Programiz, n.d. *Getting Started with SQL*. [Online]
Available at: <https://www.programiz.com/sql/getting-started>
[Accessed 29 October 2024].

satori, n.d. *Top 10 Database Security Best Practices*. [Online]
Available at: <https://satoricyber.com/database-security/top-10-database-security-best-practices/>
[Accessed 28 October 2024].

Secoda, 2024. *What is Data Confidentiality?*. [Online]
Available at: <https://www.secoda.co/glossary/data-confidentiality#:~:text=Data%20confidentiality%20typically%20refers%20to,driver's%20license%20numbers%20and%20addresses.>
[Accessed 29 October 2024].

To, V., MSFT, R. W. & Millsap, R., 2024. *SELECT (Transact-SQL)*. [Online]
Available at: <https://learn.microsoft.com/en-us/sql/t-sql/queries/select-transact-sql?view=sql-server-ver16>
[Accessed 29 October 2024].

W3Schools, n.d. *SQL SELECT Statement*. [Online]
Available at: https://www.w3schools.com/sql/sql_select.asp
[Accessed 29 October 2024].

Wolhuter, R., 2024. *ADDB_7311_Intro&LU1_Theme1And2 (Introduction)*, Cape Town: Wolhuter, Riko.

Wolhuter, R., 2024. *ADDB_7311_Theme11&12 (PL/SQL blocks)*, Cape Town: Riko Wolhuter.

Wolhuter, R., 2024. *ADDB_7311_Theme3&4 (SQL Developer + SQL Plus)*, Cape Town: Riko Wolhuter.

Wolhuter, R., 2024. *ADDB_7311_Theme5&6 (SQL basics, TCL + Exercise)*, Cape Town: Riko Wolhuter.

Wolhuter, R., 2024. *ADDB_7311_Theme7&8 (DBCA + Database Memory + TableSpace Management + Terminating a Session)*, Cape Town: Riko Wolhuter.

Wolhuter, R., 2024. *Control Structures + Cursors*, Cape Town: Riko Wolhuter.

Wolhuter, R., 2024. *Database security*, Cape Town: Riko Wolhuter.

Wolhuter, R., 2024. *Exception handling PL/SQL*, Cape Town: Riko Wolhuter.

Wolhuter, R., 2024. *Exception handling PL/SQL #2*, Cape Town: Riko Wolhuter.

Wolhuter, R., 2024. *Exception handling PL/SQL #3 + Subprograms*, Cape Town: Riko Wolhuter.

Wolhuter, R., 2024. *External Procedures - Java/C*, Cape Town: Riko Wolhuter.

Wolhuter, R., 2024. *Indexes, sequences, views, and synonyms*, Cape Town: Riko Wolhuter.

Wolhuter, R., 2024. *PL/SQL blocks practical introduction*, Cape Town: Riko Wolhuter.

Wolhuter, R., 2024. *PL/SQL Lexical Units*, Cape Town: Riko Wolhuter.

Wolhuter, R., 2024. *PL/SQL Triggers*, Cape Town: Riko Wolhuter.

Wolhuter, R., 2024. *Procedures and Functions - PL/SQL*, Cape Town: Riko Wolhuter.