Nonjabulo Mathenjwa

ST10077892

INSY7213

Summative Project

10 November 2025

Question 1

A screenshot of a computer

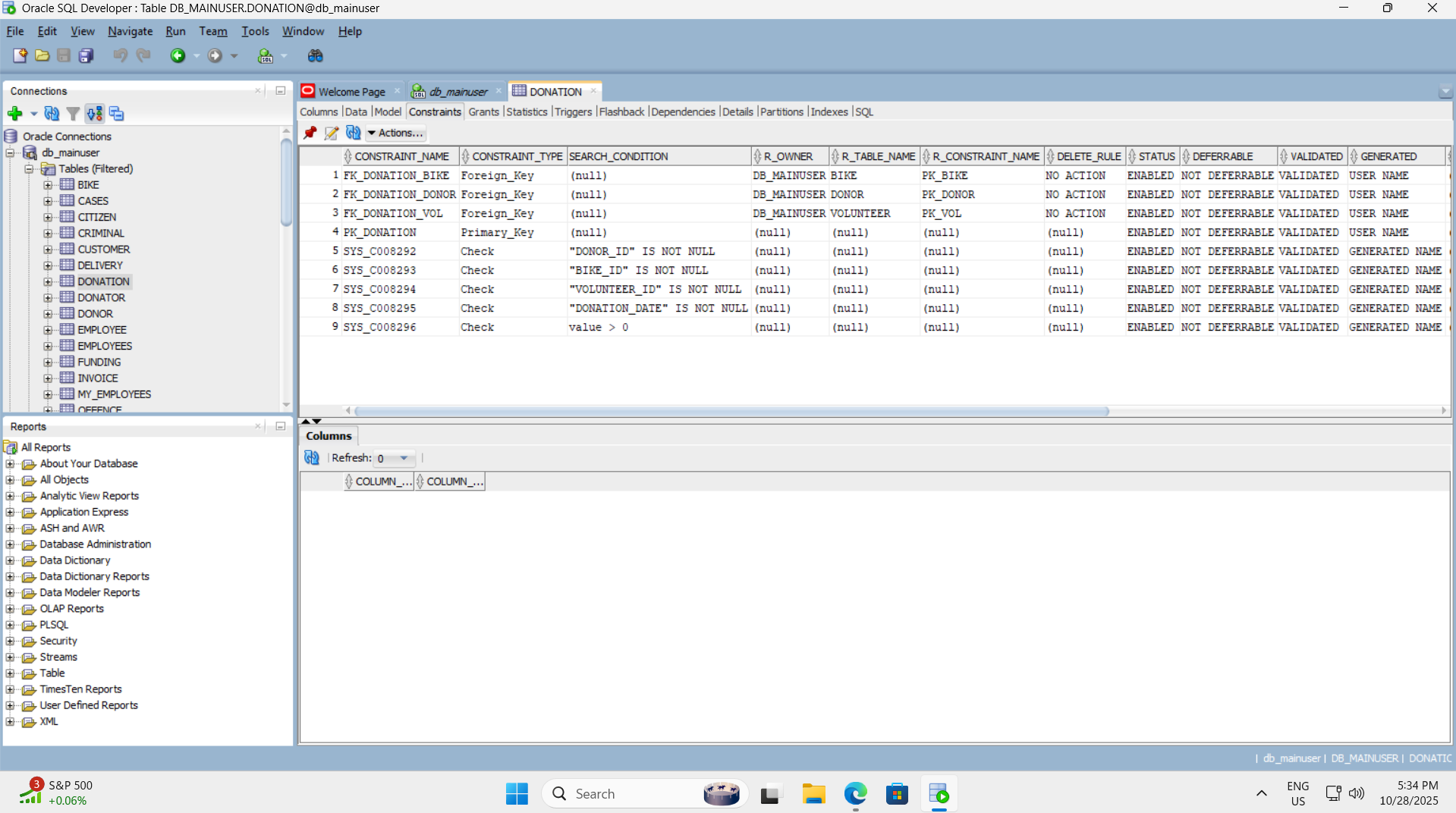
AI-generated content may be incorrect.

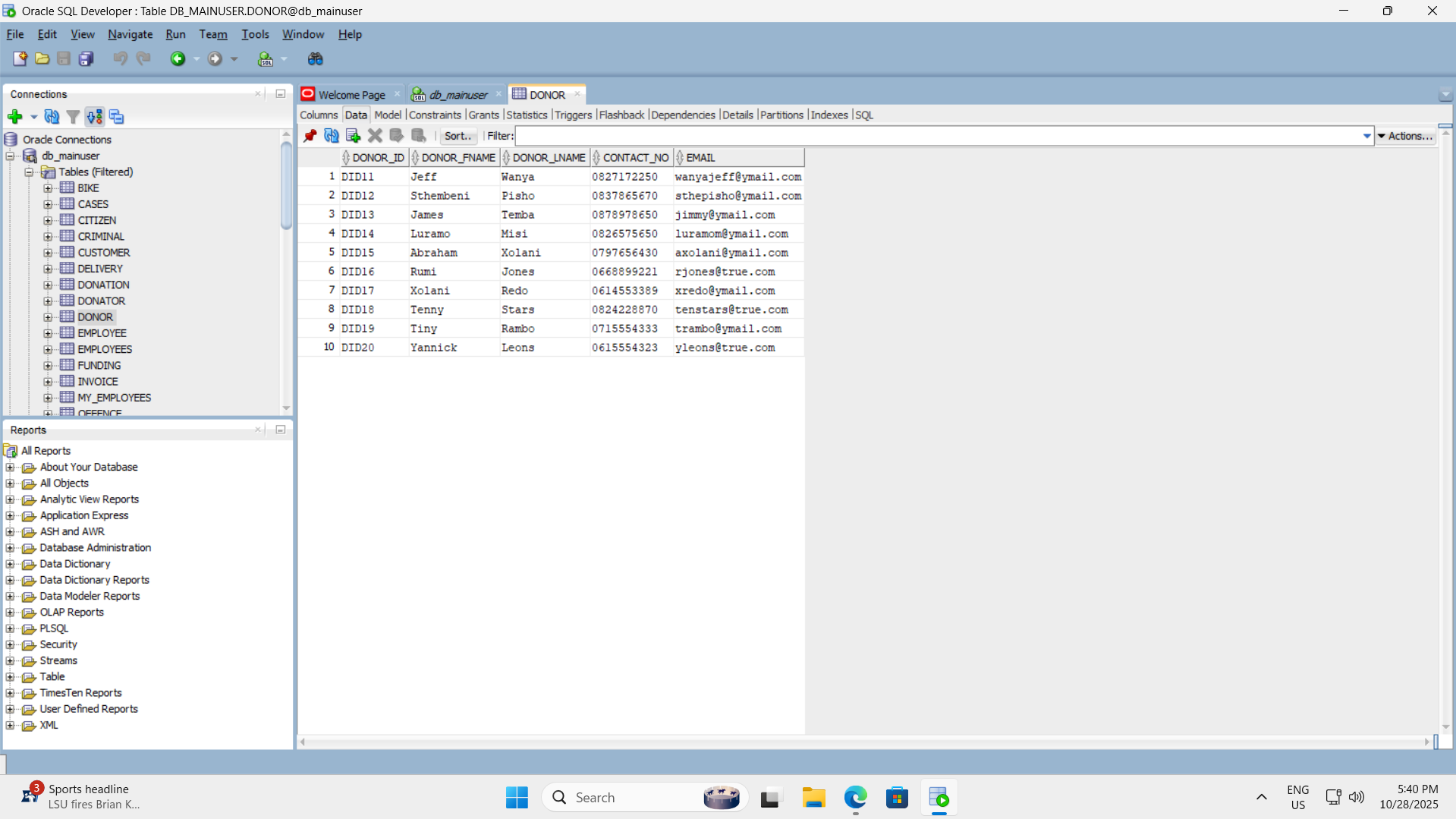
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.





A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

------------------------------------------------------------

-- TABLE: VOLUNTEER

------------------------------------------------------------

CREATE TABLE Volunteer (

vol\_id VARCHAR2(10) CONSTRAINT pk\_vol PRIMARY KEY,

vol\_fname VARCHAR2(30) NOT NULL,

vol\_sname VARCHAR2(30) NOT NULL,

contact VARCHAR2(15) NOT NULL,

address VARCHAR2(50),

email VARCHAR2(50) UNIQUE

);

------------------------------------------------------------

-- TABLE: DONOR

------------------------------------------------------------

CREATE TABLE Donor (

donor\_id VARCHAR2(10) CONSTRAINT pk\_donor PRIMARY KEY,

donor\_fname VARCHAR2(30) NOT NULL,

donor\_lname VARCHAR2(30) NOT NULL,

contact\_no VARCHAR2(15) NOT NULL,

email VARCHAR2(50) UNIQUE

);

------------------------------------------------------------

-- TABLE: BIKE

------------------------------------------------------------

CREATE TABLE Bike (

bike\_id VARCHAR2(10) CONSTRAINT pk\_bike PRIMARY KEY,

description VARCHAR2(50) NOT NULL,

bike\_type VARCHAR2(30) NOT NULL,

manufacturer VARCHAR2(30) NOT NULL

);

------------------------------------------------------------

-- TABLE: DONATION

------------------------------------------------------------

CREATE TABLE Donation (

donation\_id NUMBER CONSTRAINT pk\_donation 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\_donation\_donor FOREIGN KEY (donor\_id) REFERENCES Donor(donor\_id),

CONSTRAINT fk\_donation\_bike FOREIGN KEY (bike\_id) REFERENCES Bike(bike\_id),

CONSTRAINT fk\_donation\_vol FOREIGN KEY (volunteer\_id) REFERENCES Volunteer(vol\_id)

);

------------------------------------------------------------

-- INSERT DATA: VOLUNTEER

------------------------------------------------------------

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: DONOR

------------------------------------------------------------

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: BIKE

------------------------------------------------------------

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: DONATION

------------------------------------------------------------

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'));

------------------------------------------------------------

-- COMMIT ALL CHANGES

------------------------------------------------------------

COMMIT;

------------------------------------------------------------

-- VERIFY DATA

------------------------------------------------------------

SELECT 'VOLUNTEER' AS TABLE\_NAME, COUNT(\*) AS RECORDS FROM Volunteer

UNION

SELECT 'DONOR', COUNT(\*) FROM Donor

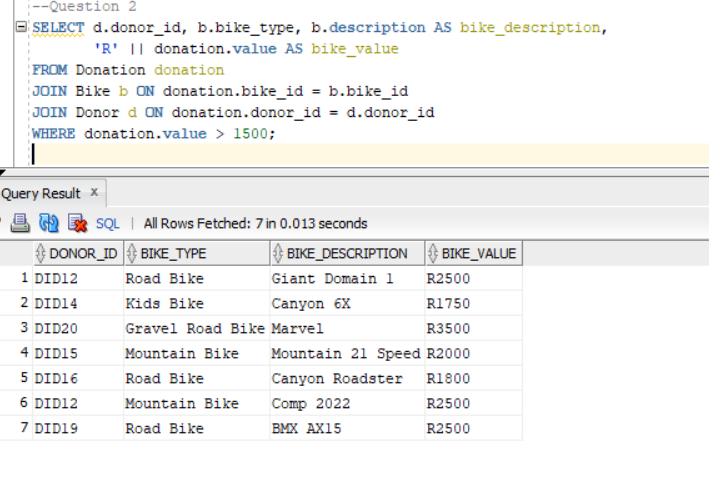
UNION

SELECT 'BIKE', COUNT(\*) FROM Bike

UNION

SELECT 'DONATION', COUNT(\*) FROM Donation;

Question 2



--Question 2

SELECT d.donor\_id, b.bike\_type, b.description AS bike\_description,

'R' || donation.value AS bike\_value

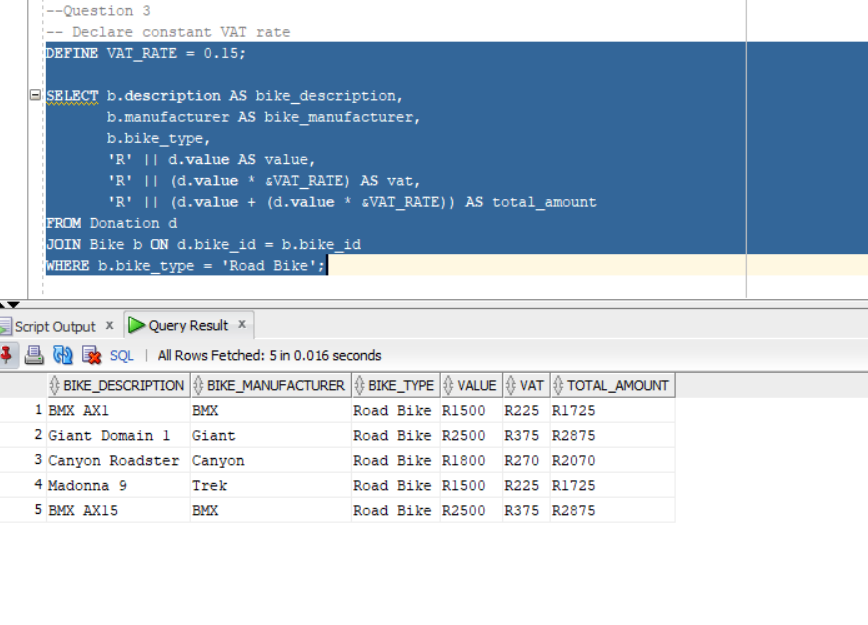
FROM Donation donation

JOIN Bike b ON donation.bike\_id = b.bike\_id

JOIN Donor d ON donation.donor\_id = d.donor\_id

WHERE donation.value > 1500;

Question 3



--Question 3

-- Declare constant VAT rate

DEFINE VAT\_RATE = 0.15;

SELECT b.description AS bike\_description,

b.manufacturer AS bike\_manufacturer,

b.bike\_type,

'R' || d.value AS value,

'R' || (d.value \* &VAT\_RATE) AS vat,

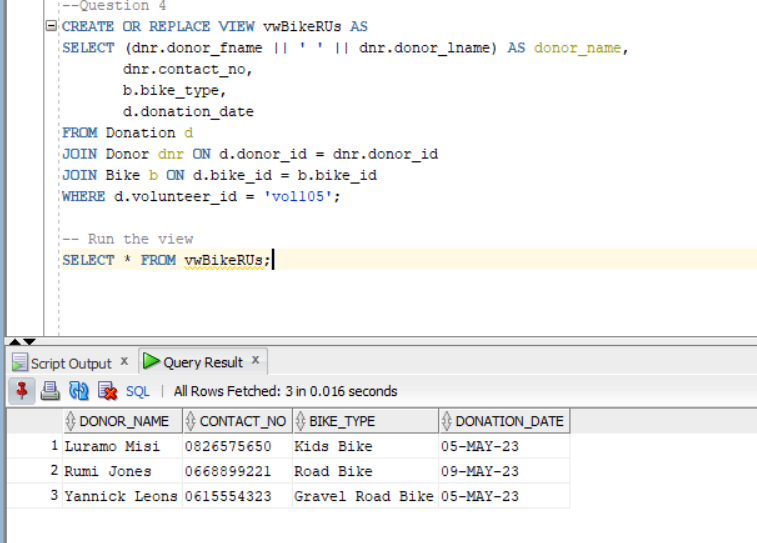
'R' || (d.value + (d.value \* &VAT\_RATE)) AS total\_amount

FROM Donation d

JOIN Bike b ON d.bike\_id = b.bike\_id

WHERE b.bike\_type = 'Road Bike';

Question 4



--Question 4

CREATE OR REPLACE VIEW vwBikeRUs AS

SELECT (dnr.donor\_fname || ' ' || dnr.donor\_lname) AS donor\_name,

dnr.contact\_no,

b.bike\_type,

d.donation\_date

FROM Donation d

JOIN Donor dnr ON d.donor\_id = dnr.donor\_id

JOIN Bike b ON d.bike\_id = b.bike\_id

WHERE d.volunteer\_id = 'vol105';

-- Run the view

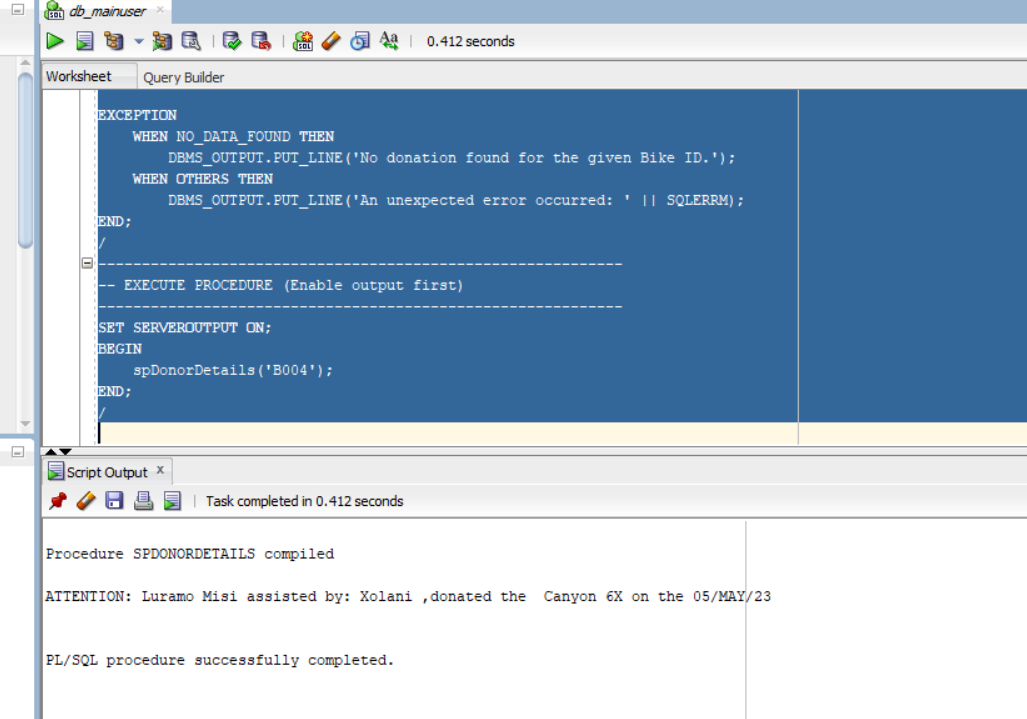
SELECT \* FROM vwBikeRUs;

Benefits of using a view:

1. Simplifies complex queries for users, improving accessibility.

2. Provides data security by restricting access to sensitive columns.

Question 5



-- =========================================================

-- QUESTION 5: PROCEDURE spDonorDetails

-- Displays donor’s name, volunteer’s first name, bike description and date

-- =========================================================

CREATE OR REPLACE PROCEDURE spDonorDetails(p\_bike\_id IN VARCHAR2) AS

v\_donor\_name VARCHAR2(50);

v\_contact\_no VARCHAR2(15);

v\_vol\_name VARCHAR2(50);

v\_bike\_desc VARCHAR2(50);

v\_date DATE;

BEGIN

-- Select donor, volunteer, bike, and date for the given Bike ID

SELECT (dnr.donor\_fname || ' ' || dnr.donor\_lname),

dnr.contact\_no,

v.vol\_fname,

b.description,

d.donation\_date

INTO v\_donor\_name, v\_contact\_no, v\_vol\_name, v\_bike\_desc, v\_date

FROM Donation d

JOIN Donor dnr ON d.donor\_id = dnr.donor\_id

JOIN Volunteer v ON d.volunteer\_id = v.vol\_id

JOIN Bike b ON d.bike\_id = b.bike\_id

WHERE d.bike\_id = p\_bike\_id;

-- Exact sample output format from the paper

DBMS\_OUTPUT.PUT\_LINE(

'ATTENTION: ' || v\_donor\_name ||

' assisted by: ' || v\_vol\_name ||

' ,donated the ' || v\_bike\_desc ||

' on the ' || TO\_CHAR(v\_date, 'DD/MON/YY')

);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No donation found for the given Bike ID.');

WHEN OTHERS THEN

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

END;

/

------------------------------------------------------------

-- EXECUTE PROCEDURE (Enable output first)

------------------------------------------------------------

SET SERVEROUTPUT ON;

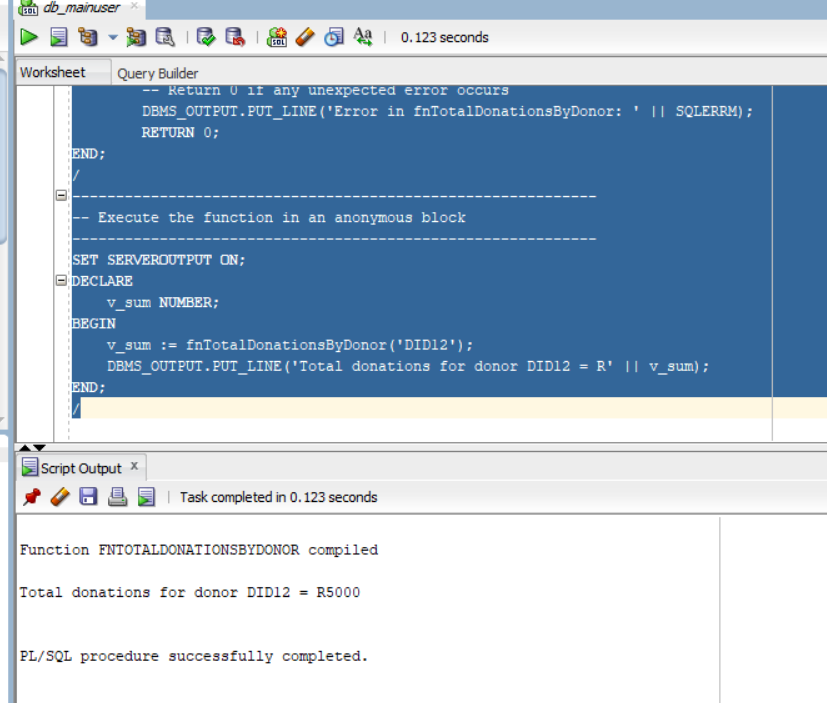
BEGIN

spDonorDetails('B004');

END;

/

Question 6



-- =========================================================

-- QUESTION 6: FUNCTION TO CALCULATE TOTAL DONATION VALUE BY DONOR

-- =========================================================

CREATE OR REPLACE FUNCTION fnTotalDonationsByDonor(p\_donor\_id IN VARCHAR2)

RETURN NUMBER

AS

v\_total NUMBER := 0;

BEGIN

-- Sum all donation values for the given donor

SELECT NVL(SUM(value),0)

INTO v\_total

FROM Donation

WHERE donor\_id = p\_donor\_id;

RETURN v\_total;

EXCEPTION

WHEN OTHERS THEN

-- Return 0 if any unexpected error occurs

DBMS\_OUTPUT.PUT\_LINE('Error in fnTotalDonationsByDonor: ' || SQLERRM);

RETURN 0;

END;

/

------------------------------------------------------------

-- Execute the function in an anonymous block

------------------------------------------------------------

SET SERVEROUTPUT ON;

DECLARE

v\_sum NUMBER;

BEGIN

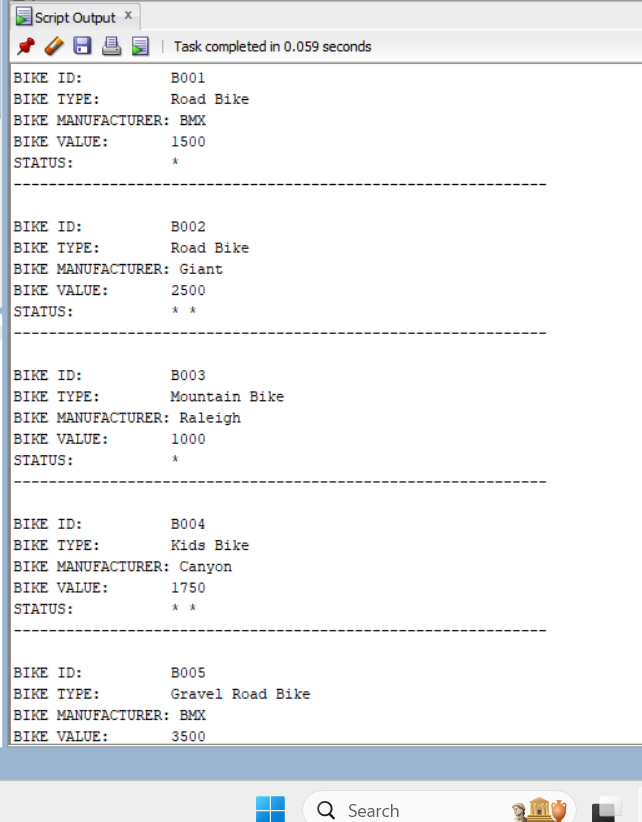
v\_sum := fnTotalDonationsByDonor('DID12');

DBMS\_OUTPUT.PUT\_LINE('Total donations for donor DID12 = R' || v\_sum);

END;

/

Question 7



-- =========================================================

-- QUESTION 7: REPORT USING IF...ELSIF TO ASSIGN STAR STATUS

-- =========================================================

SET SERVEROUTPUT ON;

DECLARE

CURSOR c\_bikes IS

SELECT b.bike\_id,

b.bike\_type,

b.manufacturer,

d.value

FROM Bike b

JOIN Donation d ON b.bike\_id = d.bike\_id

ORDER BY b.bike\_id;

v\_status VARCHAR2(5);

BEGIN

FOR rec IN c\_bikes LOOP

-- Determine the status using IF / ELSIF

IF rec.value <= 1500 THEN

v\_status := '\*';

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

v\_status := '\* \*';

ELSE

v\_status := '\* \* \*';

END IF;

-- Print the formatted report to match the required layout

DBMS\_OUTPUT.PUT\_LINE('BIKE ID: ' || rec.bike\_id);

DBMS\_OUTPUT.PUT\_LINE('BIKE TYPE: ' || rec.bike\_type);

DBMS\_OUTPUT.PUT\_LINE('BIKE MANUFACTURER:' || ' ' || rec.manufacturer);

DBMS\_OUTPUT.PUT\_LINE('BIKE VALUE: ' || rec.value);

DBMS\_OUTPUT.PUT\_LINE('STATUS: ' || v\_status);

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

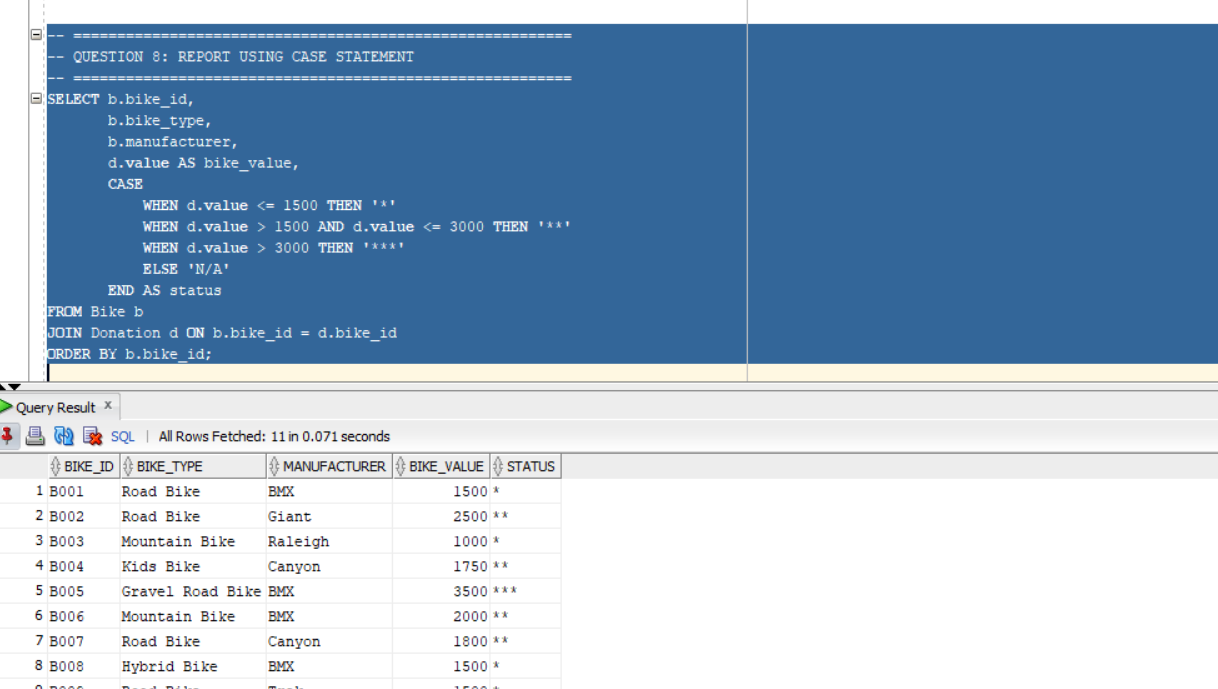
DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

END;

/

Question 8



-- =========================================================

-- QUESTION 8: REPORT USING CASE STATEMENT

-- =========================================================

SELECT b.bike\_id,

b.bike\_type,

b.manufacturer,

d.value AS bike\_value,

CASE

WHEN d.value <= 1500 THEN '\*'

WHEN d.value > 1500 AND d.value <= 3000 THEN '\*\*'

WHEN d.value > 3000 THEN '\*\*\*'

ELSE 'N/A'

END AS status

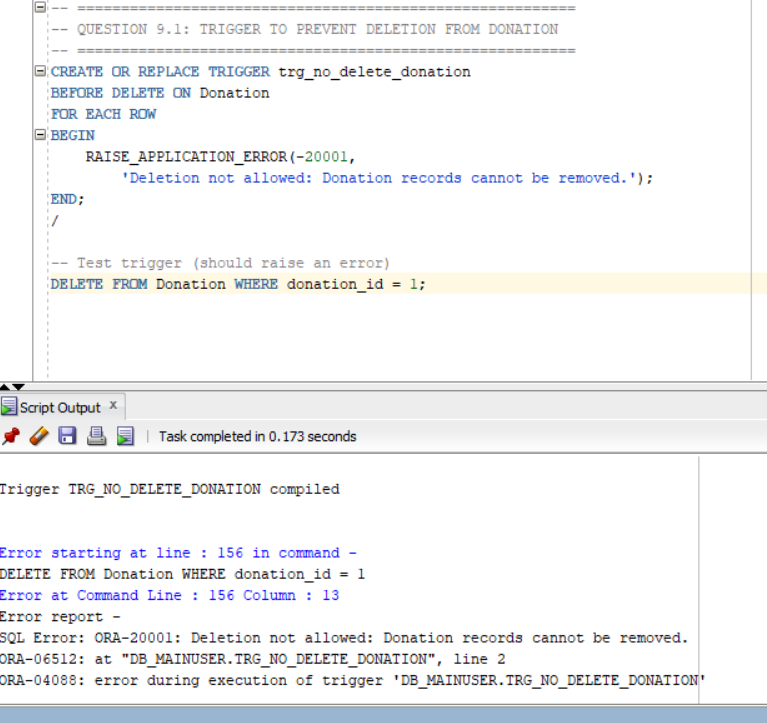
FROM Bike b

JOIN Donation d ON b.bike\_id = d.bike\_id

ORDER BY b.bike\_id;

Question 9

9.1



-- =========================================================

-- QUESTION 9.1: TRIGGER TO PREVENT DELETION FROM DONATION

-- =========================================================

CREATE OR REPLACE TRIGGER trg\_no\_delete\_donation

BEFORE DELETE ON Donation

FOR EACH ROW

BEGIN

RAISE\_APPLICATION\_ERROR(-20001,

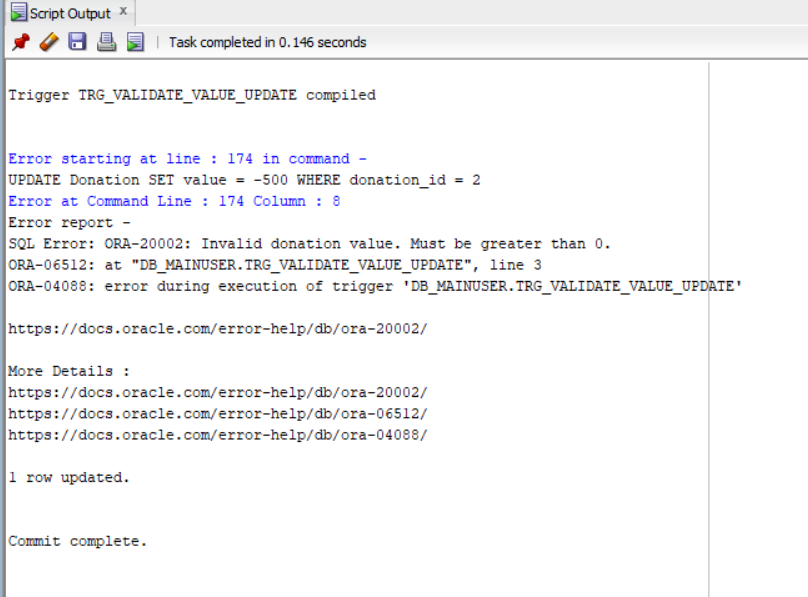
'Deletion not allowed: Donation records cannot be removed.');

END;

/

-- Test trigger (should raise an error)

DELETE FROM Donation WHERE donation\_id = 1;

9.2 

-- =========================================================

-- QUESTION 9.2: TRIGGER TO ENSURE VALID VALUE (>0) ON UPDATE

-- =========================================================

CREATE OR REPLACE TRIGGER trg\_validate\_value\_update

BEFORE UPDATE OF value ON Donation

FOR EACH ROW

BEGIN

IF :NEW.value <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20002,

'Invalid donation value. Must be greater than 0.');

END IF;

END;

/

-- Test invalid update (should fail)

UPDATE Donation SET value = -500 WHERE donation\_id = 2;

-- Test valid update (should succeed)

UPDATE Donation SET value = 3200 WHERE donation\_id = 2;

COMMIT;

Question 10

Maintaining BikesRUs Data Availability, Integrity, and Confidentiality  
  
Overview  
Sensitive community data, such as volunteer information, gift values, and donor contact details, is stored by BikesRUs. Maintaining public confidence and adhering to data-protection guidelines depend on protecting this information. This technical report describes how database architecture, access restrictions, and operational policies can be used to enforce confidentiality, integrity, and availability (CIA).  
  
1. Confidentiality: Protecting Personal Information  
  
Data can only be accessed by those who are authorized thanks to confidentiality.  
a) RBAC, or role-based access control  
Make separate database roles with the very minimum of rights, such as role\_admin, role\_sales, and role\_receive.

CREATE ROLE role\_sales;

GRANT SELECT ON Bike TO role\_sales;

GRANT SELECT, INSERT ON Donation TO role\_sales;

b) Password policies and user authentication  
Every volunteer who uses the system needs to have an Oracle user account with expiration policies and secure passwords.  
  
c) Cryptography  
  
At Rest: To safeguard donor tables on disk, use Oracle Transparent Data Encryption (TDE).  
  
In Transit: To prevent data and credentials from being sent in plain text, enable SSL/TLS for client-server communications.  
  
d) Masking Data  
Limit critical fields in reports to regular volunteers (e.g., only displaying partial phone numbers).  
  
e) Safe Backups  
To avoid disclosure in the event that hardware is taken, backups must be encrypted and kept off-site or on a secure Azure/AWS vault.

2. Integrity: Maintaining Correct and Trustworthy Data  
  
Integrity guarantees that data is accurate, consistent, and shielded from unwanted alteration.  
a) Limitations and Keys  
The PRIMARY KEY, FOREIGN KEY, and CHECK(value > 0) constraints that your schema already employs automatically guard against invalid relationships and negative contribution values.  
  
b) Business Rules and Triggers  
Integrity is directly protected by the triggers in Question 9, which mandate that no gift be removed and that modified values stay positive.  
  
c) Trails of Audits  
Turn on Oracle auditing or make a shadow table to record all modifications:

CREATE TABLE Donation\_Audit (

audit\_id NUMBER GENERATED BY DEFAULT AS IDENTITY,

donation\_id NUMBER,

changed\_by VARCHAR2(30),

old\_value NUMBER,

new\_value NUMBER,

change\_date TIMESTAMP DEFAULT SYSTIMESTAMP

);

A simple AFTER UPDATE trigger can insert into Donation\_Audit whenever donations are modified.

d) Hashing or Checksums  
Critical transaction data (value + date) can be hashed with DBMS\_CRYPTO.HASH to detect tampering.

e) Input Validation  
All front-end or script inputs must be validated before hitting the database, preventing SQL injection and bad data.

3. Availability: Making Data Always Available  
  
Availability ensures that systems and data are reachable.  
a) A strategy for backup and recovery  
Use RMAN to schedule incremental daily backups and complete weekly backups. To confirm recoverability, conduct restoration drills once a month.  
  
b) High Availability (HA)  
Install a cloud replica or standby database (such as an Azure DB replica or Oracle Data Guard). During outages, this enables rapid failover.  
  
c) Alerts and System Monitoring  
To keep an eye on disk space, performance, and unsuccessful jobs, use open-source programs like Nagios or Oracle Enterprise Manager.  
  
d) Power and Physical Protection  
To avoid data loss from power outages, keep servers in a safe, climate-controlled space with a UPS backup.  
  
e) Capacity Planning Index columns (bike\_id, donor\_id, volunteer\_id) are commonly utilized in joins and gather data to maintain query responsiveness.

4. Safety Procedures  
  
Patch management: Apply OS and database security upgrades on a regular basis.  
  
User Training: Volunteers should be aware of data handling guidelines and phishing scams.  
  
Establish precise procedures for identifying and handling data breaches in your incident response plan.  
  
5. Conclusion  
  
Encryption and least-privilege access are used to safeguard BikesRUs data; constraints, triggers, and audits are used to ensure integrity; and redundancy and backups are used to provide availability.  
By putting these safeguards in place, BikesRUs can continue to empower the Gugulethu community in a safe and sustainable manner by ensuring that the outlet's systems are reliable, robust, and compliant.