INSY7212 - Summative Project

Ethan Edwards - ST10442420

Question 1

```
------------------------------
-- Basic Tables (Question 1) --
------------------------------

CREATE TABLE volunteers (
    volunteer_id    VARCHAR2(16)    PRIMARY KEY,
    first_name      VARCHAR2(32)    NOT NULL,
    surname         VARCHAR2(32)    NOT NULL,
    contact_number  VARCHAR2(16)    NOT NULL,
    address         VARCHAR2(128)   NOT NULL,
    email           VARCHAR2(64)    NOT NULL
);

CREATE TABLE donors (
    donor_id        VARCHAR2(16) PRIMARY KEY,
    first_name      VARCHAR2(32) NOT NULL,
    surname         VARCHAR2(32) NOT NULL,
    contact_number  VARCHAR2(16) NOT NULL,
    email           VARCHAR2(64) NOT NULL
);

CREATE TABLE bikes (
    bike_id         VARCHAR2(16) PRIMARY KEY,
    bike_desc       VARCHAR2(64) NOT NULL,
    bike_type       VARCHAR2(32) NOT NULL,
    manufacturer    VARCHAR2(16) NOT NULL
);

--------------------------------
-- Joining Tables (Question 1) --
--------------------------------

CREATE TABLE donations (
    donation_id     NUMBER          GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY,
    donor_id        VARCHAR2(16)    NOT NULL,
    bike_id         VARCHAR2(16)    NOT NULL,
    donation_value  NUMBER(10,2)    NOT NULL,
    volunteer_id    VARCHAR2(16)    NOT NULL,
    donation_date   DATE            NOT NULL,
```

```
    CONSTRAINT fk_donors_donations
        FOREIGN KEY (donor_id)
        REFERENCES donors(donor_id),

    CONSTRAINT fk_bikes_donations
        FOREIGN KEY (bike_id)
        REFERENCES bikes(bike_id),

    CONSTRAINT fk_volunteers_donations
        FOREIGN KEY (volunteer_id)
        REFERENCES volunteers(volunteer_id)
);

-----------------------------------
-- INSERT Statements (Question 1) --
-----------------------------------

-- Volunteers
INSERT ALL
    INTO volunteers (volunteer_id, first_name, surname, contact_number,
address, email)
        VALUES ('vol101', 'Kenny', 'Temba', '0677277521', '10 Sands Road',
'kennyt@bikerus.com')
    INTO volunteers (volunteer_id, first_name, surname, contact_number,
address, email)
        VALUES ('vol102', 'Mamelodi', 'Marks', '0737377522', '20 Langes
Street', 'mamelodim@bikerus.com')
    INTO volunteers (volunteer_id, first_name, surname, contact_number,
address, email)
        VALUES ('vol103', 'Ada', 'Andrews', '0817117523', '31 Williams
Street', 'adanyaa@bikerus.com')
    INTO volunteers (volunteer_id, first_name, surname, contact_number,
address, email)
        VALUES ('vol104', 'Evans', 'Tambala', '0697215244', '1 Free Road',
'evanst@bikerus.com')
    INTO volunteers (volunteer_id, first_name, surname, contact_number,
address, email)
        VALUES ('vol105', 'Xolani', 'Samson', '0727122255', '12 Main Road',
'xolanis@bikerus.com')
SELECT * FROM dual;

-- Donors
INSERT ALL
```

```
    INTO donors (donor_id, first_name, surname, contact_number, email)
        VALUES ('DID11', 'Jeff', 'Wanya', '0827172250',
'wanyajeff@ymail.com')
    INTO donors (donor_id, first_name, surname, contact_number, email)
        VALUES ('DID12', 'Sthembeni', 'Pisho', '0837865670',
'sthepisho@ymail.com')
    INTO donors (donor_id, first_name, surname, contact_number, email)
        VALUES ('DID13', 'James', 'Temba', '0878978650', 'jimmy@ymail.com')
    INTO donors (donor_id, first_name, surname, contact_number, email)
        VALUES ('DID14', 'Luramo', 'Misi', '0826575650',
'luramom@ymail.com')
    INTO donors (donor_id, first_name, surname, contact_number, email)
        VALUES ('DID15', 'Abraham', 'Xolani', '0797656430',
'axolani@ymail.com')
    INTO donors (donor_id, first_name, surname, contact_number, email)
        VALUES ('DID16', 'Rumi', 'Jones', '0668899221', 'rjones@true.com')
    INTO donors (donor_id, first_name, surname, contact_number, email)
        VALUES ('DID17', 'Xolani', 'Redo', '0614553389', 'xredo@ymail.com')
    INTO donors (donor_id, first_name, surname, contact_number, email)
        VALUES ('DID18', 'Tenny', 'Stars', '0824228870',
'tenstars@true.com')
    INTO donors (donor_id, first_name, surname, contact_number, email)
        VALUES ('DID19', 'Tiny', 'Rambo', '0715554333', 'trambo@ymail.com')
    INTO donors (donor_id, first_name, surname, contact_number, email)
        VALUES ('DID20', 'Yannick', 'Leons', '0615554323',
'yleons@true.com')
SELECT * FROM dual;

-- Bikes
INSERT ALL
    INTO bikes (bike_id, bike_desc, bike_type, manufacturer)
        VALUES ('B001', 'BMX AX1', 'Road Bike', 'BMX')
    INTO bikes (bike_id, bike_desc, bike_type, manufacturer)
        VALUES ('B002', 'Giant Domain 1', 'Road Bike', 'Giant')
    INTO bikes (bike_id, bike_desc, bike_type, manufacturer)
        VALUES ('B003', 'Ascent 26In', 'Mountain Bike', 'Raleigh')
    INTO bikes (bike_id, bike_desc, bike_type, manufacturer)
        VALUES ('B004', 'Canyon 6X', 'Kids Bike', 'Canyon')
    INTO bikes (bike_id, bike_desc, bike_type, manufacturer)
        VALUES ('B005', 'Marvel', 'Gravel Road Bike', 'BMX')
    INTO bikes (bike_id, bike_desc, bike_type, manufacturer)
        VALUES ('B006', 'Mountain 21 Speed', 'Mountain Bike', 'BMX')
    INTO bikes (bike_id, bike_desc, bike_type, manufacturer)
```

```
        VALUES ('B007', 'Canyon Roadster', 'Road Bike', 'Canyon')
    INTO bikes (bike_id, bike_desc, bike_type, manufacturer)
        VALUES ('B008', 'Legion 101', 'Hybrid Bike', 'BMX')
    INTO bikes (bike_id, bike_desc, bike_type, manufacturer)
        VALUES ('B009', 'Madonna 9', 'Road Bike', 'Trek')
    INTO bikes (bike_id, bike_desc, bike_type, manufacturer)
        VALUES ('B010', 'Comp 2022', 'Mountain Bike', 'Trek')
    INTO bikes (bike_id, bike_desc, bike_type, manufacturer)
        VALUES ('B011', 'BMX AX15', 'Road Bike', 'BMX')
SELECT * FROM dual;

-- Donations (Cannot be done as a union)
INSERT INTO donations (donor_id, bike_id, donation_value, volunteer_id,
donation_date)
VALUES ('DID11', 'B001', 1500, 'vol101', DATE '2023-05-01');

INSERT INTO donations (donor_id, bike_id, donation_value, volunteer_id,
donation_date)
VALUES ('DID12', 'B002', 2500, 'vol101', DATE '2023-05-03');

INSERT INTO donations (donor_id, bike_id, donation_value, volunteer_id,
donation_date)
VALUES ('DID13', 'B003', 1000, 'vol103', DATE '2023-05-03');

INSERT INTO donations (donor_id, bike_id, donation_value, volunteer_id,
donation_date)
VALUES ('DID14', 'B004', 1750, 'vol105', DATE '2023-05-05');

INSERT INTO donations (donor_id, bike_id, donation_value, volunteer_id,
donation_date)
VALUES ('DID15', 'B006', 2000, 'vol101', DATE '2023-05-07');

INSERT INTO donations (donor_id, bike_id, donation_value, volunteer_id,
donation_date)
VALUES ('DID16', 'B007', 1800, 'vol105', DATE '2023-05-09');

INSERT INTO donations (donor_id, bike_id, donation_value, volunteer_id,
donation_date)
VALUES ('DID17', 'B008', 1500, 'vol101', DATE '2023-05-15');

INSERT INTO donations (donor_id, bike_id, donation_value, volunteer_id,
donation_date)
VALUES ('DID18', 'B009', 1500, 'vol103', DATE '2023-05-19');
```

```sql
INSERT INTO donations (donor_id, bike_id, donation_value, volunteer_id,
donation_date)
VALUES ('DID12', 'B010', 2500, 'vol103', DATE '2023-05-25');

INSERT INTO donations (donor_id, bike_id, donation_value, volunteer_id,
donation_date)
VALUES ('DID20', 'B005', 3500, 'vol105', DATE '2023-05-05');

INSERT INTO donations (donor_id, bike_id, donation_value, volunteer_id,
donation_date)
VALUES ('DID19', 'B011', 2500, 'vol103', DATE '2023-05-30');
```

Question 2

```
SELECT
    dr.donor_id as "Donor ID",
    b.bike_type as "Bike Type",
    b.bike_desc as "Bike Description",
    dn.donation_value as "Bike Value"
FROM
    donations dn
        JOIN
            bikes b
        ON
            dn.bike_id = b.bike_id
        JOIN
            donors dr
        ON
            dn.donor_id =  dr.donor_id
WHERE
    dn.donation_value > 1500
;
```

Question 3

```
SET SERVEROUTPUT ON;

DECLARE
    v_vat_rate CONSTANT NUMBER(10,2) := 0.15;

    CURSOR c_bikes IS
        SELECT
            b.bike_desc,
            b.bike_type,
            b.manufacturer,
            d.donation_value
        FROM
            bikes b
                JOIN
                    donations d
                ON
                    d.bike_id = b.bike_id;

    v_bike          c_bikes%ROWTYPE;
    v_vat_amount    DECIMAL;
    v_total_amount  DECIMAL;
BEGIN
    OPEN c_bikes;

    LOOP
        FETCH
            c_bikes
        INTO
            v_bike;

        EXIT WHEN
            c_bikes%NOTFOUND;

        v_vat_amount := v_bike.donation_value * v_vat_rate;
        v_total_amount := v_bike.donation_value + v_vat_amount;

        DBMS_OUTPUT.PUT_LINE(RPAD('BIKE DESCRIPTION:', 30)  ||
v_bike.bike_desc);
        DBMS_OUTPUT.PUT_LINE(RPAD('BIKE MANUFACTURER:', 30) ||
v_bike.manufacturer);
        DBMS_OUTPUT.PUT_LINE(RPAD('BIKE TYPE:', 30)         ||
```

```
v_bike.bike_type);
        DBMS_OUTPUT.PUT_LINE(RPAD('VALUE:', 30)             ||
v_bike.donation_value);
        DBMS_OUTPUT.PUT_LINE(RPAD('VAT:', 30)               ||
v_vat_amount);
        DBMS_OUTPUT.PUT_LINE(RPAD('TOTAL AMNT:', 30)        ||
v_total_amount);

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

    END LOOP;

    CLOSE c_bikes;
END;
/
```

Question 4

```
CREATE OR REPLACE VIEW
    vwBikesRUs
AS
    SELECT
        dr.first_name || ',' || dr.surname AS DONOR_NAME,
        dr.contact_number AS CONTACT_NO,
        b.bike_type,
        dn.donation_date
    FROM
        donations dn
            JOIN
                donors dr
            ON
                dn.donor_id = dr.donor_id
            JOIN
                bikes b
            ON
                dn.bike_id = b.bike_id
    WHERE
        dn.volunteer_id = 'vol105'
;

SELECT * FROM vwBikesRUs;

--------------------------------------------------------------------------
-----
-- BENEFITS TO USING A VIEW:
-- 1. Using views allows the end user to run simpler queries. Instead of
having
--    to type out complex queries, a simple SELECT can be run on the view.
This
--    is useful for when less technically-able users need to query the
database
--
-- 2. Once created, a view is stored and can be run elsewhere. Views are
also
--    preserved after a session ends, and can be used in the next one
without
--    having to be remade.
--------------------------------------------------------------------------
-----
```

Question 5

```
SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE
    spDonorDetails(p_bike_id IN bikes.bike_id%TYPE)
AS
        v_donor_name        VARCHAR2(32);
        v_contact_number    donors.contact_number%TYPE;
        v_volunteer_name    volunteers.first_name%TYPE;
        v_donation_date     donations.donation_date%TYPE;
        v_bike_desc         bikes.bike_desc%TYPE;
BEGIN
    SELECT
        dr.first_name || ' ' || dr.surname,
        dr.contact_number,
        v.first_name,
        dn.donation_date,
        b.bike_desc
    INTO
        v_donor_name,
        v_contact_number,
        v_volunteer_name,
        v_donation_date,
        v_bike_desc
    FROM
        donations dn
            JOIN
                donors dr
            ON
                dn.donor_id = dr.donor_id
            JOIN
                volunteers v
            ON
                dn.volunteer_id = v.volunteer_id
            JOIN
                bikes b
            ON
                dn.bike_id = b.bike_id
    WHERE
        b.bike_id = p_bike_id
    ;
```

```
    DBMS_OUTPUT.PUT_LINE('ATTENTION: '
        || v_donor_name
        || ' (contact number: '
        || v_contact_number
        || ') assisted by: '
        || v_volunteer_name
        || ', donated the '
        || v_bike_desc
        || ' on the '
        || v_donation_date
    );

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No bike with ID ' || p_bike_id || ' exists');
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Multiple donations for bike with ID ' ||
p_bike_id || ' exist');
END;
/

--------------------------------------------------------------------------
-----
-- EXPLANATION OF EXCEPTION HANDLING:
-- Exception handling has been used in this function to aide in usability
in the
-- case of user error. If a user makes a mistake, such as entering a
-- non-existent BikeID, or if there is a problem in the database and the
same
-- bike has been donated twice, they will be notified with a clean and easy
to
-- understand error message.
--------------------------------------------------------------------------
-----

BEGIN
    spDonorDetails('B004');
END;
/
```

Question 6

```
--------------------------------------------------------------------------------
-----
-- FUNCTION OBJECTIVE:
-- This function takes in the name of a bike brand, and then determines the
-- the total value of those bikes. This may be useful to determine which
bike
-- brand is the most popular donation, and will aide BikesRUs in finding
donation
-- sources.
--------------------------------------------------------------------------------
-----

CREATE OR REPLACE FUNCTION
    sfManufacturerValue(p_manufacturer IN bikes.manufacturer%TYPE)
RETURN
    NUMBER
IS
    v_total donations.donation_value%TYPE;
    BEGIN
        SELECT
            SUM(d.donation_value)
        INTO
            v_total
        FROM
            donations d
                JOIN
                    bikes b
                ON
                    d.bike_id = b.bike_id
        WHERE
            b.manufacturer = p_manufacturer
        GROUP BY
            b.manufacturer
        ;

        RETURN v_total;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE(
                'No bikes are from the manufacturer '
                    || p_manufacturer
```

```
            );
    END
;
/

SELECT
    b.manufacturer,
    sfManufacturerValue(b.manufacturer) as TOTAL_VALUE
FROM
    bikes b
GROUP BY
    b.manufacturer
;
```

Question 7

```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR c_bikes IS
        SELECT
            b.bike_id,
            b.bike_type,
            b.manufacturer,
            d.donation_value
        FROM
            donations d
                JOIN
                    bikes b
                ON
                    d.bike_id = b.bike_id
        ;

    v_rec       c_bikes%ROWTYPE;
    v_rating    VARCHAR2(100);
BEGIN
    OPEN c_bikes;

    LOOP
        FETCH c_bikes INTO v_rec;
        EXIT WHEN c_bikes%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE(RPAD('BIKE ID:', 30) || v_rec.bike_id);
        DBMS_OUTPUT.PUT_LINE(RPAD('BIKE TYPE:', 30) || v_rec.bike_type);
        DBMS_OUTPUT.PUT_LINE(RPAD('BIKE MANUFACTURER:', 30) ||
v_rec.manufacturer);
        DBMS_OUTPUT.PUT_LINE(RPAD('BIKE VALUE:', 30) ||
v_rec.donation_value);

        IF v_rec.donation_value > 3000 THEN
            v_rating := '* * *';
        ELSIF v_rec.donation_value > 1500 THEN
            v_rating := '* *';
        ELSE
            v_rating := '*';
        END IF;
```

```
        DBMS_OUTPUT.PUT_LINE(RPAD('STATUS:', 30) || v_rating);

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

    CLOSE c_bikes;
END;
/
```

Question 8

```
SELECT
    b.bike_id,
    b.bike_type,
    b.manufacturer,
    d.donation_value AS VALUE,
    CASE
        WHEN d.donation_value > 3000 THEN '* * *'
        WHEN d.donation_value > 1500 THEN '* *'
        ELSE '*'
    END AS STATUS
FROM
    donations d
        JOIN
            bikes b
        ON
            d.bike_id = b.bike_id
;
```

Question 9.1

```
CREATE OR REPLACE TRIGGER
    tr_donation_no_delete
BEFORE DELETE ON
    donations
FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20001, 'Donation records cannot be deleted');
END;
/

DELETE FROM
    donations
WHERE
    donation_id = 1
;
```

Question 9.2

```
Question 9.2
CREATE OR REPLACE TRIGGER
    tr_valid_bike_value
BEFORE UPDATE ON
    donations
FOR EACH ROW
BEGIN
    IF :NEW.donation_value <= 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Donation value must be greater
than zero');
    END IF;
END;
/

UPDATE
    donations
SET
    donation_value = 0
WHERE
    donation_id = 1
;
```

Technical Report

Integrity and confidentiality are critical to the success of a database-oriented information system. BikesRUs will need to ensure that their database system is properly configured, else they will risk losing or exposing the private data of their customers and donors, and may experience outages that will affect the performance of the business **(Sherrill, 2024)**. This report details tools and suggestions that can be followed to mitigate these risks.

Confidentiality

Confidentiality refers to the security and privacy of sensitive data - data that a business does not want exposed **(Allemann, 2024)**. In the case of BikesRUs, sensitive data will include the donors and volunteer tables, which contain personal information, as well as the bikes and donations tables, which contain data that could be valuable to criminals, such as which bikes have the highest value. Below are several tools and techniques that can be used to keep this data confidential:

Transparent Data Encryption

Transparent Data Encryption (TDE) encrypts database files that are stored on disk. Tables and their respective data are only decrypting when needed and when proper authorization is provided **(Sharma, 2022)**. TDE adds a substantial layer of protection to a database and reduces that number of potential intrusion points in a system.

Oracle Net Encryption

Oracle Net Encryption allows data to be securely transmitted between databases and their front-end applications **(Fowler, 2020)**. This ensures that data is protected from man-in-the-middle attacks and prevents sensitive data from being eavesdropped. Oracle Net Encryption works by encrypting data before it is sent, and then decrypting it on the receiving end, ensuring that only those with decryption keys can access the database.

Internal Encryption

Data within tables should also be encrypted. This adds another layer of security as even if the data in a table is leaked, it is still not usable until it is decrypted. This can be done with tools such as DBMS_CRYPTO **(Singh, 2024)**, which provides tools within an Oracle Database to encrypt and decrypt arbitrary data.

Example: Encrypting donor national ID numbers

```
UPDATE donors
SET id_number = DBMS_CRYPTO.ENCRYPT(
    UTL_I18N.STRING_TO_RAW('8001015009087', 'AL32UTF8'),
    DBMS_CRYPTO.AES_CBC_PKCS5,
    UTL_I18N.STRING_TO_RAW('encryptionkey', 'AL32UTF8')
);
```

Integrity

Constraints

There are several constraints built into the SQL language that can be used to ensure the integrity of data **(Bhattacharya, 2024)**. These constraints enforce structure on the data in a database, reducing the potential for errors, and ensure that entered data meets strict criteria. Two recommended constraints are shown below:

Foreign Keys

Foreign keys enforce a structure to relationships in a database **(Custer, 2023)**. This ensures that relationships are consistent and are fulfilled when required. In the case of BikesRUs, a constraint is set requiring that each donation references a bike:

```
CREATE TABLE donations (
    bike_id         VARCHAR2(16)    NOT NULL,

    CONSTRAINT fk_bikes_donations
        FOREIGN KEY (bike_id)
        REFERENCES bikes(bike_id),
);
```

Unique

Certain tables may have rows that should not be duplicated. This is typically the case when the details of unique real world items are stored, such as people **(Bhattacharya, 2024)**. For example, to prevent the same donor's details from being entered twice, a constraint can be put on info that may be unique to them, such as their email address:

```
CREATE TABLE donors (
```

```
    email           VARCHAR2(64) UNIQUE NOT NUL,
);
```

Triggers

Triggers can be used to run arbitrary checks before changes are made to a database **(Middha, 2023)**. These can be used to enforce certain conditions and reject invalid data, thus ensuring the integrity of the database. Triggers are set to run before either an INSERT, UPDATE or DELETE query. These are already present in the BikesRUs database, as shown below:

```
CREATE OR REPLACE TRIGGER
    tr_donation_no_delete
BEFORE DELETE ON
    donations
FOR EACH ROW
BEGIN
    RAISE_APPLICATION_ERROR(-20001, 'Donation records cannot be deleted');
END;
/

DELETE FROM
    donations
WHERE
    donation_id = 1
;

CREATE OR REPLACE TRIGGER
    tr_valid_bike_value
BEFORE UPDATE ON
    donations
FOR EACH ROW
BEGIN
    IF :NEW.donation_value <= 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Donation value must be greater
than zero');
    END IF;
END;
/

UPDATE
    donations
```

```
SET
    donation_value = 0
WHERE
    donation_id = 1
;
```

SQL Databases such as the one used by BikesRUs require integrity and confidentiality to be useful to the organisation that owns them. While ensuring that a database has these properties is difficult, it can be achieved through the use of several popular and well proven techniques. This report outlines several of them, and provides recommendations that, if followed, will ensure that the system used by BikesRUs is confidential and will retain integrity throughout its use.

References

Allemann, G. (2024) “What is confidential data? And how does it differ from sensitive data?,” *Data Quality Matters*, 19 July. Available at: https://blog.masterdata.co.za/2024/07/19/what-is-confidential-data-and-how-does-it-differ-from-sensitive-data/ (Accessed: November 2, 2025).

Bhattacharya, A. (2024) “Constraints in SQL,” *Medium*, 27 July. Available at: https://medium.com/@AnweshaB/constraints-in-sql-6b68630d822d (Accessed: November 3, 2025).

Custer, C. (2023) *What is a foreign key? (With SQL examples)*, *Cockroach Labs*. Available at: https://www.cockroachlabs.com/blog/what-is-a-foreign-key/ (Accessed: November 3, 2025).

Fowler, H.E. (2020) *Oracle database network encryption(Native vs. TLS/SSL)*, *Quest Oracle Community*. Available at: https://questoraclecommunity.org/learn/blogs/oracle-database-network-encryption-native-vs-tls-ssl/ (Accessed: November 2, 2025).

Middha, D. (2023) *Introduction to triggers in SQL server*. Available at: https://www.c-sharpcorner.com/uploadfile/63f5c2/triggers-in-sql-server/ (Accessed: November 3, 2025).

Sharma, R. (2022) *Oracle data security withTDE | Rackspace Technology*. Available at: https://www.rackspace.com/blog/transparent-data-encryption (Accessed: November 2, 2025).

Sherrill, A. (2024) “What is database security? - tenhats,” 15 November. Available at: https://tenhats.com/what-is-database-security/ (Accessed: November 2, 2025).

Singh, S. (2024) *Encrypting and decrypting data in oracle: a step-by-step guide*, *SmartTechWays – Innovative Solutions for Smart Businesses*. Available at: https://smarttechways.com/2024/06/13/encrypting-and-decrypting-data-in-oracle-a-step-by-step-guide/ (Accessed: November 2, 2025).