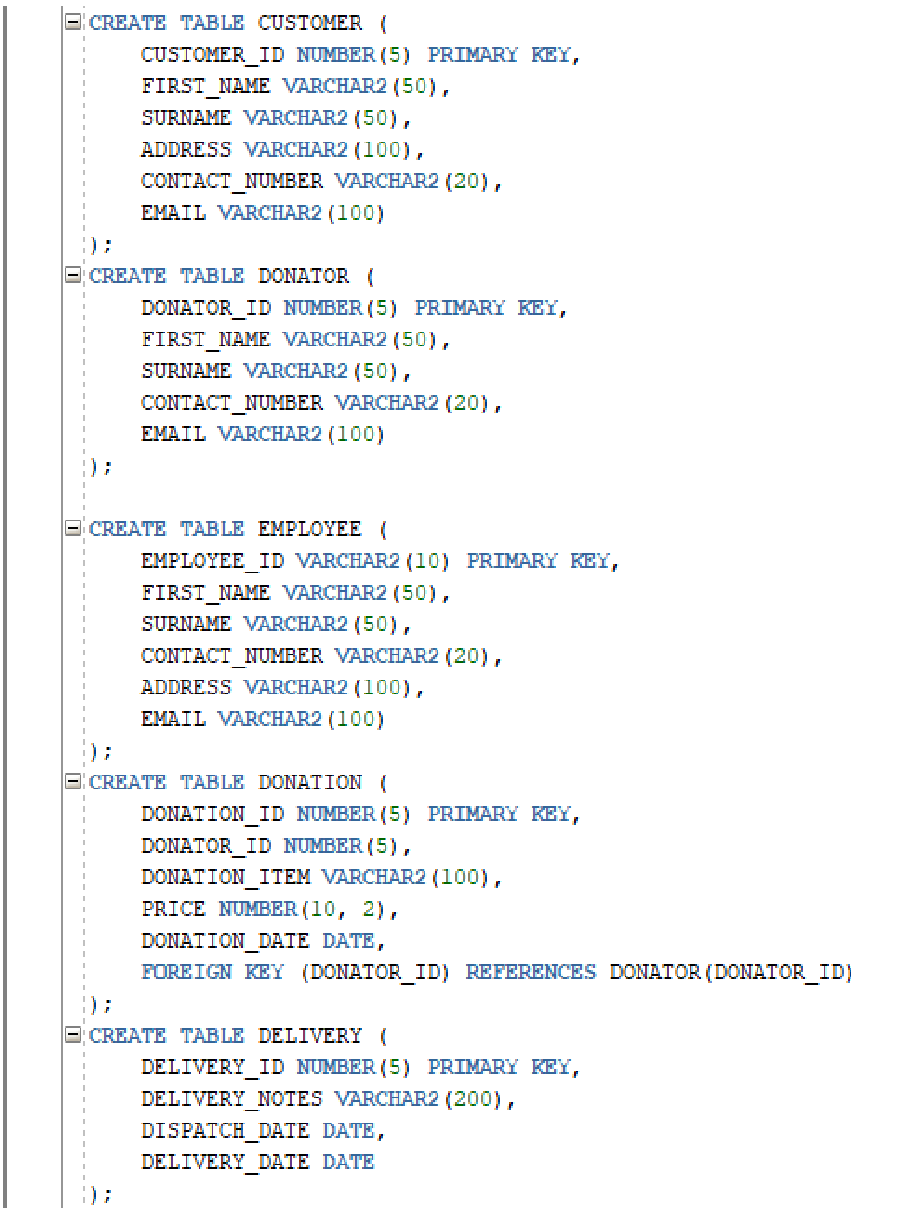
INSY7213 A2

ST10460792

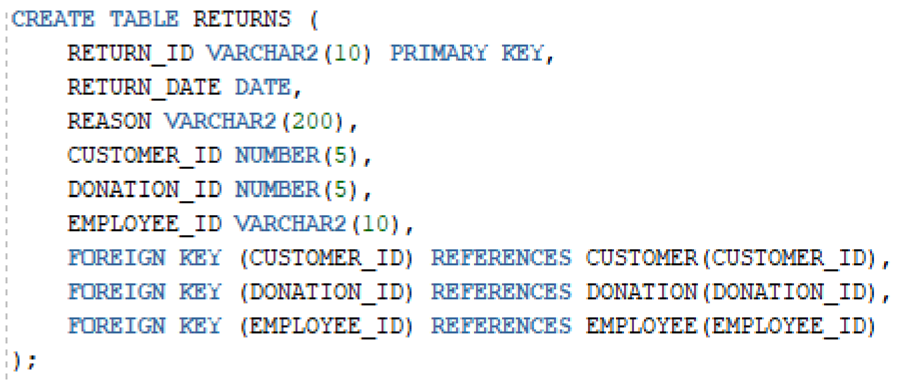
Goshen Mtambo

Group 3

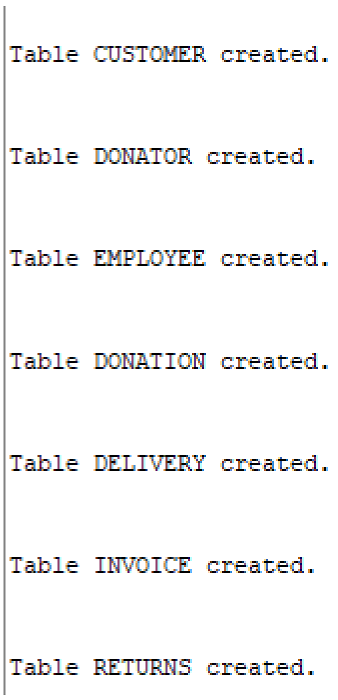
Question 1



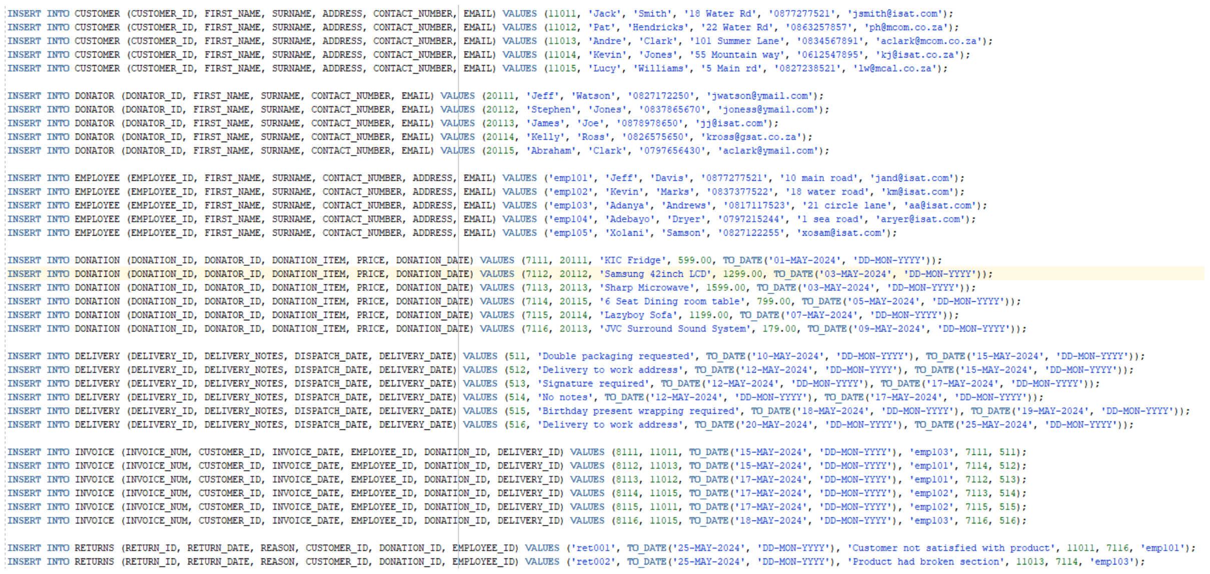
```sql
CREATE TABLE CUSTOMER (
    CUSTOMER_ID NUMBER(5) PRIMARY KEY,
    FIRST_NAME VARCHAR2(50),
    SURNAME VARCHAR2(50),
    ADDRESS VARCHAR2(100),
    CONTACT_NUMBER VARCHAR2(20),
    EMAIL VARCHAR2(100)
);
CREATE TABLE DONATOR (
    DONATOR_ID NUMBER(5) PRIMARY KEY,
    FIRST_NAME VARCHAR2(50),
    SURNAME VARCHAR2(50),
    CONTACT_NUMBER VARCHAR2(20),
    EMAIL VARCHAR2(100)
);

CREATE TABLE EMPLOYEE (
    EMPLOYEE_ID VARCHAR2(10) PRIMARY KEY,
    FIRST_NAME VARCHAR2(50),
    SURNAME VARCHAR2(50),
    CONTACT_NUMBER VARCHAR2(20),
    ADDRESS VARCHAR2(100),
    EMAIL VARCHAR2(100)
);
CREATE TABLE DONATION (
    DONATION_ID NUMBER(5) PRIMARY KEY,
    DONATOR_ID NUMBER(5),
    DONATION_ITEM VARCHAR2(100),
    PRICE NUMBER(10, 2),
    DONATION_DATE DATE,
    FOREIGN KEY (DONATOR_ID) REFERENCES DONATOR(DONATOR_ID)
);
CREATE TABLE DELIVERY (
    DELIVERY_ID NUMBER(5) PRIMARY KEY,
    DELIVERY_NOTES VARCHAR2(200),
    DISPATCH_DATE DATE,
    DELIVERY_DATE DATE
);
```



```sql
CREATE TABLE RETURNS (
    RETURN_ID VARCHAR2(10) PRIMARY KEY,
    RETURN_DATE DATE,
    REASON VARCHAR2(200),
    CUSTOMER_ID NUMBER(5),
    DONATION_ID NUMBER(5),
    EMPLOYEE_ID VARCHAR2(10),
    FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER(CUSTOMER_ID),
    FOREIGN KEY (DONATION_ID) REFERENCES DONATION(DONATION_ID),
    FOREIGN KEY (EMPLOYEE_ID) REFERENCES EMPLOYEE(EMPLOYEE_ID)
);
```



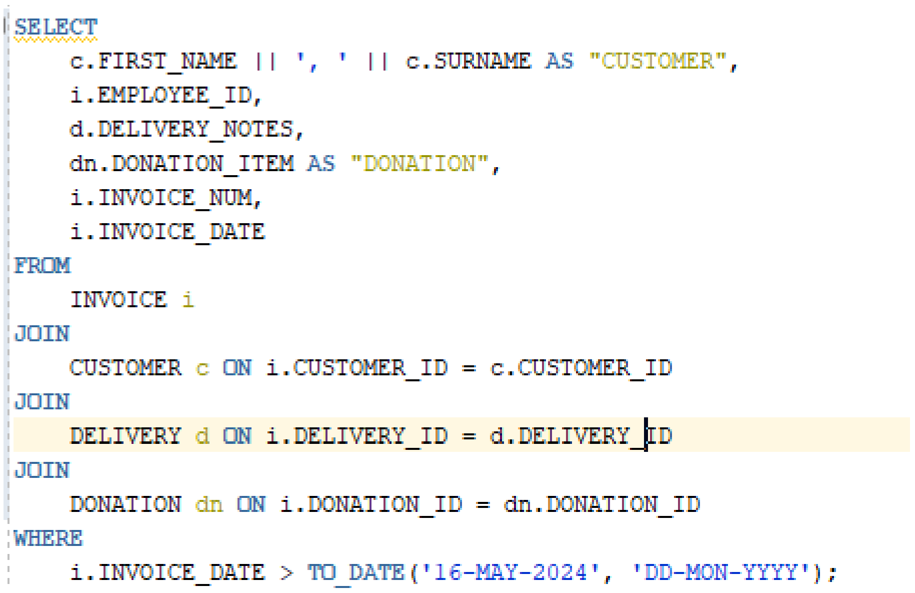
```
Table CUSTOMER created.


Table DONATOR created.


Table EMPLOYEE created.


Table DONATION created.


Table DELIVERY created.


Table INVOICE created.


Table RETURNS created.
```



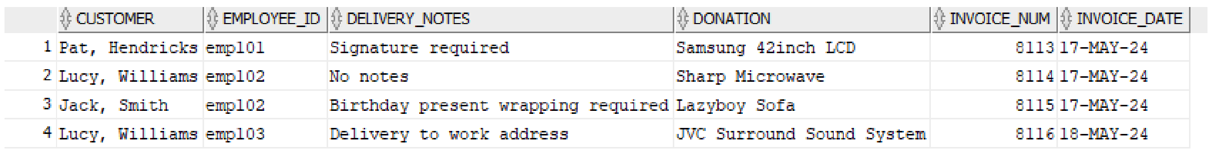
```sql
INSERT INTO CUSTOMER (CUSTOMER_ID, FIRST_NAME, SURNAME, ADDRESS, CONTACT_NUMBER, EMAIL) VALUES (11011, 'Jack', 'Smith', '18 Water Rd', '0877277521', 'jsmith@isat.com');
INSERT INTO CUSTOMER (CUSTOMER_ID, FIRST_NAME, SURNAME, ADDRESS, CONTACT_NUMBER, EMAIL) VALUES (11012, 'Pat', 'Hendricks', '22 Water Rd', '0863257857', 'ph@mcom.co.za');
INSERT INTO CUSTOMER (CUSTOMER_ID, FIRST_NAME, SURNAME, ADDRESS, CONTACT_NUMBER, EMAIL) VALUES (11013, 'Andre', 'Clark', '101 Summer Lane', '0834567891', 'aclark@mcom.co.za');
INSERT INTO CUSTOMER (CUSTOMER_ID, FIRST_NAME, SURNAME, ADDRESS, CONTACT_NUMBER, EMAIL) VALUES (11014, 'Kevin', 'Jones', '55 Mountain way', '0612547895', 'kj@isat.co.za');
INSERT INTO CUSTOMER (CUSTOMER_ID, FIRST_NAME, SURNAME, ADDRESS, CONTACT_NUMBER, EMAIL) VALUES (11015, 'Lucy', 'Williams', '5 Main rd', '0827238521', 'lw@mcal.co.za');

INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, EMAIL) VALUES (20111, 'Jeff', 'Watson', '0827172250', 'jwatson@ymail.com');
INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, EMAIL) VALUES (20112, 'Stephen', 'Jones', '0837865670', 'joness@ymail.com');
INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, EMAIL) VALUES (20113, 'James', 'Joe', '0878978650', 'jj@isat.com');
INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, EMAIL) VALUES (20114, 'Kelly', 'Ross', '0826575650', 'kross@gsat.co.za');
INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, EMAIL) VALUES (20115, 'Abraham', 'Clark', '0797656430', 'aclark@ymail.com');

INSERT INTO EMPLOYEE (EMPLOYEE_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, ADDRESS, EMAIL) VALUES ('emp101', 'Jeff', 'Davis', '0877277521', '10 main road', 'jand@isat.com');
INSERT INTO EMPLOYEE (EMPLOYEE_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, ADDRESS, EMAIL) VALUES ('emp102', 'Kevin', 'Marks', '0837377522', '18 water road', 'km@isat.com');
INSERT INTO EMPLOYEE (EMPLOYEE_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, ADDRESS, EMAIL) VALUES ('emp103', 'Adanya', 'Andrews', '0817117523', '21 circle lane', 'aa@isat.com');
INSERT INTO EMPLOYEE (EMPLOYEE_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, ADDRESS, EMAIL) VALUES ('emp104', 'Adebayo', 'Dryer', '0797215244', '1 sea road', 'aryer@isat.com');
INSERT INTO EMPLOYEE (EMPLOYEE_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, ADDRESS, EMAIL) VALUES ('emp105', 'Xolani', 'Samson', '0827122255', 'xosam@isat.com');

INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION_ITEM, PRICE, DONATION_DATE) VALUES (7111, 20111, 'KIC Fridge', 599.00, TO_DATE('01-MAY-2024', 'DD-MON-YYYY'));
INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION_ITEM, PRICE, DONATION_DATE) VALUES (7112, 20112, 'Samsung 42inch LCD', 1299.00, TO_DATE('03-MAY-2024', 'DD-MON-YYYY'));
INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION_ITEM, PRICE, DONATION_DATE) VALUES (7113, 20113, 'Sharp Microwave', 1599.00, TO_DATE('03-MAY-2024', 'DD-MON-YYYY'));
INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION_ITEM, PRICE, DONATION_DATE) VALUES (7114, 20115, '6 Seat Dining room table', 799.00, TO_DATE('05-MAY-2024', 'DD-MON-YYYY'));
INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION_ITEM, PRICE, DONATION_DATE) VALUES (7115, 20114, 'Lazyboy Sofa', 1199.00, TO_DATE('07-MAY-2024', 'DD-MON-YYYY'));
INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION_ITEM, PRICE, DONATION_DATE) VALUES (7116, 20113, 'JVC Surround Sound System', 179.00, TO_DATE('09-MAY-2024', 'DD-MON-YYYY'));

INSERT INTO DELIVERY (DELIVERY_ID, DELIVERY_NOTES, DISPATCH_DATE, DELIVERY_DATE) VALUES (511, 'Double packaging requested', TO_DATE('10-MAY-2024', 'DD-MON-YYYY'), TO_DATE('15-MAY-2024', 'DD-MON-YYYY'));
INSERT INTO DELIVERY (DELIVERY_ID, DELIVERY_NOTES, DISPATCH_DATE, DELIVERY_DATE) VALUES (512, 'Delivery to work address', TO_DATE('12-MAY-2024', 'DD-MON-YYYY'), TO_DATE('15-MAY-2024', 'DD-MON-YYYY'));
INSERT INTO DELIVERY (DELIVERY_ID, DELIVERY_NOTES, DISPATCH_DATE, DELIVERY_DATE) VALUES (513, 'Signature required', TO_DATE('12-MAY-2024', 'DD-MON-YYYY'), TO_DATE('17-MAY-2024', 'DD-MON-YYYY'));
INSERT INTO DELIVERY (DELIVERY_ID, DELIVERY_NOTES, DISPATCH_DATE, DELIVERY_DATE) VALUES (514, 'No notes', TO_DATE('12-MAY-2024', 'DD-MON-YYYY'), TO_DATE('17-MAY-2024', 'DD-MON-YYYY'));
INSERT INTO DELIVERY (DELIVERY_ID, DELIVERY_NOTES, DISPATCH_DATE, DELIVERY_DATE) VALUES (515, 'Birthday present wrapping required', TO_DATE('18-MAY-2024', 'DD-MON-YYYY'), TO_DATE('19-MAY-2024', 'DD-MON-YYYY'));
INSERT INTO DELIVERY (DELIVERY_ID, DELIVERY_NOTES, DISPATCH_DATE, DELIVERY_DATE) VALUES (516, 'Delivery to work address', TO_DATE('20-MAY-2024', 'DD-MON-YYYY'), TO_DATE('25-MAY-2024', 'DD-MON-YYYY'));

INSERT INTO INVOICE (INVOICE_NUM, CUSTOMER_ID, INVOICE_DATE, EMPLOYEE_ID, DONATION_ID, DELIVERY_ID) VALUES (8111, 11011, TO_DATE('15-MAY-2024', 'DD-MON-YYYY'), 'emp103', 7111, 511);
INSERT INTO INVOICE (INVOICE_NUM, CUSTOMER_ID, INVOICE_DATE, EMPLOYEE_ID, DONATION_ID, DELIVERY_ID) VALUES (8112, 11013, TO_DATE('15-MAY-2024', 'DD-MON-YYYY'), 'emp101', 7114, 512);
INSERT INTO INVOICE (INVOICE_NUM, CUSTOMER_ID, INVOICE_DATE, EMPLOYEE_ID, DONATION_ID, DELIVERY_ID) VALUES (8113, 11012, TO_DATE('17-MAY-2024', 'DD-MON-YYYY'), 'emp101', 7112, 513);
INSERT INTO INVOICE (INVOICE_NUM, CUSTOMER_ID, INVOICE_DATE, EMPLOYEE_ID, DONATION_ID, DELIVERY_ID) VALUES (8114, 11015, TO_DATE('17-MAY-2024', 'DD-MON-YYYY'), 'emp102', 7113, 514);
INSERT INTO INVOICE (INVOICE_NUM, CUSTOMER_ID, INVOICE_DATE, EMPLOYEE_ID, DONATION_ID, DELIVERY_ID) VALUES (8115, 11011, TO_DATE('17-MAY-2024', 'DD-MON-YYYY'), 'emp102', 7115, 515);
INSERT INTO INVOICE (INVOICE_NUM, CUSTOMER_ID, INVOICE_DATE, EMPLOYEE_ID, DONATION_ID, DELIVERY_ID) VALUES (8116, 11015, TO_DATE('18-MAY-2024', 'DD-MON-YYYY'), 'emp103', 7116, 516);

INSERT INTO RETURNS (RETURN_ID, RETURN_DATE, REASON, CUSTOMER_ID, DONATION_ID, EMPLOYEE_ID) VALUES ('ret001', TO_DATE('25-MAY-2024', 'DD-MON-YYYY'), 'Customer not satisfied with product', 11011, 7116, 'emp101');
INSERT INTO RETURNS (RETURN_ID, RETURN_DATE, REASON, CUSTOMER_ID, DONATION_ID, EMPLOYEE_ID) VALUES ('ret002', TO_DATE('25-MAY-2024', 'DD-MON-YYYY'), 'Product had broken section', 11013, 7114, 'emp103');
```

```
1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.
```

Question 2



```sql
SELECT
    c.FIRST_NAME || ', ' || c.SURNAME AS "CUSTOMER",
    i.EMPLOYEE_ID,
    d.DELIVERY_NOTES,
    dn.DONATION_ITEM AS "DONATION",
    i.INVOICE_NUM,
    i.INVOICE_DATE
FROM
    INVOICE i
JOIN
    CUSTOMER c ON i.CUSTOMER_ID = c.CUSTOMER_ID
JOIN
    DELIVERY d ON i.DELIVERY_ID = d.DELIVERY_ID
JOIN
    DONATION dn ON i.DONATION_ID = dn.DONATION_ID
WHERE
    i.INVOICE_DATE > TO_DATE('16-MAY-2024', 'DD-MON-YYYY');
```



	CUSTOMER	EMPLOYEE_ID	DELIVERY_NOTES	DONATION	INVOICE_NUM	INVOICE_DATE
1	Pat, Hendricks	emp101	Signature required	Samsung 42inch LCD	8113	17-MAY-24
2	Lucy, Williams	emp102	No notes	Sharp Microwave	8114	17-MAY-24
3	Jack, Smith	emp102	Birthday present wrapping required	Lazyboy Sofa	8115	17-MAY-24
4	Lucy, Williams	emp103	Delivery to work address	JVC Surround Sound System	8116	18-MAY-24

Question 3

1. Create the new table.

2. Implement a solution to automatically generate a unique ID.

```
CREATE TABLE Funding (
    funding_id NUMBER GENERATED ALWAYS AS IDENTITY,
    funder VARCHAR2(100),
    funding_amount NUMBER(12, 2)
);
```

```
Table FUNDING created.
```

3. Provide an example of the INSERT statement.

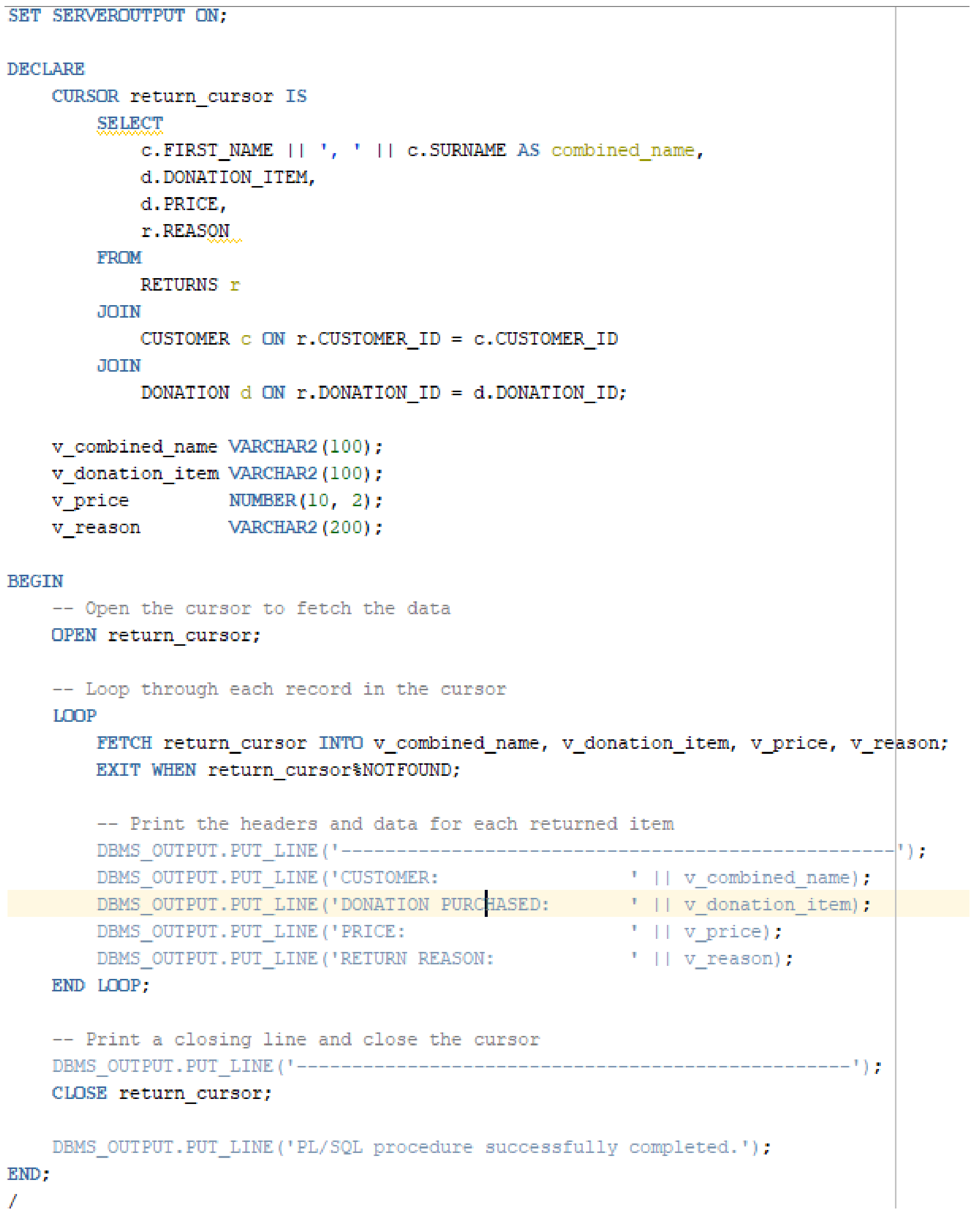
```
INSERT INTO Funding (funder, funding_amount)
VALUES ('ABC Foundation', 500000.00);
```

```
1 row inserted.
```

4. Add a brief comment to justify the solution.

The **`GENERATED ALWAYS AS IDENTITY`** clause is the modern Oracle standard for creating an auto-incrementing primary key. It's preferred because it's simpler and more efficient than older methods like using sequences and triggers separately.

Question 4

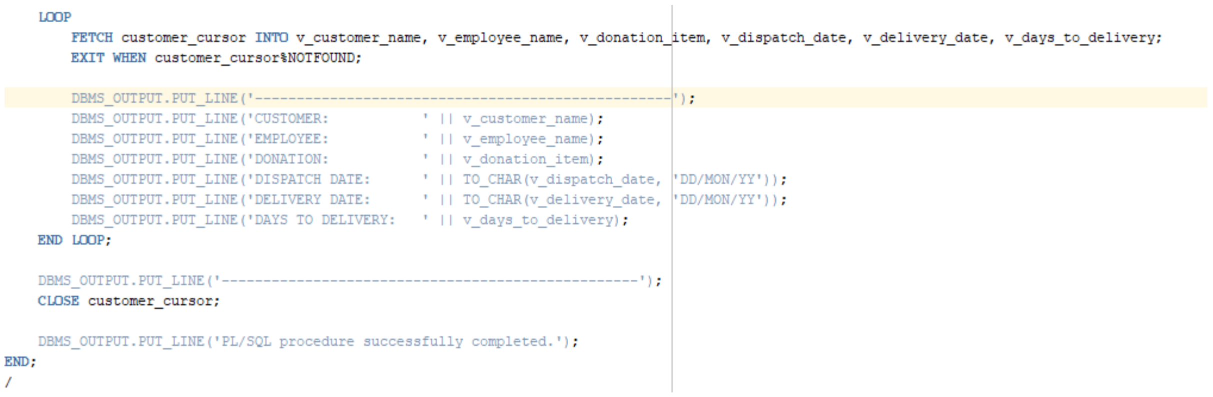


```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR return_cursor IS
        SELECT
            c.FIRST_NAME || ', ' || c.SURNAME AS combined_name,
            d.DONATION_ITEM,
            d.PRICE,
            r.REASON
        FROM
            RETURNS r
        JOIN
            CUSTOMER c ON r.CUSTOMER_ID = c.CUSTOMER_ID
        JOIN
            DONATION d ON r.DONATION_ID = d.DONATION_ID;

    v_combined_name VARCHAR2(100);
    v_donation_item VARCHAR2(100);
    v_price         NUMBER(10, 2);
    v_reason        VARCHAR2(200);

BEGIN
    -- Open the cursor to fetch the data
    OPEN return_cursor;

    -- Loop through each record in the cursor
    LOOP
        FETCH return_cursor INTO v_combined_name, v_donation_item, v_price, v_reason;
        EXIT WHEN return_cursor%NOTFOUND;

        -- Print the headers and data for each returned item
        DBMS_OUTPUT.PUT_LINE('--------------------------------------------------');
        DBMS_OUTPUT.PUT_LINE('CUSTOMER:                  ' || v_combined_name);
        DBMS_OUTPUT.PUT_LINE('DONATION PURCHASED:        ' || v_donation_item);
        DBMS_OUTPUT.PUT_LINE('PRICE:                     ' || v_price);
        DBMS_OUTPUT.PUT_LINE('RETURN REASON:             ' || v_reason);
    END LOOP;

    -- Print a closing line and close the cursor
    DBMS_OUTPUT.PUT_LINE('--------------------------------------------------');
    CLOSE return_cursor;

    DBMS_OUTPUT.PUT_LINE('PL/SQL procedure successfully completed.');
END;
/
```

```
--------------------------------------------------
CUSTOMER:                 Jack, Smith
DONATION PURCHASED:       JVC Surround Sound System
PRICE:                    179
RETURN REASON:            Customer not satisfied with product
--------------------------------------------------
CUSTOMER:                 Andre, Clark
DONATION PURCHASED:       6 Seat Dining room table
PRICE:                    799
RETURN REASON:            Product had broken section
--------------------------------------------------
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

Question 5

```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR customer_cursor IS
        SELECT
            c.FIRST_NAME || ' ' || c.SURNAME AS customer_name,
            e.FIRST_NAME || ' ' || e.SURNAME AS employee_name,
            dn.DONATION_ITEM,
            dl.DISPATCH_DATE,
            dl.DELIVERY_DATE,
            dl.DELIVERY_DATE - dl.DISPATCH_DATE AS days_to_delivery
        FROM
            INVOICE i
        JOIN
            CUSTOMER c ON i.CUSTOMER_ID = c.CUSTOMER_ID
        JOIN
            EMPLOYEE e ON i.EMPLOYEE_ID = e.EMPLOYEE_ID
        JOIN
            DELIVERY dl ON i.DELIVERY_ID = dl.DELIVERY_ID
        JOIN
            DONATION dn ON i.DONATION_ID = dn.DONATION_ID
        WHERE
            i.CUSTOMER_ID = 11011;

    v_customer_name      VARCHAR2(100);
    v_employee_name      VARCHAR2(100);
    v_donation_item      VARCHAR2(100);
    v_dispatch_date      DATE;
    v_delivery_date      DATE;
    v_days_to_delivery   NUMBER;

BEGIN
    OPEN customer_cursor;
```

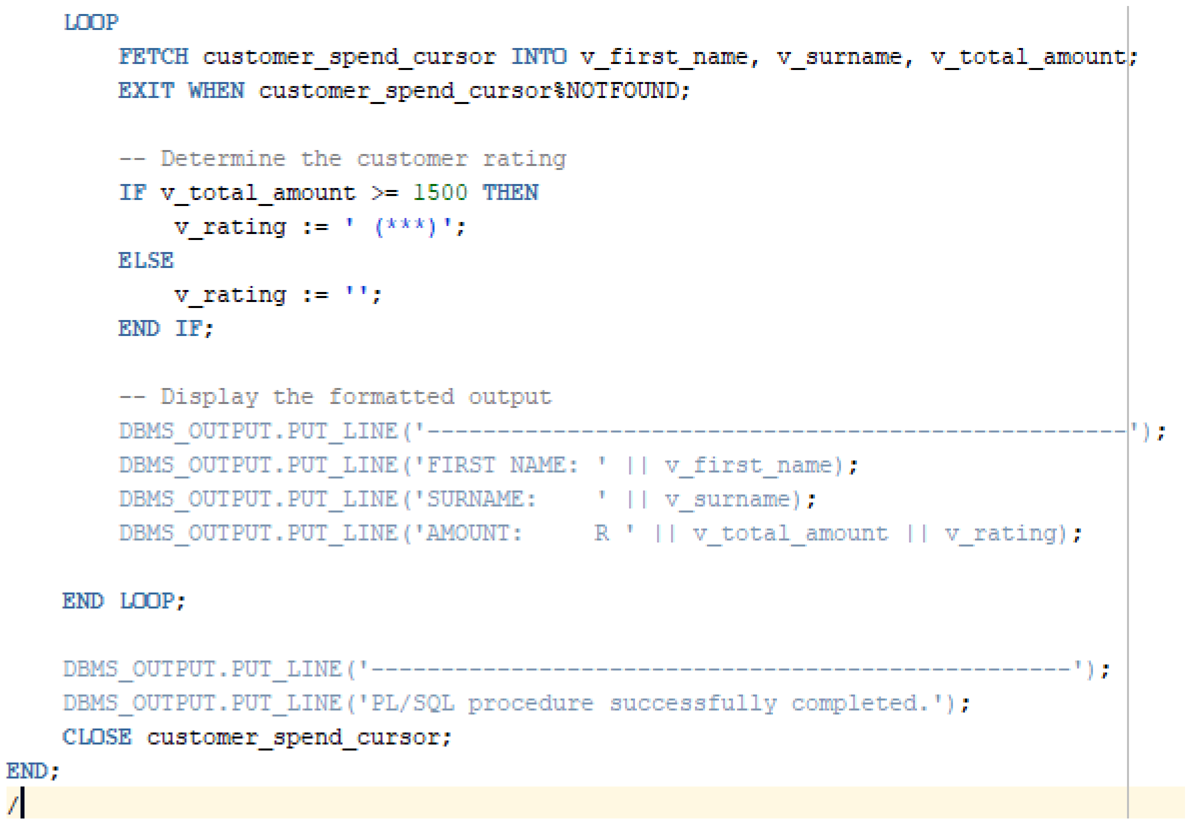


```
    LOOP
        FETCH customer_cursor INTO v_customer_name, v_employee_name, v_donation_item, v_dispatch_date, v_delivery_date, v_days_to_delivery;
        EXIT WHEN customer_cursor%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE('--------------------------------------------------');
        DBMS_OUTPUT.PUT_LINE('CUSTOMER:          ' || v_customer_name);
        DBMS_OUTPUT.PUT_LINE('EMPLOYEE:          ' || v_employee_name);
        DBMS_OUTPUT.PUT_LINE('DONATION:          ' || v_donation_item);
        DBMS_OUTPUT.PUT_LINE('DISPATCH DATE:     ' || TO_CHAR(v_dispatch_date, 'DD/MON/YY'));
        DBMS_OUTPUT.PUT_LINE('DELIVERY DATE:     ' || TO_CHAR(v_delivery_date, 'DD/MON/YY'));
        DBMS_OUTPUT.PUT_LINE('DAYS TO DELIVERY:  ' || v_days_to_delivery);
    END LOOP;

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

    DBMS_OUTPUT.PUT_LINE('PL/SQL procedure successfully completed.');
END;
/
```

```
--------------------------------------------------
CUSTOMER:           Jack Smith
EMPLOYEE:           Adanya Andrews
DONATION:           KIC Fridge
DISPATCH DATE:      10/MAY/24
DELIVERY DATE:      15/MAY/24
DAYS TO DELIVERY:   5
--------------------------------------------------
CUSTOMER:           Jack Smith
EMPLOYEE:           Kevin Marks
DONATION:           Lazyboy Sofa
DISPATCH DATE:      18/MAY/24
DELIVERY DATE:      19/MAY/24
DAYS TO DELIVERY:   1
--------------------------------------------------
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

Question 6

```
SET SERVEROUTPUT ON;

DECLARE
    CURSOR customer_spend_cursor IS
        SELECT
            c.FIRST_NAME,
            c.SURNAME,
            SUM(d.PRICE) AS total_amount
        FROM
            INVOICE i
        JOIN
            CUSTOMER c ON i.CUSTOMER_ID = c.CUSTOMER_ID
        JOIN
            DONATION d ON i.DONATION_ID = d.DONATION_ID
        GROUP BY
            c.CUSTOMER_ID, c.FIRST_NAME, c.SURNAME
        ORDER BY
            c.SURNAME;

    v_first_name   VARCHAR2(50);
    v_surname      VARCHAR2(50);
    v_total_amount NUMBER(10, 2);
    v_rating       VARCHAR2(5);

BEGIN
    OPEN customer_spend_cursor;
```

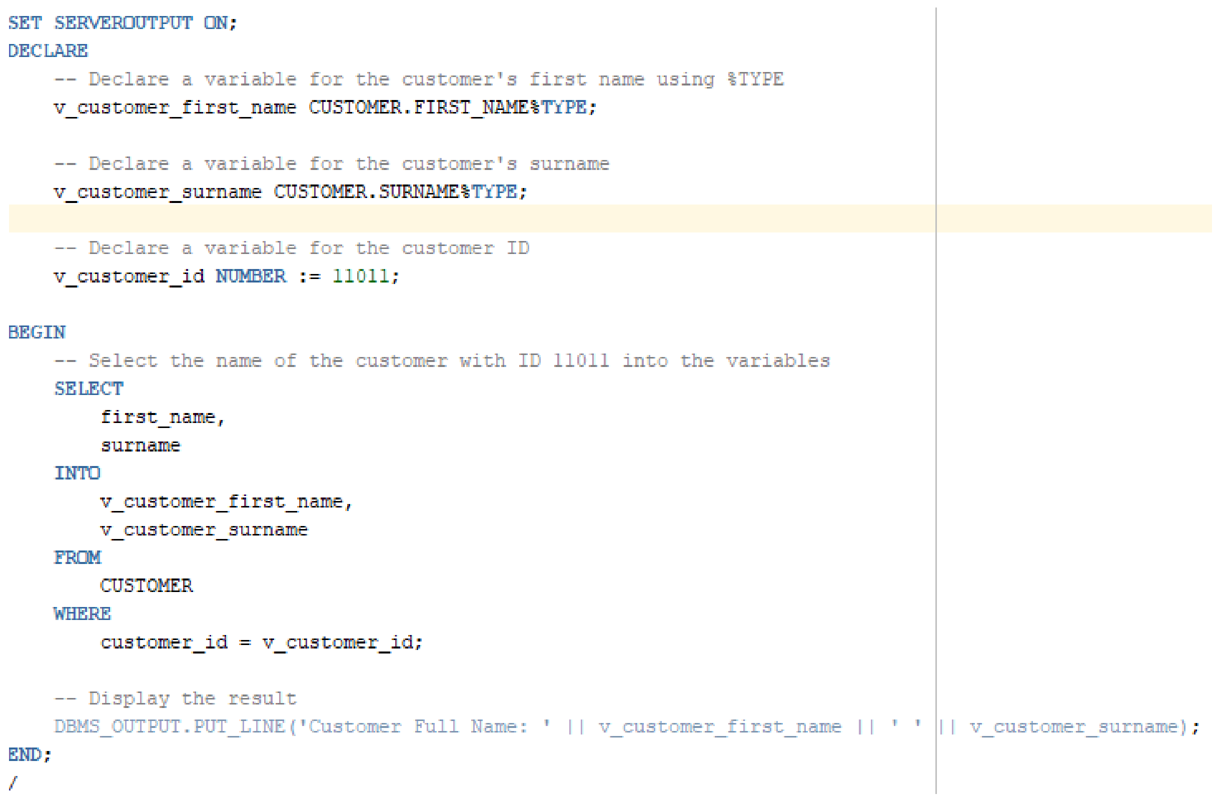


```
    LOOP
        FETCH customer_spend_cursor INTO v_first_name, v_surname, v_total_amount;
        EXIT WHEN customer_spend_cursor%NOTFOUND;

        -- Determine the customer rating
        IF v_total_amount >= 1500 THEN
            v_rating := ' (***)';
        ELSE
            v_rating := '';
        END IF;

        -- Display the formatted output
        DBMS_OUTPUT.PUT_LINE('--------------------------------------------------');
        DBMS_OUTPUT.PUT_LINE('FIRST NAME: ' || v_first_name);
        DBMS_OUTPUT.PUT_LINE('SURNAME:    ' || v_surname);
        DBMS_OUTPUT.PUT_LINE('AMOUNT:     R ' || v_total_amount || v_rating);

    END LOOP;

    DBMS_OUTPUT.PUT_LINE('--------------------------------------------------');
    DBMS_OUTPUT.PUT_LINE('PL/SQL procedure successfully completed.');
    CLOSE customer_spend_cursor;
END;
/
```

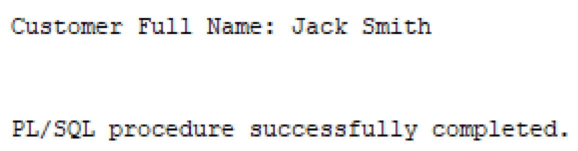
```
--------------------------------------------------
FIRST NAME: Andre
SURNAME:    Clark
AMOUNT:     R 799
--------------------------------------------------
FIRST NAME: Pat
SURNAME:    Hendricks
AMOUNT:     R 1299

DECLARE
*
ERROR at line 1:
ORA-06502: PL/SQL: numeric or value error: character string buffer too small
ORA-06512: at line 32

https://docs.oracle.com/error-help/db/ora-06502/


More Details :
https://docs.oracle.com/error-help/db/ora-06502/
https://docs.oracle.com/error-help/db/ora-06512/
```

Question 7

7.1

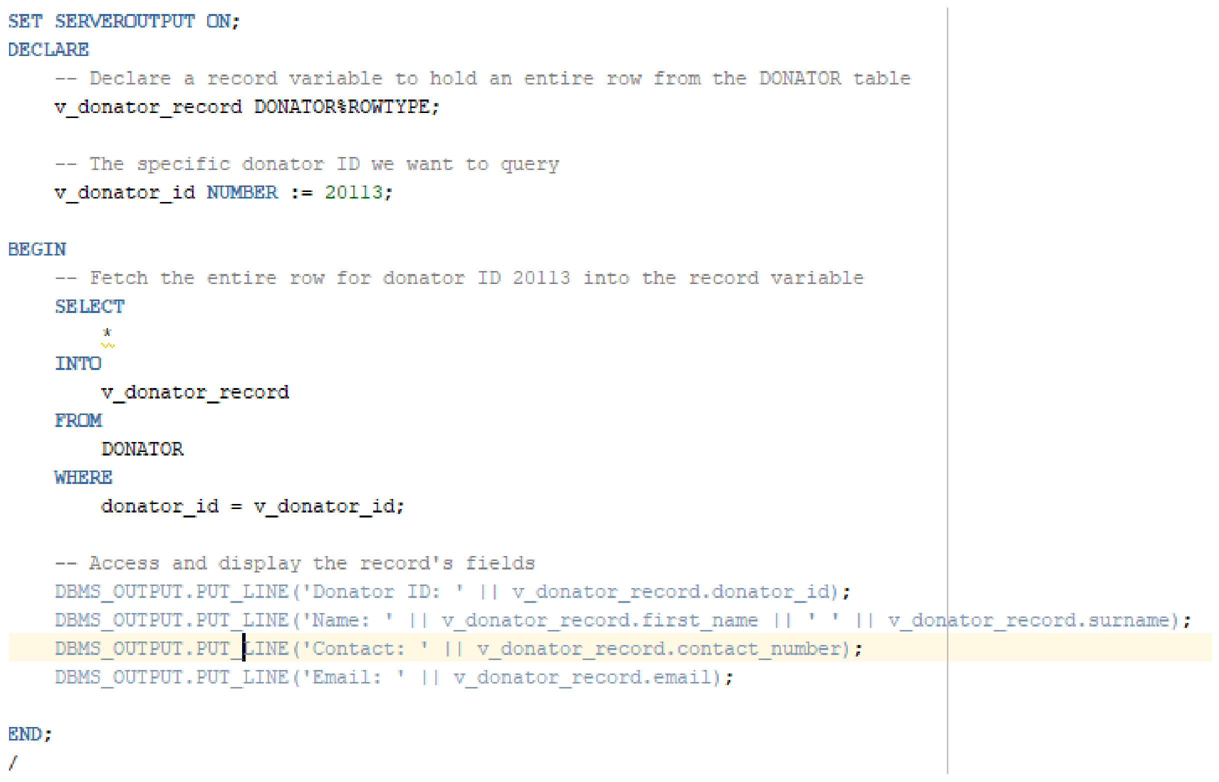


```
SET SERVEROUTPUT ON;
DECLARE
    -- Declare a variable for the customer's first name using %TYPE
    v_customer_first_name CUSTOMER.FIRST_NAME%TYPE;

    -- Declare a variable for the customer's surname
    v_customer_surname CUSTOMER.SURNAME%TYPE;

    -- Declare a variable for the customer ID
    v_customer_id NUMBER := 11011;

BEGIN
    -- Select the name of the customer with ID 11011 into the variables
    SELECT
        first_name,
        surname
    INTO
        v_customer_first_name,
        v_customer_surname
    FROM
        CUSTOMER
    WHERE
        customer_id = v_customer_id;

    -- Display the result
    DBMS_OUTPUT.PUT_LINE('Customer Full Name: ' || v_customer_first_name || ' ' || v_customer_surname);
END;
/
```

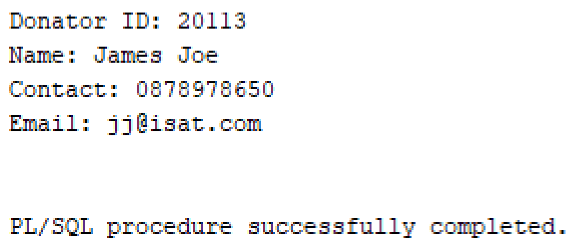


```
Customer Full Name: Jack Smith


PL/SQL procedure successfully completed.
```

7.2

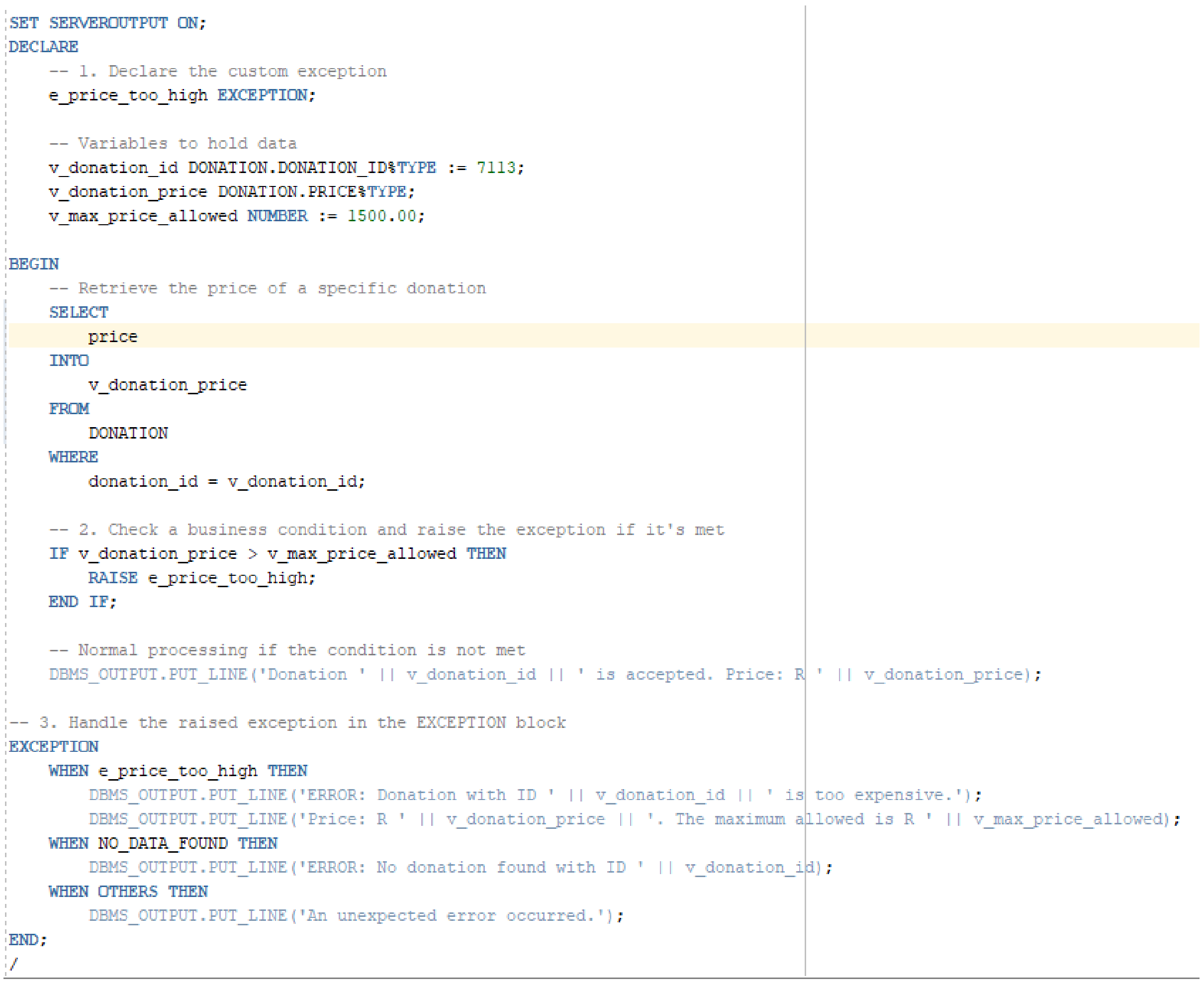


```
SET SERVEROUTPUT ON;
DECLARE
    -- Declare a record variable to hold an entire row from the DONATOR table
    v_donator_record DONATOR%ROWTYPE;

    -- The specific donator ID we want to query
    v_donator_id NUMBER := 20113;

BEGIN
    -- Fetch the entire row for donator ID 20113 into the record variable
    SELECT
        *
    INTO
        v_donator_record
    FROM
        DONATOR
    WHERE
        donator_id = v_donator_id;

    -- Access and display the record's fields
    DBMS_OUTPUT.PUT_LINE('Donator ID: ' || v_donator_record.donator_id);
    DBMS_OUTPUT.PUT_LINE('Name: ' || v_donator_record.first_name || ' ' || v_donator_record.surname);
    DBMS_OUTPUT.PUT_LINE('Contact: ' || v_donator_record.contact_number);
    DBMS_OUTPUT.PUT_LINE('Email: ' || v_donator_record.email);

END;
/
```

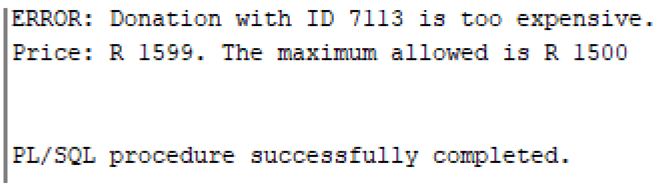


```
Donator ID: 20113
Name: James Joe
Contact: 0878978650
Email: jj@isat.com


PL/SQL procedure successfully completed.
```

7.3

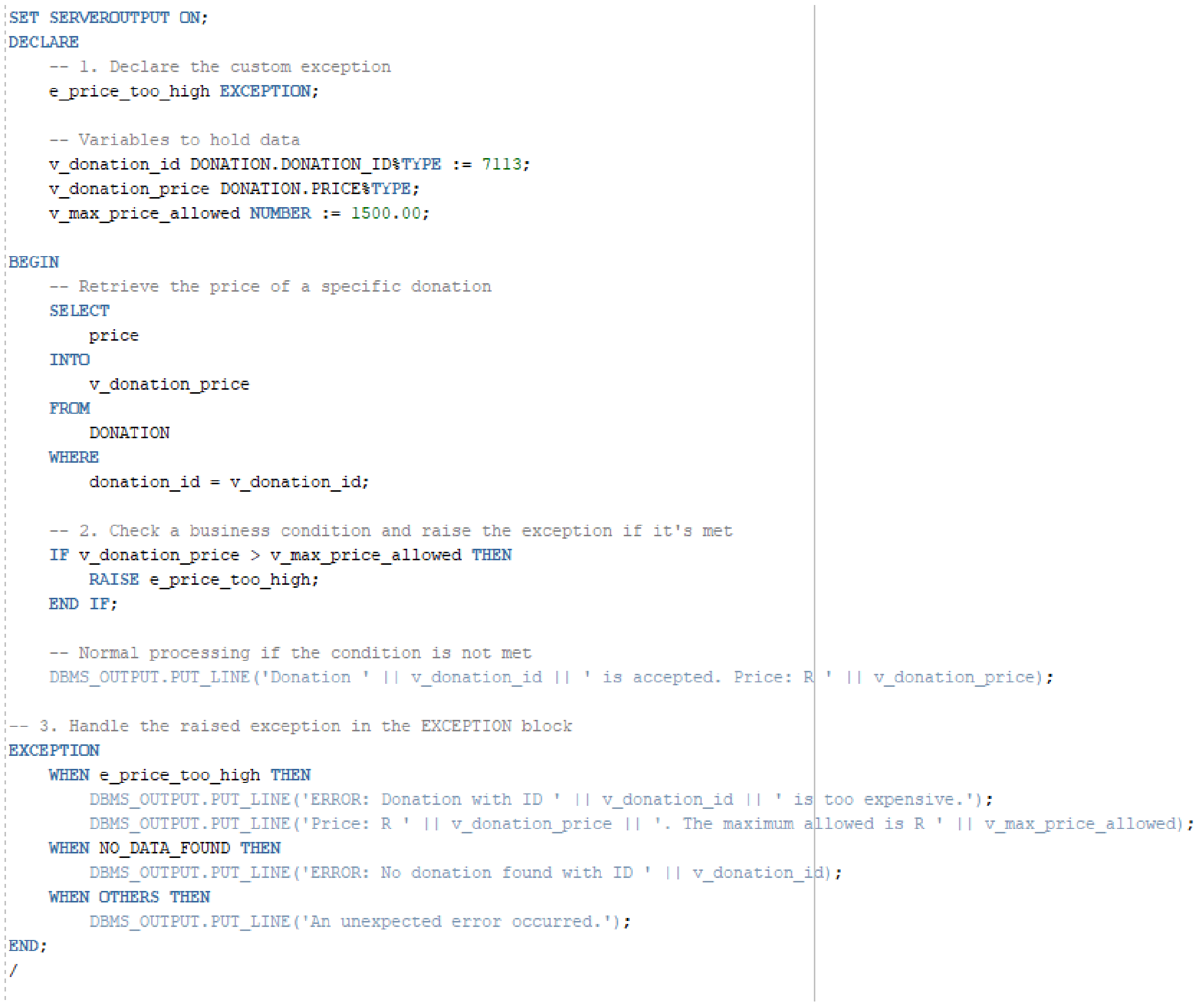


```sql
SET SERVEROUTPUT ON;
DECLARE
    -- 1. Declare the custom exception
    e_price_too_high EXCEPTION;

    -- Variables to hold data
    v_donation_id DONATION.DONATION_ID%TYPE := 7113;
    v_donation_price DONATION.PRICE%TYPE;
    v_max_price_allowed NUMBER := 1500.00;

BEGIN
    -- Retrieve the price of a specific donation
    SELECT
        price
    INTO
        v_donation_price
    FROM
        DONATION
    WHERE
        donation_id = v_donation_id;

    -- 2. Check a business condition and raise the exception if it's met
    IF v_donation_price > v_max_price_allowed THEN
        RAISE e_price_too_high;
    END IF;

    -- Normal processing if the condition is not met
    DBMS_OUTPUT.PUT_LINE('Donation ' || v_donation_id || ' is accepted. Price: R ' || v_donation_price);

-- 3. Handle the raised exception in the EXCEPTION block
EXCEPTION
    WHEN e_price_too_high THEN
        DBMS_OUTPUT.PUT_LINE('ERROR: Donation with ID ' || v_donation_id || ' is too expensive.');
        DBMS_OUTPUT.PUT_LINE('Price: R ' || v_donation_price || '. The maximum allowed is R ' || v_max_price_allowed);
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('ERROR: No donation found with ID ' || v_donation_id);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred.');
END;
/
```

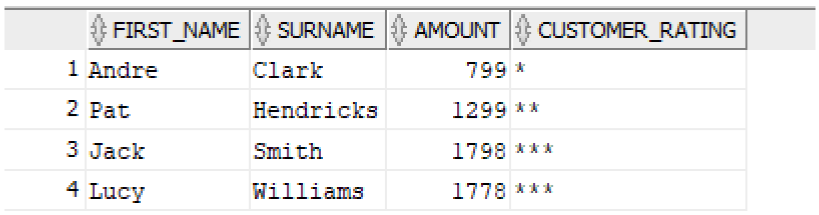


```
ERROR: Donation with ID 7113 is too expensive.
Price: R 1599. The maximum allowed is R 1500


PL/SQL procedure successfully completed.
```

Question 8



```
SET SERVEROUTPUT ON;
DECLARE
    -- 1. Declare the custom exception
    e_price_too_high EXCEPTION;

    -- Variables to hold data
    v_donation_id DONATION.DONATION_ID%TYPE := 7113;
    v_donation_price DONATION.PRICE%TYPE;
    v_max_price_allowed NUMBER := 1500.00;

BEGIN
    -- Retrieve the price of a specific donation
    SELECT
        price
    INTO
        v_donation_price
    FROM
        DONATION
    WHERE
        donation_id = v_donation_id;

    -- 2. Check a business condition and raise the exception if it's met
    IF v_donation_price > v_max_price_allowed THEN
        RAISE e_price_too_high;
    END IF;

    -- Normal processing if the condition is not met
    DBMS_OUTPUT.PUT_LINE('Donation ' || v_donation_id || ' is accepted. Price: R ' || v_donation_price);

-- 3. Handle the raised exception in the EXCEPTION block
EXCEPTION
    WHEN e_price_too_high THEN
        DBMS_OUTPUT.PUT_LINE('ERROR: Donation with ID ' || v_donation_id || ' is too expensive.');
        DBMS_OUTPUT.PUT_LINE('Price: R ' || v_donation_price || '. The maximum allowed is R ' || v_max_price_allowed);
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('ERROR: No donation found with ID ' || v_donation_id);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred.');
END;
/
```



	FIRST_NAME	SURNAME	AMOUNT	CUSTOMER_RATING
1	Andre	Clark	799	*
2	Pat	Hendricks	1299	**
3	Jack	Smith	1798	***
4	Lucy	Williams	1778	***