ADDB7311 Practical Assignment 2

ST10261338 Matthew Westermeyer

--Question 1--

```
--creating tables
CREATE TABLE Customer (
    Customer_ID INT PRIMARY KEY,
    First_Name VARCHAR(50) NOT NULL,
    Surname VARCHAR(50) NOT NULL,
    Address VARCHAR(50),
    Contact_Number VARCHAR(20),
    Email VARCHAR(50)
);

CREATE TABLE Employee (
    Employee_ID VARCHAR(10) PRIMARY KEY,
    First_Name VARCHAR(50) NOT NULL,
    Surname VARCHAR(50) NOT NULL,
    Contact_Number VARCHAR(20),
    Address VARCHAR(50),
    Email VARCHAR(50)
);

CREATE TABLE Donator (
    Donator_ID INT PRIMARY KEY,
    First_Name VARCHAR(50) NOT NULL,
    Surname VARCHAR(50) NOT NULL,
```

```
    Contact_Number VARCHAR(20),
    Email VARCHAR(50)
);

CREATE TABLE Donation (
    Donation_ID INT PRIMARY KEY,
    Donator_ID INT,
    Donation VARCHAR(50),
    Price INT,
    Donation_Date DATE,
    FOREIGN KEY (Donator_ID) REFERENCES Donator(Donator_ID)
);

CREATE TABLE Delivery (
    Delivery_ID INT PRIMARY KEY,
    Delivery_Notes VARCHAR(100),
    Dispatch_Date DATE,
    Delivery_Date DATE
);

CREATE TABLE Returns (
    Return_ID VARCHAR(10) PRIMARY KEY,
    Return_Date DATE,
    Reason VARCHAR(250),
    Customer_ID INT,
    Donation_ID INT,
    Employee_ID VARCHAR(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)
);

CREATE TABLE Invoice (
    Invoice_Num INT PRIMARY KEY,
    Customer_ID INT,
    Invoice_Date DATE,
    Employee_ID VARCHAR(10),
    Donation_ID INT,
    Delivery_ID INT,
    FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID),
    FOREIGN KEY (Employee_ID) REFERENCES Employee(Employee_ID),
    FOREIGN KEY (Donation_ID) REFERENCES Donation(Donation_ID),
    FOREIGN KEY (Delivery_ID) REFERENCES Delivery(Delivery_ID)
);



--Adding Values


INSERT ALL
INTO Customer VALUES (11011, 'Jack', 'Smith', '18 Water Rd', '0877277521',
        'jsmith@isat.com')
INTO Customer VALUES (11012, 'Pat', 'Hendricks', '22 Water Rd', '0863277857',
        'ph@mcom.co.za')
INTO Customer VALUES (11013, 'Andre', 'Clark', '101 Summer Lane', '0834567891',
        'aclark@mcom.co.za')
INTO Customer VALUES (11014, 'Kevin', 'Jones', '55 Mountain Way', '0612547895',
        'kj@isat.co.za')
INTO Customer VALUES (11015, 'Lucy', 'Williams', '5 Main Rd', '0827238521',
        'lw@mcal.co.za')
SELECT * FROM DUAL;
```

```
INSERT ALL
INTO Employee VALUES ('emp101', 'Jeff', 'Davis', '0877277521', '10 Main Road',
        'jand@isat.com')
INTO Employee VALUES ('emp102', 'Kevin', 'Marks', '0833777522', '18 Water Road',
        'km@isat.com')
INTO Employee VALUES ('emp103', 'Adanya', 'Andrews', '0817117523', '21 Circle Lane',
        'aa@isat.com')
INTO Employee VALUES ('emp104', 'Adebayo', 'Dryer', '0797215244', '1 Sea Road',
        'aryer@isat.com')
INTO Employee VALUES ('emp105', 'Xolani', 'Samson', '0827122255', '12 Main Road',
        'xosam@isat.com')
SELECT * FROM DUAL;


-- Insert data into Donator table
INSERT ALL
INTO Donator VALUES (20111, 'Jeff', 'Watson', '0827172250', 'jwatson@ymail.com')
INTO Donator VALUES (20112, 'Stephen', 'Jones', '0837865670', 'joness@gmail.com')
INTO Donator VALUES (20113, 'James', 'Joe', '0878978650', 'jj@isat.com')
INTO Donator VALUES (20114, 'Kelly', 'Ross', '0826575650', 'kross@gsat.com')
INTO Donator VALUES (20115, 'Abraham', 'Clark', '0797656430', 'aclark@ymail.com')
SELECT * FROM DUAL;


-- Insert data into Donation table
INSERT ALL
INTO Donation VALUES (7111, 20111, 'KIC Fridge', 599, TO_DATE('01-May-2024', 'DD-MON-
        YYYY'))
INTO Donation VALUES (7112, 20112, 'Samsung 42inch LCD', 1299, TO_DATE('03-May-2024',
        'DD-MON-YYYY'))
```

```
INTO Donation VALUES (7113, 20113, 'Sharp Microwave', 1599, TO_DATE('04-May-2024',
    'DD-MON-YYYY'))

INTO Donation VALUES (7114, 20115, '6 Seat Dining Room Table', 799, TO_DATE('05-May-
    2024', 'DD-MON-YYYY'))

INTO Donation VALUES (7115, 20114, 'Lazyboy Sofa', 1199, TO_DATE('07-May-2024', 'DD-
    MON-YYYY'))

INTO Donation VALUES (7116, 20113, 'JVC Surround Sound System', 179, TO_DATE('09-May-
    2024', 'DD-MON-YYYY'))

SELECT * FROM DUAL;



-- Insert data into Delivery table

INSERT ALL

INTO Delivery VALUES (511, 'Double packaging requested', TO_DATE('10-May-2024', 'DD-
    MON-YYYY'), TO_DATE('15-May-2024', 'DD-MON-YYYY'))

INTO Delivery VALUES (512, 'Delivery to work address', TO_DATE('12-May-2024', 'DD-MON-
    YYYY'), TO_DATE('15-May-2024', 'DD-MON-YYYY'))

INTO Delivery VALUES (513, 'Signature required', TO_DATE('12-May-2024', 'DD-MON-YYYY'),
    TO_DATE('17-May-2024', 'DD-MON-YYYY'))

INTO Delivery VALUES (514, 'No notes', TO_DATE('12-May-2024', 'DD-MON-YYYY'),
    TO_DATE('15-May-2024', 'DD-MON-YYYY'))

INTO Delivery VALUES (515, 'Birthday present wrapping required', TO_DATE('18-May-2024',
    'DD-MON-YYYY'), TO_DATE('19-May-2024', 'DD-MON-YYYY'))

INTO Delivery VALUES (516, 'Delivery to work address', TO_DATE('20-May-2024', 'DD-MON-
    YYYY'), TO_DATE('25-May-2024', 'DD-MON-YYYY'))

SELECT * FROM DUAL;



-- Insert data into Returns table

INSERT ALL
```

```
INTO Returns VALUES ('ret001', TO_DATE('25-May-2024', 'DD-MON-YYYY'), 'Customer not
        satisfied with product', 11011, 7116, 'emp101')

INTO Returns VALUES ('ret002', TO_DATE('25-May-2024', 'DD-MON-YYYY'), 'Product had
        broken section', 11013, 7114, 'emp103')

SELECT * FROM DUAL;




-- Insert data into Invoice table

INSERT ALL

INTO Invoice VALUES (8111, 11011, TO_DATE('15-May-2024', 'DD-MON-YYYY'), 'emp103',
        7111, 511)

INTO Invoice VALUES (8112, 11013, TO_DATE('15-May-2024', 'DD-MON-YYYY'), 'emp101',
        7114, 512)

INTO Invoice VALUES (8113, 11012, TO_DATE('17-May-2024', 'DD-MON-YYYY'), 'emp101',
        7112, 513)

INTO Invoice VALUES (8114, 11015, TO_DATE('17-May-2024', 'DD-MON-YYYY'), 'emp102',
        7113, 514)

INTO Invoice VALUES (8115, 11011, TO_DATE('17-May-2024', 'DD-MON-YYYY'), 'emp102',
        7115, 515)

INTO Invoice VALUES (8116, 11015, TO_DATE('18-May-2024', 'DD-MON-YYYY'), 'emp104',
        7116, 516)

SELECT * FROM DUAL;
```

--Question 2--

```
--question 2

SELECT

   c.First_Name || ' ' || c.Surname AS Customer_Name,

   i.Employee_ID,

   d.Delivery_Notes,

   n.Donation AS Donation_Purchased,
```

```
    i.Invoice_Num,
    i.Invoice_Date
FROM
    Customer c
JOIN
    Invoice i ON c.Customer_ID = i.Customer_ID
JOIN
    Delivery d ON i.Delivery_ID = d.Delivery_ID
JOIN
    Donation n ON i.Donation_ID = n.Donation_ID
WHERE
    i.Invoice_Date > TO_DATE('16-May-2024', 'DD-MON-YYYY');
```

--Question 3--

```
--question 3
-- Create the Funding table
CREATE TABLE Funding (
    funding_id INT PRIMARY KEY,
    funder VARCHAR(100) NOT NULL,
    funding_amount DECIMAL(10, 2) NOT NULL
);

-- Create a sequence to generate unique funding IDs
CREATE SEQUENCE funding_id_seq START WITH 1
INCREMENT BY 1  -- Increment by 1 for each new record
NOCACHE;  -- Do not cache numbers

-- Insert example record into the Funding table
```

```
INSERT INTO Funding (funding_id, funder, funding_amount)
VALUES (funding_id_seq.NEXTVAL, 'Charity Foundation', 5000.00);
```

--This solution generates the funding id which helps against the risk of two funding ids being the same due to human error

```
Table FUNDING created.

Sequence FUNDING_ID_SEQ created.

1 row inserted.

PL/SQL procedure successfully completed.
```

--Question 4--

```
--question 4


--report generator
SET SERVEROUTPUT ON;


DECLARE
   -- Declare a cursor to fetch the required data
   CURSOR c_donations IS
      SELECT
         c.First_Name || ' ' || c.Surname AS Customer_Name,
         d.Donation AS Donation_Purchased,
         d.Price AS Donation_Price,
         r.Reason AS Return_Reason
      FROM
         Customer c
      JOIN
         Returns r ON c.Customer_ID = r.Customer_ID
      JOIN
         Donation d ON r.Donation_ID = d.Donation_ID;
```

```
    -- Variables to hold the fetched data
    v_customer_name VARCHAR(100);
    v_donation_purchased VARCHAR(100);
    v_donation_price DECIMAL(10, 2);
    v_return_reason VARCHAR(250);

BEGIN
    -- Open the cursor
    FOR record IN c_donations LOOP
        -- Fetch the data into variables
        v_customer_name := record.Customer_Name;
        v_donation_purchased := record.Donation_Purchased;
        v_donation_price := record.Donation_Price;
        v_return_reason := record.Return_Reason;

        -- Display the results
        -- Output each field with formatting
        DBMS_OUTPUT.PUT_LINE('Customer Name:       ' || v_customer_name);
        DBMS_OUTPUT.PUT_LINE('Donation Purchased:  ' || v_donation_purchased);
        DBMS_OUTPUT.PUT_LINE('Donation Price:      R' || v_donation_price);
        DBMS_OUTPUT.PUT_LINE('Return Reason:       ' || v_return_reason);
        DBMS_OUTPUT.PUT_LINE('-----------------------------------------------------');
    END LOOP;
END;
```

```
Customer Name:        Jack Smith
Donation Purchased:   JVC Surround Sound System
Donation Price:       R179
Return Reason:        Customer not satisfied with product
-----------------------------------------------------
Customer Name:        Andre Clark
Donation Purchased:   6 Seat Dining Room Table
Donation Price:       R799
Return Reason:        Product had broken section
-----------------------------------------------------


PL/SQL procedure successfully completed.
```

--Question 5--

```
--question 5
DECLARE
   -- Declare a cursor to fetch the required data
   CURSOR c_delivery IS
      SELECT
         c.First_Name || ' ' || c.Surname AS Customer_Name,
         e.First_Name || ' ' || e.Surname AS Employee_Name,
         d.Donation AS Donation_Purchased,
         del.Dispatch_Date,
         del.Delivery_Date,
         (del.Delivery_Date - del.Dispatch_Date) AS Days_Between
      FROM
         Customer c
      JOIN
         Invoice i ON c.Customer_ID = i.Customer_ID
      JOIN
         Donation d ON i.Donation_ID = d.Donation_ID
      JOIN
         Delivery del ON i.Delivery_ID = del.Delivery_ID
      JOIN
         Employee e ON i.Employee_ID = e.Employee_ID
      WHERE
         c.Customer_ID = 11011;  -- Filter for customer 11011

   -- Variables to hold the fetched data
   v_customer_name VARCHAR(100);
   v_employee_name VARCHAR(100);
   v_donation_purchased VARCHAR(100);
```

```
    v_dispatch_date DATE;
    v_delivery_date DATE;
    v_days_between INT;

BEGIN
    -- Open the cursor
    FOR record IN c_delivery LOOP
        -- Fetch the data into variables
        v_customer_name := record.Customer_Name;
        v_employee_name := record.Employee_Name;
        v_donation_purchased := record.Donation_Purchased;
        v_dispatch_date := record.Dispatch_Date;
        v_delivery_date := record.Delivery_Date;
        v_days_between := record.Days_Between;

        -- Display the results
        DBMS_OUTPUT.PUT_LINE('Customer Name:       ' || v_customer_name);
        DBMS_OUTPUT.PUT_LINE('Employee Name:       ' || v_employee_name);
        DBMS_OUTPUT.PUT_LINE('Donation Purchased:  ' || v_donation_purchased);
        DBMS_OUTPUT.PUT_LINE('Dispatch Date:       ' || v_dispatch_date);
        DBMS_OUTPUT.PUT_LINE('Delivery Date:       ' || v_delivery_date);
        DBMS_OUTPUT.PUT_LINE('Days Between:        ' || v_days_between);
        DBMS_OUTPUT.PUT_LINE('-----------------------------------------------------');
    END LOOP;
END;
```

```
PL/SQL procedure successfully completed.

Customer Name:        Jack Smith
Employee Name:        Adanya Andrews
Donation Purchased:   KIC Fridge
Dispatch Date:        10-MAY-24
Delivery Date:        15-MAY-24
Days Between:         5
------------------------------------------------------------
Customer Name:        Jack Smith
Employee Name:        Kevin Marks
Donation Purchased:   Lazyboy Sofa
Dispatch Date:        18-MAY-24
Delivery Date:        19-MAY-24
Days Between:         1
------------------------------------------------------------

PL/SQL procedure successfully completed.
```

--Question 6--

```
--question 6
DECLARE
   -- Cursor to fetch the customer names and their total amount spent
   CURSOR c_customers IS
      SELECT
         c.First_Name AS Customer_Name,
         c.Surname AS Customer_Surname,
         SUM(d.Price) AS Total_Amount_Spent
      FROM
         Customer c
      JOIN
         Invoice i ON c.Customer_ID = i.Customer_ID
      JOIN
         Donation d ON i.Donation_ID = d.Donation_ID
      GROUP BY
         c.First_Name, c.Surname;

   -- Variables to hold the fetched data
   v_customer_name VARCHAR(100);
   v_customer_surname VARCHAR(100);
   v_total_amount_spent DECIMAL(10, 2);
   v_customer_rating VARCHAR(20);
```

```
BEGIN
   -- Open the cursor
   FOR record IN c_customers LOOP
      -- Fetch the data into variables
      v_customer_name := record.Customer_Name;
      v_customer_surname := record.Customer_Surname;
      v_total_amount_spent := record.Total_Amount_Spent;

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

      -- Display the formatted results
      DBMS_OUTPUT.PUT_LINE('Customer Name:       ' || v_customer_name);
      DBMS_OUTPUT.PUT_LINE('Customer Surname:    ' || v_customer_surname);
      DBMS_OUTPUT.PUT_LINE('Total Amount Spent:  R' || v_total_amount_spent ||
         v_customer_rating);
      DBMS_OUTPUT.PUT_LINE('-----------------------------------------------------');
   END LOOP;
END;
```

```
Customer Name:       Jack
Customer Surname:    Smith
Total Amount Spent:  R1798(***)
-----------------------------------------------------
Customer Name:       Pat
Customer Surname:    Hendricks
Total Amount Spent:  R1299
-----------------------------------------------------
Customer Name:       Andre
Customer Surname:    Clark
Total Amount Spent:  R799
-----------------------------------------------------
Customer Name:       Lucy
Customer Surname:    Williams
Total Amount Spent:  R1778(***)
-----------------------------------------------------


PL/SQL procedure successfully completed.
```

--Question 7.1--

--Question 7

--The %TYPE attribute allows you to define a variable with the same data type as a column in a table (Oracle Corporation, n.d).

--Code example for 7.1

```
DECLARE
   -- Declare a variable using %TYPE to match the data type of the Email column in the Customer table
   v_email Customer.Email%TYPE;
BEGIN
   -- Assign a value to the variable
   v_email := 'new_email@example.com';

   -- Output the email
   DBMS_OUTPUT.PUT_LINE('Customer Email: ' || v_email);
END;
```

```
Customer Email: new_email@example.com

PL/SQL procedure successfully completed.
```

--Question 7.2--

--The %ROWTYPE attribute allows you to declare a record variable that can hold an entire row of data from a table (Oracle Corporation, n.d).

--code example for 7.2

```
DECLARE
   -- Declare a record variable using %ROWTYPE for the Customer table
   v_customer_record Customer%ROWTYPE;
BEGIN
   -- Select a single customer into the record variable
   SELECT * INTO v_customer_record
```

```
    FROM Customer
    WHERE Customer_ID = 11011;  -- Assuming 11011 exists in the Customer table

    -- Output the customer's name and email
    DBMS_OUTPUT.PUT_LINE('Customer Name: ' || v_customer_record.First_Name || ' ' ||
        v_customer_record.Surname);
    DBMS_OUTPUT.PUT_LINE('Customer Email: ' || v_customer_record.Email);
END;
/
```

```
Customer Name: Jack Smith
Customer Email: jsmith@isat.com

PL/SQL procedure successfully completed.
```

--Question 7.3--

--A user-defined exception allows you to create custom exceptions for specific error scenarios (Oracle Corporation, n.d).

```
--code example for 7.3
DECLARE
    -- Declare a user-defined exception
    customer_not_found EXCEPTION;

    -- Declare a variable for Customer ID
    v_customer_id INT := 99999; -- Example of a non-existent customer ID
    v_customer_email Customer.Email%TYPE; -- To hold the email of the customer
BEGIN
    -- Try to select a customer with a potentially invalid ID
    SELECT Email INTO v_customer_email
    FROM Customer
    WHERE Customer_ID = v_customer_id;
```

```
    -- If no customer is found, raise the user-defined exception
    RAISE customer_not_found;

EXCEPTION
    WHEN customer_not_found THEN
        -- Handle the custom exception
        DBMS_OUTPUT.PUT_LINE('Error: Customer with ID ' || v_customer_id || ' does not
            exist.');
    WHEN NO_DATA_FOUND THEN
        -- Handle the standard NO_DATA_FOUND exception
        DBMS_OUTPUT.PUT_LINE('Error: No data found for Customer ID ' || v_customer_id);
    WHEN OTHERS THEN
        -- Handle any other exceptions
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
END;
```

```
Error: No data found for Customer ID 99999

PL/SQL procedure successfully completed.
```

--Question 8--

```
--question 8
SELECT
    c.First_Name AS Customer_Name,
    c.Surname AS Customer_Surname,
    SUM(d.Price) AS Total_Amount_Spent,
    CASE
        WHEN SUM(d.Price) >= 1500 THEN '(***)'
        WHEN SUM(d.Price) >= 1000 AND SUM(d.Price) < 1500 THEN '(**)'
        ELSE '(*)'
    END AS Customer_Rating
FROM
```

```
    Customer c
JOIN
    Invoice i ON c.Customer_ID = i.Customer_ID
JOIN
    Donation d ON i.Donation_ID = d.Donation_ID
GROUP BY
    c.First_Name, c.Surname;
```

	CUSTOMER_NAME	CUSTOMER_SURNAME	TOTAL_AMOUNT_SPENT	CUSTOMER_RATING
1	Jack	Smith	1798	(***)
2	Pat	Hendricks	1299	(**)
3	Andre	Clark	799	(*)
4	Lucy	Williams	1778	(***)

References

Oracle Corporation, n.d. *Database PL/SQL Language Reference.* [Online]
Available at: https://docs.oracle.com/en/database/oracle/oracle-database/19/lnpls/TYPE-attribute.html
[Accessed 07 October 2024].