ADDB7311 Assignment 2

Vinay Roopchun St10258101

Contents

Question 1

Tables created successfully

```sql
-- Customer table
CREATE TABLE CUSTOMER (
    CUSTOMER_ID NUMBER PRIMARY KEY,
    FIRST_NAME VARCHAR2(50),
    SURNAME VARCHAR2(50),
    ADDRESS VARCHAR2(100),
    CONTACT_NUMBER VARCHAR2(15),
    EMAIL VARCHAR2(50)
);

-- Employee table
CREATE TABLE EMPLOYEE (
    EMPLOYEE_ID VARCHAR2(10) PRIMARY KEY,
    FIRST_NAME VARCHAR2(50),
    SURNAME VARCHAR2(50),
    CONTACT_NUMBER VARCHAR2(15),
    ADDRESS VARCHAR2(100),
    EMAIL VARCHAR2(50)
);

-- Donator table
CREATE TABLE DONATOR (
    DONATOR_ID NUMBER PRIMARY KEY,
    FIRST_NAME VARCHAR2(50),
    SURNAME VARCHAR2(50),
    CONTACT_NUMBER VARCHAR2(15),
    EMAIL VARCHAR2(50)
);
```

```sql
-- Donation table
CREATE TABLE DONATION (
    DONATION_ID NUMBER PRIMARY KEY,
    DONATOR_ID NUMBER,
    DONATION VARCHAR2(100),
    PRICE VARCHAR2(10),
    DONATION_DATE DATE,
    FOREIGN KEY (DONATOR_ID) REFERENCES DONATOR(DONATOR_ID)
);

-- Delivery table
CREATE TABLE DELIVERY (
    DELIVERY_ID NUMBER PRIMARY KEY,
    DELIVERY_NOTES VARCHAR2(255),
    DISPATCH_DATE DATE,
    DELIVERY_DATE DATE
);
```

```sql
-- Return table
CREATE TABLE RETURNS (
    RETURN_ID VARCHAR2(10) PRIMARY KEY,
    RETURN_DATE DATE,
    REASON VARCHAR2(255),
    CUSTOMER_ID NUMBER,
    DONATION_ID NUMBER,
    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)
);

-- Invoice table
CREATE TABLE INVOICE (
    INVOICE_NUM NUMBER PRIMARY KEY,
    CUSTOMER_ID NUMBER,
    INVOICE_DATE DATE,
    EMPLOYEE_ID VARCHAR2(10),
    DONATION_ID NUMBER,
    DELIVERY_ID NUMBER,
    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)
);
```

```
Table DONATOR created.

Table CUSTOMER created.

Table EMPLOYEE created.
```

```
Table DONATION created.

Table DELIVERY created.

Table RETURNS created.

Table INVOICE created.
```

```sql
-- Customer table
CREATE TABLE CUSTOMER (
    CUSTOMER_ID NUMBER PRIMARY KEY,
    FIRST_NAME VARCHAR2(50),
```

```
    SURNAME VARCHAR2(50),
    ADDRESS VARCHAR2(100),
    CONTACT_NUMBER VARCHAR2(15),
    EMAIL VARCHAR2(50)
);

-- Employee table
CREATE TABLE EMPLOYEE (
    EMPLOYEE_ID VARCHAR2(10) PRIMARY KEY,
    FIRST_NAME VARCHAR2(50),
    SURNAME VARCHAR2(50),
    CONTACT_NUMBER VARCHAR2(15),
    ADDRESS VARCHAR2(100),
    EMAIL VARCHAR2(50)
);

-- Donator table
CREATE TABLE DONATOR (
    DONATOR_ID NUMBER PRIMARY KEY,
    FIRST_NAME VARCHAR2(50),
    SURNAME VARCHAR2(50),
    CONTACT_NUMBER VARCHAR2(15),
    EMAIL VARCHAR2(50)
);

-- Donation table
CREATE TABLE DONATION (
    DONATION_ID NUMBER PRIMARY KEY,
    DONATOR_ID NUMBER,
    DONATION VARCHAR2(100),
    PRICE VARCHAR2(10),
    DONATION_DATE DATE,
    FOREIGN KEY (DONATOR_ID) REFERENCES DONATOR(DONATOR_ID)
);

-- Delivery table
CREATE TABLE DELIVERY (
    DELIVERY_ID NUMBER PRIMARY KEY,
    DELIVERY_NOTES VARCHAR2(255),
    DISPATCH_DATE DATE,
    DELIVERY_DATE DATE
);

-- Return table
CREATE TABLE RETURNS (
    RETURN_ID VARCHAR2(10) PRIMARY KEY,
    RETURN_DATE DATE,
    REASON VARCHAR2(255),
    CUSTOMER_ID NUMBER,
    DONATION_ID NUMBER,
    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)
);

-- Invoice table
CREATE TABLE INVOICE (
    INVOICE_NUM NUMBER PRIMARY KEY,
    CUSTOMER_ID NUMBER,
    INVOICE_DATE DATE,
    EMPLOYEE_ID VARCHAR2(10),
    DONATION_ID NUMBER,
    DELIVERY_ID NUMBER,
    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)
);
```

Tables populated successfully

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

-- EMPLOYEE table insert
INSERT INTO EMPLOYEE VALUES ('emp101', 'Jeff', 'Davis', '0877277521', '10 main road', 'jand@isat.com');
INSERT INTO EMPLOYEE VALUES ('emp102', 'Kevin', 'Marks', '0837377522', '18 water road', 'km@isat.com');
INSERT INTO EMPLOYEE VALUES ('emp103', 'Adanya', 'Andrews', '0817117523', '21 circle lane', 'aa@isat.com');
INSERT INTO EMPLOYEE VALUES ('emp104', 'Adebayo', 'Dryer', '0797215244', '1 sea road', 'aryer@isat.com');
INSERT INTO EMPLOYEE VALUES ('emp105', 'Xolani', 'Samson', '0827122255', '12 main road', 'xosam@isat.com');

-- DONATOR table insert
INSERT INTO DONATOR VALUES (20111, 'Jeff', 'Watson', '0827172250', 'jwatson@ymail.com');
INSERT INTO DONATOR VALUES (20112, 'Stephen', 'Jones', '0837865670', 'joness@ymail.com');
INSERT INTO DONATOR VALUES (20113, 'James', 'Joe', '0878978650', 'jj@isat.com');
INSERT INTO DONATOR VALUES (20114, 'Kelly', 'Ross', '0826575650', 'kross@gsat.com');
INSERT INTO DONATOR VALUES (20115, 'Abraham', 'Clark', '0797656430', 'aclark@ymail.com');
```

```
-- DONATION table insert
INSERT INTO DONATION VALUES (7111, 20111, 'KIC Fridge', 'R 599', TO_DATE('01-May-2024', 'DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (7112, 20112, 'Samsung 42inch LCD', 'R 1 299', TO_DATE('03-May-2024', 'DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (7113, 20113, 'Sharp Microwave', 'R 1 599', TO_DATE('03-May-2024', 'DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (7114, 20115, '6 Seat Dining room table', 'R 799', TO_DATE('05-May-2024', 'DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (7115, 20114, 'Lazyboy Sofa', 'R 1 199', TO_DATE('07-May-2024', 'DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (7116, 20113, 'JVC Surround Sound System', 'R 179', TO_DATE('09-May-2024', 'DD-Mon-YYYY'));

-- DELIVERY table insert
INSERT INTO DELIVERY VALUES (511, 'Double packaging requested', TO_DATE('10-May-2024', 'DD-Mon-YYYY'), TO_DATE('15-May-2024', 'DD-Mon-YYYY'));
INSERT INTO DELIVERY 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 VALUES (513, 'Signature required', TO_DATE('12-May-2024', 'DD-Mon-YYYY'), TO_DATE('17-May-2024', 'DD-Mon-YYYY'));
INSERT INTO DELIVERY VALUES (514, 'No notes', TO_DATE('12-May-2024', 'DD-Mon-YYYY'), TO_DATE('15-May-2024', 'DD-Mon-YYYY'));
INSERT INTO DELIVERY 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 VALUES (516, 'Delivery to work address', TO_DATE('20-May-2024', 'DD-Mon-YYYY'), TO_DATE('25-May-2024', 'DD-Mon-YYYY'));
```

```sql
-- DELIVERY table insert
INSERT INTO DELIVERY VALUES (511, 'Double packaging requested', TO_DATE('10-May-2024', 'DD-Mon-YYYY'), TO_DATE('15-May-2024', 'DD-Mon-YYYY'));
INSERT INTO DELIVERY 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 VALUES (513, 'Signature required', TO_DATE('12-May-2024', 'DD-Mon-YYYY'), TO_DATE('17-May-2024', 'DD-Mon-YYYY'));
INSERT INTO DELIVERY VALUES (514, 'No notes', TO_DATE('12-May-2024', 'DD-Mon-YYYY'), TO_DATE('15-May-2024', 'DD-Mon-YYYY'));
INSERT INTO DELIVERY 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 VALUES (516, 'Delivery to work address', TO_DATE('20-May-2024', 'DD-Mon-YYYY'), TO_DATE('25-May-2024', 'DD-Mon-YYYY'));

-- RETURNS table insert
INSERT INTO RETURNS VALUES ('ret001', TO_DATE('25-May-2024', 'DD-Mon-YYYY'), 'Customer not satisfied with product', 11011, 7116, 'emp101');
INSERT INTO RETURNS VALUES ('ret002', TO_DATE('25-May-2024', 'DD-Mon-YYYY'), 'Product had broken section', 11013, 7114, 'emp103');

-- INVOICE table insert
INSERT INTO INVOICE VALUES (8111, 11011, TO_DATE('15-May-2024', 'DD-Mon-YYYY'), 'emp103', 7111, 511);
INSERT INTO INVOICE VALUES (8112, 11013, TO_DATE('15-May-2024', 'DD-Mon-YYYY'), 'emp101', 7114, 512);
INSERT INTO INVOICE VALUES (8113, 11012, TO_DATE('17-May-2024', 'DD-Mon-YYYY'), 'emp101', 7112, 513);
INSERT INTO INVOICE VALUES (8114, 11015, TO_DATE('17-May-2024', 'DD-Mon-YYYY'), 'emp102', 7113, 514);
INSERT INTO INVOICE VALUES (8115, 11011, TO_DATE('17-May-2024', 'DD-Mon-YYYY'), 'emp102', 7115, 515);
INSERT INTO INVOICE VALUES (8116, 11015, TO_DATE('18-May-2024', 'DD-Mon-YYYY'), 'emp103', 7116, 516);
```

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

etc

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

-- EMPLOYEE table insert
INSERT INTO EMPLOYEE VALUES ('emp101', 'Jeff', 'Davis', '0877277521', '10 main road',
'jand@isat.com');
INSERT INTO EMPLOYEE VALUES ('emp102', 'Kevin', 'Marks', '0837377522', '18 water road',
'km@isat.com');
INSERT INTO EMPLOYEE VALUES ('emp103', 'Adanya', 'Andrews', '0817117523', '21 circle lane',
'aa@isat.com');
INSERT INTO EMPLOYEE VALUES ('emp104', 'Adebayo', 'Dryer', '0797215244', '1 sea road',
'aryer@isat.com');
INSERT INTO EMPLOYEE VALUES ('emp105', 'Xolani', 'Samson', '0827122255', '12 main road',
'xosam@isat.com');

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

-- DONATION table insert
INSERT INTO DONATION VALUES (7111, 20111, 'KIC Fridge', 'R 599', TO_DATE('01-May-2024', 'DD-
Mon-YYYY'));
INSERT INTO DONATION VALUES (7112, 20112, 'Samsung 42inch LCD', 'R 1 299', TO_DATE('03-
May-2024', 'DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (7113, 20113, 'Sharp Microwave', 'R 1 599', TO_DATE('03-May-
2024', 'DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (7114, 20115, '6 Seat Dining room table', 'R 799', TO_DATE('05-
May-2024', 'DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (7115, 20114, 'Lazyboy Sofa', 'R 1 199', TO_DATE('07-May-2024',
'DD-Mon-YYYY'));
INSERT INTO DONATION VALUES (7116, 20113, 'JVC Surround Sound System', 'R 179',
TO_DATE('09-May-2024', 'DD-Mon-YYYY'));

-- DELIVERY table insert
INSERT INTO DELIVERY VALUES (511, 'Double packaging requested', TO_DATE('10-May-2024', 'DD-
Mon-YYYY'), TO_DATE('15-May-2024', 'DD-Mon-YYYY'));
INSERT INTO DELIVERY 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 VALUES (513, 'Signature required', TO_DATE('12-May-2024', 'DD-Mon-YYYY'), TO_DATE('17-May-2024', 'DD-Mon-YYYY'));
INSERT INTO DELIVERY VALUES (514, 'No notes', TO_DATE('12-May-2024', 'DD-Mon-YYYY'), TO_DATE('15-May-2024', 'DD-Mon-YYYY'));
INSERT INTO DELIVERY 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 VALUES (516, 'Delivery to work address', TO_DATE('20-May-2024', 'DD-Mon-YYYY'), TO_DATE('25-May-2024', 'DD-Mon-YYYY'));

-- RETURNS table insert
INSERT INTO RETURNS VALUES ('ret001', TO_DATE('25-May-2024', 'DD-Mon-YYYY'), 'Customer not satisfied with product', 11011, 7116, 'emp101');
INSERT INTO RETURNS VALUES ('ret002', TO_DATE('25-May-2024', 'DD-Mon-YYYY'), 'Product had broken section', 11013, 7114, 'emp103');

-- INVOICE table insert
INSERT INTO INVOICE VALUES (8111, 11011, TO_DATE('15-May-2024', 'DD-Mon-YYYY'), 'emp103', 7111, 511);
INSERT INTO INVOICE VALUES (8112, 11013, TO_DATE('15-May-2024', 'DD-Mon-YYYY'), 'emp101', 7114, 512);
INSERT INTO INVOICE VALUES (8113, 11012, TO_DATE('17-May-2024', 'DD-Mon-YYYY'), 'emp101', 7112, 513);
INSERT INTO INVOICE VALUES (8114, 11015, TO_DATE('17-May-2024', 'DD-Mon-YYYY'), 'emp102', 7113, 514);
INSERT INTO INVOICE VALUES (8115, 11011, TO_DATE('17-May-2024', 'DD-Mon-YYYY'), 'emp102', 7115, 515);
INSERT INTO INVOICE VALUES (8116, 11015, TO_DATE('18-May-2024', 'DD-Mon-YYYY'), 'emp103', 7116, 516);
```

Question 2

```
COLUMN CUSTOMER FORMAT A20
COLUMN EMPLOYEE_ID FORMAT A10
COLUMN DELIVERY_NOTES FORMAT A30
COLUMN DONATION FORMAT A25
COLUMN INVOICE_NUM FORMAT A10
COLUMN INVOICE_DATE FORMAT A15

SELECT
    C.FIRST_NAME || ', ' || C.SURNAME AS CUSTOMER,
    I.EMPLOYEE_ID,
    D.DELIVERY_NOTES,
    DN.DONATION,
    I.INVOICE_NUM,
    TO_CHAR(I.INVOICE_DATE, 'DD/MON/YYYY') AS 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')
ORDER BY
    I.INVOICE_DATE;
```

```
CUSTOMER             EMPLOYEE_I DELIVERY_NOTES                 DONATION                  INVOICE_NU INVOICE_DATE
-------------------- ---------- ------------------------------ ------------------------- ---------- ---------------
Pat, Hendricks       emp101     Signature required             Samsung 42inch LCD              8113 17/MAY/2024
Lucy, Williams       emp102     No notes                       Sharp Microwave                 8114 17/MAY/2024
Jack, Smith          emp102     Birthday present wrapping requ Lazyboy Sofa                    8115 17/MAY/2024
                                ired

Lucy, Williams       emp103     Delivery to work address       JVC Surround Sound System       8116 18/MAY/2024
```


COLUMN CUSTOMER FORMAT A20
COLUMN EMPLOYEE_ID FORMAT A10
COLUMN DELIVERY_NOTES FORMAT A30
COLUMN DONATION FORMAT A25
COLUMN INVOICE_NUM FORMAT A10
COLUMN INVOICE_DATE FORMAT A15

SELECT
  C.FIRST_NAME || ', ' || C.SURNAME AS CUSTOMER,
  I.EMPLOYEE_ID,
  D.DELIVERY_NOTES,
  DN.DONATION,
  I.INVOICE_NUM,
  TO_CHAR(I.INVOICE_DATE, 'DD/MON/YYYY') AS 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')
ORDER BY
   I.INVOICE_DATE;
```

Question 3

- Create the new table.

```
CREATE TABLE Funding (
    funding_id NUMBER PRIMARY KEY,
    funder VARCHAR2(100),
    funding_amount NUMBER(10, 2)
);
```

```
Table FUNDING created.
```

```
CREATE TABLE Funding (
   funding_id NUMBER PRIMARY KEY,
   funder VARCHAR2(100),
   funding_amount NUMBER(10, 2)
);
```

- Implement a solution to automatically generate the unique ids with every new insert.

```
orksheet    Query Builder
CREATE SEQUENCE funding_sequence
    START WITH 1
    INCREMENT BY 1
    NOCACHE;
```

```
Sequence FUNDING_SEQUENCE created.
```

```
CREATE SEQUENCE funding_sequence
   START WITH 1
   INCREMENT BY 1
   NOCACHE;
```

```
CREATE OR REPLACE TRIGGER funding_before_insert
    BEFORE INSERT ON Funding
    FOR EACH ROW
BEGIN
    :NEW.funding_id := funding_sequence.NEXTVAL;
END;
```

```
Trigger FUNDING_BEFORE_INSERT compiled
```

```
CREATE OR REPLACE TRIGGER funding_before_insert
   BEFORE INSERT ON Funding
   FOR EACH ROW
BEGIN
   :NEW.funding_id := funding_sequence.NEXTVAL;
END;
```

- Provide an example of the insert statement.

```
INSERT INTO Funding (funder, funding_amount)
VALUES ('John Doe', 1000.00);
```

```
1 row inserted.
```

```
INSERT INTO Funding (funder, funding_amount)
VALUES ('John Doe', 1000.00);
```

- Add a brief comment to justify your solution.

The solution uses a sequence to generate a unique funding_id for each record, and a trigger to automatically assign this ID whenever a new record is inserted. This approach ensures that each funding_id is unique and consistently incremented, reducing the risk of errors and simplifying the data insertion process. Using a sequence and trigger is an efficient way to handle auto-generated IDs in Oracle databases.

Question 4

```oracle
DECLARE
    v_output VARCHAR2(1000);
BEGIN
    FOR rec IN (SELECT
                    'CUSTOMER: ' || C.FIRST_NAME || ', ' || C.SURNAME || CHR(10) ||
                    'DONATION PURCHASED: ' || DN.DONATION || CHR(10) ||
                    'PRICE: ' || DN.PRICE || CHR(10) ||
                    'RETURN REASON: ' || R.REASON || CHR(10) ||
                    '-----------------------------------------' AS OUTPUT
                FROM RETURNS R
                JOIN CUSTOMER C ON R.CUSTOMER_ID = C.CUSTOMER_ID
                JOIN DONATION DN ON R.DONATION_ID = DN.DONATION_ID)
    LOOP
        v_output := rec.OUTPUT;
        DBMS_OUTPUT.PUT_LINE(v_output);
    END LOOP;
END;
```

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


PL/SQL procedure successfully completed.
```

```
DECLARE
  v_output VARCHAR2(1000);
BEGIN
  FOR rec IN (SELECT
            'CUSTOMER: ' || C.FIRST_NAME || ', ' || C.SURNAME || CHR(10) ||
            'DONATION PURCHASED: ' || DN.DONATION || CHR(10) ||
            'PRICE: ' || DN.PRICE || CHR(10) ||
            'RETURN REASON: ' || R.REASON || CHR(10) ||
            '-----------------------------------------' AS OUTPUT
          FROM RETURNS R
          JOIN CUSTOMER C ON R.CUSTOMER_ID = C.CUSTOMER_ID
          JOIN DONATION DN ON R.DONATION_ID = DN.DONATION_ID)
  LOOP
    v_output := rec.OUTPUT;
```

```
      DBMS_OUTPUT.PUT_LINE(v_output);
   END LOOP;
END;
```

Question 5

```oracle
SET SERVEROUTPUT ON;

DECLARE
    v_customer_name VARCHAR2(100);
    v_employee_name VARCHAR2(100);
    v_donation VARCHAR2(100);
    v_dispatch_date DATE;
    v_delivery_date DATE;
    v_days_to_delivery NUMBER;
BEGIN
    FOR rec IN (
        SELECT
            SUBSTR(C.FIRST_NAME, 1, 1) || '.' || C.SURNAME AS CUSTOMER,
            SUBSTR(E.FIRST_NAME, 1, 1) || '.' || E.SURNAME AS EMPLOYEE,
            DN.DONATION AS DONATION,
            TO_CHAR(D.DISPATCH_DATE, 'DD/MON/YY') AS DISPATCH_DATE,
            TO_CHAR(D.DELIVERY_DATE, 'DD/MON/YY') AS DELIVERY_DATE,
            D.DELIVERY_DATE - D.DISPATCH_DATE AS DAYS_TO_DELIVERY
        FROM
            CUSTOMER C
        JOIN
            INVOICE I ON C.CUSTOMER_ID = I.CUSTOMER_ID
        JOIN
            EMPLOYEE E ON I.EMPLOYEE_ID = E.EMPLOYEE_ID
        JOIN
            DONATION DN ON I.DONATION_ID = DN.DONATION_ID
        JOIN
            DELIVERY D ON I.DELIVERY_ID = D.DELIVERY_ID
        WHERE
            C.CUSTOMER_ID = 11011
    ) LOOP
        v_customer_name := rec.CUSTOMER;
        v_employee_name := rec.EMPLOYEE;
        v_donation := rec.DONATION;
        v_dispatch_date := rec.DISPATCH_DATE;
        v_delivery_date := rec.DELIVERY_DATE;
        v_days_to_delivery := rec.DAYS_TO_DELIVERY;

        DBMS_OUTPUT.PUT_LINE('CUSTOMER: ' || v_customer_name);
        DBMS_OUTPUT.PUT_LINE('EMPLOYEE: ' || v_employee_name);
        DBMS_OUTPUT.PUT_LINE('DONATION: ' || v_donation);
        DBMS_OUTPUT.PUT_LINE('DISPATCH DATE: ' || v_dispatch_date);
        DBMS_OUTPUT.PUT_LINE('DELIVERY DATE: ' || v_delivery_date);
        DBMS_OUTPUT.PUT_LINE('DAYS TO DELIVERY: ' || v_days_to_delivery);
        DBMS_OUTPUT.PUT_LINE('----------------------------------------');
    END LOOP;
END;
```

```
CUSTOMER: J.Smith
EMPLOYEE: K.Marks
DONATION: Lazyboy Sofa
DISPATCH DATE: 18/MAY/24
DELIVERY DATE: 19/MAY/24
DAYS TO DELIVERY: 1
-------------------------------------------
CUSTOMER: J.Smith
EMPLOYEE: A.Andrews
DONATION: KIC Fridge
DISPATCH DATE: 10/MAY/24
DELIVERY DATE: 15/MAY/24
DAYS TO DELIVERY: 5
-------------------------------------------


PL/SQL procedure successfully completed.
```

```
SET SERVEROUTPUT ON;

DECLARE
  v_customer_name VARCHAR2(100);
  v_employee_name VARCHAR2(100);
  v_donation VARCHAR2(100);
  v_dispatch_date DATE;
  v_delivery_date DATE;
  v_days_to_delivery NUMBER;
BEGIN
  FOR rec IN (
    SELECT
      SUBSTR(C.FIRST_NAME, 1, 1) || '.' || C.SURNAME AS CUSTOMER,
      SUBSTR(E.FIRST_NAME, 1, 1) || '.' || E.SURNAME AS EMPLOYEE,
      DN.DONATION AS DONATION,
      TO_CHAR(D.DISPATCH_DATE, 'DD/MON/YY') AS DISPATCH_DATE,
      TO_CHAR(D.DELIVERY_DATE, 'DD/MON/YY') AS DELIVERY_DATE,
      D.DELIVERY_DATE - D.DISPATCH_DATE AS DAYS_TO_DELIVERY
    FROM
      CUSTOMER C
    JOIN
      INVOICE I ON C.CUSTOMER_ID = I.CUSTOMER_ID
    JOIN
      EMPLOYEE E ON I.EMPLOYEE_ID = E.EMPLOYEE_ID
    JOIN
      DONATION DN ON I.DONATION_ID = DN.DONATION_ID
    JOIN
      DELIVERY D ON I.DELIVERY_ID = D.DELIVERY_ID
    WHERE
      C.CUSTOMER_ID = 11011
  ) LOOP
    v_customer_name := rec.CUSTOMER;
    v_employee_name := rec.EMPLOYEE;
```

```
    v_donation := rec.DONATION;
    v_dispatch_date := rec.DISPATCH_DATE;
    v_delivery_date := rec.DELIVERY_DATE;
    v_days_to_delivery := rec.DAYS_TO_DELIVERY;

    DBMS_OUTPUT.PUT_LINE('CUSTOMER: ' || v_customer_name);
    DBMS_OUTPUT.PUT_LINE('EMPLOYEE: ' || v_employee_name);
    DBMS_OUTPUT.PUT_LINE('DONATION: ' || v_donation);
    DBMS_OUTPUT.PUT_LINE('DISPATCH DATE: ' || v_dispatch_date);
    DBMS_OUTPUT.PUT_LINE('DELIVERY DATE: ' || v_delivery_date);
    DBMS_OUTPUT.PUT_LINE('DAYS TO DELIVERY: ' || v_days_to_delivery);
    DBMS_OUTPUT.PUT_LINE('-----------------------------------------');
  END LOOP;
END;
```

Question 6

```
DECLARE
    v_first_name VARCHAR2(50);
    v_surname VARCHAR2(50);
    v_total_amount NUMBER := 0;
    v_rating VARCHAR2(10);

BEGIN
    FOR rec IN (
        SELECT
            C.FIRST_NAME,
            C.SURNAME,
            SUM(TO_NUMBER(REGEXP_REPLACE(DN.PRICE, '[^0-9]', ''))) AS TOTAL_AMOUNT
        FROM
            CUSTOMER C
        JOIN
            INVOICE I ON C.CUSTOMER_ID = I.CUSTOMER_ID
        JOIN
            DONATION DN ON I.DONATION_ID = DN.DONATION_ID
        GROUP BY
            C.FIRST_NAME, C.SURNAME
    ) LOOP
```

```
        v_first_name := rec.FIRST_NAME;
        v_surname := rec.SURNAME;
        v_total_amount := rec.TOTAL_AMOUNT;

        IF v_total_amount >= 1500 THEN
            v_rating := '(***)';
        ELSE
            v_rating := '';
        END IF;

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

    END LOOP;
END;
```

```
FIRST NAME: Jack
SURNAME: Smith
AMOUNT: R 1798 (***)
-----------------------------------------
FIRST NAME: Pat
SURNAME: Hendricks
AMOUNT: R 1299
-----------------------------------------
FIRST NAME: Andre
SURNAME: Clark
AMOUNT: R 799
-----------------------------------------
FIRST NAME: Lucy
SURNAME: Williams
AMOUNT: R 1778 (***)
-----------------------------------------


PL/SQL procedure successfully completed.
```

```
DECLARE
  v_first_name VARCHAR2(50);
  v_surname VARCHAR2(50);
  v_total_amount NUMBER := 0;
  v_rating VARCHAR2(10);

BEGIN
  FOR rec IN (
    SELECT
      C.FIRST_NAME,
      C.SURNAME,
      SUM(TO_NUMBER(REGEXP_REPLACE(DN.PRICE, '[^0-9]', ''))) AS TOTAL_AMOUNT
    FROM
      CUSTOMER C
    JOIN
```

```
        INVOICE I ON C.CUSTOMER_ID = I.CUSTOMER_ID
      JOIN
        DONATION DN ON I.DONATION_ID = DN.DONATION_ID
      GROUP BY
        C.FIRST_NAME, C.SURNAME
    ) LOOP

      v_first_name := rec.FIRST_NAME;
      v_surname := rec.SURNAME;
      v_total_amount := rec.TOTAL_AMOUNT;

      IF v_total_amount >= 1500 THEN
        v_rating := '(***)';
      ELSE
        v_rating := '';
      END IF;

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

    END LOOP;
END;
```

Question 7

%TYPE attribute;

```
DECLARE
    -- vars to store the customer name using %TYPE
    v_customer_name CUSTOMER.FIRST_NAME%TYPE;
    v_customer_surname CUSTOMER.SURNAME%TYPE;
BEGIN
    -- Assign a value to vars
    SELECT FIRST_NAME, SURNAME
    INTO v_customer_name, v_customer_surname
    FROM CUSTOMER
    WHERE CUSTOMER_ID = 11011;

    -- Output customer name
    DBMS_OUTPUT.PUT_LINE('Customer Name: ' || v_customer_name || ' ' || v_customer_surname);
END;
```

```
Customer Name: Jack Smith


PL/SQL procedure successfully completed.
```

(*What does '%Type' mean in Oracle sql?*, no date)

```
DECLARE
  -- vars to store the customer name using %TYPE
```

```
  v_customer_name CUSTOMER.FIRST_NAME%TYPE;
  v_customer_surname CUSTOMER.SURNAME%TYPE;
BEGIN
  -- Assign a value to vars
  SELECT FIRST_NAME, SURNAME
  INTO v_customer_name, v_customer_surname
  FROM CUSTOMER
  WHERE CUSTOMER_ID = 11011;

  -- Output customer name
  DBMS_OUTPUT.PUT_LINE('Customer Name: ' || v_customer_name || ' ' ||
v_customer_surname);
END;
```

%ROWTYPE attribute

```
DECLARE
    -- var to store a row of data from the CUSTOMER table
    v_customer_record CUSTOMER%ROWTYPE;
BEGIN
    -- Select the row where CUSTOMER_ID is 11011
    SELECT *
    INTO v_customer_record
    FROM CUSTOMER
    WHERE CUSTOMER_ID = 11011;

    -- Output customer information
    DBMS_OUTPUT.PUT_LINE('Customer ID: ' || v_customer_record.CUSTOMER_ID);
    DBMS_OUTPUT.PUT_LINE('Customer Name: ' || v_customer_record.FIRST_NAME || ' ' || v_customer_record.SURNAME);
    DBMS_OUTPUT.PUT_LINE('Email: ' || v_customer_record.EMAIL);
END;
```

```
Customer ID: 11011
Customer Name: Jack Smith
Email: jsmith@isat.com


PL/SQL procedure successfully completed.
```

(*%ROWTYPE Attribute*, no date)

```
DECLARE
  -- var to store a row of data from the CUSTOMER table
  v_customer_record CUSTOMER%ROWTYPE;
BEGIN
  -- Select the row where CUSTOMER_ID is 11011
  SELECT *
  INTO v_customer_record
  FROM CUSTOMER
  WHERE CUSTOMER_ID = 11011;

  -- Output customer information
  DBMS_OUTPUT.PUT_LINE('Customer ID: ' || v_customer_record.CUSTOMER_ID);
  DBMS_OUTPUT.PUT_LINE('Customer Name: ' || v_customer_record.FIRST_NAME || ' ' ||
v_customer_record.SURNAME);
```

```
    DBMS_OUTPUT.PUT_LINE('Email: ' || v_customer_record.EMAIL);
END;
```

User defined exception

```
-- Define user-defined exception
CREATE OR REPLACE PROCEDURE CheckCustomerSpending(p_customer_id IN NUMBER) IS

    -- Define the exception
    InsufficientFundsException EXCEPTION;

    -- local variables
    v_total_amount NUMBER := 0;

BEGIN
    -- Get total amount spent by the customer
    SELECT SUM(TO_NUMBER(REGEXP_REPLACE(DN.PRICE, '[^0-9]', ''))) INTO v_total_amount
    FROM CUSTOMER C
    JOIN INVOICE I ON C.CUSTOMER_ID = I.CUSTOMER_ID
    JOIN DONATION DN ON I.DONATION_ID = DN.DONATION_ID
    WHERE C.CUSTOMER_ID = p_customer_id;

    -- Check if total amount < R 1,000
    IF v_total_amount < 1000 THEN
        -- Raise the user-defined exception
        RAISE InsufficientFundsException;
    END IF;
```

```
    -- Output total amount if no exception was raised
    DBMS_OUTPUT.PUT_LINE('Total amount spent by customer: R ' || v_total_amount);

EXCEPTION
    -- Exception handling
    WHEN InsufficientFundsException THEN
        DBMS_OUTPUT.PUT_LINE('Error: Total amount spent is less than R 1,000.');
END CheckCustomerSpending;
```

```
Procedure CHECKCUSTOMERSPENDING compiled
```

(*PL/SQL - Exceptions*, no date)

```
-- Define user-defined exception
CREATE OR REPLACE PROCEDURE CheckCustomerSpending(p_customer_id IN NUMBER) IS

    -- Define the exception
    InsufficientFundsException EXCEPTION;

    -- local variables
    v_total_amount NUMBER := 0;

BEGIN
    -- Get total amount spent by the customer
    SELECT SUM(TO_NUMBER(REGEXP_REPLACE(DN.PRICE, '[^0-9]', ''))) INTO v_total_amount
```

```
    FROM CUSTOMER C
    JOIN INVOICE I ON C.CUSTOMER_ID = I.CUSTOMER_ID
    JOIN DONATION DN ON I.DONATION_ID = DN.DONATION_ID
    WHERE C.CUSTOMER_ID = p_customer_id;

    -- Check if total amount < R 1,000
    IF v_total_amount < 1000 THEN
        -- Raise the user-defined exception
        RAISE InsufficientFundsException;
    END IF;

    -- Output total amount if no exception was raised
    DBMS_OUTPUT.PUT_LINE('Total amount spent by customer: R ' || v_total_amount);

EXCEPTION
    -- Exception handling
    WHEN InsufficientFundsException THEN
        DBMS_OUTPUT.PUT_LINE('Error: Total amount spent is less than R 1,000.');
END CheckCustomerSpending;
```

Question 8

Worksheet | Query Builder

```
SELECT
    C.FIRST_NAME,
    C.SURNAME,
    SUM(TO_NUMBER(REGEXP_REPLACE(DN.PRICE, '[^0-9]', ''))) AS AMOUNT,
    CASE
        WHEN SUM(TO_NUMBER(REGEXP_REPLACE(DN.PRICE, '[^0-9]', ''))) >= 1500 THEN '***'
        WHEN SUM(TO_NUMBER(REGEXP_REPLACE(DN.PRICE, '[^0-9]', ''))) BETWEEN 1000 AND 1400 THEN '**'
        ELSE '*'
    END AS CUSTOMER_RATING
FROM
    CUSTOMER C
JOIN
    INVOICE I ON C.CUSTOMER_ID = I.CUSTOMER_ID
JOIN
    DONATION DN ON I.DONATION_ID = DN.DONATION_ID
GROUP BY
    C.FIRST_NAME, C.SURNAME
ORDER BY
    C.FIRST_NAME;
```

FIRST_NAME	SURNAME	AMOUNT	CUS
Andre	Clark	799	*
Jack	Smith	1798	***
Lucy	Williams	1778	***
Pat	Hendricks	1299	**

```
CUSTOMER_RATING
----------------
*
***
***
**
```

```
SELECT
   C.FIRST_NAME,
   C.SURNAME,
   SUM(TO_NUMBER(REGEXP_REPLACE(DN.PRICE, '[^0-9]', ''))) AS AMOUNT,
   CASE
      WHEN SUM(TO_NUMBER(REGEXP_REPLACE(DN.PRICE, '[^0-9]', ''))) >= 1500 THEN '***'
      WHEN SUM(TO_NUMBER(REGEXP_REPLACE(DN.PRICE, '[^0-9]', ''))) BETWEEN 1000 AND
1400 THEN '**'
      ELSE '*'
   END AS CUSTOMER_RATING
FROM
   CUSTOMER C
JOIN
   INVOICE I ON C.CUSTOMER_ID = I.CUSTOMER_ID
JOIN
   DONATION DN ON I.DONATION_ID = DN.DONATION_ID
GROUP BY
   C.FIRST_NAME, C.SURNAME
ORDER BY
   C.FIRST_NAME;
```

Reference list

What does '%Type' mean in Oracle sql? (no date).

https://stackoverflow.com/questions/3790658/what-does-type-mean-in-oracle-sql.

%ROWTYPE Attribute (no date).

https://docs.oracle.com/cd/B12037_01/appdev.101/b10807/13_elems042.htm.

PL/SQL - Exceptions (no date). https://www.tutorialspoint.com/plsql/plsql_exceptions.htm.

-