7/10/2024

ADDB 7311

ASSIGNMENT 2

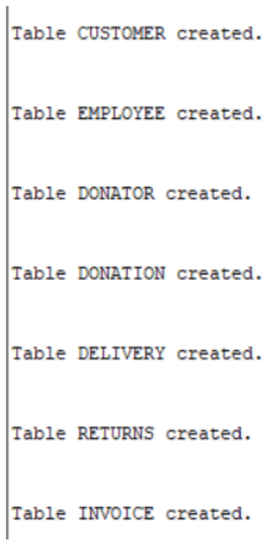
Sajana Bidesi
ST10249843

CONTENTS

QUESTION 1

CREATING THE TABLES

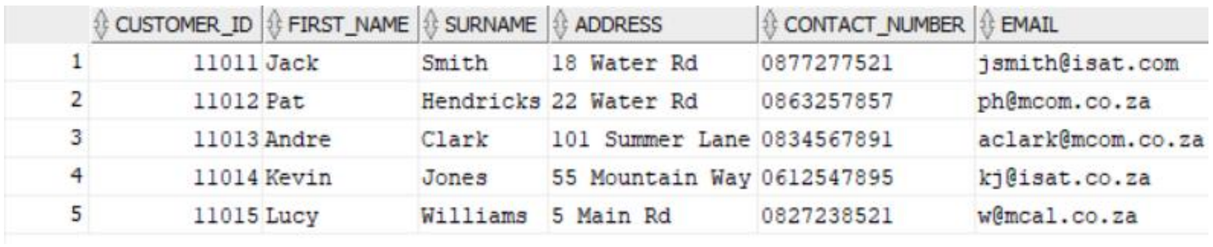
```
-- CREATE TABLE CUSTOMER

CREATE TABLE CUSTOMER

(

   CUSTOMER_ID NUMBER PRIMARY KEY NOT NULL,

   FIRST_NAME VARCHAR(50) NOT NULL,

   SURNAME VARCHAR(50) NOT NULL,

   ADDRESS VARCHAR(100) NOT NULL,

   CONTACT_NUMBER VARCHAR(50) NOT NULL,

   EMAIL VARCHAR(50) NOT NULL

);


-- CREATE TABLE EMPLOYEE

CREATE TABLE EMPLOYEE

(

   EMPLOYEE_ID VARCHAR(50) PRIMARY KEY NOT NULL,

   FISRT_NAME VARCHAR(50) NOT NULL,

   SURNAME VARCHAR(50) NOT NULL,

   CONTACT_NUMBER VARCHAR(50) NOT NULL,

   ADDRESS VARCHAR(100) NOT NULL,

   EMAIL VARCHAR(50) NOT NULL

);


-- CREATE TABLE DONATOR

CREATE TABLE DONATOR

(

   DONATOR_ID NUMBER PRIMARY KEY NOT NULL,

   FIRST_NAME VARCHAR(50) NOT NULL,

   SURNAME VARCHAR(50) NOT NULL,
    CONTACT_NUMBER VARCHAR(50) NOT NULL,
    EMAIL VARCHAR(50) NOT NULL
);

-- CREATE TABLE DONATION
CREATE TABLE DONATION
(
    DONATION_ID NUMBER PRIMARY KEY NOT NULL,
    DONATOR_ID NUMBER NOT NULL,
    FOREIGN KEY (DONATOR_ID) REFERENCES DONATOR(DONATOR_ID),
    DONATION VARCHAR(100) NOT NULL,
    PRICE NUMBER NOT NULL,
    DONATION_DATE DATE NOT NULL
);

-- CREATE TABLE DELIVERY
CREATE TABLE DELIVERY
(
    DELIVERY_ID NUMBER PRIMARY KEY NOT NULL,
    DELIVERY_NOTES VARCHAR(100) NOT NULL,
    DISPATCH_DATE DATE NOT NULL,
    DELIVERY_DATE DATE NOT NULL
);

-- CREATE TABLE RETURNS
CREATE TABLE RETURNS
(
    RETURN_ID NUMBER PRIMARY KEY NOT NULL,
    RETURN_DATE DATE NOT NULL,
    REASON VARCHAR(100) NOT NULL,
    CUSTOMER_ID NUMBER NOT NULL,
    FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER(CUSTOMER_ID),
    DONATION_ID NUMBER NOT NULL,
    FOREIGN KEY (DONATION_ID) REFERENCES DONATION(DONATION_ID),
    EMPLOYEE_ID VARCHAR(50) NOT NULL,
    FOREIGN KEY (EMPLOYEE_ID) REFERENCES EMPLOYEE(EMPLOYEE_ID)
);

-- CREATE TABLE INVOICE
CREATE TABLE INVOICE
(
    INVOICE_NUM NUMBER PRIMARY KEY NOT NULL,
    CUSTOMER_ID NUMBER NOT NULL,
    FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER(CUSTOMER_ID),
    INVOICE_DATE DATE NOT NULL,
    EMPLOYEE_ID VARCHAR(50) NOT NULL,
    FOREIGN KEY (EMPLOYEE_ID) REFERENCES EMPLOYEE(EMPLOYEE_ID),
    DONATION_ID NUMBER NOT NULL,
    FOREIGN KEY (DONATION_ID) REFERENCES DONATION(DONATION_ID),
    DELIVERY_ID NUMBER NOT NULL,
    FOREIGN KEY (DELIVERY_ID) REFERENCES DELIVERY(DELIVERY_ID)
);
```



```
Table CUSTOMER created.


Table EMPLOYEE created.


Table DONATOR created.


Table DONATION created.


Table DELIVERY created.


Table RETURNS created.


Table INVOICE created.
```

POPULATING THE TABLES

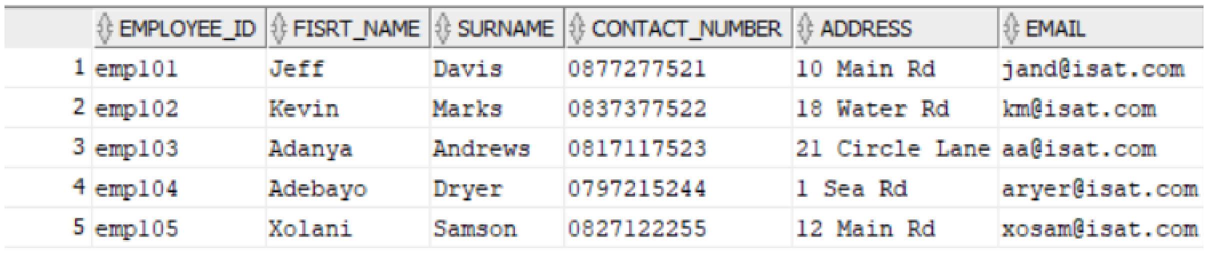
```
-- INSERT INTO CUSTOMER

INSERT ALL

   INTO CUSTOMER (CUSTOMER_ID, FIRST_NAME, SURNAME, ADDRESS, CONTACT_NUMBER, EMAIL) VALUES (11011, 'Jack', 'Smith', '18 Water Rd', '0877277521', 'jsmith@isat.com')

   INTO CUSTOMER (CUSTOMER_ID, FIRST_NAME, SURNAME, ADDRESS, CONTACT_NUMBER, EMAIL) VALUES (11012, 'Pat', 'Hendricks', '22 Water Rd', '0863257857', 'ph@mcom.co.za')

   INTO CUSTOMER (CUSTOMER_ID, FIRST_NAME, SURNAME, ADDRESS, CONTACT_NUMBER, EMAIL) VALUES (11013, 'Andre', 'Clark', '101 Summer Lane', '0834567891', 'aclark@mcom.co.za')

   INTO CUSTOMER (CUSTOMER_ID, FIRST_NAME, SURNAME, ADDRESS, CONTACT_NUMBER, EMAIL) VALUES (11014, 'Kevin', 'Jones', '55 Mountain Way', '0612547895', 'kj@isat.co.za')

   INTO CUSTOMER (CUSTOMER_ID, FIRST_NAME, SURNAME, ADDRESS, CONTACT_NUMBER, EMAIL) VALUES (11015, 'Lucy', 'Williams', '5 Main Rd', '0827238521', 'w@mcal.co.za')

SELECT * FROM dual;

SELECT * FROM CUSTOMER;
```



	CUSTOMER_ID	FIRST_NAME	SURNAME	ADDRESS	CONTACT_NUMBER	EMAIL
1	11011	Jack	Smith	18 Water Rd	0877277521	jsmith@isat.com
2	11012	Pat	Hendricks	22 Water Rd	0863257857	ph@mcom.co.za
3	11013	Andre	Clark	101 Summer Lane	0834567891	aclark@mcom.co.za
4	11014	Kevin	Jones	55 Mountain Way	0612547895	kj@isat.co.za
5	11015	Lucy	Williams	5 Main Rd	0827238521	w@mcal.co.za

```
-- INSERT INTO EMPLOYEE

INSERT ALL

   INTO EMPLOYEE (EMPLOYEE_ID, FISRT_NAME, SURNAME, CONTACT_NUMBER, ADDRESS, EMAIL) VALUES ('emp101', 'Jeff', 'Davis', '0877277521', '10 Main Rd', 'jand@isat.com')

   INTO EMPLOYEE (EMPLOYEE_ID, FISRT_NAME, SURNAME, CONTACT_NUMBER, ADDRESS, EMAIL) VALUES ('emp102', 'Kevin', 'Marks', '0837377522', '18 Water Rd', 'km@isat.com')

   INTO EMPLOYEE (EMPLOYEE_ID, FISRT_NAME, SURNAME, CONTACT_NUMBER, ADDRESS, EMAIL) VALUES ('emp103', 'Adanya', 'Andrews', '0817117523', '21 Circle Lane', 'aa@isat.com')

   INTO EMPLOYEE (EMPLOYEE_ID, FISRT_NAME, SURNAME, CONTACT_NUMBER, ADDRESS, EMAIL) VALUES ('emp104', 'Adebayo', 'Dryer', '0797215244', '1 Sea Rd', 'aryer@isat.com')

   INTO EMPLOYEE (EMPLOYEE_ID, FISRT_NAME, SURNAME, CONTACT_NUMBER, ADDRESS, EMAIL) VALUES ('emp105', 'Xolani', 'Samson', '0827122255', '12 Main Rd', 'xosam@isat.com')

SELECT * FROM dual;
```

SELECT * FROM EMPLOYEE;



	EMPLOYEE_ID	FISRT_NAME	SURNAME	CONTACT_NUMBER	ADDRESS	EMAIL
1	emp101	Jeff	Davis	0877277521	10 Main Rd	jand@isat.com
2	emp102	Kevin	Marks	0837377522	18 Water Rd	km@isat.com
3	emp103	Adanya	Andrews	0817117523	21 Circle Lane	aa@isat.com
4	emp104	Adebayo	Dryer	0797215244	1 Sea Rd	aryer@isat.com
5	emp105	Xolani	Samson	0827122255	12 Main Rd	xosam@isat.com

-- INSERT INTO DONATOR

INSERT ALL

INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, EMAIL) VALUES (20111, 'Jeff', 'Watson', '0827172250', 'jwatson@ymail.com')

INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, EMAIL) VALUES (20112, 'Stephen', 'Jones', '0837865670', 'joness@ymail.com')

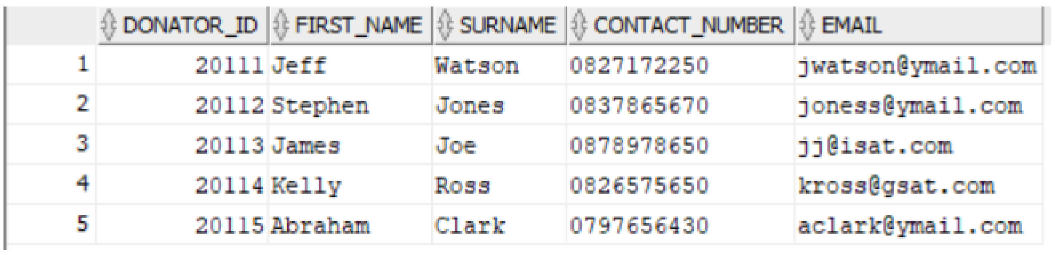
INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, EMAIL) VALUES (20113, 'James', 'Joe', '0878978650', 'jj@isat.com')

INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, EMAIL) VALUES (20114, 'Kelly', 'Ross', '0826575650', 'kross@gsat.com')

INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, EMAIL) VALUES (20115, 'Abraham', 'Clark', '0797656430', 'aclark@ymail.com')

SELECT * FROM dual;

SELECT * FROM DONATOR;



	DONATOR_ID	FIRST_NAME	SURNAME	CONTACT_NUMBER	EMAIL
1	20111	Jeff	Watson	0827172250	jwatson@ymail.com
2	20112	Stephen	Jones	0837865670	joness@ymail.com
3	20113	James	Joe	0878978650	jj@isat.com
4	20114	Kelly	Ross	0826575650	kross@gsat.com
5	20115	Abraham	Clark	0797656430	aclark@ymail.com

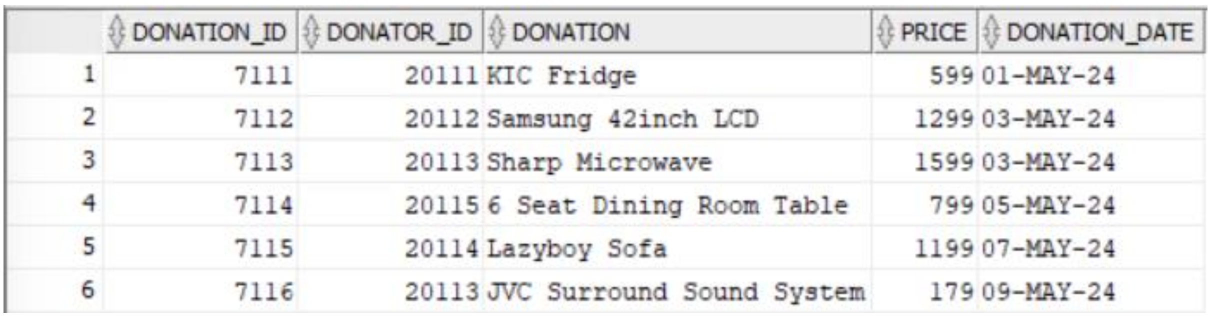
-- INSERT INTO DONATIONS

INSERT ALL

INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DONATION_DATE) VALUES (7111, 20111, 'KIC Fridge', 599, TO_DATE('01-May-2024', 'DD-Mon-YYYY'))

INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DONATION_DATE) VALUES (7112, 20112, 'Samsung 42inch LCD', 1299, TO_DATE('03-May-2024', 'DD-Mon-YYYY'))

```
   INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DONATION_DATE)
VALUES (7113, 20113, 'Sharp Microwave', 1599, TO_DATE('03-May-2024', 'DD-Mon-YYYY'))

   INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DONATION_DATE)
VALUES (7114, 20115, '6 Seat Dining Room Table', 799, TO_DATE('05-May-2024', 'DD-Mon-
YYYY'))

   INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DONATION_DATE)
VALUES (7115, 20114, 'Lazyboy Sofa', 1199, TO_DATE('07-May-2024', 'DD-Mon-YYYY'))

   INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DONATION_DATE)
VALUES (7116, 20113, 'JVC Surround Sound System', 179, TO_DATE('09-May-2024', 'DD-Mon-
YYYY'))

SELECT * FROM dual;

SELECT * FROM DONATION;
```



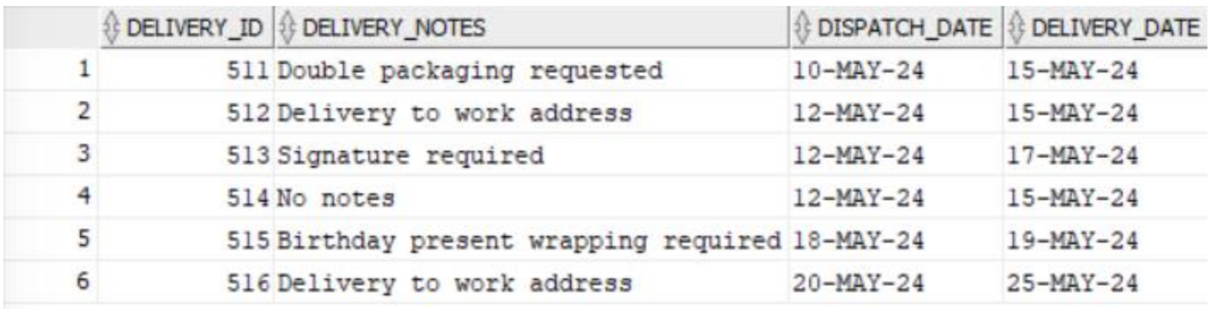
	DONATION_ID	DONATOR_ID	DONATION	PRICE	DONATION_DATE
1	7111	20111	KIC Fridge	599	01-MAY-24
2	7112	20112	Samsung 42inch LCD	1299	03-MAY-24
3	7113	20113	Sharp Microwave	1599	03-MAY-24
4	7114	20115	6 Seat Dining Room Table	799	05-MAY-24
5	7115	20114	Lazyboy Sofa	1199	07-MAY-24
6	7116	20113	JVC Surround Sound System	179	09-MAY-24

```
-- INSERT INTO DELIVERY

INSERT ALL

   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'))

   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'))

   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'))

   INTO DELIVERY (DELIVERY_ID, DELIVERY_NOTES, DISPATCH_DATE, DELIVERY_DATE) VALUES
(514, 'No notes', TO_DATE('12-May-2024', 'DD-Mon-YYYY'), TO_DATE('15-May-2024', 'DD-Mon-
YYYY'))

   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'))
```

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'))

SELECT * FROM dual;

SELECT * FROM DELIVERY;



	DELIVERY_ID	DELIVERY_NOTES	DISPATCH_DATE	DELIVERY_DATE
1	511	Double packaging requested	10-MAY-24	15-MAY-24
2	512	Delivery to work address	12-MAY-24	15-MAY-24
3	513	Signature required	12-MAY-24	17-MAY-24
4	514	No notes	12-MAY-24	15-MAY-24
5	515	Birthday present wrapping required	18-MAY-24	19-MAY-24
6	516	Delivery to work address	20-MAY-24	25-MAY-24

-- INSERT INTO RETURNS

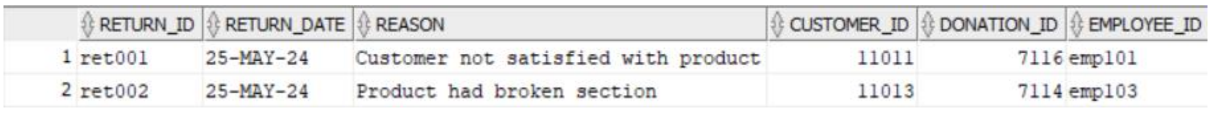
INSERT ALL

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')

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')

SELECT * FROM dual;

SELECT * FROM RETURNS;



	RETURN_ID	RETURN_DATE	REASON	CUSTOMER_ID	DONATION_ID	EMPLOYEE_ID
1	ret001	25-MAY-24	Customer not satisfied with product	11011	7116	emp101
2	ret002	25-MAY-24	Product had broken section	11013	7114	emp103

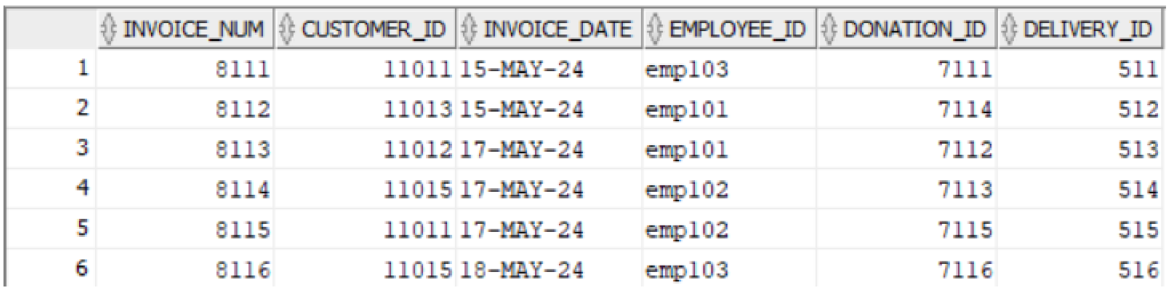
INSERT ALL

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)

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)

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)

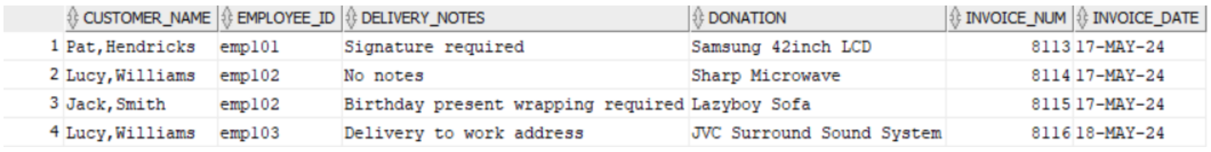
```
   INTO INVOICE (INVOICE_NUM, CUSTOMER_ID, INVOICE_DATE, EMPLOYEE_ID, DONATION_ID, DELIVERY_ID) VALUES (8114, 11015, TO_DATE('17-May-2024', 'DD-Mon-YYYY'), 'emp102', 7113, 514)

   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)

   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)

SELECT * FROM dual;

SELECT * FROM INVOICE;
```



	INVOICE_NUM	CUSTOMER_ID	INVOICE_DATE	EMPLOYEE_ID	DONATION_ID	DELIVERY_ID
1	8111	11011	15-MAY-24	emp103	7111	511
2	8112	11013	15-MAY-24	emp101	7114	512
3	8113	11012	17-MAY-24	emp101	7112	513
4	8114	11015	17-MAY-24	emp102	7113	514
5	8115	11011	17-MAY-24	emp102	7115	515
6	8116	11015	18-MAY-24	emp103	7116	516

QUESTION 2

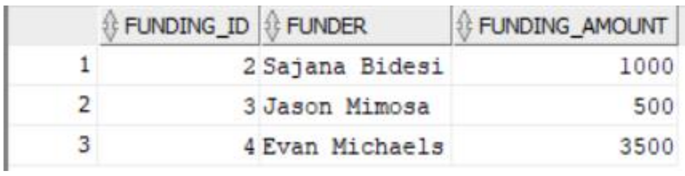
```
-- QUESTION 2
SELECT
CUST.FIRST_NAME || ',' || CUST.SURNAME AS CUSTOMER_NAME,
EMP.EMPLOYEE_ID,
DEL.DELIVERY_NOTES,
DON.DONATION,
INV.INVOICE_NUM,
INV.INVOICE_DATE
FROM
  CUSTOMER CUST
  JOIN INVOICE INV ON CUST.CUSTOMER_ID = INV.CUSTOMER_ID
  JOIN EMPLOYEE EMP ON INV.EMPLOYEE_ID = EMP.EMPLOYEE_ID
  JOIN DONATION DON ON INV.DONATION_ID = DON.DONATION_ID
  JOIN DELIVERY DEL ON INV.DELIVERY_ID = DEL.DELIVERY_ID
WHERE
  INV.INVOICE_DATE > TO_DATE('16-MAY-2024', 'DD-MON-YYYY')
ORDER BY
  DON.DONATION_DATE;
```



	CUSTOMER_NAME	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

```
-- QUESTION 3 (The IIE, 2024)

-- CREATE THE SEQUENCE TO INCREMENT THE VALUE FOR THE ID

CREATE SEQUENCE funding_seq

START WITH 1

INCREMENT BY 1

NOCACHE;


-- CREATE THE FUNDING TABLE WITH THE PROVIDED ATTRIBUTES

CREATE TABLE FUNDING (

  FUNDING_ID INT PRIMARY KEY,

  FUNDER VARCHAR(255) NOT NULL,

  FUNDING_AMOUNT NUMBER NOT NULL

);


-- INSERT A RECORT USING THE SEQUENCE

INSERT INTO Funding (FUNDING_ID, FUNDER, FUNDING_AMOUNT)

VALUES (funding_seq.NEXTVAL, 'Sajana Bidesi', 1000);

INSERT INTO Funding (FUNDING_ID, FUNDER, FUNDING_AMOUNT)

VALUES (funding_seq.NEXTVAL, 'Jason Mimosa', 500);

INSERT INTO Funding (FUNDING_ID, FUNDER, FUNDING_AMOUNT)

VALUES (funding_seq.NEXTVAL, 'Evan Michaels', 3500);


SELECT * FROM FUNDING;
```

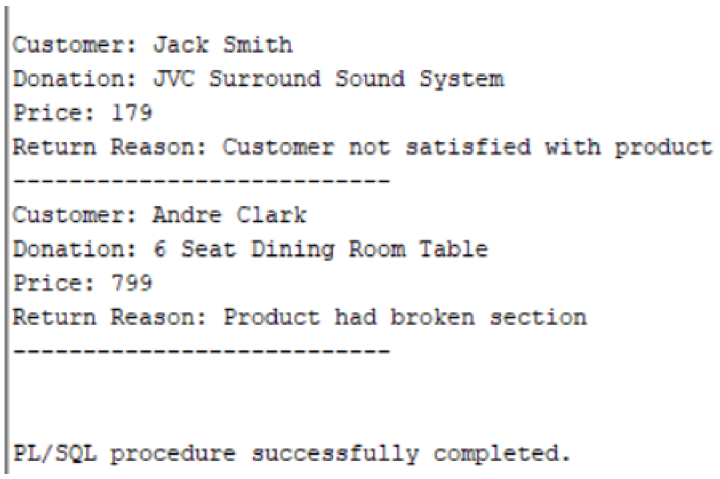


	FUNDING_ID	FUNDER	FUNDING_AMOUNT
1	2	Sajana Bidesi	1000
2	3	Jason Mimosa	500
3	4	Evan Michaels	3500

In Oracle databases, a sequence is a database object that generates a sequential series of unique numbers, which can be used primarily for creating primary keys or unique identifiers for rows in a table. When you create a sequence, you define its starting point, the increment value, and other optional parameters like caching and maximum values. Sequences are not tied to any specific table, allowing them to generate unique values across different tables. Each time a sequence is referenced (typically using `NEXTVAL`), it produces the next number in the sequence, ensuring that no two rows receive the same identifier, which helps maintain data integrity and uniqueness within the database. (Oracle, 2019; The IIE, 2024)

QUESTION 4

```
-- QUESTION 4
SET SERVEROUTPUT ON;

DECLARE
  CURSOR c_donations IS
    SELECT
      c.FIRST_NAME || ' ' || c.SURNAME AS Customer,
      d.DONATION,
      d.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;

BEGIN
  FOR rec IN c_donations LOOP
    DBMS_OUTPUT.PUT_LINE('Customer: ' || rec.Customer);
    DBMS_OUTPUT.PUT_LINE('Donation: ' || rec.DONATION);
    DBMS_OUTPUT.PUT_LINE('Price: ' || rec.PRICE);
    DBMS_OUTPUT.PUT_LINE('Return Reason: ' || rec."Return Reason");
    DBMS_OUTPUT.PUT_LINE('---------------------------');
  END LOOP;
END;
/
```

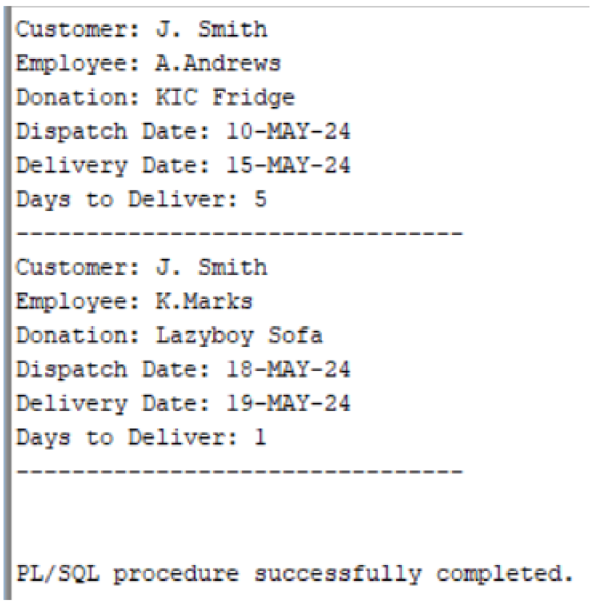


```
Customer: Jack Smith
Donation: JVC Surround Sound System
Price: 179
Return Reason: Customer not satisfied with product
---------------------------
Customer: Andre Clark
Donation: 6 Seat Dining Room Table
Price: 799
Return Reason: Product had broken section
---------------------------


PL/SQL procedure successfully completed.
```

QUESTION 5

```
-- QUESTION 5(The IIE, 2024)
SET SERVEROUTPUT ON;

BEGIN
  FOR rec IN (
    SELECT
      SUBSTR(c.FIRST_NAME, 1, 1) || '.' AS customer_initial,
      c.SURNAME AS customer_surname,
      SUBSTR(e.FISRT_NAME, 1, 1) || '.' || e.SURNAME AS employee_name,
      d.DONATION AS donation,
      del.DISPATCH_DATE AS dispatch_date,
      del.DELIVERY_DATE AS delivery_date,
      (del.DELIVERY_DATE - del.DISPATCH_DATE) AS days_to_deliver
    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 d ON i.DONATION_ID = d.DONATION_ID
    JOIN
      DELIVERY del ON i.DELIVERY_ID = del.DELIVERY_ID
    WHERE
      c.CUSTOMER_ID = 11011
  ) LOOP
    DBMS_OUTPUT.PUT_LINE('Customer: ' || rec.customer_initial || ' ' || rec.customer_surname);
    DBMS_OUTPUT.PUT_LINE('Employee: ' || rec.employee_name);
    DBMS_OUTPUT.PUT_LINE('Donation: ' || rec.donation);
    DBMS_OUTPUT.PUT_LINE('Dispatch Date: ' || rec.dispatch_date);
```

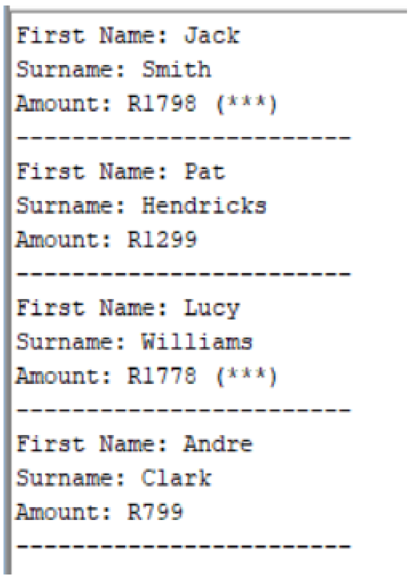
```
        DBMS_OUTPUT.PUT_LINE('Delivery Date: ' || rec.delivery_date);
        DBMS_OUTPUT.PUT_LINE('Days to Deliver: ' || rec.days_to_deliver);
        DBMS_OUTPUT.PUT_LINE('--------------------------------');
    END LOOP;
END;
/
```



```
Customer: J. Smith
Employee: A.Andrews
Donation: KIC Fridge
Dispatch Date: 10-MAY-24
Delivery Date: 15-MAY-24
Days to Deliver: 5
--------------------------------
Customer: J. Smith
Employee: K.Marks
Donation: Lazyboy Sofa
Dispatch Date: 18-MAY-24
Delivery Date: 19-MAY-24
Days to Deliver: 1
--------------------------------


PL/SQL procedure successfully completed.
```

QUESTION 6

```
-- QUESTION 6(The IIE, 2024)
SET SERVEROUTPUT ON;
BEGIN
 FOR rec IN (
  SELECT
   c.FIRST_NAME,
   c.SURNAME,
   SUM(d.PRICE) AS TotalSpent,
   CASE
    WHEN SUM(d.PRICE) >= 1500 THEN '(***)'
    ELSE ''
   END AS Rating
  FROM CUSTOMER c
  JOIN INVOICE i ON c.CUSTOMER_ID = i.CUSTOMER_ID
  JOIN DONATION d ON i.DONATION_ID = d.DONATION_ID
  WHERE c.FIRST_NAME IN ('Jack', 'Lucy', 'Pat', 'Andre')
  GROUP BY c.FIRST_NAME, c.SURNAME
 )
 LOOP
  DBMS_OUTPUT.PUT_LINE('First Name: ' || rec.FIRST_NAME);
  DBMS_OUTPUT.PUT_LINE('Surname: ' || rec.SURNAME);
  DBMS_OUTPUT.PUT_LINE('Amount: R' || rec.TotalSpent || ' ' || rec.Rating);
  DBMS_OUTPUT.PUT_LINE('------------------------');
 END LOOP;
END;
/
```



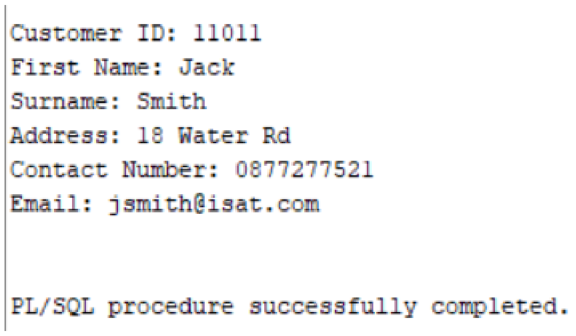
```
First Name: Jack
Surname: Smith
Amount: R1798 (***)
------------------------
First Name: Pat
Surname: Hendricks
Amount: R1299
------------------------
First Name: Lucy
Surname: Williams
Amount: R1778 (***)
------------------------
First Name: Andre
Surname: Clark
Amount: R799
------------------------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

QUESTION 7

QUESTION 7.1

```
-- QUESTION 7.1(Moore et al., 2019)
DECLARE
   -- DECLARE VARIABLES USING %TYPE ATTRIBUTE
   v_customer_id CUSTOMER.CUSTOMER_ID%TYPE;
   v_first_name CUSTOMER.FIRST_NAME%TYPE;
   v_surname CUSTOMER.SURNAME%TYPE;
   v_address CUSTOMER.ADDRESS%TYPE;
   v_contact_number CUSTOMER.CONTACT_NUMBER%TYPE;
   v_email CUSTOMER.EMAIL%TYPE;
BEGIN
   -- ASSIGN A SAMPLE CUSTOMER ID FOR FETCHING DETAILS
   v_customer_id := 11011;

   -- FETCH CUSTOMER DETAILS INTO VARIABLES
   SELECT FIRST_NAME, SURNAME, ADDRESS, CONTACT_NUMBER, EMAIL
   INTO v_first_name, v_surname, v_address, v_contact_number, v_email
   FROM CUSTOMER
   WHERE CUSTOMER_ID = v_customer_id;

   -- OUTPUT CUSTOMER DETAILS
   DBMS_OUTPUT.PUT_LINE('Customer ID: ' || v_customer_id);
   DBMS_OUTPUT.PUT_LINE('First Name: ' || v_first_name);
   DBMS_OUTPUT.PUT_LINE('Surname: ' || v_surname);
   DBMS_OUTPUT.PUT_LINE('Address: ' || v_address);
   DBMS_OUTPUT.PUT_LINE('Contact Number: ' || v_contact_number);
   DBMS_OUTPUT.PUT_LINE('Email: ' || v_email);
EXCEPTION
   WHEN NO_DATA_FOUND THEN
```

```
    DBMS_OUTPUT.PUT_LINE('No customer found with ID: ' || v_customer_id);
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

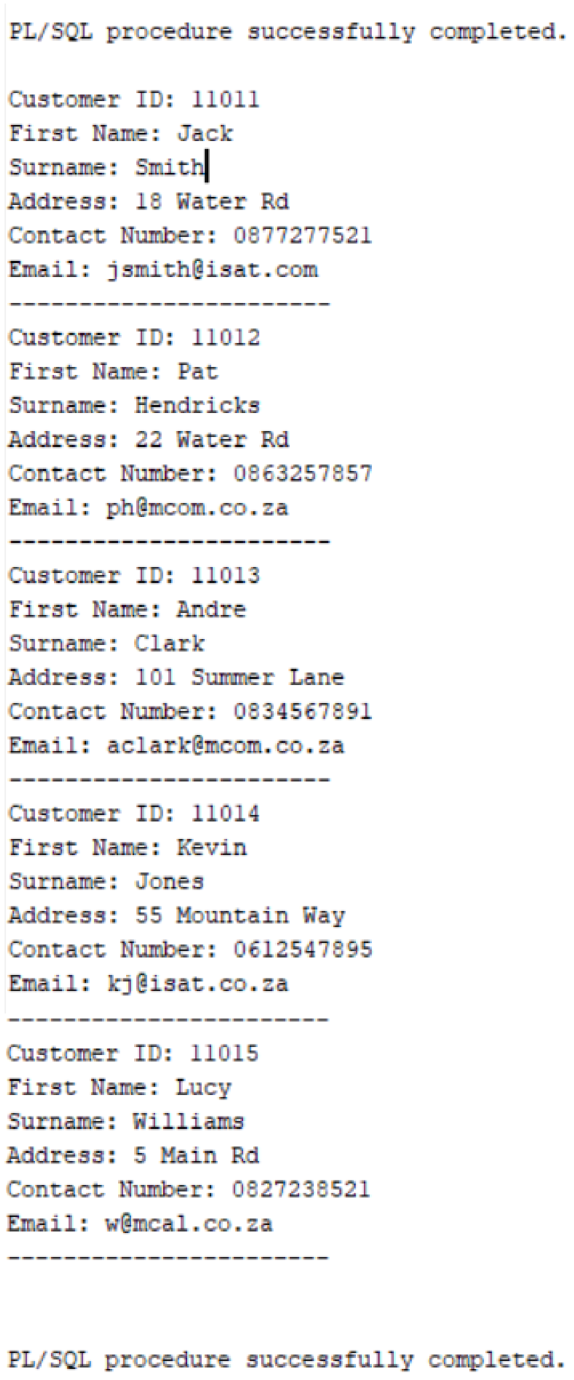


```
Customer ID: 11011
First Name: Jack
Surname: Smith
Address: 18 Water Rd
Contact Number: 0877277521
Email: jsmith@isat.com


PL/SQL procedure successfully completed.
```

QUESTION 7.2

```
-- QUESTION 7.2(L. Jayapalan et al., 2020)
DECLARE
  -- DECLARE A RECORD VARIABLE THAT CAN HOLD A ROW FROM THE CUSTOMER TABLE
  customer_record CUSTOMER%ROWTYPE;

  -- DECLARE A CURSOR TO SELECT ALL CUSTOMERS
  CURSOR customer_cursor IS
    SELECT * FROM CUSTOMER;
BEGIN
  OPEN customer_cursor;

  -- LOOP THROUGH EACH ROW RETURNED BY THE CURSOR
  LOOP
    FETCH customer_cursor INTO customer_record;

    -- EXIT THE LOOP WHEN NO MORE ROWS ARE FOUND
    EXIT WHEN customer_cursor%NOTFOUND;

    -- DISPLAY THE CUSTOMER DETAILS
    DBMS_OUTPUT.PUT_LINE('Customer ID: ' || customer_record.CUSTOMER_ID);
    DBMS_OUTPUT.PUT_LINE('First Name: ' || customer_record.FIRST_NAME);
    DBMS_OUTPUT.PUT_LINE('Surname: ' || customer_record.SURNAME);
    DBMS_OUTPUT.PUT_LINE('Address: ' || customer_record.ADDRESS);
    DBMS_OUTPUT.PUT_LINE('Contact Number: ' || customer_record.CONTACT_NUMBER);
    DBMS_OUTPUT.PUT_LINE('Email: ' || customer_record.EMAIL);
    DBMS_OUTPUT.PUT_LINE('-----------------------');
  END LOOP;

  CLOSE customer_cursor;
END;
```

/

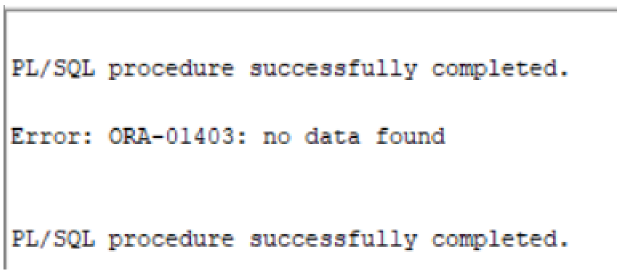


```
PL/SQL procedure successfully completed.

Customer ID: 11011
First Name: Jack
Surname: Smith
Address: 18 Water Rd
Contact Number: 0877277521
Email: jsmith@isat.com
-----------------------
Customer ID: 11012
First Name: Pat
Surname: Hendricks
Address: 22 Water Rd
Contact Number: 0863257857
Email: ph@mcom.co.za
-----------------------
Customer ID: 11013
First Name: Andre
Surname: Clark
Address: 101 Summer Lane
Contact Number: 0834567891
Email: aclark@mcom.co.za
-----------------------
Customer ID: 11014
First Name: Kevin
Surname: Jones
Address: 55 Mountain Way
Contact Number: 0612547895
Email: kj@isat.co.za
-----------------------
Customer ID: 11015
First Name: Lucy
Surname: Williams
Address: 5 Main Rd
Contact Number: 0827238521
Email: w@mcal.co.za
-----------------------


PL/SQL procedure successfully completed.
```

QUESTION 7.3

```
-- QUESTION 7.3(TutorialsPoint, n.d.)
SET SERVEROUTPUT ON;
-- PL/SQL block to define a custom exception
-- This block verifies if the donation price surpasses a certain limit
-- If it does, a custom exception is triggered
DECLARE
   -- Define an exception for exceeding the allowed donation price
   price_limit_exceeded EXCEPTION;

   -- Define a variable to store the donation price
   v_price NUMBER;

   -- Maximum permitted donation price
   max_price CONSTANT NUMBER := 2000; -- Example limit set to 1000

BEGIN
   -- Fetch the donation price from the DONATION table
   SELECT PRICE INTO v_price
   FROM DONATION
   WHERE DONATION_ID = 7117; -- Assuming donation ID 1 for illustration

   -- Verify if the price goes beyond the limit
   IF v_price > max_price THEN
      -- Trigger the custom exception if the price is above the limit
      RAISE price_limit_exceeded;
   END IF;

   -- If no exception is raised, insert data into the INVOICE table
   INSERT INTO INVOICE (INVOICE_NUM, CUSTOMER_ID, INVOICE_DATE, EMPLOYEE_ID,
DONATION_ID, DELIVERY_ID)
```

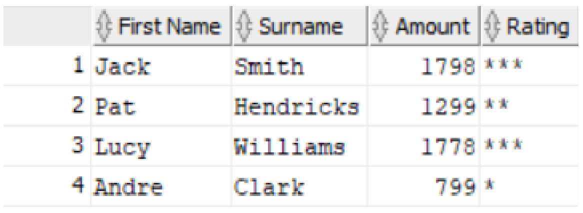
```
  VALUES (8117, 11011, SYSDATE, 'emp103', 7117, 517);


  DBMS_OUTPUT.PUT_LINE('Invoice successfully created.');


EXCEPTION
  -- Handle the custom exception if the donation price is too high
  WHEN price_limit_exceeded THEN
    DBMS_OUTPUT.PUT_LINE('Error: Donation price exceeds the allowed maximum of ' ||
max_price);


  -- Handle any other potential exceptions
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```



```
PL/SQL procedure successfully completed.

Error: ORA-01403: no data found


PL/SQL procedure successfully completed.
```

QUESTION 8

```
-- QUESTION 8

SELECT

  c.FIRST_NAME AS "First Name",

  c.SURNAME AS "Surname",

  SUM(d.PRICE) AS "Amount",

  CASE

    WHEN SUM(d.PRICE) >= 1500 THEN '***'

    WHEN SUM(d.PRICE) BETWEEN 1000 AND 1400 THEN '**'

    ELSE '*'

6

  END AS "Rating"

FROM CUSTOMER c

JOIN INVOICE i ON c.CUSTOMER_ID = i.CUSTOMER_ID

JOIN DONATION d ON i.DONATION_ID = d.DONATION_ID

WHERE c.FIRST_NAME IN ('Jack', 'Lucy', 'Pat', 'Andre')

GROUP BY c.FIRST_NAME, c.SURNAME;
```



	First Name	Surname	Amount	Rating
1	Jack	Smith	1798	***
2	Pat	Hendricks	1299	**
3	Lucy	Williams	1778	***
4	Andre	Clark	799	*

REFERENCES

L. Jayapalan, Belden, E., Huey, P., Moore, S. and Rich, K. (2020). *%ROWTYPE Attribute*. [online] Oracle Help Center. Available at: https://docs.oracle.com/en/database/oracle/oracle-database/21/lnpls/ROWTYPE-attribute.html.

Moore, S., L. Jayapalan, Huey, P., Belden, E., Rich, K. and Moore, V. (2019). *%TYPE Attribute*. [online] Oracle Help Center. Available at: https://docs.oracle.com/en/database/oracle/oracle-database/19/lnpls/TYPE-attribute.html.

Oracle (2019). *CREATE SEQUENCE*. [online] Oracle Help Center. Available at: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/CREATE-SEQUENCE.html.

The IIE (2024). *IIE Module Manual ADDB7311/ADDB6311 ADVANCED DATABASES MODULE MANUAL 2024 (First Edition: 2024)*.

TutorialsPoint (n.d.). *PL/SQL - Exceptions - Tutorialspoint*. [online] www.tutorialspoint.com. Available at: https://www.tutorialspoint.com/plsql/plsql_exceptions.htm.