



Varsity College - School of IT



# Advanced Databased (ADDB7311) Assignment 2

---

Bachelor of Computer Science in Application Development

Submitted by:  
**Cameron Chetty - ST10251759**

October 07, 2024

# QUESTION 1

## Schema for Creating Tables and Insert Scripts

-- Customer Table

```
CREATE TABLE CUSTOMER (
```

```
    CUSTOMER_ID NUMBER(5) PRIMARY KEY,  
    FIRST_NAME VARCHAR2(50),  
    SURNAME VARCHAR2(50),  
    ADDRESS VARCHAR2(100),  
    CONTACT_NUMBER VARCHAR2(10),  
    EMAIL VARCHAR2(100)
```

```
);
```

```
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', 'aclarke@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');
```

```

CREATE TABLE CUSTOMER (
    CUSTOMER_ID NUMBER(5) PRIMARY KEY,
    FIRST_NAME VARCHAR2(50),
    SURNAME VARCHAR2(50),
    ADDRESS VARCHAR2(100),
    CONTACT_NUMBER VARCHAR2(10),
    EMAIL VARCHAR2(100)
);

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', 'Iw@mcal.co.za');

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	Iw@mcal.co.za

-- Employee Table

CREATE TABLE EMPLOYEE (

EMPLOYEE\_ID VARCHAR2(10) PRIMARY KEY,

FIRST\_NAME VARCHAR2(50),

SURNAME VARCHAR2(50),

CONTACT\_NUMBER VARCHAR2(10),

ADDRESS VARCHAR2(100),

EMAIL VARCHAR2(100)

);

INSERT INTO EMPLOYEE (EMPLOYEE\_ID, FIRST\_NAME, SURNAME, CONTACT\_NUMBER, ADDRESS, EMAIL)

VALUES ('emp101', 'Jeff', 'Davis', '0877272510', '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', '12 main road', 'xosam@isat.com');
```

The screenshot shows the Oracle SQL Developer interface. The top menu bar includes 'File', 'Edit', 'Tools', 'Database', 'Session', 'Help', and 'Admin1'. Below the menu is a toolbar with icons for New, Open, Save, Print, Copy, Paste, Undo, Redo, and Font Size. The main window has tabs for 'Worksheet' and 'Query Builder', with 'Worksheet' selected. The code area contains the following SQL:

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

INSERT INTO EMPLOYEE (EMPLOYEE_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, ADDRESS, EMAIL)
VALUES ('emp101', 'Jeff', 'Davis', '0877272510', '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', '12 main road', 'xosam@isat.com');

SELECT * FROM EMPLOYEE;
```

The bottom part of the screenshot shows the 'Script Output' tab with three tabs: 'Script Output', 'Query Result', and 'Query Result 1'. The 'Query Result 1' tab displays the following table:

EMPLOYEE_ID	FIRST_NAME	SURNAME	CONTACT_NUMBER	ADDRESS	EMAIL
1 emp101	Jeff	Davis	0877272510	10 main road	jand@isat.com
2 emp102	Kevin	Marks	0837377522	18 water road	km@isat.com
3 emp103	Adanya	Andrews	0817117523	21 circle lane	aa@isat.com
4 emp104	Adebayo	Dryer	0797215244	1 sea road	aryer@isat.com
5 emp105	Xolani	Samson	0827122255	12 main road	xosam@isat.com

-- Donator Table

```
CREATE TABLE DONATOR (
    DONATOR_ID NUMBER(5) PRIMARY KEY,
    FIRST_NAME VARCHAR2(50),
    SURNAME VARCHAR2(50),
    CONTACT_NUMBER VARCHAR2(10),
    EMAIL VARCHAR2(100)
);
```

```
INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER,
EMAIL)
```

```
VALUES (20111, 'Jeff', 'Watson', '0827117250', 'jwatson@ymail.com');
```

```
INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER,
EMAIL)
```

```
VALUES (20112, 'Stephen', 'Jones', '0837866570', 'joness@ymail.com');
```

```
INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER,
EMAIL)
```

```
VALUES (20113, 'James', 'Joe', '0897878650', 'jj@isat.com');
```

```
INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER,
EMAIL)
```

```
VALUES (20114, 'Kelly', 'Ross', '0826575650', 'kross@gSAT.com');
```

```
INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER,
EMAIL)
```

```
VALUES (20115, 'Abraham', 'Clark', '0797656430', 'aclark@ymail.com');
```

-- Donator Table

```
CREATE TABLE DONATOR (
    DONATOR_ID NUMBER(5) PRIMARY KEY,
    FIRST_NAME VARCHAR2(50),
    SURNAME VARCHAR2(50),
    CONTACT_NUMBER VARCHAR2(10),
    EMAIL VARCHAR2(100)
);

INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, EMAIL)
VALUES (20111, 'Jeff', 'Watson', '0827117250', 'jwatson@ymail.com');

INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, EMAIL)
VALUES (20112, 'Stephen', 'Jones', '0837866570', 'joness@ymail.com');

INSERT INTO DONATOR (DONATOR_ID, FIRST_NAME, SURNAME, CONTACT_NUMBER, EMAIL)
VALUES (20113, 'James', 'Joe', '0897878650', 'jj@isat.com');

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

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

SELECT * FROM DONATOR;
```

Script Output X | Query Result 2 X

SQL | All Rows Fetched: 5 in 0.012 seconds

	DONATOR_ID	FIRST_NAME	SURNAME	CONTACT_NUMBER	EMAIL
1	20111	Jeff	Watson	0827117250	jwatson@ymail.com
2	20112	Stephen	Jones	0837866570	joness@ymail.com
3	20113	James	Joe	0897878650	jj@isat.com
4	20114	Kelly	Ross	0826575650	kross@gsat.com
5	20115	Abraham	Clark	0797656430	aclark@ymail.com

-- Donation Table

```
CREATE TABLE DONATION (
    DONATION_ID NUMBER(5) PRIMARY KEY,
    DONATOR_ID NUMBER(5),
    DONATION VARCHAR2(100),
    PRICE NUMBER(7, 2),
    DONATION_DATE DATE,
    FOREIGN KEY (DONATOR_ID) REFERENCES DONATOR(DONATOR_ID)
);
```

```
INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE,
DONATION_DATE)
```

```
VALUES (7111, 20111, 'KIC Fridge', 599, TO_DATE('1-May-2024', 'DD-Mon-YYYY'));
```

```
INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE,
DONATION_DATE)
```

```
VALUES (7112, 20112, 'Samsung 42inch LCD', 1299, TO_DATE('3-May-2024', 'DD-Mon-YYYY'));
```

```
INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE,
DONATION_DATE)
```

```
VALUES (7113, 20113, 'Sharp Microwave', 1599, TO_DATE('3-May-2024', 'DD-Mon-YYYY'));
```

```
INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE,
DONATION_DATE)
```

```
VALUES (7114, 20115, '6 Seat Dining Room Table', 799, TO_DATE('5-May-2024', 'DD-Mon-YYYY'));
```

```
INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE,
DONATION_DATE)
```

```
VALUES (7115, 20114, 'Lazyboy Sofa', 1199, TO_DATE('7-May-2024', 'DD-Mon-YYYY'));
```

```
INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE,
DONATION_DATE)
```

```
VALUES (7116, 20113, 'JVC Surround Sound System', 179, TO_DATE('9-May-2024', 'DD-Mon-YYYY'));
```

Admin1

Worksheet Query Builder

```

CREATE TABLE DONATION (
    DONATION_ID NUMBER(5) PRIMARY KEY,
    DONATOR_ID NUMBER(5),
    DONATION VARCHAR2(100),
    PRICE NUMBER(7, 2),
    DONATION_DATE DATE,
    FOREIGN KEY (DONATOR_ID) REFERENCES DONATOR(DONATOR_ID)
);

INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DONATION_DATE)
VALUES (7111, 20111, 'KIC Fridge', 599, TO_DATE('1-May-2024', 'DD-Mon-YYYY'));

INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DONATION_DATE)
VALUES (7112, 20112, 'Samsung 42inch LCD', 1299, TO_DATE('3-May-2024', 'DD-Mon-YYYY'));

INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DONATION_DATE)
VALUES (7113, 20113, 'Sharp Microwave', 1599, TO_DATE('3-May-2024', 'DD-Mon-YYYY'));

INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DONATION_DATE)
VALUES (7114, 20115, '6 Seat Dining Room Table', 799, TO_DATE('5-May-2024', 'DD-Mon-YYYY'));

INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DONATION_DATE)
VALUES (7115, 20114, 'Lazyboy Sofa', 1199, TO_DATE('7-May-2024', 'DD-Mon-YYYY'));

INSERT INTO DONATION (DONATION_ID, DONATOR_ID, DONATION, PRICE, DONATION_DATE)
VALUES (7116, 20113, 'JVC Surround Sound System', 179, TO_DATE('9-May-2024', 'DD-Mon-YYYY'));

```

Script Output | Query Result 2 | Query Result 3

All Rows Fetched: 6 in 0.012 seconds

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

-- Delivery Table

```

CREATE TABLE DELIVERY (
    DELIVERY_ID NUMBER(5) PRIMARY KEY,
    DELIVERY_NOTES VARCHAR2(255),
    DISPATCH_DATE DATE,
    DELIVERY_DATE DATE
);

```

```
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('15-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'));
```

```

CREATE TABLE DELIVERY (
    DELIVERY_ID NUMBER(5) PRIMARY KEY,
    DELIVERY_NOTES VARCHAR2(255),
    DISPATCH_DATE DATE,
    DELIVERY_DATE DATE
);

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('15-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'));

SELECT * FROM DELIVERY;

```

Script Output | All Rows Fetched: 6 in 0.011 seconds

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

-- Returns Table

```

CREATE TABLE RETURNS (
    RETURN_ID VARCHAR2(10) PRIMARY KEY,
    RETURN_DATE DATE,
    REASON VARCHAR2(255),
    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)
);

```

```

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', 11011,  
7114, 'emp103');
```

The screenshot shows the Oracle SQL Developer interface. In the top-left corner, there's a toolbar with various icons. Below it is a menu bar with 'Worksheet' and 'Query Builder' tabs. The main area contains the following SQL code:

```
CREATE TABLE RETURNS (  
    RETURN_ID VARCHAR2(10) PRIMARY KEY,  
    RETURN_DATE DATE,  
    REASON VARCHAR2(255),  
    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)  
);  
  
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', 11011, 7114, 'emp103');  
  
SELECT * FROM RETURNS;
```

Below the code editor, there's a tab bar with 'Script Output', 'Query Result 4', and 'Query Result 5'. The 'Query Result 5' tab is active, showing the results of the SELECT query:

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	11011	7114	emp103

-- Invoice Table

```
CREATE TABLE INVOICE (  
  
    INVOICE_NUM NUMBER(5) PRIMARY KEY,  
  
    CUSTOMER_ID NUMBER(5),  
  
    INVOICE_DATE DATE,  
  
    EMPLOYEE_ID VARCHAR2(10),  
  
    DONATION_ID NUMBER(5),  
  
    DELIVERY_ID NUMBER(5),  
  
    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)

);

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);

Admin1

Worksheet Query Builder

```

CREATE TABLE INVOICE (
    INVOICE_NUM NUMBER(5) PRIMARY KEY,
    CUSTOMER_ID NUMBER(5),
    INVOICE_DATE DATE,
    EMPLOYEE_ID VARCHAR2(10),
    DONATION_ID NUMBER(5),
    DELIVERY_ID NUMBER(5),
    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)
);

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);

```

Script Output x | Query Result 4 x | Query Result 5 x | Query Result 6 x

SQL | All Rows Fetched: 6 in 0.01 seconds

	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

### Code for SQL

SELECT

```
CUSTOMER.FIRST_NAME || ',' || CUSTOMER.SURNAME AS CUSTOMER,  
EMPLOYEE.EMPLOYEE_ID,  
DELIVERY.DELIVERY_NOTES,  
DONATION.DONATION,  
INVOICE.INVOICE_NUM,  
TO_CHAR(INVOICE.INVOICE_DATE, 'DD/MON/YY') AS INVOICE_DATE
```

FROM

```
INVOICE
```

```
JOIN CUSTOMER ON INVOICE.CUSTOMER_ID = CUSTOMER.CUSTOMER_ID  
JOIN EMPLOYEE ON INVOICE.EMPLOYEE_ID = EMPLOYEE.EMPLOYEE_ID  
JOIN DELIVERY ON INVOICE.DELIVERY_ID = DELIVERY.DELIVERY_ID  
JOIN DONATION ON INVOICE.DONATION_ID = DONATION.DONATION_ID
```

WHERE

```
INVOICE.INVOICE_DATE > TO_DATE('16-MAY-2024', 'DD-MON-YYYY')
```

ORDER BY

```
INVOICE.INVOICE_DATE;
```

### Screenshots of executing the Code

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab is active, displaying the SQL query. The 'Query Result' tab at the bottom shows the results of the executed query.

```
SELECT  
    CUSTOMER.FIRST_NAME || ',' || CUSTOMER.SURNAME AS CUSTOMER,  
    EMPLOYEE.EMPLOYEE_ID,  
    DELIVERY.DELIVERY_NOTES,  
    DONATION.DONATION,  
    INVOICE.INVOICE_NUM,  
    TO_CHAR(INVOICE.INVOICE_DATE, 'DD/MON/YY') AS INVOICE_DATE  
FROM  
    INVOICE  
JOIN CUSTOMER ON INVOICE.CUSTOMER_ID = CUSTOMER.CUSTOMER_ID  
JOIN EMPLOYEE ON INVOICE.EMPLOYEE_ID = EMPLOYEE.EMPLOYEE_ID  
JOIN DELIVERY ON INVOICE.DELIVERY_ID = DELIVERY.DELIVERY_ID  
JOIN DONATION ON INVOICE.DONATION_ID = DONATION.DONATION_ID  
WHERE  
    INVOICE.INVOICE_DATE > TO_DATE('16-MAY-2024', 'DD-MON-YYYY')  
ORDER BY  
    INVOICE.INVOICE_DATE;
```

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

## Code for SQL

-- Created the FUNDING table

-- This table has three columns: FUNDING\_ID, FUNDER, and FUNDING\_AMOUNT.

-- FUNDING\_ID is the primary key and will be auto-generated.

CREATE TABLE FUNDING (

    FUNDING\_ID NUMBER PRIMARY KEY, -- Unique identifier for each funding record

    FUNDER VARCHAR2(100), -- Name of the funder (up to 100 characters)

    FUNDING\_AMOUNT NUMBER(10, 2) -- Amount of funding with two decimal places

);

-- Created a sequence to automatically generate unique FUNDING\_ID values

-- The sequence will start at 1 and increment by 1 with each new record.

CREATE SEQUENCE FUNDING\_SEQ

START WITH 1 -- Start the sequence from 1

INCREMENT BY 1; -- Increase the sequence by 1 for each new record

-- Example insert statement

-- This statement adds a new record to the FUNDING table.

-- The FUNDING\_ID is automatically generated, so it's not included in the insert.

INSERT INTO FUNDING (FUNDING\_ID, FUNDER, FUNDING\_AMOUNT)

VALUES (FUNDING\_SEQ.NEXTVAL, 'ABC Charitable Trust', 5000.00);

INSERT INTO FUNDING (FUNDING\_ID, FUNDER, FUNDING\_AMOUNT)

VALUES (FUNDING\_SEQ.NEXTVAL, 'XYZ Foundation', 10000.00);

SELECT \* FROM FUNDING;

-- Explanation of the solution:

-- A sequence is a database object that automatically generates a unique numeric value, typically used to create unique identifiers for rows in a table.

-- I decided to use a sequence to ensure the automatic generation of unique IDs for the FUNDING\_ID column. This approach guarantees

-- that each record in the FUNDING table receives a unique identifier without manual intervention

-- , reducing the risk of duplicate entries and maintaining data integrity.

--The sequence is simple to implement and ensures that IDs are consistently incremented with each new insert operation

--(The IIE, 2024).

## Screenshots of executing the Code

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections pane shows an Oracle connection named 'Admin1'. The Reports pane is also visible. The central area has two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab contains the following SQL code:

```
-- Created the FUNDING table
-- This table has three columns: FUNDING_ID, FUNDER, and FUNDING_AMOUNT.
-- FUNDING_ID is the primary key and will be auto-generated.

CREATE TABLE FUNDING (
    FUNDING_ID NUMBER PRIMARY KEY, -- Unique identifier for each funding record
    FUNDER VARCHAR2(100), -- Name of the funder (up to 100 characters)
    FUNDING_AMOUNT NUMBER(10, 2) -- Amount of funding with two decimal places
);

-- Created a sequence to automatically generate unique FUNDING_ID values
-- The sequence will start at 1 and increment by 1 with each new record.
CREATE SEQUENCE FUNDING_SEQ
START WITH 1 -- Start the sequence from 1
INCREMENT BY 1; -- Increase the sequence by 1 for each new record

-- Example insert statement
-- This statement adds a new record to the FUNDING table.
-- The FUNDING_ID is automatically generated, so it's not included in the insert.

INSERT INTO FUNDING (FUNDING_ID, FUNDER, FUNDING_AMOUNT)
VALUES (FUNDING_SEQ.NEXTVAL, 'ABC Charitable Trust', 5000.00);

INSERT INTO FUNDING (FUNDING_ID, FUNDER, FUNDING_AMOUNT)
```

The 'Script Output' tab below shows the results of the execution:

```
Table FUNDING created.

Sequence FUNDING_SEQ created.

1 row inserted.

1 row inserted.
```

The screenshot shows the Oracle SQL Developer interface again. The 'Worksheet' tab contains the same SQL code as the previous screenshot, but the 'Query Result' tab is active, displaying the following data:

FUNDING_ID	FUNDER	FUNDING_AMOUNT
1	ABC Charitable Trust	5000
2	XYZ Foundation	10000

## QUESTION 4

### Code for SQL

```
SET SERVEROUTPUT ON;
BEGIN
    FOR rec IN (
        SELECT
            c.FIRST_NAME || ',' || c.SURNAME AS CUSTOMER,
            d.DONATION AS DONATION_PURCHASED,
            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
    ) LOOP
        -- Print the combined customer name, donation, price, and return reason
        DBMS_OUTPUT.PUT_LINE('CUSTOMER:      ' || rec.CUSTOMER);
        DBMS_OUTPUT.PUT_LINE('DONATION PURCHASED:  ' ||
            rec.DONATION_PURCHASED);
        DBMS_OUTPUT.PUT_LINE('PRICE:          ' || rec.PRICE);
        DBMS_OUTPUT.PUT_LINE('RETURN REASON:    ' || rec.RETURN_REASON);
        DBMS_OUTPUT.PUT_LINE('-----');
    END LOOP;
END;
/
```

## Screenshots of executing the Code

The screenshot shows the Oracle SQL Developer interface. The top window is titled "Worksheet" and contains a PL/SQL block. The bottom window is titled "Script Output" and displays the results of the executed code.

```
FOR rec IN (
    SELECT
        c.FIRST_NAME || ' ' || c.SURNAME AS CUSTOMER,
        d.DONATION AS DONATION_PURCHASED,
        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
) LOOP
    -- Print the combined customer name, donation, price, and return reason
    DBMS_OUTPUT.PUT_LINE('CUSTOMER: ' || rec.CUSTOMER);
    DBMS_OUTPUT.PUT_LINE('DONATION PURCHASED: ' || rec.DONATION_PURCHASED);
    DBMS_OUTPUT.PUT_LINE('PRICE: ' || rec.PRICE);
    DBMS_OUTPUT.PUT_LINE('RETURN REASON: ' || rec.RETURN_REASON);
    DBMS_OUTPUT.PUT_LINE('-----');
END LOOP;
END;
/
```

Script Output

Task completed in 0.108 seconds

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

PL/SQL procedure successfully completed.
```

# QUESTION 5

## Code for SQL

```
SET SERVEROUTPUT ON;
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,
    d.DONATION AS DONATION,
    TO_CHAR(del.DISPATCH_DATE, 'DD/MON/YY') AS DISPATCH_DATE,
    TO_CHAR(del.DELIVERY_DATE, 'DD/MON/YY') AS DELIVERY_DATE,
    (del.DELIVERY_DATE - del.DISPATCH_DATE) AS DAYS_TO_DELIVERY
  FROM
    INVOICE inv
  JOIN
    CUSTOMER c ON inv.CUSTOMER_ID = c.CUSTOMER_ID
  JOIN
    EMPLOYEE e ON inv.EMPLOYEE_ID = e.EMPLOYEE_ID
  JOIN
    DONATION d ON inv.DONATION_ID = d.DONATION_ID
  JOIN
    DELIVERY del ON inv.DELIVERY_ID = del.DELIVERY_ID
  WHERE
    c.CUSTOMER_ID = 11011
) LOOP
-- Print the formatted customer name, employee name, donation, and dates
DBMS_OUTPUT.PUT_LINE('CUSTOMER: ' || rec.CUSTOMER);
DBMS_OUTPUT.PUT_LINE('EMPLOYEE: ' || rec.EMPLOYEE);
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 DELIVERY: ' || rec.DAYS_TO_DELIVERY);
```

```

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

```

## Screenshots of executing the Code

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab contains the following PL/SQL code:

```

JOIN
    CUSTOMER c ON inv.CUSTOMER_ID = c.CUSTOMER_ID
JOIN
    EMPLOYEE e ON inv.EMPLOYEE_ID = e.EMPLOYEE_ID
JOIN
    DONATION d ON inv.DONATION_ID = d.DONATION_ID
JOIN
    DELIVERY del ON inv.DELIVERY_ID = del.DELIVERY_ID
WHERE
    c.CUSTOMER_ID = 11011
) LOOP
    -- Print the formatted customer name, employee name, donation, and dates
    DBMS_OUTPUT.PUT_LINE('CUSTOMER:      ' || rec.CUSTOMER);
    DBMS_OUTPUT.PUT_LINE('EMPLOYEE:     ' || rec.EMPLOYEE);
    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);

```

The 'Script Output' tab shows the results of the execution:

```

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

```

At the bottom of the 'Script Output' tab, it says "PL/SQL procedure successfully completed."

## QUESTION 6

### Code for SQL

```
SET SERVEROUTPUT ON;
BEGIN
    FOR rec IN (
        SELECT
            c.FIRST_NAME,
            c.SURNAME,
            SUM(d.PRICE) AS TOTAL_AMOUNT,
            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
        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.TOTAL_AMOUNT || rec.RATING);
        DBMS_OUTPUT.PUT_LINE('-----');
    END LOOP;
END;
/
```

## Screenshots of executing the Code

The screenshot shows the Oracle SQL Developer interface during the execution of a PL/SQL procedure. The 'Worksheet' tab displays the procedure code, and the 'Script Output' tab shows the results and completion message.

**Procedure Code (Worksheet):**

```
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
) LOOP
    DBMS_OUTPUT.PUT_LINE('FIRST NAME: ' || rec.FIRST_NAME);
    DBMS_OUTPUT.PUT_LINE('SURNAME: ' || rec.SURNAME);
    DBMS_OUTPUT.PUT_LINE('AMOUNT: R ' || rec.TOTAL_AMOUNT || rec.RATING);
    DBMS_OUTPUT.PUT_LINE('-----');
END LOOP;
END;
```

**Script Output (Output Tab):**

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

Task completed in 0.116 seconds

PL/SQL procedure successfully completed.

# QUESTION 7

Q.7.1)

## Code for SQL

```
SET SERVEROUTPUT ON;
```

```
-- Q.7.1 %TYPE Attribute
```

```
-- Example
```

```
DECLARE
```

```
    -- Declare a variable 'customer_contact' with the same data type as  
    CUSTOMER.CONTACT_NUMBER
```

```
    customer_contact CUSTOMER.CONTACT_NUMBER%TYPE;
```

```
BEGIN
```

```
    -- Retrieve the contact number of a specific customer into the 'customer_contact' variable
```

```
    SELECT CONTACT_NUMBER INTO customer_contact
```

```
    FROM CUSTOMER
```

```
    WHERE CUSTOMER_ID = 11011;
```

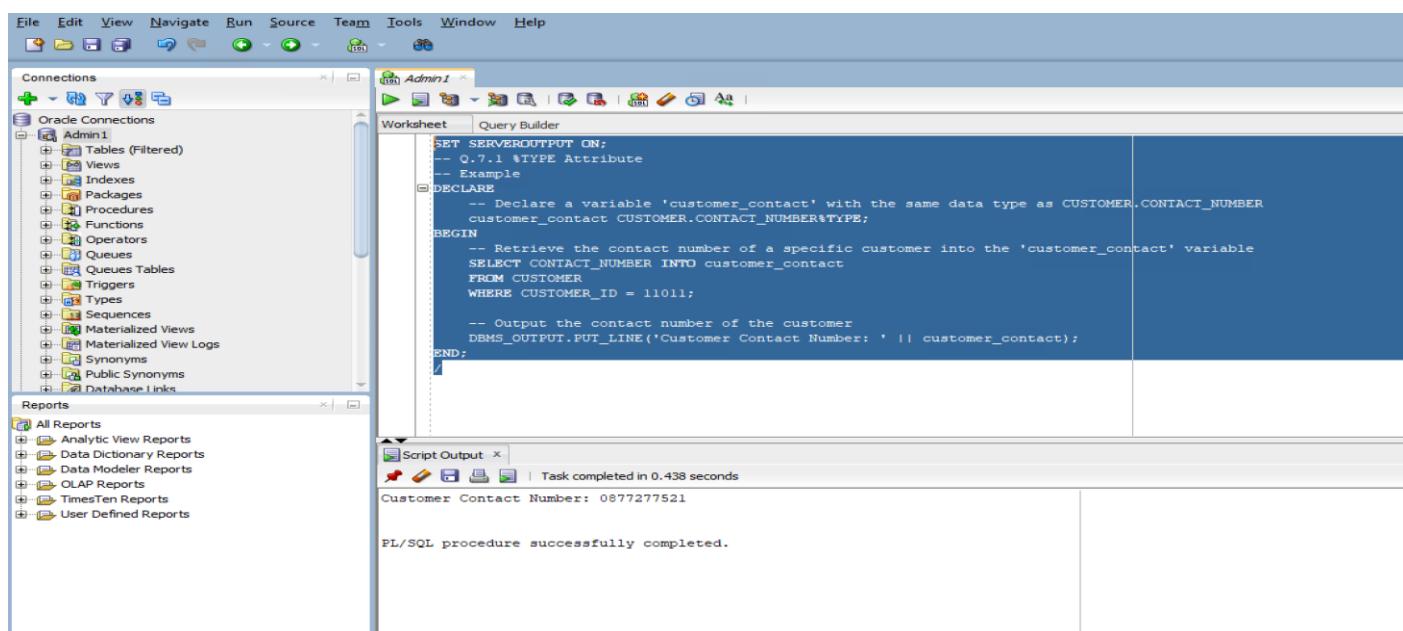
```
    -- Output the contact number of the customer
```

```
    DBMS_OUTPUT.PUT_LINE('Customer Contact Number: ' || customer_contact);
```

```
END;
```

```
/
```

## Screenshots of executing the Code



## Explanation

The %TYPE attribute in PL/SQL is used to declare a variable with the same data type as an existing database column or another variable. In this example, customer\_contact is declared using the %TYPE attribute, inheriting the data type from the CONTACT\_NUMBER column of the CUSTOMER table. This ensures that if the data type of CONTACT\_NUMBER changes in the future, the variable will automatically adapt without requiring changes in the PL/SQL code (The IIE, 2024).

Q.7.2)

## Code for SQL

```
SET SERVEROUTPUT ON;
-- Q.7.2 %ROWTYPE Attribute
-- Example
DECLARE
    -- Declare a variable 'donation_record' to hold an entire row from the DONATION table
    donation_record DONATION%ROWTYPE;
BEGIN
    -- Fetch the entire row for a specific donation ID
    SELECT * INTO donation_record
    FROM DONATION
    WHERE DONATION_ID = 7111;

    -- Output the details of the donation
    DBMS_OUTPUT.PUT_LINE('Donation ID: ' || donation_record.DONATION_ID);
    DBMS_OUTPUT.PUT_LINE('Donator ID: ' || donation_record.DONATOR_ID);
    DBMS_OUTPUT.PUT_LINE('Donation: ' || donation_record.DONATION);
    DBMS_OUTPUT.PUT_LINE('Price: ' || donation_record.PRICE);
    DBMS_OUTPUT.PUT_LINE('Donation Date: ' || donation_record.DONATION_DATE);
END;
/
```

## Screenshots of executing the Code

The screenshot shows the Oracle SQL Developer interface. On the left, there are two panes: 'Connections' (listing 'Admin1' connection) and 'Reports'. The main workspace contains a 'Worksheet' tab with a query builder interface. The code in the worksheet is:

```
SET SERVEROUTPUT ON;
-- Q.7.2 %ROWTYPE Attribute
-- Example
DECLARE
    -- Declare a variable 'donation_record' to hold an entire row from the DONATION table
    donation_record DONATION%ROWTYPE;
BEGIN
    -- Fetch the entire row for a specific donation ID
    SELECT * INTO donation_record
    FROM DONATION
    WHERE DONATION_ID = 7111;

    -- Output the details of the donation
    DBMS_OUTPUT.PUT_LINE('Donation ID: ' || donation_record.DONATION_ID);
    DBMS_OUTPUT.PUT_LINE('Donator ID: ' || donation_record.DONATOR_ID);
    DBMS_OUTPUT.PUT_LINE('Donation: ' || donation_record.DONATION);
    DBMS_OUTPUT.PUT_LINE('Price: ' || donation_record.PRICE);
    DBMS_OUTPUT.PUT_LINE('Donation Date: ' || donation_record.DONATION_DATE);
END;
```

Below the worksheet is a 'Script Output' pane showing the results of the execution:

```
Task completed in 0.228 seconds
Donation ID: 7111
Donator ID: 20111
Donation: KIC Fridge
Price: 599
Donation Date: 01-MAY-24

PL/SQL procedure successfully completed.
```

## Explanation

The %ROWTYPE attribute in PL/SQL is used to declare a record variable that matches the structure (i.e., the columns and their data types) of a table or a cursor. The variable `donation_record` is declared using %ROWTYPE, meaning it can hold all the column values from a single row in the `DONATION` table. This allows us to fetch and display all the relevant details of the donation in a single select statement (The IIE, 2024).

Q.7.3)

## Code for SQL

```
SET SERVEROUTPUT ON;
-- Set server output on to display output
SET SERVEROUTPUT ON;
DECLARE
    -- Declare variables
    v_customer_id NUMBER := 11011;
    v_donation_amount NUMBER := 1200; -- Example donation amount
```

```

v_max_limit NUMBER := 1000; -- Set a maximum allowed donation limit
-- Declare a user-defined exception for donation limit exceeded
ex_donation_limit EXCEPTION;

BEGIN
    -- Check if the donation amount exceeds the maximum limit
    IF v_donation_amount > v_max_limit THEN
        -- Raise the user-defined exception
        RAISE ex_donation_limit;
    ELSE
        -- If within limit, proceed with normal process
        DBMS_OUTPUT.PUT_LINE('Donation processed for customer ' || v_customer_id || ' with amount: ' || v_donation_amount);
    END IF;
EXCEPTION
    -- Handle the user-defined exception
    WHEN ex_donation_limit THEN
        DBMS_OUTPUT.PUT_LINE('Error: Donation amount of ' || v_donation_amount || ' exceeds the allowed limit of ' || v_max_limit);
END;
/

```

# Screenshots of executing the Code

If donation amount exceeds the limit:

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the 'Connections' tree, which includes 'Admin1' and various database objects like Tables, Views, Indexes, Procedures, Functions, Operators, Queues, Queues Tables, Triggers, Types, Sequences, Materialized Views, Materialized View Logs, Synonyms, Public Synonyms, and Database Links. Below this is the 'Reports' section with options like All Reports, Analytic View Reports, Data Dictionary Reports, Data Modeler Reports, OLAP Reports, TimesTen Reports, and User Defined Reports.

The main workspace is titled 'Worksheet' and contains the following PL/SQL code:

```
SET SERVEROUTPUT ON;
-- Set server output on to display output
SET SERVEROUTPUT ON;
DECLARE
    -- Declare variables
    v_customer_id NUMBER := 11011;
    v_donation_amount NUMBER := 1200; -- Example donation amount
    v_max_limit NUMBER := 1000; -- Set a maximum allowed donation limit
    -- Declare a user-defined exception for donation limit exceeded
    ex_donation_limit EXCEPTION;
BEGIN
    -- Check if the donation amount exceeds the maximum limit
    IF v_donation_amount > v_max_limit THEN
        -- Raise the user-defined exception
        RAISE ex_donation_limit;
    ELSE
        -- If within limit, proceed with normal process
        DBMS_OUTPUT.PUT_LINE('Donation processed for customer ' || v_customer_id || ' with amount: ' || v_donation_amount);
    END IF;
EXCEPTION
    -- Handle the user-defined exception
    WHEN ex_donation_limit THEN
        DBMS_OUTPUT.PUT_LINE('Error: Donation amount of ' || v_donation_amount || ' exceeds the allowed limit of ' || v_max_limit);
END;
/
```

The 'Script Output' window at the bottom shows the results of the execution:

```
Task completed in 0.097 seconds
Error: Donation amount of 1200 exceeds the allowed limit of 1000
PL/SQL procedure successfully completed.
```

If donation amount is within the limit (e.g., set v\_donation\_amount := 800)

The screenshot shows the Oracle SQL Developer interface, identical to the previous one but with a different parameter value. The 'Worksheet' pane contains the same PL/SQL code as before, but with v\_donation\_amount set to 800:

```
-- Set server output on to display output
SET SERVEROUTPUT ON;
DECLARE
    -- Declare variables
    v_customer_id NUMBER := 11011;
    v_donation_amount NUMBER := 800; -- Example donation amount
    v_max_limit NUMBER := 1000; -- Set a maximum allowed donation limit
    -- Declare a user-defined exception for donation limit exceeded
    ex_donation_limit EXCEPTION;
BEGIN
    -- Check if the donation amount exceeds the maximum limit
    IF v_donation_amount > v_max_limit THEN
        -- Raise the user-defined exception
        RAISE ex_donation_limit;
    ELSE
        -- If within limit, proceed with normal process
        DBMS_OUTPUT.PUT_LINE('Donation processed for customer ' || v_customer_id || ' with amount: ' || v_donation_amount);
    END IF;
EXCEPTION
    -- Handle the user-defined exception
    WHEN ex_donation_limit THEN
        DBMS_OUTPUT.PUT_LINE('Error: Donation amount of ' || v_donation_amount || ' exceeds the allowed limit of ' || v_max_limit);
END;
/
```

The 'Script Output' window shows the results:

```
Task completed in 0.095 seconds
Donation processed for customer 11011 with amount: 800
PL/SQL procedure successfully completed.
```

## Explanation

A user-defined exception in PL/SQL allows you to create and handle custom error conditions specific to your application's logic. A user-defined exception `ex_donation_limit` is declared. The code checks if the donation amount exceeds the defined limit. If it does, the exception is raised and handled with a custom error message. If the amount is within the limit, a success message is displayed (The IIE, 2024).

# QUESTION 8

## Code for SQL

SELECT

```
c.FIRST_NAME,  
c.SURNAME,  
SUM(d.PRICE) AS AMOUNT,  
CASE  
    WHEN SUM(d.PRICE) >= 1500 THEN '***' -- 3-star rating for amounts >= 1500  
    WHEN SUM(d.PRICE) BETWEEN 1000 AND 1400 THEN '**' -- 2-star rating for amounts  
        between 1000 and 1400  
    ELSE '*' -- 1-star rating for amounts < 1000  
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;
```

## Screenshots of executing the Code

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections pane shows an Oracle connection named 'Admin1' with various schema objects listed. The central area contains a 'Worksheet' tab with the SQL query. Below it, the 'Script Output' tab shows the executed query and its results. The results are displayed in a table:

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

## **REFERENCE LIST**

The IIE. 2024. Databases Advanced Databases Module Manual 2024, 1st ed. The IIE.