

ASSIGNMENT 2

Question 1

CREATE TABLE called supplier with the following fields .

supplier_id numeric(10) - > Primary key with constraint name

supplier_name varchar2(50) - >unique name

contact_name varchar2(50)

phone_no varchar2(10)- >unique name

city varchar2(10)

Region → should accept only ('N', 'NW', 'NE', 'S', 'SE', 'SW', 'W', 'E')

1. Insert 5 records
2. Display the details of the supplier who comes from Florida and their supplier id 500;
3. Add phone number in the supplier table using DDL command
4. Delete the unused column in the supplier table
5. Write a sql command to delete supplier table.
6. Create a view named supplier_contact . Include supplier_id,supplier_name,phone_no

```
/* QUESTION 01 */  
CREATE TABLE supplier  
(supplier_id NUMBER(10),  
supplier_name VARCHAR2(50) UNIQUE,  
contact_name VARCHAR2(50),  
phone_no VARCHAR2(10) UNIQUE,  
city VARCHAR2(10),  
region VARCHAR2(2),  
CONSTRAINT supplier_supplierid_pk PRIMARY KEY(supplier_id),  
CONSTRAINT supplier_region_ck  
CHECK (region IN ('N', 'NW', 'NE', 'S', 'SE', 'SW', 'W', 'E')));
```

ASSIGNMENT 2

```
/* QUESTION 01 */
CREATE TABLE supplier
(
  supplier_id NUMBER(10),
  supplier_name VARCHAR2(50) UNIQUE,
  contact_name VARCHAR2(50),
  phone_no VARCHAR2(10) UNIQUE,
  city VARCHAR2(10),
  region VARCHAR2(2),
  CONSTRAINT supplier_supplierid_pk PRIMARY KEY(supplier_id),
  CONSTRAINT supplier_region_ck
  CHECK (region IN ('N', 'NW', 'NE', 'S', 'SE', 'SW', 'W', 'E')));

--#1
INSERT INTO supplier VALUES (500, 'Kyle Lowry', 'Ayahna Cornish-Lowry', '123-1234', 'Florida', 'S');
INSERT INTO supplier VALUES (501, 'Kawhi Leonard', 'Kishele Shipley', '234-2345', 'Vaughan', 'SW');
INSERT INTO supplier VALUES (502, 'Danny Green', 'Fred VanVleet', '345-3456', 'Maple', 'SW');
INSERT INTO supplier VALUES (503, 'Serge Ibaka', 'Pascal Siakam', '456-4567', 'Aurora', 'NE');
INSERT INTO supplier VALUES (504, 'Jonas Valanciunas', 'Egle Valanciuniene', '567-5678', 'Newmarket', 'N');

SELECT * FROM supplier;
```

Script Output x

Task completed in 0.159 seconds

Table SUPPLIER created.

```
--#1
INSERT INTO supplier VALUES (500, 'Kyle Lowry', 'Ayahna Cornish-Lowry', '123-1234', 'Florida', 'S');
INSERT INTO supplier VALUES (501, 'Kawhi Leonard', 'Kishele Shipley', '234-2345', 'Vaughan', 'SW');
INSERT INTO supplier VALUES (502, 'Danny Green', 'Fred VanVleet', '345-3456', 'Maple', 'SW');
INSERT INTO supplier VALUES (503, 'Serge Ibaka', 'Pascal Siakam', '456-4567', 'Aurora', 'NE');
INSERT INTO supplier VALUES (504, 'Jonas Valanciunas', 'Egle Valanciuniene', '567-5678', 'Newmarket', 'N');

SELECT * FROM supplier;
```

Script Output x

Task completed in 0.527 seconds

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

SUPPLIER_ID	SUPPLIER_NAME	CONTACT_NAME	PHONE_NO	CITY	RE
500	Kyle Lowry	Ayahna Cornish-Lowry	123-1234	Florida	S
501	Kawhi Leonard	Kishele Shipley	234-2345	Vaughan	SW
502	Danny Green	Fred VanVleet	345-3456	Maple	SW
503	Serge Ibaka	Pascal Siakam	456-4567	Aurora	NE
504	Jonas Valanciunas	Egle Valanciuniene	567-5678	Newmarket	N

```
--#2
SELECT * FROM supplier WHERE supplier_id = 500 AND city = 'Florida';
```

ASSIGNMENT 2

```
--#2
SELECT * FROM supplier WHERE supplier_id = 500 AND city = 'Florida';
```

Script Output x

Task completed in 0.132 seconds

SUPPLIER_ID	SUPPLIER_NAME	CONTACT_NAME	PHONE_NO	CITY	RE
500	Kyle Lowry	Ayahna Cornish-Lowry	123-1234	Florida	S

```
--#3
```

```
ALTER TABLE supplier ADD(phone#_test VARCHAR2(10));
```

```
DESC supplier;
```

```
--#3
ALTER TABLE supplier ADD(phone#_test VARCHAR2(10));
DESC supplier;
```

Script Output x

Task completed in 0.354 seconds

Table SUPPLIER altered.

Name	Null?	Type
SUPPLIER_ID	NOT NULL	NUMBER(10)
SUPPLIER_NAME		VARCHAR2(50)
CONTACT_NAME		VARCHAR2(50)
PHONE_NO		VARCHAR2(10)
CITY		VARCHAR2(10)
REGION		VARCHAR2(2)
PHONE#_TEST		VARCHAR2(10)

```
--#4
```

```
ALTER TABLE supplier DROP COLUMN phone#_test;
```

```
DESC supplier;
```

```
--#4
ALTER TABLE supplier DROP COLUMN phone#_test;
DESC supplier;
```

Script Output x

Task completed in 0.407 seconds

Table SUPPLIER altered.

Name	Null?	Type
SUPPLIER_ID	NOT NULL	NUMBER(10)
SUPPLIER_NAME		VARCHAR2(50)
CONTACT_NAME		VARCHAR2(50)
PHONE_NO		VARCHAR2(10)
CITY		VARCHAR2(10)
REGION		VARCHAR2(2)

```
--#5
```

```
DROP TABLE SUPPLIER;
```

```
--#5
DROP TABLE SUPPLIER;
```

Script Output x

Task completed in 0.162 seconds

Table SUPPLIER dropped.

```
--#6
```

```
CREATE VIEW supplier_contact AS
```

ASSIGNMENT 2

```
SELECT supplier_id, supplier_name, phone_no FROM supplier;
```

```
SELECT * FROM
```

```
supplier_contact;
```

The screenshot shows a SQL IDE interface. At the top, a SQL script is displayed in a blue editor window:

```
--#6  
CREATE VIEW supplier_contact AS  
SELECT supplier_id, supplier_name, phone_no FROM supplier;  
  
SELECT * FROM  
supplier_contact;
```

Below the editor, a 'Script Output' window shows the execution results. It indicates that the task was completed in 0.21 seconds and that the view 'SUPPLIER_CONTACT' was created successfully. Below this message, a table displays the data from the view:

SUPPLIER_ID	SUPPLIER_NAME	PHONE_NO
500	Kyle Lowry	123-1234
501	Kawhi Leonard	234-2345
502	Danny Green	345-3456
503	Serge Ibaka	456-4567
504	Jonas Valanciunas	567-5678

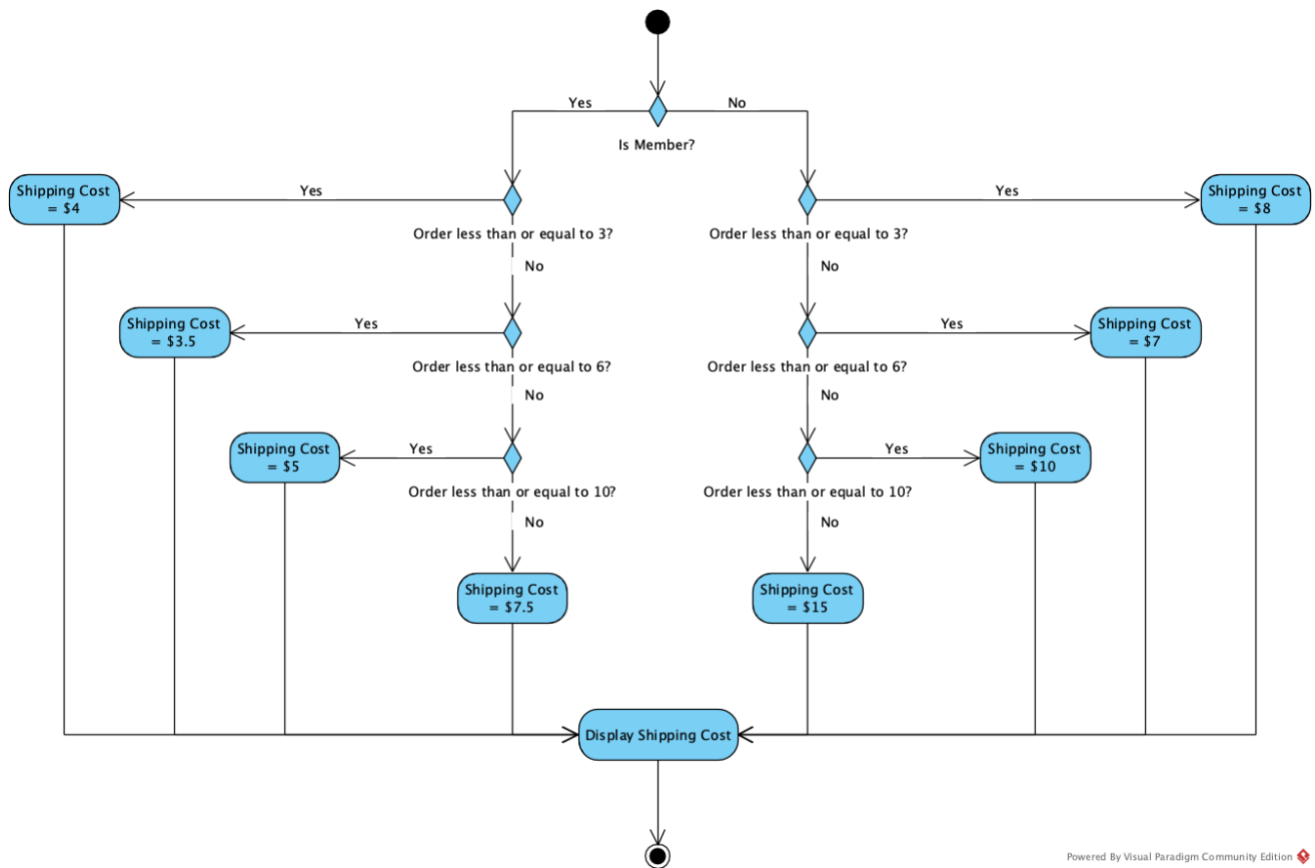
Question 2

Ecommerce site determines shipping cost based on the products ordered and membership. The valid rates are displayed in the following table:

QUANTITY	REGULAR SHIPPING COST	MEMBERS SHIPPING COST
Up to 3	\$ 8.00	\$ 4.00
4-6	\$7.00	\$ 3.50
7-10	\$10.00	\$5.00
>10	\$15.00	\$7.50

1. Create a **flowchart** to outline the processing steps in order to handle this calculation.

ASSIGNMENT 2



2. Create a pl/sql block to complete the above task. Include variable that holds Y OR N to include membership status and a variable to denote the number of items purchased. Verify with different values

```
3. DECLARE
4.     order_quantity NUMBER(3) := 7;
5.     is_member CHAR(1) := 'N';
6.
7.     quantity_up_to_three CONSTANT NUMBER(1) := 3;
8.     quantity_up_to_six CONSTANT NUMBER(1) := 6;
9.     quantity_up_to_ten CONSTANT NUMBER(2) := 10;
10.
11.     regular_ship_up_to_three CONSTANT NUMBER(1) := 8;
12.     regular_ship_up_to_six CONSTANT NUMBER(1) := 7;
13.     regular_ship_up_to_ten CONSTANT NUMBER(2) := 10;
14.     regular_ship_more_than_ten CONSTANT NUMBER(2) := 15;
15.
```

ASSIGNMENT 2

```
16.  member_ship_up_to_three CONSTANT NUMBER(1) := 4;
17.  member_ship_up_to_six  CONSTANT NUMBER(2,1) := 3.5;
18.  member_ship_up_to_ten  CONSTANT NUMBER(1) := 5;
19.  member_ship_more_than_ten CONSTANT NUMBER(2,1) := 7.5;
20.
21.  error_message VARCHAR2(100) := 'Membership status should be Y or N';
22. BEGIN
23.  IF is_member = 'Y' THEN
24.      IF order_quantity <= quantity_up_to_three THEN
25.          DBMS_OUTPUT.PUT_LINE(member_ship_up_to_three);
26.      ELSIF order_quantity <= quantity_up_to_six THEN
27.          DBMS_OUTPUT.PUT_LINE(member_ship_up_to_six);
28.      ELSIF order_quantity <= quantity_up_to_ten THEN
29.          DBMS_OUTPUT.PUT_LINE(member_ship_up_to_ten);
30.      ELSE
31.          DBMS_OUTPUT.PUT_LINE(member_ship_more_than_ten);
32.      END IF;
33.  ELSIF is_member = 'N' THEN
34.      IF order_quantity <= quantity_up_to_three THEN
35.          DBMS_OUTPUT.PUT_LINE(regular_ship_up_to_three);
36.      ELSIF order_quantity <= quantity_up_to_six THEN
37.          DBMS_OUTPUT.PUT_LINE(regular_ship_up_to_six);
38.      ELSIF order_quantity <= quantity_up_to_ten THEN
39.          DBMS_OUTPUT.PUT_LINE(regular_ship_up_to_ten);
40.      ELSE
41.          DBMS_OUTPUT.PUT_LINE(regular_ship_more_than_ten);
42.      END IF;
43.  ELSE
44.      DBMS_OUTPUT.PUT_LINE(error_message);
45.  END IF;
46. END;
47. /
```

ASSIGNMENT 2

```
DECLARE
    order_quantity NUMBER(3) := 7;
    is_member CHAR(1) := 'N';

    quantity_up_to_three CONSTANT NUMBER(1) := 3;
    quantity_up_to_six CONSTANT NUMBER(1) := 6;
    quantity_up_to_ten CONSTANT NUMBER(2) := 10;

    regular_ship_up_to_three CONSTANT NUMBER(1) := 8;
    regular_ship_up_to_six CONSTANT NUMBER(1) := 7;
    regular_ship_up_to_ten CONSTANT NUMBER(2) := 10;
    regular_ship_more_than_ten CONSTANT NUMBER(2) := 15;

    member_ship_up_to_three CONSTANT NUMBER(1) := 4;
    member_ship_up_to_six CONSTANT NUMBER(2,1) := 3.5;
    member_ship_up_to_ten CONSTANT NUMBER(1) := 5;
    member_ship_more_than_ten CONSTANT NUMBER(2,1) := 7.5;

    error_message VARCHAR2(100) := 'Membership status should be Y or N';
BEGIN
    IF is_member = 'Y' THEN
        IF order_quantity <= quantity_up_to_three THEN
            DBMS_OUTPUT.PUT_LINE(member_ship_up_to_three);
        ELSIF order_quantity <= quantity_up_to_six THEN
            DBMS_OUTPUT.PUT_LINE(member_ship_up_to_six);
        ELSIF order_quantity <= quantity_up_to_ten THEN
            DBMS_OUTPUT.PUT_LINE(member_ship_up_to_ten);
        ELSE
            DBMS_OUTPUT.PUT_LINE(member_ship_more_than_ten);
        END IF;
    ELSIF is_member = 'N' THEN
        IF order_quantity <= quantity_up_to_three THEN
            DBMS_OUTPUT.PUT_LINE(regular_ship_up_to_three);
        
```

Script Output x

Task completed in 0.135 seconds

PL/SQL procedure successfully completed.

Dbms Output x

Buffer Size: 20000

LoganKim x

10

Messages - Log x

Messages Statements Logging Page

Question 3

Run below script to create the table

DROP TABLE messages;

CREATE TABLE messages(results NUMBER(3));

a) Insert the numbers 1 through 10, excluding 6 and 8.

b) Commit before the end of the block.

c) Execute a SELECT statement to verify that your PL/SQL block worked.

```
DROP TABLE messages;
```

```
CREATE TABLE messages(results NUMBER(3));
```

```
BEGIN
```

```
FOR i in 1..10 LOOP
```

```
    IF i = 6 or i = 8 THEN
```

```
        null;
```

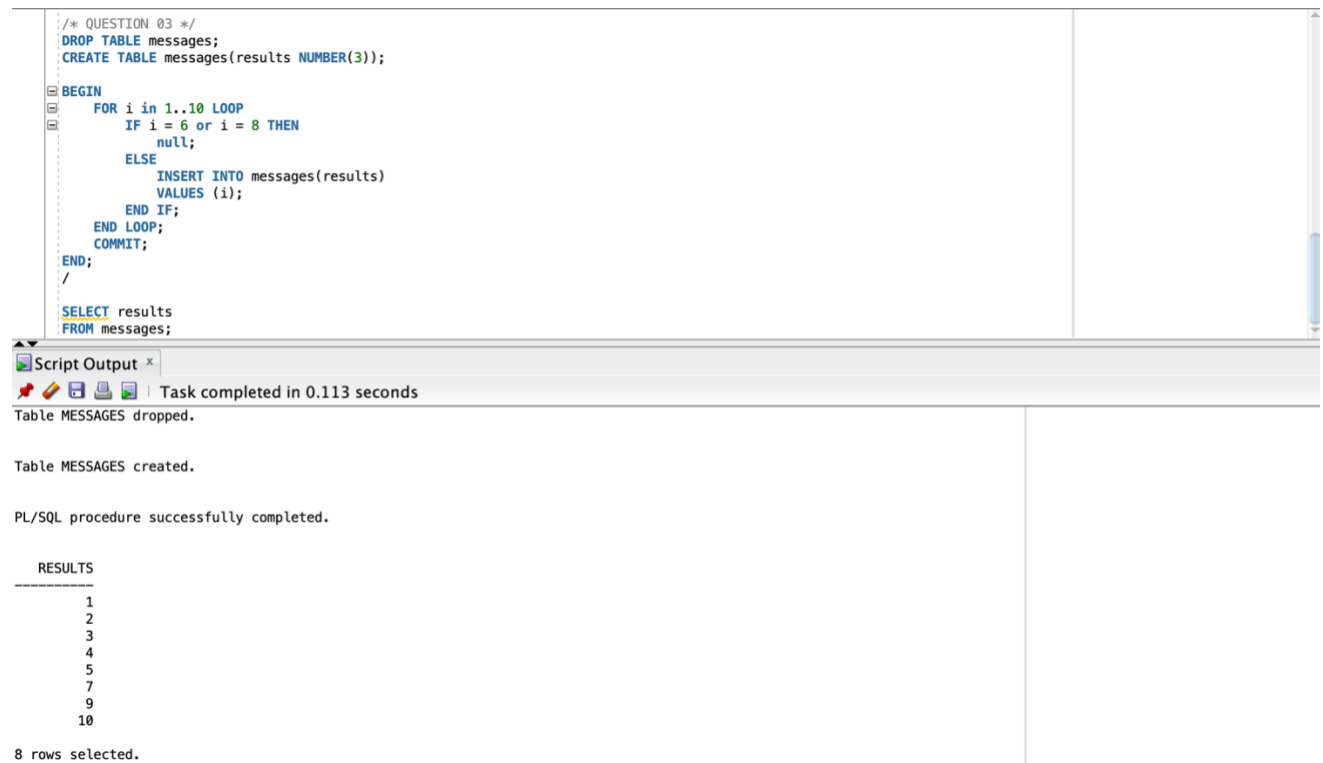
```
    ELSE
```

```
        INSERT INTO messages(results)
```

```
VALUES (i);
```

ASSIGNMENT 2

```
END IF;  
END LOOP;  
COMMIT;  
END;  
/  
  
SELECT results  
FROM messages;
```



The screenshot shows a SQL IDE with a script editor and a script output window. The script editor contains the following PL/SQL code:

```
/* QUESTION 03 */  
DROP TABLE messages;  
CREATE TABLE messages(results NUMBER(3));  
  
BEGIN  
  FOR i in 1..10 LOOP  
    IF i = 6 or i = 8 THEN  
      null;  
    ELSE  
      INSERT INTO messages(results)  
        VALUES (i);  
    END IF;  
  END LOOP;  
  COMMIT;  
END;  
/  
  
SELECT results  
FROM messages;
```

The script output window shows the following messages:

```
Script Output x  
Task completed in 0.113 seconds  
Table MESSAGES dropped.  
  
Table MESSAGES created.  
  
PL/SQL procedure successfully completed.  
  
RESULTS  
-----  
1  
2  
3  
4  
5  
7  
9  
10  
  
8 rows selected.
```

Submission:

- Copy your code to a MS-word file
- Include a screenshot of the output of each code segment.
- This assignment should be done individually
- Submit your work to e-centennial
- Email submission will be ignored