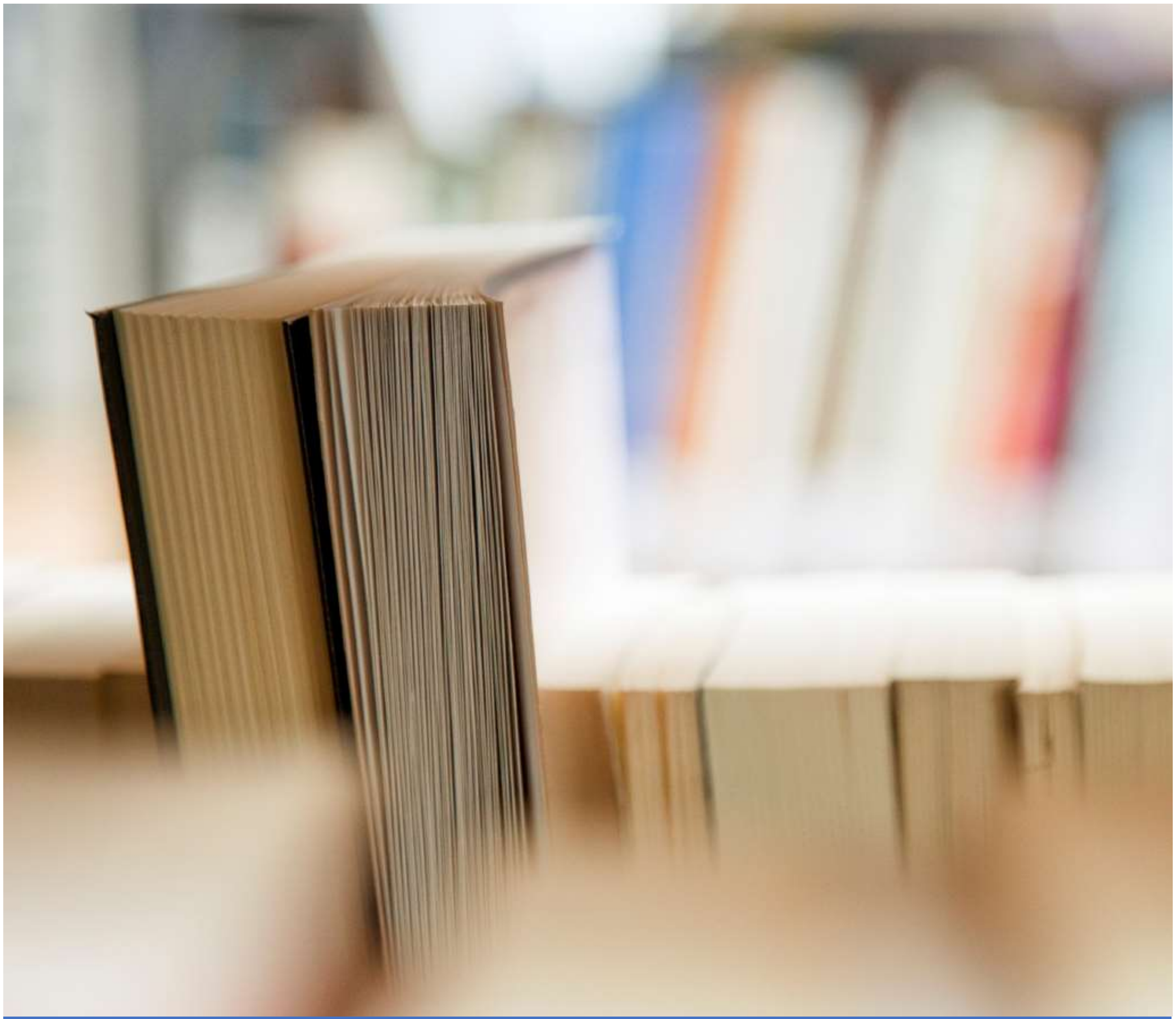

Assignment -1

Advance Database S1

Saifur Rahman 2018-1-60-048 CSE 464

1 (Basic Table Manipulation) Write a program that declares an integer variable called num, assigns a value to it, and computes and inserts into the tempp table the value of the variable itself, its square, and its cube. Consider the following DDL statements to create tempp table.
CREATE TABLE tempp (item number, square number, CUBE number);

Code:

```
CREATE TABLE tempp ( item number, square number, CUBE number );

DECLARE

    nm NUMBER := 3;

    item_a NUMBER;

    item_b NUMBER;

    item_c NUMBER;

BEGIN

    INSERT INTO tempp values (nm, nm*nm, nm*nm*nm);


    SELECT item, square, cube

    INTO item_a, item_b, item_c

    FROM tempp;

    dbms_output.put_line(item_a || ' ' ||item_b || ' ' || item_c);

END;
```

Output:

```
Statement processed.
3 9 27
```

2 (Conditional Statements) Input three positive integers representing the sides of a triangle, and determine whether they form a valid triangle. Hint: In a triangle, the sum of any two sides must always be greater than the third side. Display the output on the screen using dbms_output.put_line.

Code:

```
DECLARE

    side1 NUMBER := 5;

    side2 NUMBER := 7;

    side3 NUMBER := 3;

BEGIN

    dbms_output.put_line('STUDENT ID: 2018-1-60-048');

    IF side1 + side2 > side3 AND

        side1 + side3 > side2 AND

        side3 + side2 > side1 THEN

        dbms_output.put_line(side1||', '||side2||' and '||side3||' form a
valid triangle.');


    ELSE

         dbms_output.put_line(side1||', '||side2||' and '||side3||'  do not
form a valid triangle.');


    END IF;


END;
```

Output:

```
STUDENT ID: 2018-1-60-048
5, 7 and 3 form a valid triangle.
```

3 (Conditional Statements) Program should accept the age of the user. Depending upon the following conditions it should output:-
IF age <18 years, “child”
IF age >= 18 years and <21 years, “major”
IF age>= 21years, “adult”
Display the output on the screen using dbms_output.put_line.

Code:

```
DECLARE

    side1 NUMBER;

BEGIN

    side1 := 20;

    dbms_output.put_line('STUDENT ID: 2018-1-60-048');

    IF side1 < 18 THEN

        dbms_output.put_line('child');

    ELSIF side1 >=18 AND 21> side1 THEN

         dbms_output.put_line('major');

    ELSE

        dbms_output.put_line('adult');


    END IF;


END;
```

Output:

```
STUDENT ID: 2018-1-60-048
major
```

4 (Conditional Statements) Suppose the grade obtained by a student depends upon his scores and the grading rule is as follows. :-

Scores	Grades
95-100	A
70-84	B
70-84	C
60-69	D
0-59	F

Write a PL/SQL block to accept a student’s marks and accordingly output his grade. Display the output on the screen using dbms_output.put_line.

Code:

```
DECLARE
   mark number:=92;
    grade varchar2(1);
BEGIN
    dbms_output.put_line('STUDENT ID: 2018-1-60-048');

    IF mark <=59 THEN
        grade := 'F';
    ELSIF mark <=69 THEN
        grade := 'D';
    ELSIF mark <=84 THEN
        grade := 'C';
    ELSIF mark <=94 THEN
        grade := 'B';
    ELSIF mark <=100 THEN
        grade := 'A';
    END IF;
    dbms_output.put_line(grade);
END;
```

Output:

```
STUDENT ID: 2018-1-60-048
B
```

5 (Conditional Statements) A company manufactures three products:- computer stationery, fixed disks and computers. The following codes are used to indicate them:-

```
Product Code
Computer Stationery 1
Fixed Disks 2
Computers 3
The company has a discount policy as follows:
Product Order amount Discount
```

```
rate
Computer stationery 5000 and above 12%
Computer stationery 3000 and above 8%
Computer stationery Below 3000 2%
Fixed disks 20000 and above 10%
Fixed disks 15000 and above 5%
Computers 50000 and above 10%
Computers 25000 and above 5%
```

Write a program to accept the order details i.e. product code and order amounts for the products, calculate the discount amounts as per this policy and output the net order amount. Display the output on the screen using dbms_output.put_line.

```
Code:
CREATE OR REPLACE PROCEDURE PROC_4_2018_1_60_048(code in number, amount in
number, net out number)
IS
  discount number := 0;
BEGIN
    dbms_output.put_line('STUDENT ID: 2018-1-60-048');
    IF code = 1 THEN
        IF amount < 3000 THEN
            discount := amount * 0.02;
        ELSIF amount < 5000 THEN
            discount := amount * 0.08;
        ELSE
            discount := amount * 0.12;
        END IF;
    ELSIF code = 2 THEN
        IF amount >= 20000 THEN
            discount := amount * 0.10;
        ELSIF amount >=15000 THEN
            discount := amount * 0.05;
        END IF;
    ELSIF code = 3 THEN
        IF amount >= 50000 THEN
            discount := amount * 0.10;
        ELSIF amount >=25000 THEN
            discount := amount * 0.05;
        END IF;
    END IF;
    net := amount-discount;
END;
/
DECLARE
   a number;
BEGIN
   PROC_4_2018_1_60_048(2,15000,a);
   dbms_output.put_line('Net order amount:'||a);
END;
/
```

<table>
<tr><td></td><td>Output:
STUDENT ID: 2018-1-60-048
Net order amount:14250</td></tr>
<tr><td>6</td><td>(Iteration) Write a program that examines all the numbers from 1 to 999, displaying all those for which the sum of the cubes of the digits equal the number itself. Display the output on the screen using dbms_output.put_line.
Code:
<pre>
CREATE OR REPLACE PROCEDURE PROC_5_2018_1_60_048
IS
digit number := 0;
reminder number := 0;
res number := 0;
BEGIN
    dbms_output.put_line('STUDENT ID: 2018-1-60-048');
    FOR i IN 1..999
    LOOP
        digit := i;
        reminder := 0;
        res := 0;
        WHILE digit > 0
        LOOP
            reminder := MOD(digit,10); ---1%10 = 1
            res := res + (reminder*reminder*reminder); ---0 + 1 = 1
            digit := FLOOR(digit/10);
        END LOOP;
        IF res = i THEN
            dbms_output.put_line('Number: '||i);
        END IF;
    END LOOP;
END;
/

EXECUTE PROC_5_2018_1_60_048;
</pre>Output
<pre>
Statement processed.
STUDENT ID: 2018-1-60-048
Number: 1
Number: 153
Number: 370
Number: 371
Number: 407
</pre></td></tr>
<tr><td>7</td><td>(Procedure) The CUSTOMER table of a state electricity board consists of the following fields:</td></tr>
</table>

Meter_Number varchar2(4)
Meter_Type char(1)
Previous_Reading Number
Current_Reading Number
Customer_Type char(1)
Last_Bill_payment char(1) (values could be ‘Y’ or ‘N’)
There are two types of meters.
• 3- phase coded as „T‟
• 1-phase coded as „S‟
There are 4 types of customers.
• Agricultural coded as „A‟
• Industrial coded as „I‟
• Commercial coded as „C‟
• Residential coded as „R‟
Formulae used:
Units used = Current Reading - Previous Reading
Rate = 1/ 1.25/ 1.50/ 1.30 for A/I/C/R respectively.
Amount = rate*units used
Surcharge = 5% for single phase
10% for 3 phase
Excise = 30% of (amount +Surcharge)
Net = Amount +Surcharge + Excise
Write a procedure calculate_bill to calculate the bill for each customer. Consider the following customer table to read input data.

```
CREATE TABLE CUSTOMER (
Meter_Number varchar2(4),
Meter_Type char(1),
Previous_Reading number,
Current_Reading number,
Customer_Type char(1),
Last_Bill_payment char(1),
check(Last_Bill_payment = 'Y' OR Last_Bill_payment = 'N')
);
--insert dummy data into table customer
```

Meter_number	Meter_Type	Previous_Reading	Current_Reading	Customer_Type	Last_Bill_payment
1000	S	3000	5000	A	Y
1001	T	3000	5000	R	Y
1002	S	400	2000	R	Y

After calculating the bill, the procedure should insert the total Amount, Surcharge, Excise and Net into the Bill table as shown below.

```
CREATE TABLE BILL (
Meter_Number varchar2(4) PRIMARY KEY,
units number,
rate number,
amount number,
surcharge number,
Excise number,
Net number
);
```

Code:
```
CREATE TABLE CUSTOMER (
Meter_Number varchar2(4),
Meter_Type char(1),
Previous_Reading number,
Current_Reading number,
Customer_Type char(1),
Last_Bill_payment char(1),
check(Last_Bill_payment = 'Y' OR Last_Bill_payment = 'N')
);

INSERT INTO CUSTOMER VALUES (1000,'S',3000,5000,'A','Y');
INSERT INTO CUSTOMER VALUES (1001,'T',3000,5000,'R','Y');
INSERT INTO CUSTOMER VALUES (1002,'S',400,2000,'R','Y');

CREATE TABLE BILL (
Meter_Number varchar2(4) PRIMARY KEY,
units number,
rate number,
amount number,
surcharge number,
Excise number,
Net number
);

CREATE OR REPLACE PROCEDURE PROC_7_2018_1_60_048
IS
    CURSOR CUSTOMERS IS
        SELECT * FROM CUSTOMER;
    units number;
    rate number;
    amount number;
    surcharge number;
    Excise number;
    Net number;

BEGIN
    dbms_output.put_line('STUDENT ID: 2018-1-60-048');
    FOR i IN CUSTOMERS
        LOOP
            units:=i.current_reading - i.previous_reading;
            rate := 1;

            IF i.customer_type = 'A' THEN
                rate := 1;
            ELSIF i.customer_type  = 'I' THEN
                rate := 1.25;
            ELSIF i.customer_type = 'C' THEN
                rate := 1.50;
            ELSIF i.customer_type = 'R' THEN
```

```
                    rate := 1.30;
                END IF;
                amount:= rate * units;

                IF i.meter_type = 'T' THEN
                    surcharge:=amount*0.1;
                ELSIF i.meter_type = 'S' THEN
                    surcharge:=amount*0.05;
                END IF;
                Excise:= (amount + surcharge) * 0.3;
                Net:= amount + surcharge + Excise;

                INSERT INTO BILL VALUES (i.meter_number,units ,rate, amount, 
surcharge, Excise, Net);
        END LOOP;
END;
/

EXECUTE PROC_7_2018_1_60_048;
SELECT * FROM BILL;
```

Output:

```
Table created. 1 row(s) inserted. 1 row(s) inserted. 1 row(s) inserted. Table created. Procedure
created. Statement processed.
STUDENT ID: 2018-1-60-048
```

Result Set 1

METER_NUMBER	UNITS	RATE	AMOUNT	SURCHARGE	EXCISE	NET
1000	2000	1	2000	100	630	2730
1001	2000	1.3	2600	260	858	3718
1002	1600	1.3	2080	104	655.2	2839.2

Download CSV

```
3 rows selected.
STUDENT ID: 2018-1-60-048
```

8 (Exception) Write a PL/SQL block that prompts the user to enter the salary of an employee. Your program should display the name of the employee (from the EMP table as shown below) who‟s getting that salary. If more than 1 employee is receiving that salary, or if no employees exist getting that salary, your program should display appropriate messages. Use too_many_rows and no_data_found exceptions to achieve this. Display the results on the screen using dbms_output.put_line.

```
CREATE TABLE emp (empno varchar(4), empname varchar(30), designation
varchar2(10), category char(1), basicsalary number(4), joined date
);
```

Code:

```
create table emp1 (
empname varchar(24),
basicsalary number);
INSERT INTO emp1 VALUES('Nasim Bahadur',3000);
INSERT INTO emp1 VALUES('Mustakim Mishu',4000);
INSERT INTO emp1 VALUES('Mokhlesur Rahman',5000);
INSERT INTO emp1 VALUES('Shahida Afrin',6000);
INSERT INTO emp1 VALUES('Kazi Sumaiya Akter',7000);
INSERT INTO emp1 VALUES('Mazharul Islam',2000);
INSERT INTO emp1 VALUES('Mahamud Alam',2342);
INSERT INTO emp1 VALUES('Mamud ur Rahman',7564);
INSERT INTO emp1 VALUES('Tahmid Shahriar',5987);
INSERT INTO emp1 VALUES('Muhit Hasan',2352);
INSERT INTO emp1 VALUES('Ibrahim Khalil',6456);

DECLARE
    emp_salary number := 5000;
    emp_name varchar(24);
BEGIN
   SELECT empname INTO emp_name
   FROM emp1
   WHERE basicsalary = emp_salary;
   DBMS_OUTPUT.PUT_LINE('Employee name is :' || emp_name);
  EXCEPTION
      WHEN TOO_MANY_ROWS THEN
      DBMS_OUTPUT.PUT_LINE ('More than one employee getting the
salary');
      WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE ('No employee name found');

END;

Output:
Table created.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
Statement processed.
Employee name is :Mokhlesur Rahman
```

9	(User-defined Exception) Create a user-defined exception by the name of exp_check. Select the empname and joined date of all employees into a cursor from the emp table as shown in problem

Your program should calculate the experience of all the employees in years, and insert the empname and experience of each employee into tempp table. Create the table beforehand.
If any employee has experience less than 2 years, the program should be aborted with a suitable message. Raise the user-defined exception exp_check to achieve this. Display the results on the screen using dbms_output.put_line.

```
Code:
CREATE TABLE emp2 (empno varchar(24), empname varchar(30), designation
        varchar2(100), categoryy char(12), basicsalary number, joined
date
        );
CREATE TABLE temp (empname varchar(30), joined date);
INSERT INTO emp2 VALUES ('1001','Sabbir','Man','a',50000,date '2022-1-
12');
INSERT INTO emp2 VALUES ('1002','Tanvir','Assist','b',35000,date '2022-
1-12');
INSERT INTO emp2 VALUES ('1002','Suzon','Mana','a',50000,date '2012-1-
12');
INSERT INTO emp2 VALUES ('1003','Tasnuva','Clark','b',15000, date '2021-
11-18');
INSERT INTO emp2 VALUES ('1004','Shoili','Mana','a',50000,date '2012-1-
22');



DECLARE
exp_check EXCEPTION;
CURSOR C1 IS
SELECT empname, joined FROM emp2;
BEGIN
FOR i IN C1
LOOP
IF(trunc(months_between(sysdate,i.joined)/12)<2) THEN
RAISE exp_check;
ELSE
INSERT INTO temp VALUES(i.empname,
trunc(months_between(sysdate,i.joined)/12));
END IF;
END LOOP;
EXCEPTION
WHEN exp_check THEN
DBMS_OUTPUT.PUT_LINE('Experiance is less then 2 years not allowed');
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Unidentified error occured');
END;
```

10 (Function) Write a PL/SQL function to take three parameters, the sides of a triangle. The sides of the triangle should be accepted from the user. The function should return a Boolean value - true

if the triangle is valid, false otherwise. A triangle is valid if the length of each side is less than the sum of the lengths of the other two sides. Check if the dimensions entered by the user can form a valid triangle.
Display the results on the screen using dbms_output.put_line.

Code:

```
CREATE OR REPLACE FUNCTION FUNC_8_2018_1_60_048 (side1 number,side2
number, side3 number)
RETURN BOOLEAN
IS
BEGIN
    dbms_output.put_line('STUDENT ID: 2018-1-60-048');
    IF side1 + side2 > side3 AND
        side1 + side3 > side2 AND
        side3 + side2 > side1 THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
END;
/


CREATE OR REPLACE PROCEDURE PROC_8_2018_1_60_048
IS
side1 number := 5;
side2 number := 6;
side3 number := 10;
BEGIN
    IF FUNC_8_2018_1_60_048(side1,side2,side3) THEN
        dbms_output.put_line(side1||', '||side2||' and '||side3||' form
a valid triangle.');
    ELSE
        dbms_output.put_line(side1||', '||side2||' and '||side3||'  do
not form a valid triangle.');
    END IF;
END;
/

EXECUTE PROC_8_2018_1_60_048;
```

Output:

```
Function created. Procedure created. Statement processed.
STUDENT ID: 2018-1-60-048
5, 6 and 10 form a valid triangle.
```