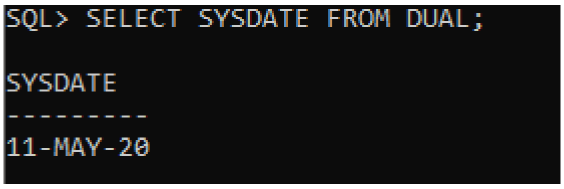
Anubhav singh

IC2K1654

Assignment 4

Date and Time Function

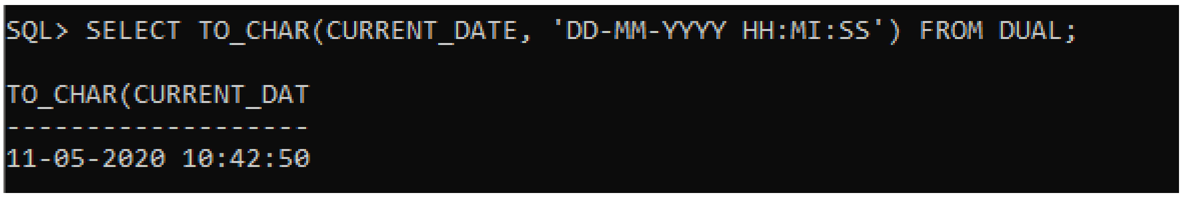
1.



```
SQL> SELECT SYSDATE FROM DUAL;

SYSDATE
---------
11-MAY-20
```

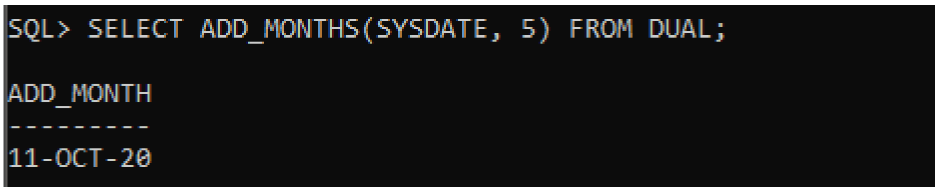
2.



```
SQL> SELECT TO_CHAR(CURRENT_DATE, 'DD-MM-YYYY HH:MI:SS') FROM DUAL;

TO_CHAR(CURRENT_DAT
-------------------
11-05-2020 10:42:50
```

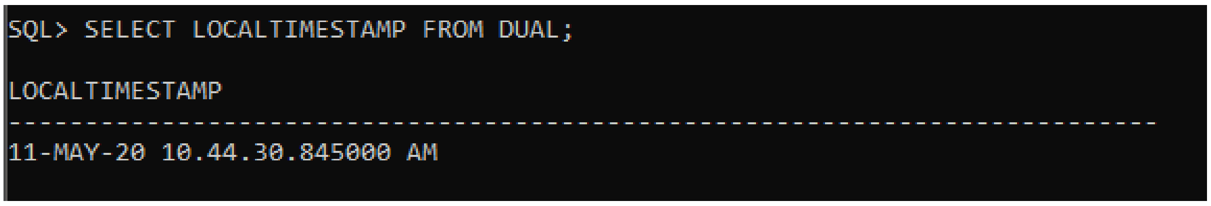
3.



```
SQL> SELECT ADD_MONTHS(SYSDATE, 5) FROM DUAL;

ADD_MONTH
---------
11-OCT-20
```

4.



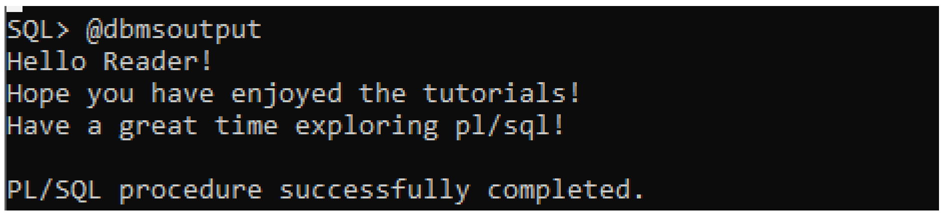
```
SQL> SELECT LOCALTIMESTAMP FROM DUAL;

LOCALTIMESTAMP
---------------------------------------------------------------------------
11-MAY-20 10.44.30.845000 AM
```

DBMS OUTPUT

1. Using DBMS_OUTPUT function

```
DECLARE
lines dbms_output.chararr;
num_lines number;
BEGIN
dbms_output.enable;
dbms_output.put_line('Hello Reader!');
dbms_output.put_line('Hope you have enjoyed the tutorials!');
dbms_output.put_line('Have a great time exploring pl/sql!');
num_lines := 3;
dbms_output.get_lines(lines, num_lines);
FOR i IN 1..num_lines
LOOP
dbms_output.put_line(lines(i));
END LOOP;
END;
/
```

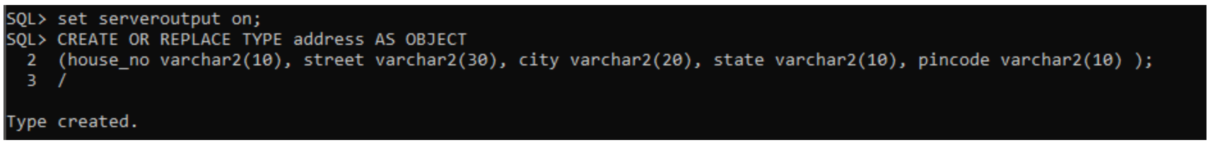


```
SQL> @dbmsoutput
Hello Reader!
Hope you have enjoyed the tutorials!
Have a great time exploring pl/sql!

PL/SQL procedure successfully completed.
```

OBJECT ORIENTED

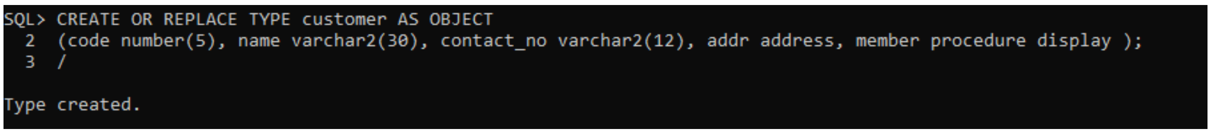
1. **Type Creation**



```
SQL> set serveroutput on;
SQL> CREATE OR REPLACE TYPE address AS OBJECT
  (house_no varchar2(10), street varchar2(30), city varchar2(20), state varchar2(10), pincode varchar2(10) );
  /

Type created.
```

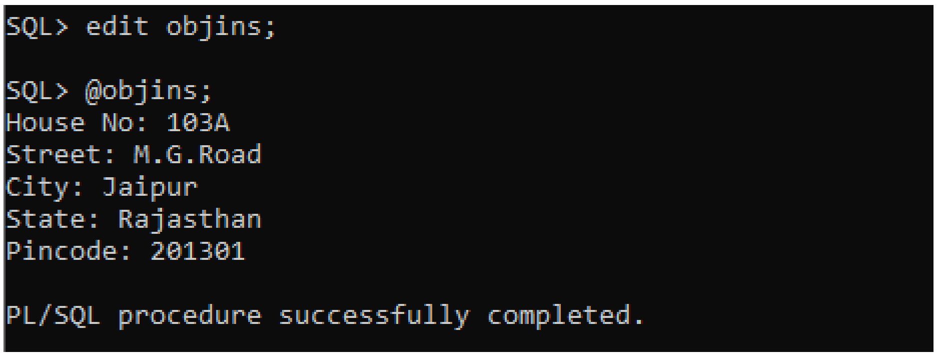
2. **Type Creation**



```
SQL> CREATE OR REPLACE TYPE customer AS OBJECT
  (code number(5), name varchar2(30), contact_no varchar2(12), addr address, member procedure display );
  /

Type created.
```

3. **Instantiating Object**

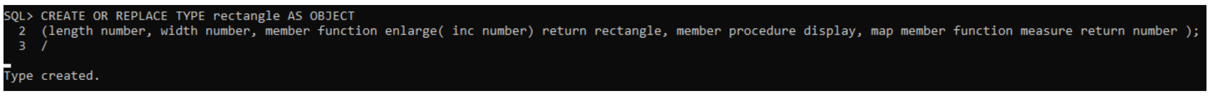


```
SQL> edit objins;

SQL> @objins;
House No: 103A
Street: M.G.Road
City: Jaipur
State: Rajasthan
Pincode: 201301

PL/SQL procedure successfully completed.
```

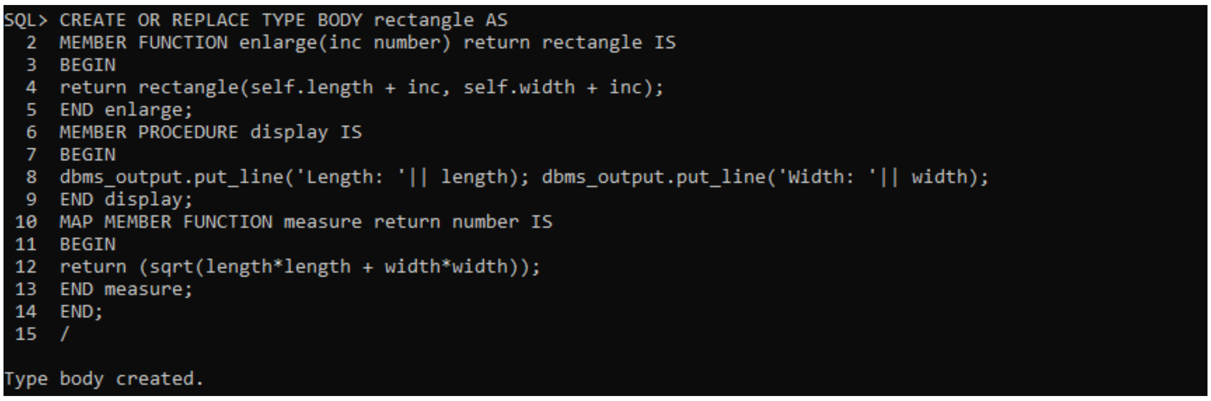
4. **Map method**



```
SQL> CREATE OR REPLACE TYPE rectangle AS OBJECT
  (length number, width number, member function enlarge( inc number) return rectangle, member procedure display, map member function measure return number );
  /

Type created.
```

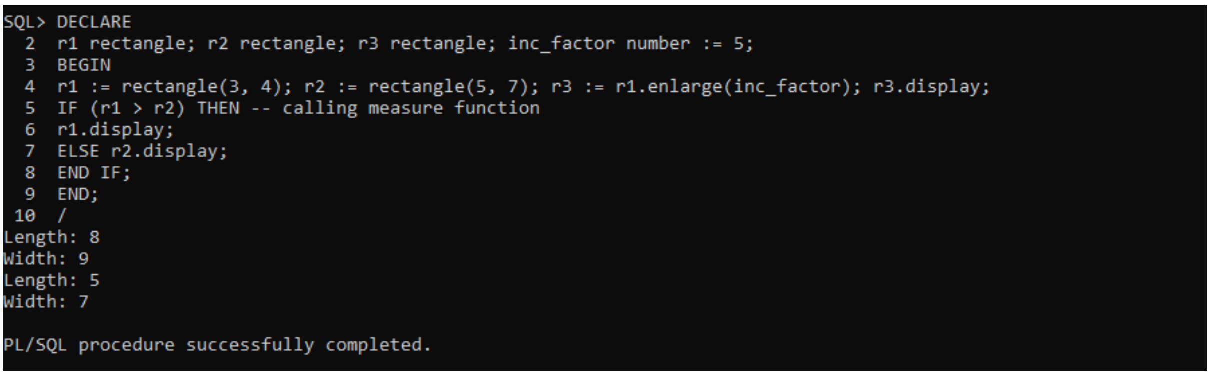
5. **Type creation using Map**



```
SQL> CREATE OR REPLACE TYPE BODY rectangle AS
  MEMBER FUNCTION enlarge(inc number) return rectangle IS
  BEGIN
  return rectangle(self.length + inc, self.width + inc);
  END enlarge;
  MEMBER PROCEDURE display IS
  BEGIN
  dbms_output.put_line('Length: '|| length); dbms_output.put_line('Width: '|| width);
  END display;
  MAP MEMBER FUNCTION measure return number IS
  BEGIN
  return (sqrt(length*length + width*width));
  END measure;
  END;
  /

Type body created.
```

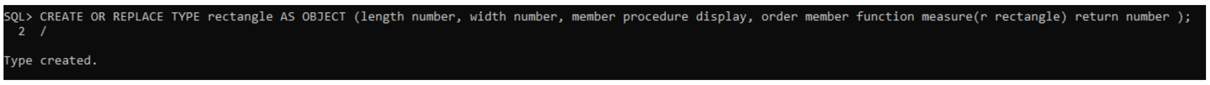
6. **Using Type body**



```
SQL> DECLARE
r1 rectangle; r2 rectangle; r3 rectangle; inc_factor number := 5;
BEGIN
r1 := rectangle(3, 4); r2 := rectangle(5, 7); r3 := r1.enlarge(inc_factor); r3.display;
IF (r1 > r2) THEN -- calling measure function
r1.display;
ELSE r2.display;
END IF;
END;
/
Length: 8
Width: 9
Length: 5
Width: 7

PL/SQL procedure successfully completed.
```

7. **Order method**

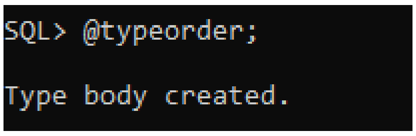


```
SQL> CREATE OR REPLACE TYPE rectangle AS OBJECT (length number, width number, member procedure display, order member function measure(r rectangle) return number );
/

Type created.
```

8. **Body of Procedure**

CREATE OR REPLACE TYPE BODY rectangle AS
MEMBER PROCEDURE display IS
BEGIN
dbms_output.put_line('Length: '|| length);
dbms_output.put_line('Width: '|| width);
END display;
ORDER MEMBER FUNCTION measure(r rectangle) return number IS
BEGIN
IF(sqrt(self.length*self.length + self.width*self.width)> sqrt(r.length*r.length + r.width*r.width))
then return(1);
ELSE
return(-1);
END IF;
END measure;
END;
/

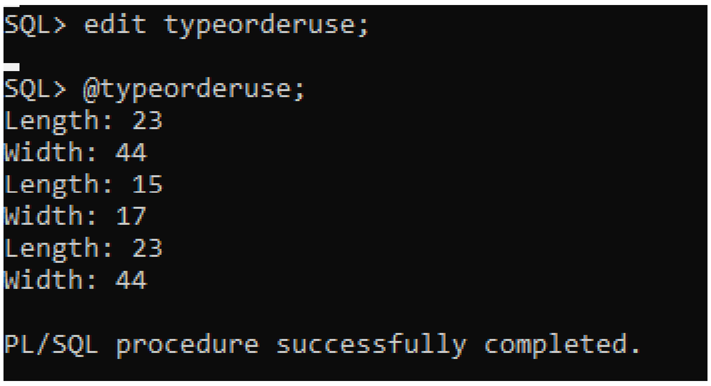


```
SQL> @typeorder;

Type body created.
```

9. **Use Type order**
DECLARE

```
r1 rectangle;
r2 rectangle;
BEGIN
r1 := rectangle(23, 44);
r2 := rectangle(15, 17);
r1.display;
r2.display;
IF (r1 > r2)
THEN
r1.display;
ELSE r2.display;
END IF;
END;
/
```

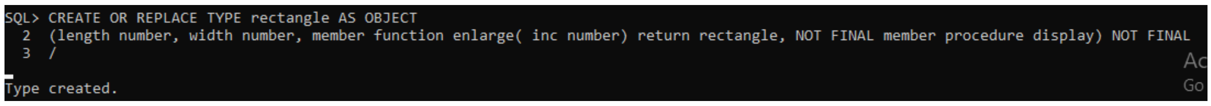


```
SQL> edit typeorderuse;

SQL> @typeorderuse;
Length: 23
Width: 44
Length: 15
Width: 17
Length: 23
Width: 44

PL/SQL procedure successfully completed.
```

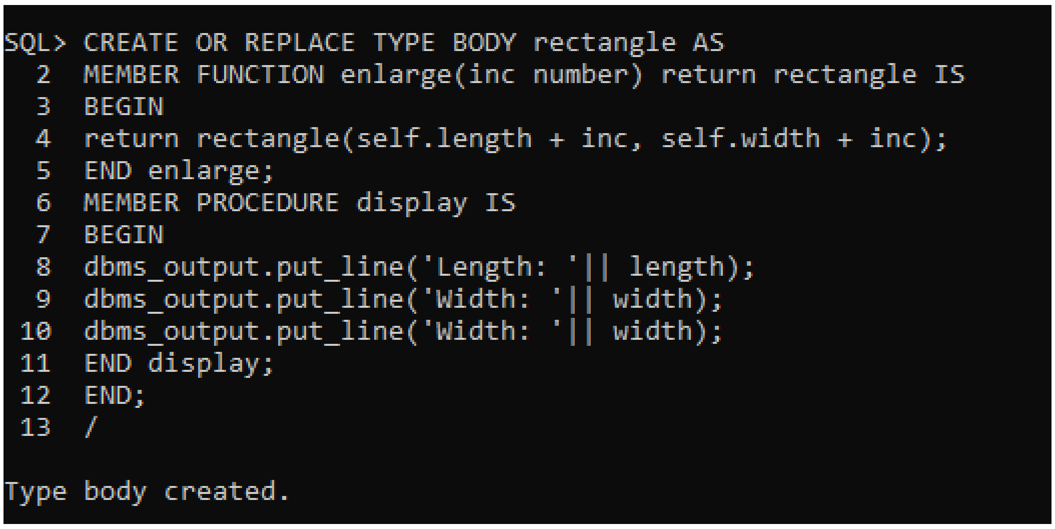
10. Inheritance



```
SQL> CREATE OR REPLACE TYPE rectangle AS OBJECT
  (length number, width number, member function enlarge( inc number) return rectangle, NOT FINAL member procedure display) NOT FINAL
  /

Type created.
```

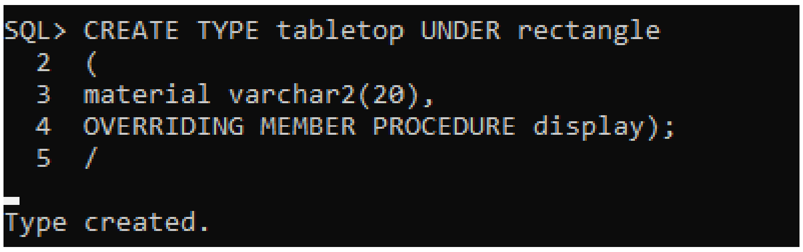
11. Type Creation



```
SQL> CREATE OR REPLACE TYPE BODY rectangle AS
  MEMBER FUNCTION enlarge(inc number) return rectangle IS
  BEGIN
  return rectangle(self.length + inc, self.width + inc);
  END enlarge;
  MEMBER PROCEDURE display IS
  BEGIN
  dbms_output.put_line('Length: '|| length);
  dbms_output.put_line('Width: '|| width);
  dbms_output.put_line('Width: '|| width);
  END display;
  END;
  /

Type body created.
```

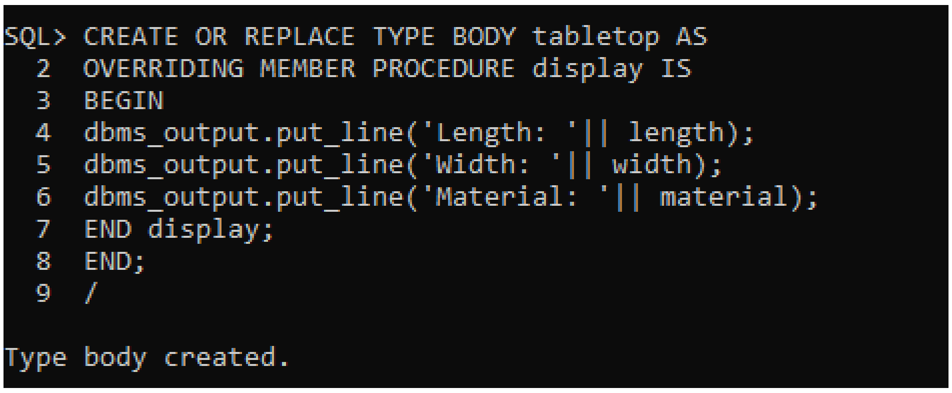
12. Child creation



```
SQL> CREATE TYPE tabletop UNDER rectangle
     (
     material varchar2(20),
     OVERRIDING MEMBER PROCEDURE display);
     /

Type created.
```

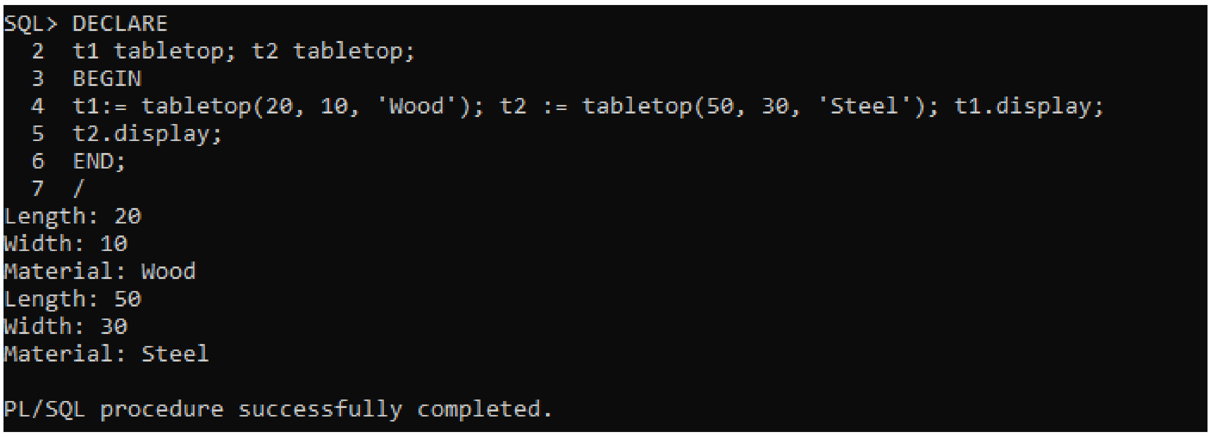
13. Child Body



```
SQL> CREATE OR REPLACE TYPE BODY tabletop AS
     OVERRIDING MEMBER PROCEDURE display IS
     BEGIN
     dbms_output.put_line('Length: '|| length);
     dbms_output.put_line('Width: '|| width);
     dbms_output.put_line('Material: '|| material);
     END display;
     END;
     /

Type body created.
```

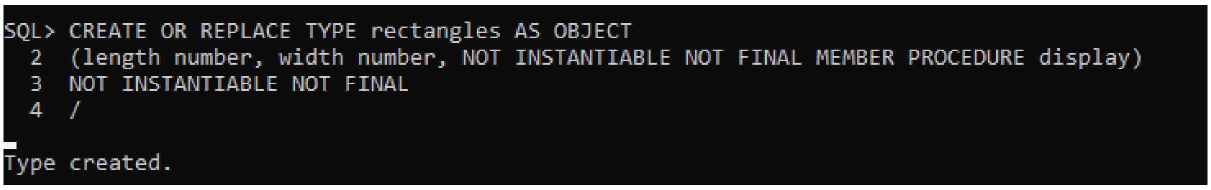
14. Performing Inheritance



```
SQL> DECLARE
     t1 tabletop; t2 tabletop;
     BEGIN
     t1:= tabletop(20, 10, 'Wood'); t2 := tabletop(50, 30, 'Steel'); t1.display;
     t2.display;
     END;
     /
Length: 20
Width: 10
Material: Wood
Length: 50
Width: 30
Material: Steel

PL/SQL procedure successfully completed.
```

15. Abstract Object



```
SQL> CREATE OR REPLACE TYPE rectangles AS OBJECT
     (length number, width number, NOT INSTANTIABLE NOT FINAL MEMBER PROCEDURE display)
     NOT INSTANTIABLE NOT FINAL
     /

Type created.
```