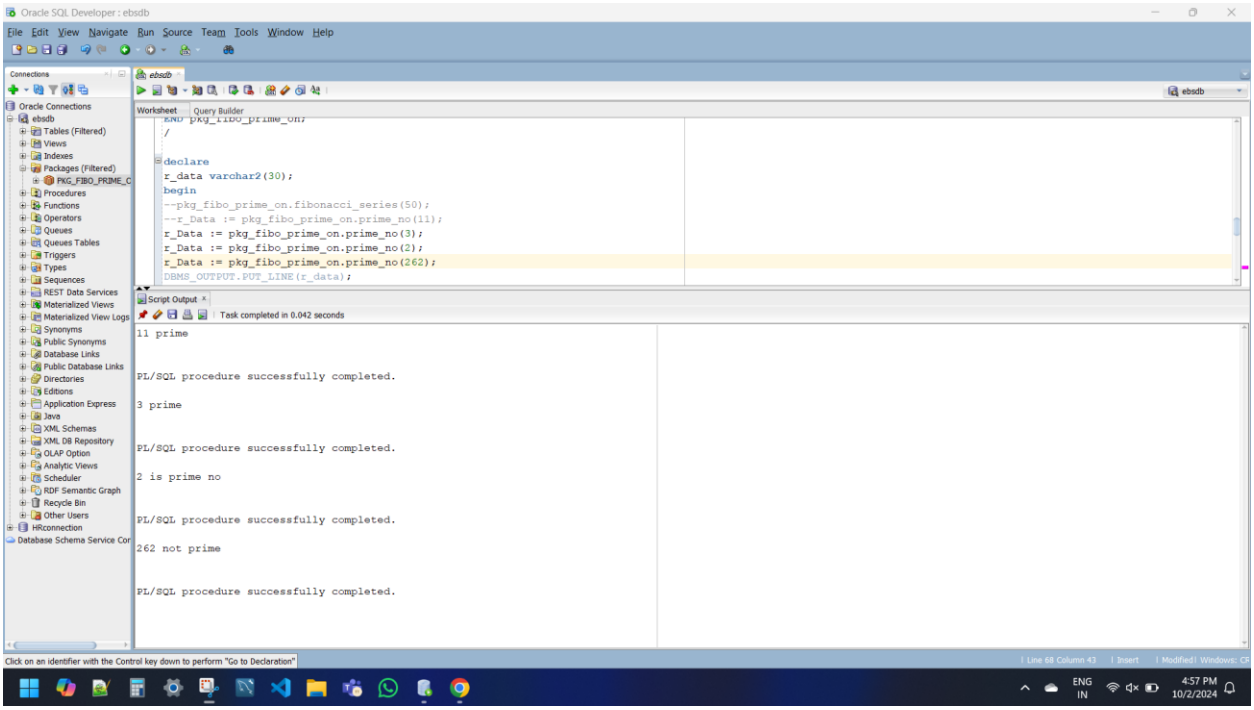
-- Packages ----

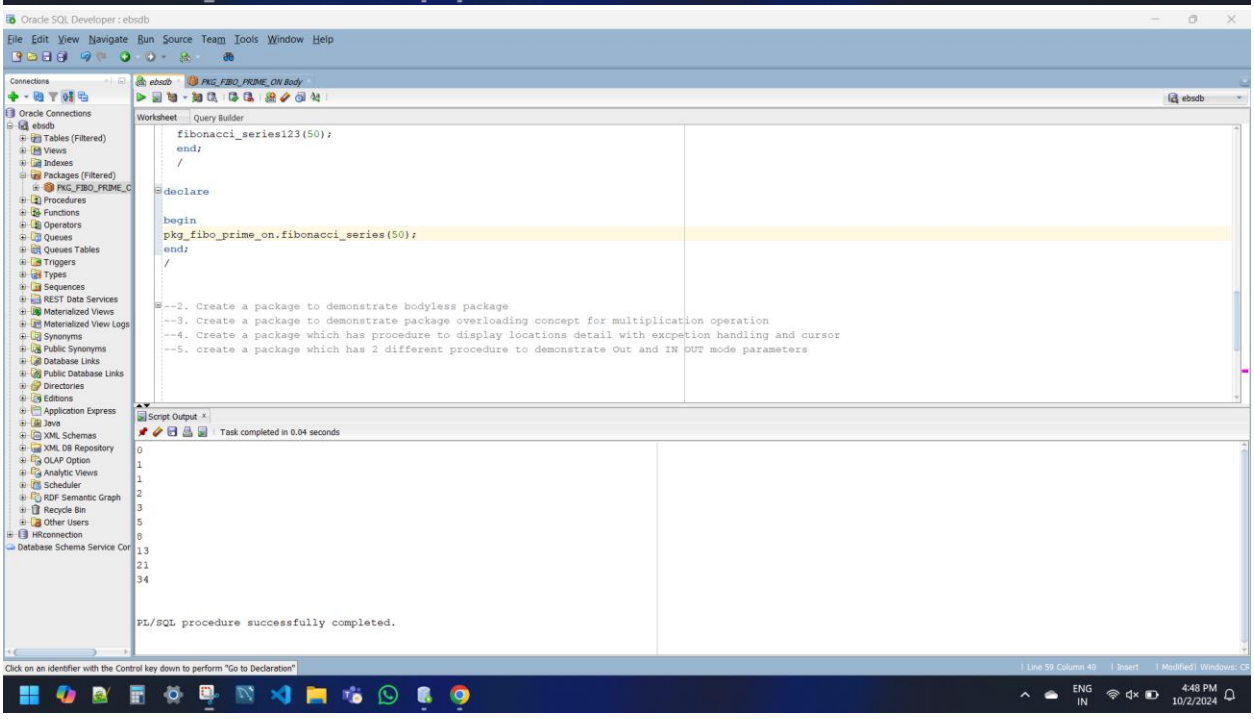
--1. Create a package which has a procedure to display fibonacci series and a function to validate if the inserted number is prime number or not

```
create or replace package pkg_fibo_prime_on is

procedure fibonacci_series(f_num in number);

function prime_no(p_num number) return varchar2;

end pkg_fibo_prime_on;

/


CREATE OR REPLACE PACKAGE BODY pkg_fibo_prime_on AS


  PROCEDURE fibonacci_series (
    f_num IN NUMBER
  ) AS


    curr_no NUMBER(8) := 0;
    next_no NUMBER(8) := 1;
    total   NUMBER(8, 2) := 0;
  BEGIN
    dbms_output.put_line(curr_no);
    dbms_output.put_line(next_no);
    LOOP
      total := curr_no + next_no;
      EXIT WHEN total > f_num;
      dbms_output.put_line(total);
```

```
      curr_no := next_no;
      next_no := total;
   END LOOP;

END fibonacci_series;

FUNCTION prime_no (
   p_num NUMBER
) RETURN VARCHAR2 AS
   is_prime BOOLEAN := true;
   r_data   VARCHAR2(50);
BEGIN
   FOR i IN 2..( p_num - 1 ) LOOP
      IF MOD(p_num, i) = 0 THEN
         is_prime := false;
         EXIT;
      END IF;
   END LOOP;

   IF is_prime = false THEN
      r_data := p_num || ' not prime';
   ELSE
      r_data := p_num || ' prime';
   END IF;

if p_num = 2 then
```

```
   r_Data := '2 is prime no';
   end if;

      RETURN r_data;
   END prime_no;

END pkg_fibo_prime_on;
/

declare
r_data varchar2(30);
begin
--pkg_fibo_prime_on.fibonacci_series(50);
--r_Data := pkg_fibo_prime_on.prime_no(11);
r_Data := pkg_fibo_prime_on.prime_no(3);
r_Data := pkg_fibo_prime_on.prime_no(2);
r_Data := pkg_fibo_prime_on.prime_no(262);
DBMS_OUTPUT.PUT_LINE(r_data);
end;
/
```





--2. Create a package to demonstrate bodyless package

create or replace package bodyless_pkg is

pi_value constant number(10,9) := 3.14658945;

meter_to_feet constant number(5,2):= 2.2;

end bodyless_pkg;

/

--3. Create a package to demonstrate package overloading concept for multiplication operation

set SERVEROUTPUT on;

create or replace package multiplication_overloading is

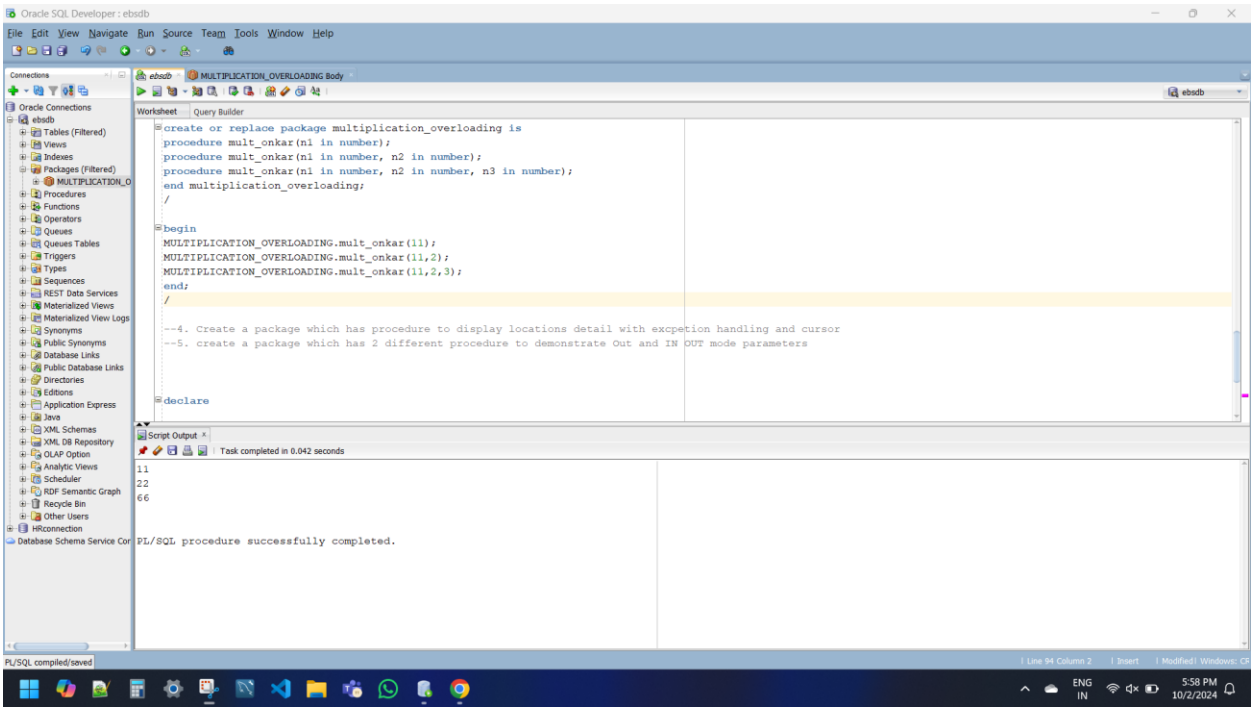
procedure mult_onkar(n1 in number);

procedure mult_onkar(n1 in number, n2 in number);

procedure mult_onkar(n1 in number, n2 in number, n3 in number);

end multiplication_overloading;

/

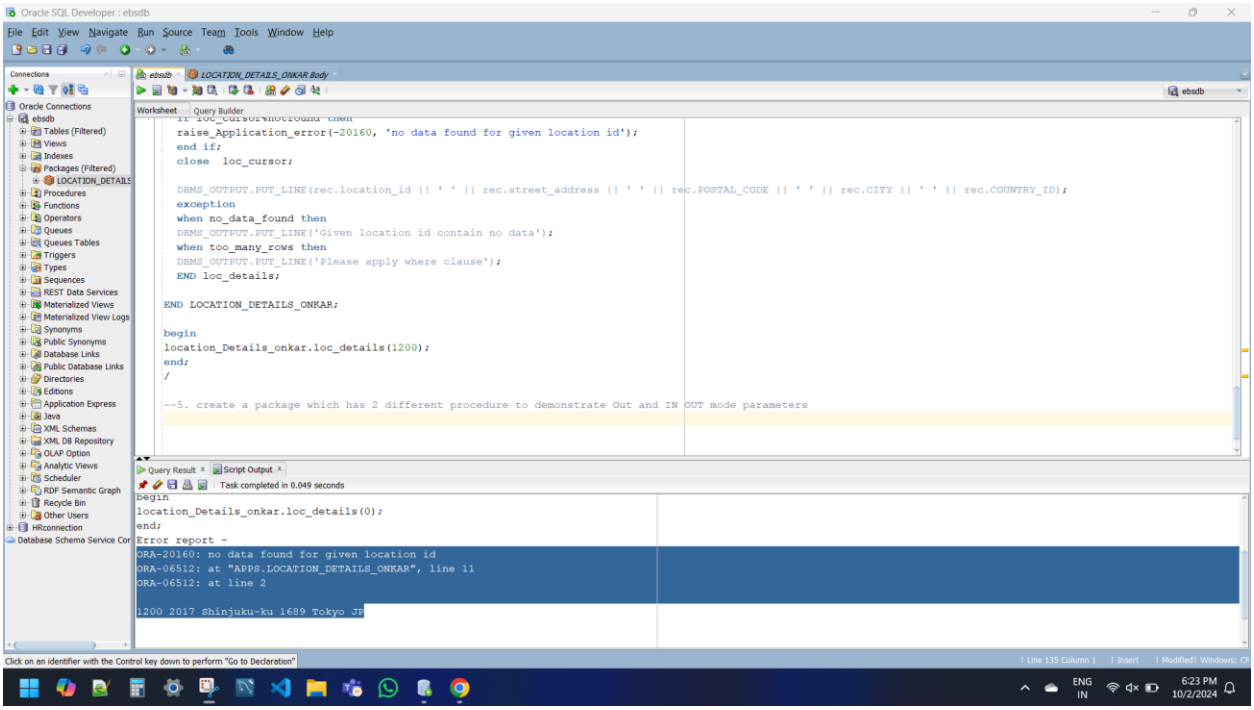


begin

MULTIPLICATION_OVERLOADING.mult_onkar(11);

```
MULTIPLICATION_OVERLOADING.mult_onkar(11,2);
MULTIPLICATION_OVERLOADING.mult_onkar(11,2,3);
end;
/

--4. Create a package which has procedure to display locations detail with excpetion handling and cursor
select * from locations;

create or replace package location_Details_onkar is
procedure loc_details(l_id in number);
end location_Details_onkar;
/

CREATE OR REPLACE
PACKAGE BODY LOCATION_DETAILS_ONKAR AS

 procedure loc_details(l_id in number) AS
 cursor loc_cursor is select * from locations where location_id = l_id;
 rec locations%rowtype;

 BEGIN
 open loc_cursor;
 fetch loc_cursor into rec;
 if loc_cursor%notfound then
 raise_Application_error(-20160, 'no data found for given location id');
```

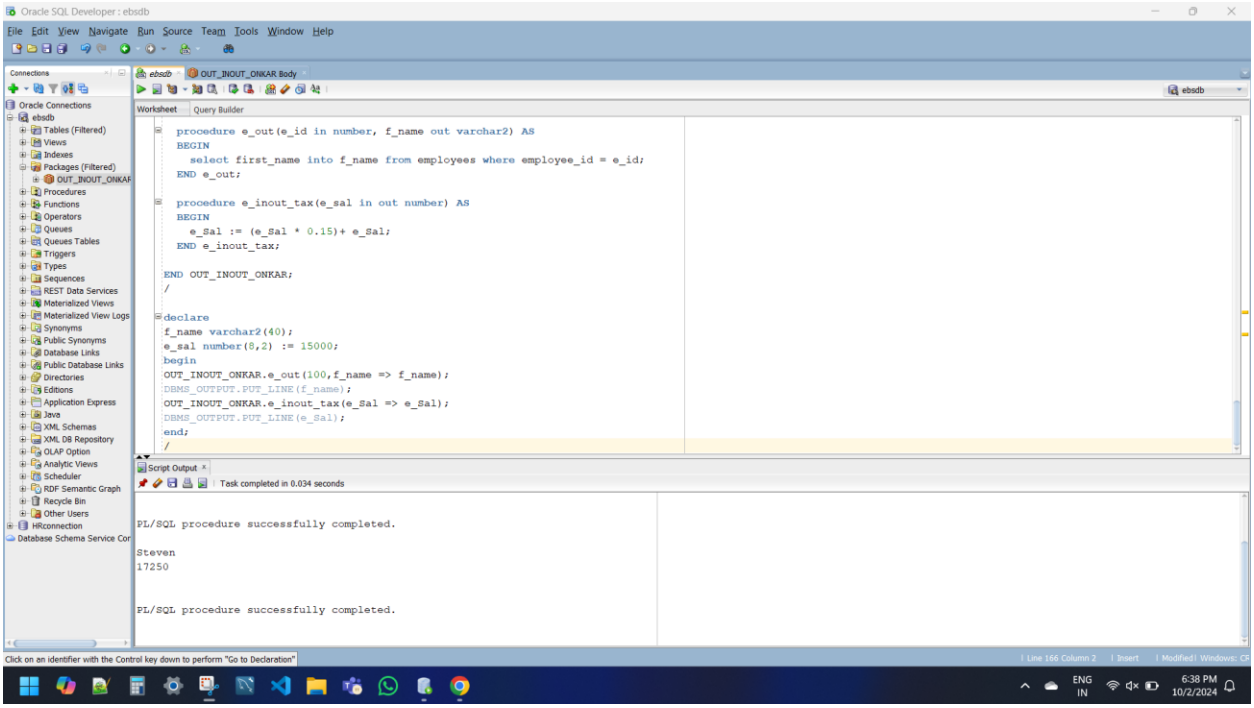
```
 end if;
 close  loc_cursor;

 DBMS_OUTPUT.PUT_LINE(rec.location_id || ' ' || rec.street_address || ' ' ||
rec.POSTAL_CODE || ' ' || rec.CITY || ' ' || rec.COUNTRY_ID);
 exception
 when no_data_found then
 DBMS_OUTPUT.PUT_LINE('Given location id contain no data');
 when too_many_rows then
 DBMS_OUTPUT.PUT_LINE('Please apply where clause');
 END loc_details;

END LOCATION_DETAILS_ONKAR;
/

begin
location_Details_onkar.loc_details(1200);
end;
/
```



--5. create a package which has 2 different procedure to demonstrate Out and IN OUT mode parameters

```
create or replace package out_inout_onkar is

procedure e_out(e_id in number, f_name out varchar2);

procedure e_inout_tax(e_sal in out number);

end out_inout_onkar;


CREATE OR REPLACE

PACKAGE BODY OUT_INOUT_ONKAR AS


 procedure e_out(e_id in number, f_name out varchar2) AS

 BEGIN

  select first_name into f_name from employees where employee_id = e_id;

 END e_out;
```

```
 procedure e_inout_tax(e_sal in out number) AS
 BEGIN
   e_Sal := (e_Sal * 0.15)+ e_Sal;
 END e_inout_tax;

END OUT_INOUT_ONKAR;
/

declare
f_name varchar2(40);
e_sal number(8,2) := 15000;
begin
OUT_INOUT_ONKAR.e_out(100,f_name => f_name);
DBMS_OUTPUT.PUT_LINE(f_name);
OUT_INOUT_ONKAR.e_inout_tax(e_Sal => e_Sal);
DBMS_OUTPUT.PUT_LINE(e_Sal);
end;
/
```



-- Trigger ------

--1. create a trigger to allow DML operation on dept table only on 5 working days

create table dept_onkar as select * from departments;

select * from dept_onkar;

create or replace trigger working_Days_onkar before update or DELETE or INSERT on dept_onkar

begin

if to_Char(sysdate, 'DY') not in ('MON','TUE','THU','FRI','SAT') then

raise_application_error(-20316,'YOU CANNOT WORK ON wed AND SUN');

end if;

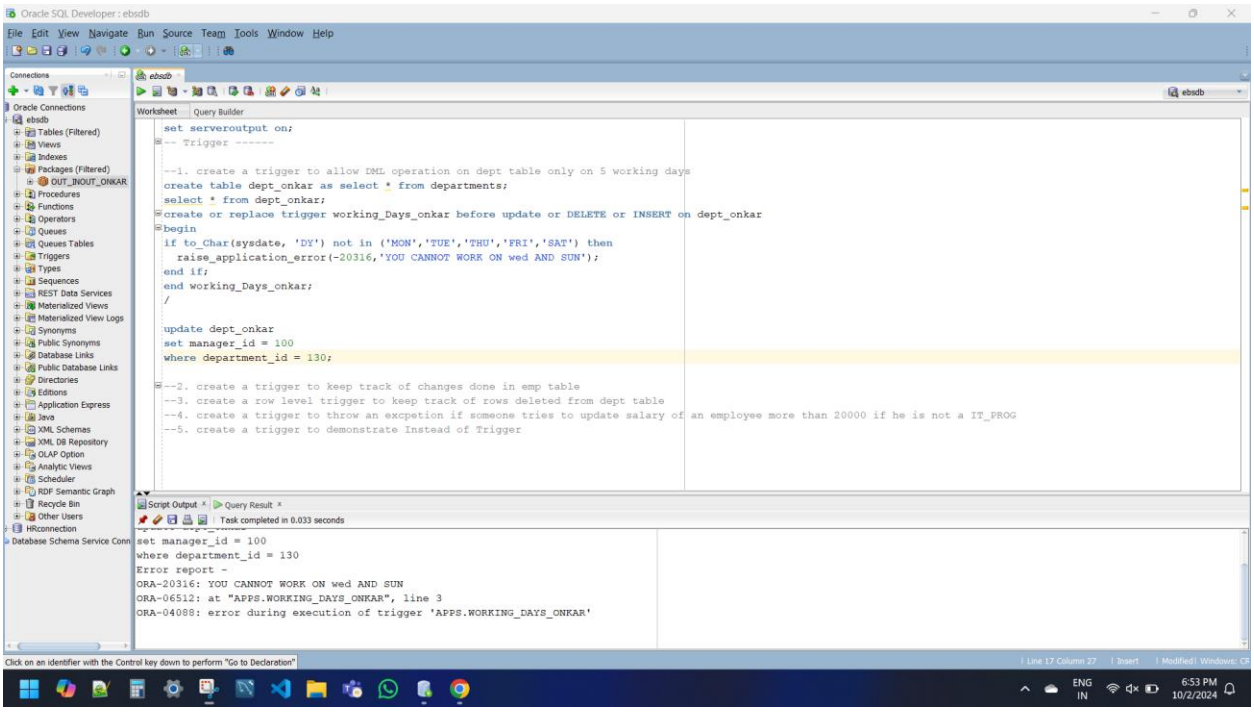
end working_Days_onkar;

/

update dept_onkar

set manager_id = 100

where department_id = 130;



--2. create a trigger to keep track of changes done in emp table

select * from emp_onkar;

create table changes_on_emp(username varchar2(20), changes_time TIMESTAMP, dml_ops varchar2(20));

select * from changes_on_emp;

create or replace trigger changes_on_emp_table AFTER UPDATE or DELETE or INSERT on emp_onkar

begin

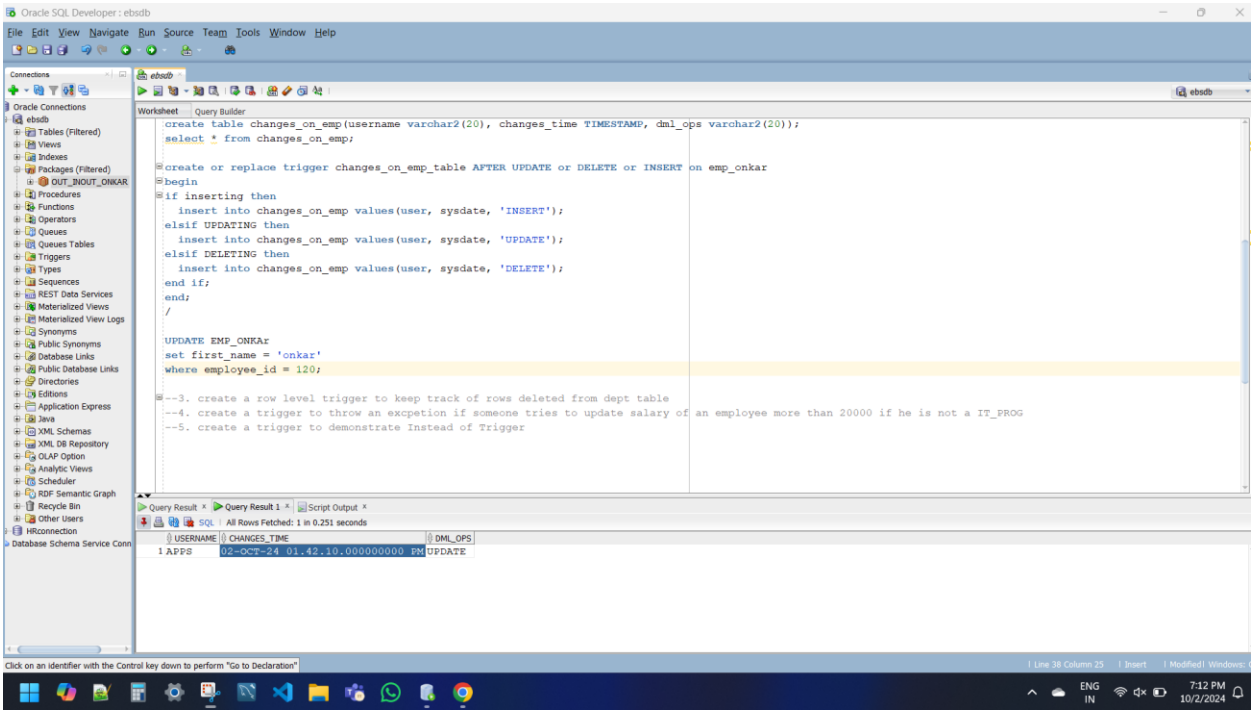
if inserting then

insert into changes_on_emp values(user, sysdate, 'INSERT');

elsif UPDATING then

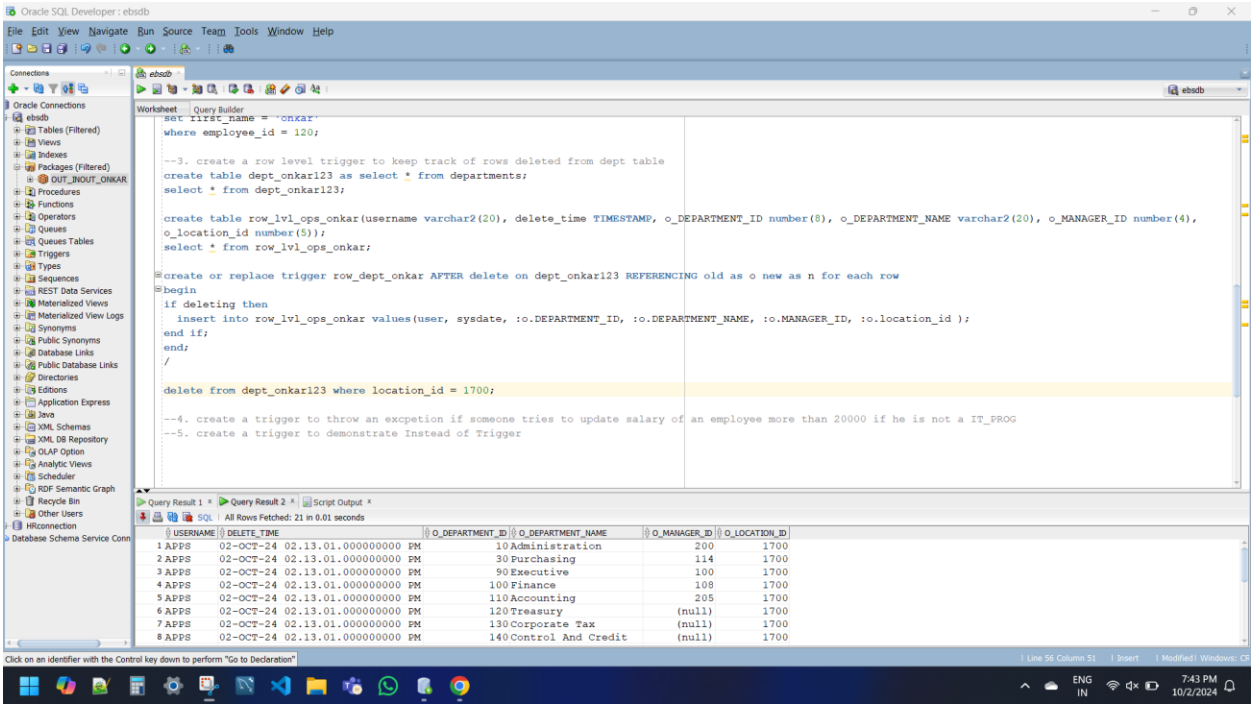
insert into changes_on_emp values(user, sysdate, 'UPDATE');

```
elsif DELETING then
  insert into changes_on_emp values(user, sysdate, 'DELETE');
end if;
end;
/

UPDATE EMP_ONKAr
set first_name = 'onkar'
where employee_id = 120;
```




```
--3. create a row level trigger to keep track of rows deleted from dept table
create table dept_onkar123 as select * from departments;
select * from dept_onkar123;
```

```
create table row_lvl_ops_onkar(username varchar2(20), delete_time TIMESTAMP,
o_DEPARTMENT_ID number(8), o_DEPARTMENT_NAME varchar2(20), o_MANAGER_ID
number(4),

o_location_id number(5));

select * from row_lvl_ops_onkar;


create or replace trigger row_dept_onkar AFTER delete on dept_onkar123 REFERENCING
old as o new as n for each row

begin

if deleting then

  insert into row_lvl_ops_onkar values(user,
sysdate, :o.DEPARTMENT_ID, :o.DEPARTMENT_NAME, :o.MANAGER_ID, :o.location_id );

end if;

end;

/


delete from dept_onkar123 where location_id = 1700;
```



--4. create a trigger to throw an excpetion if someone tries to update salary of an employee more than 20000 if he is not a IT_PROG

create table emp_onkar123 as select * from employees;

select * from emp_onkar123;

create or replace trigger update_sal_emp_trigger BEFORE update on emp_onkar123

REFERENCING new as n old as o

for each row

when(n.job_id != 'IT_PROG')

BEGIN

IF :n.salary>20000 then

raise_Application_error(-20456,'cannot update salary of non IT emp more than 20000');

end if;

END update_sal_emp_trigger;

/

update emp_onkar123

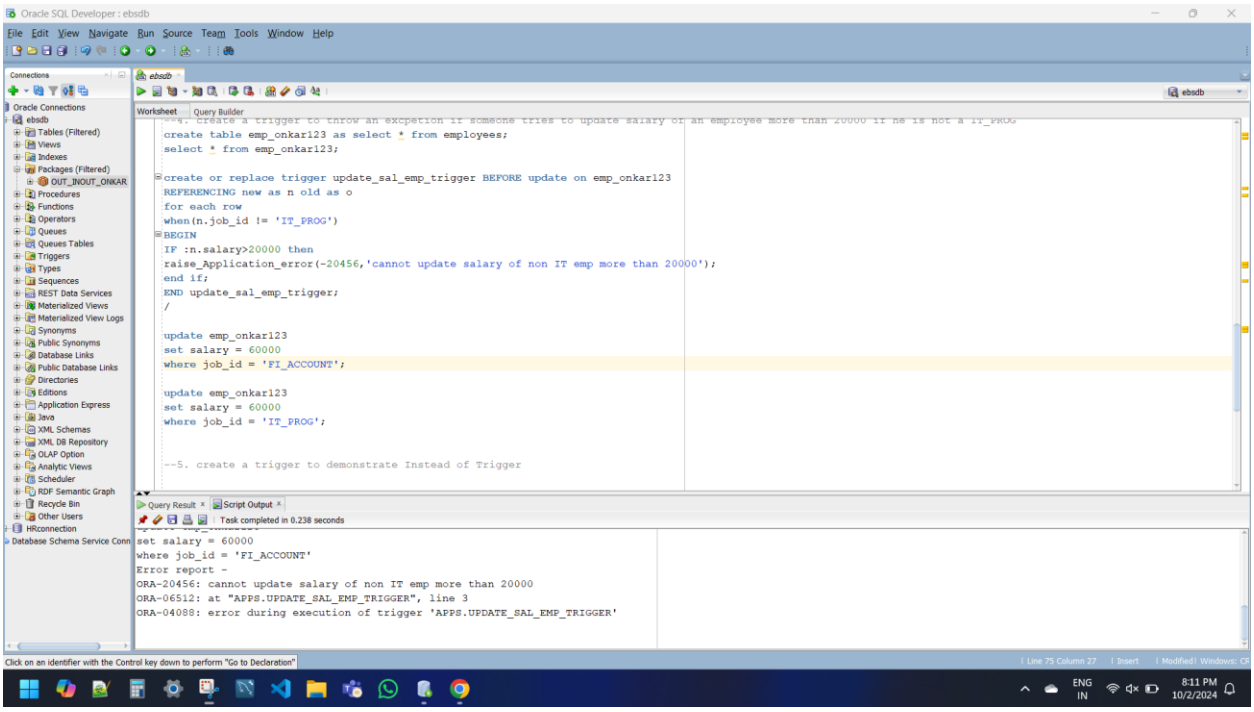
set salary = 60000

where job_id = 'FI_ACCOUNT';

update emp_onkar123

set salary = 60000

where job_id = 'IT_PROG';



--5. create a trigger to demonstrate Instead of Trigger

create view comp_view_onkar as select job_id, count(*) total_emp from emp_onkar group by job_id;

select * from comp_view_onkar;

drop view comp_view_onkar;

```
create or replace trigger instead_of_update instead of update on comp_view_onkar
REFERENCING

old as o new as n

for each row

begin

update emp_onkar

set job_id = :n.job_id

where job_id = :o.job_id;

end instead_of_update;

/


update comp_view_onkar

set job_id = 'chaimen'

where job_id = 'AD_VP';
```