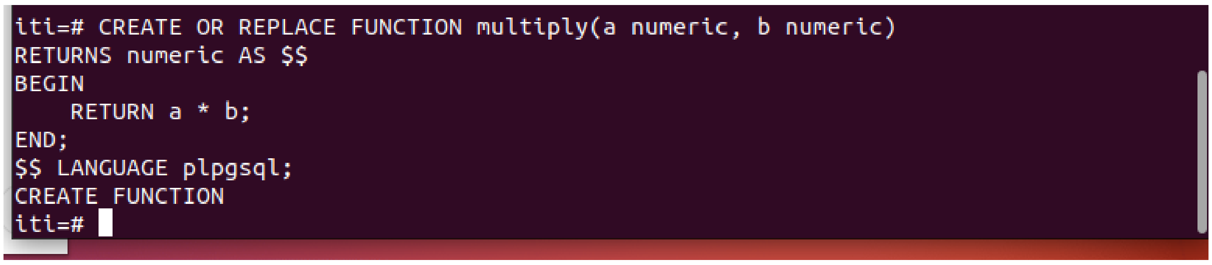
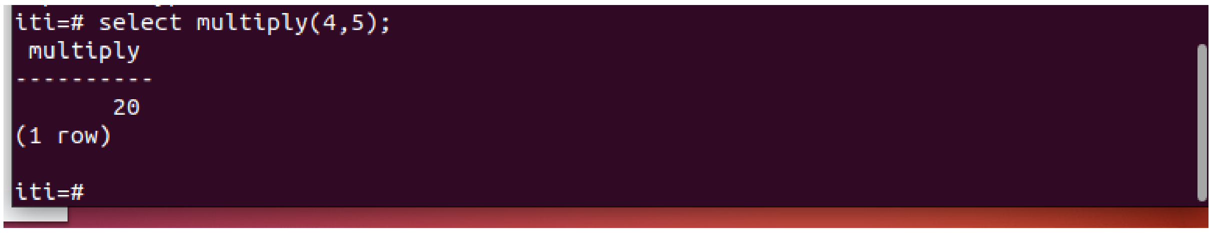
Lab 4

1. Create a multiply function that accepts two numbers and returns their product.

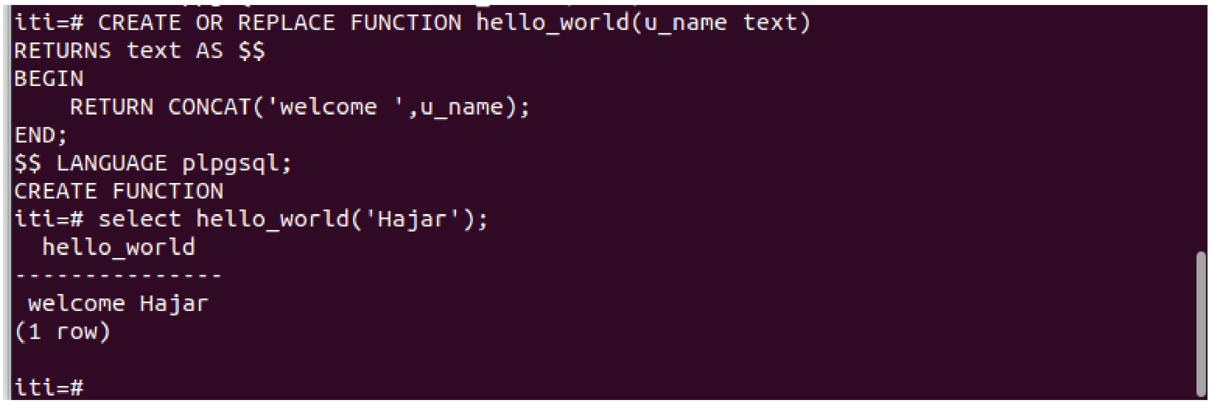


```
iti=# CREATE OR REPLACE FUNCTION multiply(a numeric, b numeric)
RETURNS numeric AS $$
BEGIN
    RETURN a * b;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
iti=#
```



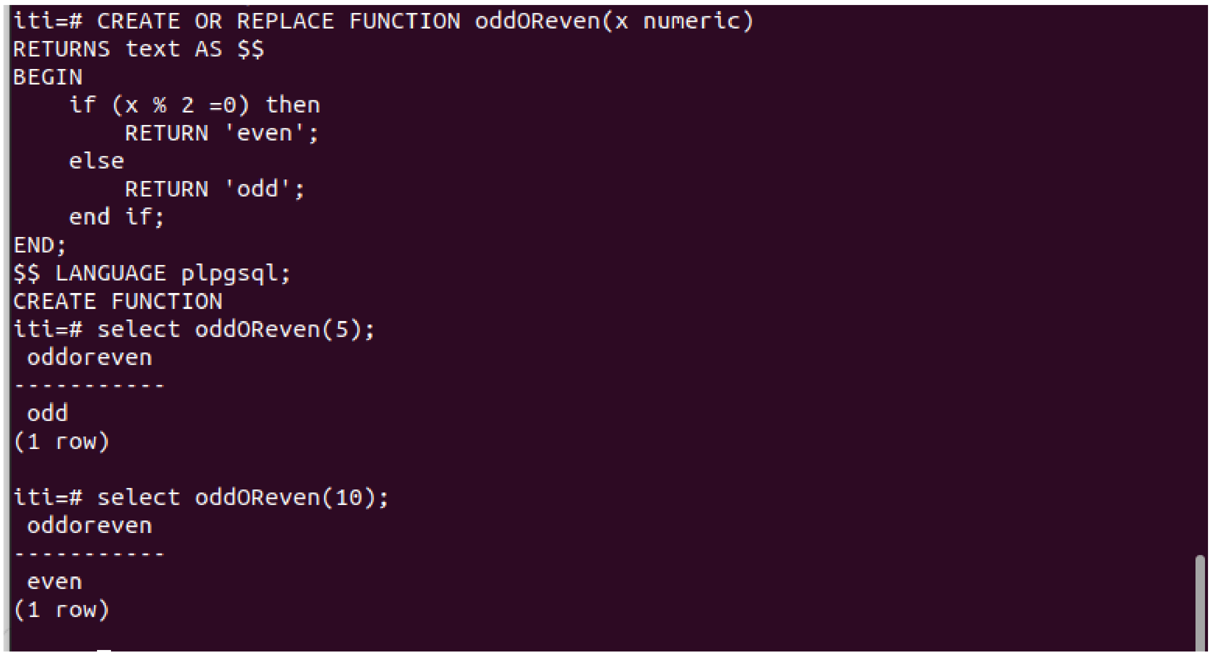
```
iti=# select multiply(4,5);
 multiply
----------
       20
(1 row)

iti=#
```

2. Create a hello_world function that takes a name as input and returns a personalized welcome message for that name.



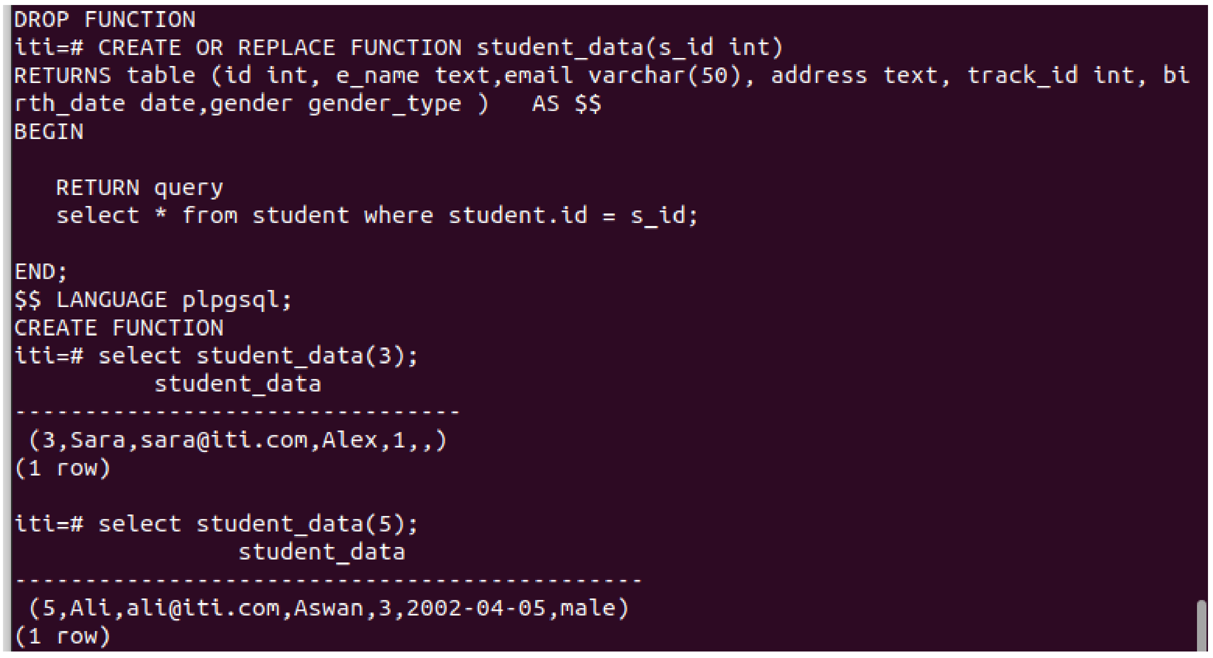
```
iti=# CREATE OR REPLACE FUNCTION hello_world(u_name text)
RETURNS text AS $$
BEGIN
    RETURN CONCAT('welcome ',u_name);
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
iti=# select hello_world('Hajar');
  hello_world
---------------
 welcome Hajar
(1 row)

iti=#
```

3. Create a function that accepts a number and determines whether it is odd or even.



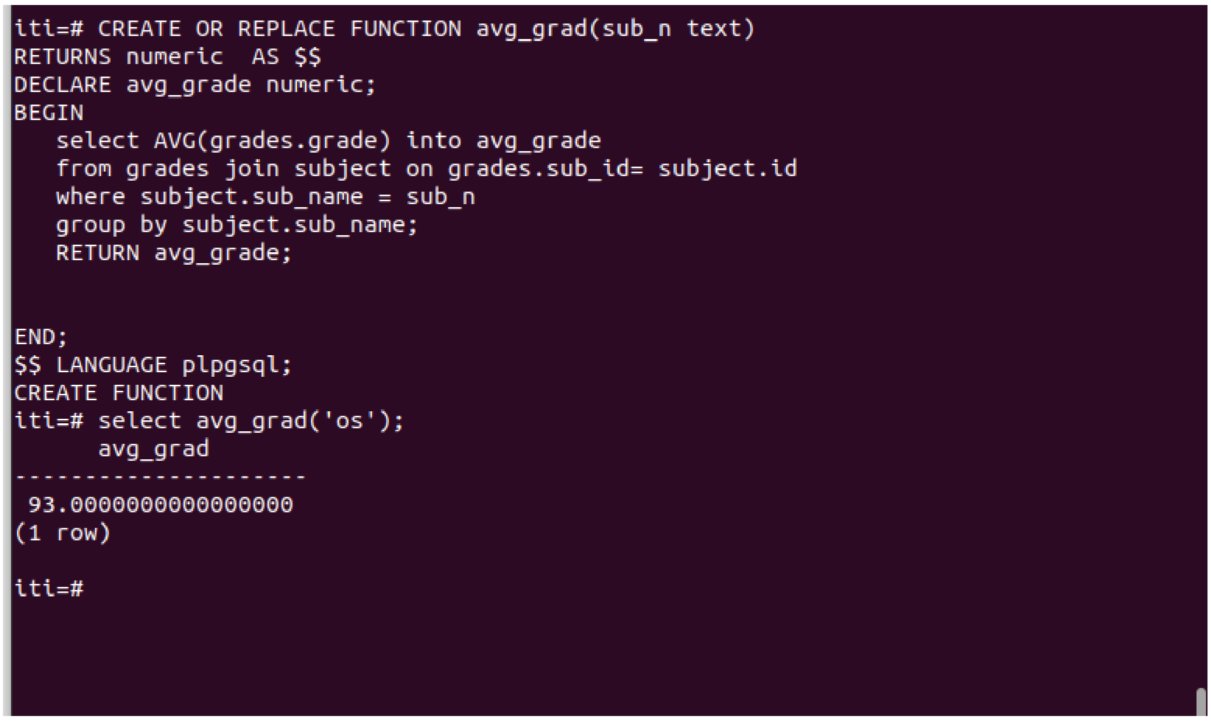
```
iti=# CREATE OR REPLACE FUNCTION oddOReven(x numeric)
RETURNS text AS $$
BEGIN
    if (x % 2 =0) then
        RETURN 'even';
    else
        RETURN 'odd';
    end if;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
iti=# select oddOReven(5);
 oddoreven
-----------
 odd
(1 row)

iti=# select oddOReven(10);
 oddoreven
-----------
 even
(1 row)
```

4. Create a function that takes a Student ID as input and retrieves all information related to that student.



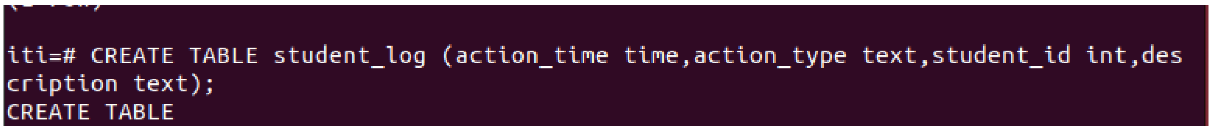
```
DROP FUNCTION
iti=# CREATE OR REPLACE FUNCTION student_data(s_id int)
RETURNS table (id int, e_name text,email varchar(50), address text, track_id int, bi
rth_date date,gender gender_type )   AS $$
BEGIN

   RETURN query
   select * from student where student.id = s_id;

END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
iti=# select student_data(3);
          student_data
--------------------------------
 (3,Sara,sara@iti.com,Alex,1,,)
(1 row)

iti=# select student_data(5);
                student_data
-------------------------------------------
 (5,Ali,ali@iti.com,Aswan,3,2002-04-05,male)
(1 row)
```

5. Implement a function that takes the name of a subject and calculates the average grades for that subject.

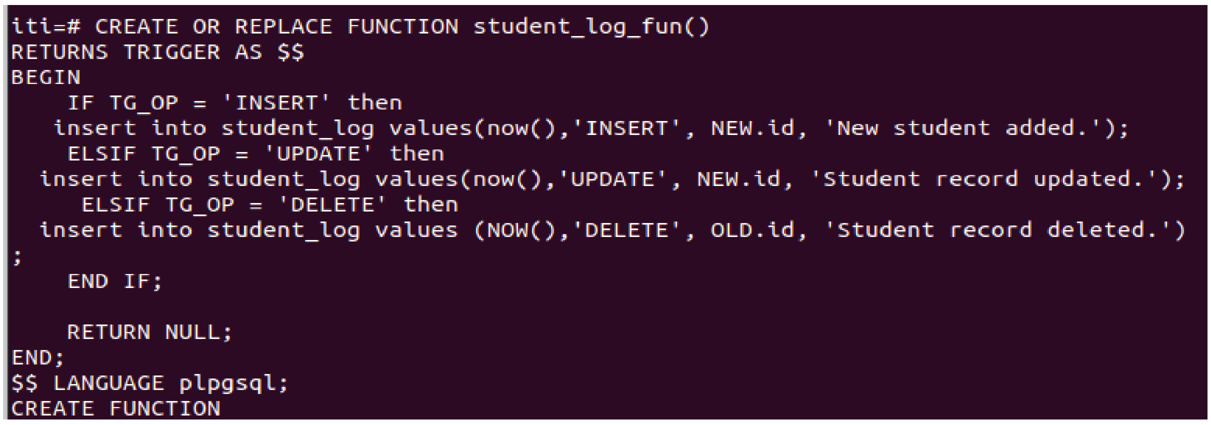


```
iti=# CREATE OR REPLACE FUNCTION avg_grad(sub_n text)
RETURNS numeric  AS $$
DECLARE avg_grade numeric;
BEGIN
   select AVG(grades.grade) into avg_grade
   from grades join subject on grades.sub_id= subject.id
   where subject.sub_name = sub_n
   group by subject.sub_name;
   RETURN avg_grade;


END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
iti=# select avg_grad('os');
      avg_grad
---------------------
 93.0000000000000000
(1 row)

iti=#
```

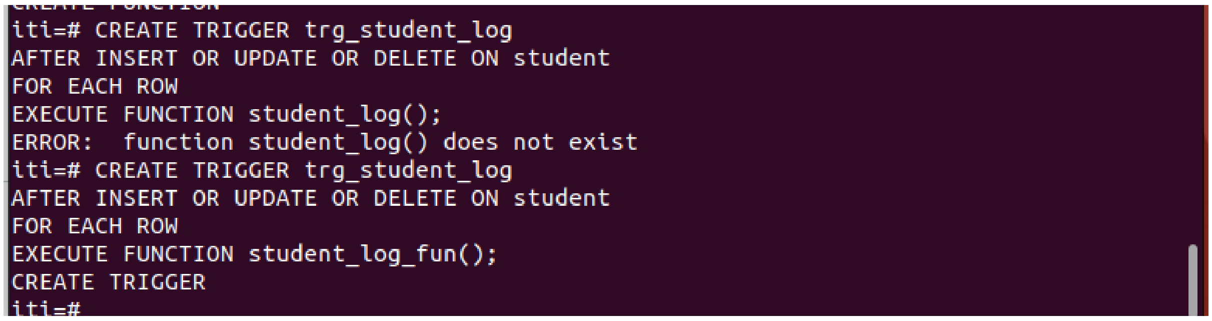
7. Create a trigger to monitor changes made to the student table, including additions, updates, and deletions. This trigger will record the time of each action and provide a description of the action in another table.



```
iti=# CREATE TABLE student_log (action_time time,action_type text,student_id int,des
cription text);
CREATE TABLE
```



```
iti=# CREATE OR REPLACE FUNCTION student_log_fun()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' then
   insert into student_log values(now(),'INSERT', NEW.id, 'New student added.');
    ELSIF TG_OP = 'UPDATE'  then
  insert into student_log values(now(),'UPDATE', NEW.id, 'Student record updated.');
     ELSIF TG_OP = 'DELETE'  then
  insert into student_log values (NOW(),'DELETE', OLD.id, 'Student record deleted.')
;
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
CREATE FUNCTION
```



```
iti=# CREATE TRIGGER trg_student_log
AFTER INSERT OR UPDATE OR DELETE ON student
FOR EACH ROW
EXECUTE FUNCTION student_log();
ERROR:  function student_log() does not exist
iti=# CREATE TRIGGER trg_student_log
AFTER INSERT OR UPDATE OR DELETE ON student
FOR EACH ROW
EXECUTE FUNCTION student_log_fun();
CREATE TRIGGER
iti=#
```