command 👍

1. Create multiply function which take two number and return the multiply of them.

test\_function.sql file contain :

CREATE FUNCTION multiply\_2\_numbers(x int , y int) RETURNS integer AS $$

DECLARE

mul int;

BEGIN

mul = x \* y;

RETURN mul;

END;

$$ LANGUAGE plpgsql;

iti=# \i D:/Learn/ITI/postgresql/test\_function.sql;

iti=# select multiply\_2\_numbers(5,10);

2. Create Hello world function which take username and return welcome message

to user using his name.

test\_function.sql file contain :

CREATE or replace FUNCTION Hello\_world(username text ) RETURNS text AS $$

DECLARE

msg text;

BEGIN

msg = concat('Welcome ' ,username , 'in our sweety website') ;

RETURN msg;

END;

$$ LANGUAGE plpgsql;

iti=# \i D:/Learn/ITI/postgresql/test\_function.sql;

iti=# select Hello\_world('amir');

3. Create function which takes number and return if this number is odd or even.

test\_function.sql file contain :

CREATE or replace FUNCTION check\_number(x int ) RETURNS text AS $$

DECLARE

res text;

BEGIN

if x % 2 = 0 then

res = ' even number' ;

else

res = ' odd number' ;

end if;

RETURN res;

END;

$$ LANGUAGE plpgsql;

iti=# \i D:/Learn/ITI/postgresql/test\_function.sql;

iti=# select check\_number(1);

4. Create AddNewStudent function which take Student firstName and lastname and

birthdate and TrackName and add this new student info at database.

test\_function.sql file contain :

CREATE or replace FUNCTION AddNewStudent(first\_name text,last\_name text,birth\_date date,track\_name text ) RETURNS text AS $$

DECLARE

stid smallint;

t\_id smallint = 1;

track\_name text = track\_name;

gender gender;

BEGIN

SELECT id INTO stid FROM students ORDER BY id DESC LIMIT 1;

stid = stid + 1;

IF track\_name = 'UI&Web' THEN

t\_id = 1;

ELSIF track\_name = 'sd' THEN

t\_id = 2;

ELSIF track\_name = 'data science' THEN

t\_id = 3;

ELSIF track\_name = 'Mobile' THEN

t\_id = 4;

ELSIF track\_name = ' AI' THEN

t\_id = 5;

else

t\_id = 1;

END IF;

INSERT INTO students(id,trackid,first\_name,last\_name,birth\_date) VALUES(stid,t\_id,first\_name,last\_name,birth\_date);

return ' created successfully';

END;

$$ LANGUAGE plpgsql;

iti=# \i D:/Learn/ITI/postgresql/test\_function.sql;

iti=# select AddNewStudent('Sanaa','Abdelhamid','1-12-2000','sd');

5. Create function which takes StudentId and return the string/text that describe the

use info(firstname, last name, age, TrackName).

test\_function.sql file contain :

CREATE or replace FUNCTION display\_info(stdid int) RETURNS text AS $$

DECLARE

track\_name text ;

newdata record;

res text;

BEGIN

SELECT \*,id into newdata FROM students where id = stdid ;

IF newdata.trackid = 1 THEN

track\_name = 'UI&Web';

ELSIF newdata.trackid = 2 THEN

track\_name = 'sd';

ELSIF newdata.trackid = 3 THEN

track\_name = 'data science';

ELSIF newdata.trackid = 4 THEN

track\_name = 'Mobile';

ELSIF newdata.trackid = 5 THEN

track\_name = ' AI';

else

track\_name = 'not found';

END IF;

res = concat('student name is : ',newdata.first\_name , ' ', newdata.last\_name, 'and his/her age is : ', age(newdata.birth\_date), ' and his/her track is : ', track\_name );

return res;

END;

$$ LANGUAGE plpgsql;

iti=# \i D:/Learn/ITI/postgresql/test\_function.sql;

iti=# selectdisplay\_info(1);

6. Create function which takes TrackName and return the students names in this

track.

test\_function.sql file contain :

CREATE or replace FUNCTION display\_students\_track(track\_name text) RETURNS TABLE(fullname text) AS $$

DECLARE

t\_id smallint = 1;

track\_name text = track\_name;

BEGIN

IF track\_name = 'UI&Web' THEN

t\_id = 1;

ELSIF track\_name = 'sd' THEN

t\_id = 2;

ELSIF track\_name = 'data science' THEN

t\_id = 3;

ELSIF track\_name = 'Mobile' THEN

t\_id = 4;

ELSIF track\_name = ' AI' THEN

t\_id = 5;

else

t\_id = 1;

END IF;

RETURN QUERY SELECT concat(first\_name, ' ', last\_name) fullname FROM students where trackid = t\_id;

END;

$$ LANGUAGE plpgsql;

iti=# \i D:/Learn/ITI/postgresql/test\_function.sql;

iti=# select display\_students\_track('sd');

7. Create function which takes student id and subject id and return score the

student in subject.

test\_function.sql file contain :

CREATE or replace FUNCTION display\_std\_score(stid int,crid int) RETURNS TABLE(score\_std int) AS $$

DECLARE

stdid int = stid;

crid int = crid;

t\_id smallint = 1;

BEGIN

return query select ex.score from courses c inner join studentcourseexam ex on ex.crsid = c.id and c.id = crid inner join students s on ex.stdid = s.id and s.id = stid;

END;

$$ LANGUAGE plpgsql;

iti=# \i D:/Learn/ITI/postgresql/test\_function.sql;

iti=# select display\_std\_score(1,1);

8. Create function which takes subject id and return the number of students who

failed in a subject (Score less than 50).

test\_function.sql file contain :

CREATE or replace FUNCTION display\_failedstd\_count(subj\_id int) RETURNS int AS $$

DECLARE

subj\_id int = subj\_id;

failed\_std int = 0;

scores int;

BEGIN

select count(ex.score) into scores from studentcourseexam ex where subj\_id = ex.crsid and ex.score < 50;

return scores;

END;

$$ LANGUAGE plpgsql;

iti=# \i D:/Learn/ITI/postgresql/test\_function.sql;

iti=# select display\_failedstd\_count(1);

9. Create function which take subject name and return the average grades for

subject

iti=# \i D:/Learn/ITI/postgresql/test\_function.sql;

test\_function.sql file contain :

CREATE or replace FUNCTION display\_averageSubj\_scores(subj\_name text) RETURNS int AS $$

DECLARE

subj\_name text = subj\_name;

failed\_std int = 0;

scores int;

BEGIN

select avg(ex.score) into scores from courses c inner join studentcourseexam ex on ex.crsid = c.id and c.name = subj\_name;

return scores;

END;

$$ LANGUAGE plpgsql;

iti=# \i D:/Learn/ITI/postgresql/test\_function.sql;

iti=# select display\_averageSubj\_scores('c');

10. Import SQL file into your database.

– all previous questions solved with import function file 😀 ;

11. Create Table called Deleted\_Students which will hold the deleted

students info (same columns as in student tables).

iti=# create table delted\_students as table students with no data;

12. Create trigger to save the deleted student from Student table to

Deleted\_Students.

CREATE or replace FUNCTION triger\_deleted\_students() RETURNS trigger AS $$

DECLARE

BEGIN

insert into delted\_students values(old.id,old.trackid,old.gender,old.birth\_date,old.first\_name,old.last\_name);

return old;

END;

$$ LANGUAGE plpgsql;

create trigger triger\_deleted before delete on students for each row execute procedure triger\_deleted\_students();

13. Try to delete student from students table and check the

Deleted\_Students if it contain the deleted students or not.

iti=# delete from students where id = 8;

iti=# select \* from delted\_students;

id | trackid | gender | birth\_date | first\_name | last\_name

----+---------+--------+------------+------------+------------

8 | 2 | | 2000-12-01 | Sanaa | Abdelhamid

(1 row)

4. Create trigger to prevent insert new Course with name length greater

than 20 chars;

iti=# DROP trigger triger\_deleted on students;

CREATE or replace FUNCTION triger\_deleted\_students() RETURNS trigger AS $$

DECLARE

BEGIN

if (length(new.name) > 20) then

return null;

else

return new;

end if;

END;

$$ LANGUAGE plpgsql;

create trigger triger\_deleted before insert on courses for each row execute procedure triger\_deleted\_students();

5. Create trigger to prevent update student names.

CREATE or replace FUNCTION triger\_update\_students() RETURNS trigger AS $$

DECLARE

BEGIN

return null;

END;

$$ LANGUAGE plpgsql;

create trigger triger\_deleted before update on students for each row execute procedure triger\_deleted\_students();

6. Create trigger to prevent update grades of students.

7. Create trigger to prevent user to insert or update Exam with Score

greater than 100 or less than zero.

8. (bonus) Create trigger to prevent any user to update/insert/delete to all

tables (Students, Exams, Tracks,..) after 7:00 PM.

19. Backup your Database to external file.

iti=# pg-dump iti;

20.Backup your Student table to external file

iti=# copy students to ‘D:/Learn/ITI/postgresql/students.sql’’;