Python 22-23 / Minia / postgreSql
lab5 for day3-Hager haron

1. Create Table called Deleted_Students which will hold the deleted students info (same columns as in student tables).

create table deleted_student (id int , name text , email text ,adress text , track_name text, birthdate date);

2. Create trigger to save the deleted student from Student table to Deleted_Students.

```
CREATE OR REPLACE FUNCTION ModifyOnDelete() RETURNS TRIGGER AS $$

BEGIN
	insert into deleted_student
values(old.id,old.name,old.email,old.adress,old.tracke_name,old.birthdate);
	return NEW;
END;
$$ LANGUAGE plpgsql;
```

3. Try to delete student from students table and check the Deleted_Students if it contain the deleted students or not.

```
select * from student ;
select * from deleted_student ;
delete from student where id = 2 ;
select * from student ;
select * from deleted_student ;
```

4. Create trigger to prevent insert new Course with name length greater than 20 chars;

```
CREATE OR REPLACE FUNCTION CheckNameLength() RETURNS TRIGGER AS $$

BEGIN

	IF LENGTH(NEW.name) > 20
	 THEN
	   RETURN NULL;
	   else
		return NEW;
	   END IF;

END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER CheckName before insert on courses
for each row
execute procedure CheckNameLength();
```

5. **Create trigger to prevent update student names.**

```
CREATE OR REPLACE FUNCTION prevent_update() RETURNS TRIGGER AS $$

BEGIN

      IF OLD.name != NEW.name
       THEN
              RETURN NULL;
        else
              return NEW;
        END IF;

END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER no_update_name before update
ON students FOR EACH ROW EXECUTE
PROCEDURE prevent_update();
```

6. **Create trigger to prevent update grades of students.?**

```
CREATE OR REPLACE FUNCTION  prevent_std_grade_update() RETURNS TRIGGER AS $
$
BEGIN
IF old.grad_std != NEW.grad_std
      THEN
              RETURN NULL;
        else
              return NEW;
        END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER prevent_student_grade_update
BEFORE UPDATE  grad_std ON exam
FOR EACH ROW EXECUTE FUNCTION prevent_std_grade_update();
```

7. Create trigger to prevent user to insert or update Exam with Score greater than 100 or less than zero.

```
CREATE OR REPLACE FUNCTION prevent_grade() RETURNS TRIGGER AS $$

BEGIN

     IF (NEW. grad_std > 100 OR NEW. grad_std < 0)
      THEN
              RETURN NULL;
        else
              return NEW;
        END IF;
```

```
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER prevent_update_grade before update
ON exam FOR EACH ROW EXECUTE
PROCEDURE prevent_grade();

CREATE TRIGGER prevent_insert_grade before insert
ON exam FOR EACH ROW EXECUTE
PROCEDURE prevent_grade();
```

8. (bonus) Create trigger to prevent any user to update/insert/delete to all tables (Students, Exams, Tracks,..) after 7:00 PM

```
create or replace function preventEditTimeOut()
returns trigger as $$
declare
curtime time;
BEGIN
curtime := CURRENT_TIME;
IF curtime >= '19:00:00' then
raise exception 'Time off: you cann’t modify after 7:00 ';
END if ;
return null;
END;
$$ LANGUAGE plpgsql;

create trigger prevent_edit
after insert or update or delete on student
for each statement execute function  preventEditTimeOut()

create trigger prevent_edit
after insert or update or delete on exam
for each statement execute function preventEditTimeOut()

create trigger prevent_edit
after insert or update or delete on tracks
for each statement execute function preventEditTimeOut()
```

9. Backup your Database to external file

```
pg_dump itilab > itilab_Backup.sql
```

10. Backup your Student table to external file

```
pg_dump -d <itilab> -t <student> > itilabTB_student.sql
```