← **Project review - SQL1 Bootcamp. Day09**

▤ Type of project		Individual
🕐 Duration		30 min
☑ Passed Peer Reviews		1/3

Git project ⌃

ssh://git@repos-ssh.21-school.ru:2289/students/SQL_beginner._Day09.ID_574108/lshe...

Copy link   **Open**

Student ⌃

👤 **lsherry@student.21-school.ru**   level 8

About ⌃

Introduction

The methodology of School 21 makes sense only if peer-to-peer reviews are done seriously.
Please read all guidelines carefully before starting the review.
- Please, stay courteous, polite, respectful and constructive in all communications during this review.

- Highlight possible malfunctions of the work done by the person and take the time to discuss and debate it.
- Keep in mind that sometimes there can be differences in interpretation of the tasks and the scope of features. Please, stay open-minded to the vision of the other.
- If you have not finished the project yet, it is compulsory to read the entire instruction before starting the review.

Guidelines

- Evaluate only the files that are in src folder on the GIT repository of the student or group.
- Ensure to start reviewing a group project only when the team is present in full.
- Use special flags in the checklist to report, for example, an "empty work" if repository does not contain the work of the student (or group) in the src folder of the develop branch, or "cheat" in case of cheating or if the student (or group) are unable to explain their work at any time during review as well as if one of the points below is not met. However, except for cheating cases, you are encouraged to continue reviewing the project to identify the problems that caused the situation in order to avoid them at the next review.
- Doublecheck that the GIT repository is the one corresponding to the student or the group.
- Meticulously check that nothing malicious has been used to mislead you.
- In controversial cases, remember that the checklist determines only the general order of the check. The final decision on project evaluation remains with the reviewer.

Main part   ⌃

Exercise 00

Checks for the file day09_ex00.sql
- The SQL script looks like below.

```
create table person_audit
(
created timestamp with time zone not null default current_timestamp,
type_event char(1) not null default 'I',
row_id     bigint not null ,
name    varchar ,
age     integer ,
gender  varchar ,
address varchar,
constraint ch_type_event check ( type_event in ('I','U', 'D') )
);
```

- The SQL for trigger function

```
CREATE OR REPLACE FUNCTION fnc_trg_person_insert_audit( )
RETURNS TRIGGER AS
$BODY$
BEGIN
```

```
    INSERT INTO person_audit(created, type_event, row_id, name, age, gender, address)
    VALUES(current_timestamp,'I', NEW.id, NEW.name, NEW.age, NEW.gender, NEW.addres
s);
    RETURN NULL;
    END;
    $BODY$
    LANGUAGE plpgsql;
```

- The SQL for trigger

```
    CREATE  TRIGGER trg_person_insert_audit
    AFTER    INSERT ON person FOR EACH ROW
    EXECUTE FUNCTION  fnc_trg_person_insert_audit( );
```

- SQL to check

```
    select * from person_audit
```

- result of SQL

```
    "2022-03-21 20:27:40.283551 +00:00"  "I"   "10"        "Damir" "22"        "male"    "Irk
utsk"
```

No   ✓ Yes

Exercise 01

Checks for the file day09_ex01.sql
- The SQL for trigger function

```
    CREATE OR REPLACE FUNCTION fnc_trg_person_update_audit( )
    RETURNS TRIGGER AS
    $BODY$
    BEGIN
    INSERT INTO person_audit(created, type_event, row_id, name, age, gender, address)
    VALUES(current_timestamp,'U', OLD.id, OLD.name, OLD.age, OLD.gender, OLD.addres
s);
    RETURN NULL;
    END;
    $BODY$
    LANGUAGE plpgsql;
```

- The SQL for trigger

```
    CREATE  TRIGGER trg_person_update_audit
    AFTER    UPDATE ON person FOR EACH ROW
    EXECUTE FUNCTION  fnc_trg_person_update_audit( );
```

- SQL to check

```
    select * from person_audit
```

- result of SQL

```
    "2022-03-21 20:27:40.283551 +00:00"  "I"  "10"      "Damir" "22"      "male"  "Irk
utsk"
    "2022-03-21 20:29:38.834865 +00:00"  "U"  "10"      "Damir" "22"      "male"  "Irk
utsk"
    "2022-03-21 20:29:39.333943 +00:00"  "U"  "10"      "Bulat" "22"      "male"  "Irk
utsk"
```

No	✓ Yes

Exercise 02

Checks for the file day09_ex02.sql
- The SQL for trigger function

```
    CREATE OR REPLACE FUNCTION fnc_trg_person_delete_audit( )
    RETURNS TRIGGER AS
    $BODY$
    BEGIN
    INSERT INTO person_audit(created, type_event, row_id, name, age, gender, address)
    VALUES(current_timestamp,'D', OLD.id, OLD.name, OLD.age, OLD.gender, OLD.addres
s);
    RETURN NULL;
    END;
    $BODY$
    LANGUAGE plpgsql;
```

- The SQL for trigger

```
    CREATE  TRIGGER trg_person_delete_audit
    AFTER   DELETE ON person FOR EACH ROW
    EXECUTE FUNCTION  fnc_trg_person_delete_audit( );
```

- SQL to check

```
    select * from person_audit order by created;
```

- result of SQL

```
    "2022-03-21 20:27:40.283551 +00:00"  "I"  "10"      "Damir" "22"      "male"  "Irk
utsk"
    "2022-03-21 20:29:38.834865 +00:00"  "U"  "10"      "Damir" "22"      "male"  "Irk
utsk"
    "2022-03-21 20:29:39.333943 +00:00"  "U"  "10"      "Bulat" "22"      "male"  "Irk
utsk"
    "2022-03-21 20:30:57.731367 +00:00"  "D"  "10"      "Damir" "22"      "male"  "Irk
utsk"
```

No	✓ Yes

Exercise 03

Checks for the file day09_ex03.sql
- The SQL script looks like below.

```
DROP trigger trg_person_delete_audit on person;
DROP trigger trg_person_update_audit on person;
DROP trigger trg_person_insert_audit on person;

drop function fnc_trg_person_delete_audit();
drop function fnc_trg_person_update_audit();
drop function fnc_trg_person_insert_audit();

truncate person_audit;
```

- The SQL for trigger function

```
CREATE OR REPLACE FUNCTION fnc_trg_person_audit( )
RETURNS TRIGGER AS
$BODY$
BEGIN
IF (TG_OP = 'INSERT') THEN
    INSERT INTO person_audit(created, type_event, row_id, name, age, gender, address)
    VALUES(current_timestamp,'I', NEW.id, NEW.name, NEW.age, NEW.gender, NEW.address);
ELSEIF (TG_OP = 'UPDATE') THEN
    INSERT INTO person_audit(created, type_event, row_id, name, age, gender, address)
    VALUES(current_timestamp,'U', OLD.id, OLD.name, OLD.age, OLD.gender, OLD.address);
ELSE
    INSERT INTO person_audit(created, type_event, row_id, name, age, gender, address)
    VALUES(current_timestamp,'D', OLD.id, OLD.name, OLD.age, OLD.gender, OLD.address);
END IF;

RETURN NULL;
END;
$BODY$
LANGUAGE plpgsql;
```

- The SQL for trigger

```
CREATE  TRIGGER trg_person_audit
AFTER   DELETE OR UPDATE OR INSERT ON person FOR EACH ROW
EXECUTE FUNCTION  fnc_trg_person_audit( );
```

- SQL to check

```
select * from person_audit order by created;
```

- result of SQL

```
"2022-03-21 20:33:30.826364 +00:00"  "I"   "10"      "Damir" "22"      "male"   "Irk
utsk"
```

```
    "2022-03-21 20:33:31.282922 +00:00"  "U" "10"      "Damir" "22"      "male"    "Irk
utsk"
    "2022-03-21 20:33:31.746362 +00:00"  "U" "10"      "Bulat" "22"      "male"    "Irk
utsk"
    "2022-03-21 20:33:32.228181 +00:00"  "D" "10"      "Damir" "22"      "male"    "Irk
utsk"
```

No	✓ Yes

Exercise 04

Checks for the file day09_ex04.sql
- The SQL script looks like below.

```
    CREATE FUNCTION fnc_persons_female( )
    RETURNS TABLE(id    person.id%TYPE,
          name    person.name%TYPE,
          age    person.age%TYPE,
          gender    person.gender%TYPE,
          address    person.address%TYPE) AS $$
    SELECT id, name, age, gender, address
       FROM person
    WHERE gender= 'female';
    $$ LANGUAGE SQL;

    CREATE FUNCTION fnc_persons_male( )
    RETURNS TABLE(id    person.id%TYPE,
          name    person.name%TYPE,
          age    person.age%TYPE,
          gender    person.gender%TYPE,
          address    person.address%TYPE) AS $$
    SELECT id, name, age, gender, address
       FROM person
    WHERE gender= 'male';
    $$ LANGUAGE SQL;
```

- The SQL to check

```
    select * from fnc_persons_male();
```

- The result of SQL

```
    "2"    "Andrey" "21"      "male"    "Moscow"
    "4"    "Denis" "13"       "male"    "Kazan"
    "7"    "Peter" "24"       "male"   "Saint-Petersburg"
    "9"    "Dmitriy"     "18"      "male"   "Samara"
```

- The SQL to check

```
    select * from fnc_persons_female();
```

- The result of SQL

```
"1"    "Anna"    "16"        "female" "Moscow"
"3"    "Kate"    "33"        "female" "Kazan"
"5"    "Elvira"  "45"        "female" "Kazan"
"6"    "Irina"   "21"        "female" "Saint-Petersburg"
"8"    "Nataly" "30"         "female" "Novosibirsk"
```

No	✓ Yes

Exercise 05

Checks for the file day09_ex05.sql
- The SQL script looks like below.

```
CREATE OR REPLACE FUNCTION fnc_persons(pgender varchar default 'female')
RETURNS TABLE(id    person.id%TYPE,
        name    person.name%TYPE,
        age    person.age%TYPE,
        gender    person.gender%TYPE,
        address    person.address%TYPE) AS $$
SELECT id, name, age, gender, address
    FROM person
WHERE gender= pgender;
$$ LANGUAGE SQL;
```

- The SQL to check

```
select * from fnc_persons(pgender := 'male');
```

- The result of SQL

```
"2"    "Andrey" "21"        "male"    "Moscow"
"4"    "Denis"  "13"        "male"    "Kazan"
"7"    "Peter"  "24"        "male"    "Saint-Petersburg"
"9"    "Dmitriy"      "18"        "male"    "Samara"
```

- The SQL to check

```
select * from fnc_persons();
```

- The result of SQL

```
"1"    "Anna"    "16"        "female" "Moscow"
"3"    "Kate"    "33"        "female" "Kazan"
"5"    "Elvira"  "45"        "female" "Kazan"
"6"    "Irina"   "21"        "female" "Saint-Petersburg"
"8"    "Nataly" "30"         "female" "Novosibirsk"
```

No	✓ Yes

Exercise 06

Checks for the file day09_ex06.sql
- The SQL script looks like below.

```
CREATE FUNCTION fnc_person_visits_and_eats_on_date(pperson varchar default 'Dmitriy',
pprice numeric default 500, pdate date default '2022-01-08')
RETURNS TABLE(name    varchar) AS $$
BEGIN
RETURN QUERY
    select p.name as pizzeria_name
    from menu inner join pizzeria p on p.id = menu.pizzeria_id
        inner join person_visits pv on menu.pizzeria_id = pv.pizzeria_id
        inner join person p2 on p2.id = pv.person_id
    where price  < pprice and p2.name = pperson and visit_date = pdate;
END;
$$ LANGUAGE PLPGSQL;
```

- The SQL to check

```
select * from fnc_person_visits_and_eats_on_date(pprice := 800);
```

- The result of SQL

```
"Papa Johns"
"DoDo Pizza"
```

- The SQL to check

```
select *
from fnc_person_visits_and_eats_on_date(pperson := 'Anna',pprice := 1300,pdate := '2022-01-01');
```

- The result of SQL

```
"Pizza Hut"
"Pizza Hut"
"Pizza Hut"
"Pizza Hut"
```

No	✓ Yes

Exercise 07

Checks for the file day09_ex07.sql
- The SQL script looks like below.

```
CREATE FUNCTION func_minimum(VARIADIC arr NUMERIC[ ])
RETURNS NUMERIC AS
$$
```

```
SELECT min( i ) FROM unnest(arr) g( i );
$$ LANGUAGE SQL;
```

- The SQL to check

```
SELECT func_minimum(VARIADIC arr => ARRAY[10.0, -1.0, 5.0, 4.4]);
```

- The result of SQL

```
"-1"
```

No   ✓ Yes

Exercise 08

Checks for the file day09_ex08.sql
- The SQL script looks like below.

```
CREATE OR REPLACE FUNCTION fnc_fibonacci(pstop integer default 10)
RETURNS TABLE(a    bigint) AS $$
WITH RECURSIVE f (a,b) AS
(SELECT 0 AS a, 1 AS b
UNION ALL
SELECT b, a+b FROM f WHERE b<pstop)
SELECT a
FROM f;
$$ LANGUAGE SQL;
```

- The SQL to check

```
select * from fnc_fibonacci(20)
```

- The result of SQL

```
"0"
"1"
"1"
"2"
"3"
"5"
"8"
"13"
```

No   ✓ Yes

Feedback   ︿

Fails ⓘ

| Forbidden functions | Empty work | Cheat |

Comment

Leave a comment...

✓ Review