




## ← Project review - SQL beginner. Team00

 Type of project	Group
 Duration	30 min
 Passed Peer Reviews	0/2

### Git project



ssh://git@repos-ssh.21-school.ru:2289/students/SQL\_beginner.\_Team00.ID\_574110/Team\_8...

Copy link

Open

### Team 8 (TL: mmargene-sql-tester)

mmargene-sql-tester lvl 0



mmargene-sql-tester Leader

level 0



kalexand-sql-tester

level 1



## 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 exercise 00

- The next command provides a DDL for table creation and further INSERTs

```
create table nodes
( point1 varchar,
  point2 varchar,
  cost numeric);
```

```
insert into nodes values ('a','b',10);
```

```
insert into nodes values ('b','a',10);
```

```
insert into nodes values ('b','c',35);
```

```
insert into nodes values ('c','b',35);
```

```
insert into nodes values ('c','a',15);
```

```
insert into nodes values ('a','c',15);
```

```
insert into nodes values ('c','d',30);
```

```
insert into nodes values ('d','c',30);
```

```
insert into nodes values ('a','d',20);
```

```
insert into nodes values ('d','a',20);
```

```
insert into nodes values ('b','d',25);
```

```
insert into nodes values ('d','b',25);
```

- The next command solves classical TSP

```
with t as (  
  with recursive _n as  
  (select point1,  
          point2,  
          cost,  
          1 as level,  
          array[point1] AS path,  
          FALSE AS cycle,  
          array[cost] AS costs  
  from nodes  
  where point1 = 'a'  
  union all  
  select nodes.point1,  
         nodes.point2,  
         nodes.cost+_n.cost as cost,  
         _n.level+1 as level,  
         _n.path || nodes.point1 AS path,  
         nodes.point1 = ANY (_n.path) AS cycle,  
         _n.costs || nodes.cost AS costs  
  from nodes inner join _n on _n.point2 = nodes.point1 and not cycle  
  )  
  select  
    cost - costs[5] as total_cost,  
    path as tour  
  from _n  
  where level =5 and  
    'a' = ANY(path) and  
    'b' = ANY(path) and
```

```

    'c' = ANY(path) and
    'd' = ANY(path)
    and path[1] = path[5]
order by cost, path)
select distinct *
from t
where total_cost = (select min(total_cost) from t)
order by 1,2;

```

- The result is below

```

"80"      "{a,b,d,c,a}"
"80"      "{a,c,d,b,a}"

```

☒ No

☐ Yes

## Exercise 01

Checks for the exercise 01

- The next command

```

with t as (
with recursive _n as
(select point1,
    point2,
    cost,
    1 as level,
    array[point1] AS path,
    FALSE AS cycle,
    array[cost] AS costs
from nodes
where point1 = 'a'
union all
select nodes.point1,
    nodes.point2,
    nodes.cost+_n.cost as cost,
    _n.level+1 as level,
    _n.path || nodes.point1 AS path,
    nodes.point1 = ANY (_n.path) AS cycle,
    _n.costs || nodes.cost AS costs
from nodes inner join _n on _n.point2 = nodes.point1 and not cycle
)
select
    cost - costs[5] as total_cost,
    path as tour
from _n
where level =5 and

```

```
'a' = ANY(path) and  
'b' = ANY(path) and  
'c' = ANY(path) and  
'd' = ANY(path)  
and path[1] = path[5]  
order by cost, path)  
select distinct *  
from t  
where total_cost = (select min(total_cost) from t) or  
total_cost = (select max(total_cost) from t)  
order by 1,2;
```

- The result is below

"80"	"{a,b,d,c,a}"
"80"	"{a,c,d,b,a}"
"95"	"{a,b,c,d,a}"
"95"	"{a,c,b,d,a}"
"95"	"{a,d,b,c,a}"
"95"	"{a,d,c,b,a}"

## Feedback



Fails 

## Comment

Leave a comment...

✓ Review