

# SCALE FOR PROJECT

## SQL / DAY 00

---

### Introduction

The methodology of School 21 makes sense only if peer-to-peer assessments are done seriously. This document will help you to do it properly.

- Please, stay courteous, polite, respectful and constructive in all communications during this assessment. The bond of trust between community 21 and you depends on it.
- 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.

### Guidelines

- Evaluate only the files that are on the GIT repository of the student or group.
- Doublecheck that the GIT repository is the one corresponding to the student or the group as long as to the project.
- Meticulously check that nothing malicious has been used to mislead you and have you assess something except the content of the official repository.
- If you have not finished the project yet, it is compulsory to read the entire instruction before starting the review.
- Use the special flags in the scale to report an empty or non-functional solution as long as a case of cheating. In these cases, the assessment is completed and the final grade is 0 (or in a case of cheating is -42). However, except for a case of cheating, you are encouraged to continue reviewing the project to identify the problems that caused the situation in order to avoid them for the next assessment.
- You must stop giving points from the first wrong exercise even if the following exercises are correct.

### Attachments

- [The exercises](#)

# Preliminaries

Respect the rules:

- The repository contains the work of the student (or group).
- The student is able to explain their work at any time during the assessment.
- The general rules are respected throughout the assessment.

Yes | No

## Exercise 00 - Database Model Creation

1. The “flyway.conf” file (located in “~/flyway-6.x.x/conf” directory) must have a manually modified options below (other parameters are in the default state):
  - flyway.url = jdbc:postgresql://hostname:5432/data  
(where *hostname* is IP-address, DNS name or hostname)
  - flyway.user = data
  - flyway.password = data
  - flyway.schemas = data
  - flyway.defaultSchema = data
2. The command “flyway info” (through the command line) returns “state = success” for all SQL scripts with right names from exercise.
3. Checks for the file **V000\_\_db\_model\_initiate.sql**
  - The common pattern of SQL script for the dictionary table “*indicator*” looks like below.

```
create table indicator
(
    id bigint NOT NULL,
    name varchar NOT NULL ,
    unit varchar NOT NULL ,
    CONSTRAINT pk_indicator PRIMARY KEY (id),
    CONSTRAINT uk_indicator_name UNIQUE (name)
);
```

- There is a comment for a table like below  
`comment on table indicator is 'Comment Text';`
- There are a comments for an each column's table like below

```
comment on column indicator.id is 'Comment Text';

comment on column indicator.name is 'Comment Text';

comment on column indicator.unit is 'Comment Text';
```

- The common pattern of SQL script for the dictionary table “country” looks like below.

```
create table country
(
    id bigint NOT NULL,
    name varchar NOT NULL ,
    par_id bigint ,
    object_type varchar NOT NULL DEFAULT 'country' ,
    CONSTRAINT pk_country PRIMARY KEY (id),
    CONSTRAINT uk_country_name_object_type UNIQUE (name, object_type),
    CONSTRAINT fk_par_id_country FOREIGN KEY (par_id) REFERENCES country(id)
);
```

- There is a comment for a table like below
- There are a comments for an each column’s table like below

```
comment on table country is 'Comment Text';

comment on column country.id is 'Comment Text';

comment on column country.name is 'Comment Text';

comment on column country.par_id is 'Comment Text';

comment on column country.object_type is 'Comment Text';
```

- The common pattern of SQL script for the operational table “country\_indicator” looks like below.

```
create table country_indicator
(
    id bigint NOT NULL,
    c_id bigint NOT NULL ,
    i_id bigint NOT NULL ,
    value numeric NOT NULL,
    actual_date timestamp NOT NULL ,
    CONSTRAINT pk_country_indicator PRIMARY KEY (id),
    CONSTRAINT uk_country_indicator_c_id_i_id_actual_date UNIQUE (c_id,i_id,actual_date),
    CONSTRAINT fk_c_id_country FOREIGN KEY (c_id) REFERENCES country(id),
    CONSTRAINT fk_i_id_indicator FOREIGN KEY (i_id) REFERENCES indicator(id)
);
```

- There is a comment for a table like below

```
comment on table country_indicator is 'Comment Text';
```

- There are a comments for an each column's table like below

```
comment on column country_indicator.id is 'Comment Text';
```

```
comment on column country_indicator.c_id is 'Comment Text';
```

```
comment on column country_indicator.i_id is 'Comment Text';
```

```
comment on column country_indicator.value is 'Comment Text';
```

```
comment on column country_indicator.actual_date is 'Comment Text';
```

#### 4. Checks for the file **V001\_\_dictionaries\_initiate.sql**

- The query below should return 3 rows

(use appropriate student's table name)

```
select *
from indicator
```

	id	name	unit
1	1	Population of country	human
2	2	Unemployment rate	percent
3	3	Infected humans COVID-19	human

- The query below should return 7 continents with value of column *par\_id* = null (use appropriate student's table name)

```
select *
from country
where object_type = 'continent'
```

	id	name	par_id	object_type
1	1	Europe	<null>	continent
2	2	Africa	<null>	continent
3	3	Antarctica	<null>	continent
4	4	Asia	<null>	continent
5	5	North America	<null>	continent
6	6	Australia	<null>	continent
7	7	South America	<null>	continent

- The query below should return 195 countries with not null value of *par\_id* column (use appropriate student's table name)

```
select count(*)
from country
where object_type = 'country'
```

Yes | No

## Exercise 01 - Data Generator for Model

1. The “flyway.conf” file (located in “~/flyway-6.x.x/conf” directory) must have a manually modified options below (other parameters are in the default state):

- flyway.url = jdbc:postgresql://hostname:5432/data  
(where *hostname* is IP-address, DNS name or hostname)
  - flyway.user = data
  - flyway.password = data
  - flyway.schemas = data
  - flyway.defaultSchema = data
2. The command “flyway info” (through the command line) returns “state = success” for all SQL scripts with right names from exercise.
  3. Students can generate data in different ways (for example generate native INSERT statements by Python or using other skills and online tools). One of them is to use the *generate\_series* function in the database directly.

The example how to generate data for each indicator is presented below

```
insert into data.country_indicator(id,c_id, i_id, value, actual_date)
select row_number() over (),
       id,
       (select id from indicator where name = 'Population of country'),
       (random()*1000000)::integer as value, ('01.' || lpad(i::varchar,2,'0') || '.2019
00:00:00')::timestamp
from data.country,
     generate_series(1,12) as k(i)
where object_type = 'country';

insert into data.country_indicator(id,c_id, i_id, value, actual_date)
select (row_number() over ()) + 2340,
       id,
       (select id from indicator where name = 'Unemployment rate') ,
       (random()*100)::integer as value, ('01.' || lpad(i::varchar,2,'0') || '.2019 00:00:00')::timestamp
from data.country,
     generate_series(1,12) as k(i)
where object_type = 'country';
```

```
insert into data.country_indicator(id,c_id, i_id, value, actual_date)
select (row_number() over ()) + 4680,
       id,
       (select id from indicator where name = 'Infected humans COVID-19') ,
       (random()*50)::integer as value,
```

```

i
from data.country,
    generate_series('2020-05-01', '2020-08-31', '1 day'::interval)as k(i)
where object_type = 'country';

```

4. The query should return the picture below (use appropriate student's table name)

```

select i.name, count(ci.c_id)
from country_indicator ci
    inner join indicator i on i.id = ci.i_id
group by i.name
order by name;

```

	name	count
1	Infected humans COVID-19	23985
2	Population of country	2340
3	Unemployment rate	2340

Yes | No

## Exercise 02 - Define Sequences

- The “flyway.conf” file (located in “~/flyway-6.x.x/conf” directory) must have a manually modified options below (other parameters are in the default state):
  - flyway.url = jdbc:postgresql://hostname:5432/data  
(where *hostname* is IP-address, DNS name or hostname)
  - flyway.user = data
  - flyway.password = data
  - flyway.schemas = data
  - flyway.defaultSchema = data
- The command “flyway info” (through the command line) returns “state = success” for all SQL scripts with right names from exercise.
- Checks for the file **V020\_\_create\_sequences.sql**
  - create all needed database sequences by commands with right naming pattern  
(use appropriate student's table and sequences names)
 

```

create sequence seq_indicator increment by 10 owned by indicator.id;
create sequence seq_country increment by 10 owned by country.id;
create sequence seq_country_indicator increment by 10 owned by country_indicator.id;

```

That's completely OK if student make a next pattern (creation + alteration):

```
create sequence seq_indicator increment by 10;
alter sequence seq_indicator owned by indicator.id;
```

- set default values for all ID columns in all database tables

(use appropriate student's table and sequences names)

```
alter table indicator alter column id set default nextval('data.seq_indicator');
alter table country alter column id set default nextval('data.seq_country');
alter table country_indicator alter column id set default
nextval('data.seq_country_indicator');
```

- set current and actual values for next iterator value

(use appropriate student's table and sequences names)

```
SELECT setval('data.seq_indicator', (select max(ID) + 1 from indicator));
SELECT setval('data.seq_country_indicator', (select max(ID) + 1 from country_indicator));
SELECT setval('data.seq_country', (select max(ID) + 1 from country));
```

That's completely OK if the student sets precalculated actual value like a static hard coded number.

```
SELECT setval('data.seq_indicator', 4);
SELECT setval('data.seq_country_indicator', 23986);
SELECT setval('data.seq_country', 217);
```

4. The query below should return all *true* check[i] for 3 rows.

(use appropriate student's table and sequences names)

```
select nextval('data.seq_indicator'),
       nextval('data.seq_country'),
       nextval('data.seq_country_indicator'),
       (select max(ID) from indicator) < (currval('data.seq_indicator') - 10) AS check1,
       (select max(ID) from country_indicator) < (currval('data.seq_country_indicator') - 10) AS
check2,
       (select max(ID) from country) < (currval('data.seq_country') - 10) AS check3;
```

	nextval ÷	nextval ÷	nextval ÷	check1 ÷	check2 ÷	check3 ÷
1	74	247	28696	• true	• true	• true

5. The query below should return all filled *column\_default* value of the column

(use appropriate student's table and sequences names)

```
select table_name, column_default
from information_schema.columns
where table_catalog = 'data' and
       table_schema = 'data' and
       column_name = 'id'
```

	table_name	column_default
1	indicator	nextval('seq_indicator'::regclass)
2	country	nextval('seq_country'::regclass)
3	country_indicator	nextval('seq_country_indicator'::regclass)

Yes | No

## Exercise 03 - Add a chronological possibility

- The “flyway.conf” file (located in “~/flyway-6.x.x/conf” directory) must have a manually modified options below (other parameters are in the default state):
  - flyway.url = jdbc:postgresql://hostname:5432/data  
(where *hostname* is IP-address, DNS name or hostname)
  - flyway.user = data
  - flyway.password = data
  - flyway.schemas = data
  - flyway.defaultSchema = data
- The command “flyway info” (through the command line) returns “state = success” for all SQL scripts with right names from exercise.
- Checks for the file **V030\_\_alter\_table\_to\_chrono.sql**
  - The commands are below for *indicator* table  
(use appropriate student’s table name)
 

```
alter table indicator add column time_start timestamp default '01.01.1972' not null;
alter table indicator add column time_end timestamp default '01.01.9999' not null;
```
  - The commands are below for *country* table  
(use appropriate student’s table name)
 

```
alter table country add column time_start timestamp default '01.01.1972' not null;
alter table country add column time_end timestamp default '01.01.9999' not null;
```
- The query below should return all filled *column\_default* value of each column  
(use appropriate student’s table name)
 

```
select table_name,column_name , column_default
from information_schema.columns
where table_catalog = 'data' and
      table_schema = 'data' and
      column_name in ('time_start', 'time_end')
order by 1
```



	table_name	column_name	column_default
1	country	time_start	'1972-01-01 00:00:00'::timestamp without time zone
2	country	time_end	'9999-01-01 00:00:00'::timestamp without time zone
3	indicator	time_start	'1972-01-01 00:00:00'::timestamp without time zone
4	indicator	time_end	'9999-01-01 00:00:00'::timestamp without time zone

5. The query below should return all true states for every check[i]

(use appropriate student's table name)

```
select min(time_start) = max(time_start) as check1,
       min(time_end) = max(time_end) as check1
from country
```

	check1	check1
1	• true	• true

6. The query below should return all true states for every check[i]

(use appropriate student's table name)

```
select min(time_start) = max(time_start) as check1,
       min(time_end) = max(time_end) as check1
from indicator
```

	check1	check1
1	• true	• true

Yes | No

## Exercise 04 - Add check constraints

- The “flyway.conf” file (located in “~/flyway-6.x.x/conf” directory) must have a manually modified options below (other parameters are in the default state):
  - flyway.url = jdbc:postgresql://hostname:5432/data  
(where *hostname* is IP-address, DNS name or hostname)
  - flyway.user = data
  - flyway.password = data
  - flyway.schemas = data
  - flyway.defaultSchema = data
- The command “flyway info” (through the command line) returns “state = success” for all SQL scripts with right names from exercise.
- Checks for the file **V040\_\_alter\_tables\_by\_cc.sql**
  - The commands are below for *indicator* table  
(use appropriate student's table name)

```

alter table indicator add constraint ch_indicator_time_start check ( time_start >=
'01.01.1972' );
alter table indicator add constraint ch_indicator_time_end check ( time_end <= '01.01.9999'
);
alter table indicator add constraint ch_indicator_unit check ( unit IN ('human', 'percent')
);

```

- The commands are below for *country* table

(use appropriate student's table name)

```

alter table country add constraint ch_country_time_start check ( time_start >= '01.01.1972'
);
alter table country add constraint ch_country_time_end check ( time_end <= '01.01.9999' );
alter table country add constraint ch_country_object_type check ( object_type IN ('country',
'continent') );

```

- The commands are below for *country\_indicator* table

(use appropriate student's table name)

```

alter table country_indicator add constraint ch_country_indicator_value check ( value >= 0 );
alter table country_indicator alter column value set NOT NULL ;

```

4. The query below should return the next snapshot of data

(use appropriate student's table name)

```

select t1.table_name, t1.constraint_name, t2.check_clause
from information_schema.table_constraints t1
INNER JOIN information_schema.check_constraints t2 ON t1.constraint_name = t2.constraint_name
where t1.constraint_catalog = 'data' and
t1.constraint_schema = 'data' and
t1.constraint_type = 'CHECK' and
t1.constraint_name like 'ch%'
order by 1;

```

table_name	constraint_name	check_clause
country	ch_country_time_start	((time_start >= '1972-01-01 00:00:00'::timestamp without time zone))
country	ch_country_time_end	((time_end <= '9999-01-01 00:00:00'::timestamp without time zone))
country	ch_country_object_type	((object_type::text = ANY ((ARRAY['country'::character varying, 'continent'::character varying]))::text[]))
country_indicator	ch_country_indicator_value	((value >= 0)::numeric)
indicator	ch_indicator_unit	((unit::text = ANY ((ARRAY['human'::character varying, 'percent'::character varying]))::text[]))
indicator	ch_indicator_time_start	((time_start >= '1972-01-01 00:00:00'::timestamp without time zone))
indicator	ch_indicator_time_end	((time_end <= '9999-01-01 00:00:00'::timestamp without time zone))

Yes | No