SQL part 9. DDL – part 2.

Constraints

1. Find out about constraints defined in table *Projects*. Use *Browser* window of *pgAdmin* tool, find node named *Constraints* under *Tables -> Projects* node. To find out which columns can't be null see *Employees* columns definitions under *Columns* node.

2. Define following constraints in *Projects* table:

Constraint name	Constraint type	Column name	Properties
pk_projects	primary key	*project_id*	
uk_projects_name	unique key	*project_name*	
	not null	*project_name*	
	not null	*start_date*	
chk_projects_end_start_date	check		*end_date* should be greater than *start_date*
chk_projects_budget	check	*project_budget*	*project_budget* should be positive
chk_projects_no_of_emp	check	*number_of_employees*	*number_of_employees* can't be negative

```
SQL> ALTER...
```

3. Try to define another constraint in *Projects* table, namely set *number_of_employees* column as not null. Did you succeed? Change column values to fulfil constraint, then define constraint once more.

```
SQL> ALTER...

SQL> UPDATE...

SQL> ALTER...
```

4. Add new column to *Projects* table. This column will show which employee manages the project. New column name is *manager_id*, its data type should be the same as data tape of column *emp_id* in *Employees* table. Next define a foreign key on *manager_id*, which points to *emp_id* in *Employees*. Constraint's name should be *projects_fk_emps*. Foreign key should restrict deletion of employee in *Employees* table if he or she manages a project. Try to accomplish the task using only one command.

```
SQL> ALTER...
```

5. Check if foreign key works. Try to set as manager of project "Advanced Data Analysis" a non-existent employees.

```
SQL> UPDATE...
```

6. Set Mark Clark as manager of project "Advanced Data Analysis". Then try to delete Mark Clark from *Employees* table. Did you succeed?

```
SQL> UPDATE...

SQL> DELETE...
```

7. Create a new table. Its name is *Assignments* and its structure is as follows:

Column name	Data type	Size	Properties
project_id	integer		Can't be null, foreign key to *project_id* of *Projects* table.
emp_id	numeric	4	Can't be null, foreign key to *emp_id* of *Employees* table.
function	variable-length string	max. 100	Can't be null, only following values are allowed: "designer", "programmer", "tester".
start_date	date		Default: current date, can't be null.
end_date	date		If set, should be greater than *start_date*.
salary	numeric	8,2	Can't be null, should be positive.

Additional information:

- multi-column primary key: name: *pk_assignments*, columns: *project_id*, *emp_id*, *start_date*.

```
SQL> CREATE TABLE...
```

8. Define at least four assignments of different employees to projects.

```
SQL> INSERT INTO...
```

9. Try to define an assignment which will violates the check constraint defined for *function* column.

```
SQL> INSERT INTO...
```

10. Remove the check constraint defined for *function* column. Next once again try do define assignment from task 9.

```
SQL> ALTER TABLE...

SQL> INSERT INTO...
```

Views

1. Define a view named *Professors*. For each professor a view should present his/her name, surname, hire date, salary and additional salary. A view should also present percentage dependence between employee's salary and additional salary (column *add_percent*). Next, create a query to retrieve all data from a view.

```
CREATE...

SELECT...
```

name	surname	hire_date	salary	add_salary	add_percent
Chris	Johnson	1975-09-15	3477.50	805.00	23.15
Carl	Jones	1973-05-01	3685.00	610.00	16.55
Andrew	Williams	1977-09-01	3232.33		0.00
Peter	Wilson	1968-07-01	4207.50		0.00

2. Define a view named *Departments_totals*, which will show department name and identifier, an average salary of employees in a department and number of employees in a department. Then, query a view.

```
CREATE...

SELECT...
```

dept_id	department	avg_salary	num_of_empls
10	ADMINISTRATION	4037.11	2
40	ALGORITHMS	3685.00	1
20	DISTRIBUTED SYSTEMS	2722.53	7
30	EXPERT SYSTEMS	1785.66	3
50	OPERATIONAL RESEARCH		0

3. Using an *Employees* table and a *Departments_totals* view find employees who earn more than an average salary in their department. Also count a difference between employees salary and average salary in his/her department.

```
SELECT...
```

surname	name	salary	department	avg_salary	diff
Johnson	Chris	3477.50	DISTRIBUTED SYSTEMS	2722.53	754.97
Smith	John	5097.01	ADMINISTRATION	4037.11	1059.90
White	Mary	3093.00	DISTRIBUTED SYSTEMS	2722.53	370.47
Williams	Andrew	3232.33	EXPERT SYSTEMS	1785.66	1446.67
Wilson	Peter	4207.50	DISTRIBUTED SYSTEMS	2722.53	1484.97

4. Using a *Departments_totals* view find department which employs the highest number of employees.

```
SELECT...
```

department	num_of_empls
DISTRIBUTED SYSTEMS	7

5. Define a view named *Emps_and_bosses*. A view should present names and surnames of employees, their salaries, names, surnames and salaries of their bosses. A view should present only those employees, whose salaries are lower than salaries of their bosses. Next, query a view.

```
CREATE...
```

employee	emp_salary	boss	boss_salary
Bell Tom	1850.00	Williams Andrew	3232.33
Clark Mark	2977.21	Smith John	5097.01
Edwards Ana	1837.50	Wilson Peter	4207.50
Green Ian	2087.20	Wilson Peter	4207.50
Jackson Peter	1062.33	Wilson Peter	4207.50
Johnson Chris	3477.50	Wilson Peter	4207.50
Jones Carl	3685.00	Smith John	5097.01
Lewis Arnold	2218.50	Johnson Chris	3477.50
White Mary	3093.00	Wilson Peter	4207.50
Williams Andrew	3232.33	Smith John	5097.01
Wilson Peter	4207.50	Smith John	5097.01
Wood Adam	1062.33	Johnson Chris	3477.50
Young Wayne	2136.50	Jones Carl	3685.00