

Указаны примеры решения заданий по курсу "Программирование баз данных".

Исходный файл представляет собой таблицу данных о транспортных средствах (таблица cars), их расположениях (таблица dislocations), заявках (таблица orders), маршрутах (таблица routes), станциях (таблица stations), работниках (таблица person) и т.д.

Представлены задания, которые необходимо решить с помощью sql-запросов.

Запросы и администрирование базы данных происходит с помощью программы "pgAdmin4".

**Задание 1.** Создать базу данных PostgreSQL и загрузить туда файлы .csv таблиц, указав Первичный ключ и Внешний ключ к каждой таблице.

Запросы для создания некоторых таблиц:

Станции				
station_id	station_name	min	max	wait_cost
ID станции	Название станции	Минимальное количество вагонов на станции	Максимальное количество вагонов на станции	Цена простоя за сутки (может быть неизвестна)
240000	Муром I	0	600	600
240809	Эсино	0	600	600
242005	Великодворье	0	600	600
242306	Комиссаровка	0	600	600
242607	Улыбышево	0	600	600
243506	Бутылицы	0	600	600
243607	Навашино	0	600	600
244208	Арзамас II	0	600	600
247009	Красный Узел	0	600	600
247206	Нуя	0	600	600
247403	Ардатов	0	600	600
248105	Канаш	0	600	600
248302	Цивильск	64	213	600
248504	Чебоксары	0	600	600

Рисунок 1. Таблица Stations

```
DROP TABLE IF EXISTS Stations CASCADE;
```

```
CREATE TABLE Stations
```

```
(
```

```
    station_id CHAR(6) NOT NULL UNIQUE,
```

```
    station_name VARCHAR(60) NOT NULL,
```

```
    min_ INT NOT NULL CONSTRAINT min_check CHECK(min_>=0),
```

```
    max_ INT NOT NULL CONSTRAINT max_check CHECK (max_ >= min_),
```

```
    wait_cost INT,
```

```
    PRIMARY KEY(station_id)
```

```
);
```

Загрузка данных в таблицу Stations через SQL Shell(psql):

```
\copy Car_types FROM 'C:\Users\Андрей\Desktop\5 сем\SQL\Task2\Raw\stations.csv'  
DELIMITER ';' HEADER ENCODING 'WIN1251' CSV;
```

Местоположение вагонов							
id	station_id	loaded_empty	cars_quantity		car_type	period	wait_time
ID расположения	станции (обязательно шесть цифр, не пуст)	Флаг загруженного вагона (0 или 1, может быть	Количество вагонов (не пустое и не может быть больше 100 и отрицательным)		Период	Период	Время ожидания
1	240000	0	1		1	1	от 1 до 5
2	240000	0	1		1	1	от 1 до 6
3	240000	0	1		1	1	от 4 до 8
4	240000	0	1		1	1	от 19 до 21
5	240000	0	1		1	1	от 19 до 23
6	240000	0	1		1	6	от 1 до 0
7	240000	0	2		1	1	от 0 до 0
8	240000	0	2		1	1	от 3 до 22
9	240000	0	3		1	-3	от 0 до 7

Рисунок 2. Таблица Dislocations

DROP TABLE IF EXISTS Dislocations CASCADE;

CREATE TABLE Dislocations

(

station\_id CHAR(6) NOT NULL REFERENCES Stations (station\_id),

loaded\_empty INT,

cars\_quantity INT NOT NULL CONSTRAINT cars\_quantity\_check CHECK (cars\_quantity<100 AND cars\_quantity >=0),

car\_type INT NOT NULL REFERENCES Car\_types (car\_type\_id),

period INT NOT NULL,

wait\_time VARCHAR (60) NOT NULL

)

Задание 2. Написать запросы на следующие задания:

- Из таблицы «Dislocations» вывести следующие данные: id станции, тип вагона, время ожидания (среднее), период. Не должно быть одинаковых строк в результате. Результат отсортировать по убыванию времени ожидания (в случае равного времени ожидания – по возрастанию периода).

	1	2	3
1	Типы вагонов		
2	car_type_id	car_type_name	max_weight
3	ID типа вагона	Наименование	Максимальный вес
4	0	цистерна	300
5	1	грузовой	500
6	2	сдвоенная цистерна	450
7	3	платформа	250

Рисунок 3. Таблица Car types

SELECT DISTINCT ON (car\_type\_name, period, (substring (wait\_time, 'от (\d+)')::int + substring (wait\_time, 'до (\d+)')::int)/2 ) car\_type\_name,

period, (substring (wait\_time, 'от (\d+)')::int + substring (wait\_time, 'до (\d+)')::int)/2 as average, dislocations.station\_id

FROM public.dislocations, public.car\_types

WHERE public.car\_types.car\_type\_id=public.dislocations.car\_type

ORDER BY average DESC, period ASC;

Вывод:

	car_type_name character varying (30)	period integer	average integer	station_id character
1	грузовой	1	107	862305
2	грузовой	1	88	860009
3	грузовой	1	87	860009
4	грузовой	1	86	864207
5	грузовой	1	83	860009
6	грузовой	1	81	864207
7	грузовой	1	80	872504
8	грузовой	1	76	860009
9	грузовой	1	71	862305
10	грузовой	1	69	830003

Рисунок 4. Вывод номер 1

2) Агрегатные функции и подзапросы, регулярные выражения:

Для каждого id станции, с минимум двумя нулями в id, вывести каждый период, среднее количество загруженных вагонов по каждому периоду, среднее количество незагруженных вагонов по каждому периоду и сумму этих средних (таблица «Dislocation»). Вывод отсортировать по убыванию периода. Если в столбце с суммой содержится ноль, то соответствующий кортеж не выводить.

```
SELECT station_id, period, round(AVG(case when loaded_empty = True then cars_quantity else 0 end),2)
as avg_loaded,
```

```
round(AVG(case when loaded_empty = False then cars_quantity else 0 end),2) as avg_empty,
```

```
round((AVG(case when loaded_empty = True then cars_quantity else 0 end) + AVG(case when
loaded_empty = False then cars_quantity else 0 end)),2) as sum_avg_loaded_avg_empty
```

```
FROM dislocations where station_id SIMILAR TO '[1-9]*0+[1-9]*0+[1-9]*'
```

```
GROUP BY station_id, period having (AVG(case when loaded_empty = True then cars_quantity else 0 end)
```

```
+ AVG(case when loaded_empty = False then cars_quantity else 0 end)) !=1
```

```
order by station_id asc, period desc;
```

	station_id character	period integer	avg_loaded numeric	avg_empty numeric	sum_avg_loaded_avg_empty numeric
1	240000	2	11.00	0.00	11.00
2	240000	1	0.25	1.50	1.75
3	240000	-3	0.00	3.00	3.00
4	250001	1	12.00	0.50	12.50
5	250707	1	0.67	0.67	1.33
6	251004	3	10.00	0.00	10.00
7	251004	1	3.00	0.33	3.33

Рисунок 5. Вывод номер 2

Для каждого периода вывести количество различных станций, соответствующих периоду, среднее по периоду число вагонов, при условии, что среднее по периоду число вагонов отличается от максимального среднего по периоду числа вагонов не более чем на 5 (таблица «Dislocation»).

```
SELECT period, COUNT(DISTINCT station_id) as count_station, round(AVG(cars_quantity),2) as
cars_quantity_avg,
```

round((select max (avg\_q) from (SELECT Avg(cars\_quantity) as avg\_q FROM dislocations GROUP BY period) as avg\_max), 2) as avg\_max

FROM dislocations GROUP BY period

HAVING (SELECT MAX(avg\_q) FROM (SELECT AVG(cars\_quantity) as avg\_q FROM dislocations GROUP BY period)) - AVG(cars\_quantity) <= 7;

	period integer	count_station bigint	cars_quantity_avg numeric	avg_max numeric
1	4	22	13.79	19.25
2	8	27	13.87	19.25
3	11	10	19.25	19.25
4	14	7	18.22	19.25
5	16	6	12.86	19.25

Рисунок 6. Вывод номер 3

### 3) Оконные функции, работа с результатами запроса, представления, join

Создать и заполнить таблицу из следующих столбцов: 1. название станции (с пятью гласными буквами); 2. название отделения (которое соответствует названию станции).

Таблицы departments и stations\_departments:

	1	2
1	Отделения	
2	department_id	name_department
3	ID отделения	Название отделения
4	1	Муромское отделение
5	2	Казанское отделение
6	3	Ижевское отделение
7	4	Горьковское отделение
8	5	Кировское отделение
9	6	Омское отделение

Рисунок 7. Таблица departments

	1	2
1	Станции и отделения	
2	station_id	department_id
3	ID станции	Название отделения
4	240000	1
5	240809	1
6	242005	1
7	242306	1
8	242607	1
9	243506	1
10	243607	1

Рисунок 8. Таблица stations\_departments

DROP table if exists station\_departments;

CREATE TABLE station\_departments AS SELECT station\_name, name\_department

FROM stations

JOIN stations\_departments ON stations.station\_id = stations\_departments.station\_id

JOIN departments ON stations\_departments.department\_id = departments.department\_id

WHERE REGEXP\_COUNT(lower(station\_name), '[аеёиоуыэюя]') = 5;

Результат выполнения запроса:

	station_name character varying (60)	name_department character varying (60)
1	Великодворье	Муромское отделение
2	Комиссаровка	Муромское отделение
3	Улыбышево	Муромское отделение
4	Новочебоксарск	Казанское отделение
5	Кожевненное	Горьковское отделение
6	Киров-Котласский	Кировское отделение
7	Подосиновец	Кировское отделение
8	Омск-Пассажирский	Омское отделение

Рисунок 9. Таблица station\_departments

### Работа с представлениями:

Выполнить проверку на наличие маршрутов для всех обязательных заявок: если будут заявки без маршрутов, то выбрать до 4 маршрутов (по возможности) на эту заявку со станций дислокации и со станций прихода заявок (станция дислокации = станция отправления маршрута, станция отправления заявки = станция назначения маршрута или конечная станция другой заявки = станция отправления маршрута, станция отправления заявки = станция назначения маршрута). Вывести отобранные маршруты и указать номер соответствующей заявки.

	1	2	3	4	5	6	7	8	9
1	Заявки								
2	order_id	station_from	station_to	revenue_per_car	car_required	car_type	time_load_one_car	time_unload_one_car	must_do
3	ID заявки	Станция отправления	Станция назначения	Доход от вагона	Число вагонов	Тип вагона	Время загрузки одного вагона	Время разгрузки одного вагона	Обязательность выполнения (0 или 1, может быть неизвестно)
4	4	863007	864207	3617,6	2719	0	1,8	1,6	1
5	5	860206	864207	3311,62	2190	0	2,4	1,6	1
6	7	865002	864207	3873,06	1739	0	2,5	1,6	1
7	8	863308	864207	3580,28	1696	0	4	1,6	1
8	9	865208	864103	3956,675	1623	0	1,8	2,9	1
9	10	862502	864207	5550,28	1525	0	3,8	1,9	1
10	11	864902	984700	43140,28	1500	0	2,8	15	1
11	12	863007	986103	32761,1	1497	0	2,8	15	1
12	13	872701	967808	41704	1495	0	3,5	16,6	0
13	20	865407	864207	4334	1294	2	3	1,8	1
14	21	864902	864207	4397,1	1292	0	2,8	1,7	1
15	22	871906	870104	2700	1283	1	2,5	1,1	1

Рисунок 10. Таблица orders

	1	2	3	4
1	Маршруты			
2	route_id	station_from	station_to	avg_cost
3	ID маршрута	Начальная станция (если 0, то записать 000102)	Конечная станция (если 0, то использовать 000206)	Стоимость маршрута
4	1	927001	903204	4626
5	2	927001	923903	5857
6	3	927001	923800	6109
7	4	927001	923706	6109
8	5	927001	923509	6413
9	6	927001	923104	10909
10	7	927001	922703	10417
11	8	927001	922008	9698
12	9	927001	921202	8599
13	10	927001	920708	7866

Рисунок 11. Таблица routes

Результат создания представления:

	order_id integer	route_id integer	station_from character	station_to character	row_number bigint
1	4	4892	862803	863007	1
2	4	4927	841402	863007	2
3	4	5028	840109	863007	3
4	4	5070	850204	863007	4
5	5	4908	862803	860206	1

Рисунок 12. Представление test\_view3

Код SQL создания представления:

```
DROP VIEW IF EXISTS test_view3;
```

```
CREATE VIEW test_view3 AS
```

```
SELECT * FROM
```

```
(
```

```
    SELECT *, ROW_NUMBER () OVER(PARTITION BY order_id ORDER BY route_id ASC ) AS  
row_number
```

```
FROM
```

```
(
```

```
    SELECT
```

```
        order_id, routes.route_id, routes.station_from, routes.station_to
```

```
FROM
```

```
    (SELECT
```

```
        orders.order_id, orders.station_from
```

```
FROM
```

```
    orders
```

```
WHERE
```

```
    must_do = 1
```

```
EXCEPT
```

```
SELECT
```

```
    orders.order_id, orders.station_from
```

```
FROM
```

```
    orders
```

```
JOIN
```

```
    routes
```

```

                                ON (routes.station_from = orders.station_from AND
routes.station_to = orders.station_to)

                                WHERE must_do = 1)

                                as orders_ids_

CROSS JOIN

                                (SELECT DISTINCT station_id FROM dislocations) as disloc_

JOIN

                                routes ON routes.station_from = disloc_.station_id AND routes.station_to =
orders_ids_.station_from

-- ORDER BY order_id

UNION

SELECT

                                order_id, routes.route_id, routes.station_from, routes.station_to

FROM

                                (SELECT

                                        orders.order_id, orders.station_from

                                FROM

                                        orders

                                WHERE

                                        must_do = 1

                                EXCEPT

                                SELECT

                                        orders.order_id, orders.station_from

                                FROM

                                        orders

                                JOIN

                                        routes

                                ON (routes.station_from = orders.station_from AND
routes.station_to = orders.station_to)

                                WHERE must_do = 1)

                                as orders_ids_

JOIN

                                routes ON routes.station_to = orders_ids_.station_from

```

```
JOIN
      (SELECT DISTINCT station_to FROM orders) as ord_ ON ord_.station_to =
routes.station_from
      ORDER BY order_id, route_id
    )
  )as sub_
WHERE sub_.row_number<=4
ORDER BY order_id,row_number
```