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	ппа	созлания	некоторых	таблин:
Janpoon	ДЛЛ	СОЗДапил	пскоторыл	таолиц.

1	Станции								
2	station_id	station_name	min	max	wait_cost				
3	ID станции	Название станции	Минимальное количество вагонов на станции	Максимальное количество вагонов на станции	Цена простоя за сутки (может быть неизвестна)				
4	240000	Муром I	0	600	600				
5	240809	Эсино	0	600	600				
6	242005	Великодворье	0	600	600				
7	242306	Комиссаровка	0	600	600				
8	242607	Улыбышево	0	600	600				
9	243506	Бутылицы	0	600	600				
10	243607	Навашино	0	600	600				
11	244208	Арзамас II	0	600	600				
12	247009	Красный Узел	0	600	600				
13	247206	Нуя	0	600	600				
14	247403	Ардатов	0	600	600				
15	248105	Канаш	0	600	600				
16	248302	Цивильск	64	213	600				
17	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\Aндрей\Desktop\5 cem\SQL\Task2\Raw\stations.csv' DELIMITER ';' HEADER ENCODING 'WIN1251' CSV;

1	Местоположение вагонов									
2	id	station_id	loaded_empty cars_quantity ca		car_type	period	wait_time			
3	ID расположения	танции (обязательно шесть цифр, не пуст	Флаг загруженного вагона (0 или 1, может быт Количество вагонов (не пустое и не может быть больше 100 и отрицательны		ьным)	Период	Время ожидания			
4	1	240000	0 1		1	1	от 1 до 5			
5	2	240000	0 1		1	1	от 1 до 6			
6	3	240000	0	1	1	1	от 4 до 8			
7	4	240000	0 1		1	1	от 19 до 21			
8	5	240000	0	1	1	1	от 1 до 23			
9	6	240000	0	1	1	6	от 0 до 0			
10	7	240000	0		1	1	от 3 до 22			
11	8	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:

1) 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).

4	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, 'oт (\d+)')::int + substring (wait_time, 'до (\d+)')::int)/2 ) car_type_name,

period, (substring (wait_time, 'oт (\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;
```

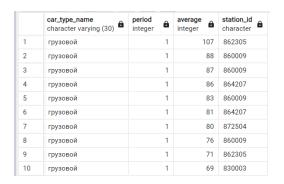


Рисунок 4. Вывод номер 1

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) <u>Оконные функции, работа с результатами запроса, представления,</u> 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

	1	2
1	Стань	ции 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 7. Таблица departments

Рисунок 8. Таблица stations_departments

DROP table if exists station departments;

 ${\tt CREATE\ TABLE\ station_departments\ AS\ SELECT\ station_name,\ name_department}$

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	Кожевенное	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.

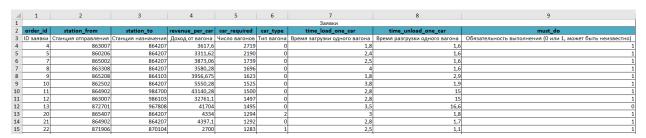


Рисунок 10. Таблица orders

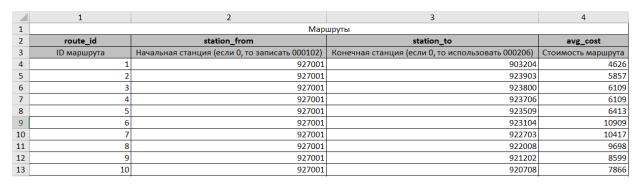


Рисунок 11. Таблица routes

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