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:



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;



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).

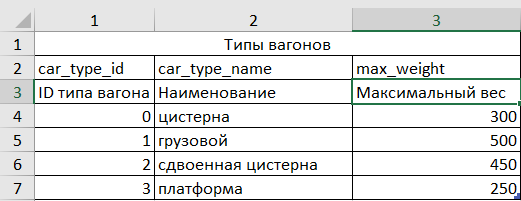


Рисунок 3. Таблица Car types

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:

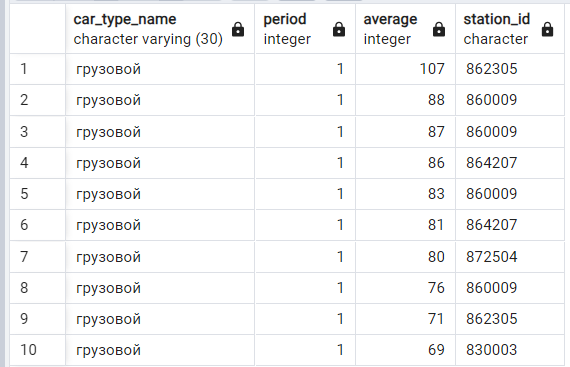


Рисунок 4. Вывод номер 1

1. *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;

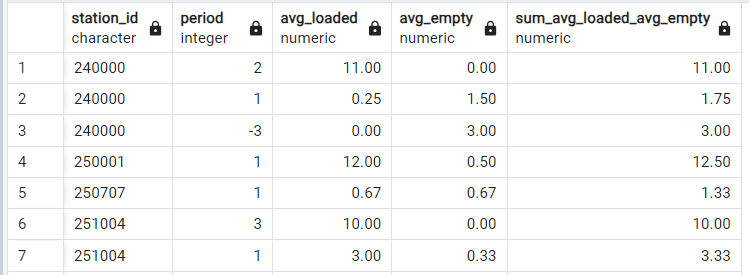


Рисунок 5. Вывод номер 2

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;

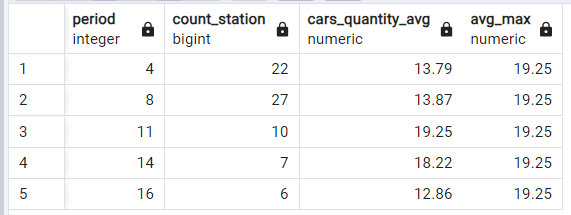


Рисунок . Вывод номер 3

1. *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:

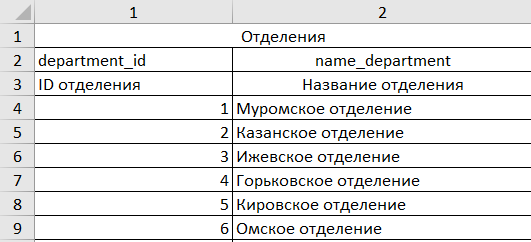
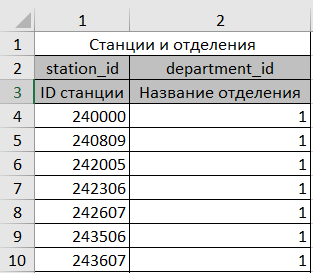


Рисунок 7. Таблица departments Рисунок 8. Таблица stations\_departments

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:

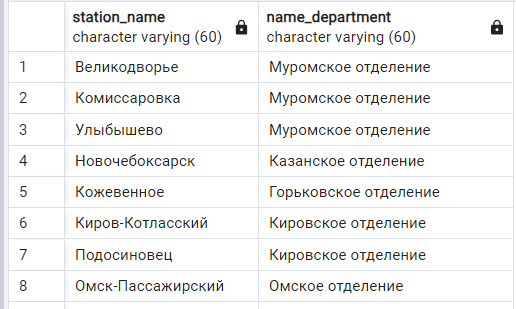


Рисунок 9. Таблица station\_departments

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.

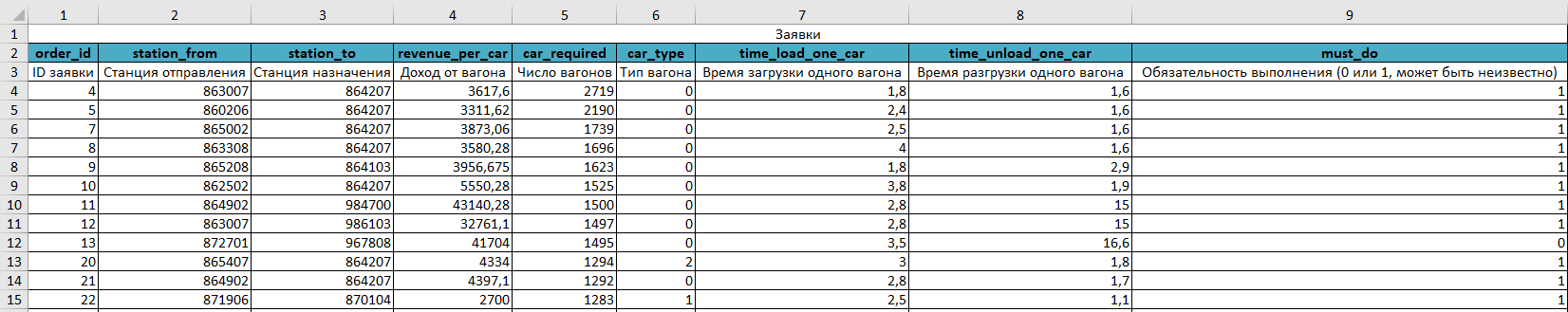


Рисунок 10. Таблица orders

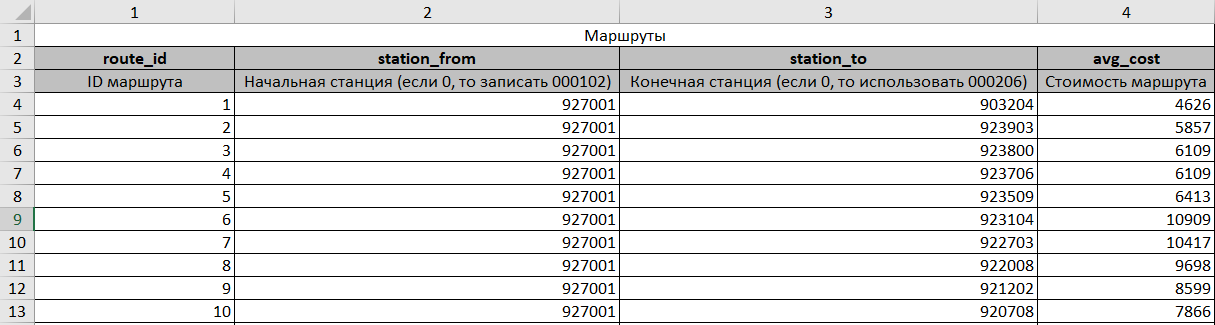


Рисунок 11. Таблица routes

Результат создания представления:

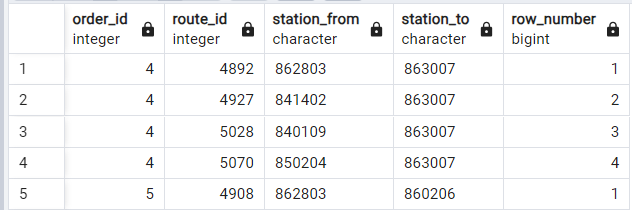


Рисунок 12. Представление test\_view3

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