**Консоль**

**Задание 1**

От разработчиков поступила задача: нужно выяснить, какие запросы шли с IP-адреса. IP-адрес состоит из четырёх чисел, они разделены точками. Тебе нужны адреса, которые начинаются с «233.201.».

Логи лежат на удалённом сервере по адресу logs/2019/12. День, когда случилась ошибка, неизвестен.

Твоя задача — узнать, какие запросы были отправлены.

В ответе приложи:

1. команду, которой тебе удалось получить нужные логи;
2. подходящие строки, например: 184.79.247.161 - - [30/12/2019:21:38:13 +0000] "PUT /alerts HTTP/1.1" 400 3557

**Задание 2**

В системе обнаружен баг. Он проявлялся 30.12.2019 и 31.12.2019 с 21:30:00 до 21:39:59. При этом появлялись ошибки с номерами 400 и 500. Твоя задача — сохранить в отдельный файл логи, которые были записаны в этот период.

Затем эти логи надо разложить по отдельным файлам: логи с одинаковой ошибкой положи в один файл. Как это сделать:

1. В домашней директории на удалённом сервере создай директорию bug1.
2. Все запросы, которые произошли в указанный период, положи в файл main.txt в директорию bug1.
3. Внутри директории bug1 создай директорию events.
4. Внутри директории events создай файлы для ошибок с номерами 400 и 500. Назови эти файлы 400.txt и 500.txt соответственно. В них выдели логи с соответствующей ошибкой из файла main.txt.

В ответе приложи:

1. команды, которые создают директории bug1 и events;
2. команду, которой ты выбираешь запросы за указанный период. Это те запросы, которыми ты отбираешь логи в файл main.txt;
3. команды, которыми ты кладёшь логи в файлы 400.txt и 500.txt из main.txt;
4. тексты файлов 400.txt и 500.txt.

https://code.s3.yandex.net/qa/schemes/project-roz.png

**База данных**

**Описание данных**

База данных о поездках такси в Чикаго:

Таблица neighborhoods — информация о районах города:

* neighborhood\_id — код района;
* name — название района.

Таблица cabs — информация о такси:

* cab\_id — идентификатор такси;
* vehicle\_id — уникальный идентификатор автомобиля;
* company\_name — компания, которой принадлежит автомобиль.

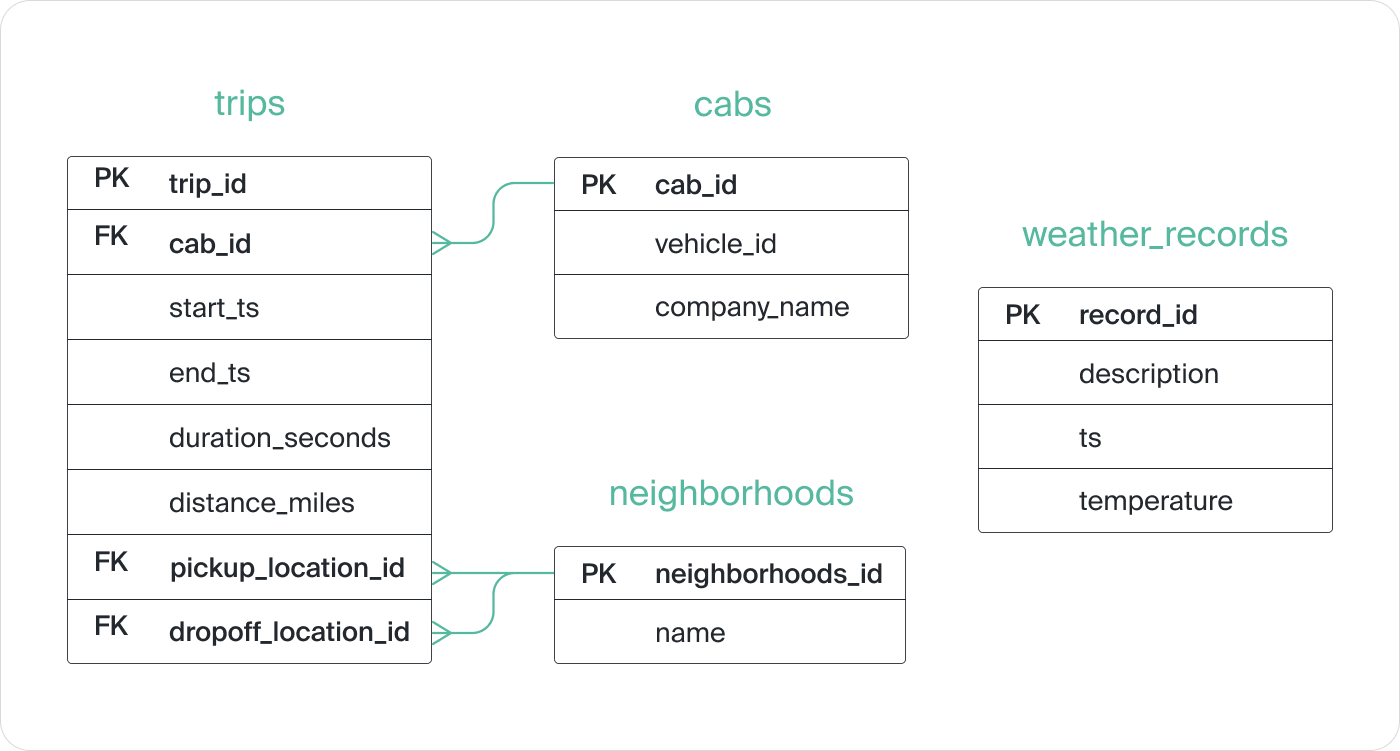
Таблица trips — информация о поездках:

* trip\_id — код поездки;
* cab\_id — идентификатор такси, на котором была совершена поездка;
* start\_ts — дата и время начала поездки (время округлено до часа);
* end\_ts — дата и время окончания поездки (время округлено до часа);
* duration\_seconds — длительность поездки в секундах;
* distance\_miles — дальность поездки в милях;
* pickup\_location\_id — код района города, в котором была начата поездка;
* dropoff\_location\_id — код района города, в котором завершилась поездка.

Таблица weather\_records — информация о погоде:

* record\_id — код записи погодных наблюдений;
* ts — дата и время наблюдения (время округлено до часа);
* temperature — температура на момент наблюдения;
* description — краткое описание погодных условий. Например, light rain или scattered clouds.

**Схема таблиц**



В базе данных нет прямой связи между таблицами trips и weather\_records. Связать эти таблицы можно по времени начала поездки (trips.start\_ts) и моменту погодных наблюдений (weather\_records.ts).

**Задание 1**

У тебя есть база данных с поездками на такси. По плану на линию обслуживания должно было выйти 10550 автомобилей — эта цифра покрывает спрос пользователей. Команде поступило много жалоб: свободных автомобилей оказалось недостаточно. Сколько такси вышло на линии на самом деле? Информация о всех машинах на линии есть в таблице cabs.

1. Зайди на удалённый сервер.
2. Подключись к базе данных chicago\_taxi, используй логин morty и пароль smith.
3. Посчитай, сколько всего автомобилей в таблице cabs. Учти, что один автомобиль может принадлежать разным компаниями.

В ответе приложи:

1. число автомобилей;
2. запрос, которым тебе удалось решить задачу.

**Задание 2**

Посчитай количество автомобилей в каждой компании из таблицы cabs. Отсортируй значения по убыванию. Команда предполагает, что некоторые компании не вывели достаточно автомобилей на линию.

Выведи те компании, в которых меньше 100 автомобилей. Поле с числом автомобилей назови cnt, поле с названием компании — company\_name.

Чтобы решить задачу, примени оператор HAVING — аналог WHERE для агрегирующих функций. Изучи в документации, как работает оператор:

(<https://postgrespro.ru/docs/postgrespro/11/queries-table-expressions#QUERIES-GROUP>)

В ответе приложи:

1. список компаний с числом автомобилей меньше 100;
2. запрос, которым тебе удалось решить задачу.

Обрати внимание: в консоль выводится неполный список. Чтобы просмотреть его полностью, нажми Enter или используй стрелки на клавиатуре.

**Задание 3**

В приложении такси рассчитывается коэффициент стоимости поездки. Если погода хорошая, значение коэффициента равно 1. Если на улице дождь или шторм, коэффициент повышается до 2. У команды есть гипотеза, что в расчётах коэффициента ошибка. Чтобы проверить расчёт коэффициента, команде нужна выборка данных: разработчик может сверить коэффициент с данными в логах и исправить баг. Твоя задача — получить выборку.

Чтобы это сделать:

1. Получи описание погодных условий из таблицы weather\_records для каждого часа.
2. Раздели все часы на две группы оператором CASE: Bad, если поле description содержит слова rain или storm; Good для всех остальных.
3. Полученное поле назови weather\_conditions.

В результирующей таблице должно быть два поля — дата и час (ts) и weather\_conditions.

Сделай выборку за период с 2017-11-05 00:00 по 2017-11-06 00:00.

В ответе приложи:

1. полученную таблицу с данными за указанный период;
2. запрос, которым удалось решить задачу.

**Задание 4**

После обновления ПО таксопарки стали сообщать, что прибыль, которую они получают, не сходится с данными, которые отдаёт приложение. Разработка предполагает, что проблема может быть в данных о количестве поездок.

Чтобы определить, есть ли баг, нужно получить выборку с количеством поездок каждого таксопарка за 15 и 16 ноября 2017 года.

1. Выведи поле company\_name. Поле с числом поездок назови trips\_amount и выведи его.
2. Результаты, полученные в поле trips\_amount, отсортируй по убыванию.

Подсказка: чтобы решить задачу, соедини таблицы cabs и trips. Примени агрегирующие функции с группировкой. Не забудь написать конструкцию с условием.

В ответе приложи:

1. полученную таблицу с данными за указанный период;
2. запрос, которым удалось решить задачу.