**Школа стажеров-аналитиков**

**Тестовое задание**

**Выполнила: Арбузова Александра**

Оглавление

[**1.** **Задачи SQL** 2](#_Toc88428846)

[**1.1. Задание 1** 2](#_Toc88428847)

[**1.2. Задание 2** 2](#_Toc88428848)

[**1.3. Задание 3** 2](#_Toc88428849)

[**1.4. Задание 4** 3](#_Toc88428850)

[**1.5. Задание 5** 4](#_Toc88428851)

[**1.6. Задание 6** 6](#_Toc88428852)

[**2.** **Математические задачи** 10](#_Toc88428853)

[**2.1. Задание 1** 10](#_Toc88428854)

[**2.2. Задание 2** 10](#_Toc88428855)

[**2.3. Задание 3** 11](#_Toc88428856)

[**3.** **Задачи на основы программирования** 12](#_Toc88428857)

# **Задачи SQL**

## **Задание 1**

**Решение:**

SELECT name, COUNT(\*) AS count

FROM Passenger INNER JOIN Pass\_in\_trip

ON Passenger.id=Pass\_in\_trip.passenger

GROUP BY name

ORDER BY count DESC, name;

**Пояснение:**

Имена пассажиров содержатся только в таблице Passenger. При этом информация о полетах, связанная с пассажирами, находится в таблице Pass\_in\_trip. Тип связи между этими таблицами – «Один ко многим», где Passenger – главная таблица, Pass\_in\_trip – связанная. Применяем к ним тип соединения INNER JOIN, так как нам не нужна информация о пассажирах, не совершавших полет.

Чтобы посчитать количество полетов, нужно сгруппировать данные по пассажирам – GROUP BY name.

## **Задание 2**

**Решение:**

SELECT DISTINCT TIMEDIFF(

(SELECT end\_pair FROM Timepair WHERE id=4),

(SELECT start\_pair FROM Timepair WHERE id=2)

) AS time

FROM Timepair;

## **Задание 3**

**Решение:**

SELECT DISTINCT Rooms.\*

FROM Reservations INNER JOIN Rooms ON Reservations.room\_id=Rooms.id

WHERE WEEK(start\_date,1)=12 AND YEAR(start\_date)=2020

OR

WEEK(end\_date,1)=12 AND YEAR(end\_date)=2020

**Пояснение:**

Сначала нужно сделать внутреннее соединение для таблиц Reservations и Rooms, где Rooms – главная таблица, Rooms.id – первичный ключ, а Reservations – зависимая таблица, Reservations.room\_id – внешний ключ.

Дальше отбираем строки из получившегося соединения по условию после ключевого слова WHERE. Здесь двойное условие через OR, так как комнаты, которые были зарезервированы в течении 12 недели 2020 года могут быть в двух вариантах:

* дата заезда на 12 неделе 2020 года, тогда дата выезда уже не важна. Это выражается условием:

WEEK(start\_date,1)=12 AND YEAR(start\_date)=2020

* дата заезда не важна, но дата выезда должна быть тоже на 12 неделе 2020 года (в таком случае я считаю, что бронь также была в течении 12 недели, даже если это был всего 1 день на 12 неделе). Это выражается условием:

WEEK(end\_date,1)=12 AND YEAR(end\_date)=2020

В обоих этих условиях используется функция WEEK(), аргумент 1 в ней означает, что неделя начинается с понедельника.

Ключевое слово DISTINCT убирает дубликаты в результирующей таблице, так как в таблице Reservations может быть много записей для одной и той же комнаты.

## **Задание 4**

**Решение 1:**

WITH extr1 AS(

SELECT classroom, COUNT(\*) AS amount

FROM Schedule

GROUP BY classroom

)

SELECT classroom

FROM extr1

WHERE amount=(SELECT MAX(amount) FROM extr1);

**Пояснение 1:**

Сначала создадим CTE extr1, в котором для каждого кабинета подсчитывается количество раз, когда он используется. Далее выполняем SELECT-запрос к результату общего табличного выражения – то есть к таблице extr1. В этом SELECT-запросе выбираются те строки, которые удовлетворяют условию после ключевого слова WHERE. То есть те, для которых значение amount будет максимальным в таблице extr1.

**Решение 2:**

SELECT classroom

FROM Schedule

GROUP BY classroom

HAVING COUNT(\*)=(

SELECT MAX(amount)

FROM(

SELECT COUNT(\*) AS amount

FROM Schedule

GROUP BY classroom

) AS Extr1

);

**Пояснение 2:**

Выбираем номера кабинетов из таблицы Schedule. Выполняем группировку таблицы по этому полю. В каждой группе подсчитываем количество записей и оставляем те группы, которые соответствуют условию после ключевого слова HAVING. То есть те, для которых подсчитанное количество является максимальным.

## **Задание 5**

**Решение:**

WITH result1 AS

(

SELECT DISTINCT date

FROM Income\_o

),

result2 AS

(

SELECT date AS dt1,

LEAD(date) OVER (ORDER BY date) AS dt2

FROM result1

)

SELECT DISTINCT

CAST(COALESCE(SUM(out) OVER (PARTITION BY dt1), 0) AS SMALLMONEY) AS qty,

dt1,dt2

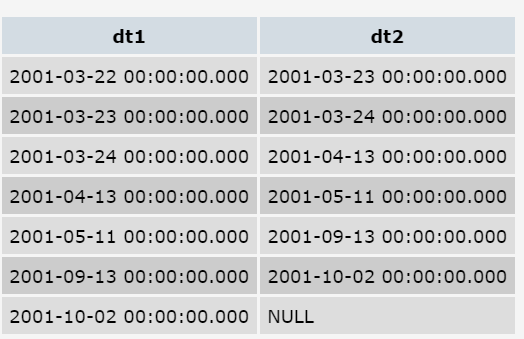
FROM result2 LEFT JOIN Outcome\_o ON

Outcome\_o.date>dt1 AND Outcome\_o.date<=dt2

WHERE dt2 IS NOT NULL

**Пояснение:**

Так как в таблице Income\_o есть повторяющиеся даты, то перед применением LEAD() нужно выбрать только строки с уникальными датами – для этого и служит CTE result1. В CTE result2 как раз находятся пары последовательных дат. Этот список выглядит так:



Необходимо помнить, что в dt2 будет NULL-значение.

Далее выполняем LEFT JOIN для result2 и Outcome\_o по условию, что дата из (Outcome\_o.date) входит в полуинтервал (dt1; dt2]. При этом выполняем суммирование по полю Outcome\_o.out для группы полей внутри каждого такого интервала (dt1; dt2] – для разных значений date из Outcome\_o может быть один и тот же интервал, а значит, одно и то же значение dt1, именно поэтому в параметрах PARTITION BY в этой функции указан столбец dt1.

При LEFT JOIN некоторым строкам из первой таблицы может не быть соответствия во второй таблице – это означает, что в этот временной интервал количество выданных денег было равно 0. Чтобы в столбце qty не было NULL-значения напротив этого интервала, была применена функция COALESCE().

****

## **Задание 6**

Мне больше нравится Решение 1 с использованием нескольких CTE, так как оно нагляднее. Решение 2 демонстрирует, как можно решить эту задачу с одним CTE, используя оконную функцию LEAD().

**Решение 1:**

WITH result2 AS

(

SELECT num, name, date, gr,

ROW\_NUMBER() OVER (PARTITION BY gr ORDER BY num) AS num\_in\_group

FROM

(

SELECT

ROW\_NUMBER() OVER (ORDER BY date, name) AS num,

name, date,

NTILE(2) OVER () AS gr

FROM Battles

) AS result1

),

left\_column AS

(

SELECT \*

FROM result2

WHERE gr=1

),

right\_column AS

(

SELECT \*

FROM result2

WHERE gr=2

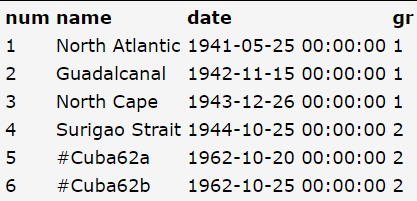
)

SELECT L.num AS rn\_1, L.name AS name\_1, L.date AS date\_1, R.num AS rn\_2, R.name AS name\_2, R.date AS date\_2

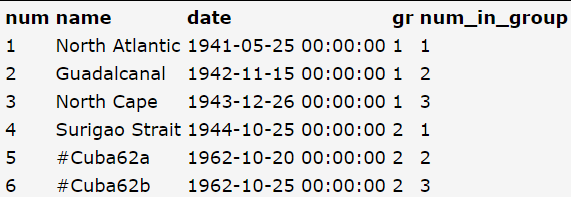
FROM left\_column L LEFT JOIN right\_column R ON L.num\_in\_group=R.num\_in\_group;

**Пояснение 1:**

Сначала выполняется подзапрос result1. В результате выводятся номера, названия и даты всех битв из таблицы Battles. Также выводится вспомогательный столбец gr – он идентифицирует к первому или ко второму суперстолбцу будет относиться каждая запись:



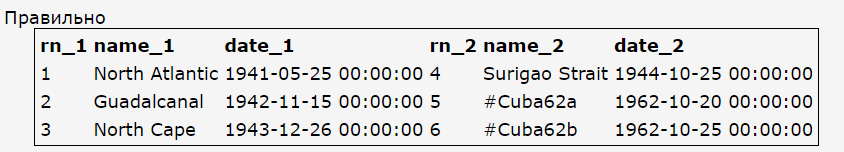
Далее выполняется запрос, являющийся CTE result2. Здесь к результирующей таблице из result1добавляется вспомогательный столбец num\_in\_group. В нем указаны номера каждой записи внутри своей группы:



Дальше выполняется CTE left\_column. В эту результирующую таблицу попадают записи из result2, у которых gr=1.

Затем выполняется CTE right\_column. В эту результирующую таблицу попадают записи из result2, у которых gr=2.

Наконец после выполнения всех CTE выполняется SELECT-запрос с выводом всех битв в 2 суперстолбца. Для этого результаты CTE left\_column соединяются с результатами CTE right\_column через LEFT JOIN. Нужно использовать именно LEFT JOIN, так как при нечетном количестве битв нужно сохранить NULL в правом суперстолбце. Соединение происходит по вспомогательному столбцу num\_in\_group.



**Решение 2:**

WITH result2 AS

(

SELECT num, name, date, gr,

ROW\_NUMBER() OVER (PARTITION BY gr ORDER BY num) AS num\_in\_group

FROM

(

SELECT

ROW\_NUMBER() OVER (ORDER BY date, name) AS num,

name, date,

NTILE(2) OVER () AS gr

FROM Battles

) AS result1

)

SELECT \*

FROM

(

SELECT num AS rn\_1, name AS name\_1, date AS date\_1,

LEAD(num) OVER (Partition by num\_in\_group ORDER BY num) AS rn\_2,

LEAD(name) OVER (Partition by num\_in\_group ORDER BY num) AS name\_2,

LEAD(date) OVER (Partition by num\_in\_group ORDER BY num) AS date\_2

FROM result2

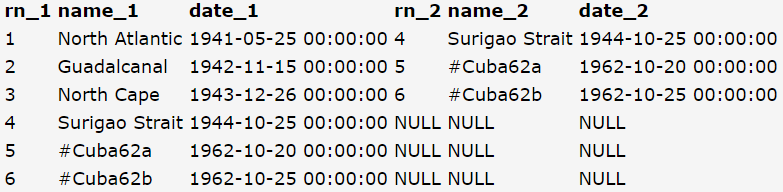
) AS result3

WHERE rn\_1 IN (SELECT num FROM result2 WHERE gr=1);

**Пояснение 2:**

Пояснение к CTE result2 можно описано в Пояснение 1 выше.

Далее в основном SELECT-запросе выполняется подзапрос result3. Здесь как раз формируются два суперстолбца. В первый попадают все номера битв из Battles. Во второй попадают те номера, которые нашлись с помощью оконной функции LEAD(). Как она работает: в результирующей таблице result2 выполняется группировка по номеру записи в группе – num\_in\_group, при этом в каждой такой группе записи сортируются по номеру битвы. Получается по 3 группы, по 2 записи в каждой. В каждой из 3 групп LEAD() для каждой из этих 2 записей возвращает следующую запись (если этой записи нет, то возвращается NULL):



Теперь осталось убрать из первого суперстолбца те записи, которые попали во второй суперстолбец – в данном случае это номера битв 4, 5, 6. Для этого после ключевого слова WHERE ставим фильтр, что номера в rn\_1 должны принадлежать списку номеров битв из первой группы (gr=1).

Получаем тот же финальный результат.

# **Математические задачи**

## **Задание 1**

На барабане всего 14 секторов.

- количество всех вариантов, в каких из 14 секторов может остановиться волчок при 6 бросаниях.

Тогда .

Нужная нам вероятность: .

Так как раундов всего 6 и волчок должен попасть за эти 6 раундов на определенные 6 секторов, то это означает, что в эти раунды он не должен попадать на другие секторы барабана. То есть волчок может попасть только на секторы 1-6, а все остальные 7 секторов с 7 по 0 включительно могут не учитываться. Так как эти 7 секторов нам не важны, то их можно принять за один сектор и решать аналогичную задачу для 7 секторов, из которых нам нужны только 6.

Рассмотрим варианты выпадения всех 6 раундов на эти 7 секторов. В данном случае . Вероятность выпадения 6 конкретных секторов из 7 возможных равна вероятности невыпадения седьмого сектора – то есть .

В то же время . Тогда . Отсюда .

Тогда ответ задачи: .

## **Задание 2**

Пусть – цена акций данного типа.

Выборка из 16 дней: .

По условию задачи эта случайная величина подчиняется нормальному закону распределения: , где и – параметры распределения.

По свойствам нормального распределения известно, что равно математическому ожиданию по заданной выборке. Точечная оценка для вычисляется как среднее. Из условия задачи известно, что средняя доходность составляет 8% от начальной цены акций. Тогда точечная оценка :

.

Из условия задачи среднее квадратическое отклонение нашей случайной величины равно 4% от цены акций. То есть .

По свойствам нормального распределения параметр равен среднему квадратическому отклонению, тогда: .

Так как искомая выборка подчиняется нормальному закону распределения, то ее выборочное среднее также распределено нормально:

.

В таком случае доверительный интервал для математического ожидания искомой случайной величины:

,

где – уровень доверия, а – уровень значимости;

– квантиль нормального распределения уровня .

99%-ый доверительный интервал означает, что , в таком случае уровень значимости . В таком случае: . Величина квантиля находится по таблице. .

Тогда наш итоговый доверительный интервал выглядит таким образом:

,

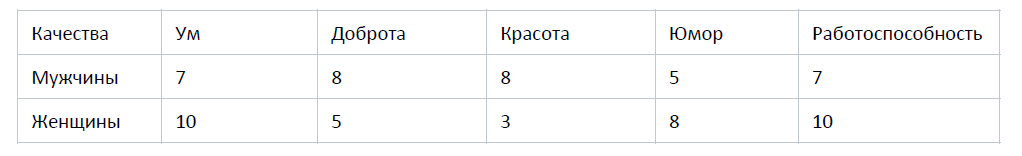
,

.

Ответ: , где – исходная цена акций.

## **Задание 3**

Теснота связи характеризуется с помощью специального показателя – коэффициента корреляции.



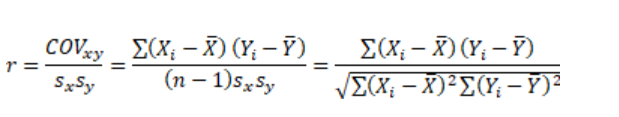
Пусть ответы мужчин представляют случайную величину X, ответы женщин – Y.

Найдем выборочные средние:

,

.

Коэффициент корреляции рассчитывается по формуле:

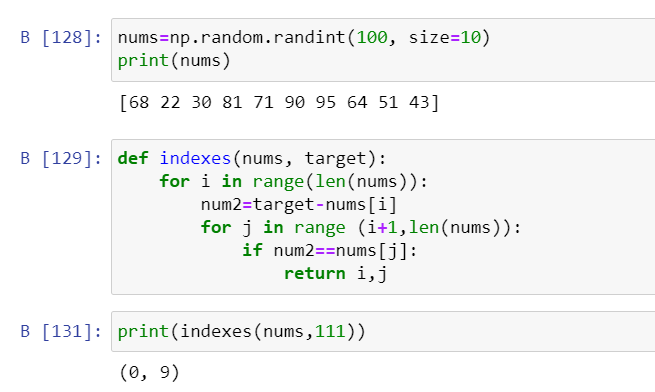


Корреляция получилась не полной отрицательной, значит взаимосвязь между величинами обратная, но не идеально линейная.

# **Задачи на основы программирования**

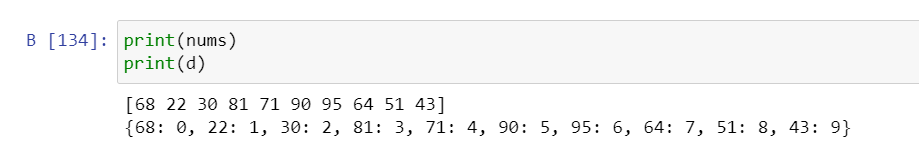
Первый способ найти пару чисел из nums, дающую в сумме число target, - просто идти по всему массиву и проверять все комбинации чисел. Такой алгоритм имеет оценку сложности O(n2), так как понадобится два вложенных цикла, что не является оптимальным решением, но все равно рассмотрим этот способ.

Во внутреннем цикле можно сократить начальный индекс, так как по условию задачи число не может суммироваться с самим собой, что отметает возможность существования пары (j, i), где j=i. А пары (j, i), где j<i проверяются на каждом предыдущем шаге, поэтому для каждого следующего шага их можно опустить:



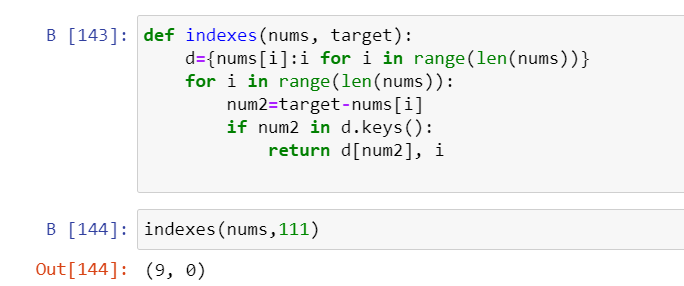
Второй способ – использование хэш-таблицы. Этот способ самый эффективный по времени, так как его временная оценка - O(n).

Создается хэш-таблица, где каждому элементу num из nums ставится в соответствие его индекс в исходном массиве nums. Для этого в функции создается словарь d, где ключом будет являться наше число, а значением – индекс, под которым хранится это число в массиве nums. Ниже указаны для сравнения массив и полученная по нему хэш-таблица (словарь d):

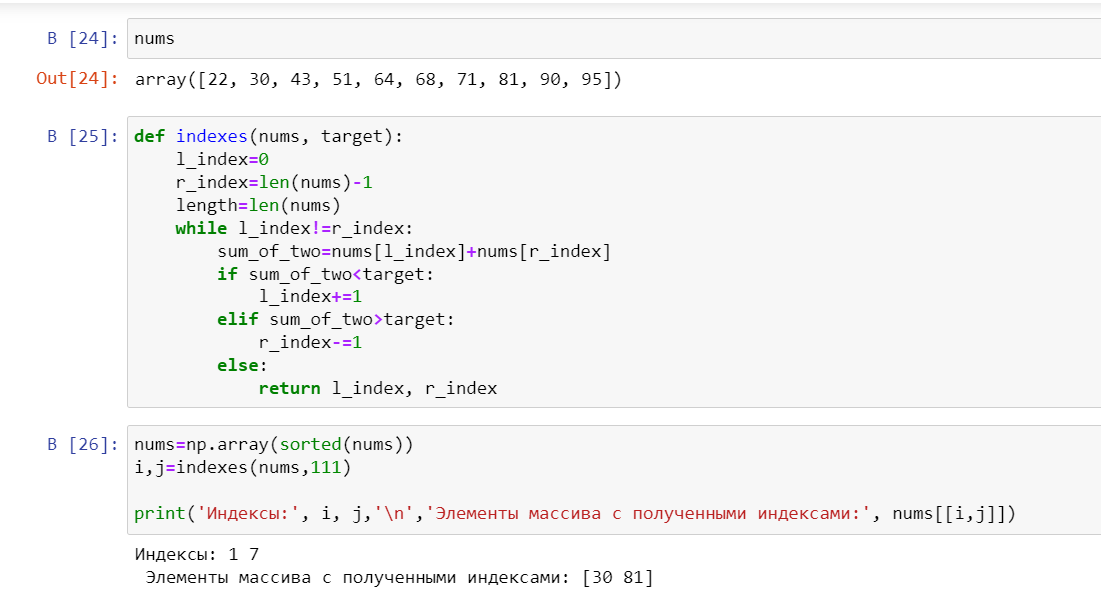


Дальше в цикле for проверяется, существует ли в этой хэш-таблице элемент (target-num). Если элемент (target-num) существует, то выводятся индексы данной пары элементов.

Результат:



И третий способ, который тоже имеет сложность O(n):



Функция indexes получает на вход отсортированный массив.

Здесь в функции indexes создаются два «указателя» на наш массив – l\_index указывает на первый элемент массива, а r\_index – на последний. Пока верно условие цикла, будем смещать эти «указатели» навстречу друг другу, пока не получим нужную сумму target. Если сумма двух элементов массива меньше заданного значения target, значит, надо двигать левый «указатель» вправо к большим числам, так как массив отсортирован по возрастанию. А если сумма двух элементов больше target, то нужно смещать влево правый «указатель», чтобы он указывал на меньшее число и сумма уменьшилась.