Шелудько А.А. ИТ2202

Вариант 5

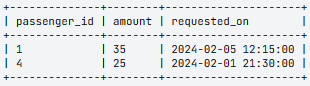
Задача 1

Напишите запрос, который извлекает из предложенной базы данных всю информацию о третьей по счету поездке каждого пассажира.

**Примечание 1.**Гарантируется, что хотя бы один из пассажиров совершил минимум три поездки.

**Примечание 2.**Записи в результирующей таблице могут быть расположены в произвольном порядке.

Результат:



Скрипт:

DROP TABLE IF EXISTS Rides; CREATE TABLE Rides ( passenger\_id INT, amount INT, requested\_on DATETIME ); INSERT INTO Rides (passenger\_id, amount, requested\_on) VALUES (1, 30, '2024-01-01 15:45:00'), (2, 25, '2024-02-12 07:00:00'), (3, 15, '2024-01-13 10:30:00'), (1, 15, '2024-02-08 18:15:00'), (1, 35, '2024-02-05 12:15:00'), (4, 70, '2024-01-20 11:55:00'), (4, 110, '2024-01-01 12:40:00'), (4, 25, '2024-02-01 21:30:00'), (3, 80, '2024-01-10 23:00:00'), (1, 10, '2024-01-02 18:10:00');

Решение: WITH RankedRides AS (

SELECT \*,

ROW\_NUMBER() OVER (PARTITION BY passenger\_id ORDER BY requested\_on) AS rn

FROM Rides

)

SELECT passenger\_id, amount, requested\_on

FROM RankedRides

WHERE rn = 3;

Задача 2

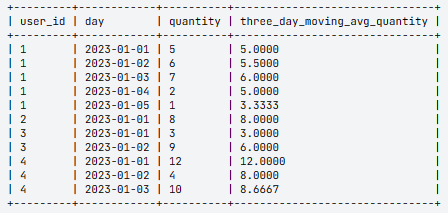
Напишите запрос, который извлекает из предложенной базы данных всю информацию о количестве постов, опубликованных каждым пользователем в каждый из дней, а также указывает для каждого дня среднее количество опубликованных пользователем постов, учитывая лишь текущий день и два предыдущих.

Поле со средним количеством постов, опубликованных пользователем, должно иметь псевдоним three\_day\_moving\_avg\_quantity.

**Примечание 1.**Если для текущей даты нет предпредыдущей даты, то вычисление среднего значения должно выполняться только на основе текущей и предыдущей дат. Если для текущей даты нет обеих предыдущих дат, то вычисление среднего значения должно выполняться только на основе текущей даты.

**Примечание 2.**Записи в результирующей таблице могут быть расположены в произвольном порядке.

Результат:



Скрипт:

DROP TABLE IF EXISTS Posts; CREATE TABLE Posts ( user\_id INT, day DATE, quantity INT ); INSERT INTO Posts (user\_id, day, quantity) VALUES (1, '2023-01-01', 5), (2, '2023-01-01', 8), (3, '2023-01-01', 3), (4, '2023-01-01', 12), (1, '2023-01-02', 6), (3, '2023-01-02', 9), (4, '2023-01-02', 4), (1, '2023-01-03', 7), (4, '2023-01-03', 10), (1, '2023-01-04', 2), (1, '2023-01-05', 1);

Решение: WITH PostsData AS (

SELECT

user\_id,

day,

quantity,

LAG(quantity, 1) OVER (PARTITION BY user\_id ORDER BY day) AS prev\_day\_quantity,

LAG(quantity, 2) OVER (PARTITION BY user\_id ORDER BY day) AS prev\_prev\_day\_quantity

FROM Posts

)

SELECT

user\_id,

day,

quantity,

(quantity + COALESCE(prev\_day\_quantity, 0) + COALESCE(prev\_prev\_day\_quantity, 0)) /

(CASE WHEN prev\_prev\_day\_quantity IS NOT NULL THEN 3

WHEN prev\_day\_quantity IS NOT NULL THEN 2

ELSE 1 END) AS three\_day\_moving\_avg\_quantity

FROM PostsData

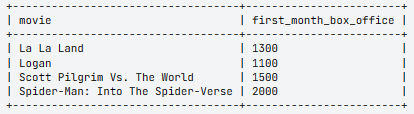
ORDER BY user\_id, day;

Задача 3

Напишите запрос, который определяет кассовые сборы каждого фильма в первый месяц показа и отображает полученный результат в виде таблицы из двух полей:

* movie — название фильма
* first\_month\_box\_office — кассовые сборы фильма в первый месяц показа

Результат:



Скрипт:

DROP TABLE IF EXISTS BoxOffice; CREATE TABLE BoxOffice ( month INT, movie VARCHAR(40), amount INT ); INSERT INTO BoxOffice (month, movie, amount) VALUES (1, 'Scott Pilgrim Vs. The World', 1500), (2, 'Scott Pilgrim Vs. The World', 1400), (2, 'Logan', 1100), (3, 'Scott Pilgrim Vs. The World', 1700), (3, 'Logan', 1200), (4, 'Scott Pilgrim Vs. The World', 1600), (4, 'Spider-Man: Into The Spider-Verse', 2000), (4, 'La La Land', 1300), (5, 'Spider-Man: Into The Spider-Verse', 2100), (6, 'Spider-Man: Into The Spider-Verse', 1800);

Решение: WITH BoxOfficeRanked AS (

SELECT

month,

movie,

amount,

ROW\_NUMBER() OVER (PARTITION BY movie ORDER BY month) AS rn

FROM BoxOffice

)

SELECT

movie,

SUM(amount) AS first\_month\_box\_office

FROM BoxOfficeRanked

WHERE rn = 1

GROUP BY movie;

Задача 4

Напишите запрос, который определяет, в какой день каждый из покупателей совершил свой последний заказ, вычисляет количество совершенных заказов в этот день и отображает полученный результат в виде таблицы из трех полей:

* customer\_id — идентификатор покупателя
* last\_order\_day — день совершения последнего заказа
* orders\_count — количество совершенных заказов в этот день

Примечание 1. Секционирование или упорядочивание окна может выполняться не только на основе исходных значений полей окна, но и на основе возвращаемых значений различных функций:

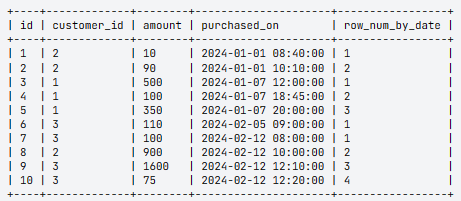
Результатом приведенного ниже запроса относительно предложенной в условии задачи таблицы Orders:

SELECT id, customer\_id, amount, purchased\_on,

ROW\_NUMBER() OVER (PARTITION BY DATE(purchased\_on)) AS row\_num\_by\_date

FROM Orders;

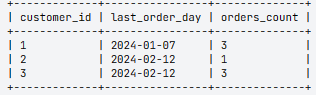
является:



Пример выше показывает, что, например, секционирование может быть выполнено с помощью функции DATE().

**Примечание 2.**Записи в результирующей таблице могут быть расположены в произвольном порядке.

Результат:



Скрипт:

DROP TABLE IF EXISTS Orders; CREATE TABLE Orders ( id INT PRIMARY KEY AUTO\_INCREMENT, customer\_id INT, amount INT, purchased\_on DATETIME ); INSERT INTO Orders (customer\_id, amount, purchased\_on) VALUES (2, 10, '2024-01-01 08:40:00'), (2, 90, '2024-01-01 10:10:00'), (1, 500, '2024-01-07 12:00:00'), (1, 100, '2024-01-07 18:45:00'), (1, 350, '2024-01-07 20:00:00'), (3, 110, '2024-02-05 09:00:00'), (3, 100, '2024-02-12 08:00:00'), (2, 900, '2024-02-12 10:00:00'), (3, 1600, '2024-02-12 12:10:00'), (3, 75, '2024-02-12 12:20:00');

Решение: WITH LastOrderDates AS (

SELECT

customer\_id,

MAX(purchased\_on) OVER (PARTITION BY customer\_id) AS last\_order\_day,

COUNT(\*) OVER (PARTITION BY customer\_id, DATE(purchased\_on)) AS orders\_count,

ROW\_NUMBER() OVER (PARTITION BY customer\_id ORDER BY purchased\_on DESC) AS rn

FROM Orders

)

SELECT

customer\_id,

last\_order\_day,

orders\_count

FROM LastOrderDates

WHERE rn = 1;

Задача 5

Иногда в результате различных сбоев банковские платежи случайно совершаются повторно, что приводит к двойному списанию денежных средств с банковской карты. Банк HSBC считает, что платеж по ошибке совершен повторно в том случае, если в течение 10 минут до него был совершен платеж от того же клиента, с той же карты и на ту же сумму. Например, платеж с идентификатором 2 считается совершенным повторно случайно, поскольку 6 минут назад был совершен ровно такой же платеж. Аналогичное справедливо для платежей с идентификаторами 5 и 7.

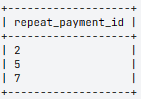
Напишите запрос, который извлекает из предложенной базы данных идентификаторы платежей, по ошибке совершенных повторно.

Поле с идентификатором случайно совершенного повторно платежа должно иметь псевдоним repeat\_payment\_id.

Записи в результирующей таблице должны быть расположены в порядке возрастания значения поля repeat\_payment\_id.

**Примечание.**Гарантируется, что платеж может быть по ошибке совершен повторно только один раз.

**Результат:**



Скрипт:

DROP TABLE IF EXISTS Payments; CREATE TABLE Payments ( id INT PRIMARY KEY AUTO\_INCREMENT, user\_id INT, card\_id INT, amount INT, completed\_on DATETIME ); INSERT INTO Payments (user\_id, card\_id, amount, completed\_on) VALUES (1, 1, 100, '2024-01-01 12:00:00'), (1, 1, 100, '2024-01-01 12:06:00'), (2, 2, 250, '2024-01-02 18:00:00'), (3, 1, 50, '2024-01-02 18:05:00'), (2, 2, 250, '2024-01-02 18:08:00'), (3, 1, 10, '2024-01-03 10:00:00'), (3, 1, 10, '2024-01-03 10:10:00'), (1, 2, 80, '2024-01-03 10:00:00'), (1, 2, 80, '2024-01-03 10:10:01'), (4, 1, 200, '2024-01-03 13:00:00');

Решение: WITH RankedPayments AS (

SELECT

id,

user\_id,

card\_id,

amount,

completed\_on,

LAG(completed\_on) OVER (PARTITION BY user\_id, card\_id, amount ORDER BY completed\_on) AS prev\_completed\_on

FROM Payments

)

SELECT

p.id AS repeat\_payment\_id

FROM RankedPayments p

WHERE TIMESTAMPDIFF(MINUTE, p.prev\_completed\_on, p.completed\_on) <= 10

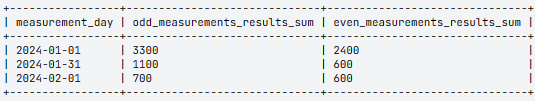
ORDER BY p.id;

Задача 6

Напишите запрос, который разбивает результаты измерений на группы в зависимости от дня, в который они были выполнены, вычисляет в рамках каждой группы сумму результатов нечетных и четных по счету измерений и отображает полученный результат в виде таблицы из трех полей:

1. measurement\_day — день, в который были выполнены измерения
2. odd\_measurements\_results\_sum — сумма результатов нечетных измерений в этот день (результат первого измерения, результат третьего измерения, и так далее)
3. even\_measurements\_results\_sum — сумма результатов четных измерений в этот день (результат второго измерения, результат четвертого измерения, и так далее)

Результат:



Скрипт:

DROP TABLE IF EXISTS Measurements; CREATE TABLE Measurements ( id INT PRIMARY KEY AUTO\_INCREMENT, result INT, received\_on DATETIME ); INSERT INTO Measurements (result, received\_on) VALUES (1100, '2024-01-01 12:00:00'), (1000, '2024-01-01 13:00:00'), (1300, '2024-01-01 14:00:00'), (1200, '2024-01-01 15:00:00'), (1100, '2024-01-01 07:00:00'), (450, '2024-01-31 08:00:00'), (600, '2024-01-31 09:00:00'), (650, '2024-01-31 10:00:00'), (700, '2024-02-01 07:00:00'), (600, '2024-02-01 18:00:00');  
WITH NumberedMeasurements AS (

SELECT

\*,

ROW\_NUMBER() OVER (PARTITION BY DATE(received\_on) ORDER BY received\_on) AS rn

FROM Measurements

),

GroupedMeasurements AS (

SELECT

DATE(received\_on) AS measurement\_day,

SUM(CASE WHEN rn % 2 = 1 THEN result ELSE 0 END) AS odd\_measurements\_results\_sum,

SUM(CASE WHEN rn % 2 = 0 THEN result ELSE 0 END) AS even\_measurements\_results\_sum

FROM NumberedMeasurements

GROUP BY DATE(received\_on)

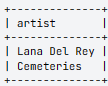
)

SELECT \* FROM GroupedMeasurements;

Задача 7

Напишите запрос, который извлекает из предложенной базы данных имена исполнителей, чьи песни чаще всего занимают первое или второе место в ежедневных чартах.

Результат:



Скрипт:

DROP TABLE IF EXISTS Songs; CREATE TABLE Songs ( id INT PRIMARY KEY AUTO\_INCREMENT, artist VARCHAR(50), title VARCHAR(50) ); INSERT INTO Songs (artist, title) VALUES ('Cemeteries', 'Empty Camps'), ('Cemeteries', 'Can You Hear Them Sing?'), ('CHVRCHES', 'Never Say Die'), ('CHVRCHES', 'My Enemy'), ('Daughter', 'Home'), ('Daughter', 'No Care'), ('Wintersleep', 'Metropolis'), ('Wintersleep', 'Wind'), ('Lana Del Rey', 'Diet Mountain Dew'), ('Lana Del Rey', 'Young And Beautiful'); DROP TABLE IF EXISTS SongCharts; CREATE TABLE SongCharts ( day DATE, place INT, song\_id INT ); INSERT INTO SongCharts (day, place, song\_id) VALUES ('2024-01-01', 1, 9), ('2024-01-01', 2, 7), ('2024-01-01', 3, 8), ('2024-01-02', 1, 9), ('2024-01-02', 2, 3), ('2024-01-02', 3, 8), ('2024-01-03', 1, 1), ('2024-01-03', 2, 10), ('2024-01-03', 3, 6), ('2024-01-04', 1, 6), ('2024-01-04', 2, 1), ('2024-01-04', 3, 9), ('2024-01-05', 1, 2), ('2024-01-05', 2, 7), ('2024-01-05', 3, 5);

WITH RankedSongs AS (

SELECT

s.artist,

sc.day,

sc.place,

ROW\_NUMBER() OVER (PARTITION BY sc.day ORDER BY sc.place) AS rn

FROM Songs s

JOIN SongCharts sc ON s.id = sc.song\_id

)

, TopArtists AS (

SELECT artist, COUNT(\*) AS top\_placements\_count

FROM RankedSongs

WHERE rn <= 2

GROUP BY artist

ORDER BY top\_placements\_count DESC

LIMIT 2

)

SELECT artist

FROM TopArtists;

Задача 8

Напишите запрос, определяющий в каждой категории два товара, продажи которых принесли наибольшую суммарную прибыль, и отображающий полученный результат в виде таблицы из трех полей:

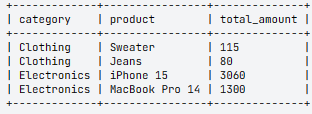
* category — категория товара
* product — название товара
* total\_amount — суммарная прибыль, которую принесли продажи товара

Если в категории находится более двух товаров, принесших наибольшую суммарную прибыль, в результирующую таблицу должны быть добавлены те, чье название меньше в лексикографическом сравнении.

**Примечание 1.**Гарантируется, что в каждой категории было продано минимум два товара.

**Примечание 2.**Записи в результирующей таблице могут быть расположены в произвольном порядке.

Результат:



Скрипт:

DROP TABLE IF EXISTS Orders; CREATE TABLE Orders ( id INT PRIMARY KEY AUTO\_INCREMENT, category VARCHAR(50), product VARCHAR(50), amount INT ); INSERT INTO Orders (category, product, amount) VALUES ('Electronics', 'iPhone 15', 1000), ('Clothing', 'Jeans', 80), ('Electronics', 'iPhone 15', 1010), ('Electronics', 'iPad Air', 350), ('Clothing', 'Sweater', 55), ('Electronics', 'iPhone 15', 1050), ('Electronics', 'MacBook Pro 14', 1300), ('Clothing', 'Sweater', 60), ('Electronics', 'iPad Air', 300), ('Electronics', 'iPad Air', 300);

WITH RankedProducts AS (

SELECT

category,

product,

SUM(amount) AS total\_amount,

ROW\_NUMBER() OVER (PARTITION BY category ORDER BY SUM(amount) DESC, product) AS rn

FROM Orders

GROUP BY category, product

)

SELECT category, product, total\_amount

FROM RankedProducts

WHERE rn <= 2;

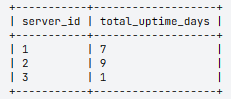
Задача 9

Каждый сервер компании может включаться и выключаться несколько раз. Если в результате всех включений и выключений суммарная наработка сервера находится в диапазоне от 24 часов (включительно) до 48 часов (не включительно), то компания Google считает, что сервер проработал 1 день, если в диапазоне от 48 часов (включительно) до 72 часов (не включительно) — 2 дня, и так далее.

Напишите запрос, определяющий количество дней, которое проработал каждый сервер, и отображающий полученный результат в виде таблицы из двух полей:

* server\_id — идентификатор сервера
* total\_uptime\_days — количество проработанных дней

Результат:



Скрипт:

DROP TABLE IF EXISTS ServerUtilization; CREATE TABLE ServerUtilization ( server\_id INT, session\_status VARCHAR(5), status\_time DATETIME ); INSERT INTO ServerUtilization (server\_id, session\_status, status\_time) VALUES (1, 'start', '2024-01-01 08:00:00'), (2, 'start', '2024-01-01 08:00:00'), (3, 'start', '2024-01-01 08:00:00'), (3, 'stop', '2024-01-02 01:00:00'), (1, 'stop', '2024-01-05 07:00:00'), (2, 'stop', '2024-01-10 09:00:00'), (1, 'start', '2024-01-11 10:00:00'), (3, 'start', '2024-01-11 10:00:00'), (3, 'stop', '2024-01-11 17:00:00'), (1, 'stop', '2024-01-15 10:00:00');

WITH ServerStatus AS (

SELECT

server\_id,

session\_status,

status\_time,

LEAD(status\_time) OVER (PARTITION BY server\_id ORDER BY status\_time) AS next\_status\_time

FROM ServerUtilization

),

ServerUptime AS (

SELECT

server\_id,

SUM(IF(session\_status = 'start', CAST(next\_status\_time AS DATE) - CAST(status\_time AS DATE), 0)) AS total\_uptime\_days

FROM ServerStatus

GROUP BY server\_id

)

SELECT

server\_id,

total\_uptime\_days

FROM ServerUptime;

Задача 10

Напишите запрос, который группирует одинаковые товары в зависимости от месяца, в который они были проданы, вычисляет в рамках каждой группы суммарную прибыль, которую принесли продажи товара, и отображает полученный результат в виде таблицы из четырех полей:

* month — полное название месяца на английском
* product — название товара
* total\_amount — суммарная прибыль, которую принесли продажи товара в этом месяце
* nearest\_prev\_month\_total\_amount — суммарная прибыль, которую принесли продажи товара в ближайшем предыдущем месяце (например, в предпредыдущем, если в предыдущем месяце товар не был продан ни разу). Если товар не был продан ни разу ни в одном из предыдущих месяцев, поле должно содержать значение NULL

Результат запроса

Скрипт для создания таблицы Orders

Примечание 1. Во время работы с группами необходимо помнить о том, что у группы можно получить лишь значения полей группировки, а также значения, полученные с помощью агрегатных функций.

Результатом приведенного ниже запроса относительно предложенной в условии задачи таблицы Orders:

SELECT product,

COUNT(\*) AS was\_ordered,

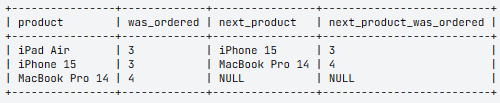
LEAD(product) OVER (ORDER BY product) AS next\_product,

LEAD(COUNT(\*)) OVER (ORDER BY product) AS next\_product\_was\_ordered

FROM Orders

GROUP BY product

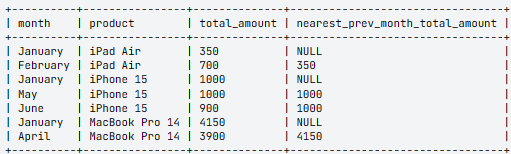
является:



Пример выше показывает, что при использовании оконных функций совместно с группами, все ограничения групп сохраняются. Так, с помощью функции LEAD() у следующей группы можно получить лишь значение поля группировки — product, либо результаты различных агрегатных вычислений, в данном случае это количество записей в группе.

**Примечание 2.**Записи в результирующей таблице могут быть расположены в произвольном порядке.

Результат



DROP TABLE IF EXISTS Orders; CREATE TABLE Orders ( id INT PRIMARY KEY AUTO\_INCREMENT, product VARCHAR(50), amount INT, purchased\_on DATE ); INSERT INTO Orders (product, amount, purchased\_on) VALUES ('iPad Air', 350, '2024-01-01'), ('MacBook Pro 14', 2100, '2024-01-01'), ('iPhone 15', 1000, '2024-01-12'), ('MacBook Pro 14', 2050, '2024-01-20'), ('iPad Air', 300, '2024-02-10'), ('iPad Air', 400, '2024-02-22'), ('MacBook Pro 14', 2000, '2024-04-20'), ('MacBook Pro 14', 1900, '2024-04-20'), ('iPhone 15', 1000, '2024-05-28'), ('iPhone 15', 900, '2024-06-25');

WITH MonthlySales AS (

SELECT

product,

DATE\_FORMAT(purchased\_on, '%Y-%m-01') AS month,

SUM(amount) AS total\_amount

FROM

Orders

GROUP BY

product, month

),

MonthlySalesWithPrev AS (

SELECT

product,

month,

total\_amount,

LAG(total\_amount) OVER (PARTITION BY product ORDER BY month) AS nearest\_prev\_month\_total\_amount

FROM

MonthlySales

)

SELECT

DATE\_FORMAT(month, '%M') AS month,

product,

total\_amount,

nearest\_prev\_month\_total\_amount

FROM

MonthlySalesWithPrev

ORDER BY

product, month;