Шелудько А.А. ИТ2202. Оконные функции.

13.1.1

В таблице ниже содержатся данные по продажам за каждый месяц в течение двух лет. Напишите SQL запрос, который выведет три колонки: месяц, год и общую сумму всех продаж.

Последняя колонка должна называться **total\_revenue**.

Итоговые данные нужно отсортировать по году и месяцу.

SELECT month,year,SUM(revenue) OVER () AS total\_revenue FROM revenues

ORDER BY year, month;

13.1.2

В таблице ниже содержатся данные по продажам за каждый месяц в течение двух лет. Напишите SQL запрос, который выведет три колонки: месяц, год и сумму продаж за год.

Последняя колонка должна называться **year\_revenue**.

Итоговые данные нужно отсортировать по году и месяцу.

SELECT month, year, SUM(revenue) OVER(PARTITION BY year) as year\_revenue FROM revenues

ORDER BY year, month;

13.1.3

В таблице ниже содержатся данные по продажам за каждый месяц в течение двух лет. Напишите SQL запрос, который выведет три колонки: месяц, год и процентный вклад каждого месяца в итоги года.

Последняя колонка должна называться **month\_percent**и данные в ней нужно округлить до 1 знака после десятичной точки.

Итоговые данные нужно отсортировать по году и месяцу.

SELECT month, year,round(revenue \* 100 / SUM(revenue) OVER(PARTITION BY year), 1) as month\_percent from revenues

ORDER BY year, month

13.1.4

Таблица ниже содержит список городов с указанием страны, в которой они находятся, а также населения.

Напишите SQL запрос, который выводит все столбцы таблицы + колонку с общим населением страны, в которой находится город. Колонка с общим населением должна называться **country\_population**.

Итоговые данные отсортируйте по населению страны и по населению города в прямом порядке.

SELECT \*, SUM(population) OVER(PARTITION BY country) as country\_population FROM cities

ORDER BY country\_population, population

13.1.5

В таблице ниже представлены товары в магазине.  
Напишите SQL-запрос, который выведет все столбцы исходной таблицы + дополнительный столбец с общей стоимостью всех товаров с учетом их количества. Последний столбец назовите **total**.

SELECT id, name, SUM(count \* price) OVER() AS total

FROM products;

13.1.6

В прошлой задаче мы посчитали общую стоимость всех товаров в магазине.  
Теперь давайте вычислим долю (процент) каждого товара в общей стоимости.

Напишите SQL-запрос, который выведет все столбцы таблицы, плюс еще один столбец с процентом, который занимает товар в общей стоимости. Не забывайте, что каждый товар помимо цены имеет и количество.

Последний столбец назовите **percent** и округлите в нем данные до одного знака после десятичной точки.

Итоговые данные отсортируйте по проценту в обратном порядке.  
Товары с одинаковой долей дополнительно отсортируйте по id.

Данный запрос позволит узнать в каких товарах «заморожена» большая часть наших активов.

SELECT \*,round((products.count\*products.price) \* 100 / SUM(products.count\*products.price) OVER(), 1) as percent

FROM products

ORDER BY percent DESC, id

13.1.7

В одной из прошлых задач мы добавили в таблицу население каждой страны. Теперь давайте посчитаем долю (процент) населения каждого города в общем населении страны.

Напишите SQL запрос, который выводит все столбцы таблицы +2 колонки: одну с общим населением страны, в котором находится город, а вторую с процентом, который текущий город занимает в стране.

Колонка с общим населением должна называться **country\_population**, а с процентом **percent**. Проценты нужно вывести с округлением до двух знаков после десятичной точки.

Итоговые данные отсортируйте

SELECT \*, SUM(population) OVER(PARTITION BY country) as

country\_population,

round(population \* 100 / SUM(population) OVER(PARTITION BY country),2) as

percent from cities

order by country\_population, population;

13.1.8

В прошлой задаче мы выяснили важность городов внутри страны. Теперь давайте вычислим долю (процент) населения каждого города в мировом масштабе.

Напишите SQL запрос, который выводит все столбцы таблицы +колонку с процентом, который текущий город занимает в мире.

Колонка с процентом должна называться **world\_percent**. Проценты нужно вывести с округлением до двух знаков после десятичной точки.

Итоговые данные отсортируйте по **world\_percent** в обратном порядке, города с одинаковым значением **world\_percent**дополнительно отсортируйте по id.

SELECT \*, ROUND( population \* 100 / SUM(population) OVER(), 2) as world\_percent

FROM cities

ORDER BY world\_percent DESC, id

13.2.1

В таблице **films** содержится информация о фильмах: id, рейтинг, год выпуска и количество голосов. Напишите SQL-запрос, который выведет три столбца: место фильма в рейтинге (колонка **place**), название фильма и его рейтинг.

Учитывайте, что места нужно расставить от 1 до 10, где 1 будет у самого популярного фильма.

Популярные фильмы должны стоять вверху таблицы.

SELECT RANK() OVER (ORDER BY rating DESC) AS place,NAME, rating

FROM films;

13.2.2

Таблица ниже предназначена для создания связи многие-ко-многим между заказами и товарами в заказах. В таблице нет первичного ключа и допускается дублирование строк, так как в одном заказе может быть несколько одинаковых товаров.

Из-за отсутствия первичного ключа нет возможности обратиться к какой-то конкретной строке, так как все они являются равнозначными. И часто для уникализации строк в подобных таблицах используют функцию **ROW\_NUMBER**.

Напишите SQL-запрос, который выведет все строки данной таблицы с нумерацией по порядку, начиная с единицы. Столбец с номерами должен называться **line\_num** и идти первым, после него должны стоять **order\_id** и **product\_id**. Данные в таблице отсортируйте по номеру заказа, а потом по номеру товара.

SELECT row\_number() OVER(ORDER BY order\_id, product\_id) as line\_num, order\_id, product\_id

FROM orders\_products

13.2.3

Сайт выводит товары по 5 штук. Выберите из таблицы **products** товары, которые пользователи увидят на 3 странице каталога при сортировке в порядке возрастания цены (**price**).

В итоговой таблице нужно вывести 4 столбца: **num** - порядковый номер товара, name, count и price.

Обратите внимание, что порядковый номер товара нужно выводить с учетом того, что мы выводим данные на третьей странице и перед ней было еще 2 страницы, которые содержали 10 товаров.

SELECT row\_number() OVER(ORDER BY price) as num, name, count, price FROM products LIMIT 10, 5

13.3.1

В таблице **films** содержится информация о фильмах: id, рейтинг, год выпуска, количество голосов и жанр. Напишите SQL-запрос, который выведет четыре столбца: жанр, место фильма в рейтинге в пределах своего жанра (колонка **genre\_place**), рейтинг фильма и его название.

Учитывайте, что места нужно расставить от 1 до N, где 1 будет у самого популярного фильма.

Итоговые данные нужно сперва отсортировать по жанру, а потом по месту фильма.

select genre, row\_number() over (partition by genre order by rating desc) as genre\_place, rating, name from films

order by genre, genre\_place

13.3.2

В таблице ниже представлены результаты соревнований по программированию. Выведите все столбцы + добавьте колонку **place** с местами, которые заняли участники. Первое место получает программист, который набрал больше всех очков.

Итоговые данные должны быть отсортированы по id.

ALTER TABLE results

ADD COLUMN place INT;

SELECT id, last\_name, first\_name, points,

RANK() OVER(ORDER BY points DESC) as place

FROM results

ORDER BY id;

13.3.3

В прошлой задаче мы проставляли спортсменам места, однако таблица оказалась некорректной.

У некоторых участников изменились набранные очки, а также появилось время выполнения задания.

Выведите все столбцы + добавьте колонку **place** с местами, которые заняли участники. Первое место получает программист, который набрал больше всех очков. Если несколько участников набрали одинаковое количество баллов, то лучшим среди них считается тот, кто выполнил задания за меньшее время.

Итоговые данные выведите в соответствии с занятыми местами - первое место сверху.

ALTER TABLE results

ADD COLUMN place INT;

SELECT id, last\_name, first\_name, points, time,

RANK() OVER(ORDER BY points DESC, time ASC) as place

FROM results

ORDER BY place;

13.3.4

Нужно получить по два самых популярных фильма каждого жанра, чтобы составить разнообразную подборку. Некоторые жанры имеют только один фильм — это не страшно, пусть так и будет.

Напишите SQL-запрос, который выведет по два самых популярных фильма каждого жанра. Итоговая таблица должна содержать 3 столбца: название, рейтинг и жанр. Финальные данные нужно отсортировать по рейтингу в обратном порядке.

PS. Для решения данной задачи нужно использовать вложенные запросы.

select name, rating, genre from (select row\_number() over (partition by genre order by rating desc) as rn, name, rating, genre

from films

) as f

where rn <=2

order by rating desc

13.3.5

Продолжаем работать с популярными фильмами и теперь пришло время выяснить какие фильмы были лучшими в 80-е, 90-е и тд.

Напишите SQL-запрос, который выведет 3 столбца: декаду (**decade**), позицию фильма (**place**) и его название.

Декаду нужно выводить в формате **ГГГ0**, где первые три цифры отвечают за год, а последняя - всегда ноль. Например, для фильма, который вышел в 1999 году декадой будет 1990.

Для вычисления декады, нужно будет воспользоваться небольшими арифметическими расчетами и округлением. Финальные данные нужно отсортировать по декаде и по позиции.

SELECT TRUNCATE(year, -1) AS decade,

ROW\_NUMBER() OVER (PARTITION BY TRUNCATE(year, -1) ORDER BY rating DESC) AS place,

name

FROM films

ORDER BY decade, place;

13.4.1

В таблицах ниже находятся клиенты и заказы, которые они сделали. Наша задача найти среди них VIP клиентов. Статус VIP пользователь получает если он сделал и оплатил (status=success) **2 и более заказов**.

Напишите SQL запрос, для вывода всех VIP клиентов. Итоговая таблица должна содержать id пользователя, его имя, фамилию и дату, когда он стал VIP клиентом. Дату выведите в колонке **vip\_date** в формате ДД.ММ.ГГГГ. Идентификатор пользователя нужно вывести в колонке с именем **id**.

SELECT

id, first\_name, last\_name, DATE\_FORMAT(date, '%d.%m.%Y') as vip\_date

FROM (

SELECT

users.id, users.first\_name, users.last\_name, sales.date,

ROW\_NUMBER() OVER (PARTITION BY users.id ORDER BY date) as sales\_num

FROM users

JOIN sales ON users.id = sales.user\_id

WHERE sales.status = 'success'

) t

WHERE sales\_num = 2

ORDER BY id

13.5.1

Таблица ниже содержит информацию об instagram-блогерах: id, название аккаунта, количество подписчиков, среднее количество лайков и среднее количество комментариев на один пост.

Напишите SQL-запрос для подбора блогеров. В первую очередь должны идти блогеры с максимальным количеством подписчиков, далее, если количество подписчиков равно, то нужно брать блогеров с максимальным количеством лайков, а затем с максимальным количеством комментариев.

В итоговой таблице выведите следующие поля: **num** — номер по порядку начиная с 1, аккаунт блогера, количество подписчиков, число лайков, число комментариев. Первыми должны идти блогеры с максимальным числом подписчиков.

SELECT

ROW\_NUMBER() OVER (ORDER BY subs DESC, avg\_likes DESC, avg\_comments DESC) as num,

blogger, subs, avg\_likes, avg\_comments

FROM bloggers

ORDER BY num

13.5.2

Таблица ниже содержит данные о публикациях блоггеров и включает в себя аккаунт блогера, номер поста, количество лайков и количество комментариев.

Напишите SQL-запрос, который пронумерует посты каждого блогера в порядке их популярности на основе лайков. Самые «залайканные» посты должны получить единицу и чем меньше лайков, тем больший номер у них должен быть. Для каждого блогера нужно вести свою отдельную нумерацию. Популярность постов выводите в колонке **post\_popularity**, которая должна стоять первой. После неё должны идти имя блогера, название поста и количество лайков.

Итоговые данные отсортируйте по имени блогера, а затем по популярности записей (самая популярная вверху).

SELECT ROW\_NUMBER() OVER(PARTITION BY blogger ORDER BY likes DESC) AS post\_popularity,blogger,post,likes

FROM bloggers\_posts

ORDER BY blogger, post\_popularity;

13.5.3

В прошлой задаче вы проранжировали посты по популярности. Теперь оставьте по одному самому популярному посту от каждого блогера и посчитайте его вклад в общую популярность этих отобранных постов.

Выведите в итоговой таблице 5 колонок: **blogger** — имя блогера, **post** — название поста, **likes** — количество лайков поста, **total\_likes** — общее количество лайков лучших постов, **percent** — процент лайков текущего поста относительно общего количества лайков. Процент следует вывести с двумя знаками после десятичной точки.

Итоговые данные отсортируйте по популярности постов: чем популярней — тем выше.

SELECT

blogger, post, likes, SUM(likes) OVER () as total\_likes,

ROUND(likes \* 100 / SUM(likes) OVER (), 2) as percent

FROM (

SELECT

ROW\_NUMBER() OVER(partition by blogger order by likes desc) as post\_popularity,

blogger, post, likes

FROM bloggers\_posts

ORDER BY blogger, post\_popularity

) pop

WHERE post\_popularity = 1

order by likes desc

13.5.4

В прошлой задаче вы вычисли процент лайков текущего поста относительно общего количества лайков, но общее количество рассчитывалось только на основе **трёх лучших постов**.

Внесите в ваш SQL-запрос изменения, чтобы он считал процент лайков каждого поста относительно **всех лайков** всех постов из таблицы **bloggers\_posts**. При этом в итоговой таблице также должно остаться три лучших поста, просто у них будет меньший процент, так как учитываются все лайки. Количество всех лайков (**total\_likes**) также будет другим.

Порядок столбцов и итоговая сортировка должны быть такими же, как и в прошлой задаче.

SELECT

blogger, post, likes, total\_likes, percent

FROM (

SELECT

num, blogger, post, likes,

SUM(likes) OVER () as total\_likes,

ROUND(likes \* 100 / SUM(likes) OVER (), 2) as percent

FROM (

SELECT

ROW\_NUMBER() OVER(partition by blogger order by likes desc) as num,

blogger, post, likes

FROM bloggers\_posts

) pop

) pop2

WHERE num = 1

ORDER BY likes DESC

13.6.1

В таблице ниже содержатся данные по доходам и расходам компании за январь 2022 года. Напишите SQL-запрос, который выведет все исходные столбцы + колонку **balance**, в которой будут отображаться остатки на счете после проведения операции.

Все операции проводились в соответствии с датой в колонке **date**. Если в одну дату было несколько операций, то они проводились в порядке следования **id**.

Итоговую таблицу отсортируйте по дате и id.

SELECT t.\*, SUM(t.money) OVER (ORDER BY t.date, t.id) AS balance FROM transactions t

ORDER BY t.date, t.id;

13.6.2

В прошлой задаче не учитывалось, что у компании изначально могли быть деньги на счетах.

Напишите SQL-запрос, который выведет все исходные столбцы таблицы **transactions** + колонку **balance**, в которой будут отображаться остатки на счете после проведения операции. Нужно учесть, что на **1 января 2022 года** на счетах компании было 10000 рублей, эти деньги также участвуют в расчете баланса.

Начальный баланс нужно также отразить в итоговой таблице с **id** равным 0, **item** равным «Начальный баланс» и суммой в 10000.

Итоговую таблицу отсортируйте по дате и id.

WITH ExtendedTransactions AS (SELECT 0 AS id,CAST('2022-01-01' AS DATE) AS date,'Начальный баланс' AS item,10000 AS money UNION ALL

SELECT id,date,item, money FROM transactions),

BalancedTransactions AS (SELECT e.id, e.date,e.item,e.money,SUM(e.money) OVER (ORDER BY e.date, e.id) AS balance FROM ExtendedTransactions e)

SELECT \* FROM BalancedTransactions

ORDER BY date, id;

13.6.3

В таблице ниже хранятся данные о доходах и расходах компании, разбитые по годам и месяцам. Изначально компания мало зарабатывала и много тратила, поэтому инвесторам приходилось вкладывать в неё свои деньги. На данный момент компания приносит прибыль и наша задача посчитать, окупились ли вложения инвесторов.

Напишите SQL-запрос, который выведет все строки и столбцы исходной таблицы + дополнительный столбец (**ror**), в котором будет рассчитываться прибыль нарастающим итогом. Прибыль — это разница между доходами и расходами компании.

SELECT year,month,income,outcome,

SUM(income - outcome) OVER (ORDER BY year, month) AS ror

FROM revenues

ORDER BY YEAR,month;

13.6.4

В прошлой задаче мы вычисляли окупаемость бизнеса. Инвесторам понравился результат и они решили открыть еще одну точку. На этот раз не на свои деньги, а на кредитные. Наша задача посчитать размер кредита, который нужно взять в банке.

Кредитных денег должно хватить на бесперебойную работу бизнеса. Если в какой-то из месяцев расходы превышают доходы, то мы берем эту разницу из заемных средств.

Максимальный долг, который может образоваться, можно посмотреть в колонке, которая выводит прибыль нарастающим итогом.

Напишите SQL-запрос, который выведет один столбец (**investment**), содержащий одну единственную строку — величину кредита, который мы должны взять, чтобы покрыть все потенциальные убытки. Размер кредита будет со знаком минус.

Учитывайте, что ноябрь 2021 был убыточным, однако этот убыток был получен уже после того как инвестиции окупились.

SELECT MIN(cumulative\_profit) AS investment

FROM (SELECT YEAR,month,income,outcome,SUM(income - outcome) OVER (ORDER BY year, month) AS cumulative\_profit FROM revenues) AS Subquery;

13.6.5

В прошлой задаче мы рассчитали размер банковского кредита, который необходим для открытия новой точки. Теперь надо вычислить срок окупаемости инвестиций. Инвестиции окупаются в тот момент, когда суммарные доходы бизнеса превышают суммарные расходы.

Напишите SQL-запрос, который выведет один столбец (months), содержащий одну единственную строку — количество месяцев, через которое бизнес окупит все вложенные в него средства.

SELECT MIN(months) AS months

FROM (SELECT year,month,SUM(income - outcome) OVER (ORDER BY year, month) AS cumulative\_profit,ROW\_NUMBER() OVER (ORDER BY year, month) AS months FROM revenues) AS Subquery

WHERE cumulative\_profit > 0

13.7.1

В таблице ниже содержатся данные по доходам индивидуального предпринимателя за 2021 год.

Наша задача подготовить эти данные для составления налоговой декларации. Особенность декларации в том, что данные в ней указываются **поквартально нарастающим итогом**.

Поможем предпринимателю с декларацией. Напишите SQL запрос, который выведет 4 столбца: **quarter** — номер квартала, **income** — суммарные поступления за этот квартал, **income\_acc** — поступления за квартал нарастающим итогом и **usn6** — величина налога нарастающим итогом. При системе налогообложения УСН 6% начисляется 6% налог на все доходы. Выведете налоги с двумя знаками после десятичной точки.

Данные в итоговой таблице должны быть отсортированы по кварталам.

select QUARTER(date) AS quarter,

SUM(income) AS income,

SUM(SUM(income)) OVER(ORDER BY QUARTER(date)) AS income\_acc,

ROUND(SUM(SUM(income)\*0.06) OVER(ORDER BY QUARTER(date)), 2) AS usn6

FROM transactions

GROUP BY QUARTER(date)

ORDER BY QUARTER(date)

13.7.2

Сгруппируйте данные в таблице **users** по полу и получите количество мужчин и женщин, а также их процентное соотношение к общему числу людей.

В первой колонке (**sex**) выведите пол, во второй (**members**) — количество , а в третьей (**percent**) — процент. Данные отсортируйте по последнему столбцу.

SELECT DISTINCT sex, COUNT(sex) OVER (PARTITION BY sex) AS members, ROUND((COUNT(sex) OVER (PARTITION BY sex) \* 100.0) / COUNT(\*) OVER(), 2) AS percent

FROM users

ORDER BY percent ASC;

13.7.3

Сгруппируйте данные в таблице **users** по возрасту и получите количество и процент клиентов каждого возраста.

В первом столбце (**age\_num**) выведете порядковый номер возраста, во втором (**age**) — сам возраст, в третьем (**clients**) — количество клиентов данного возраста, а в четвертом (**percent**) — процент клиентов данного возраста.

Данные отсортируйте по возрасту в возрастающем порядке.

SELECT ROW\_NUMBER() OVER (ORDER BY age) AS age\_num, age,COUNT(\*) AS clients,ROUND((COUNT(\*) \* 100.0) / SUM(COUNT(\*)) OVER (), 2) AS percent

FROM users

GROUP BY age

ORDER BY age;

13.7.4

В таблице **orders** хранится список заказов компании. Напишите запрос, который в первом столбце (**year**) — выведет год, во втором (**status**) — статус заказа, в третьем (**orders**) — общее количество заказов данного статуса за соответствующий год, а в четвертом (**percent**) — процент в рамках соответствующего года.

Данные отсортируйте по году и по статусу в алфавитном порядке.

SELECT

YEAR(date) AS year,

status,

COUNT(\*) AS orders,

ROUND((COUNT(\*) \* 100.0) / SUM(COUNT(\*)) OVER (PARTITION BY YEAR(date)), 2) AS percent

FROM

orders

GROUP BY

YEAR(date),

status

ORDER BY

YEAR(date),

status;

13.7.5

В таблице **orders** хранится список заказов компании. Напишите запрос, который в первом столбце (**year**) — выведет год, во втором (**user\_id**) — идентификатор пользователя, в третьем (**amount**) — сумму выполненных (success) заказов за текущий год для этого пользователя, а в четвертом (**percent**) — вклад пользователя в процентах в общую сумму доходов в рамках текущего года.

Данные отсортируйте по году и по вкладу пользователя в возрастающем порядке.

SELECT

YEAR(date) AS year,

user\_id,

SUM(amount) AS amount,

ROUND((SUM(amount) \* 100.0) / SUM(SUM(amount)) OVER (PARTITION BY YEAR(date)), 2) AS percent

FROM

orders

WHERE

status = 'success'

GROUP BY

YEAR(date),

user\_id

ORDER BY

year,

percent;

13.8.1

В таблице ниже содержится список квартир агентства недвижимости. Каждый раз, когда клиент просит подобрать варианты, ему предлагаются **три** самые подходящие позиции. Если по заданным критериям получается больше трёх вариантов, например несколько квартир с одинаковой ценой, то эти дополнительные позиции также попадают в итоговую выборку. И получается, что вместо трёх позиций клиенту отправляется больше.

Напишите SQL-запрос, который вернет список подходящих квартир. Клиента интересуют все варианты более 1 комнаты, **чем дешевле, тем лучше**.

В итоговой таблице выведите улицу, дом, цену и количество комнат.  
Финальную таблицу отсортируйте по комнатам в обратном порядке и цене в прямом.

SELECT

rooms,

street,

house,

price,

RANK() OVER (PARTITION BY rooms ORDER BY price ASC) AS price\_rank

FROM

flats

WHERE

rooms > 1

ORDER BY

rooms ASC,

price ASC;

13.8.2

В прошлой задаче мы искали варианты среди всех квартир: и двухкомнатных и трехкомнатных. Однако клиент попросил подобрать лучшие позиции отдельно по *двушкам* и отдельно по *трёшкам*, при этом сам алгоритм отбора квартир должен остаться прежним.

Напишите SQL-запрос, который вернет список подходящих квартир. Клиента интересуют все варианты более 1 комнаты, **чем дешевле, тем лучше**.

В итоговой таблице выведите количество комнат, улицу, дом и цену.  
Финальную таблицу отсортируйте по комнатам в прямом порядке и цене, также в прямом порядке.

select rooms, street, house, price from (select \*, rank() over (order by price) as rank\_num from flats

where rooms=2

union select \*, rank() over (order by price) as rank\_num from flats

where rooms=3) as flats1

where rank\_num<=3

order by rooms, price

13.8.3

Вернемся к основному алгоритму подбора квартир из первой задачи этого урока, где мы искали лучшие варианты среди всех квартир: и двухкомнатных и трехкомнатных. Но на этот раз немного изменим сам алгоритм.

Ранее, если мы получали три квартиры с одинаковой ценой, то все они попадали в итоговую выборку и более дорогие варианты не рассматривались. Теперь нам нужно получить также **минимум три** подходящих варианта, но при этом мы хотим рассмотреть квартиры с разными ценами: три лучших предложения цены.

То есть нам нужно **три лучших предложения с разными ценами** (три варианта стоимости). Если среди этих вариантов будут квартиры с одинаковой ценой, то все они должны попасть в итоговую выборку. То есть в реальности мы можем получить не три квартиры, а более.

Напишите SQL-запрос, который вернет список подходящих квартир. Клиента снова интересуют все варианты более 1 комнаты, **чем дешевле, тем лучше**.

В итоговой таблице выведите улицу, дом, цену и количество комнат.  
Финальную таблицу отсортируйте по цене в прямом порядке, а также по комнатам в обратном порядке.

WITH RankedFlats AS (

SELECT street, house,

price,

rooms,

DENSE\_RANK() OVER (ORDER BY price ASC) AS price\_rank

FROM

flats

WHERE

rooms > 1

)

SELECT

street,

house,

price,

rooms

FROM

RankedFlats

WHERE

price\_rank <= 3

ORDER BY

price ASC,

rooms DESC;

13.8.4

В киберспортивном турнире участвует 10 команд (таблица **cyber\_teams**) и каждой из них нужно сыграть по три игры (таблица **cyber\_games**) против компьютера. Результаты игр фиксируются в таблице **cyber\_results** и для подсчета набранных баллов используется следующая формула: количество убийств (**kills**) минус количество смертей (**deaths**) помноженное на три. Например, если команда в одной игре убила 60 противников и при этом 5 раз умерла, то её итоговый результат будет равен 60 - 5x3 = 45 баллов.

Напишите SQL-запрос для формирования итоговой таблицы мест после проведения турнира. В таблице должно быть 3 поля: **place** – место, **team** – название команды и **points** – общее количество набранных очков. В случае, если команды набирают одинаковое количество очков, они разделяют место между собой.

Итоговую таблицу отсортируйте по местам, а в случае одинакового места отсортируйте команды в алфавитном порядке.

WITH Points AS (SELECT r.team\_id, (r.kills - r.deaths \* 3) AS game\_points FROM cyber\_results r),

TotalPoints AS (SELECT t.team,SUM(p.game\_points) AS points FROM Points p JOIN cyber\_teams t ON p.team\_id = t.id

GROUP BY t.id, t.team),

RankedTeams AS (SELECT t.team,t.points,DENSE\_RANK() OVER (ORDER BY t.points DESC) AS place FROM TotalPoints t)

SELECT place,team, points

FROM RankedTeams

ORDER BY place ASC, team A

13.10.1

При email-рассылках часто делают несколько вариантов писем и равномерно распределяют эти варианты между всеми получателями. После рассылки маркетологи анализируют какой из вариантов принес больший эффект (конверсию) и уже после используют его для дальнейших рассылок.

Напишите SQL запрос для формирования списка рассылки. Всего планируется три тестовых письма, которые следует пронумеровать от 1 до 3-х с равномерным распределением чисел между пользователями таблицы **users**. Распределять письма будем в порядке следования идентификаторов, то есть первый блок пользователей получит единицу, следующий двойку и последний тройку.

Номер варианта следует вывести в первой колонке с названием **mail\_variant**, далее выведите **id** пользователя, его **email** и имя. Итоговые данные отсортируйте по **id**.

SELECT

id,

first\_name,

last\_name,

email,

NTILE(3) OVER (ORDER BY id ASC) AS email\_variant

FROM

users;

13.10.2

Перепишите SQL запрос из прошлого урока. Сделайте, чтобы функция **NTILE** распределяла пользователей на 4 группы, а данные обрабатывались в порядке следования **MD5** хэшей от **email** пользователей.

SELECT

id,

first\_name,

last\_name,

email,

NTILE(4) OVER (ORDER BY MD5(email)) AS email\_variant

FROM

users

ORDER BY

id;

13.10.3

Напишите SQL запрос, который выведет 5 колонок: **name** – название магазина, **first\_name** – имя покупателя, **last\_name** – фамилия покупателя, **amount** – общая сумма выполненных заказов (status="success") покупателя в текущем магазине, **c\_level** – группа (уровень) покупателя.

Всех покупателей магазин делит на четыре равные группы, нумеруя их от 1 (потратили больше всего) до 4 (потратили меньше всего). При этом для каждого магазина группы считаются по отдельности.

Итоговые данные отсортируйте по названию магазина, а после по группам в возрастающем порядке.

WITH OrderSums AS (

SELECT

o.user\_id,

o.shop\_id,

SUM(o.amount) AS amount

FROM

orders o

WHERE

o.status = 'success'

GROUP BY

o.user\_id,

o.shop\_id

), CustomerGroups AS (

SELECT

s.name AS shop\_name,

u.first\_name,

u.last\_name,

os.amount,

NTILE(4) OVER (PARTITION BY os.shop\_id ORDER BY os.amount DESC) AS c\_level

FROM

OrderSums os

JOIN users u ON os.user\_id = u.id

JOIN shops s ON os.shop\_id = s.id

)

SELECT

shop\_name AS name,

first\_name,

last\_name,

amount,

c\_level

FROM

CustomerGroups

ORDER BY

shop\_name ASC,

c\_level ASC;

13.10.4

Напишите SQL запрос, который выведет 4 колонки: **month** – номер месяца, **first\_name** – имя покупателя, **last\_name** – фамилия покупателя, **amount** – общая сумма выполненных заказов (status="success") покупателя в текущем месяце. **c\_level** выводить не нужно, так как нам интересны только покупатели из первой группы.

WITH MonthlySums AS (

SELECT

EXTRACT(MONTH FROM o.date) AS month,

u.first\_name,

u.last\_name,

SUM(o.amount) AS amount

FROM orders o

JOIN users u ON o.user\_id = u.id

WHERE o.status = 'success'

GROUP BY month, u.id

), MaxSums AS (

SELECT

month,

MAX(amount) AS max\_amount

FROM MonthlySums

GROUP BY month

)

SELECT

ms.month,

ms.first\_name,

ms.last\_name,

ms.amount

FROM MonthlySums ms

JOIN MaxSums mx ON ms.month = mx.month AND ms.amount = mx.max\_amount

ORDER BY ms.month ASC, ms.amount DESC;

13.12.1

В таблице ниже содержится информация о доходах (income) и расходах (outcome) компании за два года.

Напишите SQL-запрос, который выведет четыре столбца: **month** — номер месяца, **in2020** — доходы за каждый месяц 2020 года, **in2021** — доходы за каждый месяц 2021 года и **diff** разницу ежемесячных доходов между 2021 и 2020 годом.

В таблице должно быть только 12 строк, так как основная задача сравнить доходы 2021 года с прошлым.

Итоговые данные отсортируйте по месяцу в возрастающем порядке.

SELECT

month,

SUM(CASE WHEN year = 2020 THEN income ELSE 0 END) AS in2020,

SUM(CASE WHEN year = 2021 THEN income ELSE 0 END) AS in2021,

SUM(CASE WHEN year = 2021 THEN income ELSE 0 END) -

SUM(CASE WHEN year = 2020 THEN income ELSE 0 END) AS diff

FROM revenues

GROUP BY month

ORDER BY month;

13.12.2

Напишите SQL-запрос, который выведет четыре столбца: **quarter** — номер квартала, **in2020** — величину квартальных доходов за 2020 год, **in2021** — величину квартальных доходов за 2021 год и **diff** разницу квартальных доходов между 2021 и 2020 годом.

SELECT

CASE

WHEN month IN (1, 2, 3) THEN 1

WHEN month IN (4, 5, 6) THEN 2

WHEN month IN (7, 8, 9) THEN 3

WHEN month IN (10, 11, 12) THEN 4

END AS quarter,

SUM(CASE WHEN year = 2020 THEN income ELSE 0 END) AS in2020,

SUM(CASE WHEN year = 2021 THEN income ELSE 0 END) AS in2021,

SUM(CASE WHEN year = 2021 THEN income ELSE 0 END) -

SUM(CASE WHEN year = 2020 THEN income ELSE 0 END) AS diff

FROM revenues

GROUP BY quarter

ORDER BY quarter;

13.12.3

Выведите ожидаемые доходы на весь 2022 год (12 месяцев). Первый столбец итоговой таблицы должен называться **month** и содержать порядковый номер месяца. Второй столбец (**plan**) должен содержать величину планируемого дохода с округлением до целого.

SELECT

month,

ROUND(

income\_2021 \* (income\_2021 / income\_2020)

) AS plan

FROM

(SELECT

month,

MAX(CASE WHEN year = 2020 THEN income END) AS income\_2020,

MAX(CASE WHEN year = 2021 THEN income END) AS income\_2021

FROM revenues

GROUP BY month) AS yearly\_incomes;

13.13.1

Напишите SQL-запрос, который выведет пять столбцов: **place** — место спортсмена, **last\_name** — фамилию, **first\_name** — имя, **time** — время преодоления дистанции в формате ЧЧ:ММ:СС и **champion\_lag** — время отставания от первого места в формате ЧЧ:ММ:СС.

select

row\_number() over(order by (end\_time - start\_time)) as place,

last\_name,

first\_name,

TIME\_FORMAT(SUBTIME(end\_time, start\_time),'%H:%i:%s') as time,

TIME\_FORMAT(SUBTIME(SUBTIME(end\_time, start\_time), FIRST\_VALUE(SUBTIME(end\_time, start\_time)) OVER( order by SUBTIME(end\_time, start\_time))),'%H:%i:%s') AS champion\_lag from runners

order by place

13.15.1

Напишите SQL запрос для анализа 2021 года. Итоговая таблица должна содержать четыре колонки: **month** — номер месяца, **income** — доход за этот месяц, **diff** — разницу с апрелем в абсолютных показателях (текущий месяц - апрель), **diff\_percent** — разницу с апрелем в процентах с округлением до одного знака после десятичной точки, например, для мая это будет -13.6 (минус 13.6)

SELECT

month,

income,

income - (SELECT income FROM revenues WHERE year = 2021 AND month = 4) AS diff,

ROUND(

((income - (SELECT income FROM revenues WHERE year = 2021 AND month = 4)) /

(SELECT income FROM revenues WHERE year = 2021 AND month = 4)) \* 100,

1

) AS diff\_percent

FROM revenues

WHERE year = 2021

ORDER BY MONTH;

13.15.2

В прошлой задаче вы делали анализ эффективности продаж за 2021 год. Теперь сделайте анализ эффективности за два года. То есть сперва должен идти 2020 год, а затем 2021. В 2020 году продажи нужно сравнивать с апрелем 2020, а в 2021 году с апрелем 2021.

Относительно прошлого задания добавьте еще один столбец **year**, который должен идти вначале.

SELECT

revenues.year,

revenues.month,

revenues.income,

(income - (NTH\_VALUE(income, 4) OVER (

PARTITION BY year

ORDER BY month

RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

))) as diff,

ROUND((income / (NTH\_VALUE(income, 4) OVER (

PARTITION BY year

ORDER BY month

RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

)) - 1) \* 100, 1) as diff\_percent

FROM revenues ORDER BY year, MONTH;

13.16.1

Выведите ожидаемые доходы на весь 2022 год (12 месяцев). Первый столбец итоговой таблицы должен называться **month** и содержать порядковый номер месяца. Второй столбец (**plan**) должен содержать величину планируемого дохода с округлением до целого.

SELECT

ROW\_NUMBER() OVER(ORDER BY id) as month,

plan

FROM (

SELECT

id,

ROUND(

income \*

AVG(income) OVER (

ORDER BY year, month

ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) /

AVG(income) OVER (

ORDER BY year, month

ROWS BETWEEN 14 PRECEDING AND 12 PRECEDING)

) as plan

FROM revenues

ORDER BY id

) tmp

WHERE plan IS NOT NULL

ORDER BY month