CREATE SCHEMA lesson05;

USE lesson05;

CREATE TABLE common\_list

(

full\_name varchar(30),

birthday date,

status varchar(30),

phone\_number bigint,

commentary varchar(30),

address varchar(30),

user\_group varchar(30)

);

INSERT INTO common\_list

(full\_name, birthday, status, phone\_number, commentary, address, user\_group )

VALUES

('Иванов И. И.' , '1990-02-12','женат',126,'интернет','Можга','Работа'),

('Иванов И. И.' , '1990-02-12', 'женат', 127, 'старый', 'Можга', 'Работа'),

('Иванов И. И.' , '2001-09-18', 'холост', 527, 'личный', 'Санкт-Петербург', 'Школа'),

('Петров П. П.' , '1983-04-23', 'женат', 234, 'личный', 'Москва', 'Родня'),

('Петров П. П.' , '1983-04-24', 'женат', 235, 'рабочий', 'Москва', 'Родня'),

('Васильев В. В.' , '1998-05-21', 'холост', 456, 'личный', 'Белгород', 'Друзья'),

('Сидоров С.С.' , '2007-07-13', 'холост', 643, 'личный', 'Киров', 'Родня'),

('Сидоров С.С.' , '2007-07-13', 'холост', 654, 'старый', 'Киров', 'Родня'),

('Александров А.А.', '1987-03-05', 'женат', 412, 'рабочий', 'Уфа', 'Друзья'),

('Борисов Б.Б' , '1989-08-16', 'холост', 723, 'рабочий', 'Москва', 'Работа'),

('Михайлов М.М.' , '1997-01-08', 'женат', 876, 'личный', 'Сочи', 'Университет'),

('Михайлов М.М.' , '1997-01-08', 'женат', 875, 'рабочий', 'Сочи', 'Университет'),

('Кириллов К.К.' , '1996-12-03', 'холост', 933, 'личный', 'Киров', 'Работа')

SELECT \* FROM common\_list;

CREATE TABLE peoples

(

identifier int,

full\_name varchar(30),

birthday date,

status varchar(30)

);

INSERT INTO peoples

(identifier, full\_name, birthday, status)

VALUES

(1, 'Иванов И. И.', '1990-02-12', 'женат'),

(2, 'Иванов И. И.', '2001-09-18', 'холост'),

(3, 'Петров П. П.', '1983-04-23', 'женат'),

(4, 'Васильев В. В.', '1998-05-21', 'холост'),

(25, 'Кузьмин К.К.', '2020-05-21', 'холост');

SELECT \* FROM peoples;

CREATE TABLE telephones

(

whose\_phone int,

phone\_number bigint,

commentary varchar(30)

);

INSERT INTO telephones

(whose\_phone, phone\_number, commentary)

VALUES

(1, 123, 'личный'),

(1, 124, 'рабочий'),

(1, 125, 'для поездок'),

(1, 126, 'интернет'),

(1, 127, 'старый'),

(2, 527, 'личный'),

(3, 234, 'личный'),

(3, 235, 'рабочий'),

(4, 456, 'личный'),

(11, 999, 'личный'),

(12, 997, 'рабочий'),

(13, 995, 'личный'),

(14, 993, 'для поездок'),

(15, 991, 'для поездок');

SELECT \* FROM telephones;

WITH with\_tmp AS (

SELECT

peoples.full\_name,

count(\*) as c

FROM

peoples LEFT JOIN telephones

ON peoples.identifier = telephones.whose\_phone

GROUP BY

peoples.full\_name

HAVING

count(\*) >= 2

)

SELECT

peoples.full\_name,

with\_tmp.c

FROM peoples

JOIN with\_tmp

ON peoples.full\_name = with\_tmp.full\_name

GROUP BY peoples.full\_name;

-- === vvvv

/\*

**SELECT**

**peoples.full\_name,**

**with\_tmp.c**

**FROM peoples**

**JOIN (**

**SELECT**

**peoples.full\_name,**

**count(\*) as c**

**FROM**

**peoples LEFT JOIN telephones**

**ON peoples.identifier = telephones.whose\_phone**

**GROUP BY**

**peoples.full\_name**

**HAVING**

**count(\*) >= 2) as with\_tmp**

**ON peoples.full\_name = with\_tmp.full\_name**

**GROUP BY peoples.full\_name;**

\*/

-- === === === === === === === === ===

CREATE DATABASE mydb;

USE mydb;

CREATE TABLE users (

username VARCHAR(50) PRIMARY KEY,

password VARCHAR(50) NOT NULL,

status VARCHAR(10) NOT NULL);

CREATE TABLE users\_profile (

username VARCHAR(50) PRIMARY KEY,

name VARCHAR(50) NOT NULL,

address VARCHAR(50) NOT NULL,

email VARCHAR(50) NOT NULL,

FOREIGN KEY (username) REFERENCES users(username) ON DELETE CASCADE);

INSERT INTO users values

('admin' , '7856', 'Active'),

('staff' , '90802', 'Active'),

('manager' , '35462', 'Inactive');

INSERT INTO users\_profile values

('admin', 'Administrator' , 'Dhanmondi', 'admin@test.com' ) ,

('staff', 'Jakir Nayek' , 'Mirpur', 'zakir@test.com' ),

('manager', 'Mehr Afroz' , 'Eskaton', 'mehr@test.com' );

-- 1. Используя СТЕ, выведите всех

-- пользователей из таблицы users\_profile

-- 2. Используя СТЕ, подсчитайте количество

-- активных пользователей .

-- Задайте псевдоним результирующему окну.

-- 3. С помощью СТЕ реализуйте

-- таблицу квадратов чисел от 1 до 10:

-- === === === === === === === === ===

-- Задача:

-- Собрать дэшборд, в котором содержится информация

-- о максимальной задолженности в каждом банке,

-- а также средний размер процентной ставки в каждом

-- банке в зависимости от сегмента и количество

-- договоров всего всем банкам

CREATE TABLE something (

TB VARCHAR(50),

ID\_CLIENT INT,

ID\_DOG INT,

OSZ INT,

PROCENT\_RATE INT,

RATING INT,

SEGMENT VARCHAR(10)

);

INSERT INTO something

(TB, ID\_CLIENT, ID\_DOG, OSZ, PROCENT\_RATE, RATING, SEGMENT)

VALUES

('A', 1, 111, 100 , 6 , 10 , 'SREDN' ),

('A', 1, 222, 150 , 6 , 10 , 'SREDN' ),

('A', 2, 333, 50 , 9 , 15 , 'MMB' ),

('B', 1, 444, 200 , 7 , 10 , 'SREDN' ),

('B', 3, 555, 1000, 5 , 16 , 'CIB' ),

('B', 4, 666, 500 , 10 , 20 , 'CIB' ),

('B', 4, 777, 10 , 12 , 17 , 'MMB' ),

('C', 5, 888, 20 , 11 , 21 , 'MMB' ),

('C', 5, 999, 200 , 9 , 13 , 'SREDN' );

SELECT \* FROM something;

//5 hw

CREATE TABLE cars

(

id INT NOT NULL PRIMARY KEY,

name VARCHAR(45),

cost INT

);

INSERT cars

VALUES

(1, "Audi", 52642),

(2, "Mercedes", 57127 ),

(3, "Skoda", 9000 ),

(4, "Volvo", 29000),

(5, "Bentley", 350000),

(6, "Citroen ", 21000 ),

(7, "Hummer", 41400),

(8, "Volkswagen ", 21600);

SELECT \*

FROM cars;

ЛЕКЦИЯ

CREATE SCHEMA `lesson05` ;

use `lesson05`;

-- Базы данных и SOL

-- Оконные функции SQL: таблица для работы

CREATE TABLE sales(

sales\_employee VARCHAR(50) NOT NULL,

fiscal\_year INT NOT NULL,

sale DECIMAL(14, 2) NOT NULL,

PRIMARY KEY(sales\_employee,fiscal\_year)

);

INSERT INTO sales (sales\_employee, fiscal\_year, sale)

VALUES( 'Bob' ,2016, 100),

('Bob' ,2017, 150),

('Bob' ,2018,200),

('Alice' ,2016, 150),

('Alice' ,2017, 100),

('Alice' ,2018,200),

('John' ,2016, 200),

(' John', 2017, 150),

('John' ,2018,250);

SELECT

\*

FROM sales;

SELECT

SUM(sale)

FROM sales;

SELECT

fiscal\_year,

SUM(sale)

FROM sales

GROUP BY fiscal\_year;

-- SELECT

-- Название функции (столбец для вычислений)

-- OVER

-- (

-- PARTITION BY столбец для группировки

-- ORDER BY столбец для сортировки

-- ROWS или RANGE выражение для ограничения строк в пределах группы

-- )

-- SELECT

-- Date,

-- Medium,

-- Conversions,

-- SUM(Conversions) OVER() AS "Sum'

-- FROM Orders;

DROP TABLE `lesson05`.`orders`;

CREATE TABLE `lesson05`.`orders` (

`date` DATE NOT NULL,

`medium` VARCHAR(45) NOT NULL,

`conversions` INT NOT NULL);

TRUNCATE `lesson05`.`orders`;

INSERT INTO `lesson05`.`orders` (`date`, `medium`, `conversions`)

VALUES

('2020-05-10', 'cpa', 1),

('2020-05-10', 'cpc', 2),

('2020-05-10', 'organic', 1),

('2020-05-11', 'cpa', 1),

('2020-05-11', 'cpc', 3),

('2020-05-11', 'direct', 1),

('2020-05-11', 'organic', 2),

('2020-05-12', 'cpc', 1),

('2020-05-12', 'organic', 2);

SELECT \* FROM orders;

SELECT

`date`,

`medium`,

`conversions`,

SUM(`conversions`) OVER() AS 'Sum'

FROM orders

ORDER BY `date`;

-- vvvvvvv

SELECT

`date`,

`medium`,

`conversions`,

SUM(`conversions`) OVER(PARTITION BY `date`) AS 'Sum'

FROM orders

ORDER BY `date`;

-- vvvvvvv

SELECT

`date`,

`medium`,

`conversions`,

SUM(`conversions`) OVER(PARTITION BY `date` ORDER BY `medium`) AS 'Sum'

FROM orders

ORDER BY `date`;

--- === === === === === === ---

SELECT

`date`,

-- `medium`,

-- `conversions`,

SUM(`conversions`) AS 'Sum'

FROM orders

GROUP BY `date`;

SELECT

date,

medium,

conversions,

SUM(conversions) OVER(PARTITION BY date ORDER BY medium) AS 'Sum'

FROM orders;

-- ROWS и RANGE

-- UNBOUNDED PRECEDING - указывает,

-- что окно начинается с первой строки группы;

-- UNBOUNDED FOLLOWING - с помощью данной

-- инструкции можно указать, что окно заканчивается на

-- последней строке группы;

-- CURRENT ROW - инструкция указывает, что окно

-- начинается или заканчивается на текущей строке;

-- BETWEEN «граница окна» AND «граница окна»

-- - указывает нижнюю и верхнюю границу окна;

-- «Значение» PRECEDING - определяет число строк перед

-- текущей строкой (не допускается в предложении RANGE);

-- «Значение» FOLLOWING - определяет число строк после

-- текущей строки (не допускается в предложени RANGE);

SELECT

date,

medium,

conversions,

SUM(conversions) OVER(PARTITION BY date ORDER BY conversions ROWS BETWEEN

CURRENT ROW AND 1 FOLLOWING) AS 'Sum'

FROM orders;

-- Window Functions

-- Aggregate Ranking Value

-- AVG() DENSE\_RANK() FIRST\_VALUE()

-- COUNT() NTILE() LAST\_VALUE()

-- MAX() RANK() LAG()

-- MIN() ROW\_NUMBER() LEAD()

-- SUM() CUME\_DIST() NTH\_VALUE()

-- Агрегатные функции

-- SUM - возвращает сумму значений в столбце;

-- COUNT - вычисляет количество значений в столбце

-- (значения NULL не учитываются);

-- AVG - определяет среднее значение в столбце;

-- MAX - определяет максимальное значение в столбце:

-- MIN - определяет минимальное значение в столбце.

SELECT

date,

medium,

conversions,

SUM(conversions) OVER(PARTITION BY date) AS 'Sum',

COUNT(conversions) OVER(PARTITION BY date) AS 'Count',

AVG(conversions) OVER(PARTITION BY date) AS 'Avg',

MAX(conversions) OVER(PARTITION BY date) AS 'Max',

MIN(conversions) OVER(PARTITION BY date) AS 'Min'

FROM orders;

-- ROW NUMBER - функция возвращает номер

-- строки используется для нумерации;

-- RANK - функция возвращает ранг каждой строки.

-- В данном случае значения уже анализируются и,

-- в случае нахождения одинаковых, возвращает

-- одинаковый ранг с пропуском следующего значения;

-- DENSE\_RANK - функция возвращает ранг каждой строки.

-- Но в отличие от функции RANK, она для одинаковых

-- значений возвращает ранг, не пропуская следующий;

-- NTILE - это функция, которая позволяет определить

-- к какой группе относится текущая строка.

-- Количество групп задается в скобках.

SELECT

date,

medium,

conversions,

ROW\_NUMBER()

OVER( PARTITION BY date ORDER BY conversions) AS 'Row number',

RANK()

OVER(PARTITION BY date ORDER BY conversions) AS 'Rank',

DENSE\_RANK()

OVER(PARTITION BY date ORDER BY conversions) AS 'Dense Rank',

NTILE(3)

OVER(PARTITION BY date ORDER BY conversions) AS 'Ntile'

FROM orders;

-- LAG или LEAD - функция LAG обращается к данным

-- из предыдущей строки окна, а LEAD к данным из следующей строки.

-- Функцию можно использовать для того, чтобы

-- сравнивать текущее значение строки с предыдущим

-- или следующим. Имеет три параметра: столбец,

-- значение которого необходимо вернуть,

-- количество строк для смещения (по умолчанию 1),

-- значение, которое необходимо вернуть если после смешения

-- возврашается значение NULL:

-- FIRST VALUE или LAST VALUE - с помощью функции

-- можно получить первое и последнее значение в окне.

-- В качестве параметра принимает столбец,

-- значение которого необходимо вернуть.

SELECT

date,

medium,

conversions,

LAG(conversions) OVER(PARTITION BY date ORDER BY date) AS 'Lag',

LEAD (conversions) OVER(PARTITION BY date ORDER BY date) AS 'Lead',

FIRST\_VALUE(conversions) OVER(PARTITION BY date ORDER BY date) As 'First Value',

LAST\_VALUE (conversions) OVER(PARTITION BY date ORDER BY date) AS 'Last Value'

FROM orders;

-- Представления

-- CREATE [OR REPLACE] VIEW view\_name AS

-- SELECT columns

-- FROM tables

-- [WHERE conditions];

-- OR REPLACE - необязательный.

-- Если вы не укажете этот атрибут и

-- VIEW уже существует, оператор

-- CREATE VIEW вернет ошибку.

-- view\_name - имя VIEW,

-- которое вы хотите создать в MySQL.

-- WHERE conditions - необязательный.

-- Условия, которые должны быть выполнены

-- для записей, которые должны быть

-- включены в VIEW.

-- CREATE TABLE `lesson05`.`orders` (

-- `date` DATE NOT NULL,

-- `medium` VARCHAR(45) NOT NULL,

-- `conversions` INT NOT NULL);

SELECT \* FROM orders;

CREATE VIEW orders\_view

AS SELECT date, conversions

FROM orders

WHERE medium = 'organic';

SELECT \* FROM orders\_view;

DROP VIEW orders\_view;

-- JOINs

-- ALTER VIEW view\_name AS

-- SELECT columns

-- FROM table

-- WHERE conditions;

-- ALTER VIEW hardware\_suppliers AS

-- SELECT supplier\_id, supplier \_name, address, city

-- FROM suppliers

-- WHERE category\_type = 'Hardware';

ALTER VIEW orders\_view AS

SELECT conversions, medium

FROM orders

WHERE medium = 'organic';

SELECT \* FROM orders\_view;