Базы данных и SQL. Обучение в записи

Урок 10. Семинар: SQL – оконные функции

Ссылка на репозиторий, с выполненным Д/З:
https://github.com/olgashenkel/Databases-and-SQL

ЧАСТЬ I.

Домашнее задание

1. Создайте представление, в которое попадут автомобили стоимостью до 25 000 долларов

2. Изменить в существующем представлении порог для стоимости: пусть цена будет до 30 000 долларов (используя оператор ALTER VIEW)

3. Создайте представление, в котором будут только автомобили марки “Шкода” и “Ауди”

```
mysql> SELECT * FROM Cars;
+----+------------+--------+
| Id | Name       | Cost   |
+----+------------+--------+
|  1 | Audi       |  52642 |
|  2 | Mercedes   |  57127 |
|  3 | Skoda      |   9000 |
|  4 | Volvo      |  29000 |
|  5 | Bentley    | 350000 |
|  6 | Citroen    |  21000 |
|  7 | Hummer     |  41400 |
|  8 | Volkswagen |  21600 |
+----+------------+--------+
```

1. *Ход выполнения задания 1.1:*

Создайте представление, в которое попадут автомобили стоимостью до 25 000 долларов.

1)

```sql
USE seminar_5;

/*
1.  Ззадание 1:
Создайте представление, в которое попадут
автомобили стоимостью до 25 000 долларов.
*/

-- Создание и наполнение данными таблицы cars
DROP TABLE IF EXISTS cars;
CREATE TABLE cars
(
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(20),
    cost INT
);

INSERT INTO cars(name, cost)
VALUES
('Audi', 52642),
('Mercedes', 57127),
('Skoda', 9000),
('Volvo', 29000),
('Bentley', 350000),
('Citroen', 21000),
('Hummer', 41400),
('Volksvagen', 21600);
```

2)

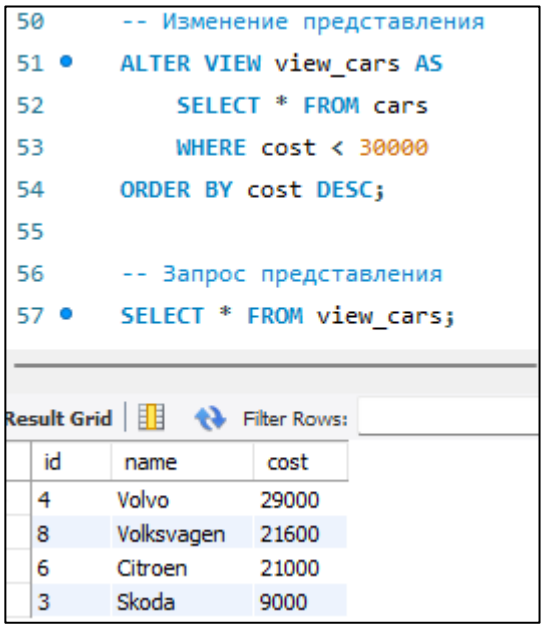
```sql
-- Создание представления
CREATE OR REPLACE VIEW view_cars AS
    SELECT *
    FROM cars
    WHERE cost < 25000
    ORDER BY cost DESC;

-- Запрос созданного представления
SELECT * FROM view_cars;
```

Result Grid | Filter Rows: | Export

id	name	cost
8	Volksvagen	21600
6	Citroen	21000
3	Skoda	9000

2. *Ход выполнения задания 1.2:*

Изменить в существующем представлении порог стоимости: пусть цена будет до 30000 долларов (используя оператор ALTER VIEW).



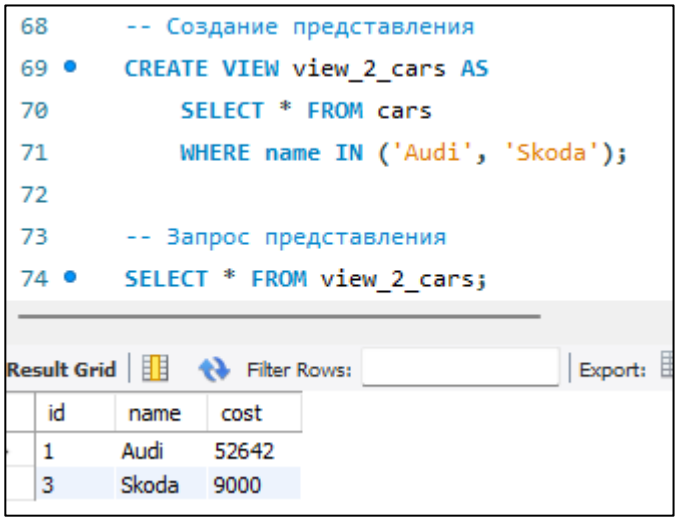
```
50      -- Изменение представления
51      ALTER VIEW view_cars AS
52          SELECT * FROM cars
53          WHERE cost < 30000
54      ORDER BY cost DESC;
55
56      -- Запрос представления
57      SELECT * FROM view_cars;
```

Result Grid | Filter Rows:

id	name	cost
4	Volvo	29000
8	Volksvagen	21600
6	Citroen	21000
3	Skoda	9000

3. *Ход выполнения задания 1.3:*

Создайте представление, в котором будут только автомобили марки Skoda и Audi.



```
68      -- Создание представления
69      CREATE VIEW view_2_cars AS
70          SELECT * FROM cars
71          WHERE name IN ('Audi', 'Skoda');
72
73      -- Запрос представления
74      SELECT * FROM view_2_cars;
```

Result Grid | Filter Rows: | Export:

id	name	cost
1	Audi	52642
3	Skoda	9000

Домашнее задание

Вывести название и цену для всех анализов, которые продавались 5 февраля 2020 и всю следующую неделю.

Есть таблица анализов Analysis:
an_id — ID анализа;
an_name — название анализа;
an_cost — себестоимость анализа;
an_price — розничная цена анализа;
an_group — группа анализов.
Есть таблица групп анализов Groups:
gr_id — ID группы;
gr_name — название группы;
gr_temp — температурный режим хранения.
Есть таблица заказов Orders:
ord_id — ID заказа;
ord_datetime — дата и время заказа;
ord_an — ID анализа.

1. Ход выполнения задания 2:

Вывести название и цену всех анализов, которые продавались 5 февраля 2020 и всю следующую неделю.

1)

```
USE semnar_5;

-- Создание и заполнение таблицы analysis
DROP TABLE IF EXISTS analysis;
CREATE TABLE analysis
(
    an_id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    an_name VARCHAR(45) NOT NULL,
    an_cost DECIMAL(10,2),
    an_price DECIMAL(10,2),
    an_group VARCHAR(10)
);

INSERT INTO analysis(an_name, an_cost, an_price, an_group)
VALUE
('analysis_001', 50, 89, 'A'),
('analysis_002', 60, 95, 'B'),
('analysis_003', 70, 105, 'C'),
('analysis_004', 80, 110, 'D'),
('analysis_005', 90, 120, 'A'),
('analysis_006', 100, 140, 'B'),
('analysis_007', 110, 150, 'C'),
('analysis_008', 120, 170, 'D'),
('analysis_009', 130, 160, 'A'),
('analysis_010', 140, 180, 'B');

SELECT * FROM analysis;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

an_id	an_name	an_cost	an_price	an_group
1	analysis_001	50.00	89.00	A
2	analysis_002	60.00	95.00	B
3	analysis_003	70.00	105.00	C
4	analysis_004	80.00	110.00	D
5	analysis_005	90.00	120.00	A
6	analysis_006	100.00	140.00	B
7	analysis_007	110.00	150.00	C
8	analysis_008	120.00	170.00	D
9	analysis_009	130.00	160.00	A
10	analysis_010	140.00	180.00	B

2)

```
-- Создание и заполнение таблицы grups
DROP TABLE IF EXISTS grups;
CREATE TABLE grups
(
    gr_id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    gr_name VARCHAR(45) NOT NULL,
    gr_temp DECIMAL(5,2)
);

INSERT INTO grups(gr_name, gr_temp)
VALUE
('group_001', 10),
('group_002', 11),
('group_003', 12),
('group_004', 13),
('group_005', 14),
('group_006', 15),
('group_007', 16),
('group_008', 17),
('group_009', 18),
('group_010', 19);

SELECT * FROM grups;
```

Result Grid | Filter Rows: | Edit: | Exp

gr_id	gr_name	gr_temp
1	group_001	10.00
2	group_002	11.00
3	group_003	12.00
4	group_004	13.00
5	group_005	14.00
6	group_006	15.00
7	group_007	16.00
8	group_008	17.00
9	group_009	18.00
10	group_010	19.00

3)

```sql
-- Создание и заполнение таблицы orders
CREATE TABLE orders
(
    ord_id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    ord_datetime DATETIME,
    ord_an INT NOT NULL,
    FOREIGN KEY (ord_an) REFERENCES analysis (an_id)
);

INSERT INTO orders(ord_datetime, ord_an)
VALUE
('2020-02-04 00:00:00', 1),
('2020-02-04 00:01:00', 2),
('2020-02-04 00:02:00', 3),
('2020-02-04 12:00:00', 4),
('2020-02-05 00:00:00', 5),
('2020-02-05 00:00:00', 6),
('2020-02-05 00:00:00', 7),
('2020-02-05 00:00:00', 8),
('2020-02-06 00:00:00', 9),
('2020-02-06 00:00:00', 10),
('2020-02-06 00:00:00', 1),
('2020-02-06 00:00:00', 2),
('2020-02-07 00:00:00', 3),
('2020-02-14 00:00:00', 4),
('2020-02-24 00:00:00', 5),
('2020-02-08 00:00:00', 6),
('2020-02-08 00:00:00', 7),
('2020-02-09 00:00:00', 8),
('2020-02-14 00:00:00', 9),
('2020-02-15 00:00:00', 10);

SELECT * FROM orders;
```

Result Grid | Filter Rows: | Edit: | Export/

ord_id	ord_datetime	ord_an
1	2020-02-04 00:00:00	1
2	2020-02-04 00:01:00	2
3	2020-02-04 00:02:00	3
4	2020-02-04 12:00:00	4
5	2020-02-05 00:00:00	5

4)

```sql
-- Вывести название и цену всех анализов,
-- которые продавались 5 февраля 2020 и всю следующую неделю
SELECT an.an_name, an.an_price, ord.ord_datetime
FROM analysis AS an
INNER JOIN orders AS ord
    ON an.an_id = ord.ord_an
    WHERE ord.ord_datetime >= '2020-02-05'
        AND ord.ord_datetime < '2020-02-05' + INTERVAL 7 DAY;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

an_name	an_price	ord_datetime
analysis_005	120.00	2020-02-05 00:00:00
analysis_006	140.00	2020-02-05 00:00:00
analysis_007	150.00	2020-02-05 00:00:00
analysis_008	170.00	2020-02-05 00:00:00
analysis_009	160.00	2020-02-06 00:00:00
analysis_010	180.00	2020-02-06 00:00:00
analysis_001	89.00	2020-02-06 00:00:00
analysis_002	95.00	2020-02-06 00:00:00
analysis_003	105.00	2020-02-07 00:00:00
analysis_006	140.00	2020-02-08 00:00:00
analysis_007	150.00	2020-02-08 00:00:00
analysis_008	170.00	2020-02-09 00:00:00

Домашнее задание

Добавьте новый столбец под названием «время до следующей станции». Чтобы получить это значение, мы вычитаем время станций для пар смежных станций. Мы можем вычислить это значение без использования оконной функции SQL, но это может быть очень сложно. Проще это сделать с помощью оконной функции LEAD . Эта функция сравнивает значения из одной строки со следующей строкой, чтобы получить результат. В этом случае функция сравнивает значения в столбце «время» для станции со станцией сразу после нее.

train_id integer	station character varying(20)	station_time time without time zone	time_to_next_station interval
110	San Francisco	10:00:00	00:54:00
110	Redwood City	10:54:00	00:08:00
110	Palo Alto	11:02:00	01:33:00
110	San Jose	12:35:00	
120	San Francisco	11:00:00	01:49:00
120	Palo Alto	12:49:00	00:41:00
120	San Jose	13:30:00	

2. *Ход выполнения задания 3:*

Добавьте новый столбец под названием «время до следующей станции».

```
11 •  SELECT *,
12        TIMEDIFF(
13        LEAD (station_time)
14        OVER (PARTITION BY id_train), station_time) AS time_to_next_station
15    FROM train;
16
17
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id_train	station	station_time	time_to_next_station
110	San Francisco	10:00:00	00:54:00
110	Redwood City	10:54:00	00:08:00
110	Palo Alto	11:02:00	01:33:00
110	San Jose	12:35:00	NULL
120	San Francisco	11:00:00	01:49:00
120	Palo Alto	12:49:00	00:41:00
120	San Jose	13:30:00	NULL