

## Мелехин Александр Кс-30 Вариант 9 Лабораторная работа 5

### Данные таблицы для лабораторной работы 5

Таблица salers

	saler_id [PK] integer	saler_name character varying (100)	saler_sex character varying (100)	saler_age integer
1	1	Иванов	Мужской	20
2	2	Петрова	Женский	19
3	3	Сидорова	Женский	21

Таблица brands

	brand_id [PK] integer	brand_name character varying (100)
1	1	Самсунг
2	2	Леново
3	3	Сони

Таблица sales

	sale_id [PK] integer	sale_date date	brand integer	price numeric	sale_count integer	saler integer
1	1	2005-01-03	1	12000	5	1
2	2	2005-01-15	2	8000	4	2
3	3	2005-02-02	1	25000	3	3
4	4	2005-03-02	1	10000	5	1
5	5	2005-02-14	3	11000	3	3
6	6	2005-04-04	1	19000	4	2
7	7	2005-01-07	2	16500	4	[null]
8	8	2005-01-07	[null]	12500	3	2

## Задание 1

**Задание:** Создайте объединение из двух запросов, которое показало бы имена ОБЪЕКТов, некоторый числовой параметр (значения NULL не показывать) их в дочерней таблице и текстовый параметр. Строки набора, которые имеют значения числового параметра больше среднего, должны иметь текстовый параметр "Выше среднего", а те, которые имеют меньше среднего текстовый параметр "Ниже среднего". Результат отсортируйте по алфавиту имен.

### SQL код для задания:

```
SELECT salers.saler_name AS "имя объекта", sales.price AS "числовой  
параметр",
```

```
    CASE
```

```
        WHEN sales.price > (SELECT AVG(price) FROM sales) THEN 'Выше  
среднего'
```

```
            WHEN sales.price > (SELECT AVG(price) FROM sales) THEN  
'Ниже среднего'
```

```
        ELSE 'Средний'
```

```
    END AS "текстовый параметр"
```

```
FROM sales
```

```
JOIN salers ON sales.saler = salers.saler_id
```

```
WHERE sales.price IS NOT NULL
```

```
UNION ALL
```

```
SELECT salers.saler_name AS "имя объекта", sales.sale_count AS "числовой  
параметр",
```

```
    CASE
```

```
        WHEN sales.sale_count > (SELECT AVG(sale_count) FROM sales) THEN  
'Выше среднего'
```

```
            WHEN sales.sale_count < (SELECT AVG(sale_count) FROM sales)  
THEN 'Ниже среднего'
```

```
        ELSE 'Средний'
```

```
    END AS "текстовый параметр"
```

FROM sales




JOIN salers ON sales.saler = salers.saler\_id

WHERE sales.sale\_count IS NOT NULL

ORDER BY "имя объекта";

**Пояснение:** запрос формирует объединение данных для продавцов, где указывается имя объекта, числовой параметр (цена или количество продаж), а также текстовое поле, указывающее, выше или ниже среднего значение числового параметра. Сортировка осуществляется по имени объекта в алфавитном порядке.

### Результат

	имя объекта character varying (100) 	числовой параметр numeric 	текстовый параметр text 
1	Иванов	5	Выше среднего
2	Иванов	10000	Средний
3	Иванов	15000	Выше среднего
4	Иванов	5	Выше среднего
5	Иванов	5	Выше среднего
6	Иванов	12000	Средний
7	Петрова	19000	Выше среднего
8	Петрова	4	Средний
9	Петрова	8000	Средний
10	Петрова	3	Ниже среднего
11	Петрова	12500	Средний
12	Петрова	4	Средний
13	Сидорова	3	Ниже среднего
14	Сидорова	11000	Средний
15	Сидорова	25000	Выше среднего
16	Сидорова	3	Ниже среднего

## Задание 2

**Задание:** Создайте объединение из двух запросов, которое показало бы имена ОБЪЕКТов, некоторый числовой параметр в дочерней таблице (значения NULL не показывать) и текстовый параметр. Строки набора, которые имеют максимальное значение числового параметра, должны, кроме того, иметь текстовый параметр "Наивысший", а те, которые имеют минимальное значение "Низший". Результат отсортируйте по алфавиту имен в обратном порядке.

### SQL код для задания:

```
SELECT brands.brand_name AS "имя объекта", sales.price AS "числовой  
параметр",
```

```
    CASE
```

```
        WHEN sales.price = (SELECT MAX(price) FROM sales) THEN  
'Наивысший'
```

```
        WHEN sales.price = (SELECT MIN(price) FROM sales) THEN 'Низший'
```

```
        ELSE 'Средний'
```

```
    END AS "текстовый параметр"
```

```
FROM sales
```

```
JOIN brands ON sales.brand = brands.brand_id
```

```
WHERE sales.price IS NOT NULL
```

```
UNION ALL
```

```
SELECT brands.brand_name AS "имя объекта", sales.sale_count AS "числовой  
параметр",
```

```
    CASE
```

```
        WHEN sales.sale_count = (SELECT MAX(sale_count) FROM sales) THEN  
'Наивысший'
```

```
        WHEN sales.sale_count = (SELECT MIN(sale_count) FROM sales) THEN  
'Низший'
```

```
        ELSE 'Средний'
```

```
    END AS "текстовый параметр"
```

FROM sales




JOIN brands ON sales.brand = brands.brand\_id

WHERE sales.sale\_count IS NOT NULL

ORDER BY "имя объекта" DESC;

**Пояснение:** объединение отображает объекты с наивысшим и низшим значением для числовых параметров, добавляя текстовый параметр "Наивысший" или "Низший" соответственно. Результаты отсортированы в обратном алфавитном порядке по имени объекта.

### Результат

	имя объекта character varying (100) 	числовой параметр numeric 	текстовый параметр text 
1	Сони	11000	Средний
2	Сони	3	Низший
3	Самсунг	10000	Средний
4	Самсунг	19000	Средний
5	Самсунг	5	Наивысший
6	Самсунг	25000	Наивысший
7	Самсунг	3	Низший
8	Самсунг	5	Наивысший
9	Самсунг	4	Средний
10	Самсунг	12000	Средний
11	Самсунг	15000	Средний
12	Самсунг	5	Наивысший
13	Леново	16500	Средний
14	Леново	8000	Низший
15	Леново	4	Средний
16	Леново	4	Средний

### Задание 3

**Задание:** создайте внешнее объединение двух запросов.

#### SQL код для задания:

```
SELECT s.saler_id AS id, s.saler_name AS name, 'SALER — MATCH' AS
match_status

FROM salers s

WHERE s.saler_id = ANY (SELECT saler FROM sales)

UNION

SELECT s.saler_id AS id, s.saler_name AS name, 'SALER — NO MATCH' AS
match_status

FROM salers s

WHERE s.saler_id <> ALL (SELECT saler FROM sales)

UNION

SELECT sa.saler AS id, NULL AS name, 'SALE — MATCH' AS match_status

FROM sales sa

WHERE sa.saler = ANY (SELECT saler_id FROM salers)

UNION

SELECT sa.saler AS id, NULL AS name, 'SALE — NO MATCH' AS match_status

FROM sales sa

WHERE sa.saler <> ALL (SELECT saler_id FROM salers)

ORDER BY name DESC;
```

**Пояснение:** запрос выполняет внешнее объединение для брендов и продавцов. Выводит название объекта, числовой параметр и тип объекта.

#### Результат

	id integer	name character varying	match_status text
1	3	[null]	SALE — MATCH
2	1	[null]	SALE — MATCH
3	2	[null]	SALE — MATCH
4	3	Сидорова	SALER — MATCH
5	2	Петрова	SALER — MATCH
6	1	Иванов	SALER — MATCH

## Задание 4

**Задание:** Создайте запрос на пересечение однотипных запросов с разными условиями отбора строк

### SQL код для задания:

```
SELECT sale_id, sale_date, price
```

```
FROM sales
```

```
WHERE price > 10000
```

```
INTERSECT
```

```
SELECT sale_id, sale_date, price
```

```
FROM sales
```

```
WHERE sale_count > 3;
```

**Пояснение:** запрос выбирает пересечение строк, удовлетворяющих двум условиям: цена должна быть выше 10000, а количество продаж больше 3.

### Результат

	sale_id integer	sale_date date	price numeric
1	7	2005-01-07	16500
2	6	2005-04-04	19000
3	1	2005-01-03	12000

## Задание 5

**Задание:** Создайте запрос на вычитание однотипных запросов с разными условиями отбора строк

### SQL код для задания:

```
SELECT sale_id, sale_date, price
```

```
FROM sales
```

```
WHERE price > 10000
```

```
EXCEPT
```

```
SELECT sale_id, sale_date, price
```

```
FROM sales
```

```
WHERE sale_count <= 3;
```

**Пояснение:** запрос выбирает строки, где цена превышает 10000, но количество продаж более 3, путем исключения строк с количеством продаж, меньшим или равным 3.

### Результат

	sale_id integer	sale_date date	price numeric
1	7	2005-01-07	16500
2	6	2005-04-04	19000
3	1	2005-01-03	12000



## Задание 6

**Задание:** Создайте модифицируемое представление (с опцией проверки), которое ограничивает доступ к определенным строкам и столбцам в родительской таблице.

### SQL код для задания:

```
CREATE VIEW limited_sales_view AS
```

```
SELECT sale_id, sale_date, price
```

```
FROM sales
```

```
WHERE price > 15000;
```

```
SELECT * FROM limited_sales_view;
```

**Пояснение:** создается представление для таблицы *sales*, доступное только для строк с ценой выше 15000.

### Результат

	sale_id integer	sale_date date	price numeric
1	3	2005-02-02	25000
2	6	2005-04-04	19000
3	7	2005-01-07	16500

## Задание 7

**Задание:** Создайте представление "Itog\_query" для просмотра и модификации данных, в котором отражены данные исходной таблицы с наименованиями полей вашего варианта задания в Лаб. №1

### SQL код для задания:

```
INSERT INTO public.sales(sale_date, brand, price, sale_count, saler) VALUES ('3.1.2015', 1, NULL, 5, 1); ← понадобится для задания 8
```

```
CREATE VIEW Itog_query AS
```

```
SELECT s.sale_id, s.sale_date, b.brand_name, s.price, s.sale_count, sl.saler_name  
FROM sales s
```

```
FULL JOIN brands b ON s.brand = b.brand_id
```

```
FULL JOIN salers sl ON s.saler = sl.saler_id;
```

```
SELECT * FROM Itog_query;
```

**Пояснение:** представление объединяет данные из таблиц *sales*, *brands* и *salers*, позволяя просматривать полную информацию о продажах, включая бренд и имя продавца.

### Результат

	sale_id integer	sale_date date	brand_name character varying (100)	price numeric	sale_count integer	saler_name character varying (100)
1	1	2005-01-03	Самсунг	12000	5	Иванов
2	2	2005-01-15	Леново	8000	4	Петрова
3	3	2005-02-02	Самсунг	25000	3	Сидорова
4	4	2005-03-02	Самсунг	10000	5	Иванов
5	5	2005-02-14	Сони	11000	3	Сидорова
6	6	2005-04-04	Самсунг	19000	4	Петрова
7	7	2005-01-07	Леново	16500	4	[null]
8	8	2005-01-07	[null]	12500	3	Петрова
9	9	2015-01-03	Самсунг	[null]	5	Иванов

## Задание 8

**Задание:** С помощью созданного представления "Itog\_query" произведите обновления в строке, содержащей NULL-значения.

### SQL код для задания:

```
CREATE OR REPLACE RULE update_itog_query AS
ON UPDATE TO Itog_query
DO INSTEAD
UPDATE sales
SET price = NEW.price,
    sale_date = NEW.sale_date,
    sale_count = NEW.sale_count
WHERE sale_id = NEW.sale_id;
UPDATE Itog_query
SET price = 15000
WHERE sale_id = (SELECT sale_id FROM sales WHERE price IS NULL);
SELECT * FROM Itog_query;
```

**Пояснение:** запрос обновляет значение *price* в строке, где оно было NULL, установив значение в 15000.

### Результат

	sale_id integer	sale_date date	brand_name character varying (100)	price numeric	sale_count integer	saler_name character varying (100)
1	1	2005-01-03	Самсунг	12000	5	Иванов
2	2	2005-01-15	Леново	8000	4	Петрова
3	3	2005-02-02	Самсунг	25000	3	Сидорова
4	4	2005-03-02	Самсунг	10000	5	Иванов
5	5	2005-02-14	Сони	11000	3	Сидорова
6	6	2005-04-04	Самсунг	19000	4	Петрова
7	7	2005-01-07	Леново	16500	4	[null]
8	8	2005-01-07	[null]	12500	3	Петрова
9	9	2015-01-03	Самсунг	15000	5	Иванов

## Задание 9

**Задание:** Создайте представление (с возможностью модификации и с опцией проверки) для дочерней таблицы.

**SQL код для задания:**

```
CREATE VIEW brands_view AS
```

```
SELECT brand_id, brand_name
```

```
FROM brands
```

```
WHERE brand_name != 'Леново';
```

```
SELECT * FROM brands_view;
```

**Пояснение:** представление ограничивает доступ к производителям Леново.

### Результат

	brand_id integer	brand_name character varying (100)
1	1	Самсунг
2	3	Сони

## Задание 10

**Задание:** Создайте представление "Avg\_Obj", которое бы показывало усредненные значения ОБЪЕКТов для каждого ОБЪЕКТа после его имени.

**SQL код для задания:**

```
CREATE VIEW Avg_Obj AS  
  
SELECT brand_name AS "производитель", ROUND(AVG(sales.price), 0) AS  
"средняя цена"  
  
FROM sales  
  
JOIN brands ON sales.brand = brands.brand_id  
  
GROUP BY brands.brand_name;  
  
SELECT * FROM Avg_Obj;
```

**Пояснение:** представление *Avg\_Obj* показывает усредненные значения цен для каждого бренда, группируя данные по названию бренда.

### Результат

	производитель character varying (100) 🔒	средняя цена numeric 🔒
1	Леново	12250
2	Сони	11000
3	Самсунг	16200