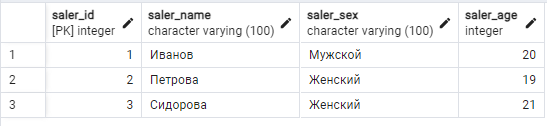
**Мелехин Александр Кс-30 Вариант 9 Лабораторная работа 5**

**Данные таблицы для лабораторной работы 5**

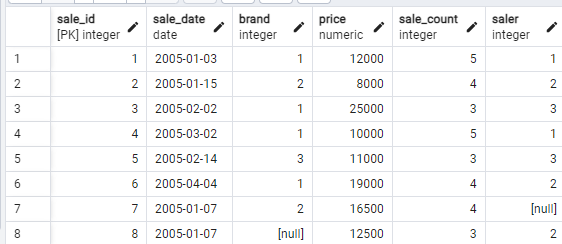
**Таблица salers**



**Таблица brands**



**Таблица sales**



**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", 'Выше среднего' AS "текстовый параметр"

FROM sales

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

WHERE sales.price > (SELECT AVG(price) FROM sales)

UNION ALL

SELECT salers.saler\_name AS "имя объекта", sales.price AS "числовой параметр", 'Ниже среднего' AS "текстовый параметр"

FROM sales

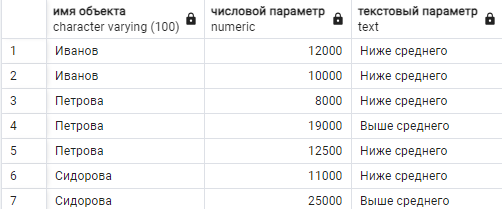
JOIN salers ON sales.saler = salers.saler\_id

WHERE sales.price < (SELECT AVG(price) FROM sales)

ORDER BY "имя объекта";

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

**Результат**



**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.sale\_count AS

"числовой параметр", 'Наивысший' AS "текстовый параметр"

FROM sales

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

WHERE sales.sale\_count = (SELECT MAX(sale\_count) FROM sales)

UNION

SELECT brands.brand\_name AS "имя объекта", sales.sale\_count AS

"числовой параметр", 'Низший' AS "текстовый параметр"

FROM sales

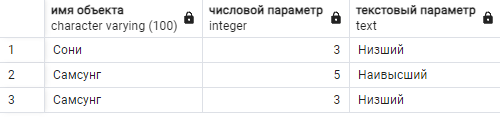
JOIN brands ON sales.brand = brands.brand\_id

WHERE sales.sale\_count = (SELECT MIN(sale\_count) FROM sales)

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**



**Задание 3**

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

**SQL код для задания:**

SELECT

sales.sale\_id AS "Номер продажи",

salers.saler\_name AS "Имя продавца",

brands.brand\_name AS "Марка",

sales.price AS "Цена",

sales.sale\_count AS "Количество продаж",

sales.sale\_date AS "Дата продажи"

FROM

sales

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

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

UNION

SELECT

NULL AS "Номер продажи",

salers.saler\_name AS "Имя продавца",

NULL AS "Марка",

NULL AS "Цена",

NULL AS "Количество продаж",

NULL AS "Дата продажи"

FROM

salers

WHERE NOT EXISTS (

SELECT 1

FROM sales

WHERE sales.saler = salers.saler\_id

)

ORDER BY "Имя продавца" DESC;

**Пояснение:** запрос показывает все продажи с соответствующими продавцами и марками.

**Результат**



**Задание 4**

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

**SQL код для задания:**

SELECT sale\_id, sale\_date, price

FROM sales

WHERE price > 10000

INTERSECT

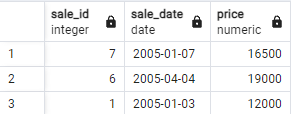
SELECT sale\_id, sale\_date, price

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**



**Задание 5**

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

**SQL код для задания:**

SELECT sale\_id, sale\_date, price

FROM sales

WHERE price > 10000

EXCEPT

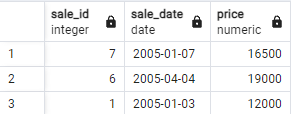
SELECT sale\_id, sale\_date, price

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**



**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

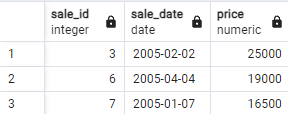
FROM sales

WHERE price > 15000;

SELECT \* FROM limited\_sales\_view;

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

**Результат**



**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

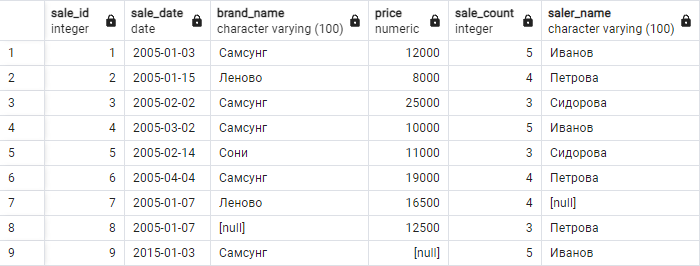
FULL JOIN brands b ON s.brand = b.brand\_id

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**



**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

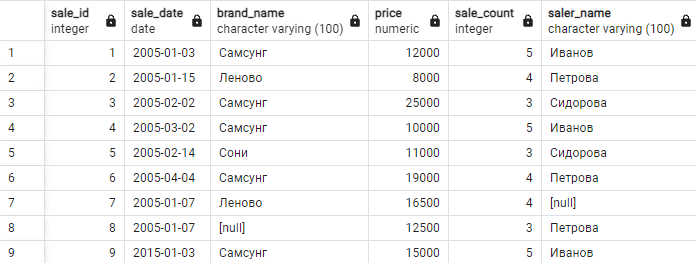
SET price = 15000

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**



**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

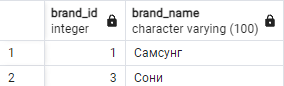
FROM brands

WHERE brand\_name != 'Леново';

SELECT \* FROM brands\_view;

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

**Результат**



**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**

