

# Базы данных и SQL. Обучение в записи

## Урок 6. Семинар: SQL – выборка данных, сортировка, агрегатные функции

### ЧАСТЬ I.

Домашнее задание

Домашнее задание

===== ТАБЛИЦА 1: ПРОДАВЦЫ (SALESPEOPLE) =====

snum	sname	city	comm
1001	Peel	London	.12
1002	Serres	San Jose	.13
1004	Motika	London	.11
1007	Rifkin	Barcelona	.15
1003	Axelrod	New York	.10

===== ТАБЛИЦА 2: ЗАКАЗЧИКИ (CUSTOMERS) =====

cnum	cname	city	rating	snum
2001	Hoffman	London	100	1001
2002	Giovanni	Rome	200	1003
2003	Liu	SanJose	200	1002
2004	Grass	Berlin	300	1002
2006	Clemens	London	100	1001
2008	Cisneros	SanJose	300	1007
2007	Pereira	Rome	100	1004

===== ТАБЛИЦА 3: ЗАКАЗЫ (ORDERS) =====

onum	amt	odate	cnum	snum
3001	18.69	10/03/1990	2008	1007
3003	767.19	10/03/1990	2001	1001
3002	1900.10	10/03/1990	2007	1004
3005	5160.45	10/03/1990	2003	1002
3006	1098.16	10/03/1990	2008	1007
3009	1713.23	10/04/1990	2002	1003
3007	75.75	10/04/1990	2004	1002
3008	4723.00	10/05/1990	2006	1001
3010	1309.95	10/06/1990	2004	1002
3011	9891.88	10/06/1990	2006	1001

**Используя операторы языка SQL, создать таблицы SALESPEOPLE, CUSTOMERS, ORDERS. Заполнить их данными.**

```
DROP DATABASE IF EXISTS seminar_3;  
CREATE DATABASE IF NOT EXISTS seminar_3;
```

```
USE seminar_3;
```

```
DROP TABLE IF EXISTS salespeople;  
CREATE TABLE salespeople  
(  
    snum INT,  
    sname VARCHAR(45),  
    city VARCHAR(45),  
    comm INT  
);
```

```
DROP TABLE IF EXISTS customers;  
CREATE TABLE customers  
(  
    cnum INT,  
    cname VARCHAR(45),  
    city VARCHAR(45),  
    rating INT,  
    snum INT  
);
```

```
DROP TABLE IF EXISTS orders;
```

```
CREATE TABLE orders
(
    onum INT,
    amt DECIMAL(10,2),
    odate DATE,
    cnum INT,
    snum INT
);
```

---

```
INSERT salespeople (snum, sname, city, comm)
VALUES
(1001, 'Peel', 'London', 12),
(1002, 'Serres', 'San Jose', 13),
(1004, 'Motika', 'London', 11),
(1007, 'Rifkin', 'Barcelona', 15),
(1003, 'Axelrod', 'New York', 10);
```

```
INSERT customers (cnum, cname, city, rating, snum)
VALUES
(2001, 'Hoffman', 'London', 100, 1001),
(2002, 'Giovanni', 'Rome', 200, 1003),
(2003, 'Liu', 'SanJose', 200, 1002),
(2004, 'Grass', 'Berlin', 300, 1002),
(2006, 'Clemens', 'London', 100, 1001),
(2008, 'Cisneros', 'SanJose', 300, 1007),
(2007, 'Pereira', 'Rome', 100, 1004);
```

```
INSERT orders (onum, amt, odate, cnum, snum)
VALUES
(3001, 18.69, '1990-03-10', 2008, 1007),
(3003, 767.19, '1990-03-10', 2001, 1001),
(3002, 1900.10, '1990-03-10', 2007, 1004),
(3005, 5160.45, '1990-03-10', 2003, 1002),
(3006, 1098.16, '1990-03-10', 2008, 1007),
(3009, 1713.23, '1990-04-10', 2002, 1003),
(3007, 75.75, '1990-04-10', 2004, 1002),
(3008, 4723.00, '1990-05-10', 2006, 1001),
(3010, 1309.95, '1990-06-10', 2004, 1002),
(3011, 9891.88, '1990-04-10', 2006, 1001);
```

```
SELECT * FROM customers;
SELECT * FROM orders;
SELECT * FROM salespeople;
```

## Домашнее задание

1. Напишите запрос, который вывел бы таблицу со столбцами в следующем порядке: city, sname, snum, comm. (к первой или второй таблице, используя SELECT)
2. Напишите команду SELECT, которая вывела бы оценку(rating), сопровождаемую именем каждого заказчика в городе San Jose. ("заказчики")
3. Напишите запрос, который вывел бы значения snum всех продавцов из таблицы заказов без каких бы то ни было повторений. (уникальные значения в "snum" "Продавцы")
- 4\*. Напишите запрос, который бы выбирал заказчиков, чьи имена начинаются с буквы G. Используется оператор "LIKE": ("заказчики") <https://dev.mysql.com/doc/refman/8.0/en/string-comparison-functions.html>
5. Напишите запрос, который может дать вам все заказы со значениями суммы выше чем \$1,000. ("Заказы", "amt" - сумма)
6. Напишите запрос который выбрал бы наименьшую сумму заказа. (Из поля "amt" - сумма в таблице "Заказы" выбрать наименьшее значение)
7. Напишите запрос к таблице "Заказчики", который может показать всех заказчиков, у которых рейтинг больше 100 и они находятся не в Риме.

1.

```
73 • SELECT city, sname, snum, comm
74 FROM salespeople;
```

	city	sname	snum	comm
•	London	Peel	1001	12
	San Jose	Serres	1002	13
	London	Motika	1004	11
	Barcelona	Rifkin	1007	15
	New York	Axelrod	1003	10

2.

```
76 • SELECT rating, cname
77 FROM customers
78 WHERE city = 'SanJose';
```

	rating	cname
	200	Liu
	300	Cisneros

3.

```
80 • SELECT DISTINCT snum FROM orders;
81
```

	snum
•	1007
	1001
	1004
	1002
	1003

4.

```
82 • SELECT * FROM customers
```

	cnum	cname	city	rating	snum
	2002	Giovanni	Rome	200	1003
	2004	Grass	Berlin	300	1002

5.

```

85 • SELECT * FROM orders
86     WHERE amt > 1000;
87

```

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

	onum	amt	odate	cnum	snum
▶	3002	1900.10	1990-03-10	2007	1004
	3005	5160.45	1990-03-10	2003	1002
	3006	1098.16	1990-03-10	2008	1007
	3009	1098.16	1990-04-10	2002	1003
	3008	4723.00	1990-05-10	2006	1001
	3010	1309.95	1990-06-10	2004	1002
	3011	9891.88	1990-04-10	2006	1001

6.

```

88 • SELECT MIN(amt) AS 'Наименьшая сумма заказа' FROM orders;

```

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

	Наименьшая сумма заказа
▶	18.69

7.

```

90 • SELECT * FROM customers
91     WHERE rating > 100 AND city <> 'Rome';

```

Result Grid | Filter Rows:  | Export: | Wr

	cnum	cname	city	rating	snum
▶	2003	Liu	SanJose	200	1002
	2004	Grass	Berlin	300	1002
	2008	Cisneros	SanJose	300	1007

## ЧАСТЬ II.

### Домашнее задание

Таблица для работы (из классной работы)

+ Параметры

	id	name	surname	specialty	seniority	salary	age
<input type="checkbox"/>	1	Вася	Васькин	начальник	40	100000	60
<input type="checkbox"/>	2	Петя	Петькин	начальник	8	70000	30
<input type="checkbox"/>	3	Катя	Каткина	инженер	2	70000	25
<input type="checkbox"/>	4	Саша	Сашкин	инженер	12	50000	35
<input type="checkbox"/>	5	Иван	Иванов	рабочий	40	30000	59
<input type="checkbox"/>	6	Петр	Петров	рабочий	20	25000	40
<input type="checkbox"/>	7	Сидор	Сидоров	рабочий	10	20000	35
<input type="checkbox"/>	8	Антон	Антонов	рабочий	8	19000	28
<input type="checkbox"/>	9	Юра	Юркин	рабочий	5	15000	25
<input type="checkbox"/>	10	Максим	Воронин	рабочий	2	11000	22
<input type="checkbox"/>	11	Юра	Галкин	рабочий	3	12000	24
<input type="checkbox"/>	12	Люся	Люськина	уборщик	10	10000	49

↑ ☐ Отметить все С отмеченными:

1. Отсортируйте поле “зарплата” в порядке убывания и возрастания
2. \*\* Отсортируйте по возрастанию поле “Зарплата” и выведите 5 строк с наибольшей заработной платой (возможен подзапрос)
3. Выполните группировку всех сотрудников по специальности , суммарная зарплата которых превышает 100000

```
32 • SELECT * FROM staff
33 ORDER BY salary DESC;
```

id	firstname	lastname	post	seniority	salary	age
1	Вася	Васькин	начальник	40	100000	60
2	Петя	Петькин	начальник	8	70000	30
3	Катя	Каткина	инженер	2	70000	25
4	Саша	Сашкин	инженер	12	50000	35
5	Иван	Иванов	рабочий	40	30000	59
6	Петр	Петров	рабочий	20	25000	40
7	Сидор	Сидоров	рабочий	10	20000	35
8	Антон	Антонов	рабочий	8	19000	28
9	Юра	Юркин	рабочий	5	15000	25
11	Юра	Галкин	рабочий	3	12000	24
10	Максим	Воронин	рабочий	2	11000	22
12	Люся	Люськина	уборщик	10	10000	49

1.

```
32 • SELECT * FROM staff
33 ORDER BY salary;
```

id	firstname	lastname	post	seniority	salary	age
12	Люся	Люськина	уборщик	10	10000	49
10	Максим	Воронин	рабочий	2	11000	22
11	Юра	Галкин	рабочий	3	12000	24
9	Юра	Юркин	рабочий	5	15000	25
8	Антон	Антонов	рабочий	8	19000	28
7	Сидор	Сидоров	рабочий	10	20000	35
6	Петр	Петров	рабочий	20	25000	40
5	Иван	Иванов	рабочий	40	30000	59
4	Саша	Сашкин	инженер	12	50000	35
2	Петя	Петькин	начальник	8	70000	30
3	Катя	Каткина	инженер	2	70000	25
1	Вася	Васькин	начальник	40	100000	60

2.

```
35 • SELECT post, SUM(salary) AS sum_salary FROM staff
36 GROUP BY post
37 HAVING sum_salary > 100000;
```

post	sum_salary
начальник	170000
инженер	120000
рабочий	132000

3.

## ЧАСТЬ III.

### 1. Ход выполнения задания 0.

Запрос на создание начальной таблицы, выполнять задания на основе этих данных.

Вы можете воспользоваться заготовкой fiddle <https://dbfiddle.uk/S8qGS2s4>

Или использовать код ниже для создания таблиц

```
USE seminar_3;
```

```
-- Создание таблицы Customers
```

```
DROP TABLE IF EXISTS Customers;
```

```
CREATE TABLE Customers (  
  customer_id INT PRIMARY KEY,  
  customer_name VARCHAR(255)  
);
```

```
-- Создание таблицы Orders
```

```
DROP TABLE IF EXISTS Orders;
```

```
CREATE TABLE Orders (  
  order_id INT PRIMARY KEY,  
  customer_id INT,  
  order_date DATE,  
  total_amount DECIMAL(10, 2),  
  shipper_id INT,  
  FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)  
);
```

```
-- Создание таблицы Shippers
```

```
DROP TABLE IF EXISTS Shippers;
```

```
CREATE TABLE Shippers (  
  shipper_id INT PRIMARY KEY,  
  shipper_name VARCHAR(255)  
);
```

```
-- Создание таблицы Products
```

```
DROP TABLE IF EXISTS Products;
```

```
CREATE TABLE Products (  
  product_id INT PRIMARY KEY,  
  product_name VARCHAR(255),  
  category_id INT,  
  author VARCHAR(255),  
  price DECIMAL(10, 2)  
);
```

```
-- Создание таблицы OrderDetails
```

```
DROP TABLE IF EXISTS OrderDetails;
```

```
CREATE TABLE OrderDetails (  
  order_detail_id INT PRIMARY KEY,  
  order_id INT,
```

```

product_id INT,
quantity INT,
FOREIGN KEY (order_id) REFERENCES Orders(order_id),
FOREIGN KEY (product_id) REFERENCES Products(product_id)
);

-- Создание таблицы Categories
DROP TABLE IF EXISTS Categories;

CREATE TABLE Categories (
category_id INT PRIMARY KEY,
category_name VARCHAR(255)
);

-- Наполнение таблиц данными
INSERT INTO Customers (customer_id, customer_name)
VALUES
(1, 'Иван Иванов'), (2, 'Мария Смирнова'), (3, 'Алексей Попов'), (4,
'Наталья Кузнецова'), (5, 'Дмитрий Васильев'),
(6, 'Ольга Петрова'), (7, 'Андрей Сидоров'), (8, 'Елена Алексеева'),
(9, 'Сергей Морозов'), (10, 'Ирина Фёдорова'),
(11, 'Андрей Иванов'), (12, 'Екатерина Мартынова');

INSERT INTO Shippers (shipper_id, shipper_name)
VALUES
(1, 'СДЕК'), (2, 'Почта России'), (3, 'ПЭК');

INSERT INTO Categories (category_id, category_name)
VALUES
(1, 'Художественная литература'), (2, 'Наука'), (3, 'Мистика');

INSERT INTO Products (product_id, product_name, category_id, author, price)
VALUES
(1, '1984', 1, 'Джордж Оруэлл', 250),
(2, 'Убить пересмешника', 1, 'Харпер Ли', 300),
(3, 'Великий Гэтсби', 1, 'Фрэнсис Скотт Фицджеральд', 200),
(4, 'Краткая история времени', 2, 'Стивен Хокинг', 320),
(5, 'Собака Баскервиль', 3, 'Артур Конан Дойл', 350),
(6, 'Моби Дик', 1, 'Герман Мелвилл', 400),
(7, 'Скотный двор', 1, 'Джордж Оруэлл', 220),
(8, 'Похвала Каталонии', 1, 'Джордж Оруэлл', 180),
(9, 'Дневник Анны Франк', 1, 'Анна Франк', 300),
(10, 'Краткая история времени', 2, 'Стивен Хокинг', 320);

-- Добавление заказов и деталей заказов с реалистичным распределением
INSERT INTO Orders (order_id, customer_id, order_date, total_amount, shipper_id)
VALUES
(1, 1, '2023-01-10', 750, 1),
(2, 3, '2023-01-12', 820, 2),
(3, 2, '2023-01-15', 600, 3),
(4, 4, '2023-02-01', 670, 1),
(5, 6, '2023-02-05', 550, 2),
(6, 3, '2023-02-10', 400, 3),
(7, 7, '2023-03-01', 320, 1),
(8, 8, '2023-03-05', 500, 2),

```

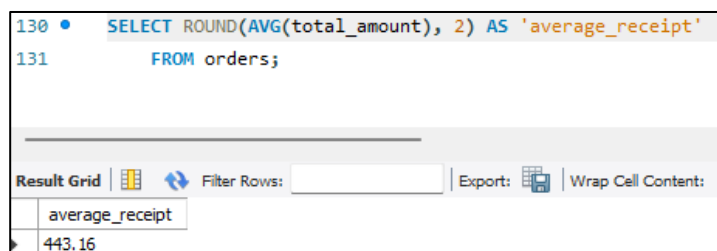
```
(9, 1, '2023-03-10', 270, 3),
(10, 5, '2023-03-15', 350, 1),
(11, 4, '2023-03-20', 420, 2),
(12, 9, '2023-04-01', 300, 3),
(13, 5, '2023-04-05', 220, 1),
(14, 6, '2023-04-10', 500, 2),
(15, 8, '2023-04-15', 450, 3),
(16, 10, '2023-05-01', 350, 1),
(17, 4, '2023-05-05', 250, 2),
(18, 9, '2023-05-10', 300, 3),
(19, 7, '2023-05-15', 400, 1);
```

```
INSERT INTO OrderDetails (order_detail_id, order_id, product_id, quantity)
VALUES
(1, 1, 1, 1), (2, 1, 2, 1), (3, 1, 4, 1),
(4, 2, 3, 1), (5, 2, 5, 1), (6, 2, 7, 1),
(7, 3, 6, 1), (8, 3, 9, 1),
(9, 4, 1, 1), (10, 4, 8, 1),
(11, 5, 2, 1), (12, 5, 5, 1),
(13, 6, 3, 1),
(14, 7, 4, 1), (15, 7, 9, 1),
(16, 8, 10, 1),
(17, 9, 5, 1), (18, 9, 6, 1),
(19, 10, 7, 1);
```

## 2. Ход выполнения задания 1.

### **Средний чек по заказам**

Выведите средний чек (*average\_receipt*) для заказов.



```
130 • SELECT ROUND(AVG(total_amount), 2) AS 'average_receipt'
131 FROM orders;
```

average_receipt
443.16

## 3. Ход выполнения задания 2.

### **Количество заказов по перевозчикам**

Выведите имя перевозчика, месяц и год заказа, а также количество уникальных заказов, доставленных каждым перевозчиком в каждый месяц и год.

Подсказка:

- Используйте *EXTRACT*, *GROUP BY* и *COUNT*.



133	•	SELECT	
134		Shippers.shipper_name,	
135		EXTRACT(MONTH FROM Orders.order_date) AS month,	
136		EXTRACT(YEAR FROM Orders.order_date) AS year,	
137		COUNT(DISTINCT Orders.order_id) AS unique_orders_count	
138		FROM Orders	
139		JOIN Shippers ON Orders.shipper_id = Shippers.shipper_id	
140		GROUP BY	
141		Shippers.shipper_name,	
142		EXTRACT(YEAR FROM Orders.order_date),	
143		EXTRACT(MONTH FROM Orders.order_date);	
144			

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
shipper_name	month	year	unique_orders_count
Почта России	1	2023	1
Почта России	2	2023	1
Почта России	3	2023	2
Почта России	4	2023	1
Почта России	5	2023	1
ПЭК	1	2023	1
ПЭК	2	2023	1
ПЭК	3	2023	1
ПЭК	4	2023	2
ПЭК	5	2023	1
СДЕК	1	2023	1
СДЕК	2	2023	1
СДЕК	3	2023	2
СДЕК	4	2023	1
СДЕК	5	2023	2

#### 4. Ход выполнения задания 3.

##### *Продукты по цене*

Выведите название и цену книг, которые стоят более 100 единиц. Отсортируйте результат по цене в порядке убывания и ограничьте вывод 5 результатами.

Подсказка:

- Используйте *WHERE*, *ORDER BY* и *LIMIT*.

145	•	SELECT product_name, price FROM Products	
146		WHERE price > 100	
147		ORDER BY price DESC	
148		LIMIT 5;	

Result Grid	Filter Rows:	Export:
product_name	price	
Моби Дик	400.00	
Собака Баскервилей	350.00	
Краткая история времени	320.00	
Краткая история времени	320.00	
Убить пересмешника	300.00	

#### 5. Ход выполнения задания 4.

##### *Количество заказов по категориям книг*

Выведите категорию книг и количество заказов, которые содержат книги этой категории. Показать только те категории, которые имеют больше одного заказа.

Подсказка:

- Используйте *GROUP BY*, *HAVING* и агрегатную функцию *COUNT*.

```
150 • SELECT Categories.category_name, COUNT(DISTINCT Orders.order_id) AS order_count
151 FROM OrderDetails
152 JOIN Products ON OrderDetails.product_id = Products.product_id
153 JOIN Categories ON Products.category_id = Categories.category_id
154 JOIN Orders ON OrderDetails.order_id = Orders.order_id
155 GROUP BY Categories.category_name
156 HAVING COUNT(DISTINCT Orders.order_id) > 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

category_name	order_count
Мистика	3
Наука	3
Художественная литература	9

## 6. Ход выполнения задания 5.

### Сумма и количество заказов по клиентам

Выведите идентификатор клиента, имя клиента, сумму и количество его заказов. Отсортируйте результат по идентификатору клиента.

Подсказка:

- Используйте *GROUP BY* и агрегатные функции *SUM* и *COUNT*.

```
158 • SELECT
159     Customers.customer_id,
160     Customers.customer_name,
161     SUM(Orders.total_amount) AS total_amount,
162     COUNT(Orders.order_id) AS order_count
163 FROM Orders
164 JOIN Customers ON Orders.customer_id = Customers.customer_id
165 GROUP BY Customers.customer_id, Customers.customer_name
166 ORDER BY Customers.customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

customer_id	customer_name	total_amount	order_count
1	Иван Иванов	1020.00	2
2	Мария Смирнова	600.00	1
3	Алексей Попов	1220.00	2
4	Наталья Кузнецова	1340.00	3
5	Дмитрий Васильев	570.00	2
6	Ольга Петрова	1050.00	2
7	Андрей Сидоров	720.00	2
8	Елена Алексеева	950.00	2
9	Сергей Морозов	600.00	2
10	Ирина Фёдорова	350.00	1