# Агрегатные функции, Вложенные запросы, Соединения

Агрегатные функции в SQL  
  
Агрегатные функции – это специальные функции, которые выполняют вычисления над группами строк и возвращают одно значение. Они часто используются в сочетании с оператором GROUP BY для агрегации данных по определенным критериям.  
  
Основные агрегатные функции:  
- COUNT() – подсчитывает количество строк в выборке.  
- SUM() – вычисляет сумму значений столбца.  
- AVG() – находит среднее значение.  
- MAX() – возвращает наибольшее значение в столбце.  
- MIN() – возвращает наименьшее значение в столбце.  
  
Примеры использования:  
```sql  
SELECT COUNT(\*) FROM employees; -- Подсчет общего количества сотрудников   
SELECT AVG(salary) FROM employees; -- Средняя зарплата сотрудников   
SELECT department, SUM(salary) FROM employees GROUP BY department; -- Суммарная зарплата по отделам   
SELECT MIN(price), MAX(price) FROM products; -- Минимальная и максимальная цена товаров   
```  
  
При работе с агрегатными функциями важно учитывать, что они игнорируют NULL-значения, кроме COUNT(\*), который считает все строки, включая NULL.  
  
Вложенные запросы (Subqueries)  
  
Вложенные запросы – это SQL-запросы, выполняемые внутри другого запроса. Они могут использоваться в SELECT, WHERE, FROM и HAVING.  
  
Типы вложенных запросов:  
1. В WHERE (фильтрация по результату вложенного запроса)  
```sql  
SELECT \* FROM employees   
WHERE salary > (SELECT AVG(salary) FROM employees);  
```  
  
2. В FROM (использование вложенного запроса как таблицы)  
```sql  
SELECT subquery.department, subquery.avg\_salary   
FROM (SELECT department, AVG(salary) AS avg\_salary FROM employees GROUP BY department) AS subquery   
WHERE avg\_salary > 50000;  
```  
  
3. В SELECT (использование подзапросов в столбцах)  
```sql  
SELECT name, salary,   
 (SELECT AVG(salary) FROM employees) AS avg\_salary   
FROM employees;  
```  
  
4. С EXISTS (проверка наличия записей в подзапросе)  
```sql  
SELECT \* FROM customers c   
WHERE EXISTS (SELECT \* FROM orders o WHERE o.customer\_id = c.id);  
```  
  
5. С IN (проверка принадлежности значений)  
```sql  
SELECT \* FROM employees   
WHERE department\_id IN (SELECT id FROM departments WHERE name = 'IT');  
```  
  
Соединения таблиц (SQL JOINs)  
  
JOIN используется для объединения данных из нескольких таблиц.  
  
Основные типы JOIN:  
- INNER JOIN – выбирает только строки, имеющие соответствие в обеих таблицах.  
- LEFT JOIN – выбирает все строки из левой таблицы и соответствующие строки из правой. Если соответствия нет, вставляется NULL.  
- RIGHT JOIN – аналогично LEFT JOIN, но возвращает все строки из правой таблицы.  
- FULL JOIN – объединяет данные из обеих таблиц, заполняя отсутствующие значения NULL.  
- CROSS JOIN – возвращает декартово произведение строк двух таблиц.  
- SELF JOIN – соединение таблицы с самой собой.  
  
Примеры:  
1. INNER JOIN (только совпадающие строки)  
```sql  
SELECT employees.name, departments.name AS department   
FROM employees   
INNER JOIN departments ON employees.department\_id = departments.id;  
```  
  
2. LEFT JOIN (все строки из левой таблицы + совпадения из правой)  
```sql  
SELECT customers.name, orders.order\_id   
FROM customers   
LEFT JOIN orders ON customers.id = orders.customer\_id;  
```  
  
3. RIGHT JOIN (все строки из правой таблицы + совпадения из левой)  
```sql  
SELECT employees.name, departments.name AS department   
FROM employees   
RIGHT JOIN departments ON employees.department\_id = departments.id;  
```  
  
4. FULL JOIN (все данные из обеих таблиц)  
```sql  
SELECT customers.name, orders.order\_id   
FROM customers   
FULL JOIN orders ON customers.id = orders.customer\_id;  
```  
  
5. CROSS JOIN (декартово произведение)  
```sql  
SELECT employees.name, projects.name   
FROM employees   
CROSS JOIN projects;  
```  
  
6. SELF JOIN (соединение таблицы с самой собой)  
```sql  
SELECT e1.name AS employee, e2.name AS manager   
FROM employees e1   
JOIN employees e2 ON e1.manager\_id = e2.id;  
```  
  
Дополнительные нюансы JOIN  
  
ON vs. USING  
Оператор USING используется, если у соединяемых таблиц есть столбцы с одинаковыми именами.  
```sql  
SELECT employees.name, departments.name   
FROM employees   
JOIN departments USING (department\_id);  
```  
  
Нестандартные соединения с условиями  
```sql  
SELECT e1.name, e2.name AS colleague   
FROM employees e1   
JOIN employees e2 ON e1.department\_id = e2.department\_id AND e1.id != e2.id;  
```  
  
Этот запрос показывает сотрудников одного отдела, исключая их самих.  
  
Вывод:  
- Агрегатные функции используются для обработки больших объемов данных (SUM, AVG, COUNT и др.).  
- Вложенные запросы позволяют делать сложные выборки, используя данные из других запросов.  
- JOIN используется для объединения данных из нескольких таблиц, упрощая анализ информации.  
  
Все эти механизмы – основа работы с реляционными базами данных, позволяя эффективно управлять и анализировать данные.