



Запросы по типу возвращаемого результата

- **Скалярные** (Scalar Subqueries) – возвращают 1 значение

```
SELECT MAX(orderid) FROM "Sales"."Orders";
```

1	11 077
---	--------

- **Многозначные** (Multi-Valued Subqueries) – возвращают список значений

```
SELECT custid
FROM "Sales"."Customers"
WHERE country = 'Mexico';
```

1	2
2	3
3	13
4	58
5	80

- **Табличные** (Table-Valued Subqueries) – возвращают таблицу

```
SELECT orderid, productid, (unitprice * qty) AS LineTotal
FROM "Sales"."OrderDetails" od;
```

orderid	productid	linetotal
1	10 248	11 168.00
2	10 248	42 98.00
3	10 248	72 174.00
4	10 249	14 167.40
5	10 249	51 1,696.00
6	10 250	41 77.00

Использование подзапросов

Использование подзапросов



Вложенные и коррелированные подзапросы

Предложение	Scalar Subqueries	Multi-Valued Subqueries	Table-Valued Subqueries
SELECT	Вычисляемый столбец	-	-
FROM	-	Derived Table (именованная таблица из одного столбца) CTE (временное представление)	
WHERE	Фильтрация записей на основе сравнения со скалярной величиной	Фильтрация записей с использованием предикатов [NOT] IN, SOME, ALL, ANY, [NOT] EXISTS*	Фильтрация записей с использованием предиката [NOT] EXISTS*
GROUP BY	-	-	-
HAVING	Фильтрация групп на основе сравнения со скалярной величиной	-	-
ORDER BY	-	-	-

Вложенный подзапрос

Внешний запрос:

```
SELECT orderid, productid, unitprice,
qty
FROM "Sales"."OrderDetails"
WHERE orderid = ( );
```

Внутренний запрос:

```
SELECT MAX(orderid) AS lastorder
FROM "Sales"."Orders";
```

Коррелированный подзапрос

Внешний запрос:

```
SELECT orderid, empid, orderdate
FROM "Sales"."Orders" AS 01
WHERE orderdate = ( );
```

Внутренний запрос:

```
SELECT MAX(orderdate)
FROM "Sales"."Orders" AS 02
WHERE 02.empid = 01.empid;
```



Использование скалярного вложенного запроса



```
SELECT orderid, productid, unitprice, qty
FROM "Sales"."OrderDetails"
WHERE orderid =
  (SELECT MAX(orderid) FROM "Sales"."Orders");
```

```
SELECT orderid,
       orderdate,
       (SELECT min(orderdate)
        FROM "Sales"."Orders") AS FirstDate
  FROM "Sales"."Orders";
```

- Если вложенный запрос возвращает пустую выборку (empty set), результат конвертируется в NULL



Использование многозначного вложенного запроса с одним столбцом

- Список заказов, сделанных клиентами из 'Mexico'

```
SELECT custid, orderid
  FROM "Sales"."Orders"
 WHERE custid IN ( SELECT custid
                      FROM "Sales"."Customers"
                     WHERE country = 'Mexico');
```

- Список деталей заказов, включающих продукты, цена которых превышает минимальные цены продуктов в каждой категории

```
SELECT *
  FROM "Sales"."OrderDetails" od
 WHERE unitprice > ALL( SELECT min(unitprice)
                           FROM "Production"."Products" p
                          GROUP BY categoryid );
```

Использование многозначного вложенного запроса с несколькими столбцами



- Вы можете использовать IN (ANY, ALL, SOME) для сравнения по нескольким столбцам:
 - В предложении WHERE необходимо сгруппировать имена столбцов в скобках
 - Количество сгруппированных столбцов должно соответствовать целевому списку подзапроса, и иметь те же типы данных

Список заказов 76 клиента для которых город доставки не совпадает с городом клиента:

```
SELECT custid, orderid, shipcity
  FROM "Sales"."Orders"
 WHERE (custid, shipcity) != ALL (SELECT custid, city
                                    FROM "Sales"."Customers")
   AND custid = 76;
```



Использование скалярного коррелированного запроса

- Список заказов с общей суммой каждого заказа

```
SELECT orderid, o.custid,
       (SELECT SUM(unitprice*qty)
        FROM "Sales"."OrderDetails" od
       WHERE od.orderid = o.orderid) AS OrderTotal
  FROM "Sales"."Orders" o;
```

- Список последних заказов, оформленных каждым сотрудником

```
SELECT orderid, empid, orderdate
  FROM "Sales"."Orders" AS 01
 WHERE orderdate =
       (SELECT MAX(orderdate)
        FROM "Sales"."Orders" AS 02
       WHERE 02.empid = 01.empid)
 ORDER BY empid, orderdate;
```

Использование предиката EXISTS



WHERE [NOT] EXISTS (subquery)

- Если вложенный запрос возвращает хотя бы одну запись - EXISTS возвращает TRUE
- Если вложенный запрос не возвращает ни одной записи - EXISTS возвращает FALSE

```
SELECT custid, companyname
FROM "Sales"."Customers" AS c
WHERE EXISTS (
    SELECT 1
    FROM "Sales"."Orders" AS o
    WHERE c.custid=o.custid);
```

```
SELECT custid, companyname
FROM "Sales"."Customers" AS c
WHERE NOT EXISTS (
    SELECT 1
    FROM "Sales"."Orders" AS o
    WHERE c.custid=o.custid);
```

Использование табличных выражений

Производные таблицы (Derived Tables)



- Производные таблицы — это именованные выражения запроса, созданные во внешнем операторе SELECT
 - Не хранится в базе данных — представляет собой виртуальную реляционную таблицу
 - Область видимости — запрос, в котором она определена

Производная таблица должна

- Иметь псевдоним
- Иметь уникальные имена для всех столбцов
- Не использовать предложение ORDER BY (без LIMIT или OFFSET/FETCH)
- Не ссылаться несколько раз в одном запросе

Производная таблица может

- Использовать внутренние или внешние псевдонимы для столбцов
- Обращаться к параметрам и/или переменным
- Быть вложенной в другие производные таблицы

Использование производных таблиц

```
SELECT orderyear, COUNT(DISTINCT custid) AS cust_count
FROM (
    SELECT date_part('year',orderdate), custid
    FROM "Sales"."Orders" ) AS derived_year(orderyear, custid)
GROUP BY orderyear;
```

	orderyear	cust_count
1	2 006	67
2	2 007	86
3	2 008	81

```
SELECT orderyear, COUNT(DISTINCT custid) AS cust_count
FROM (
    SELECT date_part('year',orderdate) AS orderyear, custid
    FROM "Sales"."Orders" AS o
    WHERE o.empid=9 ) AS derived_year
GROUP BY orderyear;
```

	orderyear	cust_count
1	2 006	5
2	2 007	16
3	2 008	16

Общие табличные выражения

- CTE (Common Table Expression) — это именованные табличные выражения, используемые для разбиения сложных запросов на простые части
 - Можно представить как определения временных таблиц, существующих только для одного запроса
- CTE определяется в предложении WITH
 - Все возвращаемые столбцы должны иметь уникальные имена
 - CTE поддерживают несколько определений
 - На CTE допустимо ссылаться несколько раз в одном запросе

```
WITH CTE_year AS
(
  SELECT date_part('year',orderdate) AS orderyear, custid
  FROM "Sales"."Orders"
  WHERE empid = 2
)
SELECT orderyear, COUNT(DISTINCT custid) AS cust_count
FROM CTE_year
GROUP BY orderyear;
```



	orderyear	cust_count
1	2 006	15
2	2 007	35
3	2 008	34



Рекурсивный запрос WITH

- Для определения рекурсивного запроса WITH используется указание **RECURSIVE**
- Рекурсивный запрос WITH включает:
 - не рекурсивную часть
 - оператор UNION (или UNION ALL)
 - рекурсивную часть
- Только в рекурсивной части можно обратиться к результату запроса

```
WITH RECURSIVE t(n)
AS (
  SELECT (1)
  UNION ALL
  SELECT n+1
  FROM t WHERE n < 10
)
SELECT n FROM t;
```

