

Запросы с группировкой





- Работают с набором значений столбца
- Возвращают скалярное значение (без имени столбца)
- Игнорируют NULL-значения, кроме COUNT(*)
- Могут использоваться в:
 - Предложениях SELECT, HAVING и ORDER BY
 - Часто используются с предложением GROUP BY

```
SELECT AVG(unitprice::numeric) AS avg_price,
          MIN(qty)AS min_qty,
          MAX(discount) AS max_discount
FROM "Sales"."OrderDetails";
```

<u>-</u>	¹²³ avg_price ¹	123 min_qty 13	max_discount T:
1	26.2185	1	0,25





Общие • sum • min max avq count string_agg

Статистические corr covar regr_ stddev_ var

Дополнительные grouping array_agg • bit_and • bit or bool_and • bool_or • json_agg • json_object_agg

Использование DISTINCT в агрегатных функциях



- Используйте DISTINCT с агрегатными функциями, чтобы агрегировать только уникальные значения.
- Агрегаты с DISTINCT исключают повторяющиеся значения, а не строки (в отличие от SELECT DISTINCT).

```
SELECT empid
      , date_part('year',orderdate) AS orderyear
      , COUNT(custid) AS all_custs
      , COUNT(DISTINCT custid) AS unique_custs
FROM "Sales"."Orders"
                                                              orderyear Tall_custs Tall_read unique_custs
                                                      empid
GROUP BY empid
                                                                    2 006
                                                                                 26
         , date_part('year',orderdate);
                                                                    2 006
                                                                    2 006
                                                                                              16
                                                                    2 006
                                                                                              26
                                                                    2 006
                                                                                              10
                                                            5
```



Обработка NULL-значений

- Большинство агрегатных функций игнорируют NULL:
 - COUNT(<column>) игнорирует NULL-значения
 - COUNT(*) подсчитывает все строки
- NULL может давать неправильные результаты (например, при использовании AVG)
 - Используйте **COALESCE** для замены NULL-значений перед агрегированием

<u> </u>	¹²³ c1	V:	¹²³ c2 7 ‡
1		1	25
2		2	30
3		3	[NULL]
4		4	34
5		5	[NULL]
6		6	12
7		7	24
8		8	[NULL]
9		9	35

SELECT AVG(col2) AS AvgWithNULLs,

AVG(COALESCE(col2,0)) AS AvgWithoutNULL

FROM public.testnull;





Использование GROUP BY

Логический порядок	Предложение	Комментарий
5	SELECT	
1	FROM	
2	WHERE	
3	GROUP BY	Создает группы
4	HAVING	Отфильтровывает группы
6	ORDER BY	



Использование GROUP BY

```
SELECT <select_list>
FROM <table_source>
WHERE <search_condition>
GROUP BY <group_by_list>
HAVING <group_condition>;
```

- GROUP BY создает группы из исходных записей в соответствии с уникальной комбинацией значений, указанных в предложении GROUP BY
 - Детальные строки «теряются» после обработки предложения GROUP BY
 - Если запрос использует GROUP BY, все последующие этапы работают с группами, а не с исходными строками
 - HAVING, SELECT и ORDER BY должны возвращать одно значение для каждой группы.
 - Bce столбцы в SELECT, HAVING и ORDER BY должны появляться в предложении GROUP BY или быть входными данными для агрегатных выражений

GROUP BY Workflow

SELECT custid , COUNT(*) AS "Count(*)"
FROM "Sales"."Orders" o
WHERE empid IN (8, 9)
GROUP BY custid
HAVING COUNT(*) >5;

orderid empid empid

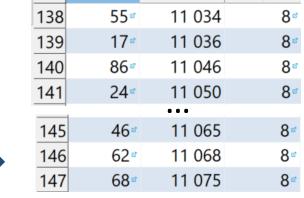


SELECT custid, orderid, empid
FROM "Sales"."Orders" o;

			· • ,
	custid T	orderid **	empid T
1	85≝	10 248	5 ₫
2	79战	10 249	6₫
3	34₫	10 250	4
4	84₫	10 251	3₫
		• • •	
826	20₫	11 072	∆ ♂

826	20₫	11 072	4 🗈
827	58₫	11 073	2 *
828	76◎	11 074	7₫
829	68₫	11 075	8 🗈
830	9₫	11 076	4 ∞

WHERE empid IN (8, 9)



custid



GROUP BY custid

	custid T:	count
1	87≝	5
2	54战	1
3	29战	2
4	71 战	5
	• • •	
63	91 🗈	1
64	58₫	1
65	8 🛚	1

HAVING COUNT(*) >5;



	custid count	
1	24	6
2	20₫	7

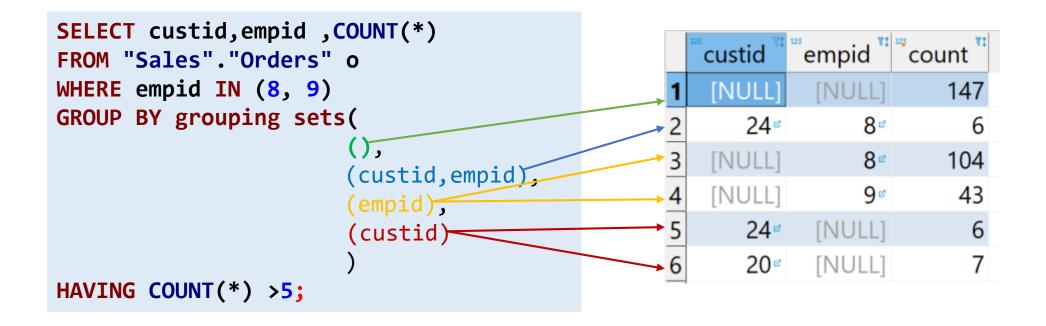


GROUPING SETS

- Оператор GROUPING SETS это расширение предложения GROUP BY
- GROUPING SETS группирует записи на основе задаваемых пользователем **наборов для группировки**
 - Наборы для группировки должны включать в себя столбцы, по которым вы группируете с помощью предложения **GROUP BY**
 - Набор для группировки обозначается списком столбцов, разделенных запятыми, заключенных в круглые скобки:









CUBE

- CUBE это расширение предложения GROUP BY
- CUBE позволяет создавать все возможные группы на основе указанных столбцов измерений

CUBE (c1,c2,c3)

```
(c1, c2, c3)
(c1, c2)
(c2, c3)
(c1,c3)
(c1)
(c2)
(c3)
```

```
SELECT custid,empid ,COUNT(*)
FROM "Sales"."Orders" o
WHERE empid IN (8, 9)
GROUP BY cube(custid, empid)
HAVING COUNT(*) >5;
```

	custid T	empid **	count
1	[NULL]	[NULL]	147
2	24₫	8 8	6
3	24	[NULL]	6
4	20₫	[NULL]	7
5	[NULL]	8 8	104
6	[NULL]	9₫	43



ROLLUP

- В отличие от CUBE, ROLLUP не генерирует все возможные группы на основе указанных столбцов
- ROLLUP предполагает иерархию входных столбцов и генерирует все группы, которые имеют смысл с учетом иерархии
 - По этой причине ROLLUP часто используется для создания промежуточных и общих итогов для отчетов

ROLLUP(c1,c2,c3)

```
(c1, c2, c3)
(c1, c2)
(c1)
()
```

<pre>SELECT custid,empid ,COUNT(*)</pre>
FROM "Sales"."Orders" o
WHERE empid IN (8, 9)
<pre>GROUP BY rollup(custid, empid)</pre>
HAVING COUNT(*) >5;

	custid Ta	empid **	count
1	[NULL]	[NULL]	147
2	24战	8 🗈	6
3	24₫	[NULL]	6
4	20₫	[NULL]	7



Функция GROUPING

- Функция GROUPING возвращает целочисленную битовую маску, показывающую, какие аргументы не вошли в текущий набор группирования
 - Бит равен 0 если аргумент является членом текущего набора группировок
 - Бит равен 1 если аргумент не является членом текущего набора группировок
- Аргументы функции GROUPING должны в точности соответствовать выражениям, заданным в предложении GROUP BY

