Запросы с группировкой

Агрегатные функции

- 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";
```

	avg_price	min_qty	max_discount
1	26.2185	1	0,25

Категории агрегатных функций

Общие	Статистические	Дополнительные
• sum • min • max • avg • count • string_agg	• corr • covar_ • regr_ • stddev_ • var_	• grouping • array_agg • bit_and • bit_or • bool_and • bool_or • json_agg • json_object_agg

https://postgrespro.ru/docs/postgresql/14/functions-aggregate

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"
GROUP BY  empid
       , date_part('year',orderdate);
```

	empid	orderyear	all_custs	unique_custs
1	1	2 006	26	22
2	2	2 006	16	15
3	3	2 006	18	16
4	4	2 006	31	26
5	5	2 006	11	10

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

	c1	c2
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;
```

	avgwithnulls	avgwithoutnull
1	26,6666666667	17,7777777778

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.
  - Все 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;
```

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

	custid	orderid	empid
1	85	10 248	5
2	79	10 249	6
3	34	10 250	4
4	84	10 251	3
...			
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	orderid	empid
138	55	11 034	8
139	17	11 036	8
140	86	11 046	8
141	24	11 050	8
...			
145	46	11 065	8
146	62	11 068	8
147	68	11 075	8

```
GROUP BY custid
```

	custid	count
1	87	5
2	54	1
3	29	2
4	71	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:

```
SELECT c1, c2, aggregate_function(c3)
FROM table_name
GROUP BY
GROUPING SETS ( (c1, c2),   -- Группировка по столбцам c1 и c2
                (c1),       -- Группировка по столбцу c1
                (c2),       -- Группировка по столбцу c2
                () );       -- Отсутствие группировки (Общий итог)
```

GROUPING SETS

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

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

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	empid	count
1	[NULL]	[NULL]	147
2	24	8	6
3	24	[NULL]	6
4	20	[NULL]	7
5	[NULL]	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)
()
```

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

	custid	empid	count
1	[NULL]	[NULL]	147
2	24	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

```
select custid, empid, COUNT(*) as qw_Orders,
       GROUPING(custid, empid)
from "Sales"."Orders"
where custid in (24,44)
group by
GROUPING sets( (), --11
               (custid, empid), --00
               (custid),  --01
              (empid)     --10
              );
```

	custid	empid	qw_orders	grouping
1	[NULL]	[NULL]	34	3
...				
13	24	4	2	0
14	24	7	2	0
15	44	4	3	0
16	44	[NULL]	15	1
17	24	[NULL]	19	1
18	[NULL]	8	9	2
19	[NULL]	9	1	2
20	[NULL]	7	5	2