Использование операторов для работы с наборами

Что изучим

OVER – оконные функции

Оконные функции делятся на:

- Агрегатные функции
- Ранжирующие функции
- Функции смещения
- Аналитические функции

Data Science Academy

Агрегатные функции

SUM/ AVG / COUNT/ MIN/ MAX

Data Science Academy

Предложение **OVER** помогает «открыть окно», т.е. определить строки, с которым будет работать та или иная функция.

```
select distinct
product_cd,
sum(avail_balance) over () as sum_bal
from account;
```

PRODUCT_CD	SUM_BAL
SAV	170754.46
MM	170754.46
CHK	170754.46
SBL	170754.46
BUS	170754.46
CD	170754.46

Data Science Academy

Предложение **PARTITION BY** не является обязательным, но дополняет **OVER** и показывает, как именно мы разделяем строки, к которым будет применена функция.

```
select
cust_id,
sum(avail_balance) over (partition by cust_id) as sum_bal
from account
where cust_id = 1;
```

CUST_ID	SUM_BAL
1	4557.75
1	4557.75
1	4557.75

Data Science Academy

Использование всех агрегатных функции в запросе

```
select
cust_id,
product_cd,
open_emp_id,
open_branch_id,
sum(avail_balance) over (partition by product_cd) as all_sum,
max(avail_balance) over (partition by product_cd) as all_max,
avg(avail_balance) over (partition by cust_id) as cust_avg,
min(avail_balance) over (partition by cust_id, product_cd) as min_prod_cust
from account
order by cust_id;
```

CUST_ID	PRODUCT_CD	OPEN_EMP_ID	OPEN_BRANCH_ID	ALL_SUM	ALL_MAX	CUST_AVG	MIN_PROD_CUST
10	BUS	16	4	9345.55	9345.55	11787.56	0
11	BUS	10	2	9345.55	9345.55	9345.55	9345.55
6	CD	1	1	19500	10000	5061.185	10000
7	CD	10	2	19500	10000	5000	5000
9	CD	1	1	19500	10000	3657.07333333333	1500
1	CD	10	2	19500	10000	1519.25	3000

Data Science Academy

Можно комбинировать оконные функции OVER

```
sum(avail_balance) over (partition by product_cd)
/ count(avail_balance) over (partition by cust_id)
```

Data Science Academy

Функции ранжирования

ROW_NUMBER

RANK

DENSE_RANK

NTILE

ROW_NUMBER – нумерует строки в результирующем наборе.

RANK - присваивает ранг для каждой строки, если найдутся одинаковые значения, то следующий ранг присваивается с пропуском.

DENSE_RANK - присваивает ранг для каждой строки, если найдутся одинаковые значения, то следующий ранг присваивается без пропуска.

NTILE – помогает разделить результирующий набор на группы.

Data Science Academy

Для понимания, проранжируем таблицу по убыванию дней открытия:

```
select
    cust_id,
    product_cd,
    open_emp_id,
    open_branch_id,
    row_number() over (order by open_date DESC) as row_date,
    rank() over (order by open_date DESC) as rank_date,
    dense_rank() over (order by open_date DESC) as denserank_date,
    ntile(4) over (order by open_date DESC) as ntile_date
from account;
```

CUST_ID	PRODUCT_CD	OPEN_EMP_ID	OPEN_BRANCH_ID	ROW_DATE	RANK_DATE	DENSERANK_DATE	NTILE_DATE
6	CD	1	1	1	1	1	1
9	MM	1	1	2	2	2	1
4	MM	1	1	3	3	3	1
1	CD	10	2	4	4	4	1
9	CD	1	1	5	4	4	1
11	BUS	10	2	6	6	5	1
13	SBL	13	3	7	7	6	2

Data Science Academy

Решим задачу. Нужно вывести клиентов, у которых максимальный остаток по продуктам

```
select
*
from
(select
   cust_id,
   product_cd,
   row_number() over (partition by product_cd order by avail_balance DESC) as rn
from account)
where rn = 1;
```

CUST_ID	PRODUCT_CD	RN
11	BUS	1
6	CD	1
12	CHK	1
9	MM	1
4	SAV	1
13	SBL	1

Data Science Academy

Функции смещения

LAG / LEAD/

FIRST_VALUE / LAST_VALUE

Data Science Academy

Оконные функции смещения помогут нам, когда необходимо обратиться к строке в наборе данных из окна, относительно текущей строки с некоторым смещением. Проще говоря, узнать, какое значение (событие/ дата) идет после/до текущей строки. Похоже на отличную штуку в предобработке лога данных.

LAG — смещение назад.
LEAD — смещение вперед.
FIRST_VALUE — найти первое значение набора данных.
LAST_VALUE — найти последнее значение набора данных.
LAG и LEAD имеют следующие аргументы:
- Столбец, значение которого необходимо вернуть
- На сколько строк выполнить смешение (дефолт =1)

Data Science Academy

Найдём продукты, которые были открыты. Сместим данные по датам и продуктам

```
select
cust_id,
open_date,
lead(product_cd) over (partition by cust_id order by open_date) as lead_opendt,
lag(open_date) over (partition by cust_id order by open_date) as lag_opendt
from account;
```

CUST_ID	OPEN_DATE	LEAD_OPENDT	LAG_OPENDT
1	15.01.2000	SAV	
1	15.01.2000	CD	15.01.2000
1	30.06.2004		15.01.2000
2	12.03.2001	SAV	
2	12.03.2001		12.03.2001
3	23.11.2002	MM	

Data Science Academy

Функции FIRST_VALUE и LAST_VALUE

Эти функции позволяют для каждой строки выдать первое значение ее окна и последнее.

```
select
cust_id,
first_value(avail_balance) over (partition by cust_id order by avail_balance) as first_v,
last_value(avail_balance) over (partition by cust_id order by avail_balance) as last_v
from account;
```

CUST_ID	FIRST_V	LAST_V
1	500	500
1	500	1057.75
1	500	3000

Data Science Academy

Поэтому FIRST и LAST value можно использовать в комбинации с DENSE_RANK

```
select
cust_id,
first_value(avail_balance) over (partition by cust_id order by avail_balance) as first_v,
last_value(avail_balance) over (partition by cust_id order by avail_balance) as last_v,
min(avail_balance) keep (dense_rank first order by avail_balance) over (partition by cust_id) as f_bal
max(avail_balance) keep (dense_rank last order by avail_balance) over (partition by cust_id) as l_bal
from account;
```

CUST_ID	FIRST_V	LAST_V	F_BAL	L_BAL
1	500	500	500	3000
1	500	1057.75	500	3000
1	500	3000	500	3000

Data Science Academy

Выполнение вычислений для строк в группе по плавающему окну (интервалу)

Для некоторых аналитических функций, например, агрегирующих, можно дополнительно указать объем строк, участвующих в вычислении, выполняемом для каждой строки в группе. Этот объем, своего рода контекст строки, называется "окном", а границы окна могут задаваться различными способами.

{ROWS / RANGE} {{UNBOUNDED / выражение} PRECEDING / CURRENT ROW }

{ROWS / RANGE}
BETWEEN
{{UNBOUNDED PRECEDING / CURRENT ROW /
{UNBOUNDED / выражение 1}{PRECEDING / FOLLOWING}}
AND
{{UNBOUNDED FOLLOWING / CURRENT ROW /
{UNBOUNDED / выражение 2}{PRECEDING / FOLLOWING}}

windowing_clause	Description
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW	Последняя строка в окне изменяется с изменением текущей строки (по умолчанию)
RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING	Первая строка в окне изменяется с изменением текущей строки
RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING	Все строки включены в окно независимо от текущей строки

Рассмотрим пример:

```
SELECT
    cust_id,
    product_cd,
    avail_balance,
    SUM(avail_balance)
    OVER (PARTITION BY product_cd ORDER BY open_date
    ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) sum_bal_1,

    SUM(avail_balance)
    OVER (PARTITION BY product_cd ORDER BY open_date
    RANGE BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) sum_bal_2,

    SUM(avail_balance)
    OVER (PARTITION BY product_cd ORDER BY open_date
    RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) sum_bal_3
FROM account
where product_cd = 'CD';
```

CUST_ID	PRODUCT_CD	AVAIL_BALANCE	SUM_BAL_1	SUM_BAL_2	SUM_BAL_3
7	CD	5000	5000	19500	19500
1	CD	3000	8000	14500	19500
9	CD	1500	9500	14500	19500
6	CD	10000	19500	10000	19500