Аналитические функции ORACLE

Графеева Н.Г.

2016

Аналитика, трудно отображаемая средствами стандартного SQL

- Подсчет нарастающих итогов (показать нарастающие итоги по зарплате построчно для каждого сотрудника);
- Подсчет процентов в группе (какой процент от общей зарплаты составляет зарплата отдельного сотрудника);
- Выборка первых N сотрудников с наибольшими зарплатами;
- Подсчет скользящего среднего (получить среднее значение по предыдущим N строкам);
- Выполнение ранжирующих запросов (показать ранг зарплаты сотрудника среди других сотрудников)

Назначение аналитических функций

- Они расширяют язык SQL так, что подобные операции не только проще записываются, но и ***быстрее выполняются*** по сравнению с использованием чистого языка SQL. Говорят, что эти расширения сейчас изучаются комитетом ANSI SQL с целью включения в спецификацию языка SQL.

Основные группы аналитических функций

- В ORACLE имеется по крайней мере 26 аналитических функций, которые достаточно условно могут быть разбиты на 4 группы:
- функции ранжирования;
- Функции агрегирования;
- оконные функции;
- функции, позволяющие “заглянуть” вперед или “оглянуться” назад.

Контекст использования аналитических функций

- Имя Функции(<аргумент>,< аргумент >, ...)
- OVER
- (
- [конструкция фрагментации]
- [конструкция упорядочения]
- [конструкция окна]
-)

Конструкция фрагментации

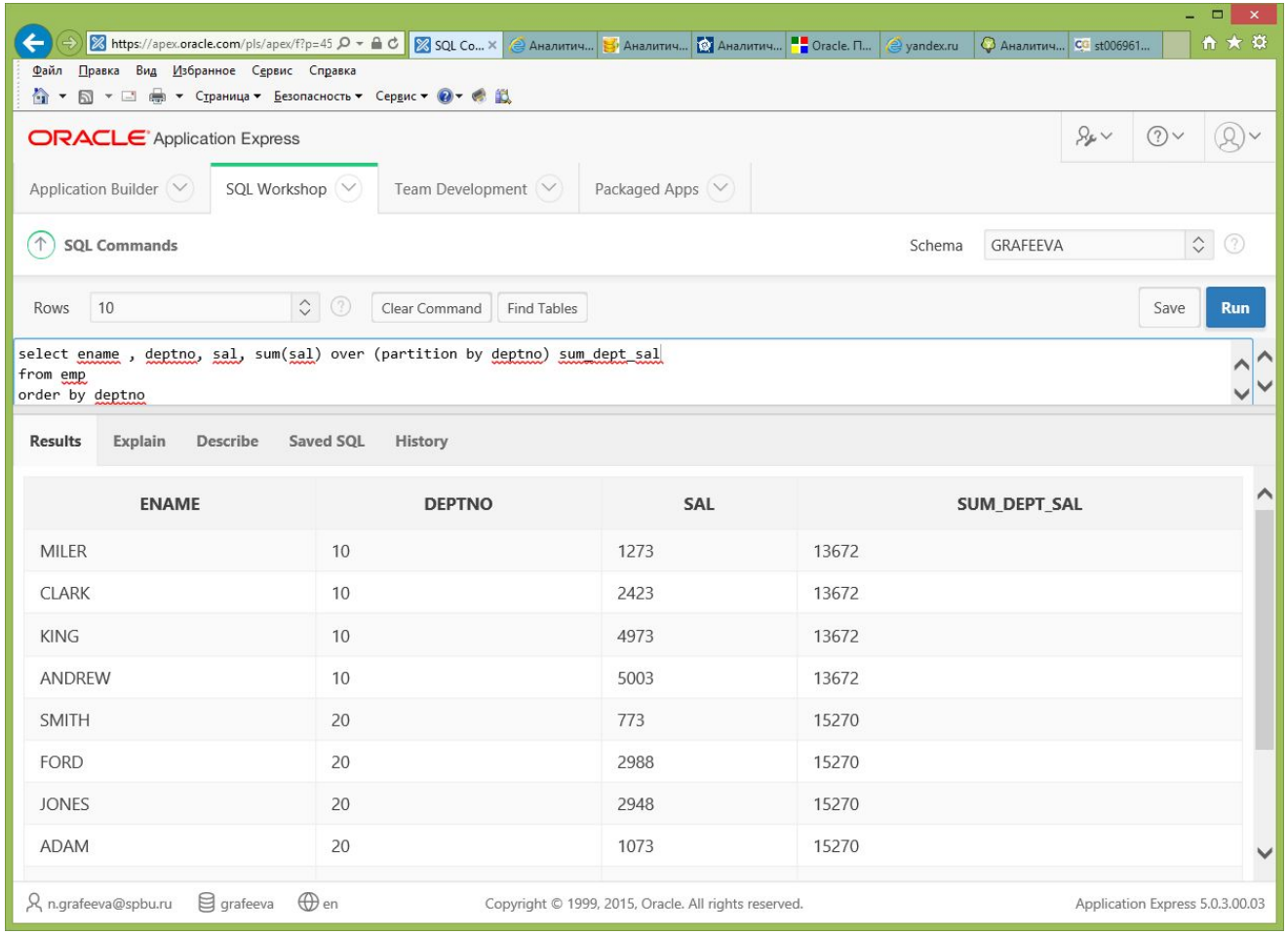
- **PARTITION BY выражение [, выражение] [, выражение]**

- Конструкция задает область применения аналитических функций (группы).
- Если не указать конструкцию фрагментации, все результирующее множество считается одной группой.

Пример 1 - запрос

```
select ename , deptno, sal,
sum(sal) over (partition by deptno) sum_dept_sal
from emp
order by deptno
```

Пример 1 - результат

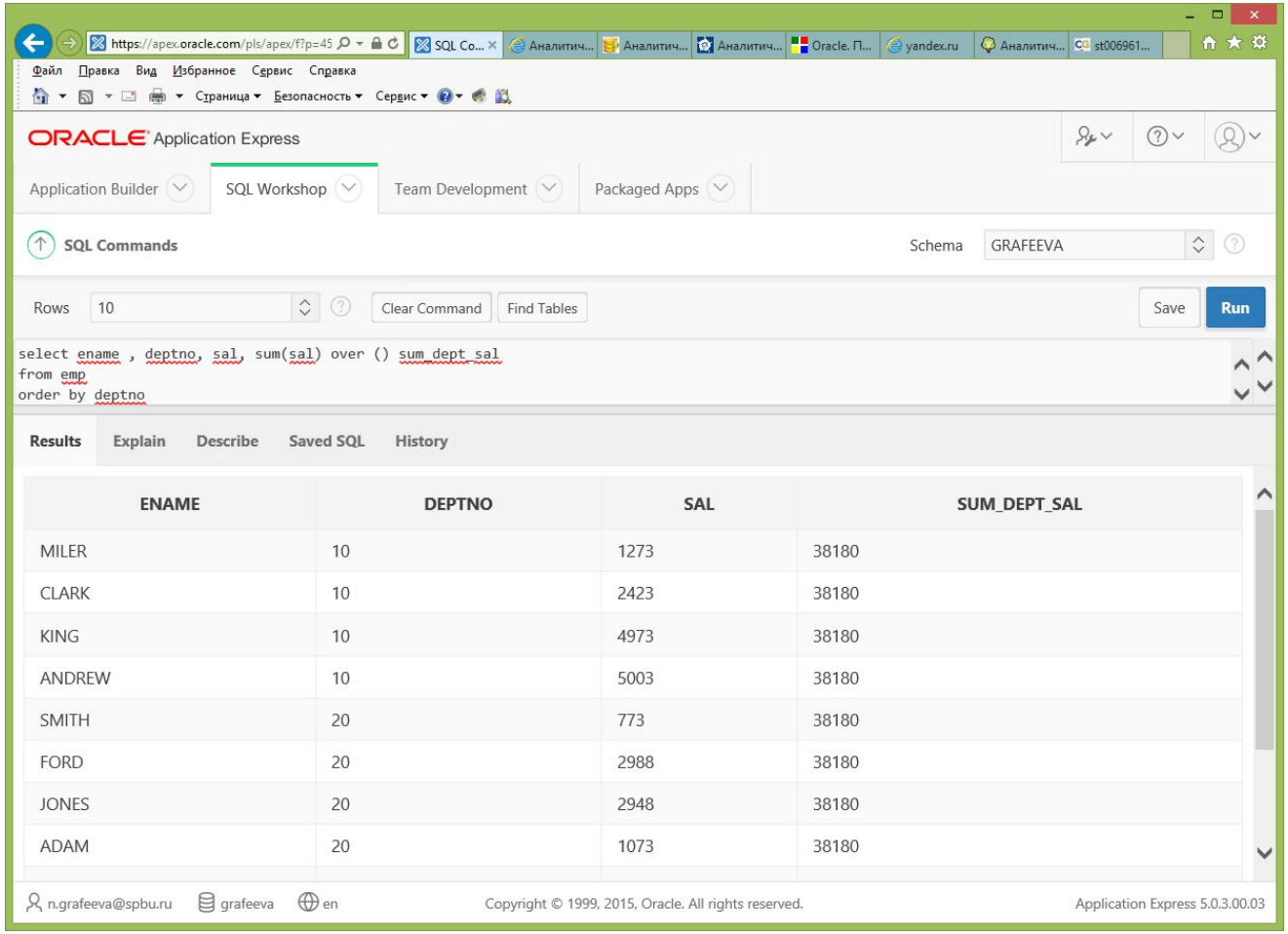


ENAME	DEPTNO	SAL	SUM_DEPT_SAL
MILER	10	1273	13672
CLARK	10	2423	13672
KING	10	4973	13672
ANDREW	10	5003	13672
SMITH	20	773	15270
FORD	20	2988	15270
JONES	20	2948	15270
ADAM	20	1073	15270

Пример 2 - запрос

```
select ename , deptno, sal,
        sum(sal) over () sum_dept_sal
from emp
order by deptno
```

Пример 2 - результат



ENAME	DEPTNO	SAL	SUM_DEPT_SAL
MILER	10	1273	38180
CLARK	10	2423	38180
KING	10	4973	38180
ANDREW	10	5003	38180
SMITH	20	773	38180
FORD	20	2988	38180
JONES	20	2948	38180
ADAM	20	1073	38180

Конструкция упорядочения

- **ORDER BY выражение [, выражение] [, выражение] [[ASC][DESC]]**

- Согласно документации “задает критерий сортировки данных в каждой группе”. Однако в действительности дело не только в сортировке…

Пример 3 - запрос

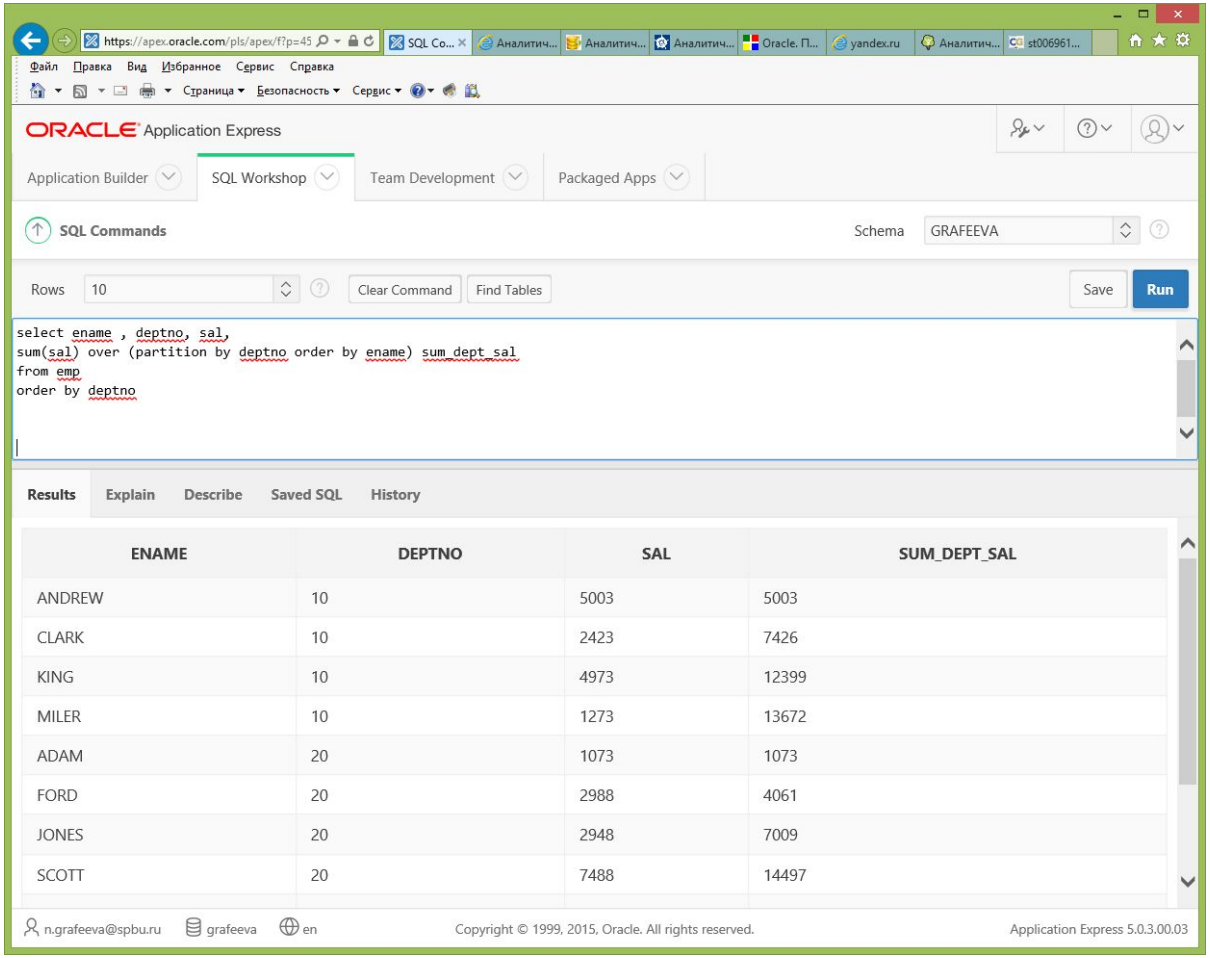
select ename , deptno, sal,

sum(sal) over (partition by deptno **order by ename**) sum_dept_sal

from emp

order by deptno

Пример 3 - результат

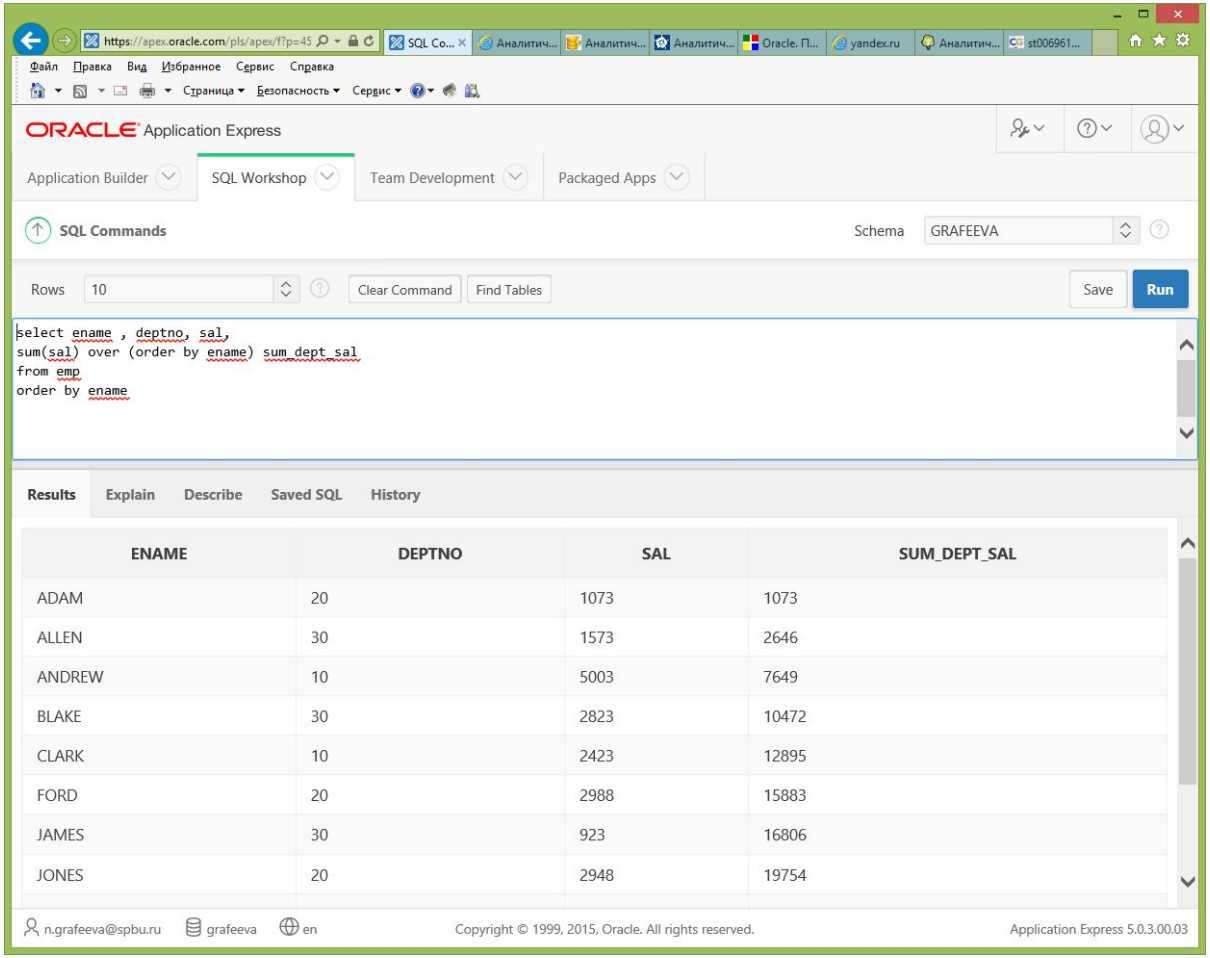


ENAME	DEPTNO	SAL	SUM_DEPT_SAL
ANDREW	10	5003	5003
CLARK	10	2423	7426
KING	10	4973	12399
MILER	10	1273	13672
ADAM	20	1073	1073
FORD	20	2988	4061
JONES	20	2948	7009
SCOTT	20	7488	14497

Пример 4 – запрос (нарастающие итоги по зарплате)

```
select ename , deptno, sal,
sum(sal) over (order by ename) sum_dept_sal
from emp
order by ename
```

Пример 4 - результат

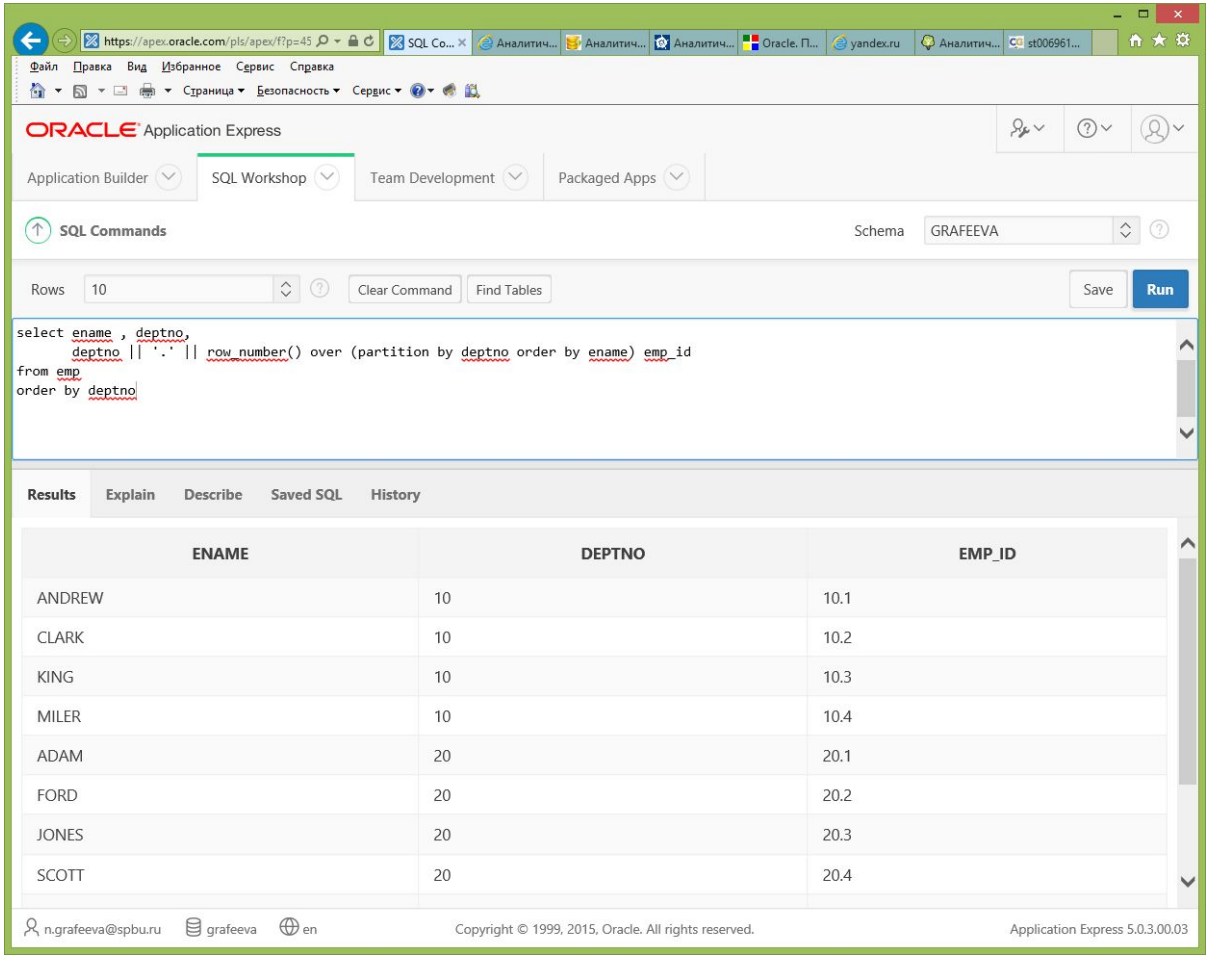


ENAME	DEPTNO	SAL	SUM_DEPT_SAL
ADAM	20	1073	1073
ALLEN	30	1573	2646
ANDREW	10	5003	7649
BLAKE	30	2823	10472
CLARK	10	2423	12895
FORD	20	2988	15883
JAMES	30	923	16806
JONES	20	2948	19754

Пример 5 - запрос

```
select ename , deptno,
      deptno || '.' || row_number() over (partition by deptno
    order by ename) emp_id
from emp
order by deptno
```

Пример 5 - резльтат



ENAME	DEPTNO	EMP_ID
ANDREW	10	10.1
CLARK	10	10.2
KING	10	10.3
MILER	10	10.4
ADAM	20	20.1
FORD	20	20.2
JONES	20	20.3
SCOTT	20	20.4

Конструкция окна

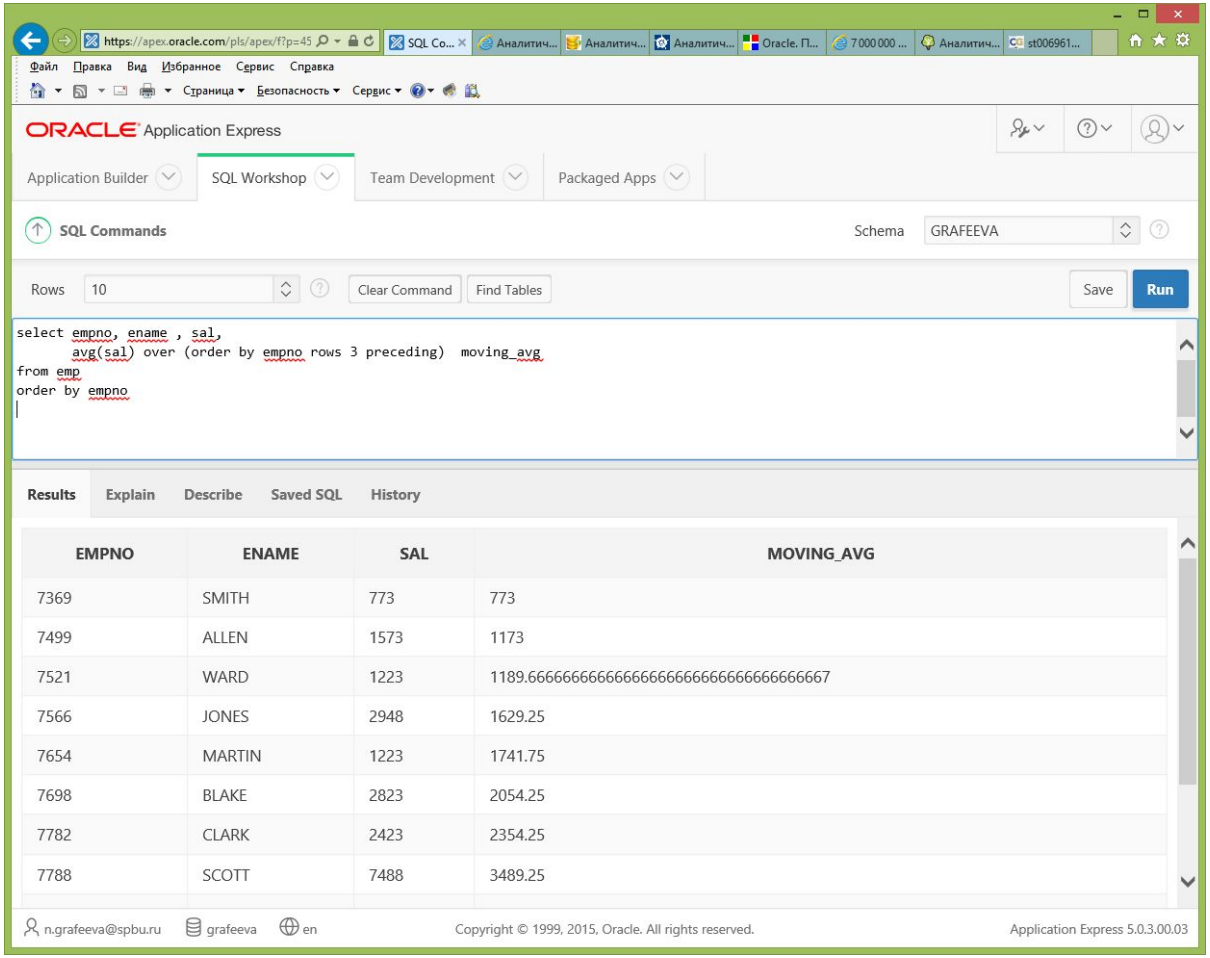
- Конструкция окна позволяет задать перемещающееся или жестко привязанное окно (набор) данных в пределах группы, с которым будет работать аналитическая функция. Возможны два типа задания конструкции окна – смещение (**ROWS**) и задание диапазона (**RANGE**). Допустимы следующие варианты задания окна:
- **ROWS n PRECEDING**
- **ROWS n FOLLOWING**
- **RANGE UNBOUNDED PRECEDING**
- **RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW**
- **RANGE n PRECEDING**
- **И т.п.**

- Например, конструкция **ROWS n PRECEDING** означает: применять аналитическую функцию к каждой строке данной группы с текущей строки до (n-1) предыдущей. Конструкция **RANGE n PRECEDING** означает: применять аналитическую функцию к каждой строке данной группы у которых значения (по которым работает конструкция **ORDER BY**) попадают в диапазон от (значения в текущей строке – n) до текущего значения.

Пример 6- запрос (смещение)

```
select empno, ename , sal,
avg(sal) over (order by empno rows 3 preceding)
   moving_avg
from emp
order by empno
```

Пример 6- результат

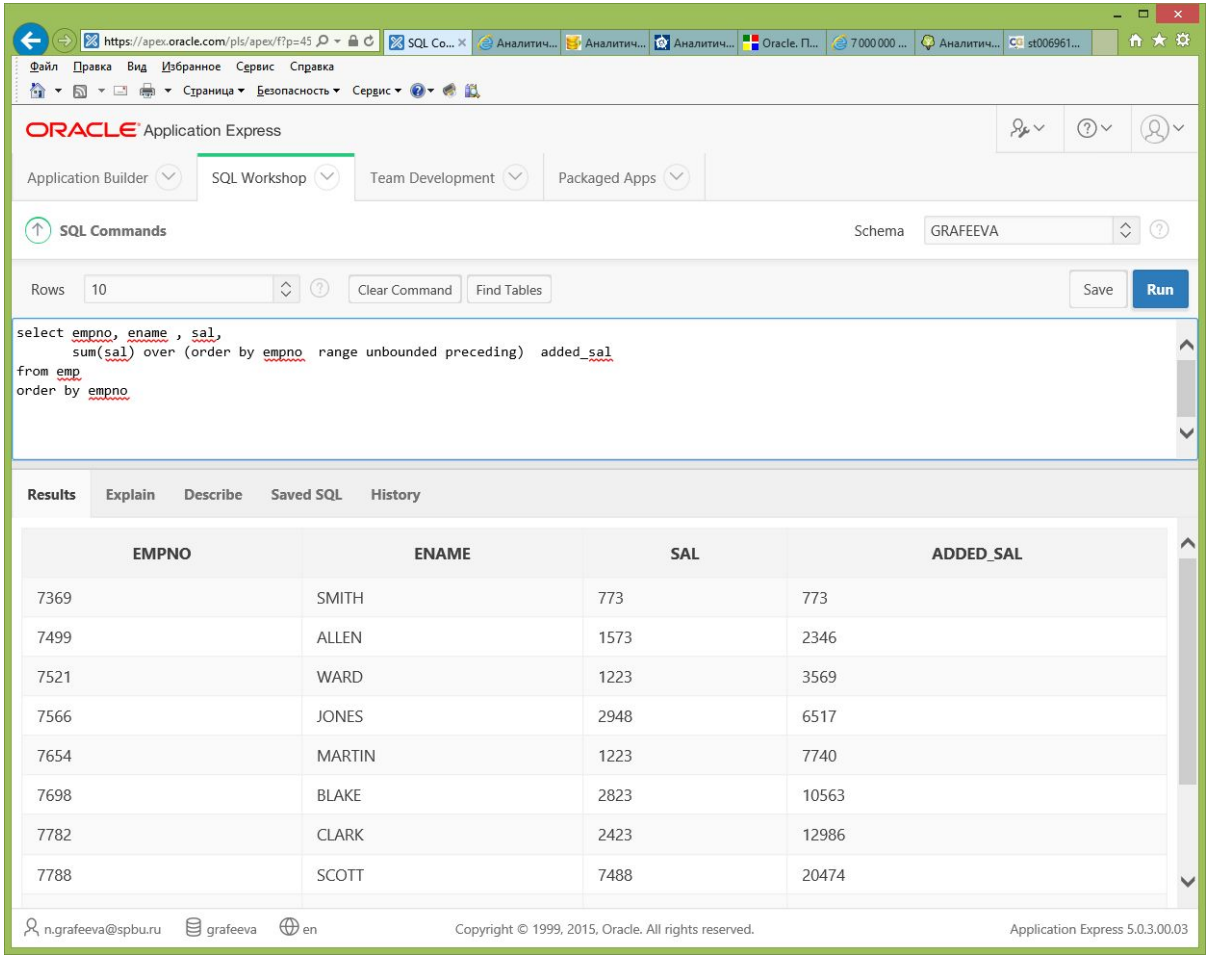


EMPNO	ENAME	SAL	MOVING_AVG
7369	SMITH	773	773
7499	ALLEN	1573	1173
7521	WARD	1223	1189.6666666666666666666666666666666667
7566	JONES	2948	1629.25
7654	MARTIN	1223	1741.75
7698	BLAKE	2823	2054.25
7782	CLARK	2423	2354.25
7788	SCOTT	7488	3489.25

Пример 7 – запрос (окно диапазона)

```
select empno, ename , sal,
sum(sal) over (order by empno  range
                    unbounded preceding)  added_sal
from emp
order by empno
```

Пример 7 - результат

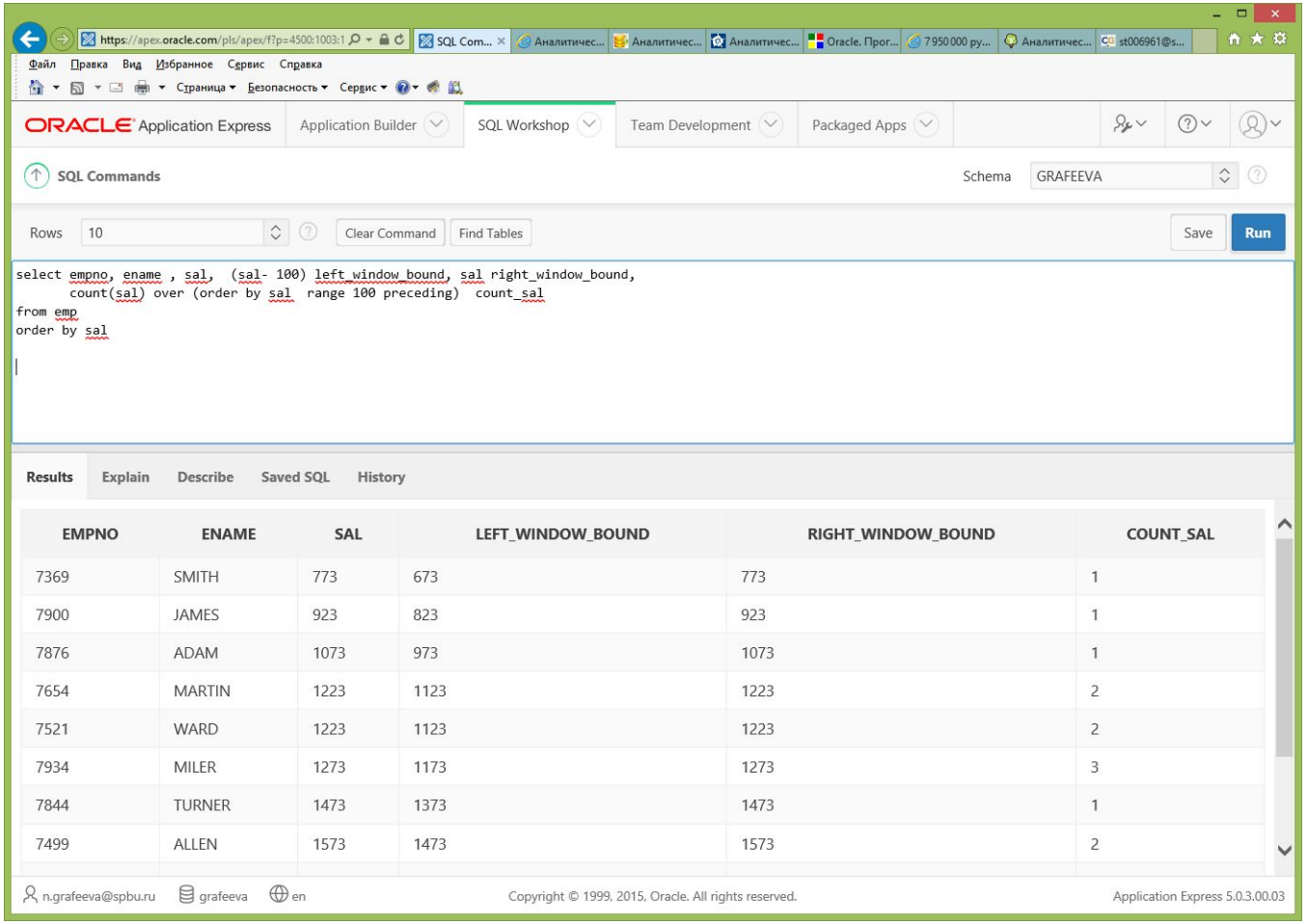


EMPNO	ENAME	SAL	ADDED_SAL
7369	SMITH	773	773
7499	ALLEN	1573	2346
7521	WARD	1223	3569
7566	JONES	2948	6517
7654	MARTIN	1223	7740
7698	BLAKE	2823	10563
7782	CLARK	2423	12986
7788	SCOTT	7488	20474

Пример 8 – запрос(численное задание дипазона)

```
select empno, ename ,
   sal,
  (sal- 100) left_window_bound,
   sal          right_window_bound,
   count(sal) over (order by sal  range 100
   preceding)  count_sal
from emp
order by sal
```

Пример 8 – результат



EMPNO	ENAME	SAL	LEFT_WINDOW_BOUND	RIGHT_WINDOW_BOUND	COUNT_SAL
7369	SMITH	773	673	773	1
7900	JAMES	923	823	923	1
7876	ADAM	1073	973	1073	1
7654	MARTIN	1223	1123	1223	2
7521	WARD	1223	1123	1223	2
7934	MILER	1273	1173	1273	3
7844	TURNER	1473	1373	1473	1
7499	ALLEN	1573	1473	1573	2

Группы аналитических функций

- Rankings and percentiles
- Lag/lead analysis
- Window calculations
- First/last analysis

Предназначение аналитических функций

Группа	Предназначение
Rankings and percentiles	Рассчитывает ранги и проценты в результирующем наборе
Window calculations	Агрегирует значения в рамках выделенного окна
Lag/lead analysis	Поиск значения по смещению от текущей строки
First/last analysis	Первые и последние значения в выделенной группе

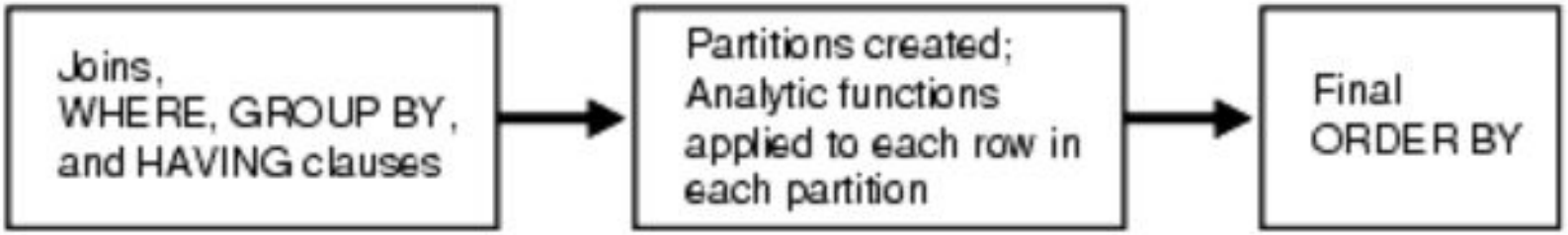
Порядок обработки аналитических функций

Обработка запросов с помощью аналитических функций происходит в три этапа:

- Во-первых, выполняются все соединения, WHERE, GROUP BY и HAVING.
- Во-вторых, результирующий набор обрабатывается аналитическими функциями.
- В-третьих, если запрос имеет опцию ORDER BY, выполняется итоговая обработка результирующего множества.

Порядок обработки аналитических функций

Processing Order



Rankings and percentiles analysis

- RANK
- DENSE_RANK
- RATIO_TO_REPORT
- CUME_DIST
- PERCENT_RANK
- NTILE
- ROW_NUMBER

Синтаксис для использования

- RANK ( ) OVER ([partition_clause] order_by_clause)
- DENSE_RANK ( ) OVER ([partition_clause] order_by_clause)
- RATIO_TO_REPORT () ([partition_clause] order_by_clause)
- CUME_DIST ( ) OVER ([partition_clause] order_by_clause)
- PERCENT_RANK ( ) OVER ([partition_clause] order_by_clause)
- NTILE (exp) OVER ([partition_clause] order_by_clause)
- ROW_NUMBER ( ) OVER ([partition_clause] order_by_clause)

Пример 9 (RANK – вычисляет относительный ранг каждой

```
SELECT JOB,
       SUM(SAL) SUM_SAL,
       RANK() OVER (ORDER BY SUM(SAL) ASC) RENC_SAL_ASC,
       RANK() OVER (ORDER BY SUM(SAL) DESC) RENC_SAL_DESC
FROM EMP
GROUP BY JOB
```

Results | Explain | Describe | Saved SQL | History

JOB	SUM_SAL	RENC_SAL_ASC	RENC_SAL_DESC
MANAGER	16610	5	1
SALESMAN	11280	4	2
PRESIDENT	10020	3	3
CLERK	8580	2	4
ANALYST	6640	1	5

5 rows returned in 0.01 seconds Download

Пример 10 (RANK)

```
SELECT JOB,
       SUM(NVL(COMM, 0)) SUM_COMM,
       RANK() OVER (ORDER BY SUM(NVL(COMM, 0)) ASC) RENC_COMM_ASC,
       RANK() OVER (ORDER BY SUM(NVL(COMM, 0)) DESC) RENC_COMM_DESC
FROM EMP
GROUP BY JOB
```

Results | Explain | Describe | Saved SQL | History

JOB	SUM_COMM	RENC_COMM_ASC	RENC_COMM_DESC
SALESMAN	4400	5	1
PRESIDENT	0	1	2
MANAGER	0	1	2
CLERK	0	1	2
ANALYST	0	1	2

5 rows returned in 0.01 seconds Download

Пример 11(DENSE_RENK – вычисляет “плотный” ранг каждой строки без промежутков)

```
SELECT JOB,
       SUM(NVL(COMM, 0)) SUM_COMM,
       DENSE_RANK() OVER (ORDER BY SUM(NVL(COMM, 0)) ASC) RENC_COMM_ASC,
       DENSE_RANK() OVER (ORDER BY SUM(NVL(COMM, 0)) DESC) RENC_COMM_DESC
FROM EMP
GROUP BY JOB
```

Results Explain Describe Saved SQL History

JOB	SUM_COMM	RENC_COMM_ASC	RENC_COMM_DESC
SALESMAN	4400	2	1
PRESIDENT	0	1	2
MANAGER	0	1	2
CLERK	0	1	2
ANALYST	0	1	2

5 rows returned in 0.01 seconds Download

Пример 12(RATIO_TO_REPORT –вычисляет соотношение текущего значение к сумме значений по всей группе)

Rows 10 | Clear Command | Find Tables | Save | Run

```
select ename, deptno, sal,
       ratio_to_report(sal) over (partition by deptno) AS sal_ratio
from emp
order by deptno;
```

Results | Explain | Describe | Saved SQL | History

ENAME	DEPTNO	SAL	SAL_RATIO
MILER	10	1273	.093110005851375073142188414277355178467
CLARK	10	2423	.177223522527794031597425394967817437098
KING	10	4973	.363736102984201287302516091281451141018
ANDREW	10	5003	.365930368636629607957870099473376243417
SMITH	20	773	.050622134905042567125081859855926653569
FORD	20	2988	.195677799607072691552062868369351669941
JONES	20	2948	.193058284217419777341191879502292075966
ADAM	20	1073	.070268500327439423706614276358873608382
SCOTT	20	7488	.490373280943025540275049115913555992142

Пример 13(CUME_DIST)

```
SELECT DEPTNO,
       JOB,
       COUNT(*) COUNT,
       CUME_DIST() OVER (PARTITION BY DEPTNO ORDER BY COUNT(*)) CUME_DIST_COUNT
FROM EMP
GROUP BY DEPTNO, JOB
```

Results | Explain | Describe | Saved SQL | History

10	PRESIDENT	2	1
10	MANAGER	2	1
10	CLERK	2	1
20	MANAGER	2	.3333333333333333333333333333333333333
20	ANALYST	4	1
20	CLERK	4	1
30	CLERK	2	.6666666666666666666666666666666666667
30	MANAGER	2	.6666666666666666666666666666666666667
30	SALESMAN	8	1

9 rows returned in 0.03 seconds Download

Определение CUME_DIST (в документации ORACLE

The **CUME_DIST** function (defined as the inverse of percentile in some statistical books) computes the position of a specified value relative to a set of values. The order can be ascending or descending. Ascending is the default. The range of values for **CUME_DIST** is from greater than 0 to 1. To compute the **CUME_DIST** of a value x in a set S of size N, you use the formula:

```
CUME_DIST(x) =  number of values in S coming before
   and including x in the specified order/ N
```

Пример 14 (NTILE – классифицирует группы по значению выражения)

```
SELECT JOB,
       SUM(SAL) SUM_SAL,
       NTILE (4) OVER (ORDER BY SUM(SAL) ASC) NTILE_SAL_ASC
FROM EMP
GROUP BY JOB
```

Results | Explain | Describe | Saved SQL | History

JOB	SUM_SAL	NTILE_SAL_ASC
ANALYST	6640	1
CLERK	8580	1
PRESIDENT	10020	2
SALESMAN	11280	3
MANAGER	16610	4

5 rows returned in 0.01 seconds Download

Определение NTILE (из документации ORACLE)

`NTILE` allows easy calculation of tertiles, quartiles, deciles and other common summary statistics. This function divides an ordered partition into a specified number of groups called **buckets** and assigns a bucket number to each row in the partition. `NTILE` is a very useful calculation because it lets users divide a data set into fourths, thirds, and other groupings.

The buckets are calculated so that each bucket has exactly the same number of rows assigned to it or at most 1 row more than the others. For instance, if you have 100 rows in a partition and ask for an `NTILE` function with four buckets, 25 rows will be assigned a value of 1, 25 rows will have value 2, and so on. These buckets are referred to as equiheight buckets.

If the number of rows in the partition does not divide evenly (without a remainder) into the number of buckets, then the number of rows assigned for each bucket will differ by one at most. The extra rows will be distributed one for each bucket starting from the lowest bucket number. For instance, if there are 103 rows in a partition which has an `NTILE(5)` function, the first 21 rows will be in the first bucket, the next 21 in the second bucket, the next 21 in the third bucket, the next 20 in the fourth bucket and the final 20 in the fifth bucket.

Упражнение 1

- Классифицируйте клиентов из demo базы ORACLE на 3 категории в зависимости от общей суммы заказов.

Пример 15 (ROW_NUMBER – возвращает смещение строки по отношению к началу упорядоченной группы)

```
SELECT JOB,
       SUM(SAL) SUM_SAL,
       ROW_NUMBER() OVER (ORDER BY SUM(SAL) ASC) ROW_NUMBER_SAL_ASC
FROM EMP
GROUP BY JOB
ORDER BY 1 DESC
```

Results | Explain | Describe | Saved SQL | History

JOB	SUM_SAL	ROW_NUMBER_SAL_ASC
SALESMAN	11280	4
PRESIDENT	10020	3
MANAGER	16610	5
CLERK	8580	2
ANALYST	6640	1

5 rows returned in 0.00 seconds Download

LAG/LEAD analysis

- Функции обеспечивают доступ к строкам в запросе с заданным смещением относительно текущей строки.

Синтаксис для использования:

```
{LAG | LEAD} ( value_expr [, offset] )
 OVER ( [partition_clause] order_by_clause )
```

Пример 16(LAD, LEAD – предыдущее и последующее значения)

```
SELECT
 DEPTNO, SUM(sal) amount,
 LAG(SUM(sal), 1) OVER (ORDER BY DEPTNO) AS prev_amount,
 LEAD(SUM(sal), 1) OVER (ORDER BY DEPTNO) AS next_amount
FROM emp
GROUP BY DEPTNO
ORDER BY DEPTNO;
```

Results Explain Describe Saved SQL History

DEPTNO	AMOUNT	PREV_AMOUNT	NEXT_AMOUNT
10	17560	-	16650
20	16650	17560	18920
30	18920	16650	-

3 rows returned in 0.03 seconds Download

Windows functions

Позволяют с легкостью вычислять:

нарастающие итоги, скользящее среднее, центральное среднее и т.п.

Работают совместно с агрегатными функциями: SUM(), AVG(), MAX(), MIN(), COUNT() и порядковыми функциями FIRST_VALUE() и LAST_VALUE()(возвращают первую и последнюю запись в окне).

Пример 17(вычисление нарастающих итогов)

```
SELECT
     JOB, SUM(sal) AS job_salary,
     SUM(SUM(sal)) OVER
      (ORDER BY JOB ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
      AS cumulative_sal
FROM emp
GROUP BY job
ORDER BY job;
```

Results | Explain | Describe | Saved SQL | History

JOB	JOB_SALARY	CUMULATIVE_SAL
ANALYST	6640	6640
CLERK	8580	15220
MANAGER	16610	31830
PRESIDENT	10020	41850
SALESMAN	11280	53130

5 rows returned in 0.03 seconds Download

Пример 18 (скользящее среднее)

```
SELECT empno, sal,
    AVG(sal) OVER(ORDER BY empno ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) moving_average
FROM emp
ORDER BY empno;
```

Results | Explain | Describe | Saved SQL | History

EMPNO	SAL	MOVING_AVERAGE
7369	810	810
7499	1610	1210
7521	1260	1226.6666666666666666666666666666666667
7566	2985	1666.25
7654	1260	1778.75
7698	2860	2091.25
7782	2460	2391.25

Пример 19 (центральное среднее)

```
SELECT empno, sal,
    AVG(sal) OVER(ORDER BY empno
        ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) centered_average
FROM emp
ORDER BY empno;
```

Results | Explain | Describe | Saved SQL | History

EMPNO	SAL	CENTERED_AVERAGE
7369	810	1210
7499	1610	1226.6666666666666666666666666666666667
7521	1260	1951.6666666666666666666666666666666667
7566	2985	1835
7654	1260	2368.3333333333333333333333333333333333
7698	2860	2193.3333333333333333333333333333333333
7782	2460	1876.6666666666666666666666666666666667
7788	310	2593.3333333333333333333333333333333333

Пример 20 (вычисление размера окна)

```
SELECT empno, sal,
    COUNT(sal) OVER(ORDER BY empno
        ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) how_many_rows
FROM emp
ORDER BY empno;
```

Results | Explain | Describe | Saved SQL | History

EMPNO	SAL	HOW_MANY_ROWS
7369	810	2
7499	1610	3
7521	1260	3
7566	2985	3
7654	1260	3
7698	2860	3
7782	2460	3
7788	310	3

Пример 21(first_value, last_value в окне)

```
SELECT empno, sal,
    first_value(sal) OVER(ORDER BY empno ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) first_value,
    last_value(sal)  OVER(ORDER BY empno  ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) last_value
FROM emp
ORDER BY empno;
```

Results | Explain | Describe | Saved SQL | History

EMPNO	SAL	FIRST_VALUE	LAST_VALUE
7369	810	810	1610
7499	1610	810	1260
7521	1260	1610	2985
7566	2985	1260	1260
7654	1260	2985	2860
7698	2860	1260	2460
7782	2460	2860	310
7788	310	2460	5010

Пример 22 (first_value, last_value в группе)

Rows 10 Clear Command Find Tables Save Run

```
select empno, ename , deptno, sal,
      first_value(sal) over (partition by deptno order by sal)  group_min_sal,
      last_value(sal)  over (partition by deptno order by sal)  group_max_sal
from emp
order by deptno, sal
```

Results Explain Describe Saved SQL History

EMPNO	ENAME	DEPTNO	SAL	GROUP_MIN_SAL	GROUP_MAX_SAL
7934	MILER	10	1273	1273	1273
7782	CLARK	10	2423	1273	2423
7839	KING	10	4973	1273	4973
8020	ANDREW	10	5003	1273	5003
7369	SMITH	20	773	773	773
7876	ADAM	20	1073	773	1073

Домашнее задание 3

- Создать приложение, отображающее в виде графиков нарастающие итоги (сумма или количество) по заказам из демонстрационной базы ORACLE (нарастающие по времени). Запрос должен быть написан с использованием аналитических функций.
- Ссылку на приложение, логин и пароль для входа отправлять по адресу: N.Grafeeva@spbu.ru
- Тема - Data_Mining_2016_job3