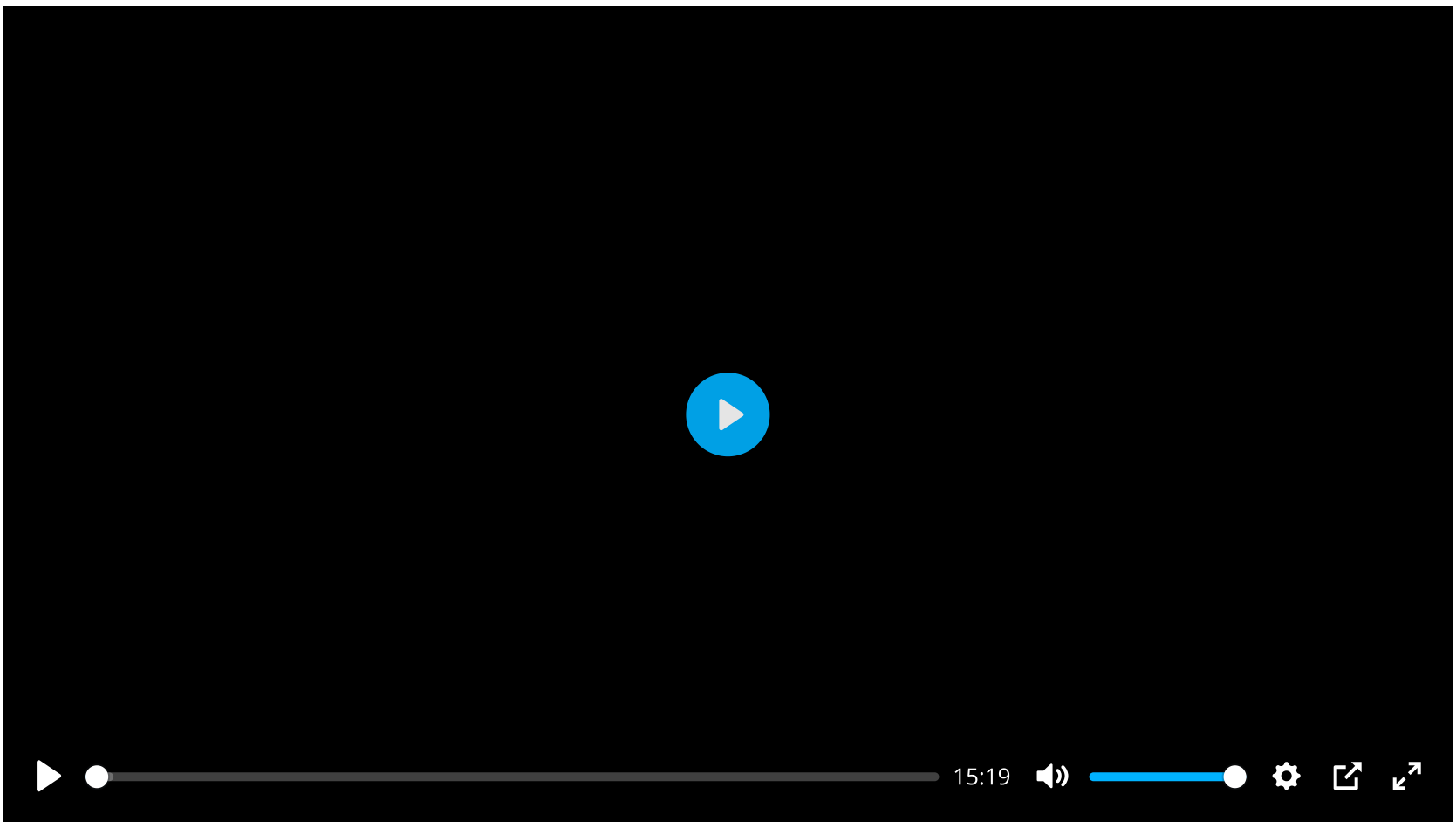
5.4 База данных глазами Data Engineer'a



Текстовая расшифровка видео:

ПРОДВИНУТЫЕ АГРЕГАЦИИ

План:

- Хитрая группировка;
- Оконные функции: ROW_NUMBER;
- Оконные функции: RANK и DENSE_RANK;
- Оконные функции: LAG и LEAD.

Хитрая группировка

Мы уже знаем, что в SQL есть процедура «Group by», фильтрация «Where» (работает до того, как сгруппировали), «Having» (работает после того, как сгруппировали).

Рассмотрим пример:

У нас есть стандартная БД с сотрудниками компании. Посмотрим максимальную зарплату по каждому отделу:

```
SELECT dept, MAX(salary) AS max_salary
FROM employees
GROUP BY dept
```

Получится следующее:

Query Query History

```
1  SELECT dept, MAX(salary) AS max_salary
2  FROM employees
3  GROUP BY dept
```

Data Output Messages Notifications

	dept text	max_salary bigint
1	Marketing	293000
2	Human Resources	291000
3	Finance	298000
4	Operations	290000
5	IT	292000

У подобного способа расчета группировок и агрегации есть одна негативная сторона – он уничтожает структуру изначальной таблицы в выводе.

Иногда нам хочется получить ту же самую таблицу, при этом добавить в нее какие-то агрегированные значения. Это решается в два шага:

- Сделать группировку;
- Сделать join с результатами группировки.

Также есть более современный и удобный механизм.

Запустим данный запрос:

```
SELECT e.*,
MAX(salary) OVER () AS max_salary
FROM employees e
```

Он выводит таблицу, которая была раньше, но при этом в конце, в колонке, добавляет максимальную зарплату для каждого сотрудника:

```
1  SELECT e.*,
2  MAX(salary) OVER () AS max_salary
3  FROM employees e
```

Data Output Messages Notifications

	t	responsibilities text	max_salary bigint
28	rketing	Fast arm style control president water medical.	298000
29	man Resources	Approach option who edge deal design class someone attack.	298000
30	ance	Item else heart dog local offer already.	298000
31	ance	Peace newspaper agreement whom ball eight those anyone.	298000
32	rketing	The produce what economic manager big control discussion.	298000
33	man Resources	Price get movement economy win reduce any order.	298000

Заметим, что мы не просто возвращаем все поля и максимальные зарплаты, но еще и делаем OVER что-то.

Рассмотрим третий запрос:

```
SELECT e.*,
MAX(salary) OVER (partition BY dept) AS max_salary
FROM employees e
```

Когда мы выполняем group by, у нас есть ключ, по которому мы группируем, и то, что схлопываем. Однако в контексте SQL есть еще концепт окна. Для того, чтобы применить агрегирующую функцию, нужно получить группу исходных данных (вся таблица, результат группировки и т.д.).

Окна создаются с помощью данной структуры:

```sql
SELECT e.*,
MAX(salary) OVER (partition BY dept) AS max_salary
FROM employees e
```

Мы специально внутри разбиваем данные по отделам и работаем в рамках этого запроса, в частности, с каждым отделом.

Если выполнить этот запрос, увидим, что осталась та же структура таблицы:

```sql
SELECT e.*,
MAX(salary) OVER (partition BY dept) AS max_salary
FROM employees e
```

Data Output | Messages | Notifications

	salary bigint	title text	dept text	responsibilities text
17	229000	Podiatrist	Finance	Economic hand performance activity party everybody you run d
18	168000	IT sales professional	Finance	Finally idea enter design serious how box wait.
19	226000	Cytogeneticist	Finance	Chance list head why standard issue likely military program pro
20	293000	Astronomer	Finance	Among policy level cold magazine find send development right
21	94000	Soil scientist	Finance	Bar modern space structure human current six senior manager
22	106000	Development worker, international aid	Finance	Economic western however become research life really use acc
23	157000	Network engineer	Finance	Once product magazine TV major fine article also trip rate sign
24	100000	Designer, interior/spatial	Finance	System American there tax explain note side man place pass m

При этом все данные сгруппированы вместе по отделам. Также для каждого отдела, в конце, для сотрудников видим максимальную зарплату.

Оконные функции: ROW_NUMBER

Существует целый ряд функций, который создан для работы внутри окон.

Одна из самых базовых вещей – присвоить порядковый номер каждому элементу внутри окна.

Рассмотрим пример:

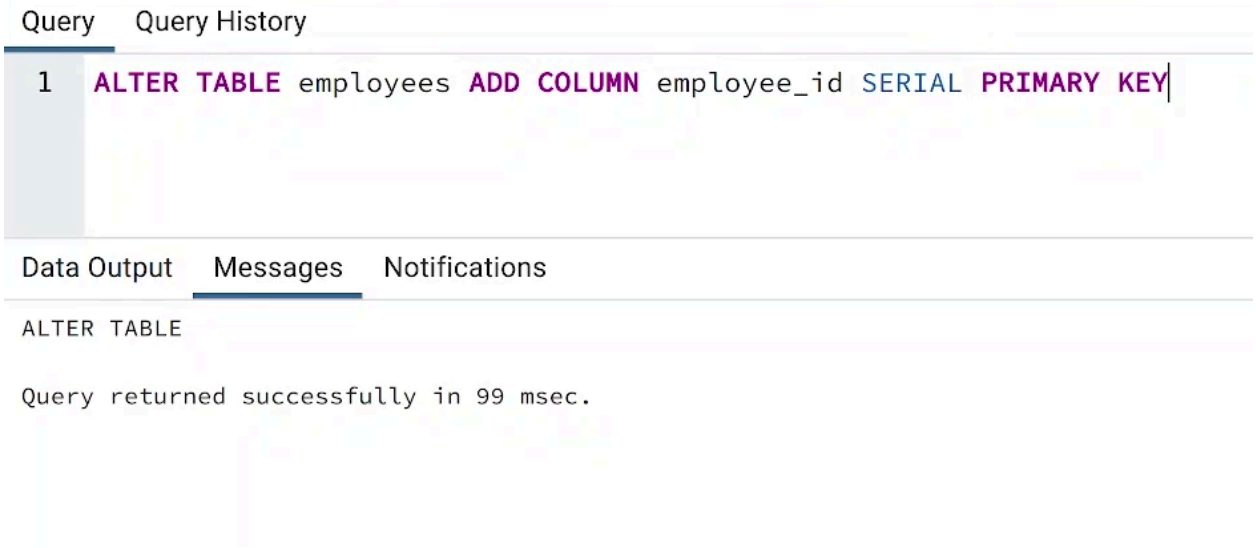
```sql
SELECT * FROM (
    SELECT e.*,
    ROW_NUMBER() OVER (partition BY dept ORDER BY employee_id) AS rn
    FROM employees e
) x
WHERE x.rn < 3;
```

В исходных данных нет указания времени, когда человек пришел на работу (нет date time). Если предположить, что сотрудники приходили в отделы в порядке ID, то как нам найти первых двух сотрудников из всех отделов?

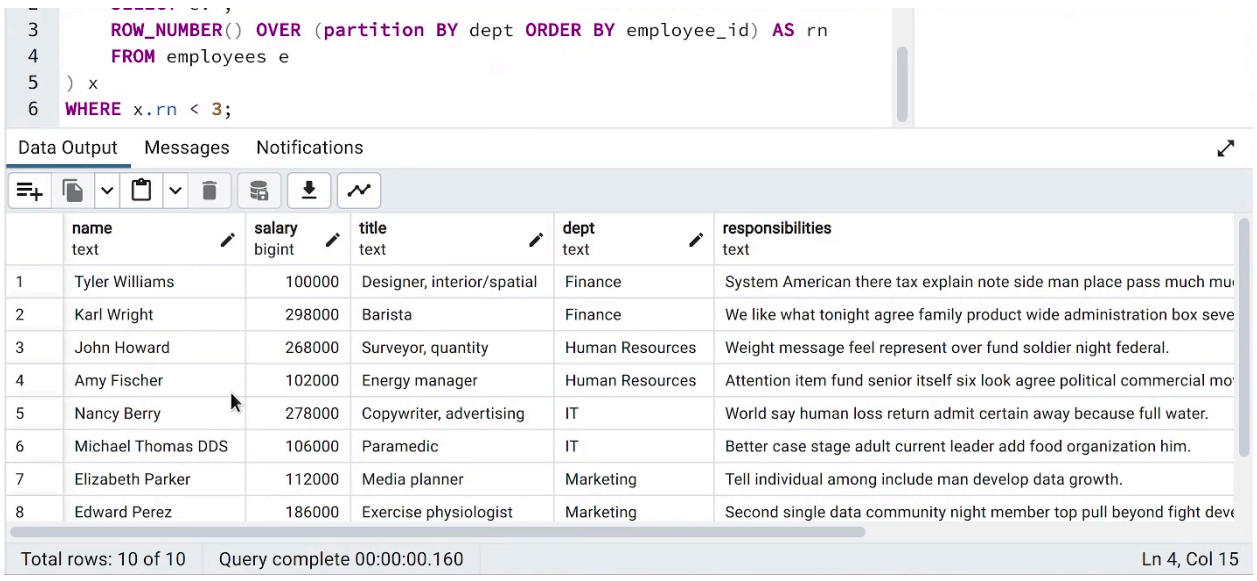
У нас получится составной запрос. Мы разбиваем на окна и в рамках каждого окна упорядочиваем по ID. После этого для каждой записи, которая получилась, вызываем функцию «ROW_NUMBER» (присваиваем порядковый номер в рамках окна). Во внешнем запросе извлекаем все записи, у которых номер меньше трех.

Выполним этот запрос.

Добавим поле «employees ADD»:



Видим следующее:



У нас получилась группировка по отделам. В рамках отдела все упорядочены по ID, но мы для каждого отдела взяли только первых двух сотрудников.

Можно сделать и в форме Group by, но этот вариант гораздо проще.

Оконные функции: RANK и DENSE_RANK

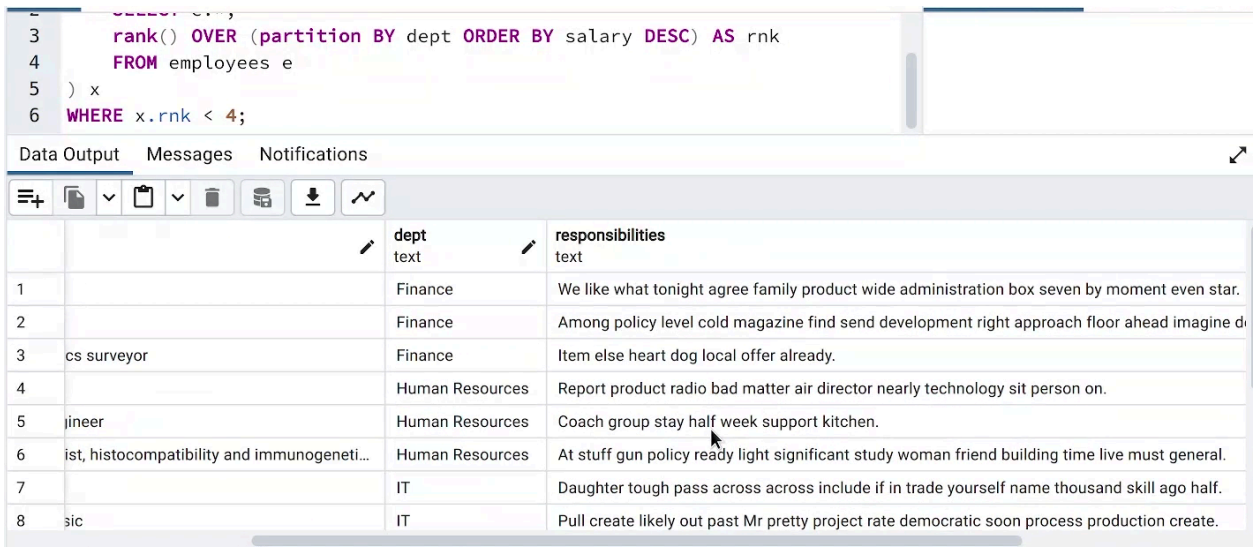
Также мы можем отранжировать людей по какому-то признаку. Например, у нас есть поле «Зарплаты». Мы можем построить ранг, иерархию людей согласно тому, сколько денег они получают в рамках своего отдела. Для этого существуют две функции:

- Rank;
- Dense rank.

Выполним Rank:

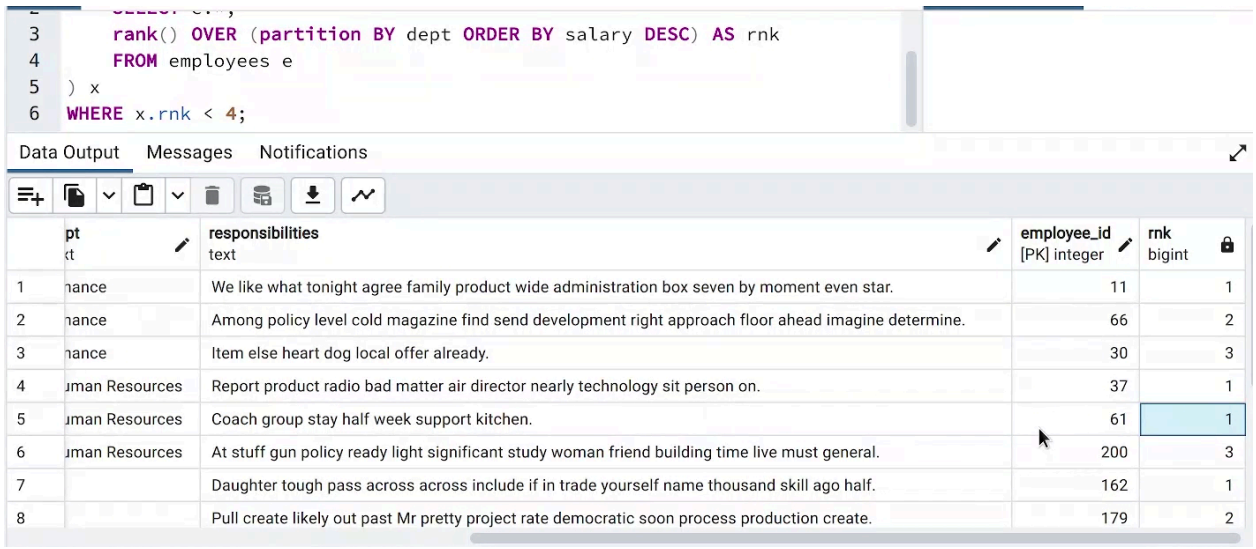
```
SELECT * FROM (
    SELECT e.*,
    rank() OVER (partition BY dept ORDER BY salary DESC) AS rnk
    FROM employees e
) x
WHERE x.rnk < 4;
```

Получаем следующее:



Если посмотреть на Ranking, можно увидеть, что два человека попали в первый ранг (они получают одинаковую зарплату), а следующий человек попал в третий ранг. **Вопрос**: где второй ранг?

Особенность работы функции Rank – проскакивание значений:

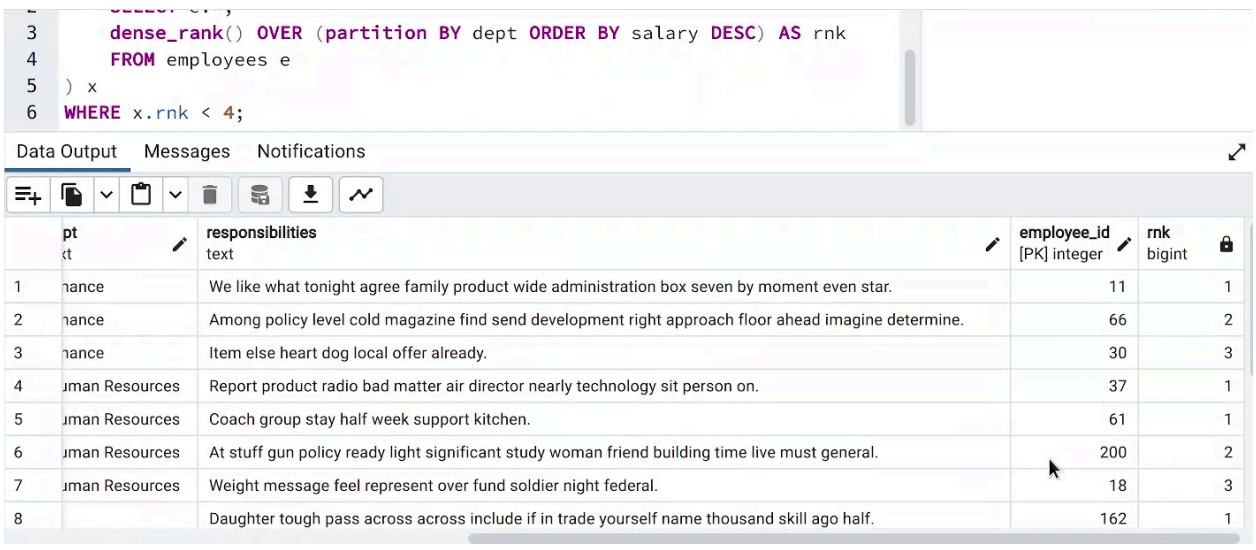


	pt text	responsibilities text	employee_id [PK] integer	rnk bigint
1	nance	We like what tonight agree family product wide administration box seven by moment even star.	11	1
2	nance	Among policy level cold magazine find send development right approach floor ahead imagine determine.	66	2
3	nance	Item else heart dog local offer already.	30	3
4	uman Resources	Report product radio bad matter air director nearly technology sit person on.	37	1
5	uman Resources	Coach group stay half week support kitchen.	61	1
6	uman Resources	At stuff gun policy ready light significant study woman friend building time live must general.	200	3
7		Daughter tough pass across across include if in trade yourself name thousand skill ago half.	162	1
8		Pull create likely out past Mr pretty project rate democratic soon process production create.	179	2

Выполним Dense rank:

```
SELECT * FROM (
    SELECT e.*,
    dense_rank() OVER (partition BY dept ORDER BY salary DESC) AS drnk
    FROM employees e
) x
WHERE x.drnk < 4;
```

Получим следующее:



	pt text	responsibilities text	employee_id [PK] integer	rnk bigint
1	nance	We like what tonight agree family product wide administration box seven by moment even star.	11	1
2	nance	Among policy level cold magazine find send development right approach floor ahead imagine determine.	66	2
3	nance	Item else heart dog local offer already.	30	3
4	uman Resources	Report product radio bad matter air director nearly technology sit person on.	37	1
5	uman Resources	Coach group stay half week support kitchen.	61	1
6	uman Resources	At stuff gun policy ready light significant study woman friend building time live must general.	200	2
7	uman Resources	Weight message feel represent over fund soldier night federal.	18	3
8		Daughter tough pass across across include if in trade yourself name thousand skill ago half.	162	1

Данная функция работает более предсказуемо и не проскакивает значения. Заметим, что при таком подходе, поскольку мы берем трех людей по первым трем Rank'ам, в Human Resources теперь четыре человека (два человека первого ранга, один человек второго и третьего ранга):

	pt text	responsibilities text	employee_id [PK] integer	rnk bigint
1	nance	We like what tonight agree family product wide administration box seven by moment even star.	11	1
2	nance	Among policy level cold magazine find send development right approach floor ahead imagine determine.	66	2
3	nance	Item else heart dog local offer already.	30	3
4	uman Resources	Report product radio bad matter air director nearly technology sit person on.	37	1
5	uman Resources	Coach group stay half week support kitchen.	61	1
6	uman Resources	At stuff gun policy ready light significant study woman friend building time live must general.	200	2
7	uman Resources	Weight message feel represent over fund soldier night federal.	18	3
8		Daughter tough pass across across include if in trade yourself name thousand skill ago half.	162	1

Оконные функции: LAG и LEAD

Представим, что мы каким-то образом упорядочиваем людей в рамках окна и хотим не просто посмотреть на окно и получить агрегат, а хотим рассчитать что-либо в группе касательно предыдущей записи по порядку.

Для этого существуют следующие функции:

- LAG;
- LID.

LAG – функция, которая смотрит в прошлое. То есть, мы можем выбрать предыдущую строчку и т.д.

LID – функция, которая смотрит вперед. Мы можем просчитать значение текущей строчки на основе строчки впереди.

Выполним простейшую функцию:

```
SELECT e.*,
lag(salary) OVER (partition BY dept ORDER BY employee_id) AS prev_emp_sal
FROM employees e
```

Data Output | Messages | Notifications

	esponsibilities ext	employee_id [PK] integer	prev_emp_sal bigint
1	System American there tax explain note side man place pass much much.	9	[null]
2	We like what tonight agree family product wide administration box seven by moment even star.	11	100000
3	Economic hand performance activity party everybody you run describe pass face company event law many certainly.	23	298000
4	tem else heart dog local offer already.	30	229000
5	Peace newspaper agreement whom ball eight those anyone.	31	289000
6	Chance list head why standard issue likely military program product crime understand anything fish fish market.	38	95000
7	Reason guess live surface entire again program onto human poor until.	42	226000
8	Try eight resource section tax continue usually suffer anyone ability.	43	206000

Total rows: 200 of 200 | Query complete 00:00:00.306 | Ln 3, Col 17

Для каждого сотрудника выведем в колонке зарплату предыдущего сотрудника, по порядку. Поскольку у первого сотрудника для каждого окна нет предыдущего, мы получаем зарплату «Null».

В функции «LAG» мы можем задать другие параметры, например, посмотреть не на одну строчку назад, а на две:

```
SELECT e.*,
lag(salary, 2) OVER (partition BY dept ORDER BY employee_id) AS prev_emp_sal
FROM employees e
```

Data Output | Messages | Notifications

	esponsibilities ext	employee_id [PK] integer	prev_emp_sal bigint
1	System American there tax explain note side man place pass much much.	9	[null]
2	We like what tonight agree family product wide administration box seven by moment even star.	11	[null]
3	Economic hand performance activity party everybody you run describe pass face company event law many certainly.	23	100000
4	tem else heart dog local offer already.	30	298000
5	Peace newspaper agreement whom ball eight those anyone.	31	229000
6	Chance list head why standard issue likely military program product crime understand anything fish fish market.	38	289000
7	Reason guess live surface entire again program onto human poor until.	42	95000
8	Try eight resource section tax continue usually suffer anyone ability.	43	226000

Total rows: 200 of 200 | Query complete 00:00:00.163 | Ln 2, Col 14

Также мы можем указать, чем хотим забить строчки с null'ами. В «LAG» третьим аргументом можем написать какое-нибудь число:

```
SELECT e.*,
lag(salary, 2, 1337) OVER (partition BY dept ORDER BY employee_id) AS prev_e
FROM employees e
```

Data Output | Messages | Notifications

	esponsibilities ext	employee_id [PK] integer	prev_emp_sal bigint
1	System American there tax explain note side man place pass much much.	9	1337
2	We like what tonight agree family product wide administration box seven by moment even star.	11	1337
3	Economic hand performance activity party everybody you run describe pass face company event law many certainly.	23	100000
4	tem else heart dog local offer already.	30	298000
5	Peace newspaper agreement whom ball eight those anyone.	31	229000
6	Chance list head why standard issue likely military program product crime understand anything fish fish market.	38	289000
7	Reason guess live surface entire again program onto human poor until.	42	95000
8	Try eight resource section tax continue usually suffer anyone ability.	43	226000

Total rows: 200 of 200 | Query complete 00:00:00.144 | Ln 3, Col 17

Рассмотрим второй пример. Данный пример более сложный:

```
SELECT e.*,
lag(salary) OVER (partition BY dept ORDER BY employee_id) AS prev_emp_sal,
CASE WHEN e.salary > lag(salary) OVER (partition BY dept ORDER BY employee_id) THEN 'Higher t
     WHEN e.salary < lag(salary) OVER (partition BY dept ORDER BY employee_id) THEN 'Lower th
     WHEN e.salary = lag(salary) OVER (partition BY dept ORDER BY employee_id) THEN 'Equal to
     END sal_range
FROM employees e
```

Для каждого сотрудника в рамках окна смотрим на предыдущее значение и выводим:

- Если у него зарплата больше, то пишем «Зарплата больше, чем у предыдущего».
- Если зарплата равна, то пишем «Equal».
- Если зарплата меньше, то пишем «Lower».

Заметим частый повтор формирования окна. Не стоит переживать, оптимизатор схлопнет их в один проход.

Запустим:

```
    WHEN e.salary < lag(salary) OVER (partition BY dept ORDER BY employee_
    WHEN e.salary = lag(salary) OVER (partition BY dept ORDER BY employee_
    END sal_range
FROM employees e
```

Data Output | Messages | Notifications

		employee_id [PK] integer	prev_emp_sal bigint	sal_range text
1	ıx explain note side man place pass much much.	9	[null]	[null]
2	e family product wide administration box seven by moment even star.	11	100000	Higher than previous
3	nce activity party everybody you run describe pass face company event law many certainly.	23	298000	Lower than previous
4	offer already.	30	229000	Higher than previous
5	ent whom ball eight those anyone.	31	289000	Lower than previous
6	ndard issue likely military program product crime understand anything fish fish market.	38	95000	Higher than previous
7	e entire again program onto human poor until.	42	226000	Lower than previous
8	tax continue usually suffer anyone ability.	43	206000	Lower than previous

Как вам урок?

Изучил, далее >