Конспект по теме "Расширенные возможности для аналитика в SQL"

Группируем данные

Если данные нужно разделить на группы по значениям полей, применяют команду **GROUP BY**:

```
 SELECT
  поле_1,
  поле_2,
  ...,
  поле_n,
  АГРЕГИРУЮЩАЯ_ФУНКЦИЯ(поле) AS here_you_are
FROM
  таблица
WHERE -- если необходимо
  условие
GROUP BY
  поле_1,
  поле_2,
  ...,
  поле_n
```

После команды **GROUP BY** перечисляют все поля из блока SELECT. Саму агрегирующую функцию включать в блок GROUP BY не нужно — с ней запрос не выполнится. GROUP BY в SQL работает аналогично методу *groupby()* в Pandas.

Конструкция GROUP BY работает для всех агрегирующих функций: COUNT(), AVG(), SUM(), MAX(), MIN(). Можно вызывать несколько функций сразу.

Сортируем данные

Итоги анализа обычно представляют в определённом порядке. Чтобы сортировать данные по указанным полям, применяют команду **ORDER BY**:

```
SELECT
  поле_1,
  поле_2,
  ...,
  поле_n,
  АГРЕГИРУЮЩАЯ_ФУНКЦИЯ(поле) AS here_you_are
FROM
  таблица
WHERE -- если нужно
  условие
GROUP BY
  поле_1,
  поле_2,
  ...,
  поле_n,
ORDER BY -- если необходимо, перечисляем только те поля,
--по которым хотим отсортировать таблицу
  поле_1,
  поле_2,
  ...,
  поле_n,
  here_you_are
```

В отличие от GROUP BY, в блоке с командой ORDER BY перечисляем только те поля, по которым хотим сортировать.

У команды ORDER BY два аргумента. Они отвечают за порядок сортировки в столбцах:

- **ASC** сортирует данные в порядке возрастания. Это аргумент ORDER BY по умолчанию.
- **DESC** сортирует данные по убыванию.

Аргументы команды ORDER BY указывают сразу после поля, по которому сортировали данные:

```
ORDER BY
  название_поля DESC
-- сортируем данные по убыванию

ORDER BY
  название_поля ASC
-- сортируем данные по возрастанию
```

Команда **LIMIT** ограничивает количество строк в выводе. Её всегда указывают последней в запросе. После LIMIT указывают требуемое число строк — n.

```
SELECT
  поле_1,
  поле_2,
  ...,
  поле_n,
  АГРЕГИРУЮЩАЯ_ФУНКЦИЯ(поле) AS here_you_are
FROM
  таблица
WHERE -- если необходимо
  условие
GROUP BY
  поле_1,
  поле_2,
  ...,
  поле_n,
ORDER BY -- если необходимо, перечисляем только те поля,
--по которых хотим отсортировать таблицу
  поле_1,
  поле_2,
  ...,
  поле_n,
  here_you_are
LIMIT -- если необходимо
  n
-- n-максимальное количество строк, которое вернёт такой запрос
```

Обработка данных в группировке

Для того, чтобы применить фильтр по строкам, используется конструкция **WHERE.** В этой конструкции в качестве параметров используются строки таблицы. Если же в фильтре нужно использовать результаты применения агрегированных функций, используется аналог **WHERE** — конструкция **HAVING**:

```
SELECT
  поле_1,
  поле_2,
  ...,
  поле_n,
  АГРЕГИРУЮЩАЯ_ФУНКЦИЯ(поле) AS here_you_are
FROM
  TABLE
WHERE -- если необходимо
  условие
GROUP BY
  поле_1,
  поле_2,
  ...,
```

```
  поле_n
HAVING
  АГРЕГИРУЮЩАЯ_ФУНКЦИЯ(поле_для_группировки) > n
ORDER BY -- если необходимо, перечисляем только те поля,
--по которых хотим отсортировать таблицу
  поле_1,
  поле_2,
  ...,
  поле_n,
  here_you_are
LIMIT -- если необходимо
  n
```

В результирующую выборку попадут только те строки, для которых результат агрегирующей функции соответствует условию блоков HAVING и WHERE.

Если команды HAVING и WHERE так похожи, почему нельзя записать все условия в одной из них? Дело в том, что WHERE отрабатывает перед группировкой данных и расчётом агрегирующей функции. Потому задать фильтр на результат агрегирующей функции в WHERE нельзя. И здесь выручает HAVING.

Обратите внимание на порядок команд:

1) GROUP BY;

2) HAVING;

3) ORDER BY.

Указывайте команды строго в этой последовательности, иначе запрос не выполнится.

Операторы и функции для работы с датами

Две основные функции для работы со временем и датой — **EXTRACT** и **DATE_TRUNC**. Обе функции вызывают в блоке SELECT.

Шаблон функции EXTRACT:

```
SELECT
  EXTRACT(часть_даты FROM столбец) AS новый_столбец_с_датой
FROM
  Таблица_со_всеми_датами
```

Название функции определяет её суть. EXTRACT *извлекает* из даты нужную часть: год, месяц, минуту. Что ещё можно получить вызовом EXTRACT:

- `century` — век;
- `day` — день;
- `doy` — день года: от 1 до 365/366;
- `isodow` — день недели: понедельник — 1, воскресенье — 7.
- `hour` — час;
- `milliseconds` — миллисекунда;
- `minute` — минута;
- `second` — секунда;
- `month` — месяц;
- `quarter` — квартал;
- `week` — неделя в году;
- `year` — год.

DATE_TRUNC *усекает* дату до часа, дня или месяца. В отличие от EXTRACT часть, до которой нужно усечь дату, записывают как строку. А столбец, откуда берут данные о времени, указывают через запятую:

```
SELECT
  DATE_TRUNC('часть_даты_до_которой_усекаем', столбец) AS новый_столбец_с_датой
FROM
  Таблица_со_всеми_датами
```

Часть даты, до которой данные нужно «обнулить», указывают в аргументе функции DATE_TRUNC:

`'microseconds'` — микросекунды;
`'milliseconds'` — миллисекунды;
`'second'` — секунда;
`'minute'` — минута;
`'hour'` — час;

`'day'` — день;
`'week'` — неделя;
`'month'` — месяц;
`'quarter'` — квартал;
`'year'` — год;
`'decade'` — декада года;
`'century'` — век.

Подзапросы

Подзапрос — это запрос в запросе, также называемый **внутренним запросом**. Он используется для получения нужной информации для применения во **внешнем запросе**.

Подзапросы могут выполняться в разных частях запроса. Если подзапрос записать в блоке FROM, то SELECT выберет данные из таблицы, полученной в результате работы подзапроса. Имя этой таблицы указывают во внутреннем запросе, к её столбцам обращаются во внешнем. Подзапрос записывают в круглых скобках:

```
SELECT
ПОДЗАПРОС_1.название_столбца,
ПОДЗАПРОС_1.название_столбца_2
FROM -- Для лучшей читабельности кода, переносите подзапрос на новую строку
  -- отделяйте подзапросы отступами
  (SELECT
    название_столбца,
    название_столбца_2
  FROM
    название_таблицы
  WHERE
    название_столбца = значение) AS ПОДЗАПРОС_1;
-- не забывайте давать имя подзапросу в блоке FROM
```

Внутренние запросы могут понадобиться в разных блоках внешнего запроса. Например, устроим подзапрос в блоке WHERE. Тогда выберутся данные из столбца со значениями, сгенерированными в результате работы подзапроса:

```
SELECT
  название_столбца,
  название_столбца_1
FROM
```

```
  название_таблицы
WHERE
  название_столбца =
    (SELECT
      столбец_1
    FROM
      название_таблицы_2
    WHERE
      столбец_1  = значение)
```

Дополним шаблон конструкцией IN, чтобы собирать данные из нескольких столбцов:

```
SELECT
  название_столбца,
  название_столбца_1
FROM
  название_таблицы
WHERE
  название_столбца IN
      (SELECT
        столбец_1
      FROM
        название_таблицы_2
      WHERE
        столбец_1 = значение_1 OR столбец_1 = значение_2)
```