1. 你的首个子查询

关于这节课的内容

到目前为止，你已经学习了如何使用 SQL 处理数据的大量知识。这节课将重点讲解以下三项内容：

1. 子查询
2. 表格表达式
3. 持久衍生表格

子查询和表格表达式都是用来通过查询创建一个表格，然后再编写一个查询来与这个新创建的表格进行互动。有时候，你要回答的问题无法通过直接处理数据库中的现有表格获得答案。但是，如果我们能通过现有的表格创建新的表格，我们就能查询这些新的表格，并回答我们的问题。这节课的查询就可以实现这一目的。如果你暂时想不出需要此类 SQL 的问题，别担心，你之后会见到大量类似的问题！每当我们需要使用现有表格创建新的表格，然后需要再次查询时，就表明我们需要使用某种子查询。在以下几个页面中，我们将一起看一个示例。然后你将自己解答一些问题，练习巩固相关技能。

当你第一次编写子查询时，可能会显得很复杂。我们尝试将子查询分解成几个不同的部分。如果你遇到问题，请重新观看上述视频。我们想要算出每个渠道每天的平均事件次数。第一个表格将提供每个渠道每天的事件次数，然后我们将使用第二个查询对这些值求平均。请你尝试自己解答这个问题。

练习1 在哪个日期-渠道对中发生的事件最多？（选择所有正确项）

- ☐ 2017年1月1日；直接渠道
- ☐ 2016年12月31日；facebook
- ☐ 2016年11月3日；直接渠道
- ☐ 2016年12月21日；直接渠道

练习2 标出以下所有关于编写子查询的正确描述。

- ☐ 原始查询位于FROM语句中。
- ☐ 原始查询位于SELECT语句中。
- ☐ 在SELECT语句中使用*，以便从原始查询中获取所有数据。
- ☐ 必须对嵌套在外部查询中的表格使用别名。

练习3 将每个渠道与相应的每日平均事件数相匹配。 A.4.90 B.1.60 C.1.67 D.1.32

描述	平均事件/日期书
direct	
facebook	
organic	
twitter	

2. 练习：关于子查询的更多内容

子查询格式

在编写子查询时，查询很容易就看起来很复杂。为了便于阅读，其实日后经常只是你自己要阅读：要记住的重要事项是，在使用子查询时，要让读者能够轻松地判断查询的哪个部分将一起执行。大部分人的做法是按照某种方式缩进子查询，上一页面的解决方案就是这么做的。

这节课的示例缩进很明显，一直到小括号。如果你嵌套了很多的子查询，则不适用，一般法则就是思考下如何以便于阅读的方式编写查询。下面给出了以多种方式编写同一查询的示例。你会发现，某些示例明显比其他的容易阅读。

格式糟糕的查询

虽然这些格式糟糕的查询和格式清晰的查询一样会执行，但是却不容易让人理解查询的作用！

以下是第一个示例，根本无法判断查询的作用：

```
SELECT * FROM (SELECT DATE_TRUNC('day',occurred_at) AS day, channel, COUNT(*) as
events FROM web_events GROUP BY 1,2 ORDER BY 3 DESC) sub;
```

下面的第二个示例不是太糟糕，但是你会发现最后一个示例依然更容易读懂。

```
SELECT *
FROM (
SELECT DATE_TRUNC('day',occurred_at) AS day,
channel, COUNT(*) as events
FROM web_events
GROUP BY 1,2
ORDER BY 3 DESC) sub;
```

格式清晰的查询

与之前的示例相比，在这个格式清晰的示例中，我们很容易就看出要从哪个表格中获取数据。此外，如果在子查询后面有 GROUP BY、ORDER BY、WHERE、HAVING 或任何其他语句，则按照外部查询的同一级别缩进，正如最后一个示例所显示的，它是上个练习的最后一个解决方案。

```
SELECT *
FROM (SELECT DATE_TRUNC('day',occurred_at) AS day,
             channel, COUNT(*) as events
      FROM web_events
      GROUP BY 1,2
      ORDER BY 3 DESC) sub;
```

下面的查询很相似，但是向外部查询应用了其他逻辑，因此按照外部查询的级别缩进。而内部查询逻辑的缩进级别与内部表格匹配。

```
SELECT *
FROM (SELECT DATE_TRUNC('day',occurred_at) AS day,
```

```
                channel, COUNT(*) as events
      FROM web_events
      GROUP BY 1,2
      ORDER BY 3 DESC) sub
GROUP BY channel
ORDER BY 2 DESC;
```

最后两个查询容易读懂多了！

子查询（第二部分）

在你写的第一个子查询中，你编写了一个子查询来创建表格，然后可以在 FROM 语句中查询该表格。但是，如果只返回一个值，则可以在逻辑语句中使用该值，例如 WHERE、HAVING，甚至 SELECT，该值可以嵌套在 CASE 语句中。

在下一页面中，我们将讲解这个示例，然后你将尝试自己回答一些问题。

专家提示

注意，在条件语句中编写子查询时，不能包含别名。这是因为该子查询会被当做单个值（或者对于 IN 情况是一组值），而不是一个表格。

同时注意，这里的查询对应的是单个值。如果我们返回了整个列，则需要使用 IN 来执行逻辑参数。如果我们要返回整个表格，则必须为该表格使用别名，并对整个表格执行其他逻辑。

使用下面的任务列表来逐步了解上个示例的步骤。

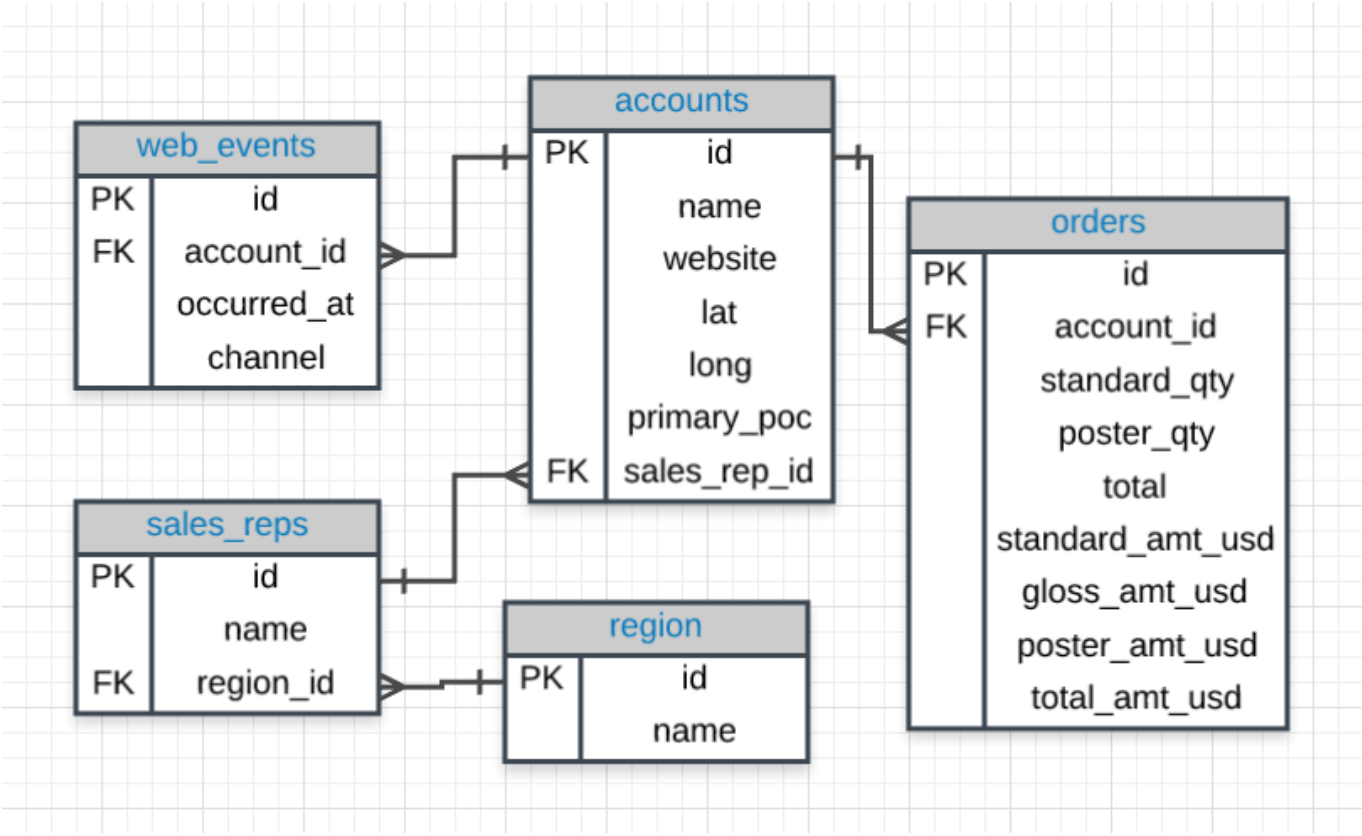
习题 1/2 第一个订单是在哪年/月下的？

- ☐ 2011年10月
- ☐ 2012年10月
- ☐ 2013年12月
- ☐ 2013年11月

习题2/2 将每个值与相应的说明相匹配 A. 112 B.209 C.377331 D.268

说明	值
在orders表格中第一个月下订单时所出售的标准纸张的平均数量。	
在orders表格中第一个月下订单时所出售的标铜版张的平均数量。	
在orders表格中第一个月下订单时所出售的标广告张的平均数量。	
在orders表格中第一个月下订单时在所有订单上的总消费。	

3. 练习：爱上子查询



更多的子查询练习

以上是数据库的 ERD，当你处理下面的练习时，可能会用到该 ERD。你应该将答案写成子查询，而不是得出一个答案并复制输出。这么做的重要性是能够让查询动态地回答问题，即使数据变化了，依然能获得正确的答案。

1. 提供每个区域拥有最高销售额 (total_amt_usd) 的销售代表的姓名。
2. 对于具有最高销售额 (total_amt_usd) 的区域，总共下了多少个订单 （total count orders） ？
3. 对于购买标准纸张数量 (standard_qty) 最多的客户（在作为客户的整个时期内），有多少客户的购买总数依然更多？
4. 对于（在作为客户的整个时期内）总消费 (total_amt_usd) 最多的客户，他们在每个渠道上有多少 web_events？
5. 对于总消费前十名的客户，他们的平均终身消费 (total_amt_usd) 是多少?
6. 比所有客户的平均消费高的企业平均终身消费 (total_amt_usd) 是多少？

4. 练习：WITH与子查询

WITH 语句经常称为公用表表达式（简称 CTE）。虽然这些表达式和子查询的目的完全一样，但是实际更常用，因为对未来的读者来说，更容易看懂其中的逻辑。

在下一部分，我们将更仔细地讲解这个示例，确保你能了解子查询和这些表达式之间的相似性，并能运用到实践中！如果你已经了解这些内容，可以跳到练习部分。

你的第一个 **WITH (CTE)**

下面是"你的第一个子查询"部分的问题和解决方案。

问题：你需要算出每个渠道每天的平均事件数。

解决方案：

```
SELECT channel, AVG(events) AS average_events
FROM (SELECT DATE_TRUNC('day',occurred_at) AS day,
channel, COUNT(*) as events
FROM web_events
GROUP BY 1,2) sub
GROUP BY channel
ORDER BY 2 DESC;
```

我们使用 WITH 语句重新编写查询。

注意：你可以获取内部查询：

```
SELECT DATE_TRUNC('day',occurred_at) AS day,
channel, COUNT(*) as events
FROM web_events
GROUP BY 1,2
```

我们在此部分放入 WITH 语句。注意，在下面我们将表格的别名设为 events：

```
WITH events AS (
SELECT DATE_TRUNC('day',occurred_at) AS day,
channel, COUNT(*) as events
FROM web_events
GROUP BY 1,2)
```

现在，我们可以像对待数据库中的任何其他表格一样使用这个新创建的 events 表格：

```
WITH events AS (
SELECT DATE_TRUNC('day',occurred_at) AS day,
channel, COUNT(*) as events
FROM web_events
GROUP BY 1,2)

SELECT channel, AVG(events) AS average_events
FROM events
GROUP BY channel
ORDER BY 2 DESC;
```

对于上述示例，我们只需一个额外的表格，但是想象下我们要创建第二个表格来从中获取数据。我们可以按照以下方式来创建额外的表格并从中获取数据：

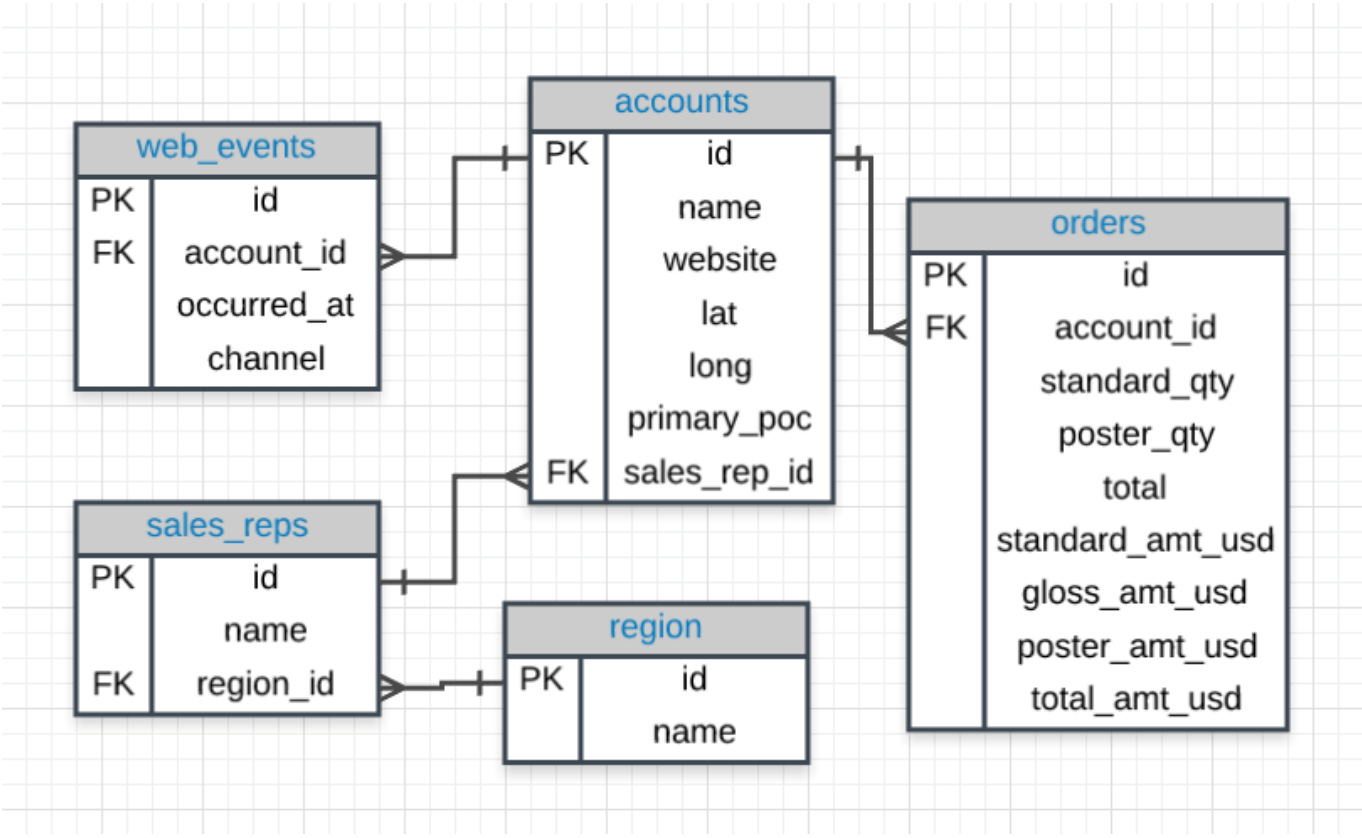
```
WITH table1 AS (
SELECT *
FROM web_events),

table2 AS (
SELECT *
FROM accounts)

SELECT *
FROM table1
JOIN table2
ON table1.account_id = table2.id;
```

然后，你可以按照相同的方式使用 WITH 语句添加越来越多的表格。底部的练习将确保你掌握了这些新查询的所有必要组成部分。

5. 练习：WITH



WITH 练习

本质上，WITH 语句和子查询执行的任务相同。因此，你可以使用 WITH 编写之前的任何一个查询。尝试使用 WITH（而不是子查询）再次执行之前的每个查询。

以上是数据库的 ERD，当你处理下面的练习时，可能会用到该 ERD。你应该将答案写成 WITH 语句，而不是得出一个答案并复制输出。这么做的重要性是查询能够动态地回答问题，即使数据变化了，依然能获得正确的答案。

1. 提供每个区域拥有最高销售额 (total_amt_usd) 的销售代表的姓名。
2. 对于具有最高销售额 (total_amt_usd) 的区域，总共下了多少个订单？
3. 对于购买标准纸张数量 (standard_qty) 最多的客户（在作为客户的整个时期内），有多少客户的购买总数（total）比该用户的购买总数（total）更多？
4. 对于（在作为客户的整个时期内）总消费 (total_amt_usd) 最多的客户，他们在每个渠道上有多少 web_events？
5. 对于总消费前十名的客户，他们的平均终身消费 (total_amt_usd) 是多少？
6. 比所有客户的平均消费高的企业平均终身消费 (total_amt_usd) 是多少？

总结

这节课是编写高级 SQL 的第一节课。可以认为，高级功能子查询和 CTE 是企业中的分析师最常用的工具。能够将问题分解为必要的表格并使用生成的表格找到答案是非常实用的技能。

如果你第一次尝试时，没有得出正确答案，别担心，可以再试一次！此外，你也可以尝试自己想出一些答案，看看是否可行。

这门课程的剩余部分属于分析师需要掌握的关键技能，但是到目前为止介绍的所有 SQL 技能足够你日常使用了。