## 1. 计算表格中的行数

接下来，你将详细学习提到的每个聚合函数以及在 SQL 中一直用到的一些其他聚合函数。我们开始吧！

NULL 是一种数据类型，表示 SQL 中没有数据。它们经常在聚合函数中被忽略了，在下个部分学习使用 COUNT 时你将首次接触到这一现象。

注意，NULL 与零不同，它们表示不存在数据的单元格。

在 WHERE 条件中表示 NULL 时，我们写成 IS NULL 或 IS NOT NULL。我们不使用 =，因为 NULL 在 SQL 中不属于值。但是它是数据的一个属性。

**NULL - 专家提示**

在以下两种常见情况下，你可能会遇到 NULL：

1. 在执行 LEFT JOIN 或 RIGHT JOIN 时，NULL 经常会发生。你在上节课见到了，左侧表格中的某些行在做连接时与右侧表格中的行如果不匹配，这些行在结果集中就会包含一些 NULL 值。
2. NULL 也可能是因为数据库中缺失数据。

试着手动数数每个表格的行数。以下是计算 accounts 表格中的行数示例：

```
SELECT COUNT(*)
FROM accounts;
```

我们也可以轻松地选择一列来放置聚合函数：

```
SELECT COUNT(accounts.id)
FROM accounts;
```

上述两个语句是对等的，但是并不始终这样，我们将在下个视频中见到这一例外情况。

注意：COUNT 不会考虑具有 NULL 值的行。因此，可以用来快速判断哪些行缺少数据。你将在下个页面中学习 GROUP BY，然后每个聚合函数就会更加有用。

## 2. 练习：SUM

与 COUNT 不同，你只能针对数字列使用 SUM。但是，SUM 将忽略 NULL 值，其他聚合函数也是这样（你将在接下来的课程中见到）。

**聚合函数提醒**

重要注意事项：聚合函数只能垂直聚合，即聚合列的值。如果你想对行进行计算，可以使用简单算术表达式。

如果你想复习下，我们在第一节课见过算术表达式，但是下一页面的练习应该会确保你依然记得如何跨行聚合数据。

以下是一个 SQL 环境，用于计算以下每个问题的答案。如果你遇到问题或想要查看答案，可以在下一页面的顶部找到答案。

1. 算出 orders 表格中的 poster_qty 纸张总订单量。
2. 算出 orders 表格中 standard_qty 纸张的总订单量。
3. 根据 orders 表格中的 total_amt_usd 得出总销售额。
4. 算出 orders 表格中每个订单在 standard 和 gloss 纸张上消费的数额。结果应该是表格中每个订单的金额。
5. 每个订单的 price/standard_qty 纸张各不相同。我想得出 orders 表格中每个销售机会的这一比例。

## 3. 练习：MIN、MAX 与 AVG

注意，此处我们同时获得了每个纸张类型的 MIN 和 MAX 订单量。但是，你也可以单独计算每个类型的订单量。

注意，MIN 和 MAX 聚合函数也会忽略 NULL 值。请参阅以下专家提示，了解关于 MAX 与 MIN 的实用技巧。

**专家提示**

从功能上来说，MIN 和 MAX 与 COUNT 相似，它们都可以用在非数字列上。MIN 将返回最小的数字、最早的日期或按字母表排序的最之前的非数字值，具体取决于列类型。MAX 则正好相反，返回的是最大的数字、最近的日期，或与“Z”最接近（按字母表顺序排列）的非数字值。

与其他软件类似，AVG 返回的是数据的平均值，即列中所有的值之和除以列中值的数量。该聚合函数同样会忽略分子和分母中的 NULL 值。

如果你想将 NULL 当做零，则需要使用 SUM 和 COUNT。但是，如果 NULL 值真的只是代表单元格的未知值，那么这么做可能不太合适。

**MEDIAN - 专家提示**

注意，中值可能是更好的衡量方式，但是仅使用 SQL 非常棘手，以至于有时候在面试中就会提到关于中值方面的问题。

根据 SQL 表格信息回答以下问题。如果你遇到问题或想要对比检查你的答案，可以在下一页面的顶部找到我的答案。

1. 最早的订单下于何时？
2. 尝试执行和第一个问题一样的查询，但是不使用聚合函数。
3. 最近的 web_event 发生在什么时候？
4. 尝试以另一种方式执行上个问题的查询，不使用聚合函数。
5. 算出每个订单在每种纸张上消费的平均 (AVERAGE) 金额，以及每个订单针对每种纸张购买的平均数量。最终答案应该有 6 个值，每个纸张类型平均销量对应一个值，以及平均数量对应一个值。
6. 我相信你都渴望知道在所有订单上消费的中值 total_usd 是多少？虽然这一概念已经超出我们的范围。注意，这比我们到目前为止介绍的基本内容要高深一点，但是我们可以按照以下方式对答案进行硬编码。

## 4. 练习：GROUP BY

主要知识点包括：

1. GROUP BY 可以用来在数据子集中聚合数据。例如，不同客户、不同区域或不同销售代表分组。
2. SELECT 语句中的任何一列如果不在聚合函数中，则必须在 GROUP BY 条件中。
3. GROUP BY 始终在 WHERE 和 ORDER BY 之间。
4. ORDER BY 有点像电子表格软件中的 SORT。

**GROUP BY - 专家提示**

在深入了解如何使用 GROUP BY 语句聚合函数之前，需要注意的是，SQL 在 LIMIT 条件之前评估聚合函数。如果不按任何列分组，则结果是 1 行，没有问题。如果按照某列分组，该列中存在大量的唯一值，超出了 LIMIT 上限，则系统会照常计算聚合结果，但是结果中会忽略某些行。

这实际上是比较不错的方式，因为你知道你将获得正确的聚合结果。如果 SQL 将表格裁剪到 100 行，然后进行聚合，结果将完全不同。上述查询的结果超过了 100 行，因此是个很好的示例。在下一部分，使用该 SQL 表格并尝试删掉 LIMIT，然后再次运行查询，看看有哪些变化。

**GROUP BY 注意事项**

现在你已经了解了 JOIN、GROUP BY 和聚合函数，SQL 开始彰显真正的强大作用了。请尝试回答以下几个问题，检验下你的技能！

**问题：GROUP BY**

根据以下 SQL 表格信息回答以下问题。如果你遇到问题或想要对比检查你的答案，可以在下一页面的顶部找到我的答案。

一个判断难点是，何时使用某个聚合函数或其他 SQL 功能最简单。请尝试回答以下问题，看看你能否找到最简单的解决方案。

1. 哪个客户（按照名称）下的订单最早？你的答案应该包含订单的客户名称和日期。
2. 算出每个客户的总销售额（单位是美元）。答案应该包括两列：每个公司的订单总销售额（单位是美元）以及公司名称。
3. 最近的 web_event 是通过哪个渠道发生的，与此 web_event 相关的客户是哪个？你的查询应该仅返回三个值：日期、渠道和客户名称。
4. 算出 web_events 中每种渠道的次数。最终表格应该有两列：渠道和渠道的使用次数。
5. 与最早的 web_event 相关的主要联系人是谁？
6. 每个客户所下的最小订单是什么（以总金额（美元）为准）。答案只需两列：客户名称和总金额（美元）。从最小金额到最大金额排序。
7. 算出每个区域的销售代表人数。最早表格应该包含两列：区域和 sales_reps 数量。从最少到最多的代表人数排序。

## 5. 练习：GROUP BY（第二部分）

主要知识点：

1. 你可以同时按照多列分组，正如此处所显示的那样。这样经常可以在大量不同的细分中更好地获得聚合结果。

2. ORDER BY 条件中列出的列顺序有区别。你是从左到右让列排序。

**GROUP BY - 专家提示**

1. GROUP BY 条件中的列名称顺序并不重要，结果还是一样的。如果运行相同的查询并颠倒 GROUP BY 条件中列名称的顺序，可以看到结果是一样的。

2. 和 ORDER BY 一样，你可以在 GROUP BY 条件中用数字替换列名称。仅当你对大量的列分组时，或者其他原因导致 GROUP BY 条件中的文字过长时，才建议这么做。

3. 提醒一下，针对 SELECT 选择出来的列，任何不在聚合函数中的列，必须出现在 GROUP BY 语句中。如果忘记了，可能会遇到错误。但是，即使查询可行，最后的结果可能也不会正确！

**问题：GROUP BY（第二部分）**

根据以下 SQL 表格信息回答以下问题。如果你遇到问题或想要对比检查你的答案，可以在下一页面的顶部找到我的答案。

1. 对于每个客户，确定他们在订单中购买的每种纸张的平均数额。结果应该有四列：客户名称一列，每种纸张类型的平均数额一列。

2. 对于每个客户，确定在每个订单中针对每个纸张类型的平均消费数额。结果应该有四列：客户名称一列，每种纸张类型的平均消费数额一列。

3. 确定在 web_events 表格中每个销售代表使用特定渠道的次数。最终表格应该有三列：销售代表的名称、渠道和发生次数。按照最高的发生次数在最上面对表格排序。

4. 确定在 web_events 表格中针对每个地区特定渠道的使用次数。最终表格应该有三列：区域名称、渠道和发生次数。按照最高的发生次数在最上面对表格排序。

## 6. 练习：DISTINCT

你可以将 DISTINCT 看做仅返回特定列的唯一值的函数。

**DISTINCT - 专家提示**

需要注意的是，在使用 DISTINCT 时，尤其是在聚合函数中使用时，会让查询速度有所减慢。

**问题：DISTINCT**

根据以下 SQL 表格信息回答以下问题。如果你遇到问题或想要对比检查你的答案，可以在下一页面的顶部找到我的答案。

1. 使用 DISTINCT 检查是否有任何客户与多个区域相关联？

2. 有没有销售代表要处理多个客户？

## 7. 练习：HAVING

**HAVING - 专家提示**

HAVING 是过滤被聚合的查询的 “整洁”方式，但是通常采用子查询的方式来实现。本质上，只要你想对通过聚合创建的查询中的元素执行 WHERE 条件，就需要使用 HAVING。 练习题 经常你会对WHERE和HAVING之间的差别感动困惑。关于HAVING和WHERE的语句，请选出以下所有正确的语句。

- ☐ WHERE子集根□据逻辑条件对返回的数据进行筛选。
- ☐ WHERE出现在FROM，JOIN和ON条件之后，但是在GROUP BY 之前。
- ☐ HAVING出现在GROUP BY□条件之后，但是在ORDER BY 条件之前。
- ☐ HAVING和WHERE相似，但是□他适合涉及聚合的逻辑语句。

**问题：HAVING**

根据以下 SQL 表格信息回答以下问题。如果你遇到问题或想要对比检查你的答案，可以在下一页面的顶部找到我的答案。

1. 有多少位销售代表需要管理超过 5 个客户？
2. 有多少个客户具有超过 20 个订单？
3. 哪个客户的订单最多？
4. 有多少个客户在所有订单上消费的总额超过了 30,000 美元？
5. 有多少个客户在所有订单上消费的总额不到 1,000 美元？
6. 哪个客户消费的最多？
7. 哪个客户消费的最少？
8. 哪个客户使用 facebook 作为与消费者沟通的渠道超过 6 次？
9. 哪个客户使用 facebook 作为沟通渠道的次数最多？
10. 哪个渠道是客户最常用的渠道？

## 8. 练习：DATE函数

在 SQL 中，按照日期列分组通常不太实用，因为这些列可能包含小到一秒的交易数据。按照如此详细的级别保存信息即有好处，又存在不足之处，因为提供了非常准确的信息（好处），但是也让信息分组变得很难（不足之处）。

幸运的是，有很多 SQL 内置函数可以帮助我们改善日期处理体验。

这里，我们看到日期存储为年、月、日、小时、分钟、秒，可以帮助我们截取信息。在下个部分，你将看到在 SQL 中我们可以使用大量函数来利用这一功能。

首先，我们要介绍的日期函数是 DATE_TRUNC。

DATE_TRUNC 使你能够将日期截取到日期时间列的特定部分。常见的截取依据包括日期、月份 和 年份。这是一篇 MODE 发表的精彩博文，介绍了关于此函数的强大功能。

DATE_PART 可以用来获取日期的特定部分，但是注意获取 month 或 dow 意味着无法让年份按顺序排列。而是按照特定的部分分组，无论它们属于哪个年份。

要了解其他日期函数，请参阅这篇文档，但是上面介绍的函数绝对够你入门了！

**问题：使用 DATE**

根据以下 SQL 表格信息回答以下问题。如果你遇到问题或想要对比检查你的答案，可以在下一页面的顶部找到我的答案。

1. Parch & Posey 在哪一年的总销售额最高？数据集中的所有年份保持均匀分布吗？
2. Parch & Posey 在哪一个月的总销售额最高？数据集中的所有月份保持均匀分布吗？
3. Parch & Posey 在哪一年的总订单量最多？数据集中的所有年份保持均匀分布吗？
4. Parch & Posey 在哪一个月的总订单量最多？数据集中的所有月份保均匀分布吗？
5. Walmart 在哪一年的哪一个月在铜版纸上的消费最多？

## 9. 练习：CASE

**CASE - 专家提示**

1. CASE 语句始终位于 SELECT 条件中。
2. CASE 必须包含以下几个部分：WHEN、THEN 和 END。ELSE 是可选组成部分，用来包含不符合上述任一 CASE 条件的情况。
3. 你可以在 WHEN 和 THEN 之间使用任何条件运算符编写任何条件语句（例如 WHERE），包括使用 AND 和 OR 连接多个条件语句。
4. 你可以再次包含多个 WHEN 语句以及 ELSE 语句，以便处理任何未处理的条件。

示例

在第一节课的练习中，你看到了以下问题：

> 创建一列用于将 standard_amt_usd 除以 standard_qty，以便计算每个订单的标准纸张的单价，将结果限制到前 10 个订单，并包含 id 和 account_id 字段。注意 - 如果你的答案正确，系统将显示一个错误，这是因为你除以了 0。当你在下个部分学习 CASE 语句时，你将了解如何让此查询不会报错。

我们来看看如何使用 CASE 语句来避免这一错误。

```
SELECT id, account_id, standard_amt_usd/standard_qty AS unit_price
FROM orders
LIMIT 10;
```

现在我们使用一个 CASE 语句，这样的话，一旦 standard_qty 为 0，我们将返回 0，否则返回 unit_price。

```
SELECT account_id, CASE WHEN standard_qty = 0 OR standard_qty IS NULL THEN 0
                        ELSE standard_amt_usd/standard_qty END AS unit_price
FROM orders
LIMIT 10;
```

该语句的第一部分将捕获任何分母为 0 并导致错误的情况，其他部分将按照常规步骤相除。你将发现对于标准纸张，所有客户的单价是 4.99 美元。这样比较合理，不会波动，并且比在上节课中向分母上加 1 来暂时解决错误这一方法更准确。

你可以使用下面的数据自己尝试下。

这道题有点难。尝试自己运行查询，确保知道所发生的情况。下一部分将让你自己尝试编写 CASE 语句。在此视频中，我们演示了以下情况：使用 WHERE 条件获取相同的信息意味着一次只能从 CASE 中获取一组数据。

像这样将数据分成几列有一些优势，这取决于你要执行的操作。但通常，这种级别的划分可能使用其他编程语言更简单，而不是使用 SQL。

**问题：CASE**

根据以下 SQL 表格信息回答以下问题。如果你遇到问题或想要对比检查你的答案，可以在下一页面的顶部找到我的答案。

1. 我们想要根据相关的消费量了解三组不同的客户。最高的一组是终身价值（所有订单的总销售额）大于 200,000 美元的客户。第二组是在 200,000 到 100,000 美元之间的客户。最低的一组是低于 under 100,000 美元的客户。请提供一个表格，其中包含与每个客户相关的级别。你应该提供客户的名称、所有订单的总销售额和级别。消费最高的客户列在最上面。

2. 现在我们想要执行和第一个问题相似的计算过程，但是我们想要获取在 2016 年和 2017 年客户的总消费数额。级别和上一个问题保持一样。消费最高的客户列在最上面。

3. 我们想要找出绩效最高的销售代表，也就是有超过 200 个订单的销售代表。创建一个包含以下列的表格：销售代表名称、订单总量和标为 top 或 not 的列（取决于是否拥有超过 200 个订单）。销售量最高的销售代表列在最上面。

4. 之前的问题没有考虑中间水平的销售代表或销售额。管理层决定也要看看这些数据。我们想要找出绩效很高的销售代表，也就是有超过 200 个订单或总销售额超过 750000 美元的销售代表。中间级别是指有超过 150 个订单或销售额超过 500000 美元的销售代表。创建一个包含以下列的表格：销售代表名称、总订单量、所有订单的总销售额，以及标为 top、middle 或 low 的列（取决于上述条件）。在最终表格中将销售额最高的销售代表列在最上面。根据上述标准，你可能会见到几个表现很差的销售代表！

## 总结

每个部分都添加了标签，以便你回顾相应的内容。我故意将特定部分的答案没有放在带标签的部分，因为这样可以促使你自己去解决问题。

现在你已经掌握了关于 SQL 的大量实用技能。连接和聚合相结合是让 SQL 成为如此强大的工具的原因之一。

如果你对某个部分不太懂，建议你再重新复习这一部分。熟能生巧，不过太长时间都陷在某个问题上也不太合适！