四 使用 WITH (Common Table Expressions) 公用表表达式

> 在创建报表时，我们经常会在一张报表中先计算一个指标，然后计算这个指标的平均值，此时可以使用WITH CTE
>
> 比如，有如下需求，创建报表计算：
>
> - 平均订单价值是多少?
> - 运输到不同国家/地区的单均订购商品数量是多少?
>
> 所有这些问题都要求我们执行多级聚合

学习目标

- 熟练使用 WITH (Common Table Expressions) 公用表表达式进行复杂查询

1 多层聚合

- 需求：计算每个订单支付的平均总价（折扣前）
- 可以使用 `SUM（unit_price *quantity）` 来计算单个订单的总价格，需要对这个结果再计算平均值
- 平均值计算使用 `AVG()` 函数，但不能嵌套调用 `AVG(SUM(unit_price * quantity))`

练习37

```sql
WITH order_total_prices AS (
  SELECT
    o.order_id,
    SUM(unit_price * quantity) AS total_price
  FROM orders o
  JOIN order_items oi
    ON o.order_id = oi.order_id
  GROUP BY o.order_id
)
SELECT
  AVG(total_price) AS avg_total_price
FROM order_total_prices;
```

- 上面的SQL满足了我们的需求，这里用到了 WITH (Common Table Expressions) 公用表表达式

查询结果

avg_total_price
1631.877819

2 WITH (Common Table Expressions) 公用表表达式 介绍

- 公用表表达式 **CTE （ common table expression）**与子查询十分类似，
- **CTE** 的基础语法如下：

```sql
WITH some_name AS (
  -- your CTE
)
SELECT
  ...
FROM some_name
```

- 需要给CTE起一个名字（上面的例子中使用了 `some_name` ），具体的查询语句写在括号中
- 在括号后面，就可以通过 `SELECT` 将CTE的结果当作一张表来使用
- 将CTE称为“内部查询”，其后的部分称为“外部查询”

- 需要先定义CTE，即在外部查询的 `SELECT` 之前定义CTE

再看一下前面的案例：

```
WITH order_total_prices AS (
  SELECT
    o.order_id,
    SUM(unit_price * quantity) AS total_price
  FROM orders o
  JOIN order_items oi
    ON o.order_id = oi.order_id
  GROUP BY o.order_id
)
SELECT
  AVG(total_price) AS avg_total_price
FROM order_total_prices;
```

- 上面的SQL中， CTE称为 `order_total_prices` ，包含两列： `order_id` 和 `total_price`
- 在外部查询中，我们使用AVG（total_price）汇总 `total_price` 列

练习38

```
WITH order_total_prices AS (
  SELECT
    o.order_id,
    SUM(unit_price * quantity) AS total_price
  FROM orders o
  JOIN order_items oi
    ON o.order_id = oi.order_id
  GROUP BY o.order_id
)
SELECT
  AVG(total_price) AS avg_total_price
FROM order_total_prices;
```

- 修改上面的SQL，计算打折后的平均订单价格

```sql
WITH order_discounted_prices AS (
  SELECT
    o.order_id,
    SUM(unit_price * quantity * (1 - discount)) AS
total_discounted_price
  FROM orders o
  JOIN order_items oi
    ON o.order_id = oi.order_id
  GROUP BY o.order_id
)
SELECT
  AVG(total_discounted_price) AS avg_total_discounted_price
FROM order_discounted_prices;
```

查询结果

avg_total_discounted_price
1525.051855

3 CTE 讲解

第一步

- 接下来我们分布来实现一个CTE查询，需求是查询每个订单的平均商品数量

练习39

- 先创建一个简单查询，用来统计每个订单的商品总数量
- 结果有两列：`order_id` 和 `item_count`.

```sql
SELECT
  order_id,
  SUM(quantity) AS item_count
FROM order_items
GROUP BY order_id;
```

查询结果

order_id	item_count
11038	37
10782	1
10725	22
10423	34
10518	29
10356	62
10963	2
10596	59
10282	8
10658	255
10283	76
10579	31
10693	111
10896	31
10660	21
10253	102
10425	30
10774	52
10615	5
10514	233

第二步

- 接下来我们可以将上面的查询转换为CTE，并进一步计算所有订单的平均商品数量

练习40

```
WITH order_items_counts AS (
  SELECT
    order_id,
    SUM(quantity) AS item_count
  FROM order_items
  GROUP BY order_id
)
...
```

- 完成上面的SQL，计算所有订单的平均商品数量

```
WITH order_items_counts AS (
  SELECT
    order_id,
    SUM(quantity) AS item_count
  FROM order_items
  GROUP BY order_id
)
SELECT
  AVG(item_count) AS avg_item_count
FROM order_items_counts;
```

查询结果

avg_item_count
61.8277

练习41

- 在创建报表需要用到多级聚合时，可以先独立创建子查询，然后转换为CTE
- 需求：统计每个类别中的平均产品数量`avg_product_count`.是多少？通过CTE实现

```
WITH products_category_count AS (
  SELECT
    category_id,
    COUNT(product_id) AS count_products
  FROM products
  GROUP BY category_id
)
SELECT
  AVG(count_products) AS avg_product_count
FROM products_category_count;
```

查询结果

avg_product_count
9.6250000000000000

4 CTE 另外一种写法

- 使用CTE时，也可以选择使用另外一种语法：

```
WITH order_total_prices (order_id, total_price) AS (
  SELECT
    o.order_id,
    SUM(unit_price * quantity)
  FROM orders o
  JOIN order_items oi
    ON o.order_id = oi.order_id
  GROUP BY o.order_id
)

SELECT
  AVG(total_price) AS avg_total_price
FROM order_total_prices;
```

- 这种写法的特点是在AS前将CTE创建的临时表字段定义好，在内部查询中，不需要使用as 起别名
- 具体使用哪种方式，根据个人喜好选择

练习42

- 用上面介绍的语法改写下面的SQL中的CTE

```
WITH products_category_count AS (
  SELECT
    category_id,
    COUNT(product_id) AS count_products
  FROM products
  GROUP BY category_id
)
SELECT
  AVG(count_products) AS avg_product_count
FROM products_category_count;
```

查询结果

avg_product_count
9.6250000000000000

5 多层聚合

- 前面的CTE练习中，我们的结果都只返回了一个值，接下来我们看一下稍微复杂的情况
- 需求：查找来自加拿大的每个客户的平均订单价值

```
WITH order_total_prices AS (
  SELECT
    o.order_id,
    o.customer_id,
    SUM(unit_price * quantity) AS total_price
  FROM orders o
  JOIN order_items oi
    ON o.order_id = oi.order_id
  GROUP BY o.order_id, o.customer_id
)

SELECT
  c.customer_id,
  c.company_name,
```

```
  AVG(total_price) AS avg_total_price
FROM order_total_prices OTP
JOIN customers c
  ON OTP.customer_id = c.customer_id
WHERE c.country = 'Canada'
GROUP BY c.customer_id, c.company_name;
```

- 在上面的查询中，我们首先编写一个CTE计算每个订单的总价（折扣前），并将 `customer_id` 列放在 `SELECT` 子句中，以便在外部查询中引用
- 在外部查询中，将数据按 `customer_id` 分组，将CTE的结果与 `customers` 表 JOIN 起来，计算自加拿大的每个客户的平均订单价值

练习43

- 需求：创建报表，统计华盛顿地区（WA）每位员工所处理订单的订单平均价格
- 报表中包含如下列：
  - `employee_id`，`first_name`，`last_name`,和 `avg_total_price`
  - 员工ID，名字，姓氏，平均订单总价
- 提示:
  - 通过CTE计算所有订单的总价格，并将处理该订单的员工ID一起返回
  - 在外部查询中，将CTE与 `employees` 表JOIN起来，计算平均值，显示所有所需信息，并按地区过滤结果

```
WITH order_total_prices AS (
  SELECT
    o.order_id,
    o.employee_id,
    SUM(unit_price * quantity) AS total_price
  FROM orders o
  JOIN order_items oi
    ON o.order_id = oi.order_id
  GROUP BY o.order_id,
    o.employee_id
)
SELECT
  e.employee_id,
```

```
  e.first_name,
  e.last_name,
  AVG(total_price) AS avg_total_price
FROM order_total_prices otp
JOIN employees e
  ON otp.employee_id = e.employee_id
WHERE e.region = 'WA'
GROUP BY e.employee_id,
  e.first_name,
  e.last_name;
```

查询结果

employee_id	first_name	last_name	avg_total_price
1	Nancy	Davolio	1643.444797
2	Andrew	Fuller	1851.554792
3	John	Smith	1677.569291
4	Margaret	Peacock	1611.222258
8	Laura	Callahan	1281.740673

练习44

- 需求：创建报表，统计运输到不同国家/地区的订单中，不同商品的平均数量
- 报表包含 `ship_country` 和 `avg_distinct_item_count` 列，并按数量降序排列

```
WITH order_distinct_items AS (
  SELECT
    o.order_id,
    o.ship_country,
    COUNT(distinct product_id) AS distinct_item_count
  FROM orders o
  JOIN order_items oi
    ON o.order_id = oi.order_id
  GROUP BY o.order_id,
    o.ship_country
)
```

```sql
SELECT
  ship_country,
  AVG(distinct_item_count) AS avg_distinct_item_count
FROM order_distinct_items
GROUP BY ship_country
ORDER BY avg_distinct_item_count DESC;
```

查询结果

ship_country	avg_distinct_item_count
Austria	3.125
Belgium	2.9474
Ireland	2.8947
Switzerland	2.8889
USA	2.8852
Germany	2.6885
Norway	2.6667
Swede	2.6216
Mexico	2.5714
Venezuela	2.5652
Denmark	2.5556
Canada	2.5
Finland	2.4545
Brazil	2.4458
UK	2.4107
France	2.3896
Spain	2.3478
Portugal	2.3077
Poland	2.2857
Argentina	2.125
Italy	1.8929

练习45

- 需求：创建报表，统计每位员工2016年所有订单中处理的平均物品数量
  - 每一个订单中有多个订单明细项在 `order_items` 表中记录
  - `order_items` 表中的 `quantity` 字段记录了订单中每件商品购买的商品数量
  - 订单中的物品数量用订单明细表中的 `quantity` 求和进行计算
- 报表中包含如下列：
  - `first_name`，`last_name`，`avg_item_count`
  - 员工名字，员工姓氏，订单平均物品数量

```
WITH order_items_counts AS (
  SELECT
    o.order_id,
    o.employee_id,
    SUM(quantity) AS item_count
  FROM orders o
  JOIN order_items oi
    ON o.order_id = oi.order_id
  WHERE o.order_date >= '2016-01-01' AND o.order_date < '2017-
01-01'
  GROUP BY o.order_id,
    o.employee_id
)
SELECT
  e.first_name,
  e.last_name,
  AVG(item_count) AS avg_item_count
FROM order_items_counts oic
JOIN employees e
  ON oic.employee_id = e.employee_id
GROUP BY e.employee_id,
  e.first_name,
  e.last_name;
```

查询结果

first_name	last_name	avg_item_count
Nancy	Davolio	62.3077
Andrew	Fuller	67.8125
John	Smith	52.2222
Margaret	Peacock	72.4000
Steven	Buchanan	70.7273
Michael	Suyama	64.2000
Robert	King	44.0909
Laura	Callahan	48.5789
Anne	Dodsworth	115.000
John	Smith	40.0000

6 多级聚合 与 CASE WHEN 组合使用

- 我们还可以将多级聚合与自定义分类结合在一起
- 需求：通过用户下单数量对用户分三组
  - 订单数少于10的客户
  - 订单数为10-20的客户
  - 订单数超过20的客户

```
WITH customer_order_counts AS (
  SELECT
    customer_id,
    CASE
      WHEN COUNT(o.order_id) > 20
        THEN 'more than 20'
      WHEN COUNT(o.order_id) <= 20 AND COUNT(o.order_id) >= 10
        THEN 'between 10 and 20'
      ELSE 'less than 10'
    END AS order_count_cat
  FROM orders o
  GROUP BY customer_id
)
```

```
SELECT
  order_count_cat,
  COUNT(customer_id) AS customer_count
FROM customer_order_counts
GROUP BY order_count_cat;
```

- 在内部查询中（CTE），我们使用`CASE WHEN`构造将客户分组，客户的分组结果保存到“order_count_cat”列中
- 接下来，我们在外部查询中使用`order_count_cat`列和`COUNT()`函数来统计每组中的客户数量。

练习46

- 需求：创建报表统计高价值和低价值客户的数量
  - 客户在折扣前支付的所有订单的总价**大于$ 20,000**，则将该客户视为“高价值”
  - 否则，将它们视为“低值”
- 报表中包含两个字段
  - `category`（`'high-value'`或`'low-value'`），`customer_count`
  - “类别”（“高价值”或“低价值”）和“用户数量”

```
WITH customer_order_values AS (
  SELECT
    customer_id,
    CASE
      WHEN SUM(quantity * unit_price) > 20000
        THEN 'high-value'
      ELSE 'low-value'
    END AS category
  FROM orders o
  JOIN order_items oi
    ON o.order_id = oi.order_id
  GROUP BY customer_id
)
SELECT
  category,
  COUNT(customer_id) AS customer_count
FROM customer_order_values
```

```
GROUP BY category;
```

查询结果

category	customer_count
high-value	20
low-value	69

练习47

- 需求：统计非素食（“ category_id”，“ 6”或“ 8”）和素食（所有其他“ category_id”值）产品的平均数量
- 报表中包含两列：
  - `product_type` （`'vegetarian'`或`'non-vegetarian'`），`avg_product_count`
  - 产品类别（素食或非素食），平均产品数量

```
WITH product_counts AS (
  SELECT
    category_id,
    CASE
      WHEN category_id IN (6, 8)
        THEN 'non-vegetarian'
      ELSE 'vegetarian'
    END AS product_type,
    COUNT(*) AS product_count
  FROM products
  GROUP BY category_id
)
SELECT
  product_type,
  AVG(product_count) AS avg_product_count
FROM product_counts
GROUP BY product_type;
```

查询结果

product_type	avg_product_count
non-vegetarian	9.0000
vegetarian	9.8333

7 三层聚合

- 有时，我们需要多层聚合完成指标计算。 例如，我们可以计算每个客户的平均订单价值，然后找到最大平均值。

```
WITH order_values AS (
  SELECT
    customer_id,
    SUM(unit_price * quantity) AS total_price
  FROM orders o
  JOIN order_items oi
    ON o.order_id = oi.order_id
  GROUP BY o.order_id, customer_id
),

customer_averages AS (
  SELECT
    customer_id,
    AVG(total_price) AS avg_total_price
  FROM order_values
  GROUP BY customer_id
)

SELECT
  MAX(avg_total_price) AS maximal_average
FROM customer_averages;
```

- 注意！这里引入了**两个CTE** 。 第二个CTE之前没有 `WITH` 关键字； 只需要用逗号隔开即可。
  - 在第一个CTE中计算每个订单的总金额
  - 在第二个CTE中，根据第一个CTE计算每个客户的平均订单价值
  - 最后，在外部查询中从第二个CTE中找到最大平均值。

练习48

- 需求：计算每位员工的平均订单价值（折后），找到最小平均值（`minimal_average`列）和最大平均值（`maximal_average`列）
- **提示：**使用两个CTE
  - 第一个CTE中，计算每位员工的折扣后的订单总价
  - 第二个CTE中，计算每个员工折扣后的平均订单价值
  - 最后，在外部查询中，使用`MIN()`和`MAX()`找到最小平均值（`minimal_average`列）和最大平均值（`maximal_average`列）

```
WITH order_values AS (
  SELECT
    employee_id,
    SUM(unit_price * quantity * (1 - discount)) AS
total_discount_price
  FROM orders o
  JOIN order_items oi
    ON o.order_id = oi.order_id
  GROUP BY o.order_id, employee_id
),
customer_averages AS (
  SELECT
    employee_id,
    AVG(total_discount_price) AS avg_discount_total_price
  FROM order_values
  GROUP BY employee_id
)
SELECT
  MIN(avg_discount_total_price) AS minimal_average,
  MAX(avg_discount_total_price) AS maximal_average
FROM customer_averages;
```

查询结果

minimal_average	maximal_average
448.000000	1797.862012

8 三层聚合2

- 需求：创建报表查找所有处理的订单数量大于平均水平的员工

```
WITH order_count_employees AS (
  SELECT
    employee_id,
    COUNT(order_id) AS order_count
  FROM orders
  GROUP BY employee_id
),

avg_order_count AS (
  SELECT
    AVG(order_count) AS avg_order_count
  FROM order_count_employees
)

SELECT
  employee_id,
  order_count,
  avg_order_count
FROM order_count_employees,
  avg_order_count
WHERE order_count > avg_order_count;
```

- 在第一个CTE中，我们计算每个员工处理的订单数量
- 在第二个CTE中，我们根据第一个CTE找到平均订单数
- 在外部查询中：我们只需在 `FROM` 子句中提供两个CTE名称，并用逗号分隔即可。 因此，我们可以在 `WHERE` 子句中引用来自两个CTE的列，并在 `SELECT` 子句中显示所有列
- 在最后的结果中显示每个具有高于平均订单数的员工及其订单数。 并显示平均计数以供参考

employee_id	order_count	avg_order_count
1	123	83
2	96	83
3	127	83
4	156	83
8	104	83

练习49

- 需求：在运往意大利的订单中，显示所有总价值高于平均水平（折扣前）的订单
- 结果包含如下字段：
  - `order_id`，`order_value`，和 `avg_order_value`
  - `avg_order_value` 列为平均订单金额

```
WITH order_values AS (
  SELECT
    o.order_id,
    SUM(quantity * unit_price) AS order_value
  FROM orders o
  JOIN order_items oi
    ON o.order_id = oi.order_id
  WHERE ship_country = 'Italy'
  GROUP BY o.order_id
),
avg_order_value AS (
  SELECT
    AVG(order_value) AS avg_order_value
  FROM order_values
)
SELECT
  order_id,
  order_value,
  avg_order_value
FROM order_values, avg_order_value
WHERE order_value > avg_order_value;
```

查询结果

order_id	order_value	avg_order_value
10300	608.00	596.612500
10404	1675.00	596.612500
10635	1380.25	596.612500
10727	1710.00	596.612500
10784	1650.00	596.612500
10812	1852.00	596.612500
10818	833.00	596.612500
10908	698.00	596.612500
10939	750.00	596.612500
11010	645.00	596.612500
11026	1030.00	596.612500

小结

接下来对本小节内容进行总结

1. 多级聚合需要共用表表达式（CTE）。 最基本的CTE如下所示：

```
WITH inner_query_name AS (
  SELECT
    function(column_1) AS agg_column_1
  FROM ...
)
SELECT
  function(agg_column_1)
FROM inner_query_name;
```

2. 我们可以在外部查询中添加一个 `GROUP BY` 子句，以显示多个业务对象的多级聚合：

```
WITH inner_query_name AS (
  SELECT
    function(column_1) AS agg_column_1,
    column_2
  FROM ...
)
SELECT function(agg_column_1)
FROM inner_query_name
GROUP BY column_2;
```

3. 我们还可以使用 `CASE WHEN` 构造将我们自己的分类引入多级聚合中：

```
WITH inner_query_name AS (
  SELECT
    function(column_1) AS agg_column_1,
    CASE
      WHEN ...
    END AS column_2
  FROM ...
)
SELECT function(agg_column_1)
FROM inner_query_name
GROUP BY column_2;
```

4. 最后，我们可以在一个查询中使用多个CTE：

```
WITH cte_1 AS (...),
cte_2 AS (...),
...,
cte_n AS (...)
SELECT
  ...
```

5. 注意事项：
   - with子句必须在引用的select语句**之前定义**,同级with关键字**只能使用一次**,多个只能**用逗号分割**；最后一个with 子句与下面的查询之间不能有逗号，**只通过右括号分割**,with 子句的查询必须用括号括起来.
   - **如果定义了with子句，但其后没有跟select查询，则会报错!**

- 前面的with子句定义的查询在后面的with子句中可以使用。但是一个with子句内部不能嵌套with子句！

练习50

- 需求: 创建报表统计运送到北美地区（美国、加拿大）的订单，平均每名员工处理的订单数量
  - 结果字段名字：`avg_order_count`

```
WITH order_countemployees AS (
  SELECT
    employee_id,
    COUNT(order_id) AS order_count
  FROM orders
  WHERE ship_country IN ('USA', 'Canada')
  GROUP BY employee_id
)
SELECT
  AVG(order_count) AS avg_order_count
FROM order_countemployees;
```

查询结果

avg_order_count
16.8889

练习51

- 需求：查找每个客户的平均订单价值（折扣后）
- 结果字段：`customer_id`和`avg_discounted_price`

```
WITH order_values AS (
  SELECT
    customer_id,
    oi.order_id,
    SUM(unit_price * quantity * (1 - discount)) AS
total_discounted_price
  FROM orders o
  JOIN order_items oi
    ON o.order_id = oi.order_id
```

```
  GROUP BY customer_id, oi.order_id
)
SELECT
  customer_id,
  AVG(total_discounted_price) AS avg_discounted_price
FROM order_values
GROUP BY customer_id;
```

查询结果

customer_id	avg_discounted_price
VINET	296
TOMSP	796.356667
HANAR	2345.812143
VICTE	918.243
SUPRD	2007.398333
CHOPS	1543.61
RICSU	1934.3779
WELLI	674.244444
HILAA	1264.931333
ERNSH	3495.832617
CENTC	100.8
OTTIK	1249.62
QUEDE	740.534444
RATTC	2838.766694
FOLKO	1556.1875
BLONP	1684.916364
WARTH	1043.246833
FRANK	1777.103967
GROSR	744.35
WHITC	1954.543214
SPLIR	1271.292222
QUICK	3938.475179
MAGAA	717.6215
TORTU	1081.215

customer_id	avg_discounted_price
MORGK	1008.44
BERGS	1384.865417
LEHMS	1284.094
ROMEY	293.458
LILAS	1148.328571
RICAR	1131.890909
REGGC	587.353333
BSBEV	608.99
COMMI	762.15
TRADH	1141.777333
HUNGO	2630.521316
WANDK	958.8425
GODOS	1144.636
OLDWO	1517.74625
LONEP	532.325
ANATR	350.7375
THEBI	840.25
DUMO	403.975
ISLAT	614.63
PERIC	707.033333
KOENE	2207.741714
SAVEA	3366.514516
BOLID	1410.95
FURIB	803.427813

customer_id	avg_discounted_price
BONAP	1291.956029
MEREP	2220.937692
PRINI	1008.988
SIMOB	2402.4425
FAMIA	586.792857
LAMAI	666.3
PICCO	2312.886
AROUT	1030.05
SEVES	1801.702778
DRACD	627.201667
EASTC	1845.129375
ANTO	1003.425357
GALED	167.34
VAFFE	1440.356818
QUEE	1978.269038
WOLZA	504.564286
HUNGC	612.64
SANTG	955.858333
BOTTM	1485.828571
LINOD	1373.047083
FOLIG	2333.38
OCEA	692.04
FRANS	257.616667
GOURL	934.903889

customer_id	avg_discounted_price
CONSH	573.033333
RANCH	568.82
LAZYK	178.5
LAUGB	174.166667
BLAUS	462.828571
NORTS	216.333333
CACTU	302.466667
GREAL	1682.495455
MAISD	1390.867857
TRAIH	523.733333
LETSS	769.118125
WILMK	451.621429
THECR	649.08
ALFKI	712.166667
FRANR	1057.386667
SPECD	605.8375
LACOR	498.0125

练习52

- 需求：统计所有订单中，低价商品和高价商品的平均进货量（到底在进便宜货，还是高价货）
- 报表中包含两个字段：`price_category`，`avg_products_on_order`
  - `price_category` 字段取值：`'cheap'`（便宜）`unit_price` <=20 其余为 `'expensive'` 贵
  - `avg_products_on_order` 每类商品的平均进货数量

```sql
WITH product_info AS (
  SELECT
    product_id,
    CASE
      WHEN unit_price <= 20.0 THEN 'cheap'
      ELSE 'expensive'
    END AS price_category,
      units_on_order
  FROM products
)
SELECT
  price_category,
  AVG(units_on_order) AS avg_products_on_order
FROM product_info
GROUP By price_category;
```

查询结果

price_category	avg_products_on_order
expensive	5.945945945945946
cheap	14