LAB - HAVING

In this lab, you will learn how to use the SQL Server `HAVING` clause to filter the groups based on specified conditions.

The `HAVING` clause is often used with the `GROUP BY` clause to filter groups based on a specified list of conditions. The following illustrates the `HAVING` clause syntax:

```
SELECT
    select_list
FROM
    table_name
GROUP BY
    group_list
HAVING
    conditions;
```

In this syntax, the `GROUP BY` clause summarizes the rows into groups and the `HAVING` clause applies one or more conditions to these groups. Only groups that make the conditions evaluate to `TRUE` are included in the result. In other words, the groups for which the condition evaluates to `FALSE` or `UNKNOWN` are filtered out.

Because SQL Server processes the `HAVING` clause after the `GROUP BY` clause, you cannot refer to the aggregate function specified in the select list by using the column alias. The following query will fail:

```
SELECT
    column_name1,
    column_name2,
    aggregate_function (column_name3) column_alias
FROM
    table_name
GROUP BY
    column_name1,
    column_name2
HAVING
    column_alias > value;
```

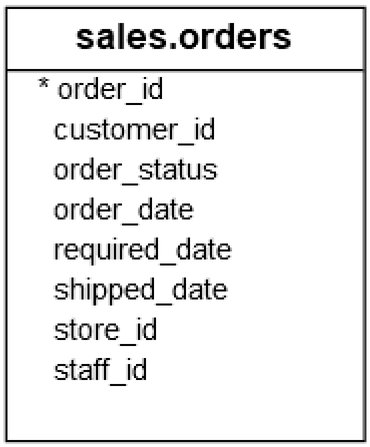
Instead, you must use the aggregate function expression in the `HAVING` clause explicitly as follows:

```
SELECT
    column_name1,
    column_name2,
    aggregate_function (column_name3) alias
FROM
    table_name
GROUP BY
    column_name1,
    column_name2
HAVING
    aggregate_function (column_name3) > value;
```

Examples

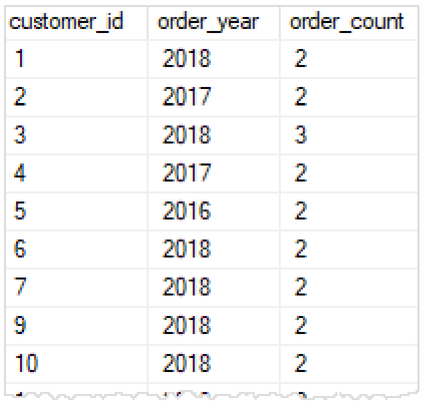
`HAVING` with the `COUNT` function example

See the following `orders` table from the sample database



The following statement uses the `HAVING` clause to find the customers who placed at least two orders per year:

```
SELECT
    customer_id,
    YEAR (order_date),
    COUNT (order_id) order_count
FROM
    sales.orders
GROUP BY
    customer_id,
    YEAR (order_date)
HAVING
    COUNT (order_id) >= 2
ORDER BY
    customer_id;
```



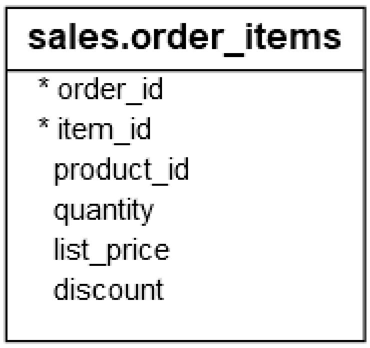
customer_id	order_year	order_count
1	2018	2
2	2017	2
3	2018	3
4	2017	2
5	2016	2
6	2018	2
7	2018	2
9	2018	2
10	2018	2

In this example:

- First, the `GROUP BY` clause groups the sales order by customer and order year. The `COUNT()` function returns the number of orders each customer placed in each year.
- Second, the `HAVING` clause filtered out all the customers whose number of orders is less than two.

`HAVING` clause with the `SUM()` function example

Consider the following `order_items` table:

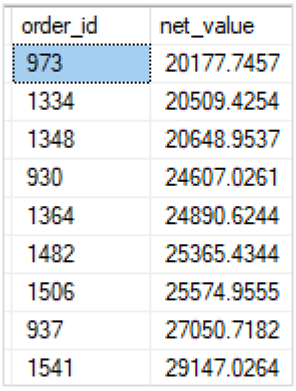


sales.order_items

* order_id
* item_id
product_id
quantity
list_price
discount

The following statement finds the sales orders whose net values are greater than 20,000:

```
SELECT
    order_id,
    SUM (
        quantity * list_price * (1 - discount)
    ) net_value
FROM
    sales.order_items
GROUP BY
    order_id
HAVING
    SUM (
        quantity * list_price * (1 - discount)
    ) > 20000
ORDER BY
    net_value;
```



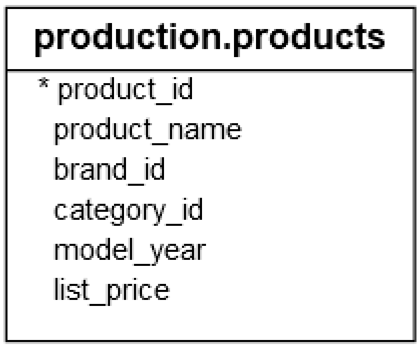
order_id	net_value
973	20177.7457
1334	20509.4254
1348	20648.9537
930	24607.0261
1364	24890.6244
1482	25365.4344
1506	25574.9555
937	27050.7182
1541	29147.0264

In this example:

- First, the `SUM()` function returns the net values of sales orders.
- Second, the `HAVING` clause filters the sales orders whose net values are less than or equal to 20,000.

`HAVING` clause with `MAX` and `MIN` functions example

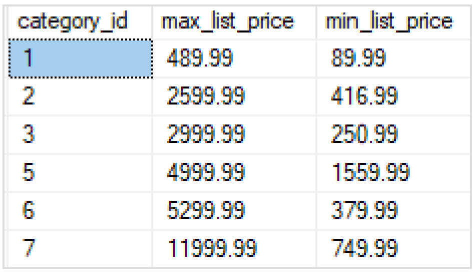
See the following `products` table:



production.products
* product_id
product_name
brand_id
category_id
model_year
list_price

The following statement first finds the maximum and minimum list prices in each product category. Then, it filters out the category which has the maximum list price greater than 4,000 or the minimum list price less than 500:

```
SELECT
    category_id,
    MAX (list_price) max_list_price,
    MIN (list_price) min_list_price
FROM
    production.products
GROUP BY
    category_id
HAVING
    MAX (list_price) > 4000 OR MIN (list_price) < 500;
```

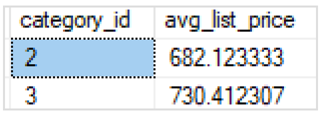


category_id	max_list_price	min_list_price
1	489.99	89.99
2	2599.99	416.99
3	2999.99	250.99
5	4999.99	1559.99
6	5299.99	379.99
7	11999.99	749.99

`HAVING` clause with `AVG()` function example

The following statement finds product categories whose average list prices are between 500 and 1,000:

```
SELECT
    category_id,
    AVG (list_price) avg_list_price
FROM
    production.products
GROUP BY
    category_id
HAVING
    AVG (list_price) BETWEEN 500 AND 1000;
```



category_id	avg_list_price
2	682.123333
3	730.412307

In this lab, you have learned how to use the SQL Server `HAVING` clause to filter groups based on specified conditions.