Module: 9 Grouping the data and Filtering the Groups

Grouping the Data Using GROUP BY clause

```
SELECT COUNT(*) AS num_prods
FROM Products
WHERE vend_id ='DLL01';

SELECT vend_id, COUNT(*) AS num_prods
FROM Products
GROUP BY vend_id;
```

important rules about use of GROUP BY Clause:

1. If an expression is used in the SELECT, that same expression must be specified in GROUP BY.
2. If there are multiple rows with NULL values, they'll all be grouped together.
3. GROUP BY clause must come after any WHERE clause and before any ORDER BY clause.

```
GROUP BY with ORDER BY clause.
SELECT vend_id, MAX(prod_price) AS avg_prod_price
FROM Products
GROUP BY vend_id
ORDER BY MAX(prod_price);

SELECT vend_id, COUNT(*) AS num_prods
FROM Products
GROUP BY vend_id
ORDER BY num_prods;

SELECT vend_id, COUNT(*) AS num_prods
FROM Products
GROUP BY vend_id
ORDER BY prod_id;
```

Filtering Groups Using HAVING clause :

Having Clause was added to SQL because WHERE Clause cann not be used with aggregate functions

WHERE clause : filters rows before grouping
GROUP BY : groups the data
HAVING clause : filters groups after aggregation are performed
ORDER BY : arrange the retrived rows selected columns of in order

Synatx

```
-----------------------------------------------------
SELECT col_name/s, AGG_FUNC(col_name)
FROM table_name
WHERE col_condition with operator value
GROUP BY col_name/s
HAVING AGG_FUNC(col_name) with operator value
ORDER BY col_name/s
```

```
SELECT cust_id, COUNT(*) AS orders
FROM Orders
GROUP BY cust_id
HAVING COUNT(*) >= 2;
```

To find vendors who have two or more products priced at 4 or more

```
SELECT vend_id, COUNT(*) AS num_prods
FROM Products
WHERE prod_price >= 4
GROUP BY vend_id
HAVING COUNT(*) >= 2;
```

```
Without the WHERE clause, an extra row would have been retrieved
SELECT vend_id, COUNT(*) AS num_prods
FROM Products
GROUP BY vend_id
HAVING COUNT(*) >= 2;
```

```
To find the order number and number of items ordered for all orders containing three
or more items:
SELECT order_num, COUNT(*) AS items
FROM OrderItems
GROUP BY order_num
HAVING COUNT(*) >= 3;
```

```
To sort the output by number of items ordered
SELECT order_num, COUNT(*) AS items
FROM OrderItems
GROUP BY order_num
HAVING COUNT(*) >= 3
ORDER BY items, order_num;
```

==========

```
--Practice Dataset
-------------------------------------------
create table order_details(
order_id int,
Customer varchar(20),
order_price int,
order_category varchar(20));

insert into order_details values (1,'Kate',2000,'grocery')
insert into order_details values (2,'Mark',3000,'Fruits')
insert into order_details values (3,'Tim',4000,'grocery')
insert into order_details values (4,'Paine',5000,'Cloths')
insert into order_details values (5,'Steve',6000,'Sports')
insert into order_details values (6,'Bill',7000,'grocery')
insert into order_details values (7,'Andy',8000,'Electronics')
insert into order_details values (8,'Grant',9000,'Sports')
insert into order_details values (9,'Pat',2500,'Cloths')
insert into order_details values (10,'Shon',3500,'Electronics')

select * from order_details;
```

```
--GROUP BY clause
-------------------------------------------
--Group by Statement is used in conjunction with aggregate functions to group by the
result set by one or more columns.

select distinct(order_category)
from order_details

select order_category, sum(order_price)
from order_details
group by order_category
--OR
select order_category, count(order_id), sum(order_price)
from order_details
group by order_category

select order_category, count(order_id) as order_id_Count, sum(order_price) as
order_price_Sum
from order_details
group by order_category


--Q.How to display the minimum price of each order category ?
select order_category, min(order_price) as MinOrder
from order_details
group by order_category --group by function deals separately with each group and
apply aggregate fuction and then returns the value for that group
```

```sql
select order_category, sum(order_price)
from order_details
where order_category ='Electronics' --Column 'order_details.order_category' is
invalid in the select list because it is not contained in either an aggregate
function or the GROUP BY clause.

select order_category,sum(order_price)
from order_details
where order_category ='Electronics'
group by order_category

select order_category, count(*) as 'count'
from order_details
group by order_category
```

HAVING Clause

```sql
-----------------------------------------------------------------
--The Having Clause was added to SQL because WHERE Clause is not used with aggregate
functions.

--Q.How to Display the order_category whose customer base is more than 1
select order_category, count(*)
from order_details
group by order_category
having count(*) > 2

--Q.How to display order_category whose having more than two customer and its min
price.
select order_category, min(order_price)
from order_details
group by order_category
having count(*)>2

--Q.how to display the order_category whose sum is grater than 10000.
select order_category,sum(order_price)
from order_details
group by order_category
having sum(order_price)> 10000

--Q.how to display sum of order_details of grocery category
select order_category, sum(order_price)
from order_details
where order_category = 'grocery'
group by order_category
```

Challenges:

---

1. Write a SQL statement that returns the number of lines (as order_lines) for each order number (order_num)
and sort the results by order_lines.

2. Write a SQL statement that returns a field named cheapest_item, which contains the lowest-cost item for each
vendor (using prod_price in the Products table), and sort the results from lowest to highest cost.

3. It's important to identify the best customers, so write a SQL statement to return the order number (order_num in
the OrderItems table) for all orders of at least 100 items.

4. Write a SQL statement to return the order number (order_num in the OrderItems table) for all orders with a total
price of at least 1000. Hint: for this one you'll need to calculate and sum the total (item_price multiplied by quantity).
Sort the results by order number.

5. What is wrong with the following SQL statement?
*/

```sql
SELECT order_num, COUNT(*) AS items
FROM OrderItems
GROUP BY items
HAVING COUNT(*) >= 3
ORDER BY items, order_num;
```