1. Introduction

- Understanding the importance of data aggregation and filtering.
- Real-world applications of these SQL clauses in business intelligence and reporting.

---

2. Sample Table Structure

Table: `sales`

Column Name	Data Type
sale_id	INT
customer_id	INT
product_id	INT
quantity	INT
price	DECIMAL(10,2)
sale_date	DATE
region	VARCHAR(50)
category	VARCHAR(50)
employee_id	INT
payment_method	VARCHAR(20)

---

3. COUNT() Function

Explanation:

- The `COUNT()` function returns the number of rows that match a specified condition.

Syntax:

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

Example:

```
SELECT COUNT(*)
FROM sales
WHERE region = 'North';
```

Result Set:

count
150

Let me know if you need a different example or more sample data!

Practice Questions:

1. Count the number of sales in the "sales" table.
2. Determine the number of transactions made via 'Credit Card'.
3. Count how many unique customers made purchases.
4. Find the number of distinct product categories sold.
5. Count the number of sales with a quantity greater than 10.
6. Count the number of orders placed by each customer.
7. Determine the number of sales transactions where the price exceeds 500.
8. Count the total number of transactions made in each region.
9. Count the number of transactions for each employee.
10. Find how many customers have placed more than 3 orders.

---

4. GROUP BY Clause

Explanation:

- The `GROUP BY` clause groups rows that have the same values in specified columns.
- It is typically used with aggregate functions like COUNT(), SUM(), AVG(), etc.

Syntax:

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
GROUP BY column_name;
```

Example:

```
SELECT customer_id, COUNT(sale_id) AS total_sales
FROM sales
GROUP BY customer_id;
```

Practice Questions:

1. Find the total number of sales made by each customer.
2. Group products by category and calculate the average price.
3. Count the number of sales per region.

4. Find the total quantity of products sold by each employee.
5. List the number of sales transactions made in each payment method.
6. Show total revenue generated in each region.
7. Determine the number of unique product categories sold in each region.
8. Get the count of sales per day.
9. Find the total quantity sold per product.
10. Display total sales per customer and filter those with more than 5 purchases.

---

5. ORDER BY Clause

Explanation:

- The `ORDER BY` clause is used to sort query results in ascending (`ASC`) or descending (`DESC`) order.

Syntax:

```
SELECT column_name
FROM table_name
ORDER BY column_name [ASC|DESC];
```

Example:

```
SELECT product_id, price
FROM sales
ORDER BY price DESC;
```

Practice Questions:

1. Retrieve product IDs sorted by price in ascending order.
2. Display sales sorted by sale date in descending order.
3. List customers sorted by their total number of purchases.
4. Sort employees by the number of transactions they handled.
5. Retrieve the top 5 highest-priced sales sorted by region.
6. Sort products by quantity sold in descending order.
7. Order sales based on payment method alphabetically.
8. Display regions ordered by total revenue generated.
9. List employees sorted by their average sale amount.
10. Sort customers based on their last purchase date.

---

6. LIMIT Clause

Explanation:

- The `LIMIT` clause restricts the number of records returned by a query.

Syntax:

```
SELECT column_name
FROM table_name
LIMIT number;
```

Example:

```
SELECT *
FROM sales
LIMIT 5;
```

Practice Questions:

1. Get the top 10 highest-priced sales.
2. Display the first 3 rows from the sales table.
3. Retrieve the latest 5 sales made.
4. Find the top 5 products based on total revenue.
5. Get the bottom 5 customers based on total spending.
6. List the 3 most recent transactions for a given customer.
7. Retrieve the top 2 employees with the highest sales volume.
8. Find the highest-priced products within each category, limiting to 1 per category.
9. Display the 5 most common payment methods used.
10. Show the top 5 regions with the highest number of sales.

---

7. HAVING Clause

Explanation:

- The `HAVING` clause is used to filter grouped records, typically used with aggregate functions.

Syntax:

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
GROUP BY column_name
HAVING condition;
```

Example:

```
SELECT category, COUNT(*) AS total_sales
FROM sales
```

```
GROUP BY category
HAVING COUNT(*) > 5;
```

Practice Questions:

1. Find product categories with more than 10 sales.
2. Show regions with an average sale price greater than 1000.
3. List employees who have handled more than 3 transactions.
4. Find products sold in more than 2 regions.
5. Display categories where total revenue is greater than 5000.
6. Show payment methods used more than 50 times.
7. Retrieve customers who have spent more than 2000 in total.
8. Find products with an average price higher than 100.
9. List regions with more than 5 unique products sold.
10. Find employees who processed more than 20 transactions.

---

8. Summary

- **GROUP BY:** Groups data by specific columns.
- **COUNT():** Counts the number of rows.
- **ORDER BY:** Sorts the result set.
- **LIMIT:** Restricts the number of rows returned.
- **HAVING:** Filters grouped results.

COUNT(*)

```
-- 1. Count the number of sales in the "sales" table.
SELECT COUNT(*) AS total_sales
FROM sales;

-- 2. Determine the number of transactions made via 'Credit Card'.
SELECT COUNT(*) AS credit_card_transactions
FROM sales
WHERE payment_method = 'Credit Card';

-- 3. Count how many unique customers made purchases.
SELECT COUNT(DISTINCT customer_id) AS unique_customers
FROM sales;

-- 4. Find the number of distinct product categories sold.
SELECT COUNT(DISTINCT category) AS distinct_categories
FROM sales;

-- 5. Count the number of sales with a quantity greater than 10.
SELECT COUNT(*) AS high_quantity_sales
FROM sales
WHERE quantity > 10;
```

```sql
-- 6. Count the number of orders placed by each customer.
SELECT customer_id, COUNT(sale_id) AS total_orders
FROM sales
GROUP BY customer_id;

-- 7. Determine the number of sales transactions where the price exceeds 500.
SELECT COUNT(*) AS high_value_sales
FROM sales
WHERE price > 500;

-- 8. Count the total number of transactions made in each region.
SELECT region, COUNT(sale_id) AS total_transactions
FROM sales
GROUP BY region;

-- 9. Count the number of transactions for each employee.
SELECT employee_id, COUNT(sale_id) AS transaction_count
FROM sales
GROUP BY employee_id;

-- 10. Find how many customers have placed more than 3 orders.
SELECT COUNT(customer_id) AS customers_with_more_than_3_orders
FROM (
    SELECT customer_id
    FROM sales
    GROUP BY customer_id
    HAVING COUNT(sale_id) > 3
) AS subquery;
```

GROUP BY

```sql
-- 1. Find the total number of sales made by each customer.
SELECT customer_id, COUNT(sale_id) AS total_sales
FROM sales
GROUP BY customer_id;

-- 2. Group products by category and calculate the average price.
SELECT category, AVG(price) AS avg_price
FROM sales
GROUP BY category;

-- 3. Count the number of sales per region.
SELECT region, COUNT(sale_id) AS sales_count
FROM sales
GROUP BY region;

-- 4. Find the total quantity of products sold by each employee.
SELECT employee_id, SUM(quantity) AS total_quantity_sold
FROM sales
```

```
GROUP BY employee_id;

-- 5. List the number of sales transactions made in each payment method.
SELECT payment_method, COUNT(sale_id) AS total_transactions
FROM sales
GROUP BY payment_method;

-- 6. Show total revenue generated in each region.
SELECT region, SUM(quantity * price) AS total_revenue
FROM sales
GROUP BY region;

-- 7. Determine the number of unique product categories sold in each region.
SELECT region, COUNT(DISTINCT category) AS unique_categories
FROM sales
GROUP BY region;

-- 8. Get the count of sales per day.
SELECT sale_date, COUNT(sale_id) AS sales_per_day
FROM sales
GROUP BY sale_date;

-- 9. Find the total quantity sold per product.
SELECT product_id, SUM(quantity) AS total_quantity_sold
FROM sales
GROUP BY product_id;

-- 10. Display total sales per customer and filter those with more than 5
purchases.
SELECT customer_id, COUNT(sale_id) AS total_sales
FROM sales
GROUP BY customer_id
HAVING COUNT(sale_id) > 5;
```

ORDER BY

```

-- 1. Retrieve product IDs sorted by price in ascending order.
SELECT product_id
FROM sales
ORDER BY price ASC;

-- 2. Display sales sorted by sale date in descending order.
SELECT *
FROM sales
ORDER BY sale_date DESC;

-- 3. List customers sorted by their total number of purchases.
SELECT customer_id, COUNT(sale_id) AS total_purchases
FROM sales
GROUP BY customer_id
```

```
ORDER BY total_purchases DESC;

-- 4. Sort employees by the number of transactions they handled.
SELECT employee_id, COUNT(sale_id) AS transaction_count
FROM sales
GROUP BY employee_id
ORDER BY transaction_count DESC;

-- 5. Retrieve the top 5 highest-priced sales sorted by region.
SELECT *
FROM sales
ORDER BY price DESC, region ASC
LIMIT 5;

-- 6. Sort products by quantity sold in descending order.
SELECT product_id, SUM(quantity) AS total_quantity_sold
FROM sales
GROUP BY product_id
ORDER BY total_quantity_sold DESC;

-- 7. Order sales based on payment method alphabetically.
SELECT *
FROM sales
ORDER BY payment_method ASC;

-- 8. Display regions ordered by total revenue generated.
SELECT region, SUM(quantity * price) AS total_revenue
FROM sales
GROUP BY region
ORDER BY total_revenue DESC;

-- 9. List employees sorted by their average sale amount.
SELECT employee_id, AVG(price) AS avg_sale_amount
FROM sales
GROUP BY employee_id
ORDER BY avg_sale_amount DESC;

-- 10. Sort customers based on their last purchase date.
SELECT customer_id, MAX(sale_date) AS last_purchase_date
FROM sales
GROUP BY customer_id
ORDER BY last_purchase_date DESC;
```

LIMIT

```
-- 1. Get the top 10 highest-priced sales.
SELECT *
FROM sales
ORDER BY price DESC
LIMIT 10;
```

```sql
-- 2. Display the first 3 rows from the sales table.
SELECT *
FROM sales
LIMIT 3;

-- 3. Retrieve the latest 5 sales made.
SELECT *
FROM sales
ORDER BY sale_date DESC
LIMIT 5;

-- 4. Find the top 5 products based on total revenue.
SELECT product_id, SUM(quantity * price) AS total_revenue
FROM sales
GROUP BY product_id
ORDER BY total_revenue DESC
LIMIT 5;

-- 5. Get the bottom 5 customers based on total spending.
SELECT customer_id, SUM(quantity * price) AS total_spending
FROM sales
GROUP BY customer_id
ORDER BY total_spending ASC
LIMIT 5;

-- 6. List the 3 most recent transactions for a given customer.
SELECT *
FROM sales
WHERE customer_id = <customer_id_placeholder>
ORDER BY sale_date DESC
LIMIT 3;

-- 7. Retrieve the top 2 employees with the highest sales volume.
SELECT employee_id, SUM(quantity) AS total_sales_volume
FROM sales
GROUP BY employee_id
ORDER BY total_sales_volume DESC
LIMIT 2;

-- 8. Find the highest-priced products within each category, limiting to 1 per
category.
SELECT DISTINCT ON (category) category, product_id, price
FROM sales
ORDER BY category, price DESC;

-- 9. Display the 5 most common payment methods used.
SELECT payment_method, COUNT(*) AS usage_count
FROM sales
GROUP BY payment_method
ORDER BY usage_count DESC
LIMIT 5;

-- 10. Show the top 5 regions with the highest number of sales.
```

```sql
SELECT region, COUNT(sale_id) AS total_sales
FROM sales
GROUP BY region
ORDER BY total_sales DESC
LIMIT 5;
```

HAVING

```sql
-- 1. Find product categories with more than 10 sales.
SELECT category, COUNT(*) AS total_sales
FROM sales
GROUP BY category
HAVING COUNT(*) > 10;

-- 2. Show regions with an average sale price greater than 1000.
SELECT region, AVG(price) AS avg_sale_price
FROM sales
GROUP BY region
HAVING AVG(price) > 1000;

-- 3. List employees who have handled more than 3 transactions.
SELECT employee_id, COUNT(*) AS total_transactions
FROM sales
GROUP BY employee_id
HAVING COUNT(*) > 3;

-- 4. Find products sold in more than 2 regions.
SELECT product_id, COUNT(DISTINCT region) AS region_count
FROM sales
GROUP BY product_id
HAVING COUNT(DISTINCT region) > 2;

-- 5. Display categories where total revenue is greater than 5000.
SELECT category, SUM(quantity * price) AS total_revenue
FROM sales
GROUP BY category
HAVING SUM(quantity * price) > 5000;

-- 6. Show payment methods used more than 50 times.
SELECT payment_method, COUNT(*) AS usage_count
FROM sales
GROUP BY payment_method
HAVING COUNT(*) > 50;

-- 7. Retrieve customers who have spent more than 2000 in total.
SELECT customer_id, SUM(quantity * price) AS total_spending
FROM sales
GROUP BY customer_id
HAVING SUM(quantity * price) > 2000;
```

```sql
-- 8. Find products with an average price higher than 100.
SELECT product_id, AVG(price) AS avg_product_price
FROM sales
GROUP BY product_id
HAVING AVG(price) > 100;

-- 9. List regions with more than 5 unique products sold.
SELECT region, COUNT(DISTINCT product_id) AS unique_products
FROM sales
GROUP BY region
HAVING COUNT(DISTINCT product_id) > 5;

-- 10. Find employees who processed more than 20 transactions.
SELECT employee_id, COUNT(*) AS transaction_count
FROM sales
GROUP BY employee_id
HAVING COUNT(*) > 20;
```