Student Name: Sarthak Sahu
Topic: DISTINCT, LIMIT, ORDER BY, Query Optimization

---

1. Practical Use of DISTINCT in SQL

Scenario

You are working with an e-commerce platform's orders table:

order_id	customer_id	product_name	order_date
101	200	Laptop	2025-01-15
102	201	Phone	2025-01-16
103	202	Laptop	2025-01-17
104	200	Tablet	2025-01-18
105	203	Phone	2025-01-19
106	204	Laptop	2025-01-20

Q1. Unique Products

```
SELECT DISTINCT product_name
FROM orders;
```

✅ **Explanation:**
This query removes duplicate products and returns only unique product names.

Q2. Unique Products Ordered by Each Customer

```
SELECT customer_id,
    COUNT(DISTINCT product_name) AS unique_products
FROM orders
GROUP BY customer_id;
```

✅ **Explanation:**
Counts how many different products each customer has ordered.

Q3. Distinct Product Count

```
SELECT COUNT(DISTINCT product_name) AS total_unique_products
FROM orders;
```

✅ **Explanation:**
Returns one number showing total distinct products ordered.

Q4. Most Recent Distinct Products (Top 3)

SELECT DISTINCT product_name

FROM orders

ORDER BY order_date DESC

LIMIT 3;

✅ **Explanation:**
Shows the 3 most recently ordered unique products.

---

2. Combining DISTINCT, LIMIT, and ORDER BY

Updated table includes more rows (page 3):

order_id	customer_id	product_name	order_date
107	205	Phone	2025-02-01
108	206	Tablet	2025-02-02

Q1. Top 2 Most Ordered Distinct Products (Last Month)

SELECT product_name,

COUNT(*) AS total_orders

FROM orders

WHERE order_date >= CURDATE() - INTERVAL 1 MONTH

GROUP BY product_name

ORDER BY total_orders DESC

LIMIT 2;

✅ **Explanation:**
Finds the most popular products based on order count.

Q2. Unique Products for Customer 200 (Top 3 Recent)

SELECT DISTINCT product_name

```
FROM orders
WHERE customer_id = 200
ORDER BY order_date DESC
LIMIT 3;
```

Q3. Top 5 Most Ordered Products (Sorted A–Z)

```
SELECT product_name,
    COUNT(DISTINCT order_id) AS total_orders
FROM orders
GROUP BY product_name
ORDER BY product_name ASC
LIMIT 5;
```

Q4. Count Unique Orders for Each Product

```
SELECT product_name,
    COUNT(DISTINCT order_id) AS unique_orders
FROM orders
GROUP BY product_name
ORDER BY unique_orders DESC;
```

---

3. Optimizing Queries with DISTINCT and Indexing

Q1. Retrieve Distinct Products Ordered in Last Month + Index Strategy

```
SELECT DISTINCT product_name
FROM orders
WHERE order_date >= CURDATE() - INTERVAL 1 MONTH;
```

✅ **Optimization Strategy:**
Create indexes:

```
CREATE INDEX idx_order_date ON orders(order_date);
CREATE INDEX idx_product_name ON orders(product_name);
```

✓ Indexing order_date speeds up filtering
✓ Indexing product_name speeds up DISTINCT search

Q2. Why DISTINCT is Expensive on Large Data

DISTINCT requires:

- Sorting large datasets
- Removing duplicates
- Extra memory + temporary tables

So on millions of rows, it becomes slower.

Q3. Top 3 Distinct Products Ordered by Customer 200

```
SELECT DISTINCT product_name
FROM orders
WHERE customer_id = 200
ORDER BY order_date DESC
LIMIT 3;
```

✅ Optimized because:

- Filters customer first
- Uses ORDER BY on indexed column
- LIMIT reduces output size

---

4. Query Optimization and Execution Plan Analysis

Q1. Execution Plan Analysis (Top 10 Popular Products)

```
EXPLAIN
SELECT DISTINCT product_name
FROM orders
WHERE order_date >= CURDATE() - INTERVAL 30 DAY
ORDER BY order_date DESC
LIMIT 10;
```

✅ EXPLAIN helps identify:

- Full table scans
- Missing indexes
- Sorting overhead

Q2. Best Index for ORDER BY + DISTINCT + LIMIT

Recommended index:

```
CREATE INDEX idx_date_product ON orders(order_date, product_name);
```

✓ Helps filtering + sorting + uniqueness

Q3. Alternative Optimized Query (Top 5 Ordered Products)

Better than raw DISTINCT:

```
SELECT product_name,
    COUNT(*) AS total_orders
FROM orders
WHERE order_date >= CURDATE() - INTERVAL 30 DAY
GROUP BY product_name
ORDER BY total_orders DESC
LIMIT 5;
```

✅ Faster because aggregation avoids unnecessary DISTINCT sorting.

---

5. Real-World Scenario and Complex Query Creation

Q1. Top 10 Most Recent Distinct Products

```
SELECT DISTINCT product_name
FROM orders
ORDER BY order_date DESC
LIMIT 10;
```

Q2. Optimization for Millions of Rows

Use indexes:

```
CREATE INDEX idx_order_date ON orders(order_date);

CREATE INDEX idx_product_customer ON orders(product_name, customer_id);
```

Also:

- Avoid SELECT *
- Use LIMIT early
- Partition large tables by date

Q3. Show Customer Who Ordered Each Product Most Recently

```
SELECT o.product_name,
    o.customer_id,
    o.order_date
FROM orders o
INNER JOIN (
  SELECT product_name,
      MAX(order_date) AS last_order
  FROM orders
  GROUP BY product_name
) latest
ON o.product_name = latest.product_name
AND o.order_date = latest.last_order
ORDER BY o.order_date DESC;
```

✅ This returns:

- Product name
- Customer who purchased it last
- Most recent order date