Detecting, handling, and preventing duplicate records in SQL, with step-by-step explanations and SQL code.

✅ 1. Find Duplicates

Use GROUP BY with HAVING COUNT(*) > 1.

🔹 Example: Find duplicate customers by name and email:

```
SELECT name, email, COUNT(*) AS cnt
FROM customers
GROUP BY name, email
HAVING COUNT(*) > 1;
```

✅ 2. Get Unique Records

Use SELECT DISTINCT to retrieve only unique rows.

```
SELECT DISTINCT name, email
FROM customers;
```

✅ 3. Remove Duplicates Using CTE + ROW_NUMBER()

You can keep the first occurrence and delete the rest:

```
WITH cte AS (
  SELECT *,
      ROW_NUMBER() OVER (PARTITION BY name, email ORDER BY id) AS rn
  FROM customers
)
DELETE FROM customers
WHERE id IN (
  SELECT id FROM cte WHERE rn > 1
);
```

🔸 Explanation:

- PARTITION BY name, email groups duplicates.

- ROW_NUMBER() ranks them.
- Keep rn = 1, delete rn > 1.

✅ 4. Prevent Duplicates Using Constraints

- Add a PRIMARY KEY or UNIQUE constraint to avoid duplicates in future inserts.

```
-- On single column

ALTER TABLE customers

ADD CONSTRAINT unique_email UNIQUE (email);


-- On multiple columns

ALTER TABLE customers

ADD CONSTRAINT unique_name_email UNIQUE (name, email);
```

🎯 Summary:

Step	Goal	SQL Used
1	Find duplicates	GROUP BY + HAVING COUNT()
2	Show unique records	SELECT DISTINCT
3	Delete duplicates	ROW_NUMBER() in CTE
4	Prevent future duplicates	PRIMARY KEY / UNIQUE

✅ Key Difference:

Query Type	What Happens
Normal SQL Query	Executes from scratch every time you run it.
Materialized View	Executes once, stores the result in disk.

🔁 Without Materialized View (Direct Query)

SELECT product_id, SUM(total_amount)

FROM sales

GROUP BY product_id

ORDER BY SUM(total_amount) DESC

LIMIT 10;

Each time this runs:

- It scans the sales table.
- Aggregates the entire dataset.
- Sorts it.
- Returns top 10.
  ➡️ This is repeated for every user, every dashboard refresh.

⛔ Problem: Slow on large tables, resource-intensive.

🗃️ With Materialized View

CREATE MATERIALIZED VIEW top_products AS

SELECT product_id, SUM(total_amount)

FROM sales

GROUP BY product_id

ORDER BY SUM(total_amount) DESC

LIMIT 10;

- It runs the query once when you create or refresh it.
- Stores the result set (like a snapshot) in disk.
- When users query the materialized view:

SELECT * FROM top_products;

➡️ It just reads the stored data.
✅ Much faster, no re-computation.

🧠 Analogy

Think of it like this:

- Normal query = asking the chef to cook from scratch every time.

- Materialized view = storing the dish in the fridge and serving it cold (or refreshed when needed).

🧰 When to Use MVs?

- Query is expensive (joins, aggregations, sorts).
- Data doesn't change often.
- You need fast read performance.

How to Analyze User Behavior to Improve Business Decisions

E-commerce / Retail

Approach:

1/ **Key Metrics** – Track cart abandonment, repeat purchases, average order value
2/ **Cohort Analysis** – Study customer purchase behavior over weeks/months
3/ **Funnel Analysis** – Identify where users drop off in the checkout process
4/ **User Segmentation** – Group customers by purchase frequency, location, or demographics
5/ **A/B Testing** – Test changes in UI, offers, or recommendations to boost conversions

Finance / Banking

Approach:

1/ **Key Metrics** – Monitor transaction volume, loan approval rates, fraud incidents
2/ **Cohort Analysis** – Analyze customer retention by account opening date
3/ **Funnel Analysis** – Examine drop-offs in loan application or onboarding flows
4/ **User Segmentation** – Segment customers by credit score, account type, or behavior
5/ **A/B Testing** – Test different loan offers or communication strategies for effectiveness

Healthcare

Approach:

1/ **Key Metrics** – Track appointment attendance, medication adherence, patient portal usage
2/ **Cohort Analysis** – Follow patient groups by treatment start date
3/ **Funnel Analysis** – Identify where patients drop off in care pathways or appointment bookings
4/ **User Segmentation** – Segment by age group, condition, or engagement level
5/ **A/B Testing** – Experiment with reminder methods or educational content to improve outcomes

SaaS / Software

Approach:

1/ **Key Metrics** – Measure feature adoption, user retention, active users
2/ **Cohort Analysis** – Analyze retention by signup date or subscription tier
3/ **Funnel Analysis** – Detect drop-offs in onboarding or upgrade flows
4/ **User Segmentation** – Group users by usage frequency, plan type, or geography
5/ **A/B Testing** – Test UI changes or new features to increase engagement or reduce churn

Media / Streaming

Approach:

1/ **Key Metrics** – Track watch time, subscription renewals, content completion rates
2/ **Cohort Analysis** – Study subscriber behavior by join date or plan
3/ **Funnel Analysis** – Analyze drop-offs in sign-up or subscription purchase
4/ **User Segmentation** – Segment by content preferences, demographics, or device type
5/ **A/B Testing** – Experiment with recommendations, pricing, or UI layouts to improve retention