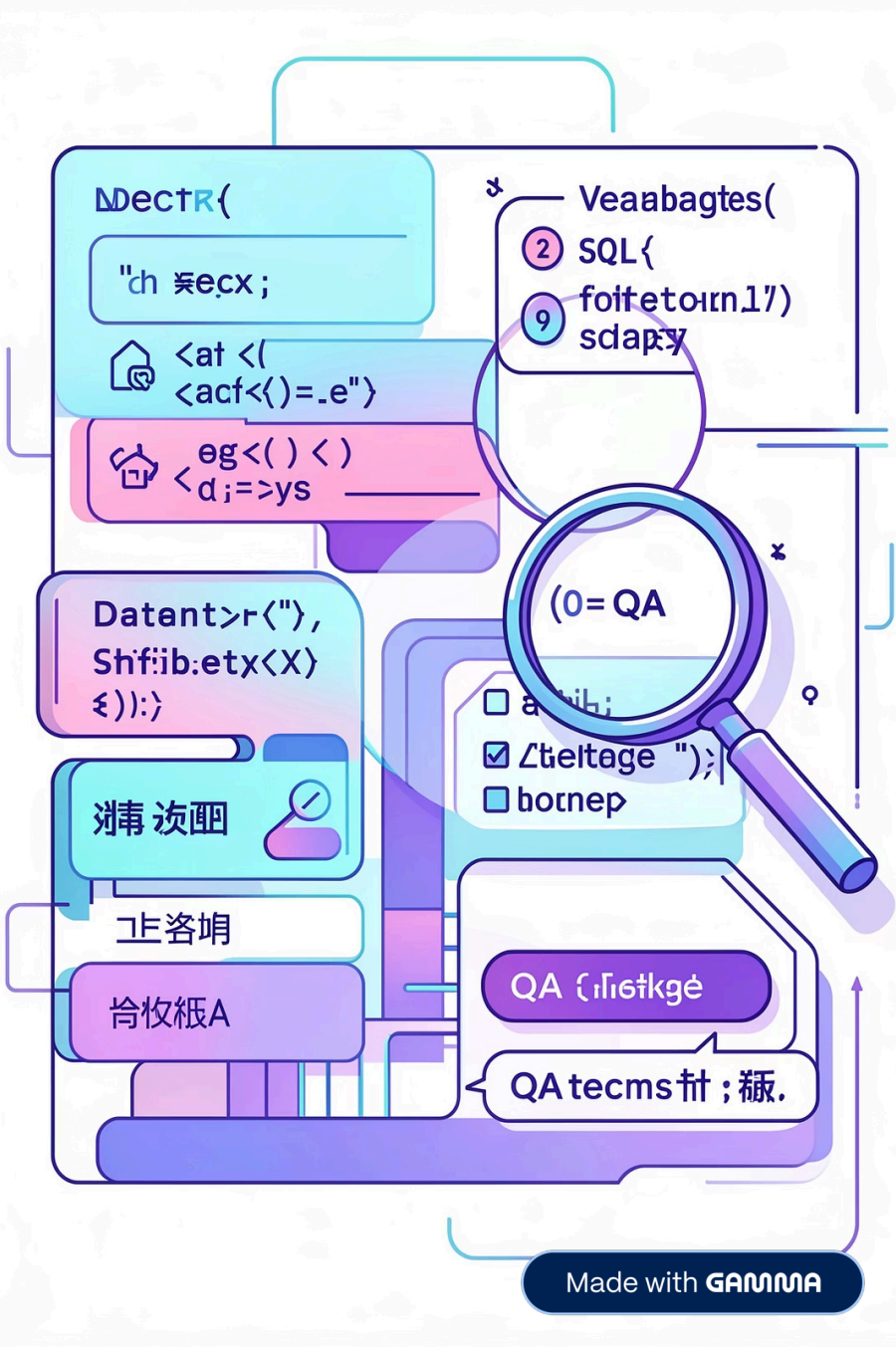
QA Training - Day 7

BIT 5th Sem (Professional Class)

Facilitator: **Neha Rouniyar**

SQL for QA: Look Deeper

You test more than just screens. SQL helps you see hidden data. This checks if data is correct and good. It's key for complete testing.



Why QA Needs SQL

Check Data

See if all your data is correct.

Check Key Info

Confirm user and order details are correct.

Find Bugs Faster

Find mistakes fast. Compare with what's stored.

Easy to Learn

It's simple to learn. No hard coding needed.

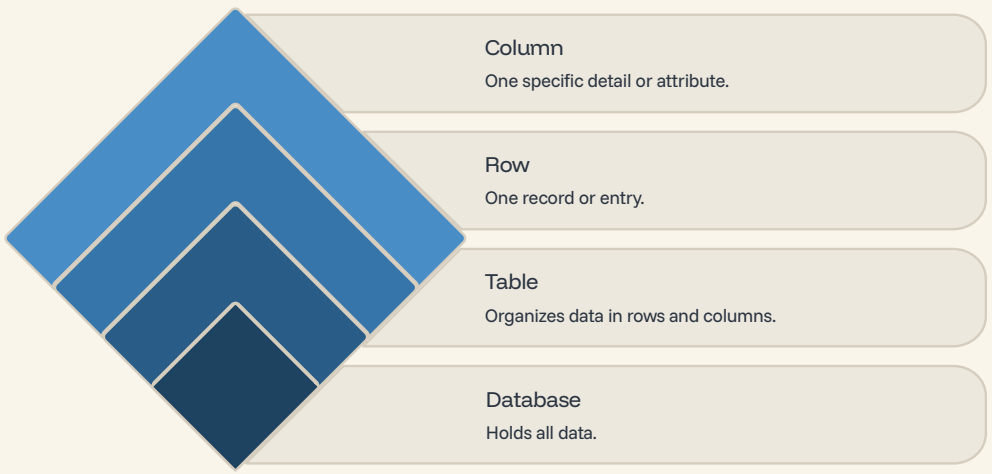
How QA uses SQL daily

QA uses SQL daily. It checks data. It makes apps work.

- See if screen data matches stored data.
- Check if saved data is right after action.
- Find missing or wrong data.
- Fix problems faster. See why they happened.

Example: screen says "Order Complete." SQL quickly checks the order. It confirms all details are right.

Learn About Databases



Here's an example

A 'users' list keeps info. It has details like 'id', 'email', 'password'. Each line is one person.

Detailed Breakdown of Database Elements:

Understanding these core concepts is crucial for navigating and querying any relational database effectively.

1. Database Level

Think of a database as a big filing cabinet. It holds everything. All your data lives here. One database can have many tables.

2. Table Level

A table is like a folder inside the cabinet. It organizes data in rows and columns. Each table stores one type of information. For example, one table for users, another for orders.

3. Row Level

A row is one complete record. It's like one sheet of paper in a folder. Each row represents one item or person. For example, one row = one user with all their details.

4. Column Level

A column is one specific piece of information. It's like one detail on a paper. All rows have the same columns. For example, 'email' column has all user emails.

Real Example: In a 'users' table, each row is one person. Columns are: id, name, email, password. So one row might be: (1, 'John', 'john@email.com', 'pass123')

SQL Command Flow

SELECT

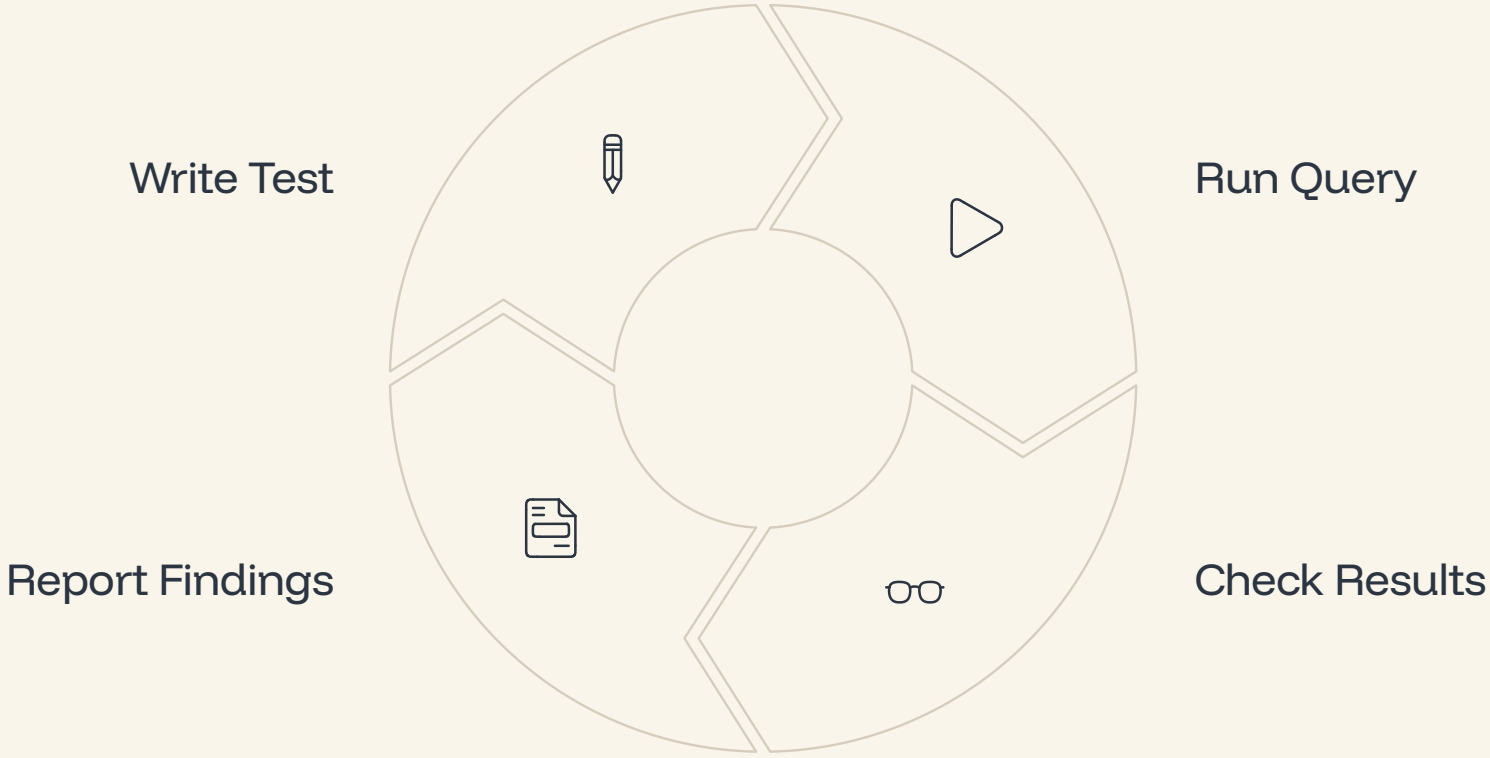
WHERE

GROUP BY

JOIN

SQL commands work together like building blocks. You start with SELECT to get data. Then use WHERE to filter it. Use JOIN to connect tables. Finally, use GROUP BY to organize results. Each step makes your query more powerful.

QA Testing Workflow



QA testing with SQL follows a cycle. First, you write a test query to check something specific. Then you run it against the database. Next, you check the results carefully. Finally, you report what you found. This cycle repeats for every test you do. Each cycle makes your testing more thorough.

How to Get Data: Use SELECT

Use `SELECT` to get data. See all data, or just some.

How to use it:

```
SELECT * FROM table_name;
```

This gets everything (`*`) from your table.

When to use it:

- Did a new user join? Check.
- Is new data saved right? Check.

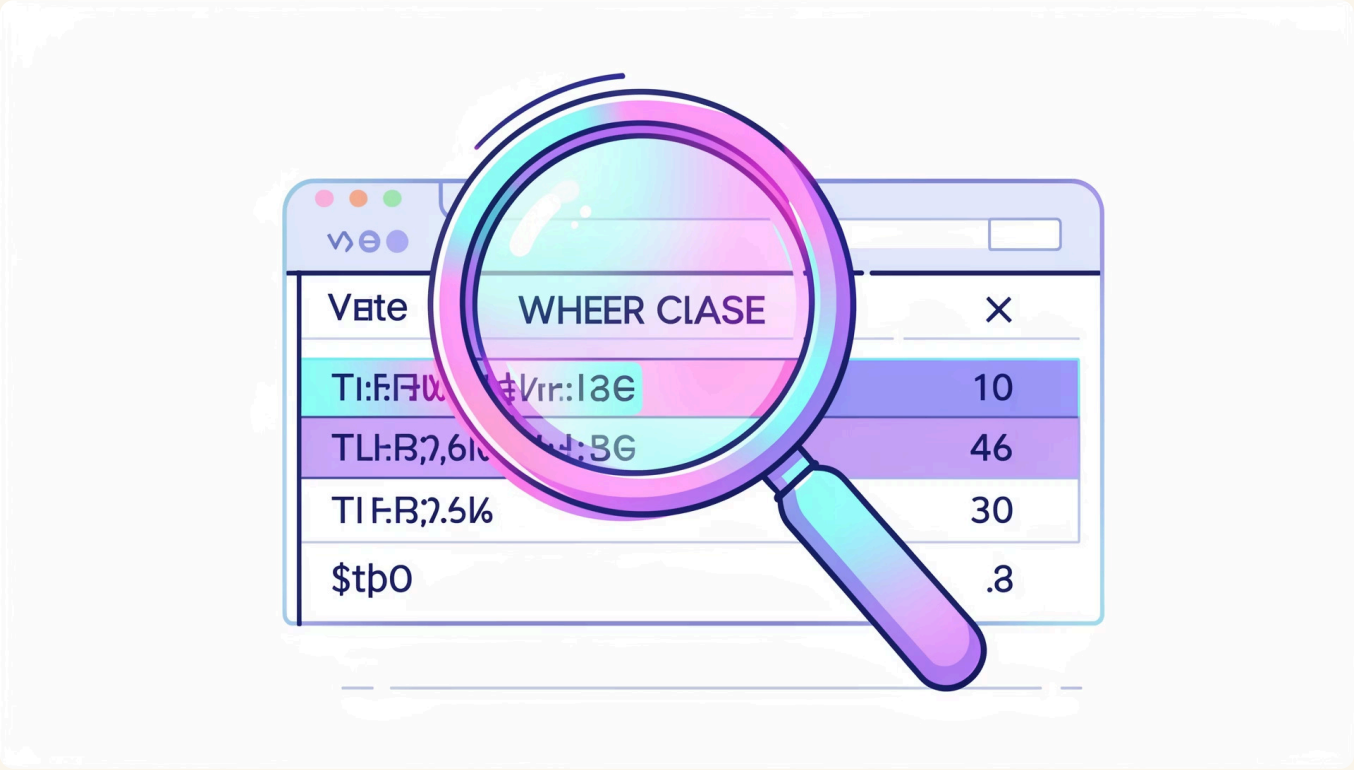
Find exact data with WHERE.

Need specific info? Use `WHERE` with `SELECT`. It helps you check things.

How it works:

```
SELECT * FROM users
WHERE email =
'test@example.com';
```

It finds all user details for that email.

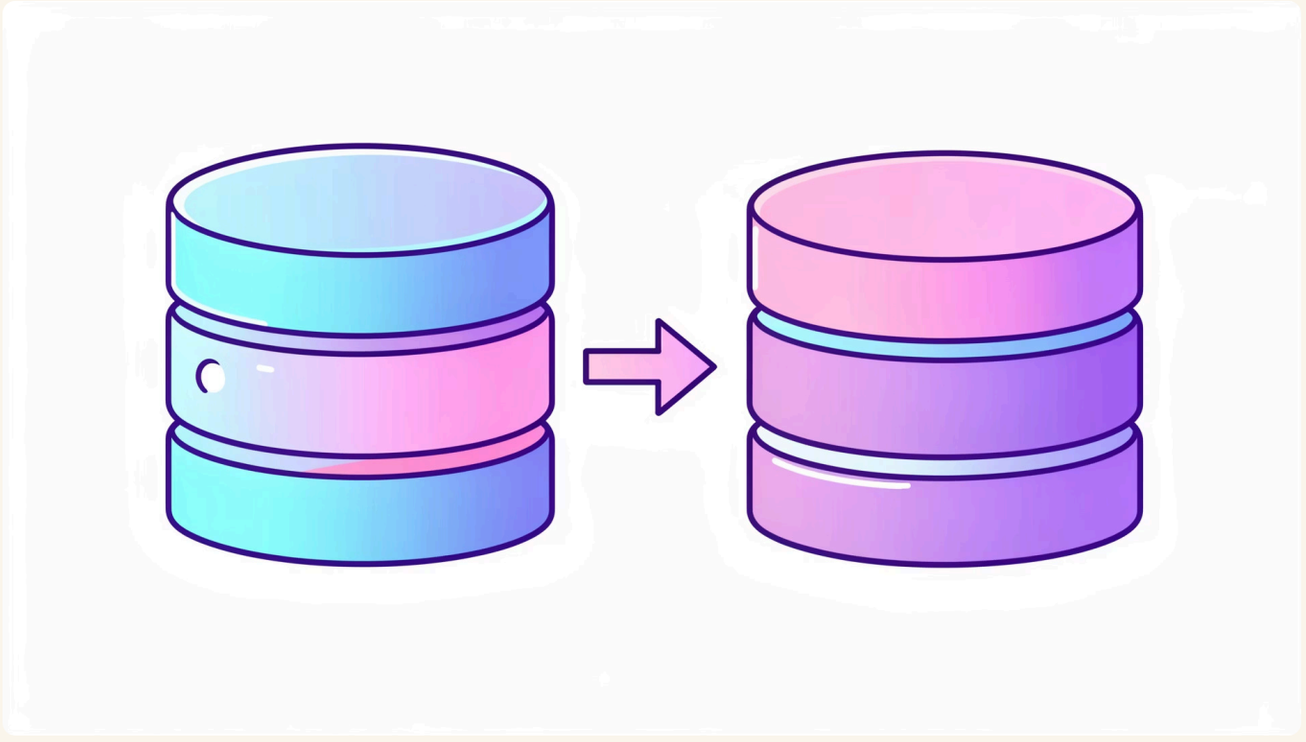


Why it helps testers:

- Check user info after changes.
- See why some logins fail.
- Grab data for your tests.

JOIN: Connect Your Info

Data often lives in separate lists. The `JOIN` command connects these lists. It uses common details.



How to write it:

```
SELECT u.email, o.order_id FROM users u JOIN orders o ON u.id = o.user_id;
```

This code pairs emails with orders. It uses the user ID to connect them.

Think of this:

One list has people. Another list has orders. `JOIN` links people to their orders.

Why it's useful:

- Check if people are linked to what they do. Like buying things.
- Ensure all data pieces connect correctly.

Easy Data Checks

Testers use SQL. It checks data fast. This helps make sure everything works.

See Bad Logins

```
SELECT * FROM
login_attempts WHERE status
= 'FAILED';
```

This finds security issues. It also finds login problems.

Count Things by Status

```
SELECT COUNT(*) FROM
orders WHERE status =
'Completed';
```

This counts things by their status. It's important for the business.

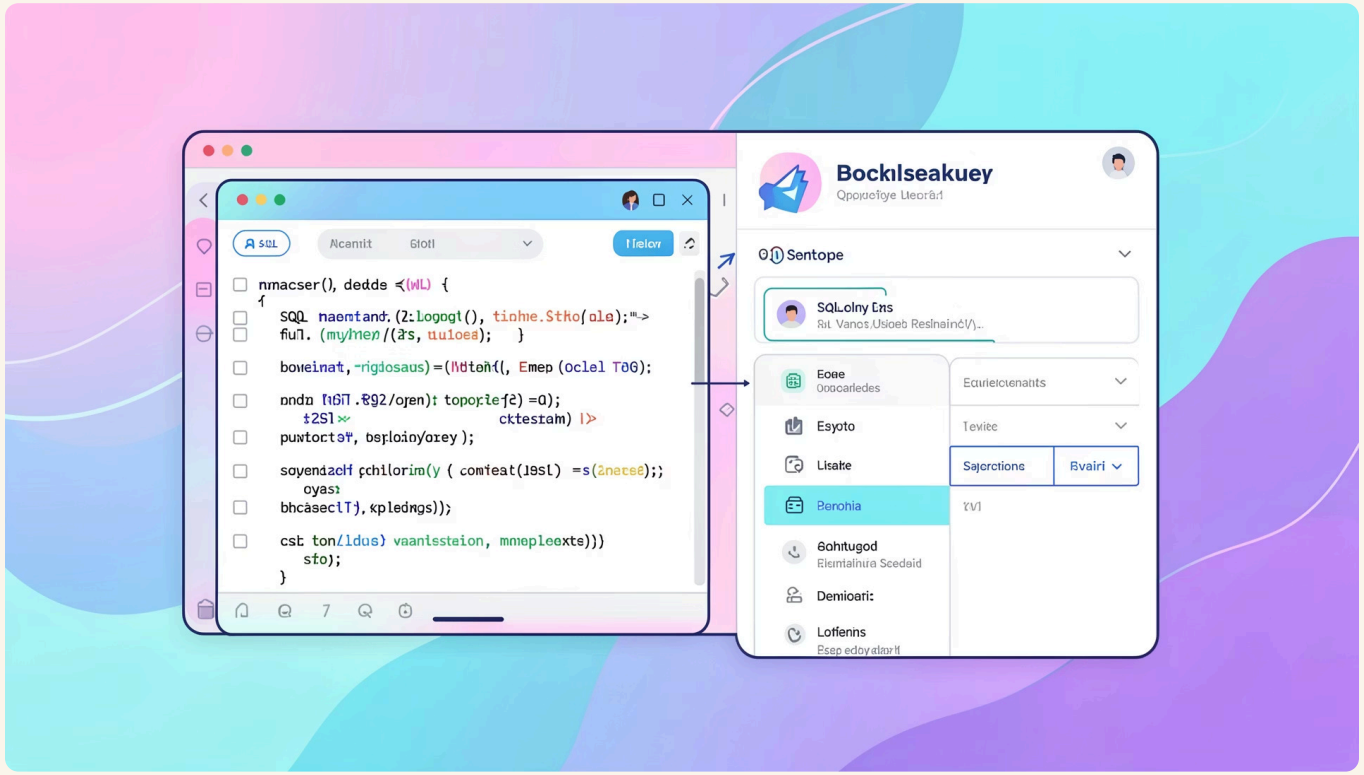
Find Repeated Items

```
SELECT email, COUNT(*)
FROM users GROUP BY email
HAVING COUNT(*) > 1;
```

This stops bad data. It makes sure each entry is new.

Tell What You Saw: Simple QA Reports

Running checks is just step one. You must share your findings clearly. Keep reports simple, short, and useful.



- **What you asked:** Share your exact code. Others can then check it.
- **Show the answer:** Add a picture of the results. Or just copy the text.
- **What you saw:** Explain clearly what was wrong.
- **Where to find the problem:** Link to the bug.

> ## Example Report:
>
> What you saw: "The order did not save. But the screen said it did."

Next Steps

Try out SQL with these tasks.

Check if people joined okay.

See if logins (good, bad) are saved.

Count things by status (like 'waiting').

Match app info with saved info.

Good tools to use:

Use tools like SQLite (for your computer).

Or DBeaver (for many types of data).

Helpful Tip: As QA, use SQL to `check info`.

Don't build big things with it.

Questions & Discussions ??