ScopeTreasure Courses

SQL COURSES · LESSON DB.11 · CODE SCOPE SERIES

Views + Data Ko Mehfooz Rakhna (Users & Permissions)

Saved questions and safe doors: CREATE VIEW, why a view is live and not a photo, and GRANT — giving each user only the access they need.

Introduction

Two problems grow with every real database. First: the principal asks for the **top students list** every single morning, and someone re-types the same long query every single day. Second: the school's database has **one password shared by everyone** — the clerk who prints reports can also delete the fees table, because nobody ever divided the doors.

This lesson solves both. A **VIEW** saves a query under a name, so it can be used like a table. And **users & permissions** (GRANT / REVOKE) give each person exactly the doors his job needs — and no others.

(Roman Urdu: VIEW = mehfooz sawal, jo table ki tarah chalta hai. Permission = har banday ko sirf us ke kaam ke darwaze.)

Key points

- **A VIEW is a saved SELECT with a name:** `CREATE VIEW top_students AS SELECT ...;` — from then on, `SELECT * FROM top_students;` works exactly like querying a table.
- **A view stores the question, not the answer:** it holds no data of its own. Every time you query it, the saved SELECT runs *now*, on the *current* data. It is a window, not a photograph — update the table and the view shows the new world instantly (Example 2 proves it).
  (Roman Urdu: VIEW photo nahi, khirki hai — andar ka manzar badla to khirki se bhi badla dikhega.)
- **Views hide what should not travel:** a view can expose names and marks while leaving phone numbers and addresses inside the base table. Paired with permissions, it is how a report-user sees *enough* and nothing more.
- **Every person gets his own account:** `CREATE USER 'teacher'@'localhost' IDENTIFIED BY '...';` — shared passwords mean shared blame; individual accounts mean every action has a name.
- **GRANT gives, REVOKE takes:** `GRANT SELECT ON school.students TO 'teacher'@'localhost';` — name the power (SELECT, INSERT, UPDATE, DELETE), the table, and the user. Start from zero and grant the minimum the job needs: a reporting account usually needs SELECT and nothing else.
- **The principle is least privilege:** the marks-entry clerk needs INSERT/UPDATE on results, not DELETE on fees. Most data disasters are not hackers — they are over-powered accounts used carelessly.

Code examples

Example 1 — CREATE VIEW: the morning list, saved once

```
CREATE VIEW top_students AS
SELECT id, name, marks FROM students
WHERE marks >= 85;

SELECT * FROM top_students;
```

What it does: The view saves the "top students" question — id, name and marks of everyone at 85 or above. Querying it feels exactly like querying a table.

What you will see:

```
+----+-----------+-------+
| id | name      | marks |
+----+-----------+-------+
|  1 | Ali Raza  |    88 |
|  2 | Sara Khan |    92 |
|  4 | Hira Shah |    95 |
+----+-----------+-------+
3 rows in set
```

Three students clear 85. The view chose its own columns too — no city, no class — just the facts the morning list needs. Tomorrow, the principal's list is one short line: `SELECT * FROM top_students;`

Example 2 — The view is alive: change the table, the view follows

```
UPDATE students SET marks = 96 WHERE id = 4;

SELECT * FROM top_students;
```

What it does: Hira's marks are corrected from 95 to 96 in the base table (a normal UPDATE from DB.3). Nobody touches the view — nobody needs to. Querying it again runs its saved SELECT against today's data.

What you will see:

```
+----+-----------+-------+
| id | name      | marks |
+----+-----------+-------+
|  1 | Ali Raza  |    88 |
|  2 | Sara Khan |    92 |
|  4 | Hira Shah |    96 |
+----+-----------+-------+
3 rows in set
```

Hira now shows 96 *inside the view*, because the view never held a copy — it holds the question. Had this been an exported spreadsheet (a true photograph), it would still say 95 and nobody would know it had gone stale.

Example 3 — A reporting user who can look, and only look

```
CREATE USER 'teacher'@'localhost' IDENTIFIED BY 'Class@123';

GRANT SELECT ON school.students TO 'teacher'@'localhost';

SHOW GRANTS FOR 'teacher'@'localhost';
```

What it does: A personal account for the teacher, granted exactly one power: SELECT on the `students` table of the `school` database. SHOW GRANTS prints the receipt. (The database here is named `school` — in earlier lessons we worked inside it without naming it.)

What you will see:

```
+-----------------------------------------------------------+
| Grants for teacher@localhost                              |
+-----------------------------------------------------------+
| GRANT USAGE ON *.* TO 'teacher'@'localhost                |
| GRANT SELECT ON school.students TO 'teacher'@'localhost  |
+-----------------------------------------------------------+
```

`USAGE` means "may log in, nothing more". Now the test that matters: logged in as teacher, `DELETE FROM students WHERE id = 1;` answers `ERROR 1142: DELETE command denied to user 'teacher'@'localhost'`. The door is not locked by politeness — it does not exist for him. If he later needs marks entry, one precise line adds it: `GRANT UPDATE (marks) ON school.students TO 'teacher'@'localhost';` — even the *column* can be the limit.

Quick recap

- **VIEW** = a SELECT saved under a name; queried like a table; stores the **question**, so it is always current.
- Views simplify repeated questions and hide columns that should not travel.

- **CREATE USER** gives every person his own named account; **GRANT** gives minimum powers; **REVOKE** takes them back.
- Least privilege: an account should hold exactly the doors its job opens — a viewer who can delete is an accident waiting for a date.

Common mistakes

- **Treating a view like a backup:** a view holds no data. Delete the base table and the "view's data" is gone with it — it was never there.
- **Exporting a spreadsheet instead of making a view:** the export is stale the moment it is saved (Example 2's photograph). If the same question is asked twice, it deserves a view.

  (Roman Urdu: Roz wahi report chahiye to VIEW banao — photo purani ho jati hai, khirki nahi.)
- **One shared admin account for the whole office:** when something breaks, there is no "who", and everyone had DELETE whether their job needed it or not.
- **GRANT ALL "to save time":** `GRANT ALL PRIVILEGES` for a report viewer is the single most common permission mistake. Grant the one power, test the denial (like Example 3's DELETE), then stop.

Practice

1. Create a view `sahiwal_students` showing name and marks of students whose city is Sahiwal. How many rows should it show today (check DB.1's table — note Hira's marks are now 96 after Example 2)?
2. A new student joins with 90 marks. Without touching the view, write down what `SELECT * FROM top_students;` shows now — which property of views makes your answer certain?
3. Design the permissions for a fee clerk: he must add fee rows and read the students table, but never change marks or delete anything. Write the CREATE USER and the exact GRANT lines.

Practice quiz

1. What does a VIEW store?

- A copy of the data, frozen at creation
- The question, not the answer: every time you query it, the saved SELECT runs now on the current data, so it is always current
- A backup of the table
- Only the table's structure

Answer: The question, not the answer: every time you query it, the saved SELECT runs now on the current data, so it is always current

2. Which statement gives a teacher read-only power on students?

- GRANT ALL PRIVILEGES ON school.students TO 'teacher'@'localhost';

- GRANT SELECT ON school.students TO 'teacher'@'localhost';, starting from zero and granting the minimum the job needs
- REVOKE SELECT ON school.students FROM 'teacher'@'localhost';
- CREATE USER 'teacher'@'localhost'; alone

Answer: GRANT SELECT ON school.students TO 'teacher'@'localhost';, starting from zero and granting the minimum the job needs

3. What is the principle of least privilege?

- Give every account every power, to save time
- An account holds exactly the doors its job opens; a viewer who can delete is an accident waiting for a date
- Give nobody any power
- Only admins get accounts

Answer: An account holds exactly the doors its job opens; a viewer who can delete is an accident waiting for a date

---

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/sql-views-permissions · More free tools and notes at https://scopetreasure.com