BY LEARNSTACK

2026 Edition

SQL Complete Handbook

From Beginner to Advanced - Queries, Design, Optimization & Interviews

SELECT & Joins | Subqueries | Window Functions | Database Design | Indexing | Stored Procedures | Interview Prep

Learn. Build. Grow.

linktr.ee/LearnStack

About This Handbook

This handbook teaches SQL from absolute basics to advanced professional use. It covers relational database fundamentals, querying, filtering, JOINs, subqueries, window functions, data modification, schema design, constraints, indexes, optimization, stored routines, transactions, SQL dialect differences, real-world query patterns, interview preparation, and a compact reference cheat sheet.

It is written for beginners, developers who want strong database skills, data analysts, CS students, backend engineers, data engineers, job seekers, and anyone who wants to use databases confidently in real projects.

[LINKS] Copyright & License	© 2026 LearnStack. All rights reser redistribution or republishing witho Gumroad - thank you for supporting
[REAL-WORLD] Built for paid learning	The goal is not to memorize SQL sy query, protect, optimize and analyz sample database so the book feels

[INFO] How to use this handbook	Read Chapters 1-8 linearly if you are new. Use Chapters 9-16 as professional reference. Practice Chapter 17 repeatedly before interviews. Keep Chapter 18 nearby when writing SQL.

Learning Path	Focus
1. SQL Basics	Tables, SELECT, WHERE, NULL, CASE
2. Querying	Sorting, limiting, aggregation, GROUP BY
3. Relationships	INNER, LEFT, FULL, CROSS and SELF JOINs
4. Advanced Querying	Subqueries, CTEs, recursive CTEs and window functions
5. Professional SQL	Design, constraints, indexing, transactions, stored logic
6. Career Ready	Patterns, interview Q&A, coding problems and cheat sheets

Table of Contents

A premium, structured SQL learning path from database basics to interview-ready mastery.

Table of Contents

Table of Contents

Introduction to Databases & SQL

LearnStack SQL Complete Handbook - 2026 Edition

01

1.1 What is a Database?

[INFO] Definition — A database is an organized collection of structured data stored electronically. A DBMS (Database Management System) is the software used to create, manage, secure and query that data.

Databases exist because spreadsheets and flat files break down when data becomes shared, large, concurrent, sensitive or mission-critical. A spreadsheet is excellent for small manual analysis, but weak for enforcing rules, handling many users at once, recovering from failure, and maintaining relationships across millions of records.

A useful analogy is a library. A pile of books is data without structure. A catalog with shelves, identifiers, authors, subjects, checkout rules and search is a database. SQL gives you the language to ask the catalog precise questions.

[REAL-WORLD] Companies use databases to run the business — Payments, user accounts, product catalogs, order history, HR records, analytics dashboards, audit logs and inventory systems are all backed by databases. SQL is the shared language between software, analytics and operations teams.

1.2 Types of Databases

Type	Description	Examples	Best For
Relational (SQL)	Data stored in tables with rows, columns, keys and relationships	PostgreSQL, MySQL, SQLite, SQL Server, Oracle	Business apps, reporting, transactions, analytics
Document	Semi-structured documents, usually JSON-like	MongoDB, CouchDB	Flexible content, event payloads, catalogs
Key-Value	Fast lookup by key	Redis, DynamoDB	Caching, sessions, counters
Column-Family	Wide-column distributed storage	Cassandra, HBase	Huge write-heavy datasets
Graph	Nodes and edges	Neo4j, Amazon Neptune	Networks, recommendations, fraud paths
Time-Series	Optimized for timestamped measurements	InfluxDB, TimescaleDB	Metrics, IoT, observability

[INFO] Focus of this handbook — This handbook focuses on relational databases - the most widely used database model in business systems for decades.

1.3 What is SQL?

[INFO] SQL — SQL (Structured Query Language) is the standard language for creating, managing and querying relational databases. Both pronunciations - S-Q-L and sequel - are accepted.

SQL originated at IBM in the 1970s and became an ANSI standard in 1986 and ISO standard in 1987. SQL is declarative: you describe what data you want, while the database optimizer decides how to retrieve it efficiently.

Sublanguage	Full Name	Commands	Purpose
DDL	Data Definition Language	CREATE, ALTER, DROP, TRUNCATE	Define database objects
DML	Data Manipulation Language	SELECT, INSERT, UPDATE, DELETE	Read and change data
DCL	Data Control Language	GRANT, REVOKE	Control permissions
TCL	Transaction Control Language	BEGIN, COMMIT, ROLLBACK, SAVEPOINT	Control transactions

1.4 How a Relational Database Works

A relational database stores data in tables. A row is one record. A column is one attribute. Primary keys uniquely identify rows. Foreign keys connect rows across tables. Relationships can be one-to-one, one-to-many or many-to-many.

[SCHEMA] LearnStack Sample Database	This book uses one consistent sample schema: employees, departments, projects, employee_projects, salaries, customers, orders, order_items and products. The employee_projects table is a bridge table for the many-to-many relationship between employees and projects.

Table: employees

Column	Type / Constraint	Meaning
employee_id	INT PRIMARY KEY	Unique employee identifier
first_name	VARCHAR(50)	Given name
last_name	VARCHAR(50)	Family name
email	VARCHAR(100)	Business email
department_id	INT FK -> departments	Department assignment
salary	DECIMAL(10,2)	Current salary
hire_date	DATE	Date joined
manager_id	INT FK -> employees	Self-reference to manager
job_title	VARCHAR(100)	Role title
is_active	BOOLEAN	Employment status

Table: departments

Column	Type / Constraint	Meaning
department_id	INT PRIMARY KEY	Unique department identifier
department_name	VARCHAR(100)	Department name
location	VARCHAR(100)	City or office
budget	DECIMAL(15,2)	Annual budget

Table: projects

Column	Type / Constraint	Meaning
project_id	INT PRIMARY KEY	Unique project identifier
project_name	VARCHAR(200)	Project name
start_date	DATE	Start date
end_date	DATE	End date
budget	DECIMAL(15,2)	Project budget
status	VARCHAR(20)	active, completed, on_hold
department_id	INT FK -> departments	Owning department

Table: employee_projects

Column	Type / Constraint	Meaning
employee_id	INT FK -> employees	Assigned employee
project_id	INT FK -> projects	Assigned project
role	VARCHAR(100)	Project role
hours_worked	DECIMAL(8,2)	Logged hours
PRIMARY KEY	(employee_id, project_id)	Composite key

Table: salaries

Column	Type / Constraint	Meaning
salary_id	INT PRIMARY KEY	Unique salary record
employee_id	INT FK -> employees	Employee
amount	DECIMAL(10,2)	Salary amount
effective_date	DATE	Effective date
end_date	DATE	End date, NULL when current

Table: customers

Column	Type / Constraint	Meaning
customer_id	INT PRIMARY KEY	Unique customer identifier
full_name	VARCHAR(100)	Customer name
email	VARCHAR(100)	Email
city	VARCHAR(100)	City
country	VARCHAR(100)	Country
created_at	TIMESTAMP	Signup timestamp

Table: orders

Column	Type / Constraint	Meaning
order_id	INT PRIMARY KEY	Unique order identifier
customer_id	INT FK -> customers	Customer
order_date	DATE	Order date
total_amount	DECIMAL(10,2)	Order total
status	VARCHAR(20)	pending, paid, shipped, cancelled

Table: order_items

Column	Type / Constraint	Meaning
item_id	INT PRIMARY KEY	Unique item line
order_id	INT FK -> orders	Order
product_id	INT FK -> products	Product
quantity	INT	Quantity
unit_price	DECIMAL(10,2)	Price at sale

Table: products

Column	Type / Constraint	Meaning
product_id	INT PRIMARY KEY	Unique product identifier
product_name	VARCHAR(200)	Product name
category	VARCHAR(100)	Category
price	DECIMAL(10,2)	Current price

Column	Type / Constraint	Meaning
stock_qty	INT	Stock available

1.5 SQL vs NoSQL - When to Use What

Criteria	SQL	NoSQL
Schema	Structured and strongly modeled	Flexible or schema-light
Scalability	Vertical scaling plus replicas/partitioning	Often built for horizontal scale
Transactions	Strong ACID support	Varies by product and model
Query Language	Standardized SQL	Product-specific APIs and query languages
Best for	Business systems, finance, reporting, integrity	Flexible documents, scale-out workloads, caching
Not ideal for	Rapidly changing nested documents without modeling	Complex joins and strict relational integrity
Examples	PostgreSQL, MySQL, SQL Server	MongoDB, Redis, Cassandra, Neo4j

[REAL-WORLD] SQL remains central

Most business applications still use a relational database at their core. Even modern data warehouses, BI tools and product analytics platforms rely heavily on SQL.

1.6 The SQL Job Landscape

SQL is used daily by data analysts, backend developers, data engineers, database administrators, BI analysts, ML engineers and product managers. Strong SQL helps you read product data, debug production issues, design reliable features, build dashboards, answer business questions and pass technical interviews.

Role	How SQL is used	Indicative salary context
Data Analyst	Dashboards, reports, cohort analysis, funnels	Entry to mid roles vary widely by city and company
Backend Developer	Feature data models, APIs, transactions, migrations	Higher with strong system design and database design
Data Engineer	Pipelines, warehouses, transformations, data quality	Strong demand where analytics platforms are mature
DBA	Security, backups, monitoring, tuning, replication	Specialist compensation depends on database depth
BI Analyst	Semantic layers, metrics, stakeholder reporting	Strong SQL plus visualization increases value
ML Engineer	Feature extraction, training data preparation	SQL complements Python and ML systems

[CAUTION] Salary ranges change

Salary numbers are market-dependent and change by year, location, company size, interview performance and specialization. Treat any salary table as guidance, not a guarantee.

Setting Up Your SQL Environment

LearnStack SQL Complete Handbook - 2026 Edition

02

2.1 Choosing a Database System

Database	Best For	License	Platform
MySQL	Web apps, LAMP stack, common hosting	Open-source + commercial	Windows, macOS, Linux
PostgreSQL	Advanced SQL, reliability, extensions	Open-source	Windows, macOS, Linux
SQLite	Learning, local apps, mobile, embedded	Public domain	Everywhere
SQL Server	Microsoft and enterprise environments	Commercial + free editions	Windows, Linux
Oracle	Large enterprise systems	Commercial	Enterprise platforms

[TIP] Best learning setup

For learning, start with SQLite for zero setup or PostgreSQL for industry depth. For web development, MySQL and PostgreSQL are the safest choices.

[REAL-WORLD] Production choices

Startups often choose PostgreSQL because it is powerful, open-source and reliable. Enterprises often keep SQL Server or Oracle because the ecosystem, tooling and legacy systems are already built around them.

2.2 Installing MySQL

Install MySQL from the official installer on Windows, Homebrew on macOS, or the package manager on Linux. After installation, verify the client works.

```sql
-- Windows: use MySQL Installer from mysql.com
-- macOS:
brew install mysql
brew services start mysql

-- Ubuntu/Debian:
sudo apt update
sudo apt install mysql-server -y

-- Verify:
mysql --version
```

Result: The terminal prints the installed MySQL client/server version.

2.3 Installing PostgreSQL

```sql
-- Windows: use the PostgreSQL installer from postgresql.org
-- macOS:
brew install postgresql@16
brew services start postgresql@16

-- Ubuntu/Debian:
sudo apt update
sudo apt install postgresql postgresql-contrib -y

-- Verify:
psql --version
```

Result: The terminal prints the installed psql version.

2.4 Installing SQLite

[INFO] SQLite — SQLite is a serverless database stored in a single file. It is perfect for learning SQL, local prototypes, mobile apps and small embedded systems.

```sql
-- Create or open a local database file:
sqlite3 learnstack.db

-- Inside SQLite shell:
.tables
.schema
.quit
```

Result: SQLite opens the database file and accepts SQL commands.

2.5 Recommended GUI Tools

Tool	Supports	Platform	Free?	Best For
DBeaver	Universal	Windows/macOS/Linux	Yes	Best overall free GUI
TablePlus	Many SQL databases	Windows/macOS	Freemium	Beautiful interface
pgAdmin	PostgreSQL	Windows/macOS/Linux	Yes	Official PostgreSQL admin
MySQL Workbench	MySQL	Windows/macOS/Linux	Yes	Official MySQL GUI
DataGrip	Universal	Windows/macOS/Linux	Paid	Professional JetBrains workflow
VS Code + SQLTools	Many SQL databases	All	Yes	Lightweight editor workflow

[TIP] Start with DBeaver — DBeaver works with almost every database and is free. It is a strong default for students, analysts and professional engineers.

2.6 Creating the LearnStack Sample Database

Run the following DDL in dependency order. Departments and products are created first because other tables reference them.

```sql
CREATE TABLE departments (
    department_id   INT PRIMARY KEY,
    department_name VARCHAR(100) NOT NULL,
    location        VARCHAR(100),
    budget          DECIMAL(15,2) CHECK (budget >= 0)
);
```

Result: Table departments is created with primary keys, foreign keys and relevant constraints.

```sql
CREATE TABLE employees (
    employee_id   INT PRIMARY KEY,
    first_name    VARCHAR(50) NOT NULL,
    last_name     VARCHAR(50) NOT NULL,
    email         VARCHAR(100) UNIQUE NOT NULL,
    department_id INT REFERENCES departments(department_id),
    salary        DECIMAL(10,2) CHECK (salary > 0),
    hire_date     DATE NOT NULL,
    manager_id    INT REFERENCES employees(employee_id),
    job_title     VARCHAR(100),
    is_active     BOOLEAN DEFAULT TRUE
);
```

Result: Table employees is created with primary keys, foreign keys and relevant constraints.

```sql
```

```
CREATE TABLE projects (
    project_id    INT PRIMARY KEY,
    project_name  VARCHAR(200) NOT NULL,
    start_date    DATE NOT NULL,
    end_date      DATE,
    budget        DECIMAL(15,2) CHECK (budget >= 0),
    status        VARCHAR(20) CHECK (status IN ('active','completed','on_hold')),
    department_id INT REFERENCES departments(department_id)
);
```

Result: Table projects is created with primary keys, foreign keys and relevant constraints.

SQL

```
CREATE TABLE employee_projects (
    employee_id  INT REFERENCES employees(employee_id),
    project_id   INT REFERENCES projects(project_id),
    role         VARCHAR(100),
    hours_worked DECIMAL(8,2) DEFAULT 0,
    PRIMARY KEY (employee_id, project_id)
);
```

Result: Table employee_projects is created with primary keys, foreign keys and relevant constraints.

SQL

```
CREATE TABLE salaries (
    salary_id      INT PRIMARY KEY,
    employee_id    INT REFERENCES employees(employee_id),
    amount         DECIMAL(10,2) NOT NULL,
    effective_date DATE NOT NULL,
    end_date       DATE
);
```

Result: Table salaries is created with primary keys, foreign keys and relevant constraints.

SQL

```
CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    full_name   VARCHAR(100) NOT NULL,
    email       VARCHAR(100) UNIQUE NOT NULL,
    city        VARCHAR(100),
    country     VARCHAR(100),
    created_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Result: Table customers is created with primary keys, foreign keys and relevant constraints.

SQL

```
CREATE TABLE products (
    product_id   INT PRIMARY KEY,
    product_name VARCHAR(200) NOT NULL,
    category     VARCHAR(100),
    price        DECIMAL(10,2) CHECK (price >= 0),
    stock_qty    INT CHECK (stock_qty >= 0)
);
```

Result: Table products is created with primary keys, foreign keys and relevant constraints.

SQL

```
CREATE TABLE orders (
    order_id     INT PRIMARY KEY,
    customer_id  INT REFERENCES customers(customer_id),
    order_date   DATE NOT NULL,
    total_amount DECIMAL(10,2) CHECK (total_amount >= 0),
    status       VARCHAR(20) CHECK (status IN ('pending','paid','shipped','cancelled'))
);
```

Result: Table orders is created with primary keys, foreign keys and relevant constraints.

SQL

```
CREATE TABLE order_items (
    item_id    INT PRIMARY KEY,
    order_id   INT REFERENCES orders(order_id),
    product_id INT REFERENCES products(product_id),
    quantity   INT CHECK (quantity > 0),
    unit_price DECIMAL(10,2) CHECK (unit_price >= 0)
);
```

Result: Table order_items is created with primary keys, foreign keys and relevant constraints.

LearnStack Free Preview

This was a free preview. Get the full book on LearnStack.

Visit: https://www.learnstack.co.in

Digital PDF delivery is handled through Gumroad email after purchase.