Day 1 SQL Learning Notes

Date: October 28, 2025

Topic: SQL Basics & DDL

Project: Student Management System Database

What I Learned Today

Database & Table Creation (DDL)

- Created my first database called `student_management` using `CREATE DATABASE`
- Built 3 tables: students, courses, and enrollments with proper structure
- Learned that DDL stands for Data Definition Language - it's for creating/modifying database structure
- Used `USE database_name` to switch between databases
- `DESCRIBE table_name` helps me see table structure anytime

Data Types I Used

- **INT** for IDs and whole numbers like student_id, course_id
- **STRING** for text like names and emails
- **DATE** for birthdates and enrollment dates
- **DECIMAL(3,2)** for GPA - learned this is better than FLOAT for precision
- **BOOLEAN** for TRUE/FALSE values like is_active status

Constraints - Rules for Data Quality

- **PRIMARY KEY** - makes column unique and not null, every table needs one (used for student_id, course_id)
- **NOT NULL** - forces user to provide value, used for important fields like name and email
- **UNIQUE** - no duplicates allowed, applied to email column
- **CHECK** - custom rules like GPA must be between 0.00 and 4.00
- **DEFAULT** - auto-fills value if nothing provided (enrollment_date defaults to today)

Inserting Data (INSERT)

- Must specify column names to avoid errors: `INSERT INTO table (col1, col2) VALUES (val1, val2)`
- Can insert multiple rows at once by separating with commas
- Inserted 10 students, 8 courses, and 21 enrollments
- Learned the hard way - column order matters! Got error when I didn't specify column names

Basic SELECT Queries

- `SELECT *` gets all columns, but better to specify exact columns needed
- `SELECT first_name, last_name FROM students` - gets only what I need
- Used `AS` to rename columns in output for better readability
- `COUNT(*)` counts total rows in table

Filtering with WHERE Clause

- **Comparison operators:** =, >, <, >=, <=, != (or <>)
  - Found students with GPA > 3.5
  - Students with GPA between 3.0 and 3.5
- **BETWEEN** for ranges: `WHERE gpa BETWEEN 3.0 AND 3.5`
- **IN** for multiple values: `WHERE state IN ('TX', 'CA', 'NY')`
- **LIKE** for pattern matching:
  - `'A%'` finds names starting with A
  - `'%ar%'` finds names containing 'ar'
  - `'%son'` finds names ending with 'son'
- **IS NULL** and **IS NOT NULL** for checking empty values
- **AND, OR, NOT** to combine conditions
  - Active students from Texas: `WHERE is_active = TRUE AND state = 'TX'`
  - Students from CA or NY: `WHERE state = 'CA' OR state = 'NY'`

Sorting Results (ORDER BY)

- `ORDER BY gpa DESC` - highest GPA first (descending)
- `ORDER BY gpa ASC` - lowest GPA first (ascending)
- ASC is default so `ORDER BY last_name` sorts A to Z automatically
- Can sort by multiple columns: `ORDER BY state, gpa DESC`

Limiting Results (LIMIT)

- `LIMIT 5` shows only first 5 rows

- Combined with ORDER BY to get top performers: `ORDER BY gpa DESC LIMIT 5`
- `OFFSET 3` skips first 3 rows - useful for pagination

Removing Duplicates (DISTINCT)

- `SELECT DISTINCT state FROM students` gave me unique states
- Used `COUNT(DISTINCT state)` to count how many unique states

Modifying Data (UPDATE & DELETE)

- Updated student GPA: `UPDATE students SET gpa = 3.95 WHERE student_id = 102`
- Can update multiple columns at once
- **Always use WHERE** or it updates everything!
- Deleted records with `DELETE FROM table WHERE condition`
- TRUNCATE removes all data but keeps table structure
- DROP removes entire table

Important Lessons Learned

- Always use WHERE in UPDATE and DELETE to avoid updating/deleting everything
- PRIMARY KEY prevents duplicate IDs automatically
- DECIMAL is better than FLOAT for exact values like money or GPA
- Specifying column names in INSERT prevents type mismatch errors
- Can combine WHERE, ORDER BY, and LIMIT in one query for powerful results
- Parentheses help when combining AND/OR conditions: `(state = 'TX' OR state = 'CA') AND gpa > 3.0`

Queries I Can Now Write Confidently

- Find top N records (top 5 students by GPA)
- Filter by multiple conditions (active students from specific states with high GPA)
- Pattern matching (find names starting with certain letters)
- Range queries (students with GPA in certain range)
- Summary counts and basic analytics
- Update and delete specific records safely