# Day 2 SQL Learning Notes

**Date:** October 29, 2025
**Topic:** DML Operations & Data Transformations
**Project:** Bank Account Management System

---

## Quick Summary

Learned how to modify and transform data using UPDATE, DELETE, CASE statements, NULL handling, and string functions. Built banking database with customers, accounts, and transactions.

---

## Key Concepts Mastered

### INSERT Operations

- Always specify column names: `INSERT INTO table (col1, col2) VALUES (val1, val2)`
- Can insert multiple rows at once (separate with commas)
- INSERT INTO SELECT copies data between tables
- DEFAULT values auto-fill if not provided

### UPDATE Operations

- **Simple:** `UPDATE table SET column = value WHERE condition`
- **Multiple columns:** Separate with commas in SET clause
- **With calculations:** `SET balance = balance + 100`
- **With CASE:** Apply different updates based on conditions
- **CRITICAL:** Always use WHERE or updates ALL rows

### DELETE Operations

- `DELETE FROM table WHERE condition`
- Test with SELECT first before deleting
- Soft delete: Mark as deleted instead of removing (safer)
- Always use WHERE clause to avoid deleting everything

### CASE Statements - IF-THEN Logic in SQL

- Like IF-THEN-ELSE in programming
- Syntax: `CASE WHEN condition THEN result ELSE default END`
- Checks top to bottom, stops at first TRUE condition
- Always ends with END keyword
- Used to categorize data (High/Medium/Low balance)
- Can use in SELECT, WHERE, ORDER BY, UPDATE

### NULL Handling

- NULL = unknown/missing, NOT zero or empty
- **IS NULL / IS NOT NULL** - check for missing values (NEVER use = NULL)
- **COALESCE(col, default)** - replace NULL with default value
- **NULLIF(col, value)** - convert specific value to NULL
- COUNT(*) includes NULLs, COUNT(column) ignores NULLs
- AVG/SUM automatically ignore NULL values

### String Functions

- **CONCAT / ||** - combine text: `first_name || ' ' || last_name`
- **UPPER/LOWER** - change case for standardization
- **REPLACE** - swap text: `REPLACE(phone, '555', 'XXX')` for masking
- **SUBSTRING** - extract part of text
- **TRIM** - remove extra spaces

### Type Conversion

- CAST changes data type: `CAST(balance AS STRING)`
- Used for calculations and formatting

---

## Real Banking Queries Built

- Customer portfolio with wealth tiers (Platinum/Gold/Silver/Bronze)
- Account health report (Overdrawn/Low Balance/Healthy)
- Transaction categorization (High/Medium/Low value)
- Risk assessment based on KYC status
- Contact information cleanup with data quality checks
- Monthly transaction summaries
- Automatic customer tier upgrades based on balance

---

# Key Formulas

**CASE Statement:**

```
CASE
   WHEN balance >= 50000 THEN 'High'
   WHEN balance >= 10000 THEN 'Medium'
   ELSE 'Low'
END
```

**NULL Handling:**

```
COALESCE(phone, 'Not Provided')  -- Replace NULL with text
NULLIF(balance, 0)  -- Convert 0 to NULL
WHERE phone IS NULL  -- Find missing values
```

**UPDATE Pattern:**

```
UPDATE table SET col = value WHERE condition
```

---

# Critical Lessons

1. **CASE = IF-THEN-ELSE** - makes decisions in SQL
2. **Always use WHERE** in UPDATE/DELETE (prevents disasters)
3. **NULL ≠ anything** - use IS NULL, not = NULL
4. **COALESCE = default values** - handles NULLs in reports
5. **Test before destroying** - SELECT before DELETE/UPDATE
6. **Order matters in CASE** - most specific condition first

---

# Common Mistakes Avoided

- Forgetting WHERE in UPDATE/DELETE (updates/deletes everything)
- Using = NULL instead of IS NULL (doesn't work)
- Missing END in CASE statements (syntax error)
- Wrong CASE order (general before specific catches everything)

- Not handling NULLs (causes unexpected results)