****

**SQL PRETRAINING SESSION 1**

**V1: PORTFOLIO HEALTH FOUNDATIONS**

**Subtitle:** A Beginner-Friendly Guide with SQL Queries, Business Logic, and Output Screenshots

**Prepared By:** Your Name (Optional)

**Date:** August 2025

**Context:** This pretraining document is part of the Skill AI Path Data Analyst Track.

It covers foundational SQL concepts, query patterns, and business logic relevant for Google, loan portfolio health during crisis situations.

All queries are structured and explained with space for visual outputs and business interpretation.

**Organization:** Skill AI Path | EduFin Training Program

# **Table of Contents**

1. Session Overview

2. Learning Objectives

3. Part 1: Data Exploration

4. Part 2: Aggregations

5. Part 3: Executive Formatting

6. Part 4: CASE Statements

7. Part 5: Data Validation

8. Skill Check & Assessment

9. Conclusion & Next Steps

# **1. Session Overview**

**Session Focus**

**Portfolio Health Assessment**

*"What is the scale and scope of our crisis?"*

This session introduces you to analyzing a loan portfolio in crisis using real-world SQL queries. You'll build foundational skills needed for any data analyst or business analyst role.

**Technical Foundation Required**

* Basic SQL syntax
* Aggregation functions (COUNT, SUM, AVG)
* Filtering with WHERE
* GROUP BY for summaries
* Currency formatting
* Case-based logic

Even if you're a beginner, this session is structured to teach everything from scratch. No prior data analytics experience is required.

**Time Investment**

4–5 hours of focused learning and practice.

You’ll write real SQL queries, interpret business meaning, and prepare data summaries that mimic real-life analyst work.

**Business Context**

Imagine you’re working for a fintech company (EduFin) that offers personal loans. A recent audit uncovered a discrepancy of ₹12 crores in the portfolio. Your task is to:

* Explore the structure and quality of customer and loan data

Identify gaps and inconsistencies

Calculate business metrics like total loan value, average loan, and risk distribution

Prepare executive summaries that decision-makers can trust

By the end of this session, you’ll be ready to present real insights to leadership — using only SQL.

LEARNING OBJECTIVES

By completing this pretraining, you will master:

- ✅ SELECT statements with proper WHERE filtering

- ✅ COUNT, SUM, AVG, MIN, MAX aggregation functions

- ✅ GROUP BY for business categorization

- ✅ Percentage calculations using subqueries

- ✅ Professional Indian currency formatting

- ✅ Basic CASE statements for business logic

- ✅ NULL handling in financial data analysis

- ✅ Executive presentation formatting

## **Part 1: Data Exploration**

**Purpose:**  
This section helps you get familiar with the structure and scale of the loan dataset. Before analyzing business performance, it’s crucial to understand how your data looks, what each column means, and if any values are missing.

**Learning Goals:**

Preview data with basic SELECT statements.

Count rows to understand volume.

Identify missing or incomplete fields using NULL checks.

**-- Preview loan and customer data**

SELECT TOP 5 \* FROM loans;

SELECT TOP 5 \* FROM customers;

**-- Count records**

SELECT COUNT(\*) as total\_loan\_records FROM loans;

SELECT COUNT(\*) as total\_customer\_records FROM customers;

**-- Check missing disbursement dates**

SELECT

COUNT(\*) as total\_records,

COUNT(disbursement\_date) as records\_with\_disbursement,

COUNT(\*) - COUNT(disbursement\_date) as missing\_disbursement\_dates

FROM loans;

**-- Payments table completeness**

SELECT TOP 5 \* FROM payments;

SELECT

COUNT(\*) as total\_payments,

COUNT(payment\_amount) as payments\_with\_amount,

COUNT(\*) - COUNT(payment\_amount) as missing\_amounts

FROM payments;

## **Part 2: Filtering and Cleansing**

**Purpose:**  
Not all data is useful. Before performing analytics, filter out the records that are incomplete, invalid, or irrelevant. This ensures high-quality inputs for your analysis.

**Learning Goals:**

* Use WHERE clause to filter by valid values.
* Remove entries with missing key fields.
* Focus only on recent or relevant data based on business rules.

**-- Filter valid loan records**

SELECT loan\_id, customer\_id, loan\_amount, loan\_status

FROM loans

WHERE disbursement\_date IS NOT NULL

AND loan\_amount > 0;

**-- Filter valid customers**

SELECT customer\_id, customer\_name, annual\_income

FROM customers

WHERE annual\_income IS NOT NULL

AND registration\_date >= '2023-01-01'

AND city\_id IS NOT NULL;

## **Part 3: Aggregation Mastery**

**Purpose:**  
Aggregation functions allow us to summarize large datasets into meaningful statistics like totals, averages, minimums, and maximums. These are the foundation for business reporting.

**Learning Goals:**

* Use SUM, AVG, MIN, MAX, and COUNT.
* Calculate basic business metrics like total portfolio size and average loan amount.
* Use conditional logic inside aggregation to break down portfolio status.

**-- Portfolio overview**

SELECT

COUNT(\*) as total\_loans,

SUM(loan\_amount) as total\_portfolio\_value,

AVG(loan\_amount) as average\_loan\_size,

MIN(loan\_amount) as smallest\_loan,

MAX(loan\_amount) as largest\_loan

FROM loans

WHERE disbursement\_date IS NOT NULL;

**-- Conditional aggregates**

SELECT

COUNT(CASE WHEN loan\_status = 'Active' THEN 1 END) as active\_loans,

COUNT(CASE WHEN loan\_status = 'Defaulted' THEN 1 END) as defaulted\_loans,

SUM(CASE WHEN loan\_status = 'Active' THEN loan\_amount ELSE 0 END) as active\_portfolio

FROM loans

WHERE disbursement\_date IS NOT NULL;

## **Part 4: Grouping and Categorization**

**Purpose:**  
GROUP BY is used to summarize data by categories like loan status or income type. This helps you derive insights like which type of loans are most common or risky.

**Learning Goals:**

* Group loans or customers by relevant business fields.
* Summarize each group with counts or averages.
* Use CASE inside GROUP BY for custom grouping like Small/Medium/Large loans.

**-- Group by loan status**

SELECT

loan\_status,

COUNT(\*) as loan\_count,

SUM(loan\_amount) as category\_total

FROM loans

WHERE disbursement\_date IS NOT NULL

GROUP BY loan\_status

ORDER BY loan\_count DESC;

**-- Group loans by range**

SELECT

CASE

WHEN loan\_amount < 500000 THEN 'Small Loans'

WHEN loan\_amount < 2000000 THEN 'Medium Loans'

ELSE 'Large Loans'

END as loan\_category,

COUNT(\*) as count,

AVG(loan\_amount) as avg\_amount

FROM loans

WHERE disbursement\_date IS NOT NULL

GROUP BY

CASE

WHEN loan\_amount < 500000 THEN 'Small Loans'

WHEN loan\_amount < 2000000 THEN 'Medium Loans'

ELSE 'Large Loans'

END

ORDER BY avg\_amount DESC;

## **Part 5: Percentage Calculations**

**Purpose:**  
Business leaders often want to know the *proportion* of values, not just raw totals. Calculating percentages lets you compare across categories.

**Learning Goals:**

* Derive percentages using subqueries and math operations.
* Present loan type or city-wise distributions as a percentage of the total.
* Round results for clean, readable reporting.

**-- Loan count percentage**

SELECT

loan\_status,

COUNT(\*) as loan\_count,

ROUND(

COUNT(\*) \* 100.0 / (

SELECT COUNT(\*)

FROM loans

WHERE disbursement\_date IS NOT NULL

),

2

) as percentage\_of\_portfolio

FROM loans

WHERE disbursement\_date IS NOT NULL

GROUP BY loan\_status

ORDER BY loan\_count DESC;

**-- Loan amount percentage**

SELECT

loan\_status,

SUM(loan\_amount) as status\_amount,

ROUND(

SUM(loan\_amount) \* 100.0 / (

SELECT SUM(loan\_amount)

FROM loans

WHERE disbursement\_date IS NOT NULL

),

2

) as percentage\_of\_value

FROM loans

WHERE disbursement\_date IS NOT NULL

GROUP BY loan\_status;

## **Part 6: Indian Currency Formatting**

**Purpose:**  
Financial reports should follow the Indian currency format (₹, lakhs, crores). Using proper formatting makes your analysis professional and executive-friendly.

**Learning Goals:**

* Use FORMAT() to convert numbers into currency.
* Convert raw numbers into lakhs/crores using math and string formatting.
* Apply localized presentation standards (like ‘₹ 25,00,000’ instead of ‘2500000’).

**-- Indian formatting**

SELECT

loan\_status,

COUNT(\*) as loan\_count,

FORMAT(SUM(loan\_amount), 'C0', 'en-IN') as portfolio\_value\_formatted,

FORMAT(AVG(loan\_amount), 'C0', 'en-IN') as average\_loan\_formatted

FROM loans

WHERE disbursement\_date IS NOT NULL

GROUP BY loan\_status;

**-- Format as Crores**

SELECT

loan\_status,

COUNT(\*) as loans,

CONCAT('₹', FORMAT(SUM(loan\_amount)/10000000, 'N2'), ' Cr') as portfolio\_crores

FROM loans

WHERE disbursement\_date IS NOT NULL

GROUP BY loan\_status;

## **Part 7: Executive Summary Reporting**

**Purpose:**  
Executives expect ready-to-use dashboards or reports. Your queries should produce data that's clean, categorized, formatted, and insightful in one view.

**Learning Goals:**

* Combine aggregates, percentages, and formatting into one result.
* Use aliases (AS) for clean column names.
* Sort output to highlight highest value or risk categories.

**-- Decision-ready summary**

SELECT

loan\_status as 'Loan Status',

COUNT(\*) as 'Number of Loans',

FORMAT(SUM(loan\_amount), 'C0', 'en-IN') as 'Portfolio Value',

CONCAT(

ROUND(COUNT(\*) \* 100.0 / (

SELECT COUNT(\*) FROM loans WHERE disbursement\_date IS NOT NULL

), 1), '%'

) as 'Portfolio %'

FROM loans

WHERE disbursement\_date IS NOT NULL

GROUP BY loan\_status

ORDER BY SUM(loan\_amount) DESC;

## **Part 8: CASE Logic – Income or Risk Categories**

**Purpose:**  
CASE statements allow you to create custom categories or apply business rules directly in SQL. This makes your analysis more intelligent and actionable.

**Learning Goals:**

* Create income tiers (Low, Medium, High) or risk levels.
* Tag records based on business-defined thresholds.
* Assign actions or recommendations for each category.

**-- Income categories**

SELECT

customer\_id,

annual\_income,

CASE

WHEN annual\_income < 300000 THEN 'Low Income'

WHEN annual\_income < 800000 THEN 'Medium Income'

WHEN annual\_income < 1500000 THEN 'High Income'

ELSE 'Premium Income'

END as income\_category

FROM customers

WHERE annual\_income IS NOT NULL;

**-- Business actions**

SELECT

loan\_status,

COUNT(\*) as loan\_count,

CASE

WHEN loan\_status = 'Defaulted' THEN 'URGENT: Immediate Action Required'

WHEN loan\_status = 'Overdue' THEN 'HIGH PRIORITY: Collection Focus'

WHEN loan\_status = 'Active' THEN 'STANDARD: Regular Monitoring'

ELSE 'REVIEW: Status Verification Needed'

END as business\_action

FROM loans

WHERE disbursement\_date IS NOT NULL

GROUP BY loan\_status;

## **Part 9: Multi-Level Business Logic**

**Purpose:**  
Business problems often require more than one condition. Nested and multiple CASE statements help encode multi-level logic, like risk + recommendation.

**Learning Goals:**

* Use CASE for layered decision making.
* Build logic chains (IF-ELSE-like) within SQL.
* Show both assessment and recommendation in one query.

**-- Risk and processing recommendation**

SELECT

customer\_id,

annual\_income,

CASE

WHEN annual\_income IS NULL THEN 'CRITICAL: Missing Income Data'

WHEN annual\_income < 200000 THEN 'HIGH RISK: Low Income Segment'

WHEN annual\_income BETWEEN 200000 AND 500000 THEN 'MEDIUM RISK: Moderate Income'

WHEN annual\_income BETWEEN 500001 AND 1000000 THEN 'LOW RISK: Stable Income'

ELSE 'PREMIUM: High Income Segment'

END as risk\_assessment,

CASE

WHEN annual\_income < 300000 THEN 'Enhanced Documentation Required'

WHEN annual\_income > 1500000 THEN 'Fast Track Eligible'

ELSE 'Standard Processing'

END as processing\_recommendation FROM customers;

## **Part 10: Data Quality and Validation**

**Purpose:**  
Bad data = bad analysis. Before presenting or using data, check for missing, invalid, or illogical entries. SQL helps automate these validations.

**Learning Goals:**

* Detect NULLs, future dates, or negative values.
* Use COALESCE() for fallback/default values.
* Present a validation summary for quality assurance.

**-- Missing values check**

SELECT

'Customers' as table\_name,

COUNT(\*) as total\_records,

COUNT(customer\_name) as valid\_names,

COUNT(phone\_number) as valid\_phones,

COUNT(annual\_income) as valid\_incomes,

COUNT(\*) - COUNT(customer\_name) as missing\_names

FROM customers

UNION ALL

SELECT

'Loans' as table\_name,

COUNT(\*) as total\_records,

COUNT(disbursement\_date) as valid\_disbursements,

COUNT(loan\_status) as valid\_status,

COUNT(loan\_amount) as valid\_amounts,

COUNT(\*) - COUNT(disbursement\_date) as missing\_disbursements

FROM loans;

**-- Safe default handling**

SELECT

customer\_id,

COALESCE(annual\_income, 0) as income\_for\_calculation,

COALESCE(phone\_number, 'Contact Missing') as contact\_status

FROM customers;

**-- Business rule validation**

SELECT

'Invalid Loan Amounts' as issue\_type,

COUNT(\*) as problem\_count

FROM loans

WHERE loan\_amount <= 0 OR loan\_amount IS NULL

UNION ALL

SELECT

'Future Disbursement Dates' as issue\_type,

COUNT(\*) as problem\_count

FROM loans

WHERE disbursement\_date > GETDATE()

UNION ALL

SELECT

'Missing Customer Data' as issue\_type,

COUNT(\*) as problem\_count

FROM customers

WHERE customer\_name IS NULL OR customer\_name = '';