**A blue and white logo

AI-generated content may be incorrect.SQL PRETRAINING SESSION 2**

**V1: GEOGRAPHIC RISK ANALYSIS**

**FOUNDATIONS**

**Subtitle**: Multi-Table Business Intelligence with JOINs, Regional Patterns & Risk Hotspots

**Context:** This pretraining document is part of the Skill AI Data Analyst Track.  
It covers **multi-table JOIN operations**, **geographic hierarchy navigation**, and **regional risk profiling** for the EduFin loan crisis.  
You’ll write real SQL queries that connect dimension and fact tables across city, state, customer and loan data.  
By the end, you'll know *where* the ₹12 crore portfolio risk is concentrated.

# Table of Contents

1. Session Overview
2. Learning Objectives
3. Part 1: Understanding Normalized Schemas
4. Part 2: JOIN Operations
5. Part 3: Geographic Aggregation
6. Part 4: Advanced Filtering
7. Skill Check & Assessment
8. Conclusion & Next Steps

# **1. Session Overview**

**Session Focus:**  
Geographic Risk Analysis — “*Where are our problems concentrated?”*

**Time Investment:**  
5–6 hours of focused learning

Business Context:  
Imagine EduFin is facing a ₹12 crore crisis. In Session 1, you identified the total size of the problem.  
Now, you'll find where the problem is concentrated — by analyzing city, state, and institution performance using multi-table JOINs.

## **2. Learning Objectives**

You will master:

* ✅INNER and LEFT JOIN logic
* ✅ Multi-table joins with dimension tables
* ✅ State → City → Customer hierarchy
* ✅ Risk % and tier-based aggregation
* ✅ Geographic filtering and drill-down
* ✅ Advanced WHERE, HAVING, and subqueries

# PART 1: UNDERSTANDING NORMALIZED SCHEMAS

## 1.1 Schema Relationships and Foreign Keys

**Purpose:**To understand how business entities like customers, cities, and states are connected using foreign keys in a normalized schema.

**Learning Objective:**Use INNER JOIN to navigate between multiple dimension tables and explore basic relationships between customers and their geography.

**-- Understanding table relationships**

**SELECT TOP** 5

c.customer\_id,

c.full\_name,

c.city\_id,

dc.city\_name,

dc.state\_id,

ds.state\_name

**FROM** customers c

**INNER JOIN** dim\_city dc **ON** c.city\_id = dc.city\_id

**INNER JOIN** dim\_state ds **ON** dc.state\_id = ds.state\_id;

**Output:**

**A black screen with white text

AI-generated content may be incorrect.**

**-- Exploring dimension table structure**

**SELECT TOP 5 \* FROM** dim\_city;

**SELECT TOP 5 \* FROM** dim\_state;

**SELECT TOP 5 \* FROM** customers;

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**-- Explore loan-customer relationship**

**SELECT TOP** 5

l.loan\_id,

l.customer\_id,

c.full\_name

**FROM** loans l

**INNER JOIN** customers c **ON** l.customer\_id = c.customer\_id;

**Output:**

**A black screen with white text

AI-generated content may be incorrect.**

**-- Explore loan-institution relationship**

**SELECT TOP** 5

l.loan\_id,

l.institution\_id,

i.institution\_name

**FROM** loans l

**INNER JOIN** institutions i **ON** l.institution\_id = i.institution\_id;

**Output:**

**A black background with white text

AI-generated content may be incorrect.**

## 1.2 Data Distribution Analysis

**Purpose:**  
To analyze customer concentration across different regions and tiers.

**Learning Objective:**  
Apply GROUP BY and LEFT JOIN to study how customers are distributed geographically at the state and city levels.

**-- Count records per state**

**SELECT TOP** 5

ds.state\_name,

**COUNT**(c.customer\_id) **as** customer\_count

**FROM** dim\_state ds

**LEFT JOIN** dim\_city dc **ON** ds.state\_id = dc.state\_id

**LEFT JOIN** customers c **ON** dc.city\_id = c.city\_id

**GROUP B**Y ds.state\_name

**ORDER BY** customer\_count **DESC;**

**Output:**

**A black and white screen with white text

AI-generated content may be incorrect.**

**-- Count records per city with state context**

**SELECT TOP** 5

ds.state\_name,

dc.city\_name,

dc.tier\_classification,

**COUNT**(c.customer\_id) **as** customer\_count

**FROM** dim\_state ds

**INNER JOIN** dim\_city dc **ON** ds.state\_id = dc.state\_id

**LEFT JOIN** customers c **ON** dc.city\_id = c.city\_id

**GROUP BY** ds.state\_name, dc.city\_name, dc.tier\_classification

**HAVING COUNT**(c.customer\_id) > 0

**ORDER B**Y ds.state\_name, customer\_count **DESC;**

**Output:**

A black screen with white text

AI-generated content may be incorrect.

# PART 2: MASTERING JOIN OPERATIONS

## 2.1 INNER JOIN - Core Business Relationships

**Purpose:**  
To retrieve only matching data between related entities, e.g., customers with loans.

**Learning Objective:**  
Connect customers and loans to analyze loan status and regional impact using INNER JOIN.

**-- Basic customer-loan relationship**

**SELECT TOP** 5 c.full\_name, c.annual\_income, l.loan\_amount, l.loan\_status, l.disbursement\_date

**FROM** customers c

**INNER JOIN** loans l **ON** c.customer\_id = l.customer\_id

**WHERE** l.disbursement\_date **IS NOT NULL**;

**Output:**

**A black board with white text

AI-generated content may be incorrect.**

**-- Geographic context for customers with loans**

**SELECT**

c.full\_name,

dc.city\_name,

ds.state\_name,

dc.tier\_classification,

**COUNT**(l.loan\_id**) as** total\_loans,

**SUM**(l.loan\_amount**) as** total\_loan\_amount

**FROM** customers c

**INNER JOIN** dim\_city dc **ON** c.city\_id = dc.city\_id

**INNER JOIN** dim\_state ds **ON** dc.state\_id = ds.state\_id

**INNER JOIN** loans l **ON** c.customer\_id = l.customer\_id

**WHERE** l.disbursement\_**date IS NOT NULL**

**GROUP BY** c.full\_name, dc.city\_name, ds.state\_name, dc.tier\_classification;

**Output:**

A black screen with white text

AI-generated content may be incorrect.

## 2.2 LEFT JOIN - Comprehensive Analysis

**Purpose:**  
To include records that may not have corresponding matches, e.g., cities with no customers or customers with no loans.

**Learning Objective:**  
Use LEFT JOIN and COALESCE to avoid data loss and identify gaps or inactive segments.

**-- All cities, even those without customers**

**SELECT**

ds.state\_name,

dc.city\_name,

dc.tier\_classification,

**COUNT**(c.customer\_id) as customer\_count,

**COALESCE(CO**UNT(c.customer\_id), 0) **as** customers\_with\_nulls\_handled

**FROM** dim\_state ds

**INNER JOIN** dim\_city dc **ON** ds.state\_id = dc.state\_id

**LEFT JOIN** customers c **ON** dc.city\_id = c.city\_id

**GROUP BY** ds.state\_name, dc.city\_name, dc.tier\_classification

**ORDER BY** customer\_count **DESC**;

**Output:**

**A black screen with white text

AI-generated content may be incorrect.**

**-- All customers with their loan information (including those without loans)**

**SELECT TOP** 5

c.customer\_id,

c.full\_name,

dc.city\_name,

ds.state\_name,

**COUNT**(l.loan\_id**) as** loan\_count,

**COALESCE**(**SUM(**l.loan\_amount), 0**) as** total\_loan\_amount,

CASE

**WHEN** **COUNT**(l.loan\_id) = 0 **THEN** 'No Loans'

**WHEN** **COUNT**(l.loan\_id) = 1 **THEN** 'Single Loan'

**ELSE** 'Multiple Loans'

**END as** loan\_profile

**FROM** customers c

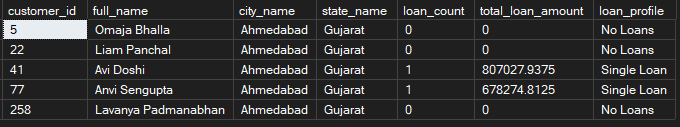
**INNER JOIN** dim\_city dc **ON** c.city\_id = dc.city\_id

**INNER JOIN** dim\_state ds **ON** dc.state\_id = ds.state\_id

**LEFT JOIN** loans l ON c.customer\_id = l.customer\_id **AND** l.disbursement\_date **IS NOT NULL**

**GROUP BY** c.customer\_id, c.full\_name, dc.city\_name, ds.state\_name;

**Output:**



## 2.3 Complex Multi-Table JOINs

**Purpose:**  
To analyze the full loan lifecycle and calculate key metrics like outstanding balance.

**Learning Objective:**  
Combine data across five tables to evaluate loans, payments, and institutional performance.

**-- Complete loan ecosystem analysis**

**SELECT**

c.full\_name,

dc.city\_name,

ds.state\_name,

i.institution\_name,

l.loan\_amount,

l.loan\_status,

l.disbursement\_date,

**COUNT**(p.payment\_id) **as** payment\_count,

**COALESCE(SUM**(p.payment\_amount), 0) **as** total\_payments,

l.loan\_amount - **COALESCE**(**SUM**(p.payment\_amount), 0) **as** outstanding\_balance

**FROM** customers c

**INNER JOIN** dim\_city dc **ON** c.city\_id = dc.city\_id

**INNER JOIN** dim\_state dsz **ON** dc.state\_id = ds.state\_id

**INNER JOIN** loans l **ON** c.customer\_id = l.customer\_id

**INNER JOIN** institutions i **ON** l.institution\_id = i.institution\_id

**LEFT JOIN** payments p **ON** l.loan\_id = p.loan\_id

**WHERE l**.disbursement\_date **IS NOT NULL**

**GROUP B**Y

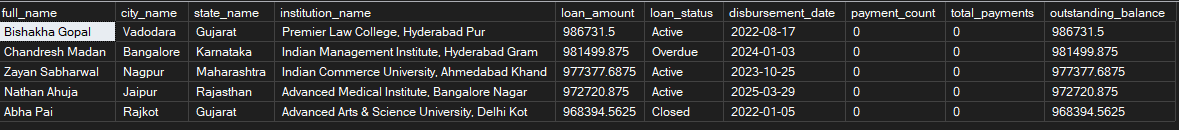
c.full\_name, dc.city\_name, ds.state\_name,

i.institution\_name,

l.loan\_amount, l.loan\_status, l.disbursement\_date

**ORDER BY** outstanding\_balance **DESC;**

**Output:**



# PART 3: GEOGRAPHIC AGGREGATION PATTERNS

## 3.1 Hierarchical Grouping

**Purpose:**  
To identify where most loans/customers are located and evaluate loan portfolio size across state and city levels.

**Learning Objective:**  
Use aggregation and formatting functions to study loan penetration by geography and tier.

**-- State-level aggregation**

**SELECT TOP** 5

ds.state\_name,

**COUNT(DISTINCT** c.customer\_id**) as** total\_customers,

**COUNT(DISTINCT** dc.city\_id**) as** cities\_with\_customers,

**COUNT**(l.loan\_id) **as** total\_loans,

**SUM**(l.loan\_amount) **as** state\_portfolio\_value,

**AVG**(l.loan\_amount) **as** avg\_loan\_size

**FROM** dim\_state ds

**INNER JOIN** dim\_city dc **ON** ds.state\_id = ds.state\_id

**INNER JOIN** customers c **ON** dc.city\_id = c.city\_id

**INNER JOIN** loans l **ON** c.customer\_id = l.customer\_id

**WHERE** l.disbursement\_date **IS NOT NULL**

**GROUP BY** ds.state\_name

**ORDER BY** state\_portfolio\_value **DESC;**

**Output:**

A screenshot of a black screen

AI-generated content may be incorrect.

**-- City-level detailed analysis**

**SELECT**

ds.state\_name,

dc.city\_name,

dc.tier\_classification,

**COUNT**(DISTINCT c.customer\_id) **as** customers,

**COUNT**(l.loan\_id) **as** loans,

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

**ROUND(AVG(l**.loan\_amount), 0) **as** avg\_loan\_amount

**FROM** dim\_state ds

**INNER JOIN** dim\_city dc **ON** ds.state\_id = ds.state\_id

**INNER JOIN** customers c **ON** dc.city\_id = c.city\_id

**INNER JOIN** loans **l ON** c.customer\_id = l.customer\_id

**WHERE** l.disbursement\_date **IS NOT NULL**

**GROUP BY** ds.state\_name, dc.city\_name, dc.tier\_classification

**HAVING COUNT**(l.loan\_id) >= 5

**ORDER BY SUM**(l.loan\_amount) **DESC;**

**Output:**

A black rectangular sign with white text

AI-generated content may be incorrect.

## 3.2 Risk Concentration Analysis

**Purpose:**  
To detect regional hotspots where risk (defaults and overdue loans) is abnormally high.

**Learning Objective:**  
Classify states into high/medium/low risk using calculated problem loan percentages.

**-- Geographic risk distribution**

**SELECT**

ds.state\_name,

**COUNT**(l.loan\_id) **as** total\_loans,

**SUM**(l.loan\_amount) **as** total\_portfolio,

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

**COUNT**(**CASE WHEN l**.loan\_status = 'Overdue' **THEN 1 END) as** overdue\_loans,

**ROUND(**

**COUNT(CASE WHEN** l.loan\_status IN ('Defaulted', 'Overdue') **THEN 1 END)** \* 100.0 / **COUNT**(l.loan\_id),

2

) as problem\_loan\_percentage,

**CASE**

**WHEN COUNT(CASE** **WHEN l**.loan\_status **IN** ('Defaulted', 'Overdue') **THEN 1 END**) \* 100.0 **/ COUNT**(l.loan\_id) > 20

**THEN** 'HIGH RISK STATE'

**WHEN** **COUNT(CASE WHEN l**.loan\_status **IN** ('Defaulted', 'Overdue') **THEN 1 END**) \* 100.0 **/ COUNT**(l.loan\_id) > 10

**THEN** 'MEDIUM RISK STATE'

**ELSE** 'LOW RISK STATE'

**END as** risk\_classification

**FROM** dim\_state ds

**INNER JOIN** dim\_city dc **ON** ds.state\_id = ds.state\_id

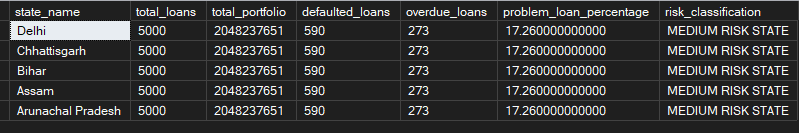
**INNER JOIN** customers c **ON** dc.city\_id = c.city\_id

**INNER JOIN** loans **l ON** c.customer\_id = l.customer\_id

**WHERE** l.disbursement\_date **IS NOT NULL**

**GROUP BY** ds.state\_name **ORDER BY** problem\_loan\_percentage **DESC;**

**Output:**



## 3.3 Cross-Regional Comparisons

**Purpose:**  
To evaluate whether a state's performance (loan size and risk) is better or worse than the national benchmark.

**Learning Objective:**  
Use CTEs and comparisons to flag states that require deeper business review.

**-- State-wise comparison with national benchmarks**

**WITH** national\_metrics AS (

**SELECT**

**AVG**(loan\_amount**) as** national\_avg\_loan,

**COUNT**(**CASE WHEN** loan\_status **IN** ('Defaulted', 'Overdue') **THEN 1 END**) \* 100.0 / **COUNT(\*)** **as** national\_problem\_rate

**FROM** loans

**WHERE** disbursement\_**date IS NOT NULL**

),

state\_metrics **AS** (

**SELECT**

ds.state\_name,

**COUNT**(l.loan\_id**) as** loans,

**AVG**(l.loan\_amount**) as** avg\_loan,

**COUNT**(**CASE WHEN** l.loan\_status **IN** ('Defaulted', 'Overdue') **THEN 1 END**) \* 100.0 / **COUN**T(l.loan\_id) as problem\_rate

**FROM** dim\_state ds

**INNER JOIN** dim\_city dc **ON** ds.state\_id = ds.state\_id

**INNER JOIN** customers c **ON** dc.city\_id = c.city\_id

**INNER JOIN** loans **l ON** c.customer\_id = l.customer\_id

**WHERE** l.disbursement\_**date IS NOT NULL**

**GROUP BY** ds.state\_name

)

**SELECT TOP** 5

sm.state\_name,

sm.loans,

**FORMAT**(sm.avg\_loan, 'C0', 'en-**IN') as** avg\_loan\_formatted,

**ROUND(**sm.problem\_rate, 2) **as** problem\_rate,

**CASE**

**WHEN** sm.avg\_loan > nm.national\_avg\_loan \* 1.2 **THEN** 'Above National Average'

**WHEN** sm.avg\_loan < nm.national\_avg\_loan \* 0.8 **THEN** 'Below National Average'

**ELSE** 'Near National Average'

**END** **as** loan\_size\_comparison,

**CASE**

**WHEN** sm.problem\_rate > nm.national\_problem\_rate \* 1.5 **THEN 'CRITICAL**: Well Above National'

**WHEN** sm.problem\_rate > nm.national\_problem\_rate **THEN 'CAUTION**: Above National'

**ELSE 'GOOD**: Below National'

**END as** risk\_comparison

**FROM s**tate\_metrics sm

**CROSS JOIN** national\_metrics nm

**ORDER BY** sm.problem\_rate **DESC;**

**Output:**

A black background with white numbers

AI-generated content may be incorrect.

# PART 4: ADVANCED FILTERING ACROSS TABLES

## 4.1 Complex WHERE Conditions

**Purpose:**  
To isolate high-value customers based on geographic and financial criteria.

**Learning Objective:**  
Use WHERE, HAVING, and aggregates to filter customers across multiple business dimensions.

**-- Find high-value customers in specific geographic regions**

**SELECT**

c.full\_name,

dc.city\_name,

ds.state\_name,

c.annual\_income,

**COUNT**(l.loan\_id) **as** loan\_count,

**SUM**(l.loan\_amount**) as** total\_borrowed

**FROM** customers c

**INNER JOIN** dim\_city dc **ON** c.city\_id = dc.city\_id

**INNER JOIN** dim\_state ds **ON** dc.state\_id = ds.state\_id

**INNER JOIN** loans l **ON** c.customer\_id = l.customer\_id

**WHERE** l.disbursement\_date **IS NOT NULL**

**AND** c.annual\_income > 1000000

**AND** dc.tier\_classification = 'Tier1'

**AND** l.loan\_status **IN** ('Active', 'Closed')

**GROUP BY** c.full\_name, dc.city\_name, ds.state\_name, c.annual\_income

**HAVING SUM**(l.loan\_amount) > 2000000

**ORDER BY** total\_borrowed **DESC;**

**Output:**



## 4.2 Subqueries for Geographic Analysis

**Purpose:**  
To find cities with higher-than-average default rates.

**Learning Objective:**  
Use subqueries in HAVING clause to compare local and national patterns.

**-- Cities with above-average default rates**

**SELECT TOP** 5

ds.state\_name,

dc.city\_name,

**COUNT(**l.loan\_id) **as** total\_loans,

**COUNT**(**CASE WHEN** l.loan\_status = 'Defaulted' **THEN 1 END**) **as** defaults,

**ROUND(**

**COUNT(CASE WHEN** l.loan\_status = 'Defaulted' **THEN 1 END)** \* 100.0 / **COUNT(**l.loan\_id),

2

**) as** default\_rate

**FROM** dim\_state ds

**INNER JOIN** dim\_city dc **ON** ds.state\_id = ds.state\_id

**INNER JOIN** customers c **ON** dc.city\_id = c.city\_id

**INNER JOIN** loans **l ON c**.customer\_id = l.customer\_id

**WHERE** l.disbursement\_date **IS NOT NULL**

**GROUP BY** ds.state\_name, dc.city\_name

**HAVING COUNT**(l.loan\_id) >= 10

**AND COUNT**(**CASE WHEN** l.loan\_status = 'Defaulted' **THEN 1 END) \*** 100.0 / **COUNT(**l.loan\_id) >

(

**SELECT COUNT**(**CASE WHEN** loan\_status = 'Defaulted' **THEN 1 END**) \* 100.0 / **COUNT(\*)**

**FROM** loans

**WHERE** disbursement\_date **IS NOT NULL**

)

**ORDER BY** default\_rate **DESC;**

**Output:**

A screen shot of a black board

AI-generated content may be incorrect.

© 2025 Skill AI Path. All Rights Reserved.  
Licensed for educational and internal business use.  
Attribution required when sharing. Commercial redistribution prohibited.

Contact Us - [tech@skillaipath.com](mailto:tech@skillaipath.com) | [skillaipath.com](http://skillaipath.com/)