IIT Guwahati

Advanced Certification Programme in Data Science Business Analytics

Week 6

Topics Covered

- Joins in MySQL
- Q and A

Joins in MySQL

Understanding MySQL Joins

Combining Data From Multiple Tables For Efficient Analysis

Definition

- Combine data from two or more related tables
- Merge rows using a common column (e.g. foreign key)
- Create meaningful connections between tables

Purpose and example

- Retrieve comprehensive information in one query
- Make data analysis efficient and organised
- **Example**: Join students and courses tables to show enrolled students with their courses

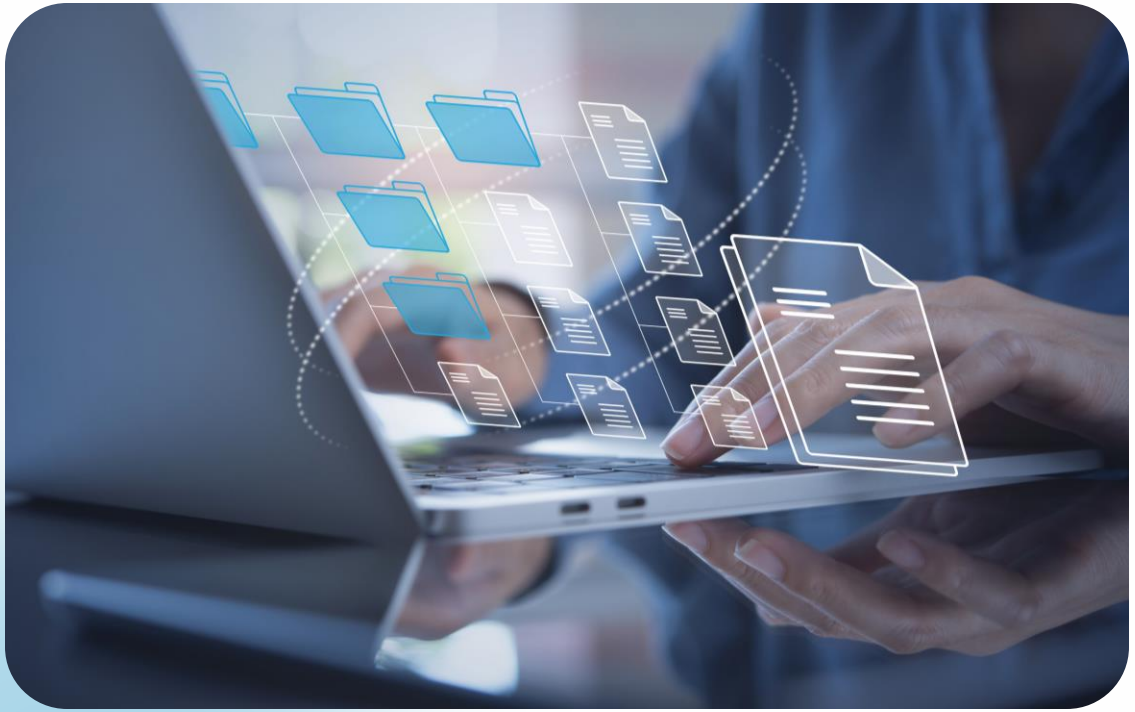
Benefits of Using MySQL Joins

Using Joins For Efficient and Organised Data Handling

Benefit	Description
Data normalisation	Link tables to reduce redundancy and get combined data
Comprehensive data retrieval	Pull related data in one query to simplify analysis
Efficiency in data management	Maintain consistency and avoid duplication by breaking data into tables
Complex queries	Enable advanced queries needing data from multiple tables

Type of Joins in MySQL

Efficient Data Retrieval with MySQL Joins



- Left join or left outer join
- Right join or right outer join
- Inner join
- Cross join
- Full outer join

Understanding LEFT JOIN in MySQL

Concept, Usage and Syntax

```
SELECT columns FROM table1
LEFT JOIN table2 ON
table1.column_name =
table2.column_name;
```

- Return all records from the left table and matching records from the right
- Show NULL for right table columns if no match found
- **Use case**: Get all records from left table even if no match in right table

Case Study: Course Enrolment with LEFT JOIN

Understanding Table Relationships in a University System

- Identify two main tables: students and enrolments
- Store student information in students table
- Record enrolled courses in enrolments table
- Use LEFT JOIN to display all students, with or without enrolments

Case Study: University Course Enrolment

Linking Students and Enrolments Using LEFT JOIN

student_id	student_name	email
1	Alina	alina@testmail.com
2	Seema	seems@testemail.com
3	Hashima	hashima@testmail.com
4	Ekta	ekta@testemail.com

Student table

student_id (Primary Key)
student_name
Email

enrollment_id	student_id	course_name
1	1	Math
2	1	Chemistry
3	2	Science
4	3	History

Enrolments table

enrollment_id (Primary Key)
student_id (Foreign Key)
course_name

University Course Enrolment Report

University Course Enrolment Report

SQL query

```
SELECT
    Students.student_id,
    Students.student_name,
    Students.email,
    Enrollments.course_name
FROM Students
LEFT JOIN
    Enrollments ON
    Students.student_id =
Enrollments.student_id;
```

- Generate a report of all students with their enrolled courses
- Consider students who may have no course enrolments
- Ensure complete student details are included
- Show students even without course enrolments
- Use LEFT JOIN to include all students
- Ensure no student is left out of the report

University Course Enrolment Report

Expected Output From LEFT JOIN

Expected output

student_id	student_name	email	Course
1	Alina	alina@testmail.com	Maths
1	Alina	alina@testmail.com	Chemistry
2	Seema	seems@testemail.com	Science
3	Hashima	hashima@testmail.com	History
4	Ekta	ekta@testmail.com	NULL

Inner Join in SQL

Combining Data from Multiple Tables Using Matching Records

```
SELECT column1, column2, ...
FROM table1
INNER JOIN table2 ON
table1.column_name =
table2.column_name;
```

- Retrieves matching records from two or more tables
- Returns rows with corresponding matches in both tables
- Filters out records without matching data
- Excludes non-matching records from the result set

E-commerce Order Management System

Case Study on Customer and Order Relationships for Business

- Analyse relationship between Customers and Orders
- Use two tables: Customers table and orders table
- Store customer details in customers table
- Record all purchases in orders table
- Understand customer purchasing patterns for insights

E-commerce Order Management System

Table Structures for Case Study

customer_id	customer_name	email
1	Alina	alina@testmail.com
2	Seema	seems@testemail.com
3	Hashima	hashima@testmail.com
4	Ekta	ekta@testemail.com

Customer table

customer_id (Primary Key)
customer_name
Email

order_id	customer_id	total_amount
1	1	300
2	1	500
3	2	200
4	3	900

Orders table

order_id (Primary Key)
customer_id (Foreign Key)
total_amount

Case Study: E-commerce Order Insights

Understanding Customer-Order Relationships with INNER JOIN

SQL query

```
SELECT
    Customers.customer_name,
    Orders.order_id,
    Customers.email,
    Orders.total_amount
FROM
    Customers
INNER JOIN
    Orders
ON
    Customers.customer_id =
Orders.customer_id;
```

- List customers along with their orders placed
- Focus only on customers who have placed orders
- Use INNER JOIN to fetch customers with confirmed purchases
- Show customers who have made at least one order

E-commerce Order Management System

Expected Output from INNER JOIN

Expected output

customer_id	customer_name	email	total_amount
1	Alina	alina@testmail.com	300
1	Alina	alina@testmail.com	500
2	Seema	seems@testemail.com	200
3	Hashima	hashima@testmail.com	900

Understanding Right Join in SQL

Ensure Complete Data Retrieval From The Right Table

```
SELECT columns
FROM table1
RIGHT JOIN table2 ON
table1.common_column =
table2.common_column;
```

- Retrieve all rows from the right table that have matching rows in the left table
- Include all right table data even if no match in left table
- Display NULL values for unmatched left table columns
- Use when right table data must be fully included in the result

Company Training Management System

Case Study on Course and Employee Data Using RIGHT JOIN

- Manage employee training enrolments efficiently
- Use RIGHT JOIN to display all courses
- Show courses even without employee enrolments
- Focus on both employees and courses data
- **Tables Involved:**
  - Employees table
  - Courses table

Company Training Management System

Table Structures for RIGHT JOIN Case Study

employee_id	employee_name	course_id
1	Alina	101
2	Seema	102
3	Hashima	NULL

Employee table

employee_id (Primary Key)
employee_name
course_id (Foreign Key)

course_id	course_name
101	Project management
102	Security
103	Development

Course table

course_id (Primary Key)
Course_name

Training Course Enrolment Report

Case Study on Listing All Courses Using RIGHT JOIN

SQL query

```
SELECT
Employees.employee_name,
Courses.course_name
FROM Employees
RIGHT JOIN Courses ON
Employees.course_id =
Courses.course_id;
```

- Generate report of all training courses with enrolled employees
- Include courses even if no employees are enrolled
- Use Right Join to show all available courses
- Ensure listing of courses with or without participants

Company Training Management System

Expected Output From RIGHT JOIN

Expected output

employee_name	course_name
Alina	Project management
Seema	Security
NULL	Development

Understanding Cross Join in SQL

Create Comprehensive Combinations of Data From Multiple Tables

```
SELECT columns
FROM table1
CROSS JOIN table2;
```

- Return a cartesian product that combines all rows from both tables
- Avoid any matching condition or relationship between tables
- Generate all possible row combinations from two or more tables
- Help in analysing every potential pairing of records from both tables

Product and Discount Pairing with Cross Join

Case Study: Combining Multiple Records

- Evaluate all possible product and discount combinations for pricing scenarios
- Help marketing team analyse various pricing options effectively
- Use Cross Join to pair each product with every available discount
- Involve products and discount tables for generating combinations

Exploring Record Combinations Using Cross Join

Generating All Possible Data Pairings

product_id	product_name	base_price
1	Laptop	1000
2	Phone	500

Products table

product_id (Primary Key)
product_name
base_price

discount_id	discount_percentage
101	10%
102	20%

Discounts table

discount_id (Primary Key)
discount_percentage

Leveraging Cross Join for Data Analysis

Generating Comprehensive Product and Discount Combinations

SQL query

```
SELECT Products.product_name,
Products.base_price,
Discounts.discount_percentage,
Products.base_price * (1 - Discounts.discount_percentage / 100) AS discounted_price
FROM Products
CROSS JOIN Discounts;
```

- Generate all possible combinations of products and discount rates
- Ensure each product is matched with every discount scenario without gaps
- Simulate pricing scenarios to analyse potential revenue outcomes
- Help marketing teams design effective discount strategies

Product and Discount Combinations

Expected Output from CROSS JOIN

Expected output

product_name	base_price	discount_percentage	discounted_price
Laptop	1000	10%	900
Laptop	1000	20%	800
Phone	500	10%	450
Phone	500	20%	400

Full Outer Join – Combining Left & Right Joins

Ensuring Retrieving All data

```
SELECT columns FROM table1

LEFT JOIN table2 ON
table1.common_column =
table2.common_column

UNION SELECT columns FROM
table1

RIGHT JOIN table2 ON
table1.common_column =
table2.common_column;
```

- Combines Left Join & Right Join to return all records from both tables
- Displays matching records where available, fills gaps with NULL
- Ensures no data is lost, even if no match exists
- Shows every row from both tables

Managing Products & Stock Efficiently

Case Study on Using Full Outer Join for a Complete Inventory

- Tracks products and warehouses in an inventory system
- Products table lists all items
- Warehouses table stores stock details
- Includes products without warehouses and warehouses with no stock
- **Tables Involved**
  - Products table
  - Warehouse table

Case Study: Product & Discount Data Mapping

Understanding Table Structures & Relationships

product_id	product_name
1	Laptops
2	Desktops
3	Monitors

Products table

product_id (Primary Key)
product_name

Warehouse_id	product_id	quantity
101	1	50
102	2	0
103	NULL	NULL

Discounts table

warehouse_id (Primary Key)
product_id (Foreign Key)
quantity

Ensuring Complete Inventory Visibility

Case Study on Using Full Outer Join for Stock Tracking

SQL query

```
SELECT Products.product_name,
Warehouses.quantity FROM Products

LEFT JOIN Warehouses ON
Products.product_id =
Warehouses.product_id

UNION

SELECT Products.product_name,
Warehouses.quantity

FROM Products

RIGHT JOIN Warehouses ON

Products.product_id =
Warehouses.product_id;
```

- Full Outer Join retrieves all products and warehouses
- Displays unmatched products and warehouses with no stock
- Ensures complete inventory visibility
- Supports transparent & efficient tracking

Product and Discount Combinations

Expected Output in Full Outer Join

Expected output

product_name	quantity
Laptops	50
Desktops	0
Monitors	NULL
NULL	NULL

Q & A

Thank you