Lecture Notes – UNION & UNIONALL

Section 1: Lecture Summary

SQL queries demonstrate **UNION** and **UNION ALL** as set operations to combine results from multiple compatible queries. Examples show rewriting simple WHERE clause queries using UNION to reduce complexity, such as listing employees by hire date or salary from companyDB, employees by IT or Finance projects using joins across Employees, EmployeeProjects, and Projects tables, and high-value orders or credit card payments from eCommerceDB Orders and Payments tables.

Section 2: Key Concepts and Explanations

Set operations like **UNION** combine results from two or more SELECT queries with matching number of columns, same order, and compatible data types/domains. **UNION** removes duplicates automatically, while **UNION ALL** retains them for performance when duplicates are expected. Use UNION to break complex queries into simpler parts, avoiding lengthy WHERE clauses or large joins—filter each table separately then unite results. Queries must produce compatible outputs; for example, integer and decimal columns can pair if domains align.

Section 3: Example Code and Use Cases

Query 1: Employees hired before 2020 or earning >90,000 (companyDB, Employees table)

```
SELECT EmpID, FirstName, LastName, HireDate, Salary
FROM Employees
WHERE HireDate < '2020-01-01'
UNION
SELECT EmpID, FirstName, LastName, HireDate, Salary
FROM Employees
WHERE Salary > 90000;
```

Query 2: Employees in IT (DeptID=2) or Finance (DeptID=3) projects (companyDB)

```
SELECT e.EmpID, e.FirstName, p.ProjectName
FROM Employees e
NATURAL JOIN EmployeeProjects ep
JOIN Projects p ON ep.ProjectID = p.ProjectID
WHERE p.DeptID = 2
```

```
UNION
SELECT e.EmpID, e.FirstName, p.ProjectName
FROM Employees e
NATURAL JOIN EmployeeProjects ep
JOIN Projects p ON ep.ProjectID = p.ProjectID
WHERE p.DeptID = 3;
```

Query 3: High-value orders (>100,000) or credit card payments (eCommerceDB)

```
SELECT OrderID, TotalAmount
FROM Orders
WHERE TotalAmount > 100000
UNION
SELECT OrderID, Amount
FROM Payments
WHERE PaymentMode = 'credit card';
```

Section 4: Key Takeaways

UNION simplifies complex queries by uniting filtered results from separate SELECTs with matching columns. Prefer UNION ALL if duplicates should remain. Practice on companyDB and eCommerceDB to verify compatibility and results. Use for readability in multi-table or multi-condition scenarios over single large JOINs or WHEREs.