# Lecture Notes – Set Operations In SQL

Section 1: Lecture Summary

Set operations in SQL include **union**, **union all**, **intersect**, and **except**. These operations combine results from two queries on the Customers table: one finding customers with Hotmail email addresses (CustomerIDs 4, 6, 13, 14) and another finding customers from Delhi (CustomerIDs 3, 6, 12, 15). Union combines results and removes duplicates (7 rows total, excluding duplicate CustomerID 6). Union all includes duplicates (8 rows). Intersect returns only common rows (CustomerID 6). Except removes common rows from the first query's result (3 rows from first query).

Section 2: Key Concepts and Explanations

Set operations require two queries with **union compatible** results, meaning the same number of columns and corresponding columns must have matching domains (same data type and semantic meaning, such as both being CustomerID as integers). Order of columns must match between queries. Database systems enforce matching column count and data types but allow different domains if types align (e.g., two integer columns like CustomerID and OrderID). Examples: SELECT CustomerID, Email, City works if both queries select these exact columns; removing City or changing order breaks compatibility.

Section 3: Example Code and Use Cases

Using eCommerceDB Customers table (CustomerID int, FirstName varchar, LastName varchar, Gender varchar, Email varchar, City varchar, JoinDate date):

Query 1: Customers with Hotmail accounts

```
SELECT CustomerID, Email, City
FROM Customers
WHERE Email LIKE '%hotmail.com';
```

Result: CustomerIDs 4, 6, 13, 14.

Query 2: Customers from Delhi

```
SELECT CustomerID, Email, City
FROM Customers
WHERE City = 'Delhi';
```

Result: CustomerIDs 3, 6, 12, 15.

Union (removes duplicates):

```
(SELECT CustomerID, Email, City FROM Customers WHERE Email LIKE
'%hotmail.com')
UNION
(SELECT CustomerID, Email, City FROM Customers WHERE City = 'Delhi');
```

Result: 7 rows (CustomerID 6 appears once).

Union all (keeps duplicates):

```
(SELECT CustomerID, Email, City FROM Customers WHERE Email LIKE
'%hotmail.com')
UNION ALL
(SELECT CustomerID, Email, City FROM Customers WHERE City = 'Delhi');
```

Result: 8 rows (CustomerID 6 appears twice).

Intersect (common rows):

```
(SELECT CustomerID, Email, City FROM Customers WHERE Email LIKE
'%hotmail.com')
INTERSECT
(SELECT CustomerID, Email, City FROM Customers WHERE City = 'Delhi');
```

Result: 1 row (CustomerID 6).

Except (first query minus common):

```
(SELECT CustomerID, Email, City FROM Customers WHERE Email LIKE
'%hotmail.com')
EXCEPT
(SELECT CustomerID, Email, City FROM Customers WHERE City = 'Delhi');
```

Result: 3 rows (CustomerIDs 4, 13, 14).

Section 4: Key Takeaways

Set operations (**union**, **union all**, **intersect**, **except**) combine two queries but require identical column count, order, and corresponding domains. Union eliminates duplicates; union all retains them. Intersect keeps only common rows; except removes common rows from the first query. Test compatibility by ensuring matching SELECT lists before applying operations.