Module 12

Using Set Operators

Zyead Ahmed.
Aspiring To Learning Data Engineer.

Module Overview

- Writing Queries with the UNION Operator
- Using EXCEPT and INTERSECT
- Using APPLY

Lesson 1: Writing Queries with the UNION Operator

- Interactions Between Sets
- Using the UNION Operator
- Using the UNION ALL Operator
- Demonstration: Using UNION and UNION ALL

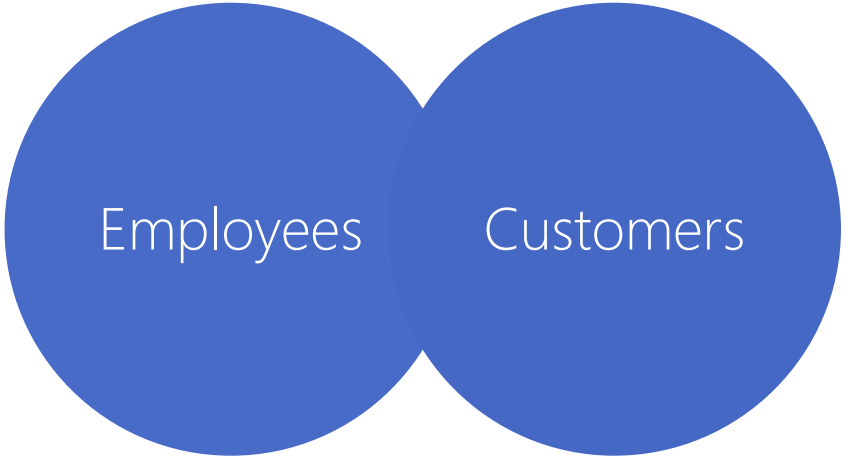
Interactions Between Sets

- The results of two input queries may be further manipulated
- Sets may be combined, compared, or operated against each other
- Both sets must have the same number of compatible columns
- ORDER BY not allowed in input queries, but may be used for result of set operation
- NULLs considered equal when comparing sets

```
<SELECT query_1>
<set_operator>
<SELECT query_2>
[ORDER BY <sort_list>]
```

Using the UNION Operator

- UNION returns a result set of distinct rows combined from both sides
- Duplicates removed during query processing (affects performance)




```
-- only distinct rows from both queries are returned
SELECT country, region, city FROM HR.Employees
UNION
SELECT country, region, city FROM Sales.Customers;
```

Using the UNION ALL Operator

- UNION ALL returns a result set with all rows from both sets
- To avoid performance penalty, use UNION ALL even if you know there are no duplicates

```
-- all rows from both queries will be returned
SELECT country, region, city FROM HR.Employees
UNION ALL
SELECT country, region, city FROM Sales.Customers;
```

Demonstration: Using UNION and UNION ALL

In this demonstration, you will see how to:

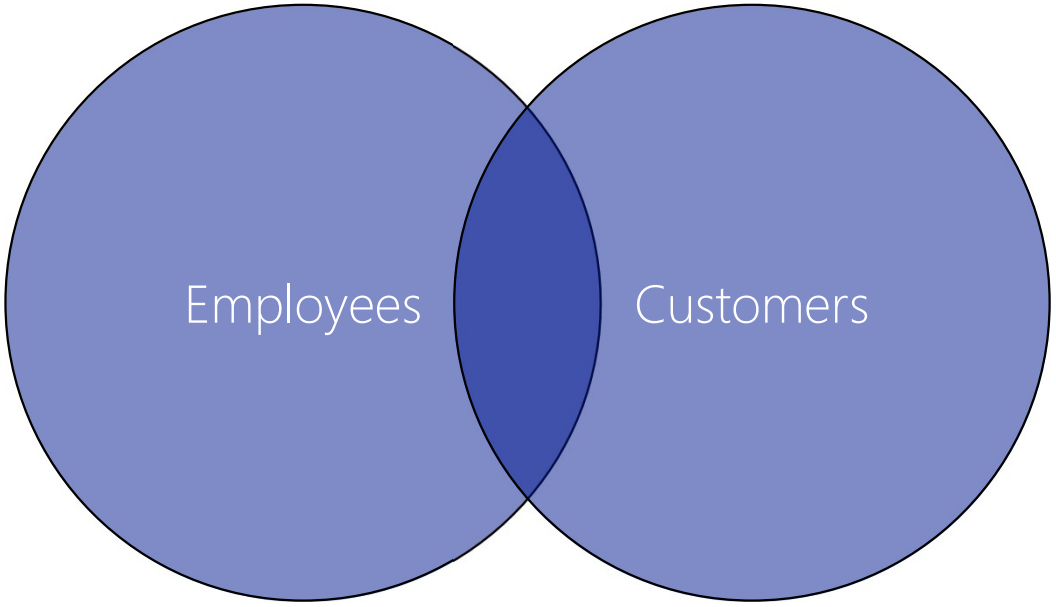
- Use UNION and UNION ALL

Lesson 2: Using EXCEPT and INTERSECT

- Using the INTERSECT Operator
- Using the EXCEPT Operator
- Demonstration: Using EXCEPT and INTERSECT

Using the INTERSECT Operator

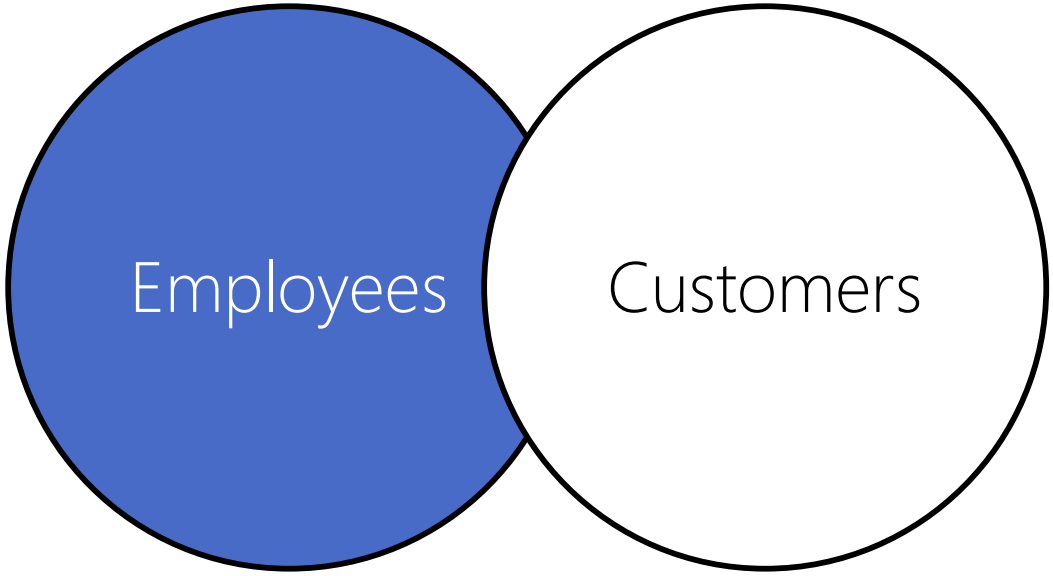
- INTERSECT returns only distinct rows that appear in both result sets




```
-- only rows that exist in both queries will be
-- returned
SELECT country, region, city FROM HR.Employees
INTERSECT
SELECT country, region, city FROM Sales.Customers;
```

Using the EXCEPT Operator

- EXCEPT returns only distinct rows that appear in the left set but not the right
  - Order in which sets are specified matters




```
-- only rows from Employees will be returned
SELECT country, region, city FROM HR.Employees
EXCEPT
SELECT country, region, city FROM Sales.Customers;
```

Demonstration: Using EXCEPT and INTERSECT

In this demonstration, you will see how to:

- Use INTERSECT and EXCEPT

Lesson 3: Using APPLY

- Using the APPLY Operator
- Using the CROSS APPLY Operator
- Using the OUTER APPLY Operator
- Demonstration: Using CROSS APPLY and OUTER APPLY

Using the APPLY Operator

- APPLY is a table operator used in the FROM clause
- Includes CROSS APPLY and OUTER APPLY
- Operates on two input tables, left and right
- Right table is often a derived table or a table-valued function

```
SELECT <column_list>
FROM        <left_table> AS <alias>
CROSS/OUTER APPLY
   <derived_table_expression or inline_TVF>
   AS <alias>
```

Using the CROSS APPLY Operator

- CROSS APPLY applies the right table expression to each row in left table
  - Conceptually similar to CROSS JOIN between two tables but can correlate data between sources

```
SELECT S.supplierid, s.companyname, P.productid,
        P.productname, P.unitprice
FROM Production.Suppliers AS S
CROSS APPLY
        dbo.fn_TopProductsByShipper(S.supplierid) AS P
```

Using the OUTER APPLY Operator

- Conceptually similar to LEFT OUTER JOIN between two tables
- OUTER APPLY adds a step to the processing used by CROSS APPLY:
  1. OUTER APPLY applies the right table expression to each row in left table
  2. OUTER APPLY adds rows for those with NULL in columns for right table

```
SELECT S.supplierid, s.companyname,
        P.productid, P.unitprice
FROM Production.Suppliers AS S
OUTER APPLY
        dbo.fn_TopProductsByShipper(S.supplierid) AS P
```

Demonstration: Using CROSS APPLY and OUTER APPLY

In this demonstration, you will see how to:

- Use APPLY