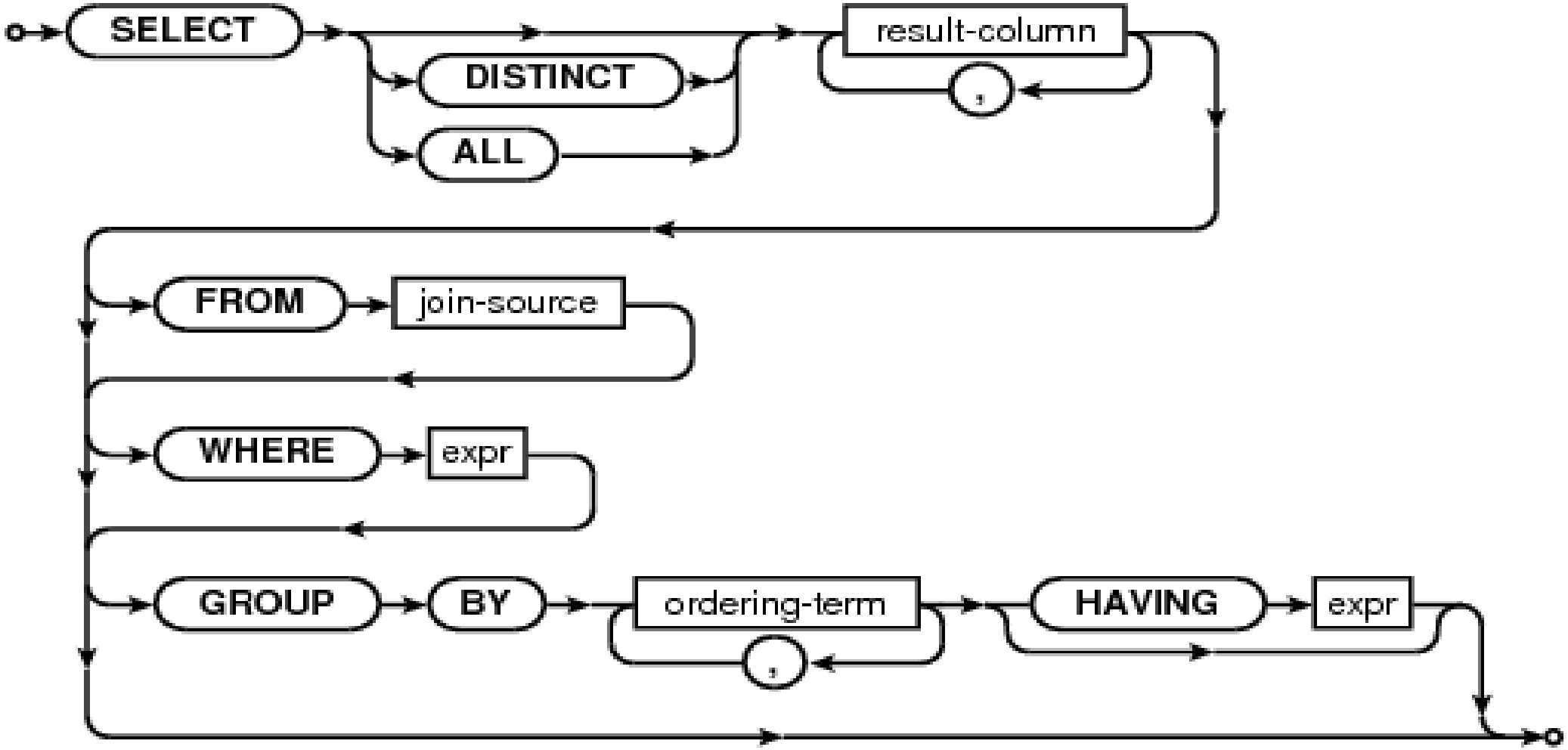
Data Warehouse

Sirojul Munir S.SI, M.KOM – Semester Genap TA 20182

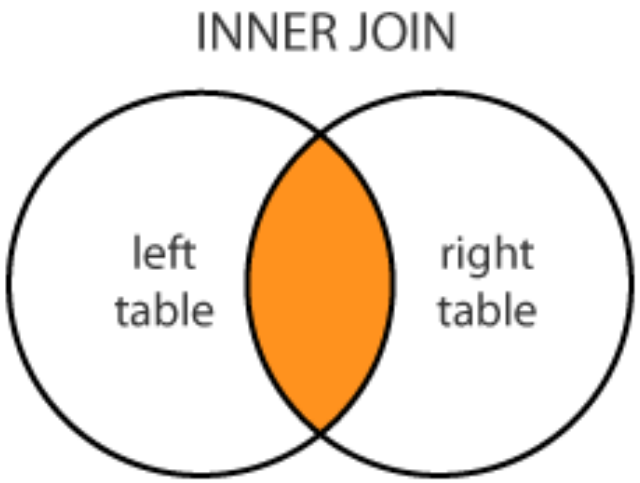
SQL for Data Warehouse (2)

SQL: SELECT Syntax



JOIN Tables

- A SQL JOIN combines records from two tables.
- A JOIN locates related column values in the two tables.
- A query can contain zero, one, or multiple JOIN operations.
- INNER JOIN is the same as JOIN; the keyword INNER is optional.



The SQL JOIN syntax

The general syntax is:

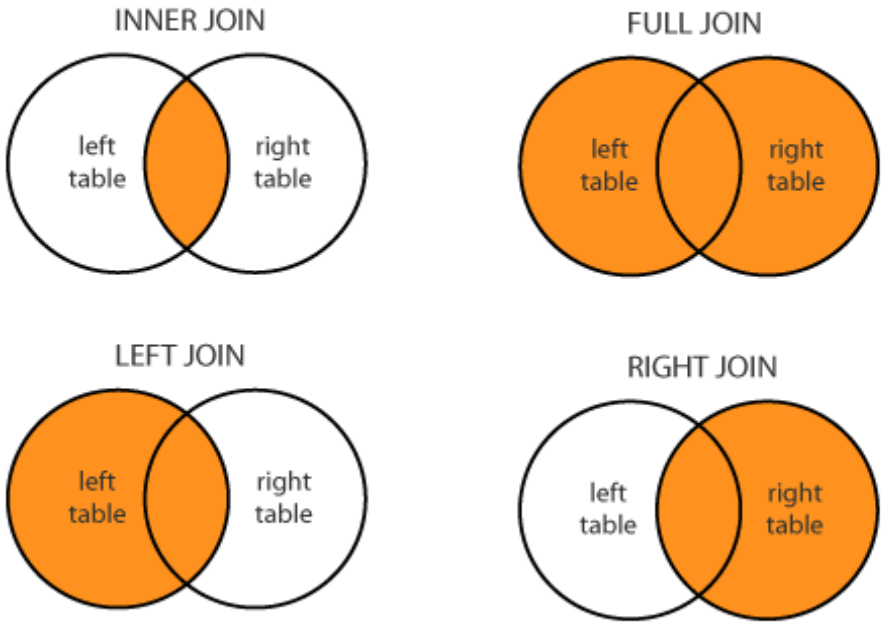
```
SELECT column-names
  FROM table-name1 JOIN table-name2
    ON column-name1 = column-name2
 WHERE condition
```

The general syntax with INNER is:

```
SELECT column-names
  FROM table-name1 INNER JOIN table-name2
    ON column-name1 = column-name2
 WHERE condition
```

JOIN Tables

- (INNER) JOIN: Select records that have matching values in both tables.
- LEFT (OUTER) JOIN: Select records from the first (left-most) table with matching right table records.
- RIGHT (OUTER) JOIN: Select records from the second (right-most) table with matching left table records.
- FULL (OUTER) JOIN: Selects all records that match either left or right table records.



All INNER and OUTER keywords are optional.

Details about the differences between these JOINs are available in subsequent tutorial pages.

The SQL JOIN syntax

The general syntax is:

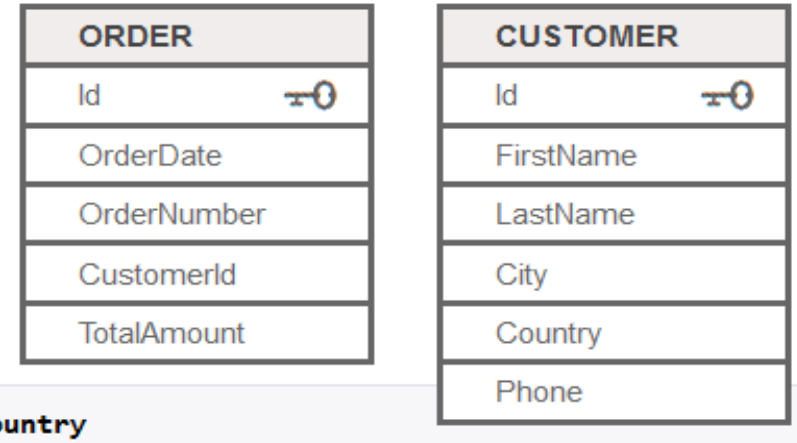
```
SELECT column-names
  FROM table-name1 JOIN table-name2
    ON column-name1 = column-name2
 WHERE condition
```

The general syntax with INNER is:

```
SELECT column-names
  FROM table-name1 INNER JOIN table-name2
    ON column-name1 = column-name2
 WHERE condition
```

JOIN Tables : 2 table

SQL JOIN Examples



Problem: List all orders with customer information

```
1.  SELECT OrderNumber, TotalAmount, FirstName, LastName, City, Country
2.    FROM [Order] JOIN Customer
3.      ON [Order].CustomerId = Customer.Id
```

In this example using table aliases for [Order] and Customer might have been useful.

Results: 830 records.

OrderNumber	TotalAmount	FirstName	LastName	City	Country
542378	440.00	Paul	Henriot	Reims	France
542379	1863.40	Karin	Josephs	Münster	Germany
542380	1813.00	Mario	Pontes	Rio de Janeiro	Brazil

JOIN Tables : 3 table

Problem: List all orders with product names, quantities, and prices

ORDER
Id
OrderDate
OrderNumber
CustomerId
TotalAmount

ORDERITEM
Id
OrderId
ProductId
UnitPrice
Quantity

PRODUCT
Id
ProductName
SupplierId
UnitPrice
Package
IsDiscontinued

```sql
SELECT O.OrderNumber, CONVERT(date,O.OrderDate) AS Date,
       P.ProductName, I.Quantity, I.UnitPrice
  FROM [Order] O
  JOIN OrderItem I ON O.Id = I.OrderId
  JOIN Product P ON P.Id = I.ProductId
ORDER BY O.OrderNumber
```

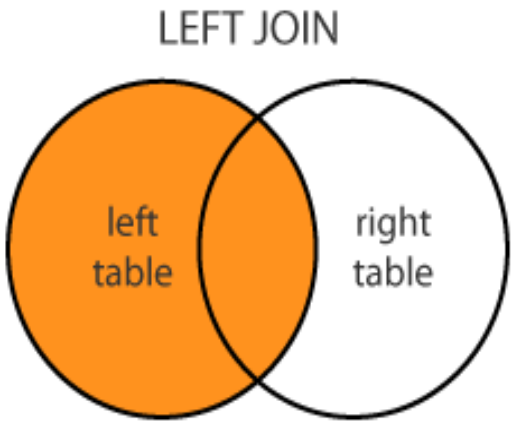
This query performs two JOIN operations with 3 tables.
The O, I, and P are table aliases. Date is a column alias.

Results: 2155 records

OrderNumber	Date	ProductName	Quantity	UnitPrice
542378	7/4/2012 12:00:00 AM	Queso Cabrales	12	14.00
542378	7/4/2012 12:00:00 AM	Singaporean Hokkien Fried Mee	10	9.80
542378	7/4/2012 12:00:00 AM	Mozzarella di Giovanni	5	34.80

JOIN Tables : LEFT JOIN

- LEFT JOIN performs a join starting with the first (left-most) table and then any matching second (right-most) table records.
- LEFT JOIN and LEFT OUTER JOIN are the same.



The general syntax is:

```
SELECT column-names
  FROM table-name1 LEFT JOIN table-name2
    ON column-name1 = column-name2
 WHERE condition
```

JOIN Tables : LEFT JOIN

The general LEFT OUTER JOIN syntax is:

```
1. SELECT OrderNumber, TotalAmount, FirstName, LastName, City, Country
2.   FROM Customer C LEFT JOIN [Order] O
3.     ON O.CustomerId = C.Id
4. ORDER BY TotalAmount
```

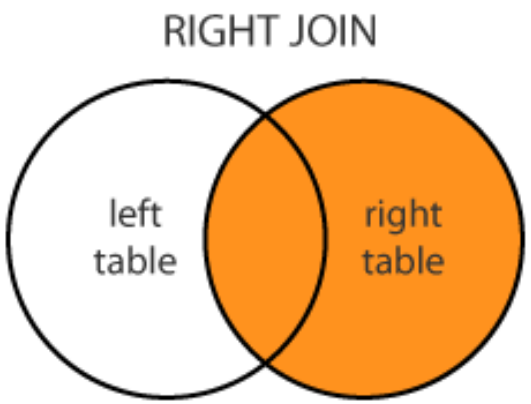
This will list all customers, whether they placed any order or not.
The ORDER BY TotalAmount shows the customers without orders first (i.e. TotalMount is NULL).

Results: 832 records

OrderNumber	TotalAmount	FirstName	LastName	City	Country
NULL	NULL	Diego	Roel	Madrid	Spain
NULL	NULL	Marie	Bertrand	Paris	France
542912	12.50	Patricio	Simpson	Buenos Aires	Argentina
542937	18.40	Paolo	Accorti	Torino	Italy
542897	28.00	Pascale	Cartrain	Charleroi	Belgium

JOIN Tables : RIGHT JOIN

- RIGHT JOIN performs a join starting with the second (right-most) table and then any matching first (left-most) table records.
- RIGHT JOIN and RIGHT OUTER JOIN are the same.



The general syntax is:

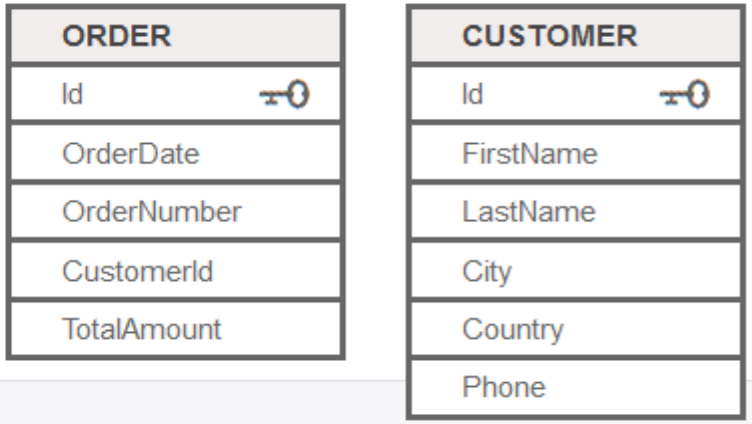
```
SELECT column-names
  FROM table-name1 RIGHT JOIN table-name2
    ON column-name1 = column-name2
 WHERE condition
```

The general RIGHT OUTER JOIN syntax is:

```
SELECT column-names
  FROM table-name1 RIGHT OUTER JOIN table-name2
    ON column-name1 = column-name2
 WHERE condition
```

JOIN Tables : RIGHT JOIN

SQL RIGHT JOIN Example



Problem: List customers that have not placed orders

```
1. SELECT TotalAmount, FirstName, LastName, City, Country
2.   FROM [Order] O RIGHT JOIN Customer C
3.     ON O.CustomerId = C.Id
4. WHERE TotalAmount IS NULL
```

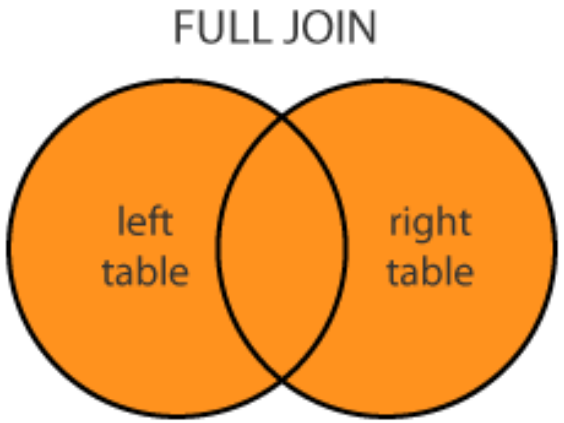
This returns customers that, when joined, have no matching order.

Results: 2 records

TotalAmount	FirstName	LastName	City	Country
NULL	Diego	Roel	Madrid	Spain
NULL	Marie	Bertrand	Paris	France

JOIN Tables : FULL JOIN

- FULL JOIN returns all matching records from both tables whether the other table matches or not.
- FULL JOIN can potentially return very large datasets.
- FULL JOIN and FULL OUTER JOIN are the same.



The general syntax is:

```
SELECT column-names
  FROM table-name1 FULL JOIN table-name2
    ON column-name1 = column-name2
 WHERE condition
```

The general FULL OUTER JOIN syntax is:

```
    SELECT column-names
  FROM table-name1 FULL OUTER JOIN table-name2
    ON column-name1 = column-name2
 WHERE condition
```

JOIN Tables : SELF JOIN

- A self JOIN occurs when a table takes a 'selfie'.
- A self JOIN is a regular join but the table is joined with itself.
- This can be useful when modeling hierarchies.
- They are also useful for comparisons within a table.

The general syntax is:

```
SELECT column-names
  FROM table-name T1 JOIN table-name T2
  WHERE condition
```

T1 and T2 are different table aliases for the same table

JOIN Tables : SELF JOIN

SQL Self JOIN Examples

CUSTOMER
Id
FirstName
LastName
City
Country
Phone

CUSTOMER
Id
FirstName
LastName
City
Country
Phone

Problem: Match customers that are from the same city and country

```
SELECT B.FirstName AS FirstName1, B.LastName AS LastName1,
       A.FirstName AS FirstName2, A.LastName AS LastName2,
       B.City, B.Country
  FROM Customer A, Customer B
 WHERE A.Id <> B.Id
   AND A.City = B.City
   AND A.Country = B.Country
 ORDER BY A.Country
```

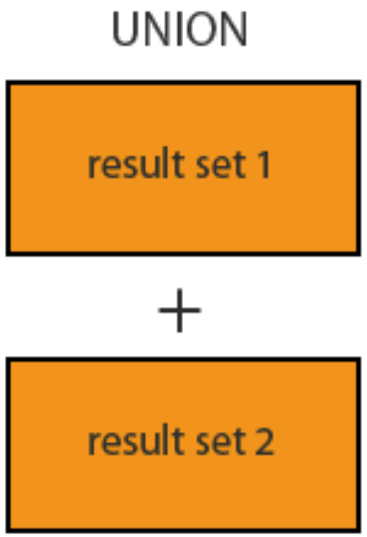
A and B are aliases for the same Customer table.

Results: 88 records

FirstName1	LastName1	FirstName2	LastName2	City	Country
Patricio	Simpson	Yvonne	Moncada	Buenos Aires	Argentina
Patricio	Simpson	Sergio	Gutiérrez	Buenos Aires	Argentina

JOIN Tables : UNION JOIN

- UNION combines the result sets of two queries.
- Column data types in the two queries must match.
- UNION combines by column position rather than column name.



The general syntax is:

```
SELECT column-names
  FROM table-name
 UNION
SELECT column-names
  FROM table-name
```

JOIN Tables : UNION JOIN

The general syntax is:

```
SELECT column-names
  FROM table-name
 UNION
SELECT column-names
  FROM table-name
```

SQL UNION Examples

CUSTOMER
Id
FirstName
LastName
City
Country
Phone

SUPPLIER
Id
CompanyName
ContactName
City
Country
Phone
Fax

Problem: List all contacts, i.e., suppliers and customers.

```
SELECT 'Customer' As Type,
       FirstName + ' ' + LastName AS ContactName,
       City, Country, Phone
  FROM Customer
UNION
SELECT 'Supplier',
       ContactName, City, Country, Phone
  FROM Supplier
```

This is a simple example in which the table alias would be useful

Results:

Type	ContactName	City	Country	Phone
Customer	Alejandra Camino	Madrid	Spain	(91) 745 6200
Customer	Alexander Feuer	Leipzig	Germany	0342-023176
Customer	Ana Trujillo	México D.F.	Mexico	(5) 555-4729

JOIN Tables : SUB QUERY

- A subquery is a SQL query within a query.
- Subqueries are nested queries that provide data to the enclosing query.
- Subqueries can return individual values or a list of records
- Subqueries must be enclosed with parenthesis

There is no general syntax; subqueries are regular queries placed inside parenthesis.
Subqueries can be used in different ways and at different locations inside a query:
Here is an subquery with the IN operator

```
SELECT column-names
  FROM table-name1
 WHERE value IN (SELECT column-name
                   FROM table-name2
                  WHERE condition)
```

Subqueries can also assign column values for each record:

```
SELECT column1 = (SELECT column-name FROM table-name WHERE condition),
       column-names
  FROM table-name
 WEHRE condition
```

JOIN Tables : SUB QUERY

SQL Subquery Examples

PRODUCT
Id
ProductName
SupplierId
UnitPrice
Package
IsDiscontinued

ORDERITEM
Id
OrderId
ProductId
UnitPrice
Quantity

Problem: List products with order quantities greater than 100.

```
SELECT ProductName
  FROM Product
 WHERE Id IN (SELECT ProductId
                FROM OrderItem
               WHERE Quantity > 100)
```

Results: 12 records

PoductName
Guaraná Fantástica
Schoggi Schokolade
Chartreuse verte

JOIN Tables : WHERE ANY, ALL

- ANY and ALL keywords are used with a WHERE or HAVING clause.
- ANY and ALL operate on subqueries that return multiple values.
- ANY returns true if any of the subquery values meet the condition.
- ALL returns true if all of the subquery values meet the condition.

The general ANY syntax is:

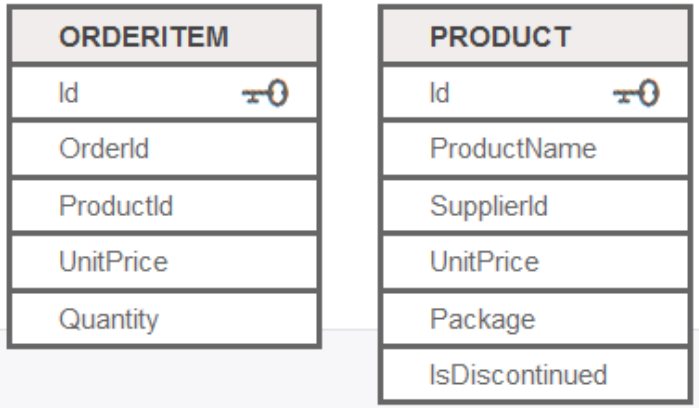
```
SELECT column-names
  FROM table-name
 WHERE column-name operator ANY
       (SELECT column-name
          FROM table-name
         WHERE condition)
```

The general ALL syntax is:

```
SELECT column-names
  FROM table-name
 WHERE column-name operator ALL
       (SELECT column-name
          FROM table-name
         WHERE condition)
```

JOIN Tables : WHERE ANY, ALL

SQL ANY Example



Problem: Which products were sold by the unit (i.e. quantity = 1)

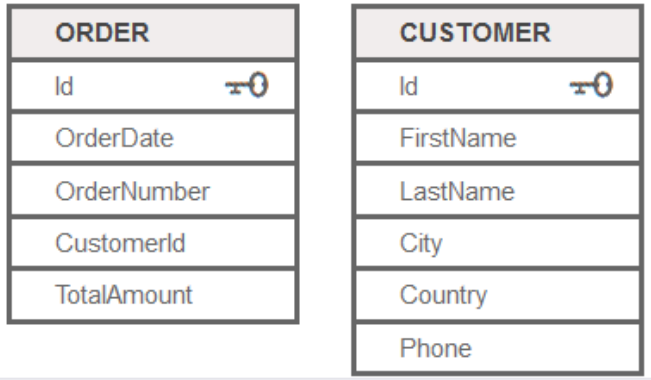
```
SELECT ProductName
  FROM Product
 WHERE Id = ANY
       (SELECT ProductId
          FROM OrderItem
         WHERE Quantity = 1)
```

Results: 17 records

ProductName
Chef Anton's Cajun Seasoning
Grandma's Boysenberry Spread
Uncle Bob's Organic Dried Pears

JOIN Tables : WHERE ANY, ALL

SQL ALL Example



Problem: List customers who placed orders that are larger than the average of each customer order

```
SELECT DISTINCT FirstName + ' ' + LastName as CustomerName
  FROM Customer, [Order]
 WHERE Customer.Id = [Order].CustomerId
   AND TotalAmount > ALL
       (SELECT AVG(TotalAmount)
          FROM [Order]
         GROUP BY CustomerId)
```

Results: 22 records

CustomerName
Art Braunschweiger
Christina Berglund
Elizabeth Lincoln

JOIN Tables : WHERE EXISTS

- WHERE EXISTS tests for the existence of any records in a subquery.
- EXISTS returns true if the subquery returns one or more records.
- EXISTS is commonly used with correlated subqueries.

The general syntax is:

```
SELECT column-names
  FROM table-name
 WHERE EXISTS
       (SELECT column-name
          FROM table-name
         WHERE condition)
```

JOIN Tables : WHERE EXIST

SQL EXISTS Example

Problem: Find suppliers with products over $100.

PRODUCT
Id
ProductName
SupplierId
UnitPrice
Package
IsDiscontinued

SUPPLIER
Id
CompanyName
ContactName
City
Country
Phone
Fax

```
SELECT CompanyName
  FROM Supplier
 WHERE EXISTS
       (SELECT ProductName
          FROM Product
         WHERE SupplierId = Supplier.Id
           AND UnitPrice > 100)
```

This is a **correlated subquery** because the subquery references the enclosing query (with Supplier.Id).

Results: 2 records

CompanyName
Plutzer Lebensmittelgroßmärkte AG
Aux joyeux ecclésiastiques

SELECT INTO

The general syntax is:

```
SELECT column-names
  INTO new-table-name
  FROM table-name
 WHERE EXISTS
       (SELECT column-name
          FROM table-name
         WHERE condition)
```

The new table will have column names as specified in the query.

SQL Server

```
CREATE TABLE new-table-name
    SELECT column-name
    FROM table-name
    WHERE condition
```

MySQL

SELECT INTO

SQL SELECT INTO Example

Problem: Copy all suppliers from USA to a new SupplierUSA table.

```
1.  SELECT * INTO SupplierUSA
2.    FROM Supplier
3.   WHERE Country = 'USA'
```

SQL Server

MySQL

CREATE TABLE SupplierUSA
SELECT * FROM Supplier WHERE Country = 'USA'

SUPPLIER
Id
CompanyName
ContactName
City
Country
Phone
Fax

Results: 4 rows affected

Here are the records in the newly created table SupplierUSA:

Id	CompanyName	ContactName	City	Country	Phone	Fax
2	New Orleans Cajun Delights	Shelley Burke	New Orleans	USA	(100) 555-4822	NULL
3	Grandma Kelly's Homestead	Regina Murphy	Ann Arbor	USA	(313) 555-5735	(313) 555-3349
16	Bigfoot Breweries	Cheryl Saylor	Bend	USA	(100) 555-4822	NULL
19	New England Seafood Cannery	Robb Merchant	Boston	USA	(617) 555-3267	(617) 555-3389

CHEAT SHEET : SQL

SQL CHEAT SHEET http://www.sqltutorial.org

QUERYING DATA FROM A TABLE

```
SELECT c1, c2 FROM t;
```
Query data in columns c1, c2 from a table

```
SELECT * FROM t;
```
Query all rows and columns from a table

```
SELECT c1, c2 FROM t
WHERE condition;
```
Query data and filter rows with a condition

```
SELECT DISTINCT c1 FROM t
WHERE condition;
```
Query distinct rows from a table

```
SELECT c1, c2 FROM t
ORDER BY c1 ASC [DESC];
```
Sort the result set in ascending or descending order

```
SELECT c1, c2 FROM t
ORDER BY c1
LIMIT n OFFSET offset;
```
Skip *offset* of rows and return the next n rows

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1;
```
Group rows using an aggregate function

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
HAVING condition;
```
Filter groups using HAVING clause

QUERYING FROM MULTIPLE TABLES

```
SELECT c1, c2
FROM t1
INNER JOIN t2 ON condition;
```
Inner join t1 and t2

```
SELECT c1, c2
FROM t1
LEFT JOIN t2 ON condition;
```
Left join t1 and t1

```
SELECT c1, c2
FROM t1
RIGHT JOIN t2 ON condition;
```
Right join t1 and t2

```
SELECT c1, c2
FROM t1
FULL OUTER JOIN t2 ON condition;
```
Perform full outer join

```
SELECT c1, c2
FROM t1
CROSS JOIN t2;
```
Produce a Cartesian product of rows in tables

```
SELECT c1, c2
FROM t1, t2;
```
Another way to perform cross join

```
SELECT c1, c2
FROM t1 A
INNER JOIN t2 B ON condition;
```
Join t1 to itself using INNER JOIN clause

USING SQL OPERATORS

```
SELECT c1, c2 FROM t1
UNION [ALL]
SELECT c1, c2 FROM t2;
```
Combine rows from two queries

```
SELECT c1, c2 FROM t1
INTERSECT
SELECT c1, c2 FROM t2;
```
Return the intersection of two queries

```
SELECT c1, c2 FROM t1
MINUS
SELECT c1, c2 FROM t2;
```
Subtract a result set from another result set

```
SELECT c1, c2 FROM t1
WHERE c1 [NOT] LIKE pattern;
```
Query rows using pattern matching %, _

```
SELECT c1, c2 FROM t
WHERE c1 [NOT] IN value_list;
```
Query rows in a list

```
SELECT c1, c2 FROM t
WHERE c1 BETWEEN low AND high;
```
Query rows between two values

```
SELECT c1, c2 FROM t
WHERE c1 IS [NOT] NULL;
```
Check if values in a table is NULL or not