



INTRO TO SQL SESSION 2 – SEP 7TH

INSTRUCTOR: SYLVIA VARGAS SQLSYLVIA@GMAIL.COM

LINKEDIN: [HTTPS://WWW.LINKEDIN.COM/IN/SYLVIAVARGAS/](https://www.linkedin.com/in/sylviaavargas/)

BLOG: [HTTPS://SYLVIAVARGAS.COM/](https://sylviaavargas.com/)

BLOG (IN PROGRESS): [HTTP://SHESATECHIE.ORG/](http://shesatechie.org/)

ALL SLIDES ON [GITHUB](https://github.com/SQLSYLVIA/GDI-SQL): [HTTPS://GITHUB.COM/SQLSYLVIA/GDI-SQL](https://github.com/SQLSYLVIA/GDI-SQL)

GDI'S INTRO TO SQL

- This class will focus on understanding how to use SQL and its basic syntax.
- Topics in this course will include: querying and aggregating data in individual tables, querying multiple related tables (JOINS), and writing subqueries.
- In this course, online SQL database tools will be used to teach students how to write queries. No installation is required.

PLAN FOR WEEK 1

- What's a database? What is SQL?

- Basics of how to query a database?

- 1. BASIC SELECT Statement
 - SELECT and FROM commands
 - COUNT command
- 2. ORDER BY clause
- 3. WHERE clause
- 4. LIKE clause
- 5. GROUP BY and HAVING
- 4. JOINS – Inner, Left and Right outer joins

PRACTICE

Let's develop it!

REVIEW

W3SCHOOLS DATABASE:

[HTTPS://WWW.W3SCHOOLS.COM/SQL/TRYSQL.ASP?FILENAME=TRYSQL_EDITOR](https://www.w3schools.com/sql/trysql.asp?filename=try_sql_editor)

Body of a SQL statement:

SQL clauses and required order

1 SELECT	selects variables
2 FROM	opens datasets/table
3 WHERE	restricts observations
4 GROUP BY	groups observations
5 HAVING	restricts groups
6 ORDER BY	sorts results



SQL SELECT

Syntax	Example
SELECT * FROM <Table>	SELECT * FROM Customers

SQL ORDER BY

Syntax	Example
<pre>SELECT * FROM <Table></pre>	<pre>SELECT * FROM Customers ORDER BY CustomerName</pre>

SQL WHERE

Syntax	Example
<pre>SELECT * from <Table> WHERE <column> = 'Value'</pre>	<pre>SELECT * FROM Customers WHERE Country = 'Mexico'</pre>



SQL LIKE

Syntax	Example
<pre>SELECT * FROM <Table> WHERE <column> = 'Value'</pre>	<pre>SELECT * FROM Customers WHERE Country = 'Mexico'</pre>



SQL DISTINCT

Syntax	Example
<pre>SELECT DISTINCT <column> FROM <Table></pre>	<pre>SELECT DISTINCT Country FROM Customers</pre>


SQL COUNT

Syntax

```
SELECT  
COUNT(*)  
FROM  
<Table>
```

Example

```
SELECT  
COUNT(*)  
Country  
FROM  
Customers
```



SQL OPERATORS: AND, OR, NOT

SQL Operators are used within the WHERE Statement and CASE Statement.

Let's focus on WHERE Statement

```
SELECT * FROM Customers  
        WHERE Country = "Argentina" OR  
        Country = "Venezuela"
```

```
SELECT * FROM Customers  
        WHERE Country = "Venezuela" AND  
        PostalCode = 5022
```

Practicing

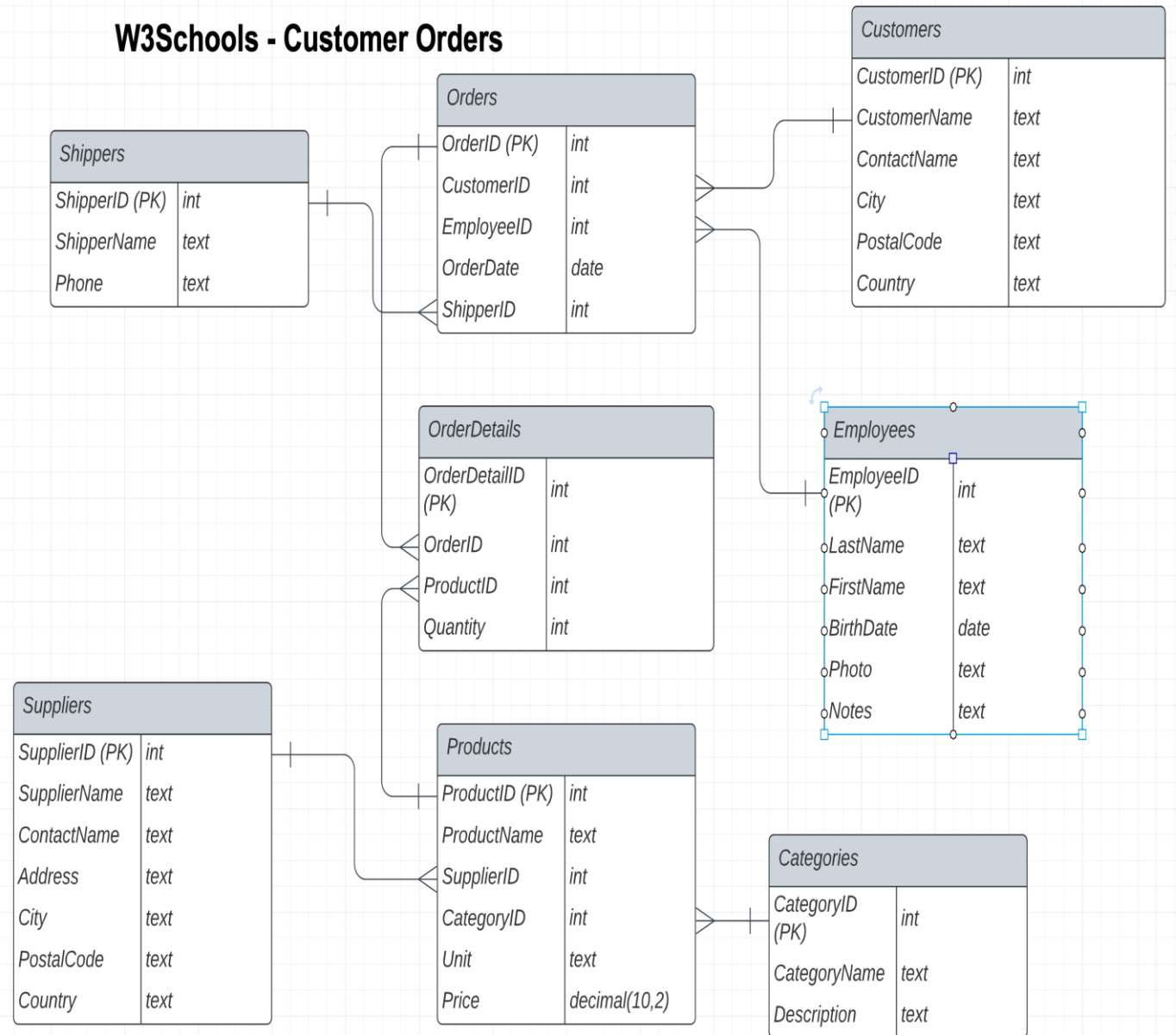
ONLINE SQL TOOLS

- W3Schools has a SQL database to query -
https://www.w3schools.com/sql/trysql.asp?filename=trysql_editor
- DB Fiddle is a site used by companies to assess SQL skills, but it is also an easy way to learning to using SQL as well as create tables.
<https://www.db-fiddle.com/f/6eXpPSRFQgzdKUCjSsbSF4/9>
- REPLIT.com is another way to learn SQL using SQLite.
I have created some examples for you to learn with.
<https://replit.com/@sqlSylvia/GDI-SQLIntro>

W3 SCHOOLS – DATA BASE

- In your browser open the following url to work with a SQL database in W3Schools.
- https://www.w3schools.com/sql/trysql.asp?filename=trysql_editor

W3Schools - Customer Orders



SQL Statement:

```
SELECT Country, COUNT(CustomerID) as CustomerCount
FROM Customers
GROUP BY Country;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 21

Country	CustomerCount
Argentina	3
Austria	2
Belgium	2
Brazil	9
Canada	3
Denmark	2

SQL GROUP BY HOW TO AGGREGATE DATA

The main purpose of grouping the records of a table based on particular columns is to perform calculations on these groups.

The GROUP BY clause is typically used with aggregate functions such as SUM(), COUNT(), AVG(), MAX(), or MIN() etc.

ANOTHER GROUP BY EXAMPLE

This example shows multiple aggregate functions in one SQL Statement as well as the use of the AS to rename a column.

SQL Statement:

[Get your own SQL server](#)

```
SELECT CustomerID, MIN(OrderDate), Max(OrderDate)
      , COUNT(OrderID) as NumOfOrders
FROM [Orders]
GROUP BY CustomerID
```

Edit the SQL Statement, and click "Run SQL" to see the result.

[Run SQL »](#)

Result:

Number of Records: 74

CustomerID	MIN(OrderDate)	Max(OrderDate)	NumOfOrders
2	1996-09-18	1996-09-18	1
3	1996-11-27	1996-11-27	1
4	1996-11-15	1996-12-16	2
5	1996-08-12	1996-12-16	3
7	1996-07-25	1997-02-05	4
8	1996-10-10	1996-10-10	1
9	1996-10-16	1996-11-25	3
10	1996-12-20	1997-01-30	4
11	1996-08-26	1996-08-26	1
13	1996-07-18	1996-07-18	1



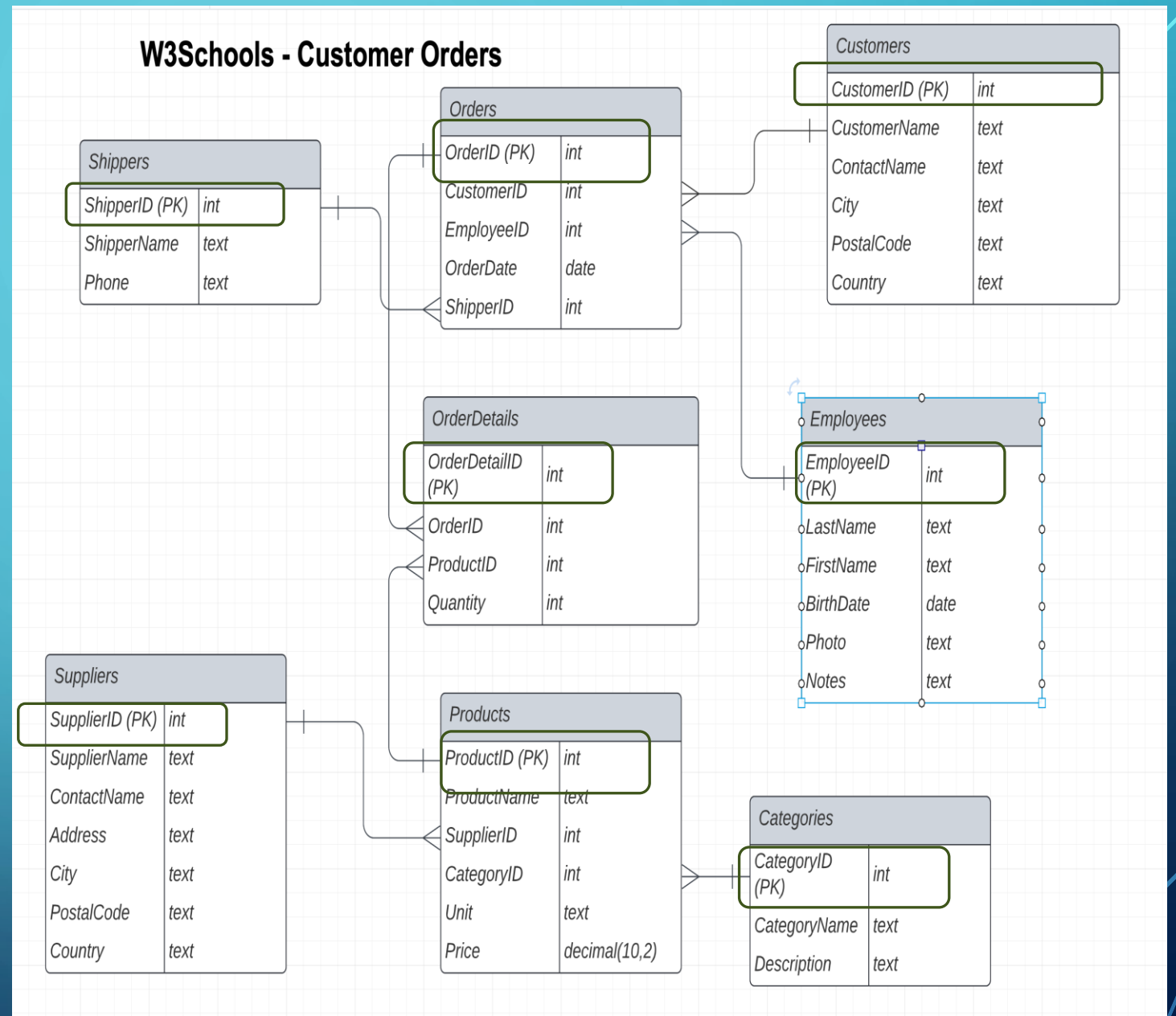
PRIMARY KEYS, FOREIGN KEY AND JOINS

SQL Join statement is used to combine data or rows from two or more tables based on a common field between them commonly known as a **Foreign Keys to Primary Keys** to a table.

UNDERSTANDING THE DATA MODEL

- **Primary Keys –**

A primary key is the column or columns that contain values that uniquely identify each row in a table. A database table must have a primary key for Optim to insert, update, restore, or delete data from a database table.



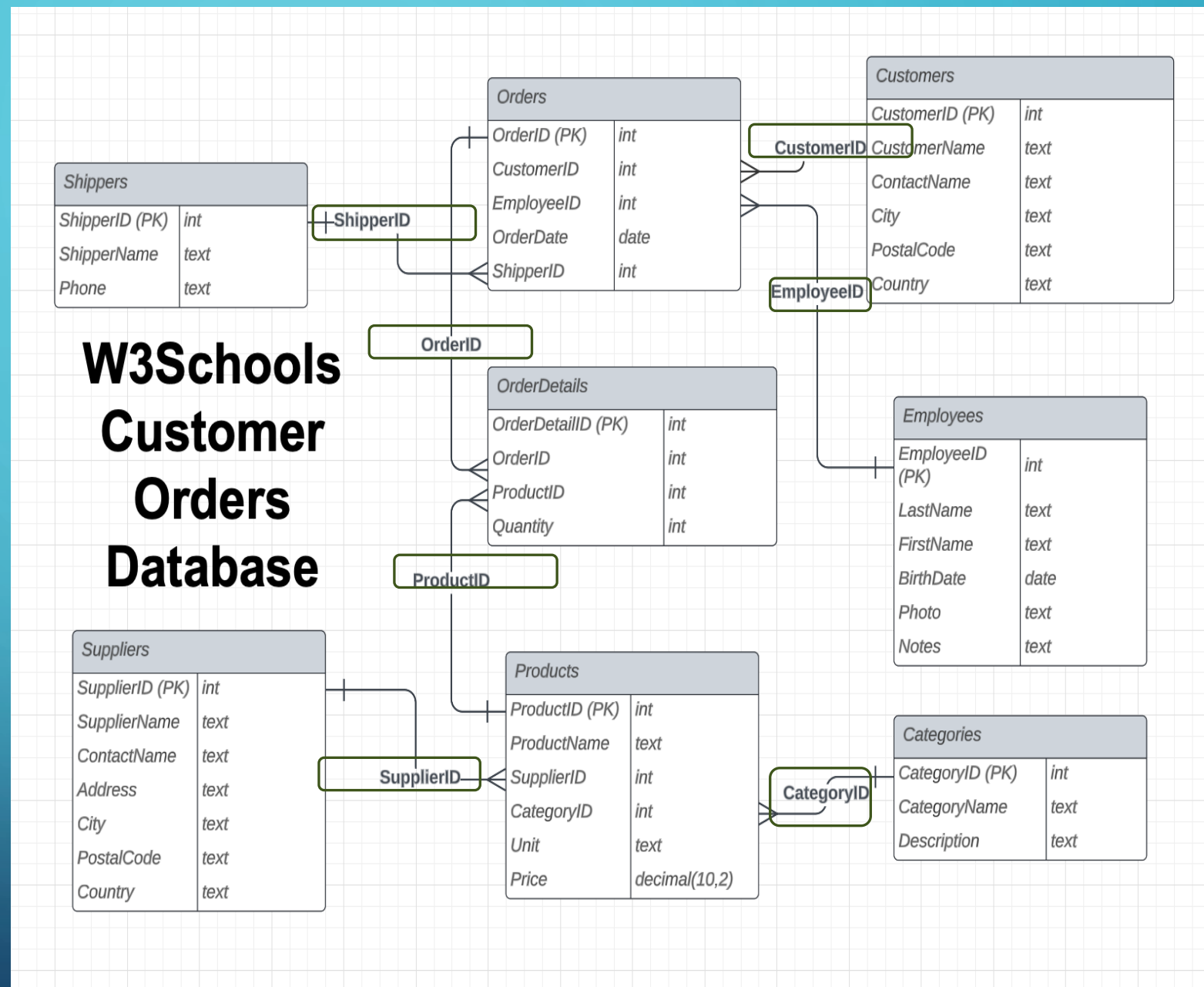
UNDERSTANDING THE DATA MODEL

- **Foreign Keys**

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table. The table with the foreign key is called the *child* table, and the table with the primary key is called the *referenced* or *parent* table.

- Note the Foreign Keys for

- CategoryID
- CustomerID
- EmployeeID
- OrderID
- ProductID
- ShipperID
- SupplierID



INNER JOIN EXAMPLE

- `SELECT ORDERID ,
SUM(OrderDetails.Quantity *
Products.Price)
FROM [OrderDetails]
JOIN [Products]
ON OrderDetails.ProductID =
Products.ProductID
GROUP BY ORDERID`

SQL Statement:

```
SELECT ORDERID, SUM(OrderDetails.Quantity * Products.Price)
FROM [OrderDetails]
JOIN [Products]
    ON OrderDetails.ProductID = Products.ProductID
GROUP BY ORDERID
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

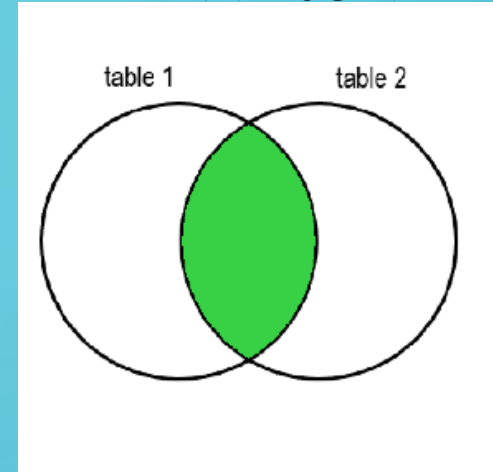
Number of Records: 196

OrderID	SUM(OrderDetails.Quantity * Products.Price)
10248	566
10249	2329.25
10250	2267.25
10251	839.5
10252	4662.5
10253	1806
10254	781.5
10255	3115.75
10256	648
10257	1400.5

JOIN REVIEW

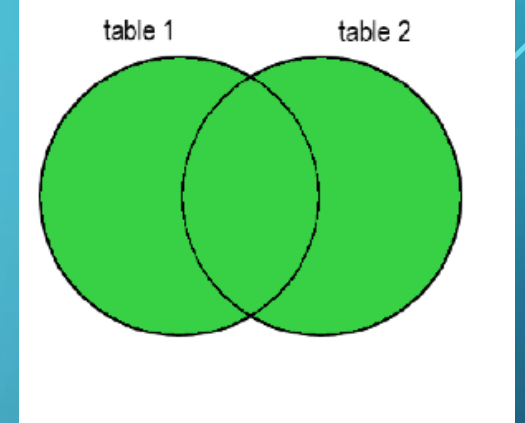
- These VENN DIAGRAMS illustrate the output of various JOIN commands.

INNER JOIN

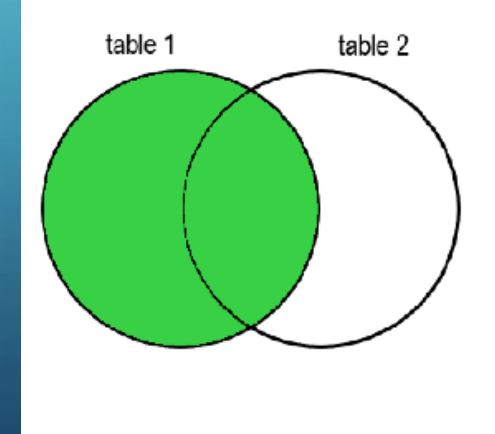


FULL OUTER JOIN

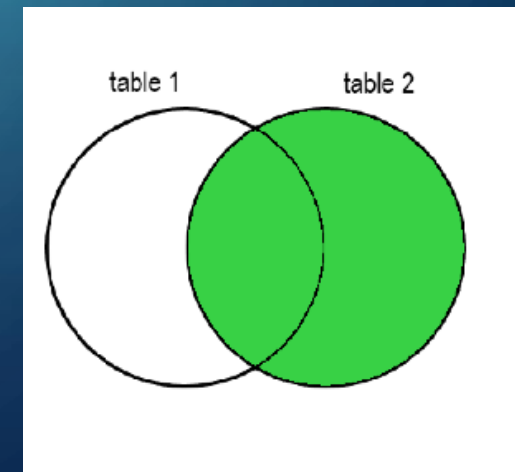
CARTESIAN PRODUCT



LEFT JOIN



RIGHT JOIN

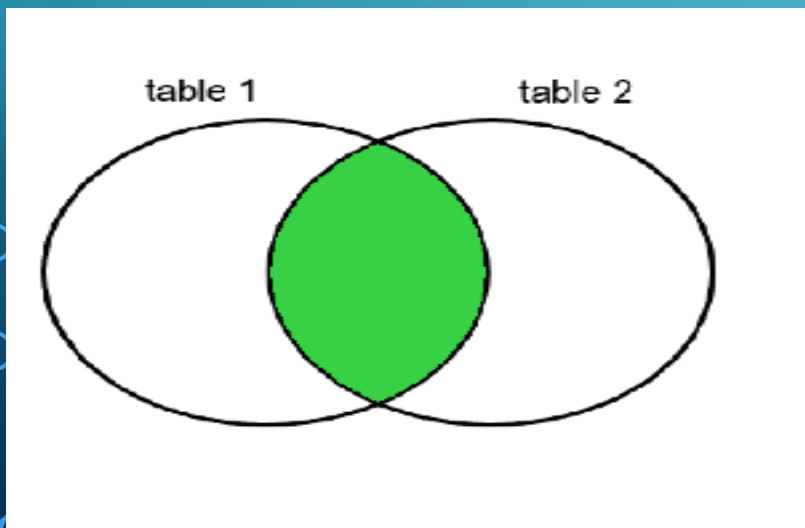


JOIN – JOINING TABLES

- A JOIN combines columns from one or more tables to a new table.
 - INNER JOIN – returns each row in the two joined tables that have matching column values
 - LEFT OUTER JOIN - The result of a **left outer join** (or simply **left join**) for tables A and B always contains all rows of the "left" table (A), even if the join-condition does not find any matching row in the "right" table (B).
 - RIGHT OUTER JOIN - A **right outer join** (or **right join**) closely resembles a left outer join, except with the treatment of the tables reversed.
 - FULL OUTER JOIN - combines the effect of applying both left and right outer joins. Where rows in the FULL OUTER JOINed tables do not match, the result set will have NULL values for every column of the table that lacks a matching row. For those rows that do match, a single row will be produced in the result set (containing columns populated from both tables).
 - CROSS JOIN – returns the Cartesian product of rows from tables in the join.
In other words, it will produce rows that combine each row from the first table with each row from the second table.

INNER JOIN

The inner join will keep only the information from the two joined tables that is related



```
SELECT Employees.LastName
```

```
, Orders.orderID
```

```
FROM Orders
```

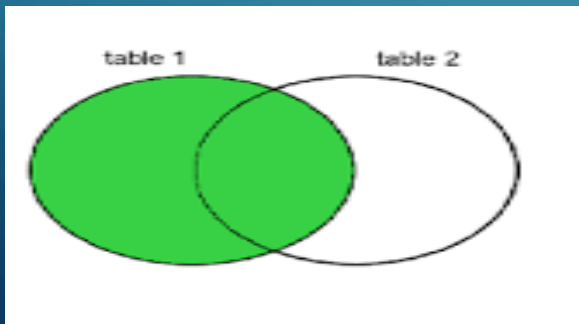
```
JOIN Employees
```

```
ON Employees.EmployeeID =
```

```
Orders.EmployeeID
```

LEFT JOIN = LEFT OUTER JOIN

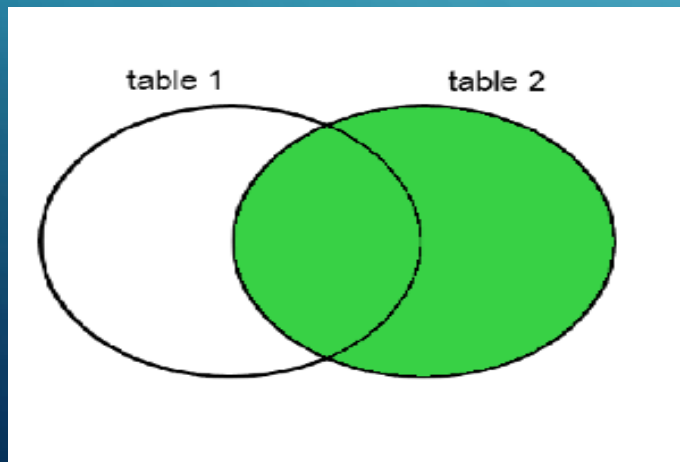
A LEFT JOIN returns all the rows from the left table and the matching rows from the right table. If there is no match in the right table, the result will contain NULL values.



```
SELECT Employees.LastName  
, Orders.OrderID  
FROM Employees  
LEFT JOIN Orders  
ON orders.EmployeeID =  
Employees.EmployeeID ;
```

RIGHT JOIN = RIGHT OUTER JOIN

A RIGHT JOIN returns all the rows from the right table and the matching rows from the left table. If there is no match in the left table, the result will contain NULL values.



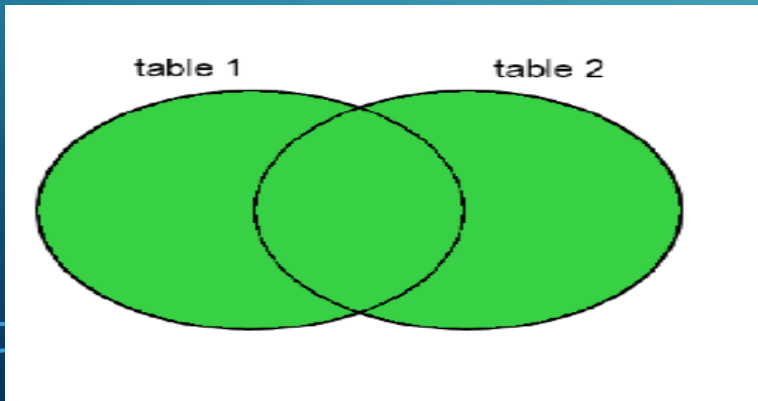
```
SELECT Employees.LastName  
       , Orders.OrderID
```

```
FROM Orders
```

```
RIGHT JOIN Employees
```

```
ON orders.EmployeeID =  
   Employees.EmployeeID ;
```

FULL OUTER JOIN =
CARTESIAN PRODUCT
OUTER....



SQL Statement:

```
SELECT Orders.customerID, OrderDetails.ProductID  
FROM Orders  
JOIN OrderDetails
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 101528

CustomerID	ProductID
90	11
90	42
90	72



JOINS: OUTER LEFT/RIGHT

```
SELECT Customers.CustomerName,  
count(Orders.OrderID)
```

```
FROM [Customers]
```

```
LEFT JOIN [Orders] ON Orders.CustomerID  
= Customers.CustomerID
```

```
GROUP BY Customers.CustomerName
```

ACTION: Re-write this statement using RIGHT
JOIN

WEEK 1: PRACTICE ASSIGNMENT

- Using the W3Schools Database answer the following questions:
 1. Which customer has the most number of Orders?
 2. Which customer has spent the most money in Orders?
 3. Are there any employees that went to Boston College? Who?
 4. How many products are there by category? List out CategoryName, and number of products?
 5. How many orders used “Speedy Express” Shipper?

What other insights can you provide?

LEARNING AND PRACTICING: SQL RESOURCES

- Online Databases for learning
 - https://www.w3schools.com/sql/trysql.asp?filename=trysql_editor
 - The following 2 databases are the same schema, but in.
 - <https://www.db-fiddle.com/f/6eXpPSRFQgzdKUCjSsbSF4/9> – No login required but you need to save your work elsewhere for backup.
 - <https://replit.com/@sqlSylvia/GDI-SQLIntro>. - replit you need an account to save your work
- SQL Reference information
 - <https://www.w3schools.com/sql/default.asp>
 - <https://www.tutorialspoint.com/sql/index.htm>