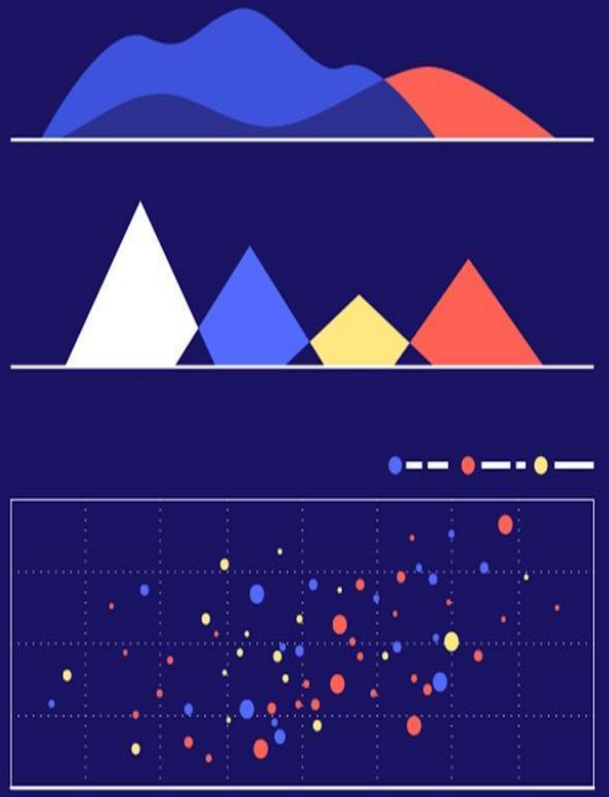
BECOME A

DATA ANALYST

3 WEEKEND BOOTCAMP WITH EXPERTS



LEARN THE FUNDAMENTALS OF THE BEST TOOLS

python

tableau SOFTWARE

SQL

MEX.4.INDUSTRY

BECOME A

DATA ANALYST

3 WEEKEND BOOTCAMP WITH EXPERTS

CONOCE A NUESTROS EXPERTOS

Marcela Álvarez

accenture

Azure Data Engineer

Forma parte del equipo de AI y Analytics de Avanade (Joint venture entre Microsoft y Accenture). Trabaja como Azure Data Engineer y se encarga de diseñar e implementar soluciones de Big Data y Analítica a clientes nacionales e internacionales.

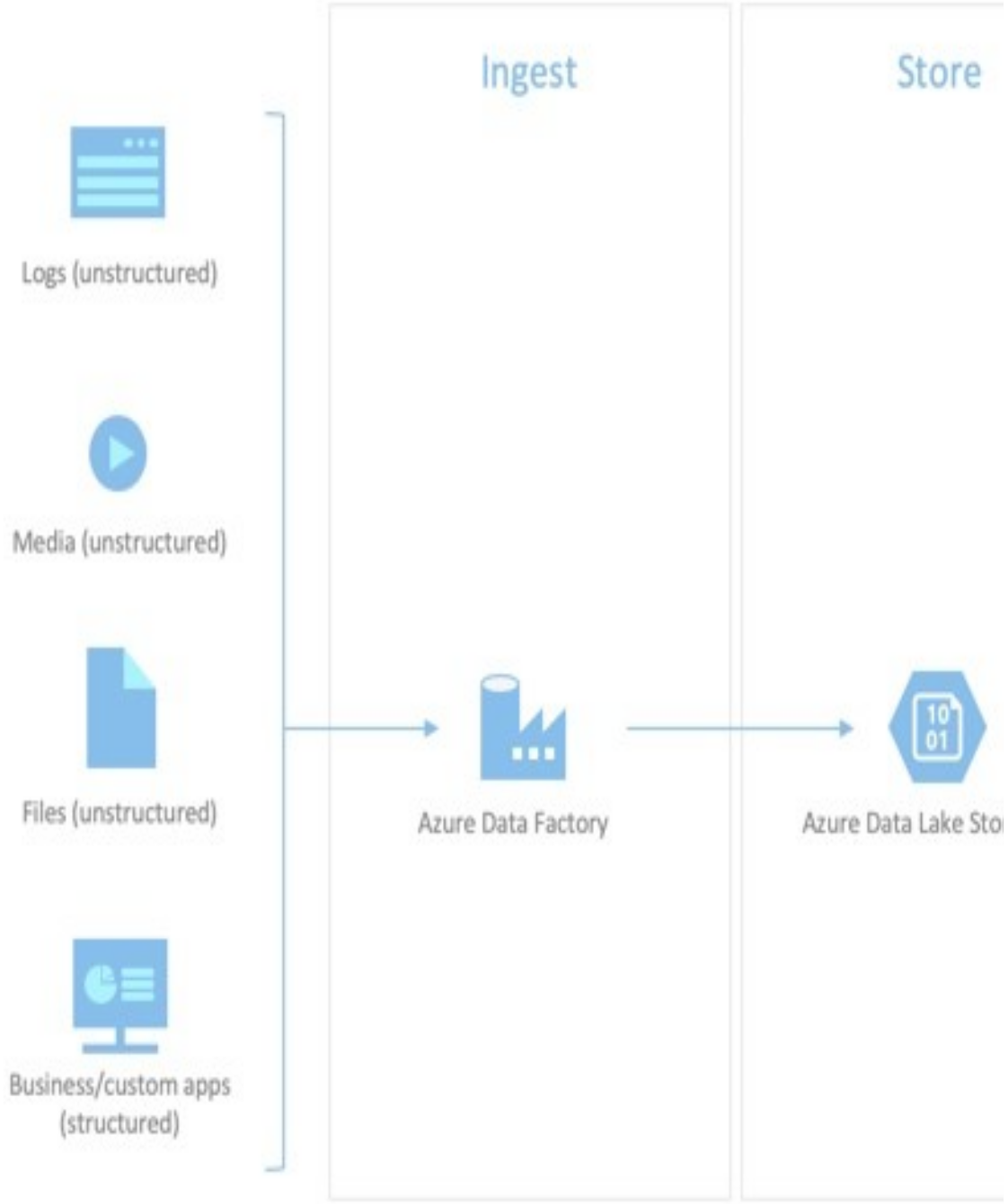
- Azure Data Factory
- Azure HDInsight
- Azure SQL Server
- Azure Storage Account
- Azure Data Lake

Certificación 70-475 de Microsoft

MEX.4.INDUSTRY

Microsoft Azure

SQL



MEX.4.INDUSTRY

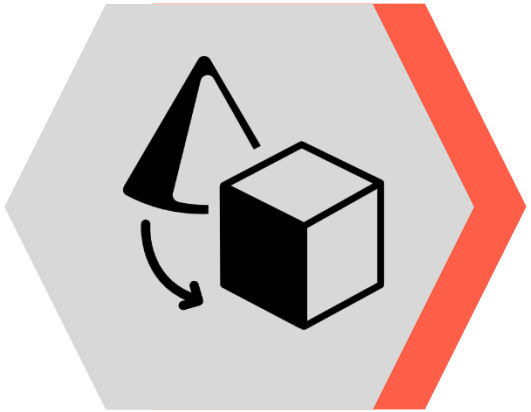
SQL Bootcamp

Contacto:
Marcela Álvarez
marcela.alvarez.rdz@gmail.com

AGENDA

MEX.4.INDUSTRY

01 ¿Qué es SQL?



02 Basic Queries

03 Advanced Queries

04 Aggregate Functions and Constraints

What Does SQL stands for?

MEX.4.INDUSTRY

1. SQL stands for Structured Query Language.

2. SQL lets you access and manipulate databases.

3. Retrieve data, insert, update, and delete records. Create databases, tables, stored procedures, views and permissions.

RDBMS

01

Stands for Relational Database Management System

02

Basis for SQL and for modern databse systems

03

Data is stored in database objects called tables.

04

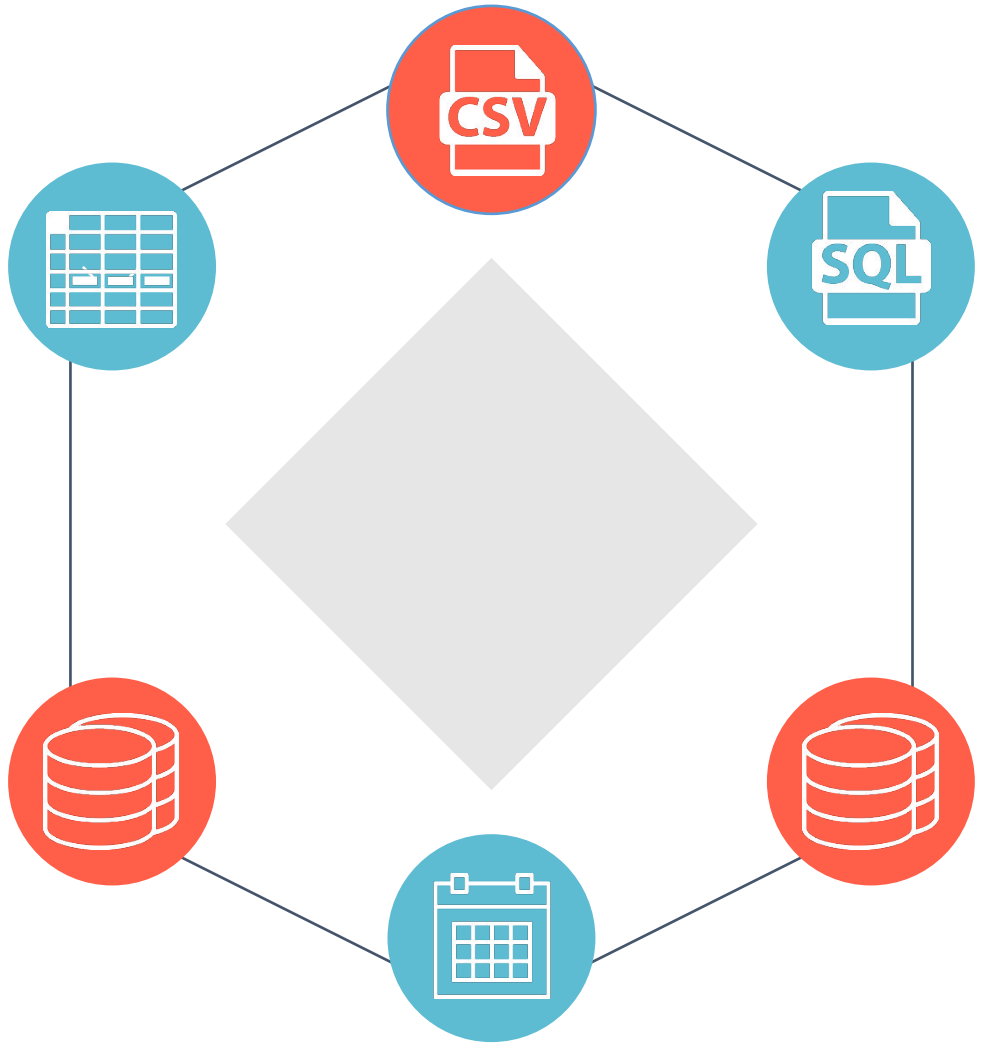
A table is a collection of related data entries, consisting of columns and rows.

05

Every table contains fields. Fields are columns in a table.

06

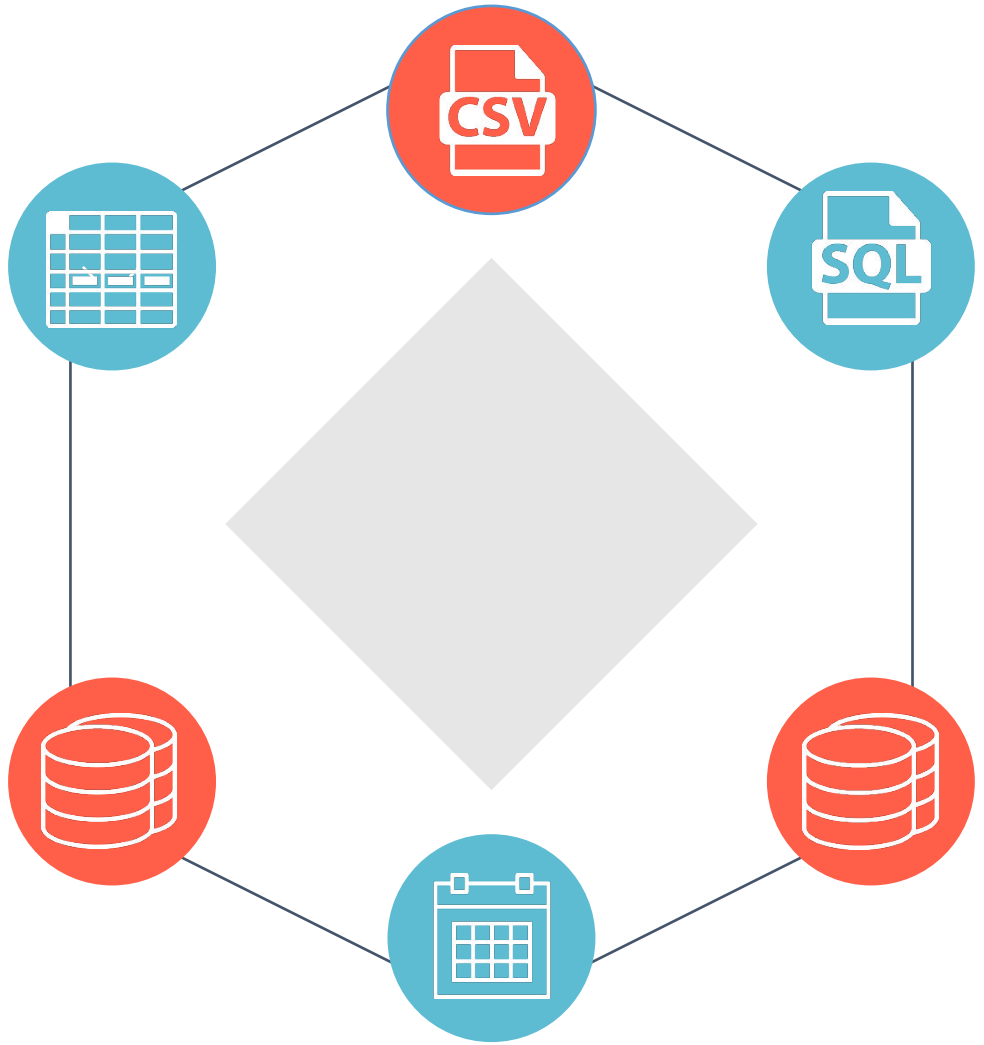
Tables also contain rows. Records are horizontal entities.



RDBMS

Results | Messages

	EmployeeId	FirstName	LastName	DepartmentName
1	1	Ken	Sanchez	Executive
2	2	Terri	Duffy	Engineering
3	3	Roberto	Tamburello	Engineering
4	4	Rob	Walters	Engineering
5	5	Gail	Erickson	Engineering
6	6	Jossef	Goldberg	Engineering
7	7	Dylan	Miller	Support
8	8	Diane	Margheim	Support
9	9	Gigi	Matthew	Support
10	10	Michael	Raheem	Support



Relational Database Concepts

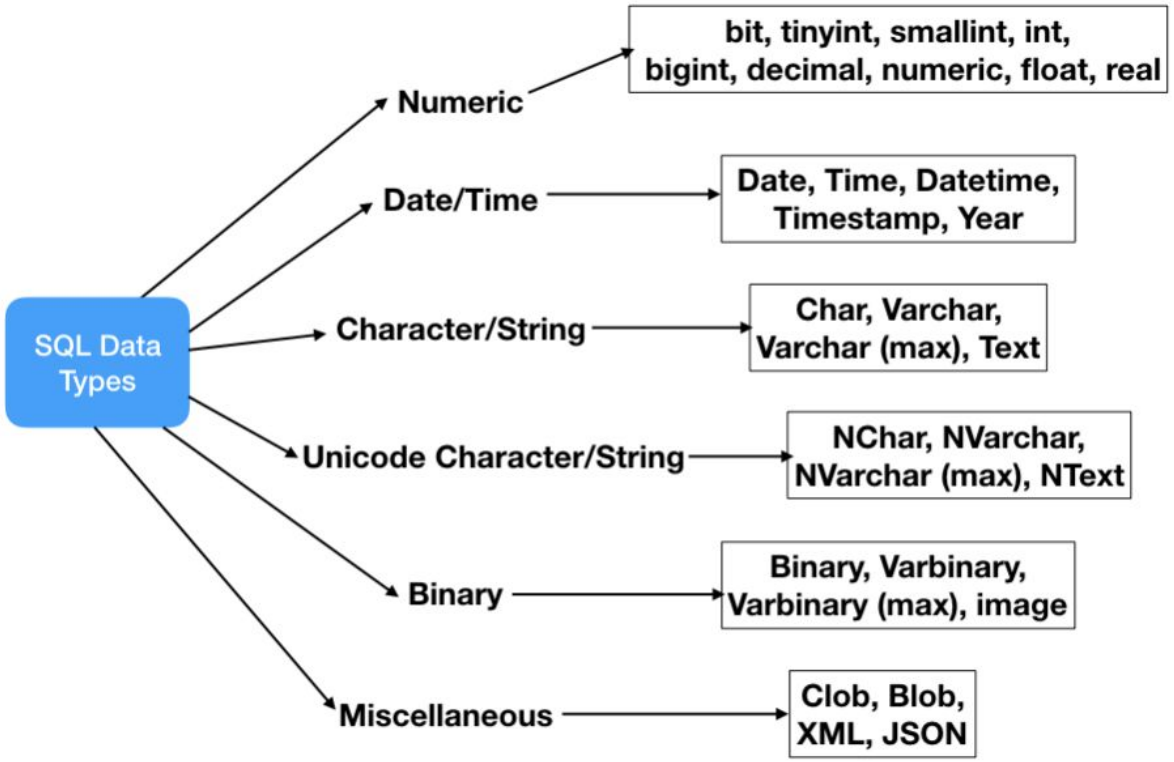
MEX.4.INDUSTRY

1. **Relationships:** A link between two tables.

2. **Datatypes:** Specifies the type of data that can exist in a specific column.

3. **Primary Keys:** Column that is used to identify records. For example in table Employee, EmployeeID is unique.

4. **Foreign Keys:** Columns that link to primary key columns in other tables creating a relationship.



Popular Databases

Oracle

SQL Server

MySQL

PostgreSQL

DML and DDL

Data Manipulation Language

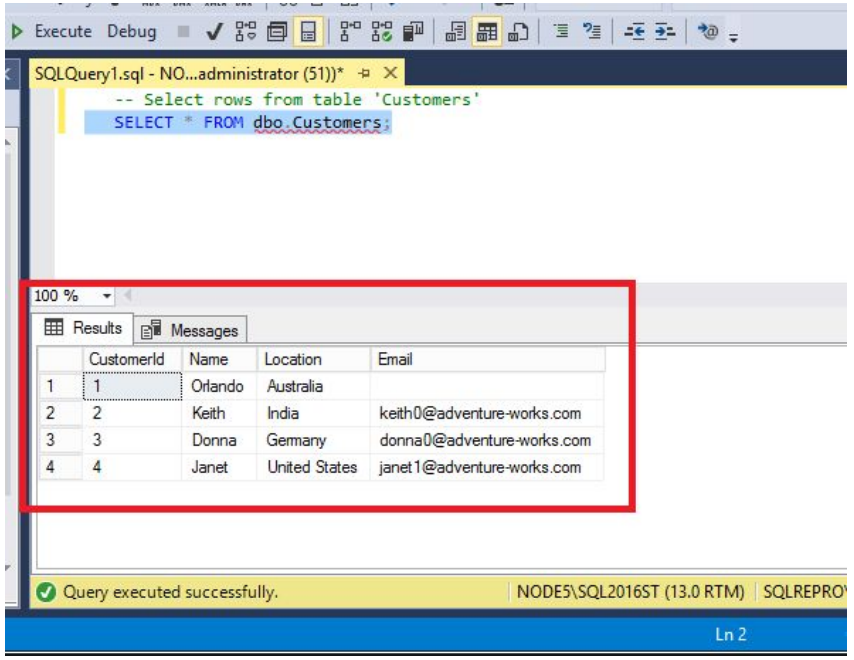
- Statements used to work with data in an existing database.
  - SELECT
  - INSERT
  - UPDATE
  - DELETE

Data Definition Language

- Used to structure objects in a database.
  - CREATE
  - ALTER
  - DROP

Some of the most important SQL commands

SELECT - extracts data from a database
UPDATE - updates data in a database
DELETE - deletes data from a database
INSERT INTO - inserts new data into a database
CREATE DATABASE - creates a new database
ALTER DATABASE - modifies a database
CREATE TABLE - creates a new table
ALTER TABLE - modifies a table
DROP TABLE - deletes a table
CREATE INDEX - creates an index (search key)
DROP INDEX - deletes an index



SQL Database commands

SQL Create DB

Statement used to create a new SQL database.

Example:

```
CREATE DATABASE testDB;
```

SQL Drop DB

Statement used to drop an existing SQL database.

Example:

```
DROP DATABASE testDB;
```

SQL Database commands

SQL Create Table

Statement used to create a new table in a database.

Example:

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

SQL Drop Table

used to drop an existing table in a database.Example:

```
DROP TABLE testTable;
```

SQL Database commands

ALTER TABLE

Statement used to add, delete, or modify columns in an existing table.

Example: Add column

```
ALTER TABLE table_name
ADD column_name datatype;
```

Example: Drop column

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

SQL Database commands

INSERT INTO

Statement used to insert new records in a table.

Example:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

Example:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

SQL Database commands

Update

Statement used to modify the existing records in a table.

Example:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

SQL Database commands

Delete

Statement used to delete existing records in a table.

Example:

```
DELETE FROM table_name WHERE condition;
```

Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

SQL Database commands

Select

Statement used to select data from a database.

Example:

```
SELECT column1, column2, ...
FROM table_name;
```

Example:

```
SELECT * FROM table_name;
```

SQL Database commands

Select Distinct

Statement used to return only different values.

Example:

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

Example:

```
SELECT DISTINCT Country FROM Customers;
```

SQL Database commands

Where

Statement used to filter records. Show only those records that fulfill a specific condition.

Example:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Example:

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

SQL Database commands

Like

Statement used in a where clause to search for a specified pattern in a column.

Wildcards:

- % - The percent sign represents zero, one, or multiple characters
- _ - The underscore represents a single character

Example:

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

Like

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%_%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

SQL Database commands

Is Null
Statement used to work with no values.

Example:

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

Example:

```
SELECT CustomerName, ContactName, Address
FROM Customers
WHERE Address IS NULL;
```

SQL Database commands

Between

Statement used to select values within a given range. Numbers, text or dates. The value is inclusive, begin and end values will be included.

Example:

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Example:

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;
```

SQL Database commands

AND, OR, NOT

- The AND operator displays a record if all the conditions separated by AND are TRUE.

- The OR operator displays a record if any of the conditions separated by OR is TRUE.

- The NOT operator displays a record if the condition(s) is NOT TRUE.

SQL Database commands

AND

Example:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

OR

Example:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

SQL Database commands

NOT

Example:

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

OR

Example:

```
SELECT * FROM Customers
WHERE NOT Country='Germany' AND NOT Country='USA';
```

SQL Database commands

Order By

Statement used to sort the result-set in ascending or descending order.
Ascending order by default.

Example:

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

Example:

```
SELECT * FROM Customers
ORDER BY Country DESC;
```

SQL Database commands

Limit/Select Top

Statement used to specify the number of records to return.

Example:

```
SELECT TOP number|percent column_name(s)
FROM table_name
WHERE condition;
```

Example:

```
SELECT * FROM Customers
WHERE ROWNUM <= 3;
```

Note: Not all database systems support the SELECT TOP clause. MySQL supports the LIMIT clause to select a limited number of records, while Oracle uses ROWNUM

SQL Database commands

Case

Statement used to go through conditions and returns a value when the first condition is met. If there is no ELSE part and no conditions are true, it returns NULL.

Example:

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    WHEN conditionN THEN resultN
    ELSE result
END;
```

SQL Aggregate Functions

AVG

The AVG() function returns the average value of a numeric column.

Example:

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

COUNT

The COUNT() function returns the number of rows that matches a specified criteria.

Example:

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

SQL Aggregate Functions

MIN

The MIN() function returns the smallest value of the selected column.

Example:

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

MAX

The MAX() function returns the largest value of the selected column.

Example:

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

SQL Aggregate Functions

SUM

The SUM() function returns the total sum of a numeric column.

Example:

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

SQL Database commands

Group By

Used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

Example:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

SQL Database commands

Having

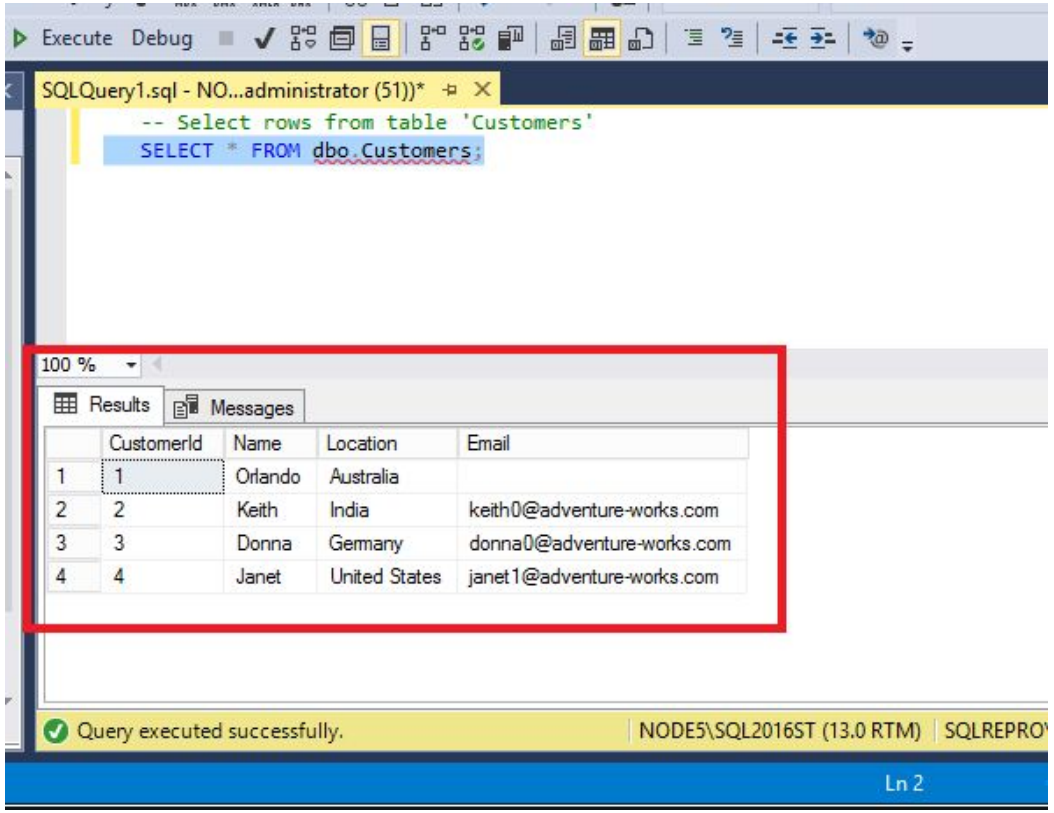
Was added to SQL because the WHERE keyword could not be used with aggregate functions.

Example:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

Query structure

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
GROUP BY *column_name(s)*
HAVING *condition*
ORDER BY *column_name(s);*



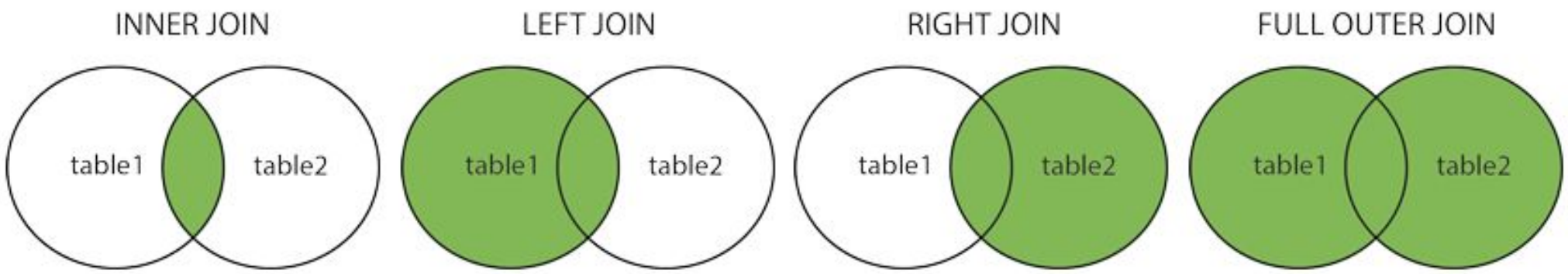
Joins

(INNER) JOIN: Returns records that have matching values in both tables.

LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table.

RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table.

FULL (OUTER) JOIN: Return all records when there is a match in either left or right table.



SQL Aggregate Functions

Inner Join

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

Left Join

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

SQL Aggregate Functions

Right Join

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

Full Join

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```

SQL Aggregate Functions

Right Join

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

Full Join

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```

SQL Constraints

NOT NULL - Ensures that a column cannot have a NULL value

UNIQUE - Ensures that all values in a column are different

PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

FOREIGN KEY - Uniquely identifies a row/record in another table

CHECK - Ensures that all values in a column satisfies a specific condition

DEFAULT - Sets a default value for a column when no value is specified

INDEX - Used to create and retrieve data from the database very quickly

SQL Constraints


```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
);
```

¡Gracias!

Contacto:
Marcela Álvarez
marcela.alvarez.rdz@gmail.com