YOUR

SQL

```
SELECT 'SQL QUICKSTART GUIDE'
FROM ANDREW_C_MADSON
WHERE 'HELPFUL' = MAXIMUM
```

GUIDE

@INSIGHTSxDESIGN

INSIGHTS
X DESIGN

KEY ANALYST SKILLS

EXCEL

SQL

VISUALIZATION

STATISTICS

CAREER

INSIGHTS X DESIGN

TABLE OF CONTENTS

INSIGHTS X DESIGN

INTRODUCTION

WHAT IS A DATA BASE?

HOW IS IT ORGANIZED?

WHAT IS SQL?

WHAT ARE THE DIFFERENT SQL DATABASES?

INSIGHTS X DESIGN

WHAT IS A DATABASE?

A database is an organized collection of data that is stored and accessed electronically. Databases are designed to hold structured data or data that can be stored in tables with rows and columns. They provide a way to manage large amounts of data efficiently and allow for complex operations and queries to be performed on the data.

INSIGHTS X DESIGN

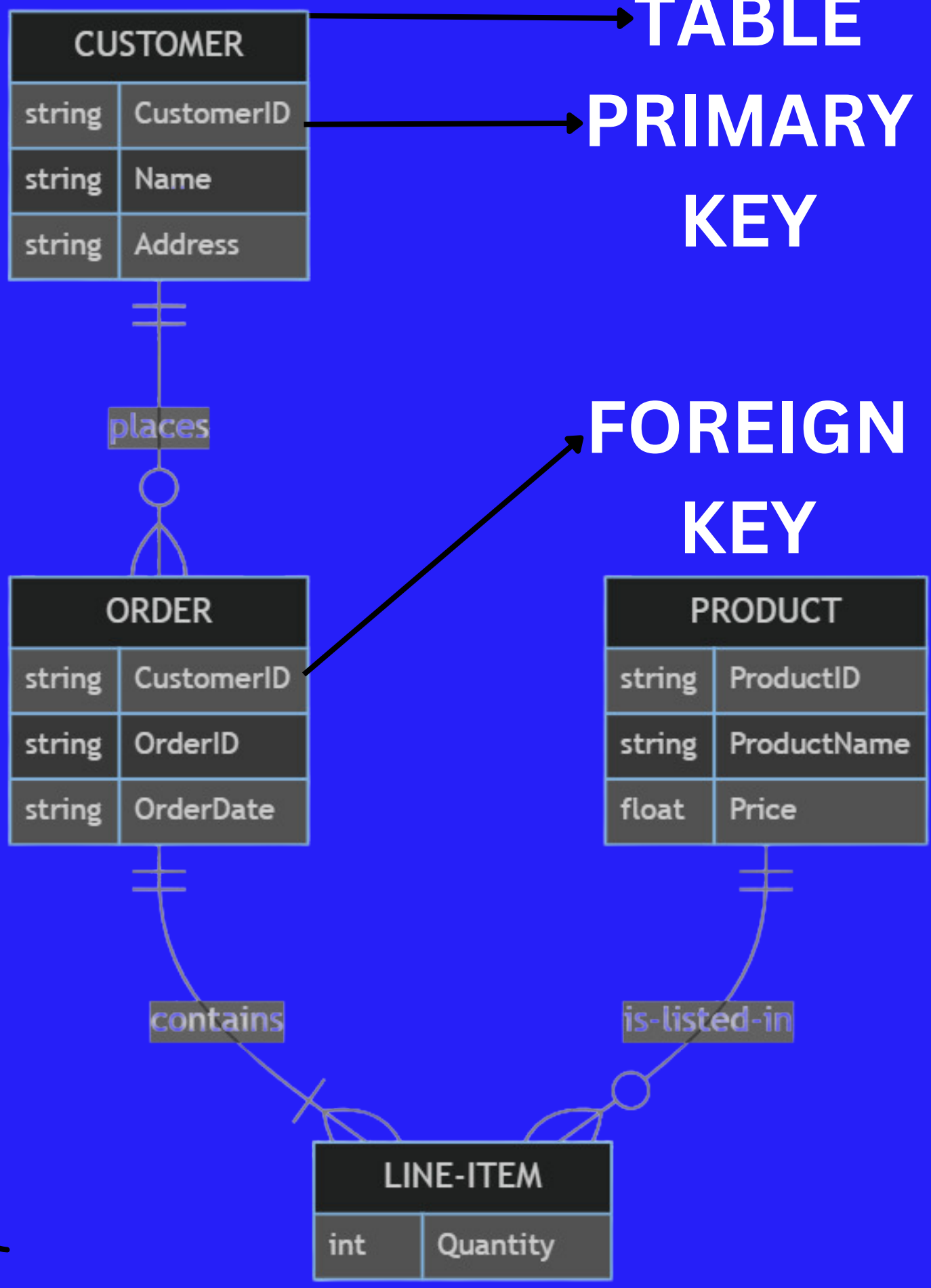
DATABASE COMPONENTS

SCHEMA	A collection of database objects, including tables, views, indexes, and procedures.
TABLE	A collection of related data held in a structured format within a database. It consists of columns, and rows.
PRIMARY KEY	A column (or a set of columns) in a table that uniquely identifies each row in that table.
FOREIGN KEY	a column (or a set of columns) in one table, that is used to link two tables together.
INDEX	Similar to an index in a book. It's a pointer to data in a table. An index in a database is a data structure that improves the speed of data retrieval operations on a database table.

INSIGHTS X DESIGN

HOW IS IT ORGANIZED?

SCHEMA



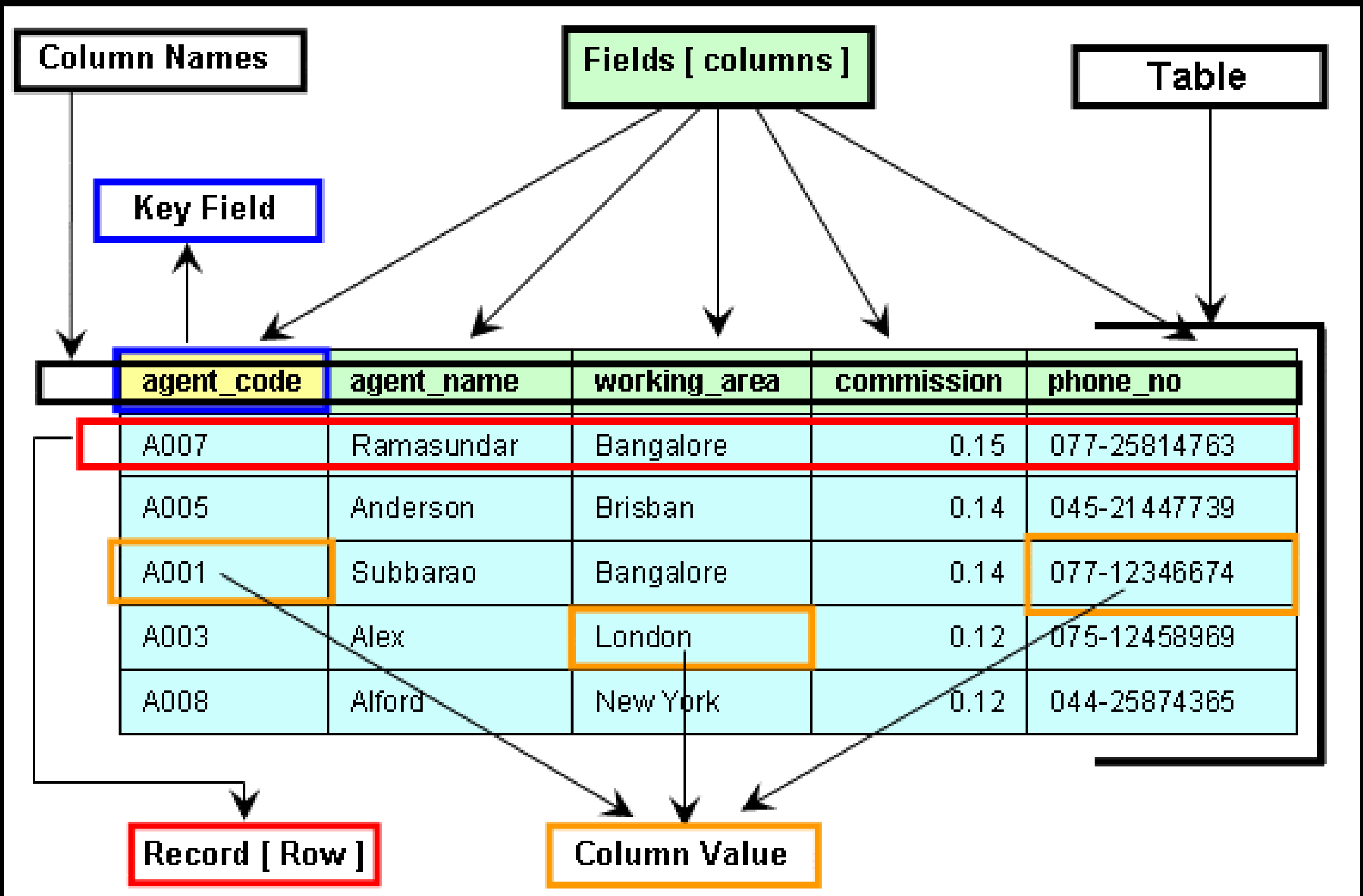
TABLE

PRIMARY KEY

FOREIGN KEY

INSIGHTS X DESIGN

WHAT IS A TABLE?



SOURCE: https://www.w3resource.com/sql/sql-basic/the-components-of-a-table.php

INSIGHTS X DESIGN

WHAT IS SQL?

SQL (Structured Query Language) is a programming language used to manage and manipulate relational databases. SQL is primarily used for managing data held in a relational database management system (RDBMS) or for processing data in a stream management system. It is particularly effective for handling structured data, i.e., data incorporating relations among entities and variables.

INSIGHTS X DESIGN

WHAT ARE THE TYPES OF SQL DATABASES?

MySQL	This is an open-source relational database management system (RDBMS) owned by Oracle. It's widely used in web applications and is a component of the LAMP web application software stack (Linux, Apache, MySQL, Perl/PHP/Python).
PostgreSQL	An open-source object-relational database system, PostgreSQL supports both SQL (relational) and JSON (non-relational) querying. It's known for its performance, robustness, and advanced features.
SQLite	Unlike most other SQL databases, SQLite is a serverless database and is embedded into the end program. It's a popular choice for local/client storage in web browsers and mobile apps.
Oracle	This is a multi-model database management system produced and marketed by Oracle Corporation. It's widely used in enterprise applications.
SQL Server	Developed by Microsoft, SQL Server is a relational database management system that supports a wide range of transaction processing, business intelligence, and analytics applications in corporate IT environments.
MariaDB	MariaDB is a community-developed, commercially supported fork of the MySQL relational database management system. It was created by the original developers of MySQL after concerns arose when Oracle acquired MySQL.
IBM DB2	An RDBMS from IBM, DB2 supports the relational model, but also supports object-relational features and non-relational structures like JSON and XML.

INSIGHTS X DESIGN

SQL BASICS

SQL SYNTAX

SELECT - FROM - WHERE

GROUP BY - HAVING

ORDER BY

AGGREGATE FUNCTIONS

INSIGHTS X DESIGN

SELECT - FROM - WHERE

SELECT	**This command is used to select data from a database. The data returned is stored in a result table, called the result-set. You specify the columns you want after the SELECT keyword. If you want to select all columns, you can use *.**
FROM	**This keyword is used to specify the table from which to select the data. You write the name of the table (or tables) after the FROM keyword.**
WHERE	**This keyword is used to filter the results. After the WHERE keyword, you specify the conditions that the data must meet. You can use operators like =, <, >, <=, >=, <> (not equal), BETWEEN, LIKE, and IN.**

INSIGHTS X DESIGN

EXAMPLE: SELECT - FROM - WHERE

SELECT	**n this example, the SELECT command is used to specify that we want the FirstName and LastName columns.**
FROM	**The FROM command is used to specify that we're selecting from the Employees table.**
WHERE	**The WHERE clause is used to filter out any employees whose Salary is not greater than 50000. So the statement will return a table of first and last names of employees who earn more than 50000.**

```sql
SELECT FirstName, LastName
FROM Employees
WHERE Salary > 50000;
```

INSIGHTS X DESIGN

GROUP BY - HAVING

GROUP BY	**The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.**
HAVING	**The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions. It is used to filter the results of a grouping. It acts like a WHERE clause but for groups of rows rather than individual rows.**

INSIGHTS
X DESIGN

EXAMPLE: GROUP BY

SELECT	**In this example, the SELECT statement is combined with GROUP BY to group the employees based on their Department and then count the number of EmployeeID in each department. The COUNT(EmployeeID) function will return the number of employees in each department.**
HAVING	**The HAVING clause is then used to filter out any departments where the count of EmployeeID (which is given the alias NumOfEmployees) is not greater than 5. So the statement will return a table of departments and their employee counts, but only for departments with more than 5 employees.**

```sql
SELECT Department, COUNT(EmployeeID) as NumOfEmployees
FROM Employees
GROUP BY Department
HAVING COUNT(EmployeeID) > 5;
```

INSIGHTS X DESIGN

ORDER BY

ORDER BY	The ORDER BY keyword in SQL is used to sort the result-set in ascending or descending order. It sorts the records in a result set by one or more columns.
ASCENDING	By default, ORDER BY sorts the data in ascending order. If you want to specify sorting in ascending order, you can use the ASC keyword, but it's not required.
DESCENDING	If you want to sort the data in descending order, you can use the DESC keyword.

INSIGHTS X DESIGN

EXAMPLE: ORDER BY

SELECT	**In this example, the SELECT statement is used to specify that we want the FirstName, LastName, and Salary columns from the Employees table.**
ORDER BY	**The ORDER BY keyword is used to sort the results by the Salary column in descending order (DESC). So the statement will return a table of first names, last names, and salaries of employees, ordered from the highest salary to the lowest.**

```sql
SELECT FirstName, LastName, Salary
FROM Employees
ORDER BY Salary DESC;
```

INSIGHTS X DESIGN

AGGREGATE FUNCTIONS

COUNT	This function returns the number of rows that matches a specified criterion.
AVG	This function returns the average value of a numeric column.
MIN	This function returns the smallest value of the selected column.
MAX	This function returns the largest value of the selected column.
SUM	This function returns the total sum of a numeric column.

INSIGHTS X DESIGN

EXAMPLE: AGGREGATE FUNCTIONS

Let's consider a simple table named 'Employees' with the following data:

EmployeeID	FirstName	LastName	Department	Salary
1	John	Doe	HR	70000
2	Jane	Smith	IT	80000
3	Mary	Johnson	HR	90000
4	James	Brown	IT	70000
5	Emily	Davis	Marketing	80000
6	Robert	Miller	HR	80000
7	Michael	Wilson	Marketing	70000

INSIGHTS X DESIGN

EXAMPLE: COUNT

```sql
SELECT COUNT(EmployeeID) as TotalEmployees
FROM Employees;
```

OUTPUT	**TotalEmployees** **7**
EXPLANATION	**This statement returns the total number of employees (EmployeeID).**

INSIGHTS
X DESIGN

EXAMPLE: AVG

```sql
SELECT AVG(Salary) as AverageSalary
FROM Employees;
```

OUTPUT	**AverageSalary** **77142.86**
EXPLANATION	**This SQL statement calculates and returns the average salary (Salary) of all employees.**

INSIGHTS
X DESIGN

EXAMPLE: MIN

```sql
SELECT MIN(Salary) as LowestSalary
FROM Employees;
```

OUTPUT	**LowestSalary** **70000**
EXPLANATION	**This SQL statement finds and returns the lowest salary (Salary) from all employees.**

INSIGHTS
X DESIGN

EXAMPLE: MAX

```sql
SELECT MAX(Salary) as HighestSalary
FROM Employees;
```

OUTPUT	**HighestSalary** **90000**
EXPLANATION	**This SQL statement finds and returns the highest salary (Salary) from all employees.**

INSIGHTS
X DESIGN

EXAMPLE: SUM

```sql
SELECT SUM(Salary) as TotalSalary
FROM Employees;
```

OUTPUT	**TotalSalary** **540000**
EXPLANATION	**This SQL statement calculates and returns the total sum of salaries (Salary) for all employees.**

JOINS

INNER	**This returns records that have matching values in both tables.**
LEFT	**This returns all records from the left table, and the matched records from the right table. If there is no match, the result is NULL on the right side.**
RIGHT	**This returns all records from the right table, and the matched records from the left table. If there is no match, the result is NULL on the left side.**
FULL	**This returns all records when there is a match in either the left or the right table. If there is no match, the result is NULL on either side.**

INSIGHTS X DESIGN

EXAMPLE: JOINS

Let's consider two tables:

TABLE A 'EMPLOYEES'

EmployeeID	FirstName	LastName
1	John	Doe
2	Jane	Smith
3	Mary	Johnson

TABLE B 'DEPARTMENTS'

DeptID	DeptName	EmployeeID
1	HR	1
2	IT	2
3	Marketing	4

EXAMPLE: INNER

TABLE A 'EMPLOYEES'

EmployeeID	FirstName	LastName
1	John	Doe
2	Jane	Smith
3	Mary	Johnson

TABLE B 'DEPARTMENTS'

DeptID	DeptName	EmployeeID
1	HR	1
2	IT	2
3	Marketing	4


```sql
SELECT Employees.FirstName, Employees.LastName,
Departments.DeptName
FROM Employees
INNER JOIN Departments ON Employees.EmployeeID =
Departments.EmployeeID;
```

OUTPUT

FirstName	LastName	DeptName
John	Doe	HR
Jane	Smith	IT

INSIGHTS X DESIGN

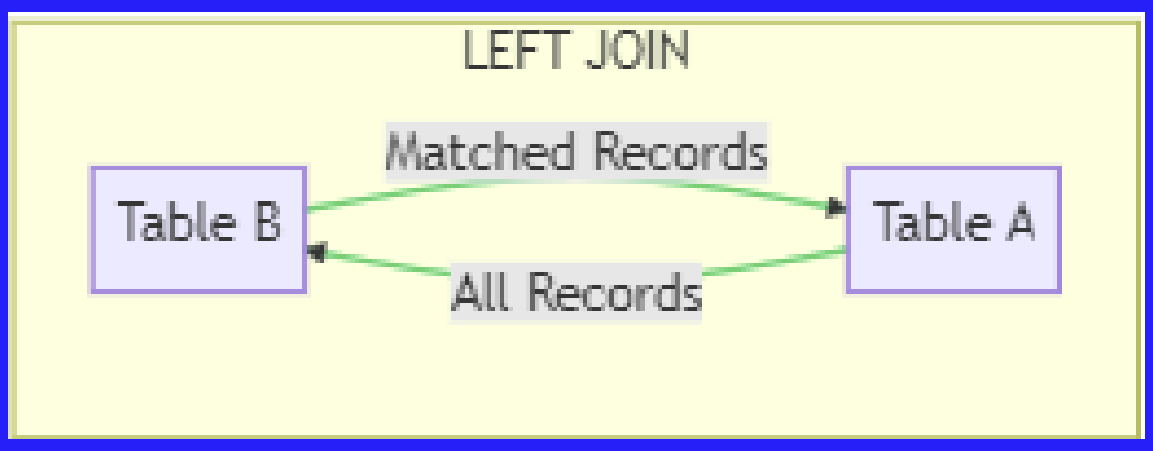
EXAMPLE: LEFT

TABLE A 'EMPLOYEES'

EmployeeID	FirstName	LastName
1	John	Doe
2	Jane	Smith
3	Mary	Johnson

TABLE B 'DEPARTMENTS'

DeptID	DeptName	EmployeeID
1	HR	1
2	IT	2
3	Marketing	4




```sql
SELECT Employees.FirstName, Employees.LastName,
Departments.DeptName
FROM Employees
LEFT JOIN Departments ON Employees.EmployeeID =
Departments.EmployeeID;
```

OUTPUT

FirstName	LastName	DeptName
John	Doe	HR
Jane	Smith	IT
Mary	Johnson	NULL

INSIGHTS X DESIGN

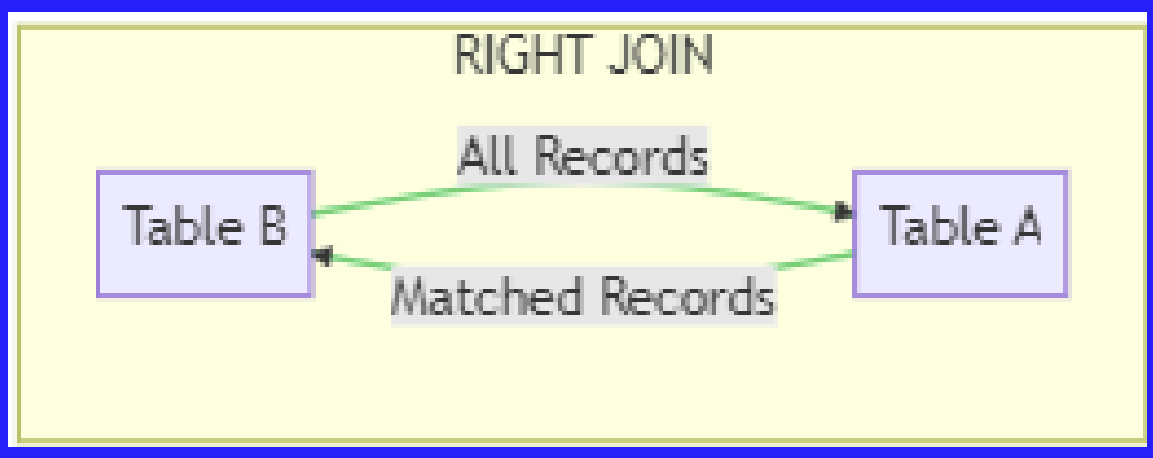
EXAMPLE: RIGHT

TABLE A 'EMPLOYEES'

EmployeeID	FirstName	LastName
1	John	Doe
2	Jane	Smith
3	Mary	Johnson

TABLE B 'DEPARTMENTS'

DeptID	DeptName	EmployeeID
1	HR	1
2	IT	2
3	Marketing	4




```sql
SELECT Employees.FirstName, Employees.LastName,
Departments.DeptName
FROM Employees
RIGHT JOIN Departments ON Employees.EmployeeID =
Departments.EmployeeID;
```

OUTPUT

FirstName	LastName	DeptName
John	Doe	HR
Jane	Smith	IT
NULL	NULL	Marketing

INSIGHTS X DESIGN

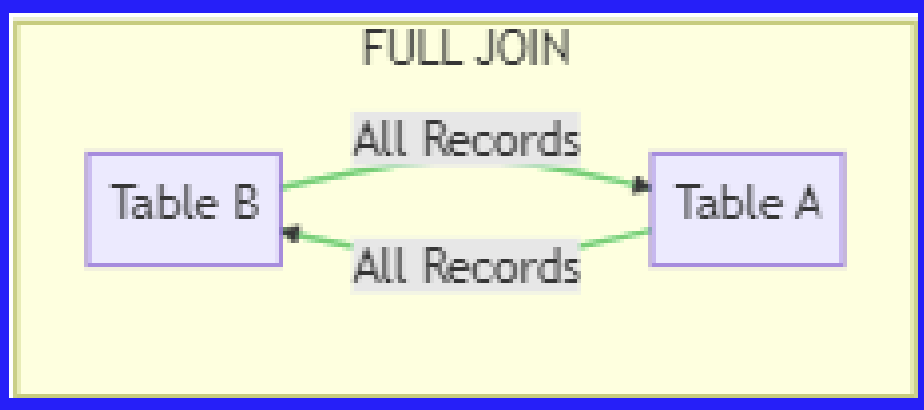
EXAMPLE: FULL

TABLE A 'EMPLOYEES'

EmployeeID	FirstName	LastName
1	John	Doe
2	Jane	Smith
3	Mary	Johnson

TABLE B 'DEPARTMENTS'

DeptID	DeptName	EmployeeID
1	HR	1
2	IT	2
3	Marketing	4




```sql
SELECT Employees.FirstName, Employees.LastName,
Departments.DeptName
FROM Employees
FULL JOIN Departments ON Employees.EmployeeID =
Departments.EmployeeID;
```

OUTPUT

FirstName	LastName	DeptName
John	Doe	HR
Jane	Smith	IT
Mary	Johnson	NULL
NULL	NULL	Marketing

INSIGHTS X DESIGN

RESOURCES

SQL: A Practical Introduction for Querying Databases

Introduction to SQL on DataCamp

The Complete SQL Bootcamp 2021: Go from Zero to Hero on Udemy

Intro to SQL: Querying and managing data on Khan Academy

Learn SQL on Codecademy

INSIGHTS X DESIGN

LIKE &

FOLLOW

FOR MORE!

@INSIGHTSxDESIGN

INSIGHTS
X DESIGN