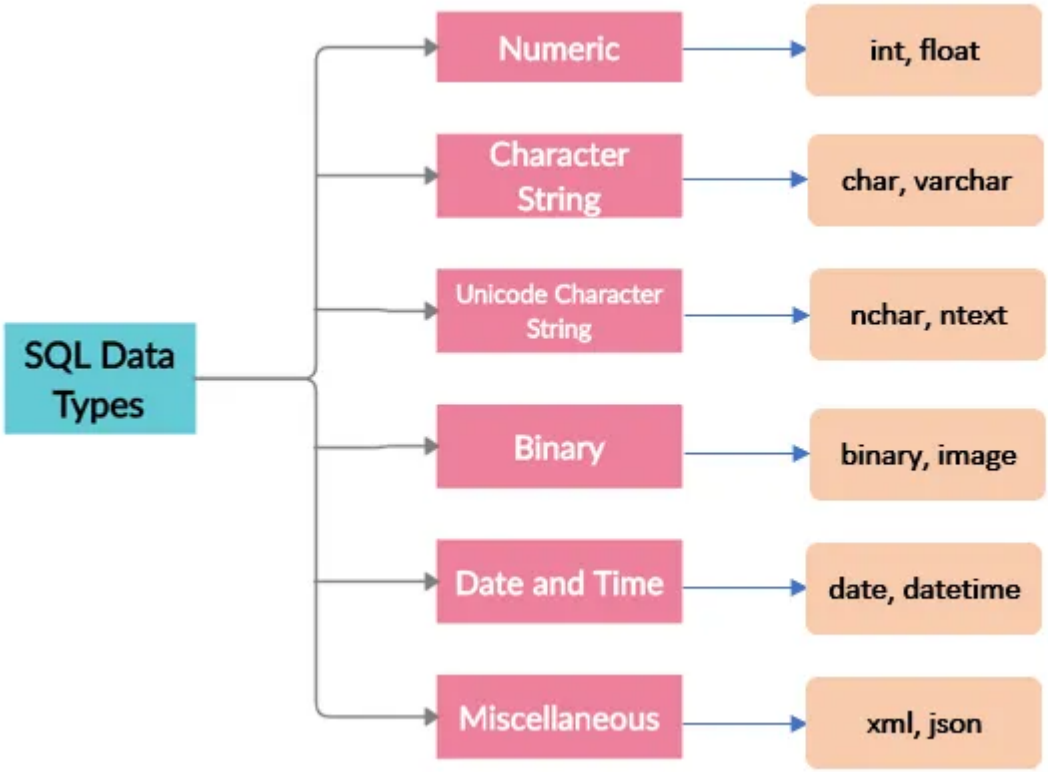
Muhammad Fahad

Apr 2 · 12 min read · Listen

SQL 101: A Step-by-Step Guide to Getting Started with SQL

Structured Query Language (SQL) is a programming language used for managing and manipulating data stored in relational databases. It provides a standardized syntax for performing various operations, such as inserting, updating, deleting and querying data. SQL can be used to retrieve specific information from a database by constructing queries that filter data based on specific criteria. Additionally, SQL allows users to create tables, views and other database objects that can help organize and optimize data storage.

SQL Datatypes



Creating Database

The Create Database statement is used to create a new SQL database.

```
CREATE DATABASE testDB;
```

Alter Database

Alter Database command is used to modify the structure of an existing database. You can use Alter Database to change database options or set new default values for specific parameters.

```
ALTER DATABASE testDB
MODIFY NAME = company;
```

Other Database Altering Operations:

- **Modifying Database Options:** You can change the collation of a database, set a new default database file location or change the default language of the database.
- **Renaming Database:** You can use Alter Database to rename a database. To do this, you need to specify the current database name and the new name you want to give to the database.
- **Setting Database State:** You can use Alter Database to set the state of a database to either **Online** or **Offline**. When a database is offline, it is not accessible to users and you can perform maintenance tasks or restore backups.
- **Authentication Mode:** You can use Alter Database to modify the authentication mode of a database. By default, SQL Server uses Windows Authentication mode, but you can switch to SQL Server Authentication mode if needed.

Drop Database

The Drop Database command is used to delete an existing database. When you drop a database, all the tables, data and other objects in the database are deleted permanently

```
DROP DATABASE company;
```

Creating Table

The Create Table statement is used to create a new table in a database.

```
CREATE TABLE emp (
    emp_ID INT,
    first_Name VARCHAR(25),
    last_Name VARCHAR(25),
    dept VARCHAR(25),
    phone INT,
    address VARCHAR(255)
);
```

- **Create Table from another Table**

A copy of an existing table can also be created using Create Table. The new table gets the same column definitions. All columns or specific columns can be selected. If you create a new table using an existing table, the new table will be filled with the existing values from the old table.

```
CREATE TABLE TestTable AS
SELECT emp_ID, first_name
FROM emp;
```

Alter Table

The Alter Table statement is used to ***add, delete or modify columns in an existing table***. It is also used to add and drop various constraints on an existing table. Some common table alteration operations in SQL include

- **Add**: This operation is used to add a new column to an existing table.

```
ALTER TABLE emp
ADD COLUMN id_no VARCHAR(20);
```

- **Alter**: This operation is used to modify the data type of an existing column.

```
ALTER TABLE emp
ALTER COLUMN phone VARCHAR(15);
```

- **Drop:** This operation is used to remove a column from an existing table.

```
ALTER TABLE emp
DROP COLUMN DateOfBirth;
```

- **Rename:** This operation is used to rename an existing table or column.

```
ALTER TABLE emp
RENAME COLUMN first_name to emp_name;
```

SQL Constraints

SQL constraints are rules or restrictions that are enforced on data in a database table to ***maintain data integrity and consistency***. They are used to prevent the insertion of invalid or incorrect data into a table and ensure that the data in the table meets certain criteria.

There are several types of SQL constraints, including:

- **Primary Key**: Primary Key (PK) ensures that each record in a table has a unique identifier that can be used to retrieve it.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
```

```
    Age int,
    CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)
);
```

- **Foreign Key:** Foreign Key (FK) ensures that data in one table is related to data in another table, and maintains referential integrity between the two tables.

```
CREATE TABLE Orders (
    OrderID int NOT NULL PRIMARY KEY,
    OrderNumber int NOT NULL,
    PersonID int FOREIGN KEY REFERENCES Persons(PersonID)
);
```

- **Not Null Constraint:** Not Null ensures that a column in a table cannot contain null values.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int
);
```

- **Unique Constraint:** Unique ensures that the values in a column or set of columns are unique.

```
CREATE TABLE Persons (
    ID int NOT NULL UNIQUE,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

- **Default Constraint:** Default ensure that to provide a default value for a column in a table, unless a different value is explicitly specified.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255) DEFAULT 'Sandnes'
);
```

Data Manipulation Language (DML)

Data Manipulation Language (DML) commands are used to manipulate or modify the data within the database. DML commands are used to ***add***, ***update***, ***delete*** or ***retrieve*** data from the database.

> *Data Manipulation Language (DML) commands include:*

- **Insert** : Insert command is used to insert one or more rows of data into a table. You can either specify the values for all columns or only for a subset of columns. If you don't specify a value for a column, it will be set to NULL.

```
INSERT INTO employees(ID, first_name, last_name, department, address)
VALUES (457,'Fahad','Arshad','IT','Peshawar');
```

Update : Update is used to modify existing data in one or more rows of a table. You can update one or more columns, and you can also use the WHERE clause to update only specific rows.

```
UPDATE employees
SET first_name = 'M.Fahad'
WHERE ID = 457;
```

Delete : Delete command is used to users to remove single or multiple existing records from the database tables.

```
DELETE FROM employees
WHERE ID=457;
```

Select: The Select statement is used to select data from a database.

```
SELECT CustomerName, City
FROM Customers;
```

Where: The WHERE clause is used to filter records.

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

• **Order By:** The ORDER BY keyword is used to sort the result-set in ascending or descending order.

```
SELECT * FROM Customers
```

```
ORDER BY Country;
```

- **Distinct:** The DISTINCT statement is used to return only distinct (different) values.

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

- **Aliases:** SQL aliases are used to give a table, or a column in a table, a temporary name and they are often used to make column names more readable. It only exists for the duration of that query. An alias is created with the AS keyword.

```
SELECT CustomerID AS ID, CustomerName AS Customer
FROM Customers;
```

Predicates in SQL

- **Between:** The `BETWEEN` operator selects values within a given range. The values can be numbers, text, or dates.

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

In: The IN operator allows you to specify multiple values in a WHERE clause.

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

Like: The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

```
SELECT * FROM Customers
WHERE CustomerName LIKE '%a';
```

Here are some examples showing different LIKE operators with '%' and '_' wildcards:

- 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 ContactName LIKE 'a%o' — Finds any values that **start with "a" and ends with "o"**

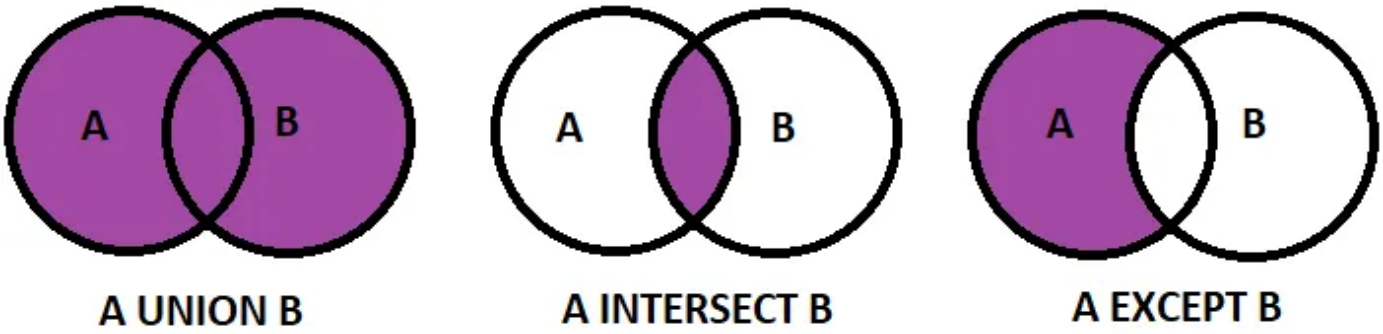
Is Null: The IS NULL operator is used to test for empty values (NULL values).

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

Top Clause: The Select Top clause is used to specify the number of records to return.

```
SELECT TOP 3 * FROM Customers;
```

Set Operators in SQL



- **Union:** The Union operator is used to combine the result-set of two or more Select statements.

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;
```

- **Intersect:** Intersect is used to combine two Select statements, but returns rows only from the first Select statement that are identical to a row in the second Select statement. This means Intersectreturns only common rows returned by the two Select statements.

```
SELECT City FROM Customers
INTERSECT
SELECT City FROM Suppliers
ORDER BY City;
```

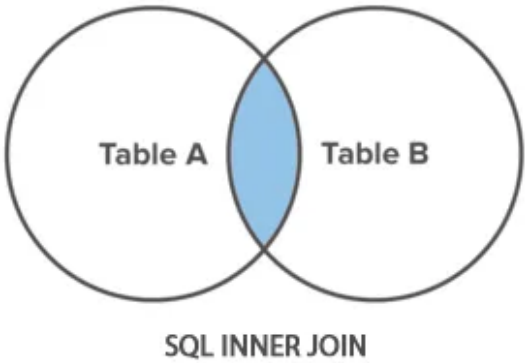
- **Except:** Except operator is used to combine two Select statements and returns rows from the first Select statement that are not returned by the second Select

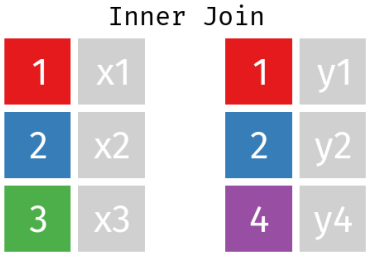
statement. This means Except returns only rows, which are not available in the second Select statement.

```
SELECT  id, name, amount, date
   FROM customer
   LEFT JOIN orders
   ON customer.id = orders. order_id
EXCEPT
   SELECT  id, name, amount, date
   FROM customer
   RIGHT JOIN orders
   ON customer.id = orders. order_id;
```

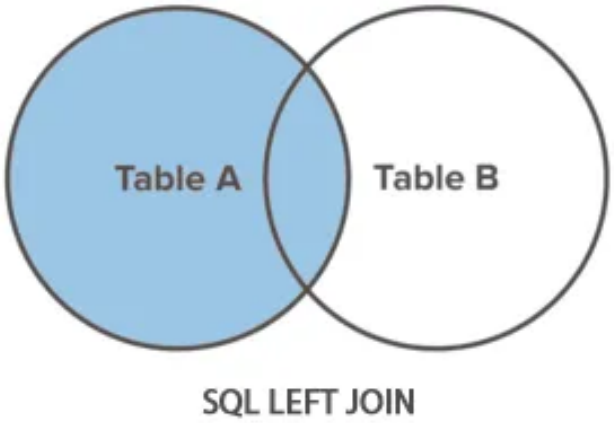
Joins in SQL

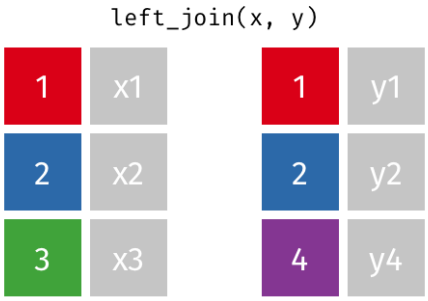
Inner Join: The Inner Join keyword selects records that have matching values in both tables.



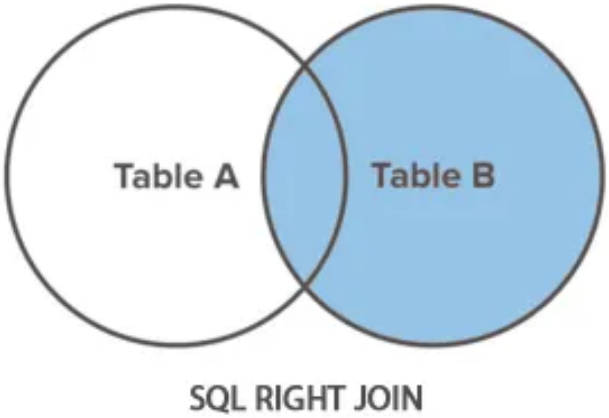


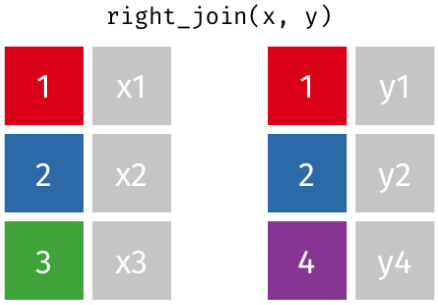
Left Join: The Left Join keyword returns all records from the left table (Table1), and the matching records from the right table (Table2). The result is 0 records from the right side, if there is no match.



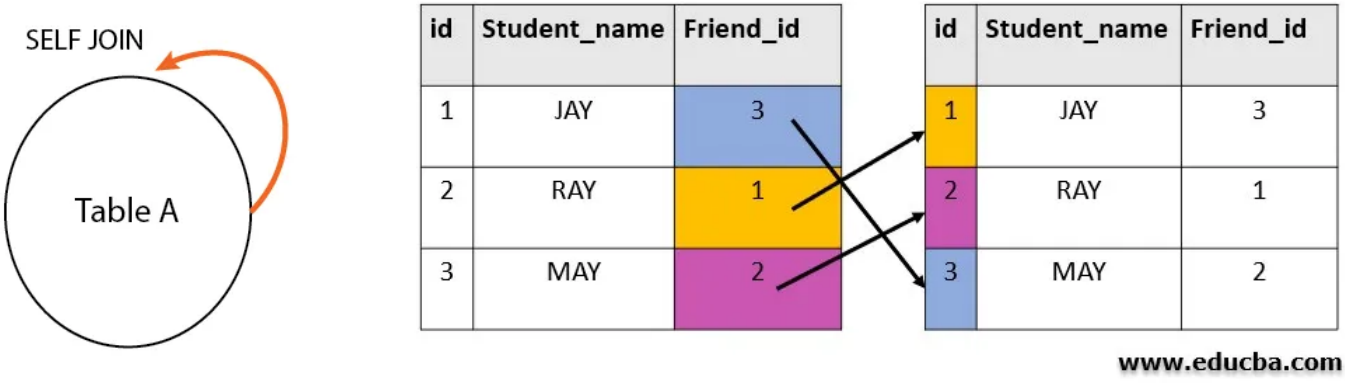


Right Join: The Right Join keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

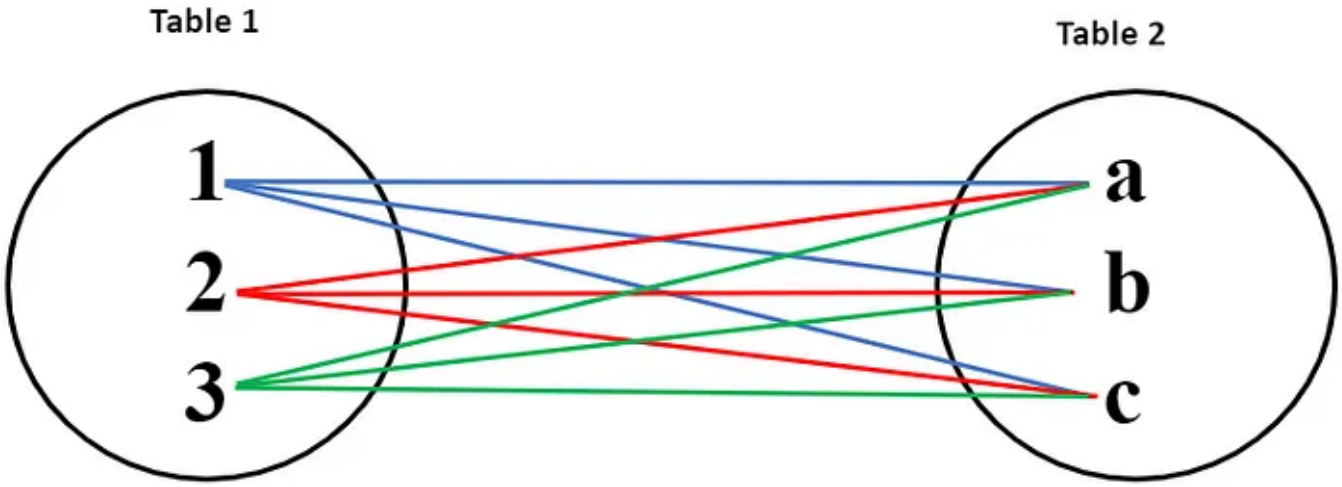




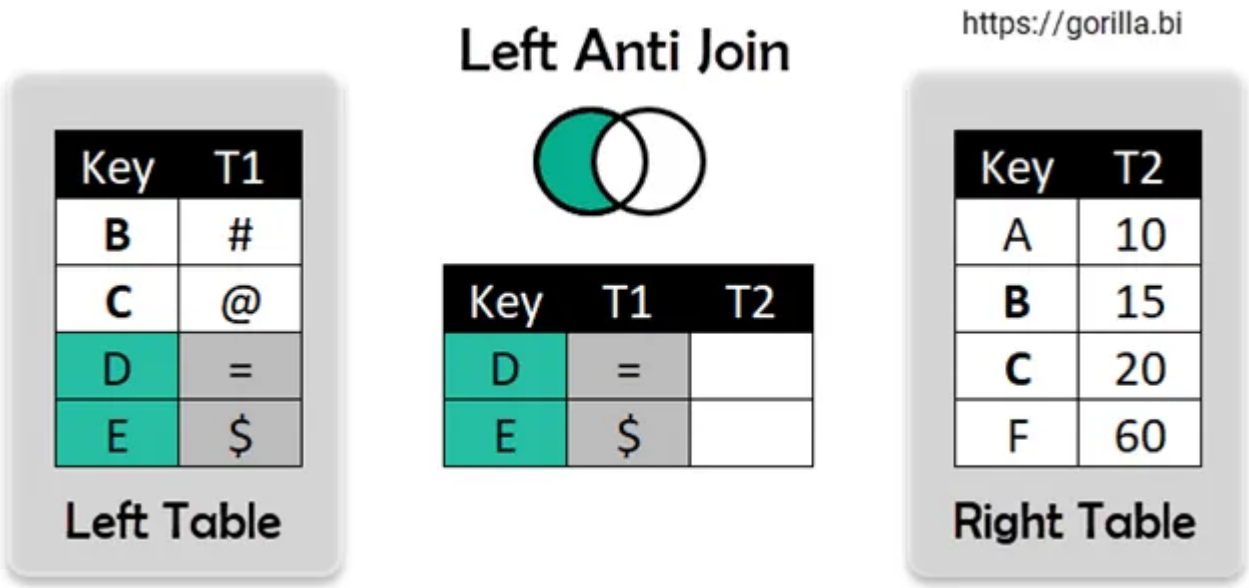
• **Self Join:** A self join is a regular join, but the table is joined with itself.



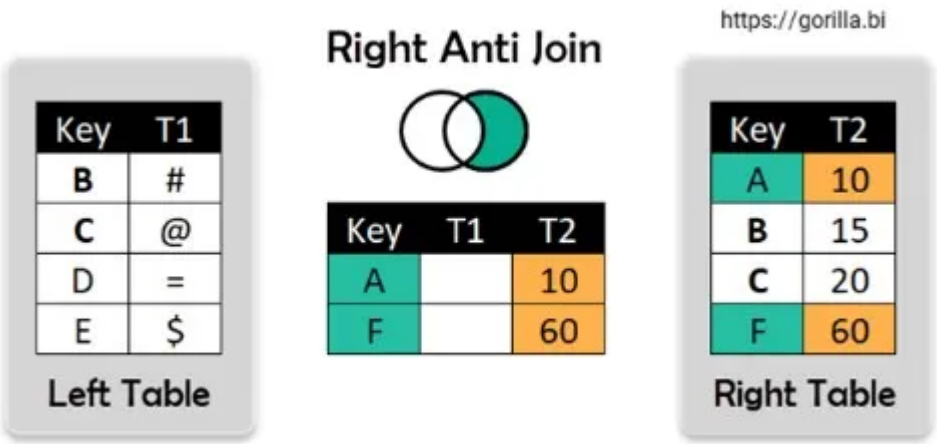
Cross Join: The Cross Join returns all records from both tables (Table 1 and Table 2).



Left anti-Join: One of the join kinds available in the Merge dialog box in Power Query is a *left anti join*, which brings in only rows from the left table that don't have any matching rows from the right table



Right anti-Join: One of the join kinds available in the Merge dialog box in Power Query is a *right anti join*, which brings in only rows from the right table that don't have any matching rows from the left table

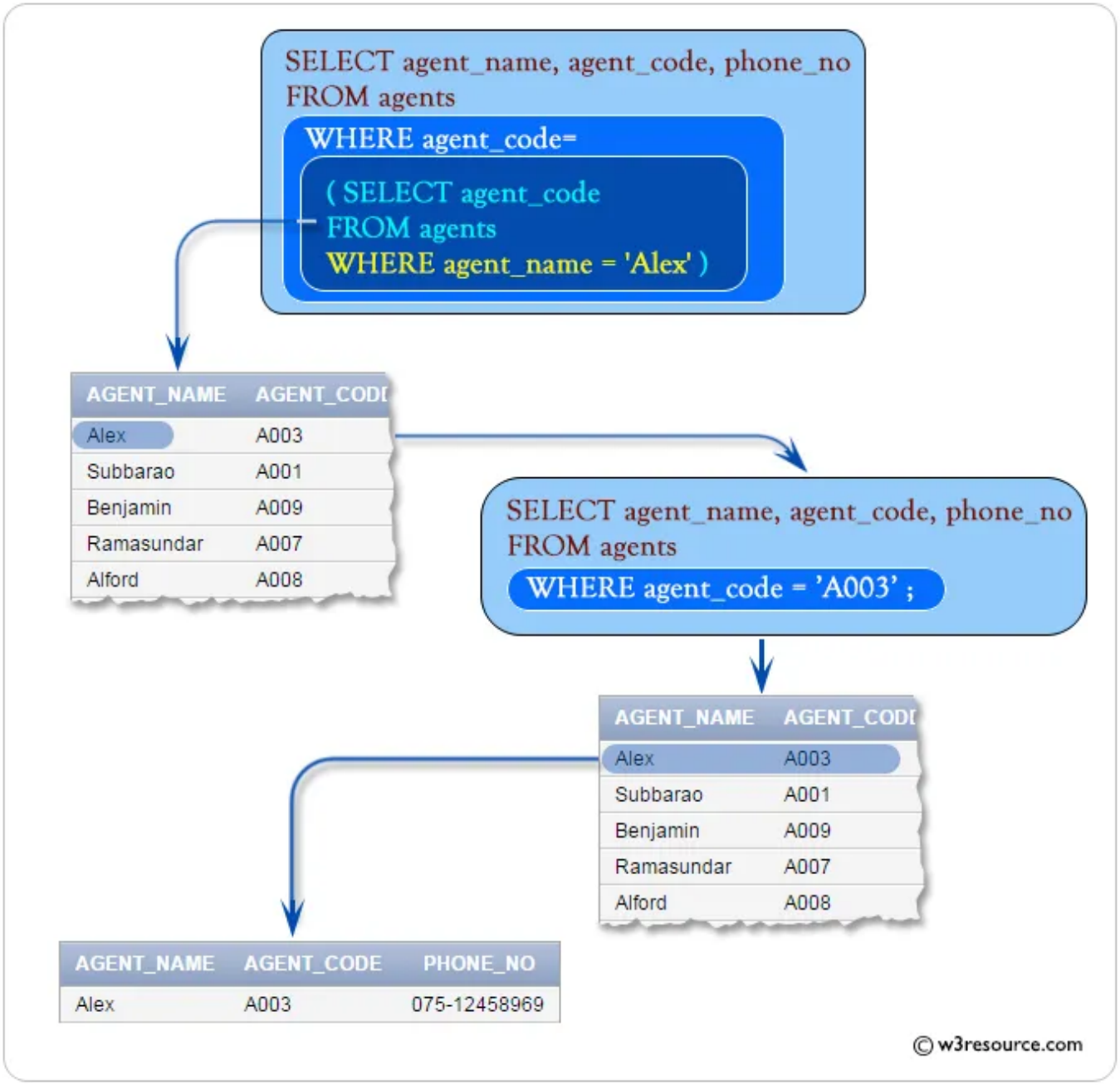


Sub Queries in SQL

A subquery in SQL is a **query that is nested inside another query.** It is a SELECT statement that is used within another SELECT, INSERT, UPDATE, or DELETE statement to retrieve data that will be used as a condition or value for the main query. In general, subqueries can make SQL queries more powerful and flexible, allowing developers to retrieve data from multiple tables and manipulate it in various ways. Below are some types of Sub Queries in SQL:

Single Row Sub Query: A single row subquery *returns 0 or 1 row to the outer SQL statement*. You can place a subquery in a WHERE clause, a HAVING clause or a FROM clause of a SELECT statement.

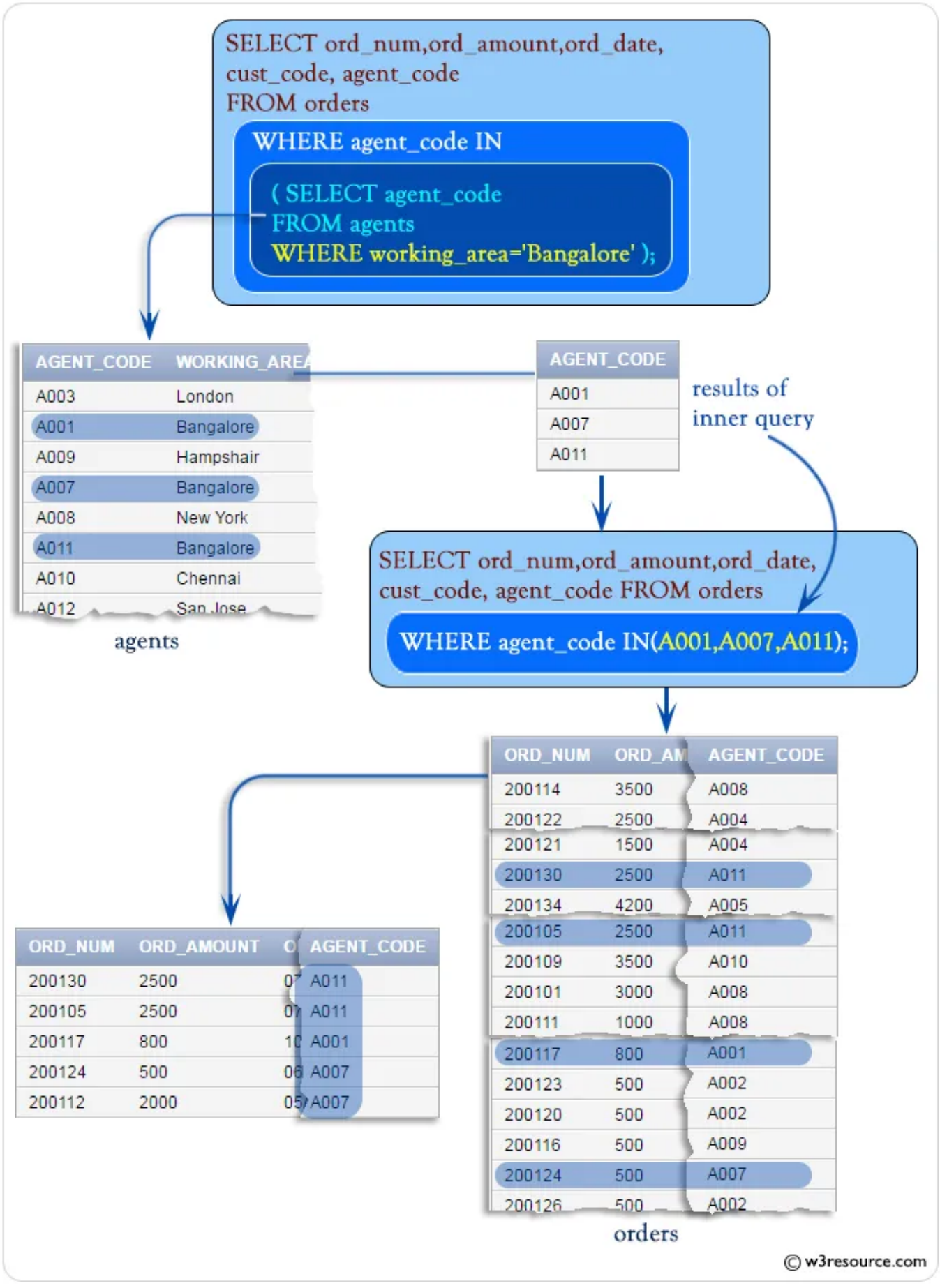
```
SELECT agent_name, agent_code, phone_no
FROM agents
WHERE agent_code =
(SELECT agent_code
FROM agents
WHERE agent_name = 'Alex');
```



Multi Row Sub Queries: Multiple row subquery *returns 1 or more rows to the outer SQL statement.* You may use the IN, ANY, or ALL operator in outer query to handle a subquery that returns multiple rows.

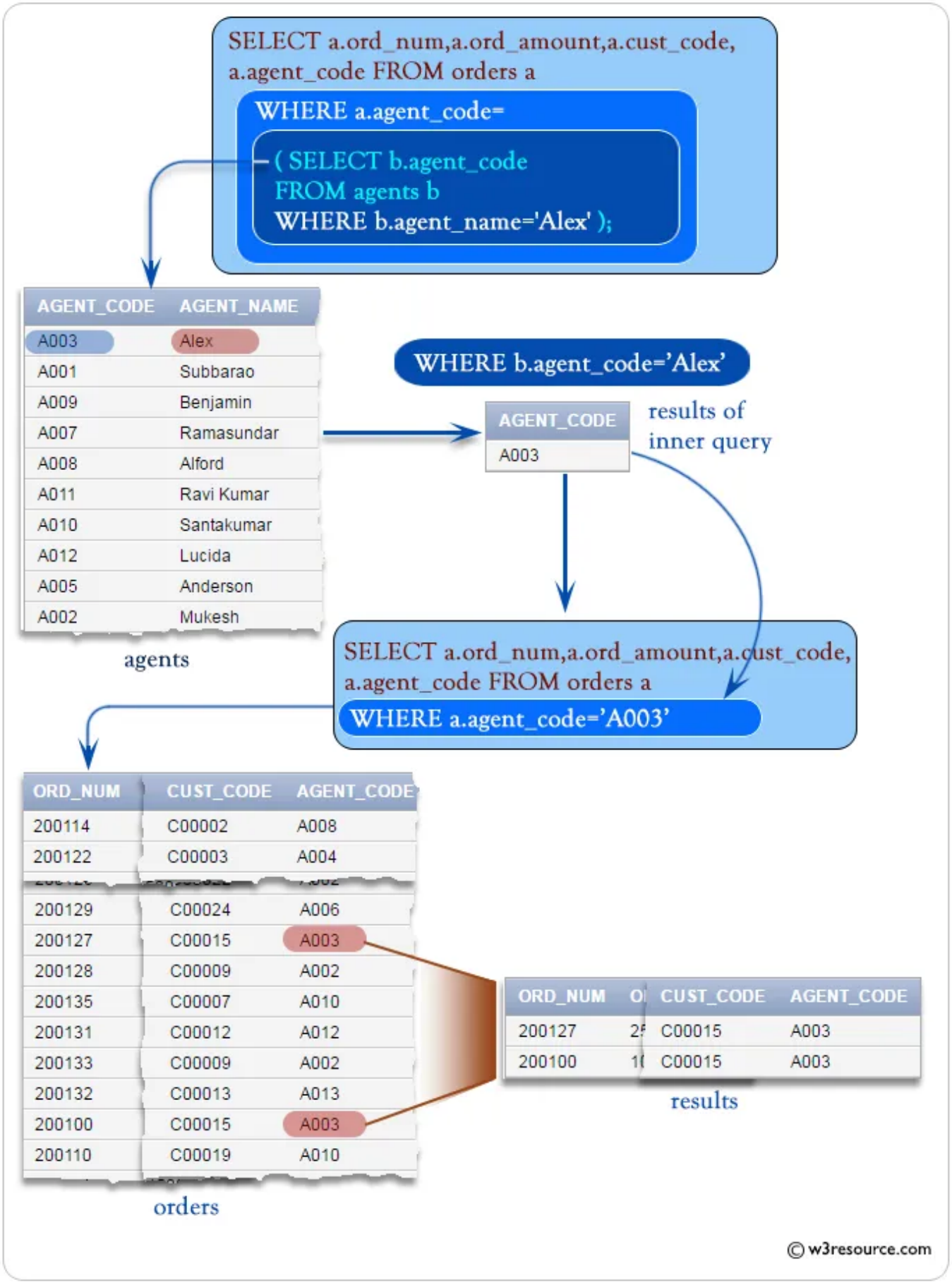
```
SELECT ord_num,ord_amount,ord_date,
cust_code, agent_code
FROM orders
WHERE agent_code IN(
```

```
SELECT agent_code FROM agents
WHERE working_area='Bangalore');
```


SELECT ord_num,ord_amount,ord_date,
cust_code, agent_code
FROM orders
WHERE agent_code IN
(SELECT agent_code
FROM agents
WHERE working_area='Bangalore');
AGENT_CODE WORKING_AREA
A003 London
A001 Bangalore
A009 Hampshair
A007 Bangalore
A008 New York
A011 Bangalore
A010 Chennai
A012 San Jose
agents
AGENT_CODE
A001
A007
A011
results of
inner query
SELECT ord_num,ord_amount,ord_date,
cust_code, agent_code FROM orders
WHERE agent_code IN(A001,A007,A011);
ORD_NUM ORD_AM AGENT_CODE
200114 3500 A008
200122 2500 A004
200121 1500 A004
200130 2500 A011
200134 4200 A005
200105 2500 A011
200109 3500 A010
200101 3000 A008
200111 1000 A008
200117 800 A001
200123 500 A002
200120 500 A002
200116 500 A009
200124 500 A007
200126 500 A002
orders
ORD_NUM ORD_AMOUNT O AGENT_CODE
200130 2500 07 A011
200105 2500 07 A011
200117 800 10 A001
200124 500 06 A007
200112 2000 05 A007
© w3resource.com

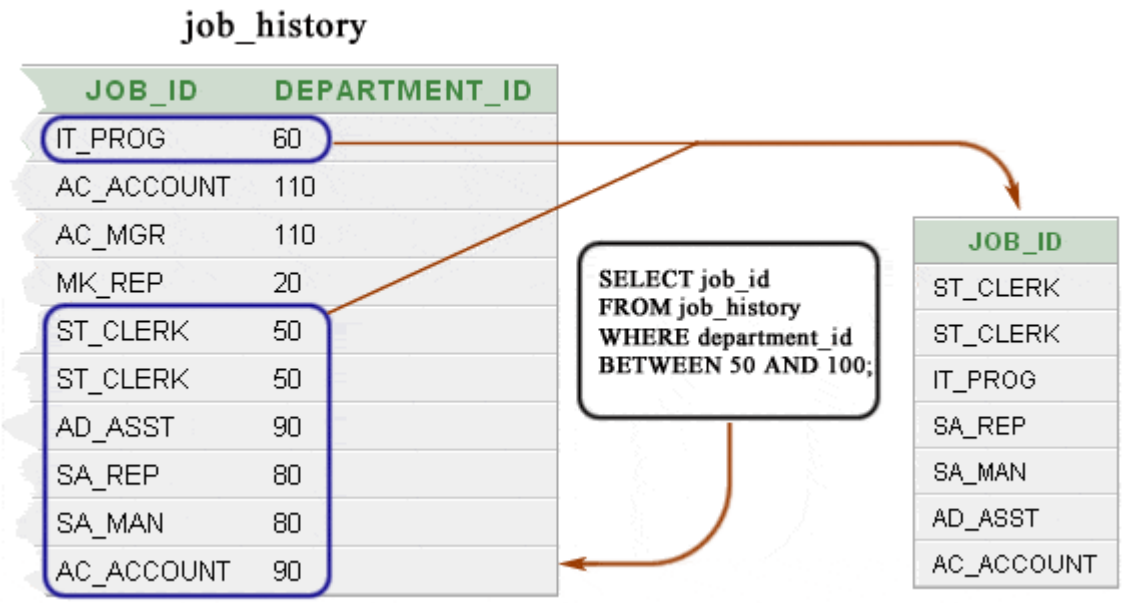
Nested Sub Queries: A subquery can be ***nested inside other subqueries. SQL has an ability to nest queries within one another.*** A subquery is a SELECT statement that is nested within another SELECT statement and which return intermediate results. SQL executes innermost subquery first, then next level.

```
SELECT a.ord_num,a.ord_amount,a.cust_code,a.agent_code
FROM orders a
WHERE a.agent_code=(
SELECT b.agent_code
FROM agents b WHERE b.agent_name='Alex');
```


SELECT a.ord_num,a.ord_amount,a.cust_code, a.agent_code FROM orders a
WHERE a.agent_code=
(SELECT b.agent_code
FROM agents b
WHERE b.agent_name='Alex');
AGENT_CODE AGENT_NAME
A003 Alex
A001 Subbarao
A009 Benjamin
A007 Ramasundar
A008 Alford
A011 Ravi Kumar
A010 Santakumar
A012 Lucida
A005 Anderson
A002 Mukesh
agents
WHERE b.agent_code='Alex'
AGENT_CODE
A003
results of inner query
SELECT a.ord_num,a.ord_amount,a.cust_code, a.agent_code FROM orders a
WHERE a.agent_code='A003'
ORD_NUM CUST_CODE AGENT_CODE
200114 C00002 A008
200122 C00003 A004
200129 C00024 A006
200127 C00015 A003
200128 C00009 A002
200135 C00007 A010
200131 C00012 A012
200133 C00009 A002
200132 C00013 A013
200100 C00015 A003
200110 C00019 A010
orders
ORD_NUM CUST_CODE AGENT_CODE
200127 C00015 A003
200100 C00015 A003
results
© w3resource.com

Co-Related Sub Query: Correlated subqueries are ***used for row-by-row processing.*** Each subquery is executed once for every row of the outer query. Acorrelated subquery is evaluated once for each row processed by the parent statement. The parent statement can be a SELECT, UPDATE, or DELETE statement.

```
SELECT job_id,AVG(salary)
FROM employees
GROUP BY job_id
HAVING AVG(salary)<
(SELECT MAX(AVG(min_salary))
FROM jobs
WHERE job_id IN
(SELECT job_id FROM job_history
WHERE department_id
BETWEEN 50 AND 100)
GROUP BY job_id);
```



Views in SQL

A view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or

more real tables in the database. You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

- **Simple View:** Simple View in SQL is the view created by involving **only single table**. In other words we can say that there is only one base table in case of Simple View in SQL.

```
CREATE VIEW Employee AS
SELECT Empid, Empname
FROM Employee
WHERE Empid = ‘030314’;
```

- **Complex View:** Complex View is created by involving **more than one table** i.e., multiple tables get projected in Complex view.

```
CREATE VIEW EmployeeByDepartment AS
SELECT e.emp_id, d.dept_id, e.emp_name
FROM Employee e, Department d
WHERE e.dept_id = d.dept_id
```

Creating View

```
CREATE VIEW [German Customers] AS
SELECT CustomerName, ContactName
FROM Customers
WHERE Country = 'Germany';
```

Altering View

```
ALTER VIEW [Top Avg Products] AS
SELECT TOP 10 ProductName, UnitPrice , CategoryName
```

```
FROM Products
INNER JOIN Categories ON Products.CategoryID = Categories.CategoryID
WHERE UnitPrice > (
  SELECT AVG(UnitPrice) FROM Products
)
SELECT * FROM [Top Avg Products]
```

Dropping View

```
DROP VIEW [German Customers];
```

Indexes in SQL

Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches or queries. Here some types of indexes used often:

- **Clustered Index:** A clustered index is used to define the order or to sort the table or arrange the data by alphabetical order just like a dictionary. It is faster than a non-clustered index. It demands less memory to execute the operation.

```
CREATE CLUSTERED INDEX idx_Order_ID
ON [Order Details] (OrderID);
```

- **Non Clustered Index:** A non-clustered index collects the data at one place and records at another place. It is slower than the clustered index. It demands more memory to execute the operations.

```
CREATE NONCLUSTERED INDEX idx_Customer_LastName
```

```
ON Customers (ContactName);
```

Creating Index

```
CREATE INDEX idx_titles ON Customers (ContactTitles);
```

Altering Index

```
ALTER INDEX idx_titles ON Customers REBUILD;
```

Drop Index

```
DROP INDEX idx_titles ON Customers;
```

Conditional Control Statement

- **Case:** The Case expression goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the `ELSE` clause. If there is no `ELSE` part and no conditions are true, it returns NULL.

```
SELECT ProductID, ProductName, UnitPrice, UnitsInStock,
CASE
    WHEN UnitsInStock <= 20 THEN 'Stock Low'
    WHEN UnitsInStock >= 20 THEN 'Stock Full'
```

```
    END AS Stock_Status
    FROM Products;
```

Essential SQL Commands for Data Management and Manipulation: Guide to Database Operations

Structured Query Language (SQL) is a programming language used to manage and manipulate data stored in relational...

medium.com

Refer to my previous article on Essential SQL Commands to learn more.

Data Data Engineer Data Engineering Sql Database