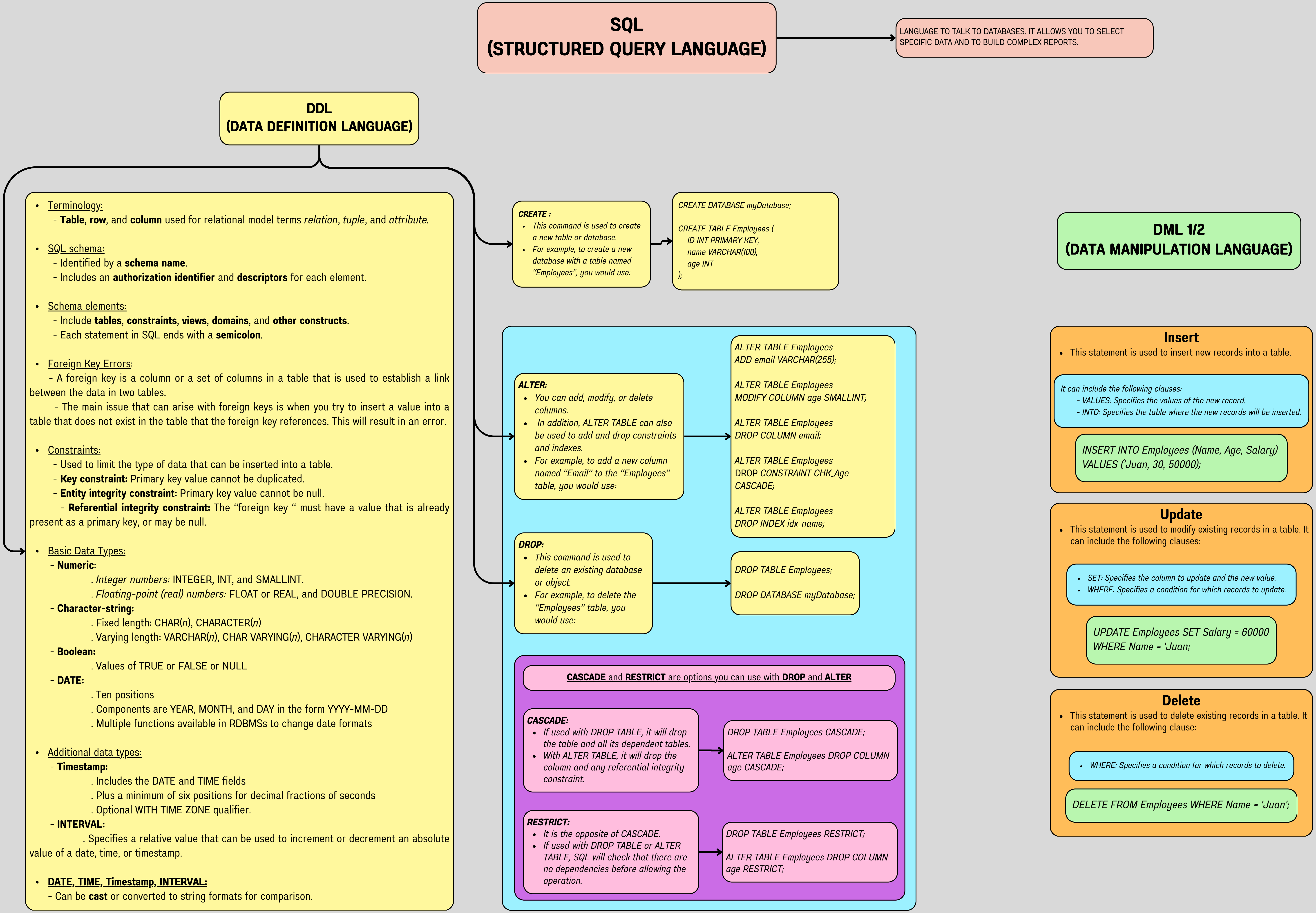


SQL
(STRUCTURED QUERY LANGUAGE)

LANGUAGE TO TALK TO DATABASES. IT ALLOWS YOU TO SELECT SPECIFIC DATA AND TO BUILD COMPLEX REPORTS.

DDL
(DATA DEFINITION LANGUAGE)

- Terminology:
  - **Table**, **row**, and **column** used for relational model terms *relation*, *tuple*, and *attribute.*
- SQL schema:
  - Identified by a **schema name**.
  - Includes an **authorization identifier** and **descriptors** for each element.
- Schema elements:
  - Include **tables**, **constraints**, **views**, **domains**, and **other constructs**.
  - Each statement in SQL ends with a **semicolon**.
- Foreign Key Errors:
  - A foreign key is a column or a set of columns in a table that is used to establish a link between the data in two tables.
  - The main issue that can arise with foreign keys is when you try to insert a value into a table that does not exist in the table that the foreign key references. This will result in an error.
- Constraints:
  - Used to limit the type of data that can be inserted into a table.
  - **Key constraint:** Primary key value cannot be duplicated.
  - **Entity integrity constraint:** Primary key value cannot be null.
  - **Referential integrity constraint:** The "foreign key " must have a value that is already present as a primary key, or may be null.
- Basic Data Types:
  - **Numeric**:
    - . *Integer numbers:* INTEGER, INT, and SMALLINT.
    - . *Floating-point (real) numbers:* FLOAT or REAL, and DOUBLE PRECISION.
  - **Character-string:**
    - . Fixed length: CHAR(*n*), CHARACTER(*n*)
    - . Varying length: VARCHAR(*n*), CHAR VARYING(*n*), CHARACTER VARYING(*n*)
  - **Boolean:**
    - . Values of TRUE or FALSE or NULL
  - **DATE:**
    - . Ten positions
    - . Components are YEAR, MONTH, and DAY in the form YYYY-MM-DD
    - . Multiple functions available in RDBMSs to change date formats
- Additional data types:
  - **Timestamp:**
    - . Includes the DATE and TIME fields
    - . Plus a minimum of six positions for decimal fractions of seconds
    - . Optional WITH TIME ZONE qualifier.
  - **INTERVAL:**
    - . Specifies a relative value that can be used to increment or decrement an absolute value of a date, time, or timestamp.
- **DATE, TIME, Timestamp, INTERVAL:**
  - Can be **cast** or converted to string formats for comparison.

CREATE :
- *This command is used to create a new table or database.*
- *For example, to create a new database with a table named "Employees", you would use:*

```
CREATE DATABASE myDatabase;

CREATE TABLE Employees (
   ID INT PRIMARY KEY,
   name VARCHAR(100),
   age INT
);
```

ALTER:
- *You can add, modify, or delete columns.*
- *In addition, ALTER TABLE can also be used to add and drop constraints and indexes.*
- *For example, to add a new column named "Email" to the "Employees" table, you would use:*

```
ALTER TABLE Employees
ADD email VARCHAR(255);

ALTER TABLE Employees
MODIFY COLUMN age SMALLINT;

ALTER TABLE Employees
DROP COLUMN email;

ALTER TABLE Employees
DROP CONSTRAINT CHK_Age
CASCADE;

ALTER TABLE Employees
DROP INDEX idx_name;
```

DROP:
- *This command is used to delete an existing database or object.*
- *For example, to delete the "Employees" table, you would use:*

```
DROP TABLE Employees;

DROP DATABASE myDatabase;
```

CASCADE and **RESTRICT** are options you can use with **DROP** and **ALTER**

CASCADE:
- *If used with DROP TABLE, it will drop the table and all its dependent tables.*
- *With ALTER TABLE, it will drop the column and any referential integrity constraint.*

```
DROP TABLE Employees CASCADE;

ALTER TABLE Employees DROP COLUMN
age CASCADE;
```

RESTRICT:
- *It is the opposite of CASCADE.*
- *If used with DROP TABLE or ALTER TABLE, SQL will check that there are no dependencies before allowing the operation.*

```
DROP TABLE Employees RESTRICT;

ALTER TABLE Employees DROP COLUMN
age RESTRICT;
```

DML 1/2
(DATA MANIPULATION LANGUAGE)

Insert

- This statement is used to insert new records into a table.

It can include the following clauses:
- *- VALUES: Specifies the values of the new record.*
- *- INTO: Specifies the table where the new records will be inserted.*

```
INSERT INTO Employees (Name, Age, Salary)
VALUES ('Juan, 30, 50000);
```

Update

- This statement is used to modify existing records in a table. It can include the following clauses:

- *SET: Specifies the column to update and the new value.*
- *WHERE: Specifies a condition for which records to update.*

```
UPDATE Employees SET Salary = 60000
WHERE Name = 'Juan;
```

Delete

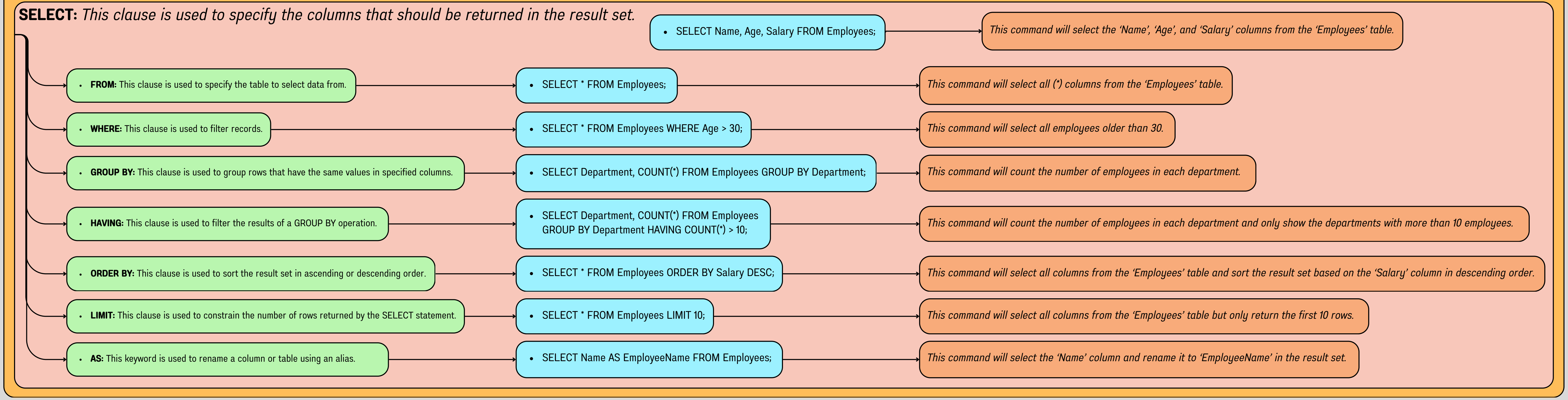
- This statement is used to delete existing records in a table. It can include the following clause:

- *WHERE: Specifies a condition for which records to delete.*

```
DELETE FROM Employees WHERE Name = 'Juan';
```

DML 2/2
(DATA MANIPULATION LANGUAGE)

Basic querying



SELECT: *This clause is used to specify the columns that should be returned in the result set.*

- SELECT Name, Age, Salary FROM Employees; → *This command will select the 'Name', 'Age', and 'Salary' columns from the 'Employees' table.*

- **FROM:** This clause is used to specify the table to select data from. → SELECT * FROM Employees; → *This command will select all (*) columns from the 'Employees' table.*
- **WHERE:** This clause is used to filter records. → SELECT * FROM Employees WHERE Age > 30; → *This command will select all employees older than 30.*
- **GROUP BY:** This clause is used to group rows that have the same values in specified columns. → SELECT Department, COUNT(*) FROM Employees GROUP BY Department; → *This command will count the number of employees in each department.*
- **HAVING:** This clause is used to filter the results of a GROUP BY operation. → SELECT Department, COUNT(*) FROM Employees GROUP BY Department HAVING COUNT(*) > 10; → *This command will count the number of employees in each department and only show the departments with more than 10 employees.*
- **ORDER BY:** This clause is used to sort the result set in ascending or descending order. → SELECT * FROM Employees ORDER BY Salary DESC; → *This command will select all columns from the 'Employees' table and sort the result set based on the 'Salary' column in descending order.*
- **LIMIT:** This clause is used to constrain the number of rows returned by the SELECT statement. → SELECT * FROM Employees LIMIT 10; → *This command will select all columns from the 'Employees' table but only return the first 10 rows.*
- **AS:** This keyword is used to rename a column or table using an alias. → SELECT Name AS EmployeeName FROM Employees; → *This command will select the 'Name' column and rename it to 'EmployeeName' in the result set.*

Advanced Querying

Summary Queries

COUNT: This function returns the number of rows that matches a specified criterion.

- SELECT COUNT(*) FROM Employees;

This command will return the total number of rows in the 'Employees' table.

SUM: This function returns the total sum of a numeric column.

- SELECT SUM(Salary) FROM Employees;

This command will return the total sum of the 'Salary' column in the 'Employees' table.

AVG: This function returns the average value of a numeric column.

- SELECT AVG(Salary) FROM Employees;

This command will return the average salary of all employees.

MAX: This function returns the highest value in a numeric column.

- SELECT MAX(Salary) FROM Employees;

This command will return the highest salary among all employees.

MIN: This function returns the lowest value in a numeric column.

- SELECT MIN(Salary) FROM Employees;

This command will return the lowest salary among all employees.

Subqueries

- A subquery is a query nested inside another query.
- It can return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Comparators

SQL uses **comparison operators** such as =, <>, >, <, >=, <= in the WHERE clause. → SELECT * FROM Employees WHERE Salary >= 50000;

Also we can use this **ADVANCED** comparators:

IN: Allows you to specify multiple values in a WHERE clause.
- SELECT * FROM Employees WHERE Department IN ('Sales', 'Marketing');

Select all employees in the Sales or Marketing depts.

BETWEEN: Values within a given range. Can be numbers, text, or dates.
- SELECT * FROM Employees WHERE Salary BETWEEN 50000 AND 60000;

Select all employees with a salary between 50000 and 60000.

DISTINCT: Only distinct (different) values.
- SELECT DISTINCT Department FROM Employees;

Select only the distinct values from 'Department'.

LIKE: Used to search for a specified pattern in a column.
-Often in conjunction with:
- %: For zero, one, or multiple characters.
- _: For a single character.
- SELECT * FROM Employees WHERE Name LIKE 'J%';

Select all employees whose name starts with "J".

IS NULL / NOT NULL: Used to test for empty (NULL) or non-empty (NOT NULL) values.
- SELECT * FROM Employees WHERE Department IS NULL;

Select all employees with no department assigned.
- SELECT * FROM Employees WHERE Department IS NOT NULL;

Select all employees with a department assigned.

EXISTS / NOT EXISTS: EXISTS is used to test the existence of any record in a subquery.
NOT EXISTS tests for absence.
- SELECT * FROM Departments WHERE EXISTS (SELECT * FROM Employees WHERE Departments.DepartmentId = Employees.DepartmentId);

Select all depts with at least one employee.
- SELECT * FROM Departments WHERE NOT EXISTS (SELECT * FROM Employees WHERE Departments.DepartmentId = Employees.DepartmentId);

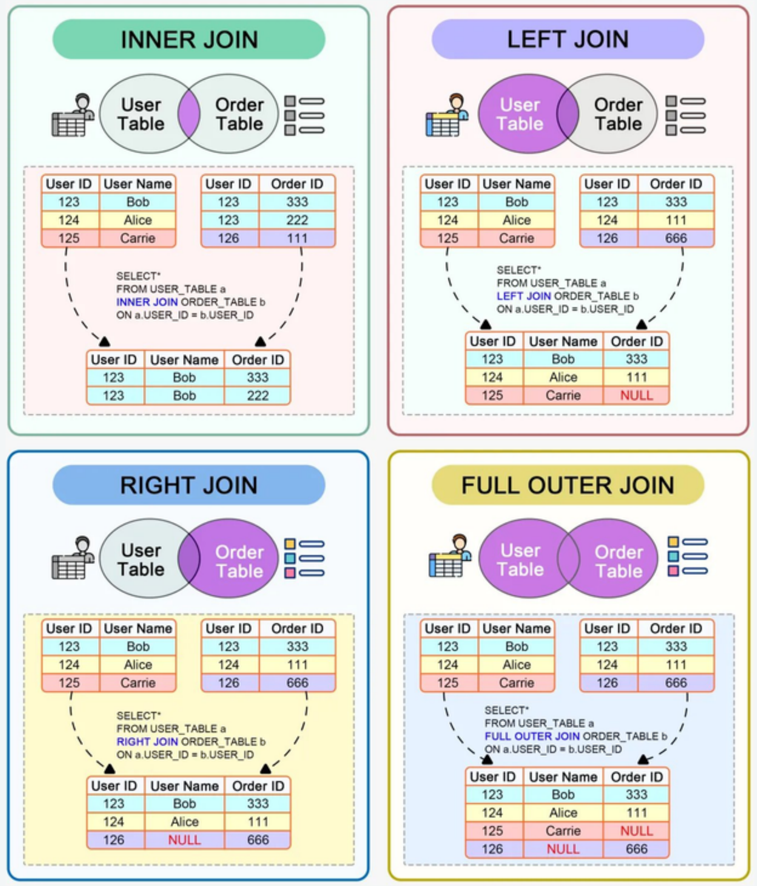
Select all departments with no employees.

ALL / ANY:
ALL is used to compare a value to all values in another value set.
ANY is used to compare a value to any values in another value set.
- SELECT * FROM Employees WHERE Salary > ALL (SELECT Salary FROM Employees WHERE Department = 'Sales');

Select all employees who earn more than all the employees in sales.
- SELECT * FROM Employees WHERE Salary > ANY (SELECT Salary FROM Employees WHERE Department = 'Sales');

Select all employees who earn more than any employee in sales.

JOIN Clauses: The JOIN clause is used to combine rows from two or more tables, based on a related column between them. There are several types of JOINs:



INNER JOIN

User ID	User Name
123	Bob
124	Alice
125	Carrie

User ID	Order ID
123	333
123	222
126	111

```
SELECT*
FROM USER_TABLE a
INNER JOIN ORDER_TABLE b
ON a.USER_ID = b.USER_ID
```

User ID	User Name	Order ID
123	Bob	333
123	Bob	222

LEFT JOIN

User ID	User Name
123	Bob
124	Alice
125	Carrie

User ID	Order ID
123	333
124	111
126	666

```
SELECT*
FROM USER_TABLE a
LEFT JOIN ORDER_TABLE b
ON a.USER_ID = b.USER_ID
```

User ID	User Name	Order ID
123	Bob	333
124	Alice	111
125	Carrie	NULL

RIGHT JOIN

User ID	User Name
123	Bob
124	Alice
125	Carrie

User ID	Order ID
123	333
124	111
126	666

```
SELECT*
FROM USER_TABLE a
RIGHT JOIN ORDER_TABLE b
ON a.USER_ID = b.USER_ID
```

User ID	User Name	Order ID
123	Bob	333
124	Alice	111
126	NULL	666

FULL OUTER JOIN

User ID	User Name
123	Bob
124	Alice
125	Carrie

User ID	Order ID
123	333
124	111
126	666

```
SELECT*
FROM USER_TABLE a
FULL OUTER JOIN ORDER_TABLE b
ON a.USER_ID = b.USER_ID
```

User ID	User Name	Order ID
123	Bob	333
124	Alice	111
125	Carrie	NULL
126	NULL	666