* **SELECT & FROM:** Think of SELECT as *what* you want and FROM as *where* you get it from. The most fundamental operation is asking for data, which we do with the SELECT statement.  
  SELECT FirstName, LastName FROM Employees;  
  This command means 'I want the first name and last name from the Employees table.'
* **\* (The Wildcard):** "The asterisk is a shortcut for 'all columns'. It's great for quickly exploring a new table, but for efficiency, it's better to specify the exact columns you need.

-- Example for SELECT and FROM: Get specific columns from the Employees table.  
SELECT FirstName, LastName, JobTitle  
FROM Employees;  
  
-- Example for \*: Get every single column from the Employees table.  
SELECT \*  
FROM Employees;

Getting all data is rarely useful. We need to filter it. The WHERE clause acts like a filter, letting only the rows that match our conditions pass through.

**WHERE (Basic Condition):** This sets up the filtering rule.

-- Example for WHERE: Get only the employees who work in the 'Sales' department.  
SELECT FirstName, Salary  
FROM Employees  
WHERE Department = 'Sales';

* **AND, OR, NOT (Logical Operators):** These let us build complex rules.

AND requires all conditions to be true

R requires just one to be true

NOT reverses a condition.

-- Example for AND: Employees in 'Sales' AND have a salary over 60000.  
SELECT FirstName, Department, Salary  
FROM Employees  
WHERE Department = 'Sales' AND Salary > 60000;  
  
-- Example for OR: Employees in 'Sales' OR in 'Marketing'.  
SELECT FirstName, Department  
FROM Employees  
WHERE Department = 'Sales' OR Department = 'Marketing';  
  
-- Example for NOT: Employees who are NOT in the 'Sales' department.  
SELECT FirstName, Department  
FROM Employees  
WHERE NOT Department = 'Sales';

* **BETWEEN:** A clean shortcut for checking if a value is within a range, including the endpoints.  
  -- Example for BETWEEN: Employees with salaries between 50000 and 70000.  
  SELECT FirstName, Salary  
  FROM Employees  
  WHERE Salary BETWEEN 50000 AND 70000;
* **IN:** Another great shortcut for checking if a value matches any value in a list. Much cleaner & easier than multiple ORs

-- Example for IN: Employees in the 'HR', 'IT', or 'Finance' departments.  
SELECT FirstName, Department  
FROM Employees  
WHERE Department IN ('HR', 'IT', 'Finance');

* **IS:** Data can be missing. We can't use “= NULL” because NULL represents the absence of a value. We must use the special IS NULL or IS NOT NULL syntax.  
  -- Example for IS NULL: Find employees who do not have a manager assigned.  
  SELECT FirstName, ManagerID  
  FROM Employees  
  WHERE ManagerID IS NULL;
* **LIKE, %, \_ (Pattern Matching):** LIKE is for searching within text data. The percent sign % matches any sequence of characters, while the underscore \_ matches exactly one character.

-- Example for LIKE and %: Find employees whose last name starts with 'S'.  
SELECT LastName  
FROM Employees  
WHERE LastName LIKE 'S%';  
  
-- Example for LIKE and \_: Find employees with a first name like 'Jon' or 'Jan'.  
SELECT FirstName  
FROM Employees  
WHERE FirstName LIKE 'J\_n';

* **DISTINCT:** Use this to remove duplicate values from your result set. If you want a clean list of all job titles, you don't want 'Sales associate’ listed 20 times.  
  -- Example for DISTINCT: Get a unique list of all Designation in the company.  
  SELECT DISTINCT Designation  
  FROM Employees;
* **ORDER BY:** This clause sorts your results. You can sort by one or more columns.  
  -- Example for ORDER BY: List employees alphabetically by their last name.  
  SELECT FirstName, LastName  
  FROM Employees  
  ORDER BY LastName;
* **ASC (Ascending):** This sorts from A-Z or lowest to highest number. It's the default, so you often don't need to type it, but it's good to be explicit.  
  -- Example for ASC: List employees by their hire date, from oldest to newest.  
  SELECT FirstName, HireDate  
  FROM Employees  
  ORDER BY HireDate ASC;
* **DESC (Descending):** This sorts from Z-A or highest to lowest number. We must specify this one.  
  -- Example for DESC: Show employees ordered by salary, from highest to lowest.  
  SELECT FirstName, Salary  
  FROM Employees  
  ORDER BY Salary DESC;
* **LIMIT:** This restricts the output to a specific number of rows. It's applied after the ORDER BY, making it perfect for 'top N' or ‘bottom N’ or ‘Worst/Best N’ style questions.  
  -- Example for LIMIT: Find the 10 highest-paid employees.  
  SELECT FirstName, Salary  
  FROM Employees  
  ORDER BY Salary DESC  
  LIMIT 10;
* **Scalar Functions (UPPER, LOWER, YEAR):** These perform an action on every single row individually.  
  -- Example for UPPER: Display all last names in uppercase.  
  SELECT UPPER(LastName)  
  FROM Employees;  
    
  -- Example for LOWER: Display all email addresses in lowercase.  
  SELECT LOWER(Email)  
  FROM Employees;  
    
  -- Example for YEAR: Show the year each employee was born.  
  SELECT FirstName, YEAR(BirthDate) AS BirthYear  
  FROM Employees;
* **Aggregate Functions:** These functions condense many rows into a single summary value.
  + **COUNT:** Counts the number of rows. COUNT(\*) counts all rows, while COUNT(column) counts non-null values in that column.  
    -- Example for COUNT: How many employees are in the company?  
    SELECT COUNT(\*) AS TotalEmployees  
    FROM Employees;
  + **SUM:** Adds up all the values in a numeric column.  
    -- Example for SUM: What is the total payroll for the company?  
    SELECT SUM(Salary) AS TotalPayroll  
    FROM Employees;
  + **AVG:** Calculates the average of a numeric column.  
    -- Example for AVG: What is the average employee salary?  
    SELECT AVG(Salary) AS AverageSalary  
    FROM Employees;
  + **MIN & MAX:** Find the minimum and maximum values in a column.  
    -- Example for MIN: What is the lowest salary in the company?  
    SELECT MIN(Salary) AS LowestSalary  
    FROM Employees;  
      
    -- Example for MAX: What is the highest salary in the company?  
    SELECT MAX(Salary) AS HighestSalary  
    FROM Employees;

We discussed in class how to find out the max/min value without using Max/Min command

[Hint: use of limit]

* **GROUP BY:** It groups rows that have the same values into summary rows.

The laundry analogy: this sorts your clothes into piles by color before you count what's in each pile.  
-- Example for GROUP BY: Count the number of employees in each department.  
SELECT Department, COUNT(\*) AS NumberOfEmployees  
FROM Employees  
GROUP BY Department;

* **HAVING:** This is the filter for your groups. WHERE filters rows *before* grouping; HAVING filters groups *after* they are created.  
  -- Example for HAVING: Show only departments that have more than 10 employees.  
  SELECT Department, COUNT(\*) AS NumberOfEmployees  
  FROM Employees  
  GROUP BY Department  
  HAVING COUNT(\*) > 10;
* **Subqueries (Nested Queries):** A subquery is a SELECT statement inside another statement. The inner query runs first, providing a value for the outer query to use.  
  -- Example for Subquery: Find all employees who earn more than the company's average salary.  
  SELECT FirstName, Salary  
  FROM Employees  
  WHERE Salary > (SELECT AVG(Salary) FROM Employees);
* **ANY & ALL:** These operators are used with subqueries that return a list of values.

-- Example for ANY: Find employees who earn more than ANY person in the 'Intern' role.  
-- (i.e., more than the lowest-paid intern)  
SELECT FirstName, Salary  
FROM Employees  
WHERE Salary > ANY (SELECT Salary FROM Employees WHERE JobTitle = 'Intern');

[This command can return other interns details as well. Why?]   
  
-- Example for ALL: Find employees who earn more than ALL people in the 'Intern' role.  
-- (i.e., more than the highest-paid intern)  
SELECT FirstName, Salary  
FROM Employees  
WHERE Salary > ALL (SELECT Salary FROM Employees WHERE JobTitle = 'Intern');

* **INNER JOIN:** This is how we combine rows from two tables based on a related column. Think of a Venn diagram; the INNER JOIN is the overlapping part in the middle, containing only the records that have a match in both tables.  
  -- Example for INNER JOIN: Show each employee's name next to their department's full name.  
  -- (Assumes we have an Employees table and a Departments table linked by DepartmentID)  
  SELECT E.FirstName, D.DepartmentName  
  FROM Employees AS E  
  INNER JOIN Departments AS D ON E.DepartmentID = D.DepartmentID;

For a clear understanding about joins, please check this   
[CSE370 Lab3.pdf](https://drive.google.com/file/d/11Rcl4LvsIrgL_CP4DpQOlzwaVf5GMY4w/)

logical order SQL processes a query, which is different from how we usually write it

1. **FROM / JOIN**: Gets the tables.
2. **WHERE**: Filters the rows.
3. **GROUP** BY: Groups the filtered rows.
4. **HAVING**: Filters the groups.
5. **SELECT**: Selects the final columns/calculations.
6. **DISTINCT**: Removes duplicates.
7. **ORDER BY**: Sorts the final result.
8. **LIMIT**: Restricts the number of rows returned.