SQL*Plus vs. Oracle SQL Developer

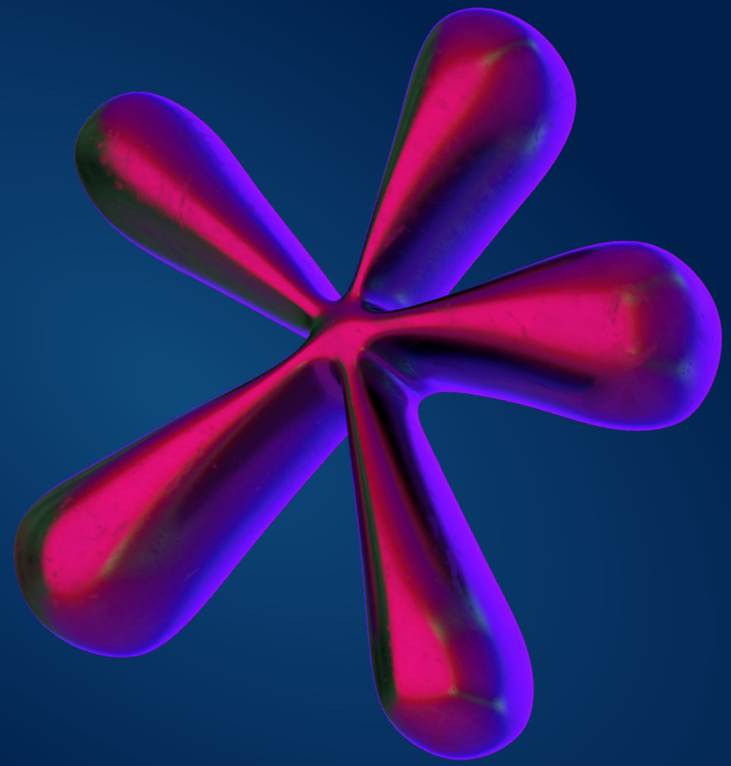
- SQL*Plus
  - A command-line interface provided by Oracle to connect to, and disconnect from, any Oracle database server.
  - Runs both SQL statements and PL/SQL blocks interactively or via script files.
  - Features an in-memory buffer for your most recently executed statement or block, plus editing, scripting, and powerful report-formatting commands .
- Oracle SQL Developer
  - A free, standalone graphical tool for database development and administration.
  - Lets you browse database objects, run and debug SQL/PL/SQL, and design reports—all from a GUI with two main tabs: Connections and Reports.
  - Supports connection management, drag-and-drop object creation, built-in code insight, data modeling, version control integration, and visual PL/SQL debugging .

Productivity Boosts in Oracle SQL Developer

Oracle SQL Developer provides a modern GUI on top of the Oracle database. Key productivity features include:

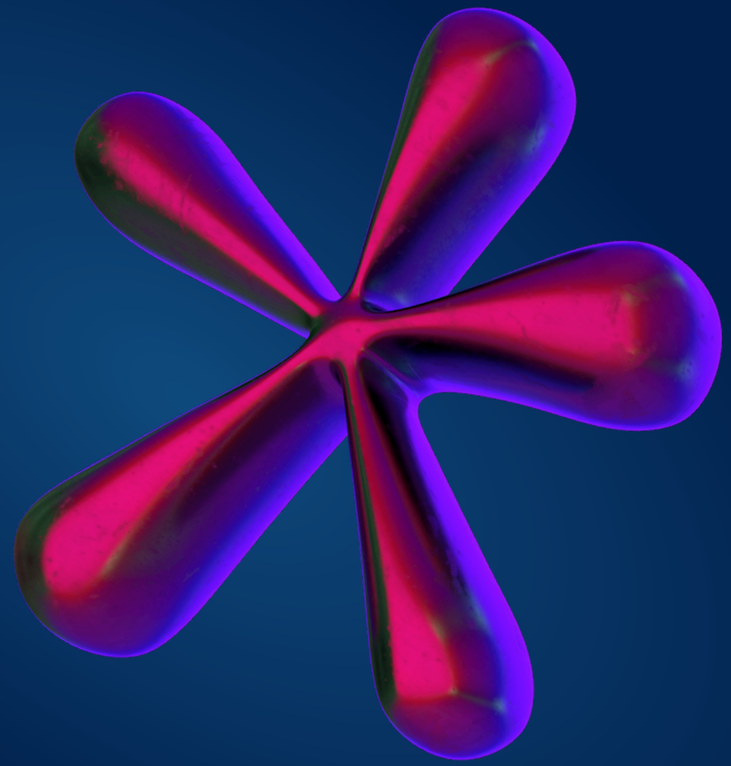
- Object Browser
  - Drill into schemas, tables, views, packages, etc., via a tree-view and right-click to generate DDL or perform data edits.
- SQL Worksheet & Code Insight
  - Write SQL/PLSQL with syntax highlighting and context-aware code completion (“Ctrl+Space”), snippets, and inline help.
- Visual PL/SQL Debugger

Set breakpoints, step through procedures/functions, and inspect variable values - all graphically.



Productivity Boosts in Oracle SQL Developer

- Data Modeling & Reports
  - Build ER diagrams, design custom reports, and schedule them.
- Version Control Integration
  - Connect projects to Git, SVN, or CVS directly within the IDE.
- Migration & Utilities
  - Wizard-based schema migrations, data pump exports/imports, and real-time monitoring dashboards.



Formulating and Using DDL Statements:

Statement	Purpose	Example
CREATE TABLE	Define a new table	CREATE TABLE dept(id NUMBER, name VARCHAR2(30));
ALTER TABLE	Modify table structure (add/drop column, etc.)	ALTER TABLE dept ADD (location VARCHAR2(20));
DROP TABLE	Remove a table and its data permanently	DROP TABLE dept;
TRUNCATE TABLE	Quickly delete all rows (cannot be rolled back)	TRUNCATE TABLE dept;
RENAME	Change the name of a schema object	RENAME dept TO departments;

Formulating and Using DML Statements:

Statement	Purpose	Example
SELECT	Retrieve data from one or more tables	SELECT name FROM dept WHERE id = 10;
INSERT	Add new rows	INSERT INTO dept VALUES(30, 'SALES');
UPDATE	Modify existing rows	UPDATE dept SET name='MARKETING' WHERE id=30;
DELETE	Remove rows	DELETE FROM dept WHERE id=30;
MERGE	UPSERT: combine insert & update logic	

1. Aliases

What & Why

Column alias: gives a temporary name to a column or expression, making output more readable.

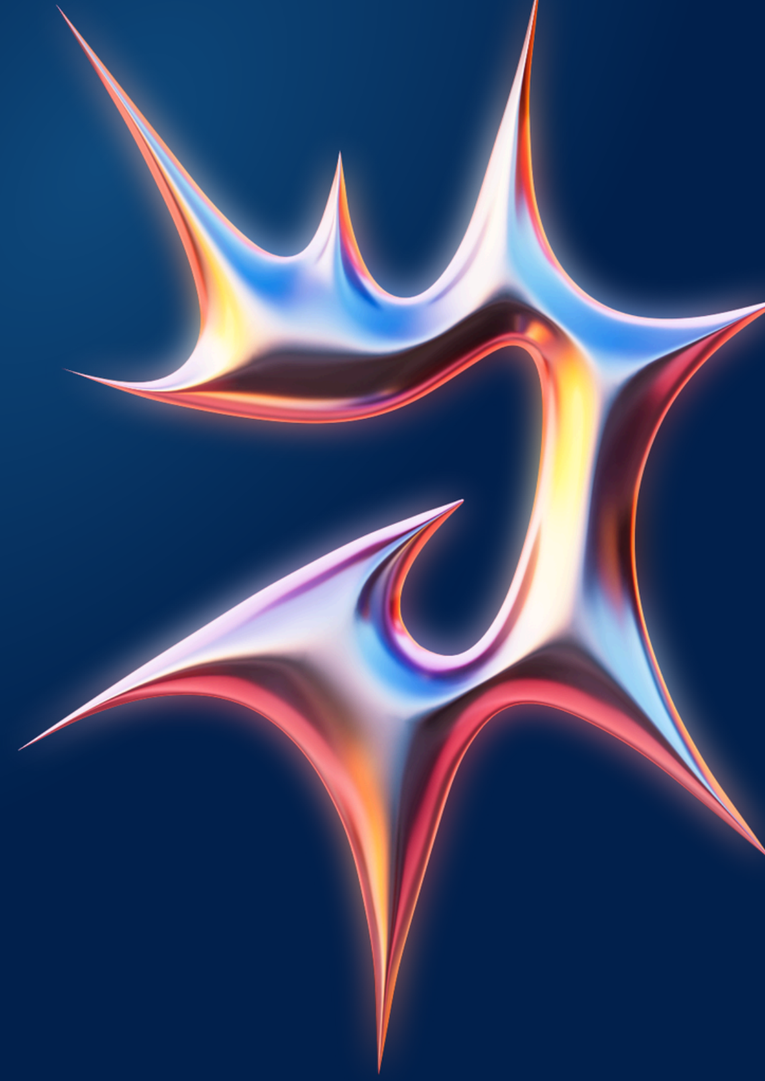
Table alias: assigns a short name to a table reference, simplifying joins and self-joins.

Example:

```
SELECT
  column_name AS alias_name,
  (salary * 1.1)    AS adjusted_salary
FROM employees AS e;
```

Why use it?

- Improves report readability.
- Shortens long table names in complex queries.
- Helps disambiguate columns when joining multiple tables.

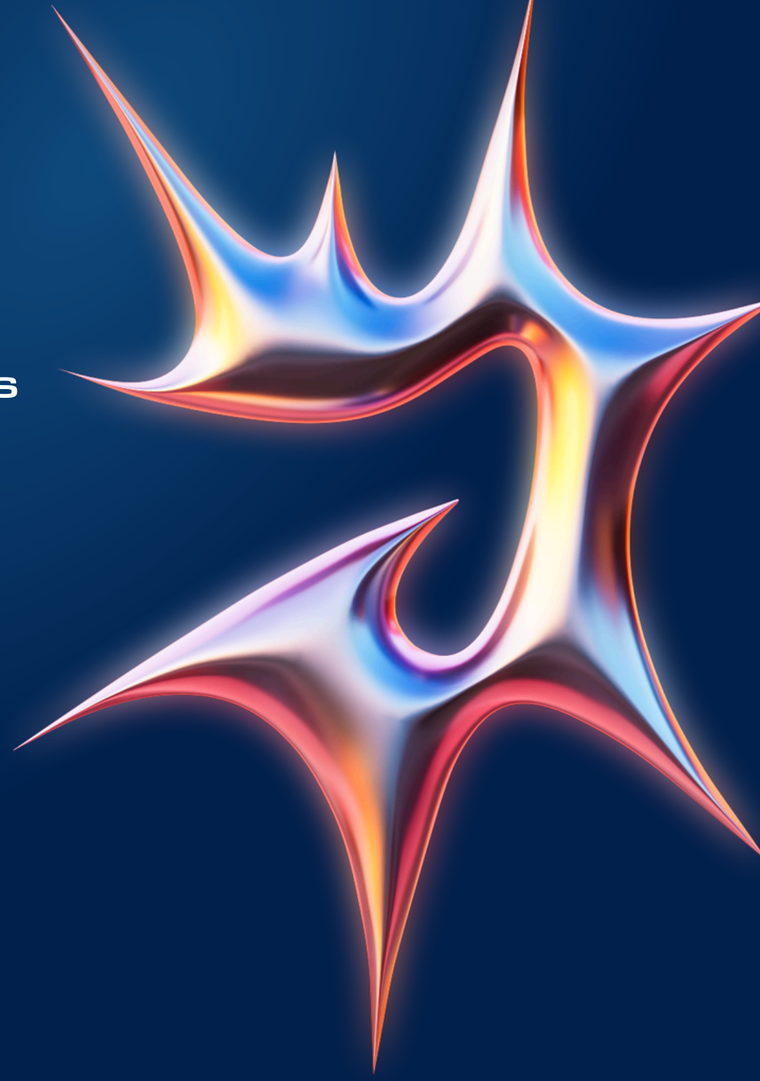


2. Set Operators

What & Why

Set operators let you combine the results of two (or more) SELECT queries that have the same number and compatible types of columns.

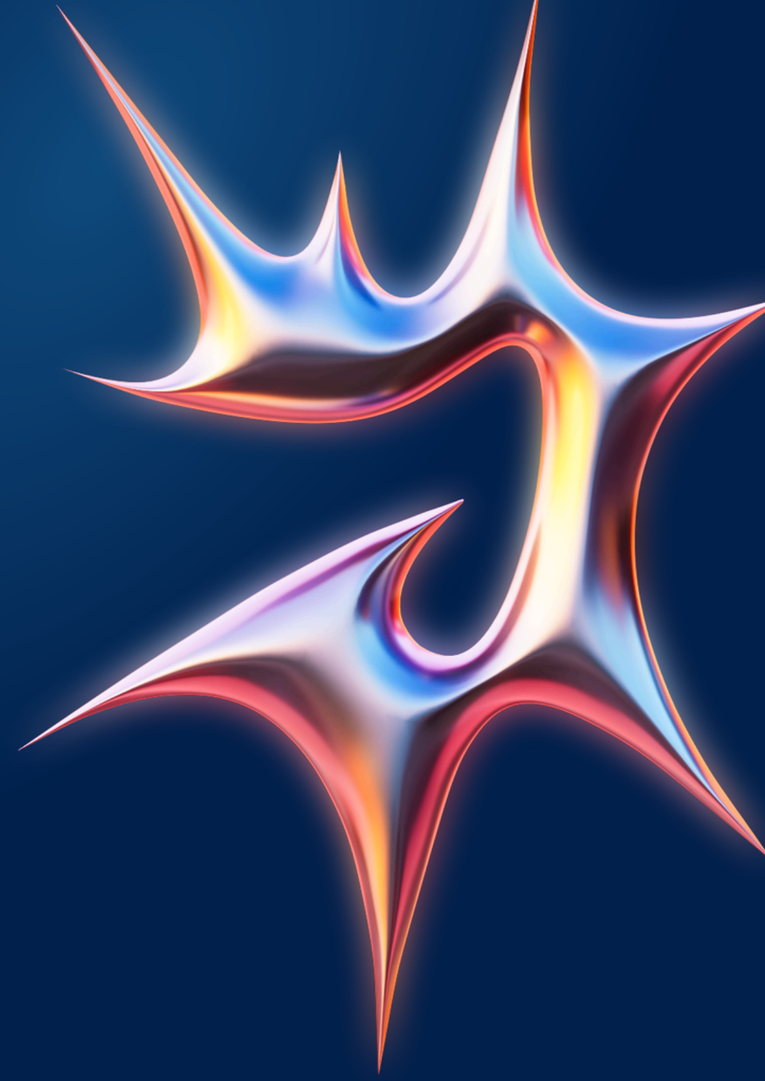
Operator	Effect
UNION	Concatenates results, **removes** duplicates.
UNION ALL	Concatenates results, **keeps** duplicates.
INTERSECT	Returns only rows **common** to both queries.
MINUS	Returns rows in the first query **not** in the second.



2. Set Operators

Example:

```
-- Employees who work in Dept 10 or Dept 20
SELECT employee_id, department_id
  FROM employees
 WHERE department_id = 10
UNION
SELECT employee_id, department_id
  FROM employees
 WHERE department_id = 20;
```



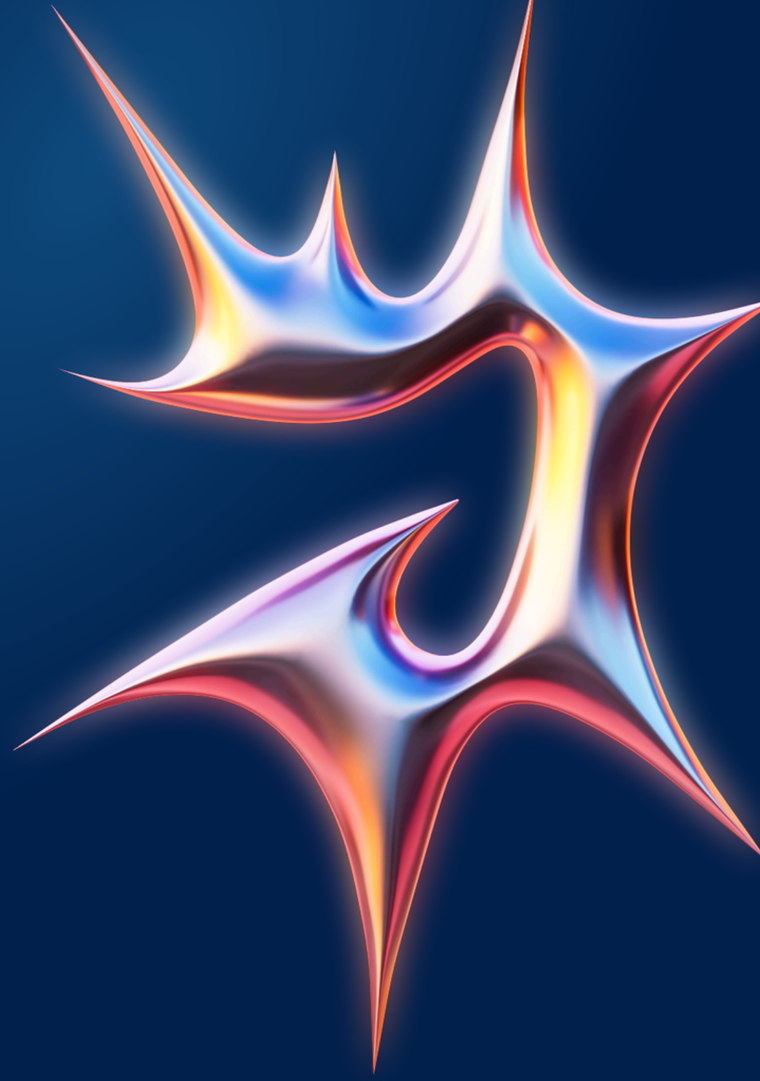
3. Subqueries

What & Why?

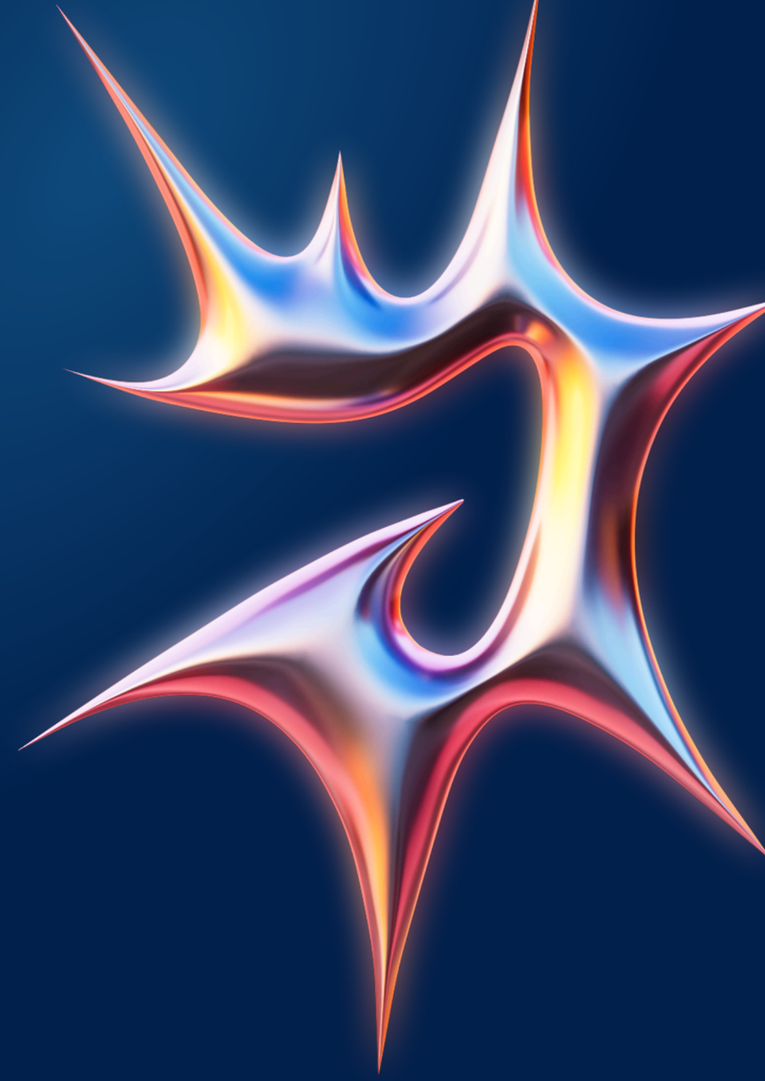
A subquery is a query nested inside another SQL statement. It can appear in the SELECT, FROM, WHERE, or HAVING clauses.

- Non-correlated subquery runs once, feeding results into the outer query.
- Correlated subquery runs once per row of the outer query (can reference outer columns).

```
-- 1. Scalar subquery in SELECTSELECT
  department_id,
  (SELECT AVG(salary)
     FROM employees
    WHERE department_id = d.department_id
  ) AS avg_dept_salary
FROM departments d;
```



3. Subqueries



Why use it?

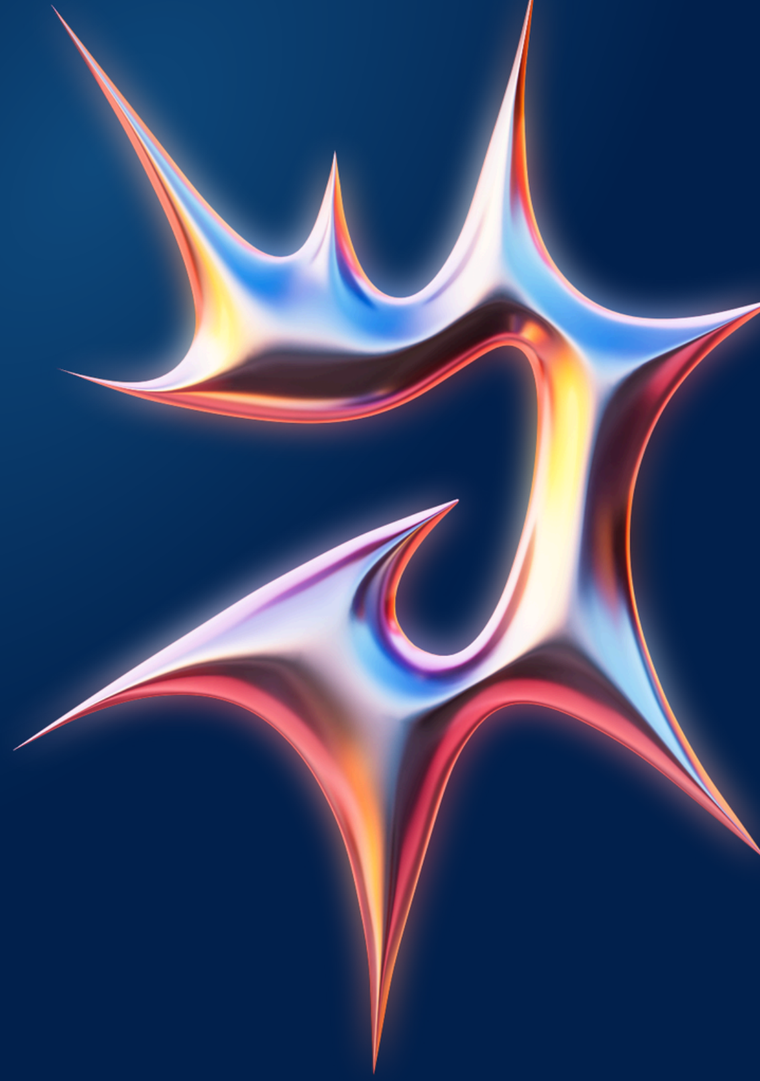
- Break complex logic into manageable pieces.
- Filter or compute values based on aggregates or conditions in other tables.
- Avoid multiple round-trips by embedding logic directly.

4. Joins

What & Why?

Joins combine rows from two or more tables based on related columns. They're fundamental to querying normalized schemas.

Join Type	Description
INNER JOIN	Returns only matching rows in both tables.
LEFT OUTER JOIN	All rows from the left table, with matching right rows.
RIGHT OUTER JOIN	All rows from the right table, with matching left rows.
FULL OUTER JOIN	All rows from both tables, matching where possible.
CROSS JOIN	Cartesian product (every row × every row).



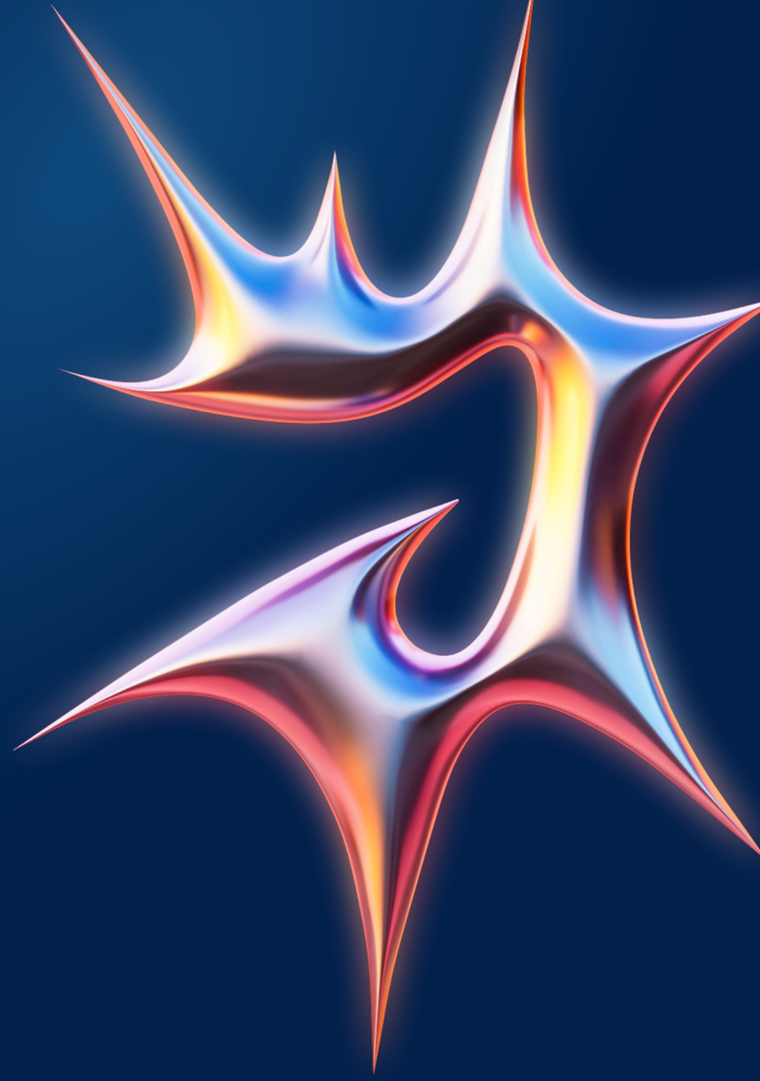
4. Joins

What & Why?

Joins combine rows from two or more tables based on related columns. They’re fundamental to querying normalized schemas.

Why use it?

- Relate data across tables (e.g., employees ↔ departments, orders ↔ customers).
- Control whether you want only matches (INNER) or also unmatched rows (OUTER).
- Build rich reports that pull descriptive and transactional data together.



Various functions supported by the GROUP BY function:

Basic & Statistical Aggregates:

Function	Description
COUNT(*)	Counts all rows in the group
COUNT(expr)	Counts non-NULL occurrences of expr
COUNT(DISTINCT expr)	Counts distinct non-NULL values of expr
SUM(expr)	Sums up numeric expr across the group
AVG(expr)	Calculates the average of numeric expr
MIN(expr)	Finds the minimum value of expr
MAX(expr)	Finds the maximum value of expr
STDDEV(expr)	Sample standard deviation of expr

Various functions supported by the GROUP BY function:

Advanced & Grouping Extensions:

Function / Clause	Description
VARIANCE(expr)	Sample variance of expr
MEDIAN(expr)	Median (middle) value of expr
LISTAGG(expr, ',')	Concatenates expr values into a delimited string
ROLLUP(col1,...,colN)	Produces hierarchical subtotals and a grand total
CUBE(col1,...,colN)	Generates all combinations of subtotals across listed columns
GROUPING(expr)	Returns 1 if expr is aggregated by ROLLUP/CUBE in this row, else 0
GROUPING_ID(col1,...,colN)	Bitmask indicating which columns are rolled up for this row


DCL
DATA
CONTROL
LANGUAGE

Statement	Purpose	Syntax	Typical Use
COMMIT	Make all changes in the current transaction permanent and release locks	COMMIT;	After a logical unit of work completes successfully (e.g., inserting an order and its line-items)
ROLLBACK	Undo all changes in the current transaction back to the last commit or to a savepoint; release locks	ROLLBACK;ROLLBACK TO SAVEPOINT savept;	When an error or constraint violation occurs, to revert to a known good state without partial updates
SAVEPOINT	Define a named marker within a transaction to which you can rollback	SAVEPOINT savept;	Before a critical phase in a long transaction, so you can roll back only that segment if something goes wrong

ICE Task:

Create an infographic that compares DDL, DML, DCL and TCL statements

Cover -

What they are
Why we use them
Your OWN examples.