SQL Cheat Sheet

1. Basic Query Structure

```
SELECT column1, column2, ...
FROM schema.table AS t
WHERE conditions
GROUP BY column1, ...
HAVING aggregate_condition
ORDER BY column1 [ASC|DESC]
LIMIT n;
```

2. JOINs

Type	Syntax	Description
INNER	… FROM A JOIN B ON A.key = B.key	Only matching rows
LEFT	… FROM A LEFT JOIN B ON A.key = B.key	All A, matching B (NULL if no match)
RIGHT	… FROM A RIGHT JOIN B ON A.key = B.key	All B, matching A
FULL	… FROM A FULL OUTER JOIN B ON A.key = B.key	All rows from both, NULL where no match
CROSS	… FROM A CROSS JOIN B	Cartesian product

3. Aggregations & GROUPING SETS

```
SELECT
  country,
  COUNT(*) AS total_orders,
  SUM(amount) AS revenue,
```

```
  AVG(amount) AS avg_order
FROM orders
GROUP BY country
WITH ROLLUP;
```

4. Common Window Functions

Function	Example	Notes
ROW_NUMBER()	ROW_NUMBER() OVER (PARTITION BY dept ORDER BY salary DESC)	Unique sequential per partition
RANK()	RANK() OVER (PARTITION BY dept ORDER BY salary DESC)	Ties share rank, gaps after ties
DENSE_RANK()	DENSE_RANK() OVER (PARTITION BY dept ORDER BY salary DESC)	Ties share rank, no gaps
NTILE(n)	NTILE(4) OVER (ORDER BY revenue DESC)	Divides rows into n buckets
LAG(expr,1)	LAG(amount,1) OVER (ORDER BY order_date)	Access previous row
LEAD(expr,1)	LEAD(amount,1) OVER (ORDER BY order_date)	Access next row
FIRST_VALUE()	FIRST_VALUE(score) OVER (PARTITION BY game ORDER BY time)	First value in window
LAST_VALUE()	LAST_VALUE(score) OVER (PARTITION BY game ORDER BY time ROWS BETWEEN 2 PRECEDING AND 2 FOLLOWING)	Last value, requires explicit frame

5. CTEs & Subqueries

```
-- Common Table Expression
WITH recent_orders AS (
  SELECT * FROM orders WHERE order_date >= CURRENT_DATE - INTERVAL '30' DAY
)
SELECT customer_id, COUNT(*) AS cnt
```

```
FROM recent_orders
GROUP BY customer_id;

-- Correlated Subquery
SELECT
  o.order_id,
  (SELECT AVG(amount) FROM order_details od WHERE od.order_id = o.order_id) AS
avg_item_price
FROM orders o;
```

6. Date & String Functions

Dates: DATE_TRUNC('month', order_date), CURRENT_TIMESTAMP
Intervals: order_date + INTERVAL '7 days', AGE(end, start)
Strings: CONCAT(first, ' ', last), SUBSTRING(col FROM 1 FOR 5)

7. Best Practices

• Use descriptive aliases: e.g., FROM sales.orders AS o JOIN sales.customers AS c ON o.customer_id = c.id

• Always specify schema to avoid ambiguity.

• Format and indent: uppercase keywords, lowercase identifiers.

• Limit data early: push filters (WHERE) before joins if selective.

• Be consistent in style and function usage (e.g., always use DENSE_RANK() if you need no gaps).

SQL Execution Order (Logical)

1. FROM Choose your base table(s)
2. WHERE Filter individual rows
3. GROUP BY Aggregate rows
4. HAVING Filter aggregated groups
5. SELECT Choose output columns
6. ORDER BY Sort final result

Common SQL JOIN Types

INNER JOIN Match only rows present in both tables

LEFT JOIN All rows from the left, matched or not

FULL OUTER JOIN All rows from both, matched or not

CARTESIAN JOIN Every row with every row (avoid!)

Window Function Example

AVG(sales) OVER (

ORDER BY date

ROWS BETWEEN 2 PRECEDING AND 2 FOLLOWING

) AS five_day_avg

SQL SubLanguages

DDL CREATE, ALTER, DROP

DML SELECT, INSERT, UPDATE, DELETE

DCL GRANT, REVOKE

TCL COMMIT, ROLLBACK

Ready to master your data pipelines and avoid costly rank disasters? Visit Gambill Data Engineering at https://gambilldata.com/coaching/or email chris.gambill@gambilldataengineering.com to learn more and get expert support.