SQL CHEAT SHEET

BEGINNER SYNTAX

SELECT	Select the columns
FROM	Which table to pull from
WHERE	Filter the data
GROUP BY	Aggregate the data
HAVING	Filter an aggregate
ORDER BY	Sort the data

AGGREGATIONS

SUM(col1)	Sum a column
COUNT(*)	Count all rows
COUNT(DISTINCT col1)	Count unique rows
AVG(col1)	Average a column
MIN(col1)	Smallest column value
MAX(col1)	Largest column value

INTERMEDIATE

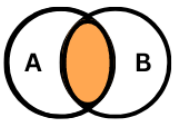
LIKE	Match on a pattern
AND	Both criteria
OR	One or the other
CASE WHEN	If then logic
IN	Filter by a list
UNION ALL	Append data
BETWEEN	Between two items
CAST	Change data type
COALESCE	First non-null value

ADVANCED

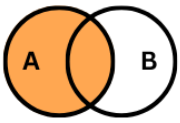
CTEs	Write clean SQL
SUBQUERIES	Nested SELECT
WINDOW FUNCTIONS	Perform calculations across rows

JOINS

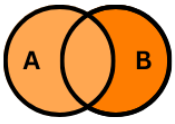
INNER JOIN



LEFT JOIN



FULL JOIN



EXAMPLE PATTERNS

Select Columns Filtered on Criteria

```
SELECT *
FROM orders
WHERE status = 'paid'
  AND date BETWEEN '2023-01-01' and '2023-03-31'
  AND email LIKE '%@thequery.io'
```

Explore Column Values

```
SELECT
  status, COUNT(*) as num
FROM orders
GROUP BY status
ORDER BY num DESC
```

Common Aggregations

```
SELECT
  COUNT(*) as num_rows,
  MIN(date) as oldest_date,
  AVG(revenue) as avg_rev
FROM orders
```

Research Duplicates w/ Subquery

```
SELECT *
FROM orders
WHERE order_id IN (
    SELECT order_id
    FROM orders
    GROUP BY order_id
    HAVING COUNT(*) > 1)
```

If Then Logic

```
SELECT
  *,
  CASE WHEN revenue < 0
      THEN 1 ELSE 0
      END AS is_refund
FROM orders
```

Joins

```
SELECT
  o.*,
  c.phone_number
FROM orders o
LEFT JOIN customers c
  ON o.customer_id = c.id
```

Unions

```
SELECT *
FROM orders_2022
UNION ALL
SELECT *
FROM orders_2023
```

Change Data Type of Column

```
SELECT
  CAST(sale_date AS DATE),
  CAST(order_id AS INT64)
FROM orders
```

Handle Nulls with Coalesce

```
SELECT
  COALESCE(primary_phone,
    mobile_phone) AS
    phone_number
FROM customers
```

CTEs

```
WITH rev_gt_100 AS (
  SELECT *,
    revenue * sales_tax AS total_amount
  FROM orders
)
SELECT *
FROM rev_gt_100
WHERE total_amount > 100
```

Window Functions

```
SELECT
  *,
  ROW_NUMBER OVER(PARTITION BY type ORDER BY date)  AS idx,
  SUM(revenue) OVER(ORDER BY date) AS running_total_revenue
FROM orders
```