

Analyx Academy

SQL Cheat Sheet — for Data Analyst

Quick reference (must-know syntax)

```
-- select + filters
SELECT col1, col2
FROM schema.table
WHERE col3 = 'X' AND col4 BETWEEN '2025-01-01' AND '2025-12-31'
ORDER BY col1 DESC
LIMIT 100;


-- group + having
SELECT cust_id, COUNT(*) AS orders, SUM(amount) AS total
FROM orders
GROUP BY cust_id
HAVING SUM(amount) > 1000;
```

```
-- joins
SELECT a.id, b.name
FROM customers a
JOIN orders b ON a.id = b.customer_id   -- inner join
LEFT JOIN refunds r ON b.id = r.order_id  -- keep orders without refunds

-- basic window function
SELECT user_id, event_ts,
     ROW_NUMBER() OVER(PARTITION BY user_id ORDER BY event_ts DESC) AS rn
FROM events;

-- common pattern: nth-highest (3rd highest) using DENSE_RANK
SELECT value FROM (
  SELECT value, DENSE_RANK() OVER (ORDER BY value DESC) rnk
  FROM table_name
) t WHERE rnk = 3;
```

Key SQL concepts with one-line notes

- **Joins (inner/left/right/full/cross)** — know when to preserve rows and when to filter them out.
- **Aggregations & GROUP BY** — conditional aggregation (SUM(CASE WHEN ...)) is heavily used in analytics.
- **Window functions** — ROW_NUMBER(), RANK(), DENSE_RANK(), LAG(), LEAD(), SUM() OVER() — essential for sessionization, running totals, cohorts.
- **CTEs (WITH)** — use to structure readability and break complex logic into steps (but know performance trade-offs in your DB).

- **Subqueries vs EXISTS vs IN** — EXISTS often faster than IN for correlated subqueries; know both.
- **Date functions & time bucketing** — DATE_TRUNC, EXTRACT, TO_CHAR — crucial for time-series analysis.
- **String functions** — CONCAT, SUBSTR, REGEXP / LIKE for parsing and cleansing.
- **Transactions & isolation basics** — know BEGIN/COMMIT/ROLLBACK and why locks matter for OLTP vs analytics.

(Modern hiring guides and interview question lists repeatedly emphasize window functions, joins, and aggregation as core skills.)

Performance & production-readiness checklist (what seniors expect)

- Avoid SELECT *. Project only needed columns.
- Push predicates into WHERE (make queries *sargable*). Use indexed columns in filters.
- Use proper joins order and join predicates to reduce intermediate row counts.
- Know how to read an **EXPLAIN/EXPLAIN ANALYZE** output and spot the expensive parts (seq scan vs index scan, nested loop vs hash join).
- Use LIMIT in exploratory queries and TOP / paging for UI queries.
- Prefer CTE for readability but know that some DBs materialize them (affects performance) — know your database behavior (Postgres, BigQuery, Redshift differ).
- Use partitioning or date-sharding for very large tables; delete/archive old partitions. (Performance, explain plans, and indexing are common advanced interview topics.)

Typical SQL interview problems & concise solution patterns

1. **Deduplicate keeping latest record per user**

```
WITH ranked AS (
  SELECT *, ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY updated_at DESC) rn
  FROM users_raw)
 SELECT * FROM ranked WHERE rn = 1;
```

2. **Running total by day**

SELECT day, SUM(amount) OVER (ORDER BY day ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) running_total

FROM daily_revenue;

3. **Cohort retention (monthly)** — partition by user signup month and count activity months using window functions & conditional aggregates.
4. **Identify repeated payments within 10 minutes** — use LAG() or self-join with timestamp difference.
5. **Nth-highest salary** — use DENSE_RANK() or OFFSET FETCH.

Many platforms and blogs show these exact problems as standard screening tasks. Practice these exact patterns.

DB-specific tips (most common stacks)

- **Postgres**: learn EXPLAIN ANALYZE, VACUUM, ANALYZE, index types (btree, gin).
- **BigQuery**: avoid SELECT * (scan cost), use PARTITION and CLUSTER.
- **SQL Server**: know execution plans and statistics updates.
- **Redshift / Snowflake**: understand distribution keys (Redshift) and micro-partitions (Snowflake).
  (Cloud data & analytics tools are hiring hotspots — Databricks, Snowflake and BigQuery are growing footprints.)

How to position yourself & what interviewers expect

Market & role context (short)

- Demand for data analysts remains strong; companies want analysts who convert data into business insights and can handle end-to-end small pipelines (SQL → transform → visualization). Recent industry writeups show rapid growth in analytics hiring and an emphasis on AI/ML-ready data platforms.

What interviewers typically expect from 2–4 yr candidates

- **Solid SQL fluency** — joins, group by, window functions, CTEs; ability to optimize queries.
- **Business-sense + metrics design** — define KPIs, convert ambiguous business questions into measurable metrics.
- **One or two end-to-end projects** — show data ingestion, cleaning, modeling, and dashboarding (Power BI / Tableau).
- **Basic scripting (Python) for data cleaning** — pandas experience is commonly requested.
- **Versioning and collaboration** — familiarity with Git, ticketing (Jira), and communicating with stakeholders.
- **SQL coding test + live problem-solving** — expect 30–60 minute SQL tests (often 2–4 tasks).

Sources and recent job guides emphasize SQL as the foundation plus Power BI/Tableau and some Python.

How to structure your resume / LinkedIn (2–4 yrs)

- Keep bullets short, measurable, and business-oriented. Use this formula: **Action + Tool + Outcome/metric**.
  Examples:
- "Built customer-churn cohort analysis using SQL + Power BI, reducing monthly churn by enabling targeted retention campaigns (identified top 3 risk segments; improved retention by X%)."

- “Optimized ETL query that reduced daily pipeline runtime from 2.5 hrs to 20 mins by adding partition pruning and rewriting heavy joins.”
  Include: tech stack line (SQL, Power BI, Python/pandas, Excel, Cloud (if any), Git).

Interview preparation checklist (practical)

- Master the common SQL patterns (dedupe, running totals, sessionization, nth highest). Practice on LeetCode/InterviewBit/Mode/Datacamp exercises.
- Be ready to explain an **end-to-end project**: data sources, cleaning decisions, metrics definition, dashboard choices, business impact.
- Prepare 3–4 STAR stories (problem, action, result) showing impact, cross-team collaboration, and problem-solving under constraints.
- Mock live-coding: practice writing queries aloud and explaining complexity, indexes, and trade-offs. Interviewers look for your thought process as much as final queries.

Technical growth roadmap (next 6–12 months)

- **Immediate (0–2 months):** polish SQL window functions, EXPLAIN output, and 10 most common analytic queries.
- **Near-term (3–6 months):** build 1-2 end-to-end projects with Power BI/Tableau + SQL backend; include a short case study in resume/portfolio.
- **Medium (6–12 months):** learn a cloud data tool (BigQuery / Snowflake / Databricks) and basic data engineering patterns (partitioning, orchestration) to stand out. Industry investment in AI & data platforms means cloud/Databricks skills are increasingly valuable.

Sample interview questions to practice (2–4 yr level)

- Write a query to find each user’s last purchase and total purchase in last 6 months.
- Given events(user_id, event, ts), compute weekly active users and MoM growth.
- Explain how you’d optimize a slow JOIN between a 10M-row fact table and a 50k-dimension table.
- Create a cohort retention table from signups and event logs.

- Explain a project where your data work changed a business decision (STAR).

Many interview banks and blog posts use these exact prompts — do timed practice.

Projects & portfolio ideas (high ROI)

- **Customer 360 dashboard**: ingestion, dedupe logic, key segments, retention cohorts, LTV estimate.
- **Sales funnel + attribution**: event sequencing, identify drop-off points, recommend experiments.
- **Fraud or duplicate transaction detection**: use window functions + thresholds to flag likely duplicates.
- **Operational reporting automation**: convert manual Excel reports to SQL-backed dashboards with alerts.

Interview-day micro-tips

- Ask clarifying questions (expected outputs, definitions of "active user", timezones). Interviewers value precise assumptions.
- Verbalize your indexing and complexity trade-offs when optimizing.
- When stuck, write a simple, correct query first, then iterate and optimize — many interviewers grade approach as much as final answer.

📍 **Head Office:** Shop no: - 10 Samarth Park Near Aviation College Sinhgad Campus, Ambegaon BK Pune 411046

🌍 **Website:** www.analyxacademy.com 📧 **Email:** info@analyxacademy.com

Contact Us:

Akash Sir: +91 72190 02125

Pratik Sir: +91 90825 54368

Suyash Sir: +91 84688 36850