**SQL - Advanced & Scenario-Based Questions**

1. **Scenario: You have a table with millions of records, and your queries are slow. How would you improve performance?**
   * Use **indexes** on frequently searched columns.
   * Optimize **joins** by using appropriate indexing and avoiding unnecessary computations.
   * Use **partitioning** for large tables to split data logically.
   * Avoid **SELECT**\*; instead, query only necessary columns.
   * Use **query optimization techniques** like Common Table Expressions (CTEs) and indexed views.
   * Perform **database normalization** to reduce redundancy but denormalization for read-heavy queries.
2. **How does indexing impact query performance, and when should you avoid using indexes?**
   * **Impact**: Indexes speed up **SELECT** queries by allowing quick lookups but slow down **INSERT, UPDATE, DELETE** operations due to maintenance overhead.
   * **Avoid**:
     + When data is **frequently updated**.
     + For **small tables** where full table scans are faster.
     + On **columns with low cardinality** (e.g., boolean values).
3. **Write a SQL query to find the second-highest salary from an Employees table.**

sql

CopyEdit

SELECT MAX(salary)

FROM Employees

WHERE salary < (SELECT MAX(salary) FROM Employees);

* + Alternative using LIMIT (for MySQL):

sql

CopyEdit

SELECT salary

FROM Employees

ORDER BY salary DESC

LIMIT 1 OFFSET 1;

1. **Explain partitioning and sharding in SQL databases.**
   * **Partitioning**: Splitting a large table into smaller, manageable pieces within the same database (Horizontal, Vertical, and Range partitioning).
   * **Sharding**: Distributing data across multiple databases (useful for scaling).
2. **How do you detect and remove duplicate records from a table?**

sql

CopyEdit

DELETE FROM Employees

WHERE id NOT IN (

SELECT MIN(id)

FROM Employees

GROUP BY name, salary

);

* + Alternative for detecting duplicates:

sql

CopyEdit

SELECT name, COUNT(\*)

FROM Employees

GROUP BY name

HAVING COUNT(\*) > 1;

1. **Write a query to calculate the running total of sales per month.**

sql

CopyEdit

SELECT month, sales,

SUM(sales) OVER (ORDER BY month) AS running\_total

FROM Sales;

1. **How do you handle deadlocks in SQL?**
   * Use **proper indexing**.
   * Execute transactions in **consistent order**.
   * Implement **lock timeouts**.
   * Avoid long-running transactions.
2. **Explain the difference between CTEs, subqueries, and temporary tables.**
   * **CTEs**: Temporary result set within a query (WITH cte AS (...)).
   * **Subqueries**: Nested queries used in SELECT, WHERE, etc.
   * **Temporary Tables**: Persist for a session (CREATE TEMP TABLE).
3. **Scenario: A report shows incorrect data due to a join issue. How would you debug and fix it?**
   * Check **join conditions** (INNER JOIN, LEFT JOIN, FULL JOIN).
   * Identify **duplicate/missing data** in either table.
   * Use **EXPLAIN ANALYZE** to see performance issues.
4. **How do you implement audit logging in an SQL database?**

* Create a dedicated **audit table**.
* Use **triggers** on INSERT, UPDATE, DELETE operations.
* Log user\_id, timestamp, operation\_type, and previous\_values.

**Python - Advanced & Scenario-Based Questions**

1. **How do you handle large datasets in Python efficiently?**
   * Use **Pandas with Dask** for parallel processing.
   * Use **memory-efficient data types** (float32 instead of float64).
   * Process data in **chunks** (pd.read\_csv(..., chunksize=10000)).
2. **Scenario: Your Python script is running slowly. How would you optimize it?**
   * Use **vectorized operations** with NumPy/Pandas.
   * Profile code using cProfile.
   * Use **multiprocessing** or **asynchronous execution**.
3. **Difference between multiprocessing and multithreading?**
   * **Multiprocessing**: Uses multiple CPU cores (best for CPU-bound tasks).
   * **Multithreading**: Uses shared memory (best for I/O-bound tasks).
4. **How does garbage collection work in Python?**
   * Uses **reference counting** and **generational garbage collection**.
5. **Find most frequent words in a large text file.**

python

CopyEdit

from collections import Counter

def most\_frequent\_words(file\_path, n=10):

with open(file\_path, 'r') as f:

words = f.read().split()

return Counter(words).most\_common(n)

1. **Handling memory leaks in Python?**
   * Use gc.collect() to trigger garbage collection.
   * Use **weak references** (weakref module).
2. **Automating a daily data pipeline?**
   * Use **Apache Airflow** or **Dagster**.
   * Automate with cron jobs.
3. **Lazy evaluation and Python generators?**
   * Generators (yield) compute values **only when needed**.
4. **Implement logging and exception handling?**

python

CopyEdit

import logging

logging.basicConfig(filename='app.log', level=logging.ERROR)

try:

1 / 0

except Exception as e:

logging.error("Exception occurred", exc\_info=True)

1. **Web scraping and storing in a database?**

* Use **BeautifulSoup** or **Scrapy**.
* Store results in SQLite/PostgreSQL.

**Power BI - Advanced & Scenario-Based Questions**

1. **Optimizing slow Power BI dashboards?**
   * Use **Aggregations** and **Performance Analyzer**.
   * Reduce dataset size by using **DirectQuery**.
2. **Best practices for writing efficient DAX formulas?**
   * Prefer **SUMX** over iterators like FILTER().
3. **Implementing dynamic row-level security (RLS)?**
   * Create a **security table** and define roles using USERNAME().
4. **Power BI Dataflows?**
   * Used for **ETL processes**, reusable across reports.
5. **Client-requested KPI calculation?**
   * Use **calculated measures** in DAX.
6. **Difference between Calculated Columns vs. Measures?**
   * Columns are precomputed, **measures are calculated dynamically**.
7. **Incremental refresh?**
   * Loads only new data instead of the entire dataset.
8. **Integrating Power BI with APIs?**
   * Use **Power Query (M code)** to fetch API data.
9. **DirectQuery vs. Import Mode performance?**
   * **Import Mode** is faster but uses memory; **DirectQuery** pulls live data.
10. **Troubleshooting refresh failures?**

* Check **gateway connections**, credentials, and query performance.

**Tableau - Advanced & Scenario-Based Questions**

**1. Scenario: Your Tableau dashboard is slow. How do you improve performance?**

* Use **Extracts** instead of live connections.
* Optimize data model (remove unnecessary joins).
* Reduce **marks** and **filters**.
* Use **Boolean/integer** filters instead of strings.
* Precompute calculations in the database.

**2. What are LOD (Level of Detail) expressions?**

LOD expressions control aggregation levels:

* **FIXED** → Aggregates at a specific dimension.
* **INCLUDE** → Includes extra dimensions beyond visualization.
* **EXCLUDE** → Ignores certain dimensions in aggregation.

**3. How do you implement custom geospatial mapping in Tableau?**

* Use **Shape files (.shp, .kml, .geojson)**.
* Join **Latitude/Longitude** data.
* Use **Mapbox or WMS Layers** for advanced customization.

**4. How do you create dynamic filters based on user selection?**

* Create a **Parameter** → Add filter options.
* Create a **Calculated Field** to apply the filter dynamically.
* Use **Dashboard Actions** for interactive filtering.

**5. How do you create a parameterized calculated field?**

* **Create a Parameter** (e.g., "Sales" or "Profit").
* **Use in a Calculated Field**:

sql

CopyEdit

CASE [Parameter]

WHEN 'Sales' THEN SUM([Sales])

WHEN 'Profit' THEN SUM([Profit])

END

**6. What is the difference between Data Blending and Data Joining?**

* **Join** → Merges tables within the **same data source** (before aggregation).
* **Blend** → Combines **different data sources** (after aggregation).

**7. How do you connect Tableau to a live streaming dataset?**

* Use **Web Data Connector (WDC)** for API-based streaming.
* Connect to **real-time databases** (BigQuery, Snowflake).
* Use **Hyper API** to update data extracts.

**8. How do you create a rolling 12-month average?**

* Use **Table Calculation**:

sql

CopyEdit

WINDOW\_AVG(SUM([Sales]), -11, 0)

* Apply it to a time-series graph.

**9. What are Table Calculations?**

Post-aggregation calculations applied within Tableau:

* **Running Total** → RUNNING\_SUM(SUM([Sales]))
* **Difference from Previous** → LOOKUP(SUM([Sales]), -1)
* **Percent of Total** → SUM([Sales]) / TOTAL(SUM([Sales]))

**10. How do you automate Tableau dashboard updates?**

* Schedule **refreshes** in **Tableau Server/Online**.
* Use **Tableau Bridge** for on-premise data.
* Automate updates via **Tableau REST API** or **tabcmd**.
* Enable **Webhooks** to trigger refreshes when data changes.