Data Science Interview Study Guide

Terms, Concepts, and Definitions

1. Database Fundamentals

Database – An organized collection of structured data stored electronically in a computer system, managed by a Database Management System (DBMS).

DBMS (Database Management System) – Software that enables users to create, read, update, and delete data in a database. It provides an interface between the database and end users or application programs.

RDBMS (Relational Database Management System) – A database system based on the relational model where data is organized into tables (relations) with rows and columns. Examples include PostgreSQL, MySQL, Oracle, and SQL Server.

Schema – The structure or blueprint of a database that defines how data is organized, including tables, columns, data types, relationships, constraints, and indexes.

Table (Relation) – A collection of related data organized in rows (records/tuples) and columns (attributes/fields). Each table represents an entity type in the database.

Primary Key – A column or combination of columns that uniquely identifies each row in a table. Primary keys must contain unique values and cannot contain NULL values.

Foreign Key – A column or set of columns in one table that references the primary key of another table, establishing a relationship between the two tables and maintaining referential integrity.

Composite Key – A primary key composed of two or more columns that together uniquely identify a row. Neither column alone is sufficient to uniquely identify records.

Surrogate Key – An artificial key (often auto-incremented integer) used as a primary key instead of a natural key. Surrogate keys have no business meaning and remain stable over time.

Natural Key – A key derived from the actual data attributes that naturally identify a record (e.g., Social Security Number, email address). Natural keys have business meaning.

Index – A database object that improves query performance by creating a fast lookup structure for specific columns. Indexes speed up SELECT queries but can slow INSERT, UPDATE, and DELETE operations.

ACID Properties – Guarantees for database transactions:

- **Atomicity** – All operations in a transaction complete successfully or none do (all-or-nothing).
- **Consistency** – Transactions transform the database from one valid state to another, maintaining all constraints.
- **Isolation** – Concurrent transactions execute independently without interference.

Durability – Once committed, transaction changes persist even in case of system failure.

Transaction – A logical unit of work consisting of one or more database operations that are executed as a single unit. Transactions follow ACID properties.

Referential Integrity – A database constraint ensuring that foreign key values in one table correspond to existing primary key values in another table, maintaining valid relationships.

Constraint – A rule enforced on database columns to ensure data validity and integrity. Common constraints include PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, CHECK, and DEFAULT.

2. SQL (Structured Query Language)

SQL Categories

DDL (Data Definition Language) – Commands that define and modify database structure. Includes CREATE, ALTER, DROP, TRUNCATE.

DML (Data Manipulation Language) – Commands that manipulate data within tables. Includes INSERT, UPDATE, DELETE.

DQL (Data Query Language) – Commands for querying and retrieving data. Primarily the SELECT statement.

DCL (Data Control Language) – Commands that control access to data. Includes GRANT, REVOKE.

TCL (Transaction Control Language) – Commands that manage database transactions. Includes COMMIT, ROLLBACK, SAVEPOINT.

SQL Joins

INNER JOIN – Returns only rows that have matching values in both tables being joined.

LEFT JOIN (LEFT OUTER JOIN) – Returns all rows from the left table and matching rows from the right table. Non-matching rows from the right table show NULL values.

RIGHT JOIN (RIGHT OUTER JOIN) – Returns all rows from the right table and matching rows from the left table. Non-matching rows from the left table show NULL values.

FULL JOIN (FULL OUTER JOIN) – Returns all rows from both tables. Non-matching rows show NULL values for columns from the table without a match.

CROSS JOIN – Returns the Cartesian product of both tables (every row from the first table paired with every row from the second table).

Advanced SQL Concepts

Subquery (Nested Query) – A query nested within another SQL query, enclosed in parentheses. Can be used in SELECT, FROM, WHERE, or HAVING clauses.

CTE (Common Table Expression) – A temporary named result set defined using WITH clause. CTEs improve readability and can be referenced multiple times in a query.

Window Function – Performs calculations across rows related to the current row using the OVER clause. Unlike aggregate functions, window functions don't collapse rows.

- **ROW_NUMBER()** – Assigns a unique sequential number to each row within a partition.
- **RANK()** – Assigns a rank with gaps (1, 2, 2, 4) when there are ties.
- **DENSE_RANK()** – Assigns a rank without gaps (1, 2, 2, 3) when there are ties.
- **LAG() / LEAD()** – Access data from previous or next rows without a self-join.

Aggregate Functions – Functions that perform calculations on multiple rows and return a single value. Common functions include COUNT(), SUM(), AVG(), MIN(), MAX().

GROUP BY – Groups rows that have the same values in specified columns into summary rows. Often used with aggregate functions.

HAVING – Filters grouped results after GROUP BY. WHERE filters before grouping; HAVING filters after.

3. Data Modeling

ERD (Entity-Relationship Diagram) – A visual representation of entities, their attributes, and relationships in a database system.

Entity – An object or concept about which data is stored (e.g., Customer, Product, Order). Represented as rectangles in ERDs.

Attribute – A property or characteristic of an entity (e.g., customer name, product price). Represented as ovals in ERDs.

Relationship – An association between entities (e.g., Customer places Order). Represented as diamonds in ERDs.

Cardinality

One-to-One (1:1) – Each entity instance in A is associated with at most one entity instance in B, and vice versa.

One-to-Many (1:N) – Each entity instance in A can be associated with multiple instances in B, but each B instance is associated with only one A instance.

Many-to-Many (M:N) – Entity instances in A can be associated with multiple instances in B, and vice versa. Requires a junction (bridge) table to implement.

Junction Table (Bridge Table) – A table used to implement many-to-many relationships. Contains foreign keys from both related tables as a composite primary key.

Conceptual Data Model – High-level view showing entities and relationships without technical details. Focuses on what data is needed, not how it's stored.

Logical Data Model – More detailed than conceptual, includes attributes, data types, and relationships. Platform-independent.

Physical Data Model – Implementation-specific model showing actual table structures, column types, indexes, and constraints for a specific DBMS.

4. Normalization

Normalization – The process of organizing data to reduce redundancy and improve data integrity by decomposing tables according to normal forms.

First Normal Form (1NF) – Requirements:

- Each column contains atomic (indivisible) values
- Each column contains values of a single type
- Each column has a unique name
- Order of rows and columns doesn't matter

Second Normal Form (2NF) – Requirements:

- Must be in 1NF
- All non-key attributes are fully dependent on the entire primary key
- No partial dependencies (relevant only for composite keys)

Third Normal Form (3NF) – Requirements:

- Must be in 2NF
- No transitive dependencies
- Non-key attributes depend only on the primary key, not on other non-key attributes

Boyce-Codd Normal Form (BCNF) – Requirements:

- Must be in 3NF
- Every determinant must be a candidate key

Denormalization – Intentionally introducing redundancy by combining tables to improve query performance. Used when read performance is more critical than storage efficiency.

Functional Dependency – A relationship where one attribute uniquely determines another attribute. Written as A → B (A determines B).

Transitive Dependency – When A → B and B → C, then A → C. A non-key attribute depends on another non-key attribute, violating 3NF.

5. Data Warehousing & Analytics

Data Warehouse – A centralized repository that stores integrated data from multiple sources, optimized for analysis and reporting rather than transaction processing.

OLTP (Online Transaction Processing) – Systems designed for managing day-to-day transactional data (e.g., order entry, banking). Optimized for fast INSERT, UPDATE, DELETE operations with normalized schemas.

OLAP (Online Analytical Processing) – Systems designed for complex queries and data analysis. Optimized for read-heavy operations with denormalized schemas.

ETL (Extract, Transform, Load) – Process of extracting data from sources, transforming it (cleaning, aggregating), and loading it into a data warehouse.

Fact Table – Central table in a star schema containing quantitative measurements (metrics/facts) and foreign keys to dimension tables. Examples: sales revenue, quantity sold, profit.

Dimension Table – Contains descriptive attributes (dimensions) that provide context for facts. Examples: customer details, product information, date/time details.

Star Schema – Data warehouse design with a central fact table connected directly to denormalized dimension tables, forming a star shape. Simple and fast for queries.

Snowflake Schema – Normalized version of star schema where dimension tables are further decomposed into sub-dimensions. Saves storage but increases query complexity.

Data Mart – A subset of a data warehouse focused on a specific business area (e.g., Sales, Marketing, Finance). Provides targeted data access for departments.

Data Lake – A storage repository that holds vast amounts of raw, unstructured, and structured data in its native format. Schema-on-read approach.

Dimensional Modeling – Design technique for data warehouses focusing on business processes and measurements, typically resulting in star or snowflake schemas.

Slowly Changing Dimension (SCD) – Dimension that changes over time. Type 1 overwrites, Type 2 adds new rows with version history, Type 3 adds columns for limited history.

6. Database Types & Technologies

NoSQL (Not Only SQL) – Non-relational databases designed for distributed data, flexible schemas, and horizontal scalability. Types include document, key-value, column-family, and graph databases.

Document Database – NoSQL database that stores data in JSON-like documents. Examples: MongoDB, Couchbase. Good for semi-structured data with varying attributes.

Key-Value Store – Simplest NoSQL type storing data as key-value pairs. Examples: Redis, DynamoDB. Extremely fast for simple lookups.

Column-Family Database – Stores data in columns rather than rows. Examples: Cassandra, HBase. Optimized for analytical queries on large datasets.

Graph Database – Stores data as nodes and edges (relationships). Examples: Neo4j, Amazon Neptune. Excellent for highly connected data and relationship queries.

Vector Database – Specialized database for storing and querying vector embeddings. Examples: Pinecone, Weaviate. Used for semantic search and AI applications.

In-Memory Database – Database that stores data in RAM rather than disk for ultra-fast access. Examples: Redis, Memcached. Used for caching and real-time applications.

7. Python & Data Analysis

Pandas – Python library for data manipulation and analysis. Provides DataFrame and Series data structures for working with structured data.

DataFrame – Two-dimensional labeled data structure with columns of potentially different types, similar to a spreadsheet or SQL table.

Series – One-dimensional labeled array capable of holding any data type. A DataFrame column is a Series.

Vectorization – Applying operations to entire arrays at once without explicit loops, resulting in faster execution and cleaner code.

GroupBy – Pandas operation that splits data into groups based on criteria, applies a function to each group, and combines results. Similar to SQL GROUP BY.

Merge/Join – Combining DataFrames based on common columns or indexes. Similar to SQL joins (inner, left, right, outer).

Pivot Table – Reshaping data to summarize values by creating a spreadsheet-style pivot table with rows, columns, and aggregated values.

Missing Data (NaN/NULL) – Absent values in datasets. Handled through methods like dropna() to remove, fillna() to replace, or interpolate() to estimate.

NumPy – Fundamental Python library for numerical computing, providing support for arrays, matrices, and mathematical functions.

SQLAlchemy – Python SQL toolkit and Object-Relational Mapping (ORM) library that provides database connectivity and allows working with databases using Python objects.

8. APIs & Web Services

API (Application Programming Interface) – Set of rules and protocols that allows different software applications to communicate with each other.

REST (Representational State Transfer) – Architectural style for web services using standard HTTP methods. RESTful APIs are stateless and use URLs to represent resources.

HTTP Methods – Verbs indicating desired actions:

GET – Retrieve data (read-only)
POST – Create new resource
PUT – Update entire resource
PATCH – Partial update of resource
DELETE – Remove resource

HTTP Status Codes – Three-digit codes indicating request outcome:

2xx Success – Request succeeded (200 OK, 201 Created)
3xx Redirection – Further action needed (301 Moved Permanently)
4xx Client Error – Client error (400 Bad Request, 404 Not Found)
5xx Server Error – Server error (500 Internal Server Error)

JSON (JavaScript Object Notation) – Lightweight data format commonly used in APIs for data exchange. Human-readable and language-independent.

Endpoint – Specific URL where an API can access resources (e.g., /api/v1/customers/123).

API Authentication – Verifying identity of API users. Common methods include API keys, OAuth, JWT (JSON Web Tokens).

Rate Limiting – Controlling the number of API requests a client can make within a time period to prevent abuse and ensure fair usage.

9. Performance & Optimization

Query Optimization – Process of improving query performance through techniques like proper indexing, avoiding SELECT *, using appropriate joins, and analyzing execution plans.

Execution Plan (Query Plan) – Database's step-by-step strategy for executing a query. Shows operations like table scans, index seeks, and join methods. Used to identify performance bottlenecks.

Indexing Strategy – Choosing which columns to index based on query patterns. Index columns used in WHERE, JOIN, ORDER BY clauses. Balance read performance against write overhead.

Partitioning – Dividing large tables into smaller, manageable pieces (partitions) based on a key (e.g., date ranges). Improves query performance and maintenance.

Caching – Storing frequently accessed data in fast-access storage (memory) to reduce database load and improve response times.

Sharding – Horizontal partitioning where data is distributed across multiple database servers. Each shard contains a subset of the data. Enables horizontal scaling.

Replication – Creating and maintaining copies of data across multiple servers for redundancy, load distribution, and disaster recovery.

Connection Pooling – Maintaining a cache of database connections that can be reused, reducing the overhead of creating new connections for each request.

10. Common Interview Topics

SQL Concepts to Master

- Writing complex JOIN queries (inner, left, right, full)
- Using window functions (ROW_NUMBER, RANK, LAG/LEAD)
- Aggregations with GROUP BY and HAVING
- Subqueries and Common Table Expressions (CTEs)
- Self-joins for hierarchical data
- Understanding NULL handling (IS NULL, COALESCE, NULLIF)

Data Modeling & Design

- Creating and interpreting ERDs
- Understanding normalization (1NF through BCNF)
- Choosing primary keys (surrogate vs. natural)
- Implementing many-to-many relationships
- When to denormalize for performance

Python & Pandas

- DataFrame manipulation (filter, sort, transform)
- Merging and joining DataFrames
- Handling missing data
- GroupBy operations and aggregations
- Working with datetime data

System Design Concepts

- Star schema vs. snowflake schema
- OLTP vs. OLAP systems
- ETL pipeline design
- Choosing between SQL and NoSQL
- Scalability strategies (vertical vs. horizontal)
- CAP theorem (Consistency, Availability, Partition tolerance)

Classic Interview Problems

- Find nth highest salary
- Identify duplicate records
- Calculate running totals and moving averages
- Find gaps in sequences
- Recursive queries (hierarchies, org charts)
- Optimize slow queries

Study Tips

- Practice writing SQL queries daily without looking at solutions
- Draw ERDs for real-world scenarios
- Explain concepts out loud to test understanding
- Use online platforms (LeetCode, HackerRank) for SQL practice
- Review execution plans to understand query performance
- Build sample projects combining SQL, Python, and data modeling
- Prepare to explain trade-offs in design decisions