Research Assignment: Section A: Database Fundamentals

① Main types of Databases:
- Relational Databases: Data organized into tables with defined relationships. Example: MySQL, PostgreSQL
- NoSQL Databases: Handle unstructured/semi-structured data; categories include Document, Key-Value, Graph, and Column-Family. Example: MongoDB, Cassandra
- Data Warehouses: Optimized for analytics and reporting on historical data. Example: Snowflake, Redshift.
- In-Memory Databases: Store data in main memory for speed. Example: SAP HANA
- Time-Series Databases: Optimized for time-stamped data. Example: InfluxDB

② Relational Database Management System (RDBMS)
- A software system used to manage a relational database.
- It uses SQL for querying and maintains data integrity through schemas and constraints.
- Data is stored in tables with rows and columns, enforcing relationships between them.

③ Primary key and Foreign key
- Primary key (PK): A column or set of columns that uniquely identifies each record (row) in a table. It cannot contain NULL values.
- Foreign key (FK): A column or set of columns that establishes a link between two tables. It references the Primary key of another table.

④ Database Normalization and Importance
- Normalization: A process of organizing columns and tables to minimize data redundancy and improve data integrity.
- Importance: Reduces data duplication, simplifies data maintenance (updates, deletions), and ensures data dependencies make sense.

⑤ Database Schema
- The blueprint or structure of the database.
- It defines the organization of data, including table names, columns, data types, indexes, views, and the relationships (constraints) between tables.

⑥ Difference between structured, semi-structured and unstructured data
- Structured: Highly organized; adheres to a fixed schema. eg: data in RDBMS tables.
- Easy to query with SQL.
- Semi-structured: Does not adhere to a strict relational structure but contains tags/markers to seperate elements and enforce hierarchy, eg: JSON, XML.
- structured: Has no predefined format or ; organization, eg: text documents, images, videos, emails). Requires advanced tools for analysis.

⑦ Difference between a Fact table and Dimension table.
- Fact Table: Stores quantitative data such metrics/facts about a business event, eg: sales amount, quantity. It contains foreign keys to dimension tables.
- Dimension Table: Stores descriptive data such as attributes related to the facts, eg: Customer name, product color, date. Provides context for the facts.

⑧ Data model and its Importance in Database Design
- Data Model: An abstract representation of real-world data structures, defining how data is connected and processed. Examples include the Conceptual, Logical and Physical models.
- Importance: Provides a common language for users/developers, ensures data consistency, and helps in planning and structuring the database for efficiency and maintainability

@ Difference between a database, a data warehouse and a data lake
- Database (OLTP): Focuses on transactional processing such as day-to-day operations, handles current, highly normalized data. Optimized for fast inserts/updates/deletes.
- Data Warehouse (OLAP): Focuses on analytical processing such as reporting/BI, stores historical, denormalized/star-schema data. Optimized for fast, complex queries over large datasets.

Data Lake: A centralized repository to store raw data at scale, including structured, semi-structured, and ustructured data, before it's been cleaned or processed.

@ Difference between a database, a data warehouse and a data lake
- Database (OLTP): Focuses on transactional processing such as day-to-day operations, handles current, highly normalized data. Optimized for fast inserts/updates/deletes.
- Data Warehouse (OLAP): Focuses on analytical processing such as

@ Data Mart differing from a data warehouse
- Data Mart: A subset of a data warehouse focused on a specific business function or department, eg: Marketing, Sales, Finance
- Difference: A data warehouse is enterprise-wide, covering many subjects. A data mart is smaller, more focused, and generally easier and quicker to implement.

Section B: SQL and Data Processing

(11) Query Language and SQL Importance
- Query Language: A programming language designed to retrieve and manage data from a database.
- SQL (Structured Query Language) Importance: It's standardized, relatively simple to learn, highly versatile across various RDBMS platforms, and is efficient for complex data manipulation.

(12) Indexes in databases and how they improve performance:
- Indexes: Special lookup tables that the database search engine can use to speed up data retrieval, similar to a book's index. They are created on one or more columns.
- Performance Improvement: They allow the database to quickly find a row's location without having to scan the entire table (a full table scan), drastically improving SELECT query speed.

(13) Transactions in databases, and ACID properties

Transaction: A single, logical unit of work, consisting of one ore more database operations, which is treated as a single indivisible operation. It either fully completes or fully fails.

- ACID Properties: Guarantee data integrity for transactions:
- Atomicity: All-or-nothing principle
- Consistency: Only valid data is written on the database.
- Isolation: Concurrent transactions don't interfere with each other.
- Durability: Once a transaction is committed, changes are permanent, even after system failure.

(14) Database engine and Impact on Performance

Database Engine/Storage engine: The underlying component of database management system that handles how data is actually stored, retrieved, and updated in memory and on disk.

- Impact on Performance: Different engines, eg: InnoDB, MyISAM fo MySQL have different strategies for indexing, locking, and transaction handling (ACID support), which critically determine the database's speed and reliability under various workloads.

(15) Views, Stored Procedures and Triggers in SQL

- View: A virtual table whose content is defined by a query. It doesn't store data itself but retrieves it from tables when accessed. Simplifies complex queries and enhances security
- Stored Procedure: A pre-compiled collection of SQL statements stored in the database. Can be executed by calling its name, reducing network traffic and improving performance.

Trigger: A special type of stored procedure that automatically executes (fires) when a specific event occurs on a table, eg: INSERT, UPDATE, DELETE, Used to enforce complex business rules.

(16) Difference between ETL (Extract, Transform, Load) and ELT (Extract, Load, Transform)

- ETL: Data is Extracted from source, transformed (cleaned, aggregated, standardized) on a seperate staging server, and then loaded into the target data warehouse. Used traditionally.
- ELT: Data is Extracted from source, Loaded directly into the target data lake/warehouse, which is often cloud-based and scalable, and then Transformed using the power of the target system's resources. More common with cloud data warehouses.

(17) Difference between batch processing and stream processing in data pipelines

- Batch Processing: Data is collected, stored, and processed in large chunks, batches, at specific intervals, eg: end of day, hourly. Suitable for tasks where results are not needed instantly.
- Stream Processing: Data is processed continuously and in real-time is generated, data-in-motion. Suitable for applications requiring immediate insights, like fraud detection or live monitoring.

(18) Join in SQL and different types of Joins

- Join: A clause used to combine rows from two or more tables based on a related column between them.
- Types: INNER JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table.
- Right (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
- Full (OUTER) JOIN: Returns records when there is a match in one of the tables.

(19) Referential integrity and its importance in relational databases

- Referential Integrity: A database concept that ensures relationships between tables are valid. It requires that every foreign key value in the child table must have a must have a matching Primary Key value in the parent table, or be NULL.
- Importance: Prevents users from creating 'orphan' records, eg: a Sale order for a Customer ID that does not exist and thus maintains the consistency and accuracy of the data relationships.

(20) Data redundancy affecting database performance and storage.
- Performance: Negatively. It makes UPDATE and DELETE operations slower and more complex, as the same data needs to be modified in multiple places. It also requires more complex indexing, potentially slowing down INSERT operations.
- Storage: Negatively. Storing the same information multiple times significantly increases the amount of disk space required.

Section C: Data Management and Analytics Concepts

(21) Cloud database management differing from on-premise databases
- Cloud: Database infrastructure is managed by a third-party provider, eg: AWS, Azure. Offers elastic scalability, pay-as-you-go pricing, automatic backups/updates, and high availability. Lower operational burden for user.
- On-Premise: Database hardware and software are owned and managed locally by the organization. Requires large upfront investment, high control over security and customization, but lacks the immediate elasticity and ease of maintenance of cloud solutions.

(22) Data governance and its importance in data management.
Data Governance: A system of policies, procedures, and standards used to manage the availability, usability, integrity, and security of data in an enterprise.
- Importance: Ensures compliance, eg: GDPR, HIPAA, improves data quality, enables better decision-making, and establishes clear accountability for data assets.

(23) Data integrity
- Data integrity: The accuracy, consistency, and reliability of data over its entire lifecycle.
- Maintenance:
- Referential Integrity: Using Foreign Keys and cascading rules.
- Entity Integrity: Using Primary Keys, ensures no duplicates/NULLs.
- Domain Integrity: Applying data type constraints and check constraints, eg: age must be positive.
- User/Application Integrity: Implementing business rules via store procedures and triggers.

(24) Data quality and Importance for analytics
- Data Quality: The degree to which data meets the required standards of accuracy, completeness, consistency, relevance, and timeliness.
- Criticality for Analytics: Poor data quality leads to flawed insights, inaccurate reports, and ultimately, poor business decisions.

(25) Role of a Data Analyst in managing and analyzing database information.
- Role: Bridges the gap between raw data and business decision-making.
- Responsibilities: Querying: Writing SQL to extract, filter, and aggregate data.
- Analysis: Interpreting data, identifying trends, and creating reports/dashboards.
- Data Quality: Identifying and reporting on data inconsistencies or errors.
- Collaboration: Working with stakeholders to define data needs and present findings.

(26) Key Responsibilities of a Data Administrator (DBA)
- Installation and Configuration: Setting up the database server and software.
- Backup and Recovery: Planning and executing data protection and disaster recovery procedures.
- Security: Managing user access, permissions and auditing.
- Performance Tuning: Optimizing queries, indexes and database configuration.
- Maintenance: Patching, monitoring, and ensuring high availability.

(27) Main steps involved in designing a data pipeline.
- Extraction: Identifying and collecting data from source systems eg: databases, APIs, files.
- Transformation/Cleansing: Cleaning, validating, standardizing and structuring the raw data.
- Loading: Moving the transformed data into the target destination, eg: data warehouse/lake
- Monitoring/Orchestration: Scheduling the jobs, managing dependencies and ensuring the pipeline runs reliably.

(27) Common challenges in managing large-scale databases
- Scalability: Handling increasing volumes of data and user traffic, read/write throughput.
- Performance: Optimizing complex queries over massive datasets and ensuring fast response times.
- High Availability/Disaster Recovery: Ensuring minimal downtime and rapid recovery from failures.
- Security: Protecting sensitive data from breaches and managing access in complex environments.
- Cost Management: Controlling infrastructure and licensing expenses, especially in the cloud.

(28) Popular database platforms and their use cases
- My SQL: (RDBMS) Open-source, popular for web applications, part of LAMP stack, and transactional systems, OLTP.
- PostgreSQL: (RDBMS) Open-source, highly feature-rich, known for data integrity and complex custom applications. Often preferred for mission-critical systems and spatial data.
- Oracle (RDBMS) Commercial, robust, widely used for large-scale, enterprise-level OLTP systems and high-volume workloads.
- Snowflake: (Cloud Data Warehouse) Cloud-native, zero management system, excels at data warehousing and analytical processing (OLAP)

(30) Main data storage formats used in analytics
- CSV (Comma Seperated Values): Simple, text-based, row-oriented format. Easy to read, but inefficient for large-scale analytics.
- JSON (JavaScript Object Notation): Text-based, semi-structured, hierarchial format. Excellent for web data and flexibility, less efficient for direct analytics
- Parquet: Columnar storage format: Highly efficient for big data analytics, faster queries, better compression, widely used in Data Lakes, eg: with Spark.
- Avro: Row-oriented storage format with a schema definition. Excellent for data interchange between systems, especially for streaming data, eg: with Kafka.