Complete SQL Guide - Questions and Answers

Introduction to SQL

1. What is SQL, and why is it essential in database management?

SQL (Structured Query Language) is a standardized programming language designed for managing and manipulating relational databases. It is essential because:

- Provides a universal interface for database operations
- Enables data retrieval, insertion, updating, and deletion
- Supports database structure creation and modification
- Ensures data integrity and security
- Facilitates complex data analysis and reporting

2. Explain the difference between DBMS and RDBMS.

DBMS (Database Management System):

- Manages any type of database
- Data stored in hierarchical or navigational form
- No relationships between data
- Examples: File systems, XML databases

RDBMS (Relational Database Management System):

- Manages relational databases specifically
- Data stored in tables with rows and columns
- Supports relationships between tables
- Follows ACID properties
- Examples: MySQL, PostgreSQL, Oracle, SQL Server

3. Describe the role of SQL in managing relational databases.

SQL serves as the primary interface for:

- **Data Definition**: Creating and modifying database structures
- **Data Manipulation**: Adding, updating, and deleting records
- **Data Querying**: Retrieving specific information

- **Access Control**: Managing user permissions
- **Transaction Control**: Ensuring data consistency

4. What are the key features of SQL?

- **Standardized Language**: Universal across different RDBMS
- **Declarative Nature**: Focus on what to do, not how
- **Comprehensive**: Handles all database operations
- **Scalable**: Works with small to enterprise databases
- **Secure**: Built-in access control mechanisms
- **Portable**: Code can work across different systems

SQL Syntax

1. What are the basic components of SQL syntax?

- **Keywords**: Reserved words (SELECT, FROM, WHERE)
- **Identifiers**: Names of tables, columns, databases
- **Literals**: Constant values (strings, numbers, dates)
- **Operators**: Comparison and logical operators
- **Comments**: Documentation within code
- **Statements**: Complete SQL commands

2. Write the general structure of an SQL SELECT statement.

```sql
SELECT column1, column2, ...
FROM table_name
WHERE condition
GROUP BY column_name
HAVING condition
ORDER BY column_name ASC|DESC
LIMIT number;
```

3. Explain the role of clauses in SQL statements.

Clauses are components that modify or filter SQL operations:

- **WHERE**: Filters rows based on conditions

- **ORDER BY**: Sorts result sets
- **GROUP BY**: Groups rows for aggregation
- **HAVING**: Filters grouped results
- **LIMIT**: Restricts number of returned rows

SQL Constraints

1. What are constraints in SQL? List and explain the different types of constraints.

Constraints are rules that enforce data integrity in tables:

- **PRIMARY KEY**: Uniquely identifies each record
- **FOREIGN KEY**: Links records between tables
- **NOT NULL**: Prevents null values
- **UNIQUE**: Ensures all values are distinct
- **CHECK**: Validates data against specific conditions
- **DEFAULT**: Provides default values for columns

2. How do PRIMARY KEY and FOREIGN KEY constraints differ?

PRIMARY KEY:

- Uniquely identifies each row in a table
- Cannot contain NULL values
- Only one per table
- Automatically creates a unique index

FOREIGN KEY:

- References PRIMARY KEY of another table
- Can contain NULL values
- Multiple foreign keys allowed per table
- Ensures referential integrity

3. What is the role of NOT NULL and UNIQUE constraints?

NOT NULL:

- Prevents empty values in columns
- Ensures data completeness

- Applied at column level

UNIQUE:

- Ensures all values in column(s) are distinct
- Allows one NULL value
- Can be applied to single or multiple columns

Main SQL Commands and Sub-commands (DDL)

1. Define the SQL Data Definition Language (DDL).

DDL (Data Definition Language) is a subset of SQL used to define and modify database structures:

- **CREATE**: Creates databases, tables, indexes
- **ALTER**: Modifies existing structures
- **DROP**: Removes databases, tables, indexes
- **TRUNCATE**: Removes all data from tables

2. Explain the CREATE command and its syntax.

CREATE command establishes new database objects:

```sql
CREATE TABLE table_name (
    column1 datatype constraints,
    column2 datatype constraints,
    ...
    table_constraints
);
```

Example:

```sql
CREATE TABLE employees (
    id INT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    salary DECIMAL(10,2) DEFAULT 0
);
```

3. What is the purpose of specifying data types and constraints during table creation?

- **Data Types**: Define the kind of data stored (INT, VARCHAR, DATE)
- **Constraints**: Ensure data integrity and validity
- **Storage Optimization**: Proper types reduce storage requirements
- **Performance**: Appropriate types improve query performance

ALTER Command

1. What is the use of the ALTER command in SQL?

ALTER modifies existing database structures without losing data:

- Add new columns or constraints
- Modify existing columns
- Drop columns or constraints
- Rename tables or columns

2. How can you add, modify, and drop columns from a table using ALTER?

Add Column:

```sql
ALTER TABLE table_name ADD column_name datatype constraints;
```

Modify Column:

```sql
ALTER TABLE table_name MODIFY column_name new_datatype;
```

Drop Column:

```sql
ALTER TABLE table_name DROP COLUMN column_name;
```

DROP Command

1. What is the function of the DROP command in SQL?

DROP permanently removes database objects:

- Deletes entire tables, databases, or indexes
- Cannot be undone without backups
- Removes both structure and data

2. What are the implications of dropping a table from a database?

- **Data Loss**: All records permanently deleted
- **Referential Integrity**: May break foreign key relationships
- **Application Impact**: Dependent queries will fail
- **Recovery**: Requires backup restoration

Data Manipulation Language (DML)

1. Define the INSERT, UPDATE, and DELETE commands in SQL.

INSERT: Adds new records to tables

```sql
INSERT INTO table_name (column1, column2) VALUES (value1, value2);
```

UPDATE: Modifies existing records

```sql
UPDATE table_name SET column1 = value1 WHERE condition;
```

DELETE: Removes records from tables

```sql
DELETE FROM table_name WHERE condition;
```

2. What is the importance of the WHERE clause in UPDATE and DELETE operations?

- **Precision**: Targets specific records for modification
- **Safety**: Prevents accidental mass updates/deletions
- **Performance**: Reduces processing overhead
- **Data Integrity**: Ensures only intended changes occur

Data Query Language (DQL)

1. What is the SELECT statement, and how is it used to query data?

SELECT retrieves data from databases:

- Most commonly used SQL command
- Can query single or multiple tables
- Supports filtering, sorting, and grouping
- Returns result sets based on specified criteria

2. Explain the use of the ORDER BY and WHERE clauses in SQL queries.

WHERE Clause:

- Filters rows before processing
- Uses conditions to select specific records
- Applied before grouping operations

ORDER BY Clause:

- Sorts result set by specified columns
- ASC (ascending) or DESC (descending)
- Applied after all other operations

Data Control Language (DCL)

1. What is the purpose of GRANT and REVOKE in SQL?

GRANT: Provides privileges to users

```sql
GRANT SELECT, INSERT ON table_name TO username;
```

REVOKE: Removes privileges from users

```sql
REVOKE SELECT, INSERT ON table_name FROM username;
```

2. How do you manage privileges using these commands?

- **User Access Control**: Restrict database operations
- **Security**: Implement least privilege principle
- **Role-Based**: Assign privileges to roles, then roles to users
- **Granular Control**: Specific permissions on specific objects

Transaction Control Language (TCL)

1. What is the purpose of the COMMIT and ROLLBACK commands in SQL?

COMMIT: Permanently saves transaction changes

```sql
COMMIT;
```

ROLLBACK: Undoes transaction changes

```sql
ROLLBACK;
```

2. Explain how transactions are managed in SQL databases.

Transaction Management ensures data consistency:

- **ACID Properties**: Atomicity, Consistency, Isolation, Durability
- **Begin Transaction**: Starts transaction block
- **Commit**: Makes changes permanent
- **Rollback**: Cancels changes
- **Savepoints**: Create intermediate rollback points

SQL Joins

1. Explain the concept of JOIN in SQL. What is the difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN?

JOIN combines records from multiple tables based on relationships:

INNER JOIN: Returns only matching records from both tables **LEFT JOIN**: Returns all records from left table, matching from right **RIGHT JOIN**: Returns all records from right table, matching from left **FULL**

OUTER JOIN: Returns all records from both tables

2. How are joins used to combine data from multiple tables?

```sql
SELECT columns
FROM table1
JOIN table2 ON table1.key = table2.key;
```

Joins enable:

- **Data Normalization**: Avoid data duplication
- **Complex Queries**: Retrieve related information
- **Reporting**: Comprehensive data analysis

SQL Group By

1. What is the GROUP BY clause in SQL? How is it used with aggregate functions?

GROUP BY groups rows with same values for aggregation:

```sql
SELECT column, COUNT(*)
FROM table_name
GROUP BY column;
```

Used with aggregate functions: COUNT, SUM, AVG, MAX, MIN

2. Explain the difference between GROUP BY and ORDER BY.

GROUP BY:

- Groups rows for aggregation
- Used with aggregate functions
- Creates summary results

ORDER BY:

- Sorts result set
- Applied to final output
- Does not change data structure

SQL Stored Procedure

1. What is a stored procedure in SQL, and how does it differ from a standard SQL query?

Stored Procedure is a precompiled collection of SQL statements:

- Stored in database server
- Can accept parameters
- Returns results or performs actions
- Reusable and faster execution

2. Explain the advantages of using stored procedures.

- **Performance**: Precompiled and optimized
- **Security**: Controlled access to data
- **Reusability**: Called from multiple applications
- **Maintenance**: Centralized business logic
- **Network Traffic**: Reduced data transfer

SQL View

1. What is a view in SQL, and how is it different from a table?

View is a virtual table based on SQL query results:

- Does not store data physically
- Dynamic - reflects current table data
- Simplifies complex queries
- Provides security layer

2. Explain the advantages of using views in SQL databases.

- **Security**: Hide sensitive columns
- **Simplicity**: Complex joins appear as simple tables
- **Consistency**: Standard interface for applications
- **Abstraction**: Hide database complexity
- **Maintenance**: Changes in underlying tables don't affect applications

SQL Triggers

1. What is a trigger in SQL? Describe its types and when they are used.

Trigger is a special stored procedure that automatically executes in response to events:

Types:

- **BEFORE**: Executes before triggering event
- **AFTER**: Executes after triggering event
- **INSTEAD OF**: Replaces triggering event (views)

2. Explain the difference between INSERT, UPDATE, and DELETE triggers.

INSERT Trigger: Fires when new records are added **UPDATE Trigger**: Fires when existing records are modified **DELETE Trigger**: Fires when records are removed

Each type can access OLD and NEW row values as appropriate.

Introduction to PL/SQL

1. What is PL/SQL, and how does it extend SQL's capabilities?

PL/SQL (Procedural Language/SQL) extends SQL with programming features:

- Variables and constants
- Control structures (loops, conditions)
- Exception handling
- Functions and procedures
- Object-oriented features

2. List and explain the benefits of using PL/SQL.

- **Performance**: Reduced network traffic
- **Productivity**: Rich programming environment
- **Integration**: Seamless SQL integration
- **Security**: Stored in database server
- **Maintainability**: Modular programming approach

PL/SQL Control Structures

1. What are control structures in PL/SQL? Explain the IF-THEN and LOOP control structures.

Control Structures manage program flow:

IF-THEN:

```sql
IF condition THEN
    statements;
END IF;
```

LOOP:

```sql
LOOP
    statements;
    EXIT WHEN condition;
END LOOP;
```

2. How do control structures in PL/SQL help in writing complex queries?

- **Decision Making**: Conditional execution
- **Repetition**: Process multiple records
- **Flow Control**: Manage program logic
- **Error Handling**: Graceful error management

SQL Cursors

1. What is a cursor in PL/SQL? Explain the difference between implicit and explicit cursors.

Cursor is a pointer to query result set:

Implicit Cursor: Automatically created for single-row queries **Explicit Cursor**: Manually declared for multi-row queries

2. When would you use an explicit cursor over an implicit one?

Use explicit cursors when:

- Processing multiple rows

- Need fine control over row processing
- Complex row-by-row operations required
- Better error handling needed

Rollback and Commit Savepoint

1. Explain the concept of SAVEPOINT in transaction management. How do ROLLBACK and COMMIT interact with savepoints?

SAVEPOINT creates intermediate rollback points within transactions:

```sql
SAVEPOINT savepoint_name;
ROLLBACK TO savepoint_name;
```

Interactions:

- **ROLLBACK TO SAVEPOINT**: Undoes changes to specific point
- **COMMIT**: Makes all changes permanent, removes all savepoints

2. When is it useful to use savepoints in a database transaction?

- **Long Transactions**: Multiple logical steps
- **Error Recovery**: Partial rollback without losing all work
- **Conditional Processing**: Different paths in same transaction
- **Batch Processing**: Rollback problematic records only

This document provides a comprehensive overview of SQL concepts, commands, and advanced features. Each section builds upon previous knowledge to create a complete understanding of database management using SQL.