Basic Concepts

1. Introduction to SQL
2. Data Types in SQL
3. Database Schema
4. Tables, Rows, and Columns
5. Constraints (Primary Key, Foreign Key, Unique, Not Null, Check, Default)

Data Definition Language (DDL)

1. **CREATE**: Creating databases, tables, views, indexes
2. **ALTER**: Modifying tables, adding/deleting columns, constraints
3. **DROP**: Deleting databases, tables, views, indexes
4. **TRUNCATE**: Removing all records from a table

Data Manipulation Language (DML)

1. **SELECT**: Retrieving data
    - WHERE clause
    - GROUP BY, HAVING, ORDER BY
    - Joins (INNER, LEFT, RIGHT, FULL)
    - Subqueries and Nested Queries
    - UNION, INTERSECT, EXCEPT
2. **INSERT**: Adding records to a table
3. **UPDATE**: Modifying existing records
4. **DELETE**: Removing specific records

Data Control Language (DCL)

1. **GRANT**: Providing access privileges
2. **REVOKE**: Removing access privileges

Transaction Control Language (TCL)

1. **COMMIT**: Saving changes
2. **ROLLBACK**: Undoing changes
3. **SAVEPOINT**: Partial rollback within a transaction
4. **SET TRANSACTION**: Defining transaction properties

Advanced SQL Queries

1. Aggregate Functions (SUM, AVG, COUNT, MAX, MIN)
2. Window Functions (ROW_NUMBER, RANK, NTILE, LAG, LEAD)
3. Common Table Expressions (CTEs)
4. Recursive Queries
5. PIVOT and UNPIVOT
6. Full-Text Search

Indexes

1. Types of Indexes (Clustered, Non-Clustered, Unique, Full-Text)
2. Index Performance Optimization
3. Covering Indexes

Stored Procedures and Functions

1. Creating Stored Procedures
2. User-Defined Functions (Scalar and Table-Valued)
3. Input and Output Parameters
4. Error Handling in Procedures
5. Transactions in Stored Procedures

Triggers

1. AFTER Triggers
2. INSTEAD OF Triggers
3. Nested and Recursive Triggers

Views

1. Creating and Managing Views
2. Indexed Views
3. Updatable Views

Performance Optimization

1. Query Execution Plans
2. Query Hints
3. Index Maintenance (Rebuilding, Reorganizing)
4. Database Normalization and Denormalization
5. SQL Profiler and Extended Events

Security

1. Authentication and Authorization
2. Role-Based Security
3. Data Encryption (Transparent Data Encryption, Column-Level Encryption)
4. Dynamic Data Masking
5. Row-Level Security

Data Types and Advanced Data Handling

1. JSON and XML in SQL
2. Hierarchical Data Handling
3. Temporal Tables
4. Computed Columns
5. User-Defined Data Types

Replication and Backup

1. Types of Replication (Transactional, Snapshot, Merge)
2. Backup Strategies (Full, Differential, Transaction Log)
3. Point-in-Time Recovery

SQL and Programming

1. Dynamic SQL
2. SQL Injection Prevention
3. Using SQL with Other Languages (C#, Python, etc.)
4. ORM Tools (Entity Framework, Hibernate)

SQL Server-Specific Features

1. SQL Server Management Studio (SSMS)
2. Database Mail
3. Always On Availability Groups
4. Data Compression
5. In-Memory OLTP

Data Warehousing and Analytics

1. Data Partitioning
2. Materialized Views
3. Data Cubes
4. OLAP and OLTP
5. Data Marts

Miscellaneous

1. Cross-Database Queries
2. Linked Servers
3. Temporal Queries
4. Distributed Transactions
5. Logging and Auditing

Let me know if you want detailed explanations or examples for any specific topic!

Basic Concepts

1. Introduction to SQL

SQL (Structured Query Language) is a standard language for managing and manipulating relational databases.

2. Data Types in SQL

- **Numeric**: INT, DECIMAL, FLOAT
- **String**: VARCHAR, CHAR, TEXT
- **Date/Time**: DATE, DATETIME, TIMESTAMP

Example:

```
CREATE TABLE Employees (
    ID INT PRIMARY KEY,
    Name VARCHAR(50),
    HireDate DATE
);
```

3. Database Schema

Represents the structure of a database, including tables, columns, and relationships.

Example:

```
CREATE SCHEMA Sales;
```

4. Constraints

- **Primary Key**: Ensures unique and non-null values.
- **Foreign Key**: Links two tables.

Example:

```
ALTER TABLE Orders
ADD CONSTRAINT FK_Customer FOREIGN KEY (CustomerID) REFERENCES Customers(ID);
```

Data Definition Language (DDL)

1. CREATE

Creates a new table.

```
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100),
    Price DECIMAL(10, 2)
);
```

2. ALTER

Modifies an existing table.

```
ALTER TABLE Products ADD Category VARCHAR(50);
```

3. DROP

Deletes a table.

```
DROP TABLE Products;
```

4. TRUNCATE

Removes all rows from a table.

```
TRUNCATE TABLE Products;
```

Data Manipulation Language (DML)

1. SELECT

Retrieves data from a table.

```
SELECT * FROM Products WHERE Price > 100;
```

2. INSERT

Adds a new row.

```
INSERT INTO Products (ProductID, ProductName, Price) VALUES (1, 'Laptop',
999.99);
```

3. UPDATE

Updates existing data.

```
UPDATE Products SET Price = 899.99 WHERE ProductID = 1;
```

4. DELETE

Removes specific rows.

```
DELETE FROM Products WHERE ProductID = 1;
```

Data Control Language (DCL)

1. GRANT

Gives permissions.

```
GRANT SELECT, INSERT ON Products TO User1;
```

2. REVOKE

Removes permissions.

```
REVOKE INSERT ON Products FROM User1;
```

Transaction Control Language (TCL)

1. COMMIT

Saves changes permanently.

```
BEGIN TRANSACTION;
INSERT INTO Products VALUES (2, 'Tablet', 399.99);
COMMIT;
```

2. ROLLBACK

Undoes changes.

```
BEGIN TRANSACTION;
INSERT INTO Products VALUES (3, 'Phone', 299.99);
ROLLBACK;
```

3. SAVEPOINT

Defines a point for partial rollback.

```
SAVEPOINT Save1;
ROLLBACK TO Save1;
```

Advanced SQL Queries

Aggregate Functions

```
SELECT AVG(Price) AS AveragePrice FROM Products;
```

Joins

```
SELECT Orders.OrderID, Customers.Name
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.ID;
```

Subqueries

```
SELECT ProductName FROM Products
WHERE Price > (SELECT AVG(Price) FROM Products);
```

Common Table Expressions (CTEs)

```
WITH SalesData AS (
    SELECT ProductID, SUM(Quantity) AS TotalSales FROM Sales GROUP BY
ProductID
)
SELECT * FROM SalesData WHERE TotalSales > 100;
```

Indexes

Creating an Index

```
CREATE INDEX idx_ProductName ON Products (ProductName);
```

Stored Procedures and Functions

Stored Procedure

```
CREATE PROCEDURE GetProductsByCategory (@Category VARCHAR(50))
AS
BEGIN
    SELECT * FROM Products WHERE Category = @Category;
END;
```

Scalar Function

```
CREATE FUNCTION GetDiscountedPrice(@Price DECIMAL, @Discount INT)
RETURNS DECIMAL AS
BEGIN
    RETURN @Price - (@Price * @Discount / 100);
END;
```

Triggers

```
CREATE TRIGGER trg_AfterInsert
ON Products
AFTER INSERT
AS
BEGIN
    PRINT 'A new product has been added.';
END;
```

Views

Creating a View

```
CREATE VIEW ProductSummary AS
SELECT ProductName, Price FROM Products;
```

Performance Optimization

Execution Plan

```
SET STATISTICS TIME ON;
SELECT * FROM Products;
```

Security

Dynamic Data Masking

```
CREATE TABLE Customers (
    ID INT,
    Name VARCHAR(100) MASKED WITH (FUNCTION = 'default()')
);
```

Data Warehousing and Analytics

Pivot Example

```
SELECT *
FROM (SELECT Year, Sales FROM SalesData) AS SourceTable
PIVOT (
    SUM(Sales) FOR Year IN ([2021], [2022])
) AS PivotTable;
```

SQL Server-Specific Features

Temporal Tables

```
CREATE TABLE EmployeeHistory (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(100),
    ValidFrom DATETIME GENERATED ALWAYS AS ROW START,
    ValidTo DATETIME GENERATED ALWAYS AS ROW END
) WITH (SYSTEM_VERSIONING = ON);
```

This document covers key SQL topics along with practical examples. Let me know if you want more examples or details!