SQL Basics

1. SQL Introduction

SELECT * FROM Customers;

2. SQL Syntax

SELECT column1, column2 FROM table_name;

3. SQL SELECT Statement

SELECT * FROM Customers;

4. SQL SELECT DISTINCT

SELECT DISTINCT Country FROM Customers;

5. SQL WHERE Clause

SELECT * FROM Customers WHERE Country='Germany';

Operators

6. SQL AND Operator

SELECT * FROM Customers WHERE Country='Germany' AND City='Berlin';

7. SQL OR Operator

SELECT * FROM Customers WHERE Country='Germany' OR Country='France';

8. SQL NOT Operator

SELECT * FROM Customers WHERE NOT Country='Germany';

9. SQL IN Operator

SELECT * FROM Customers WHERE Country IN ('Germany', 'France');

10. SQL BETWEEN Operator

SELECT * FROM Products WHERE Price BETWEEN 10 AND 20;

Sorting and Filtering

11. SQL ORDER BY

SELECT * FROM Customers ORDER BY Country ASC;

12. SQL LIKE

SELECT * FROM Customers WHERE CustomerName LIKE 'A%';

13. SQL Wildcards

SELECT * FROM Customers WHERE CustomerName LIKE '_r%';

14. SQL LIMIT

SELECT * FROM Customers LIMIT 5;

Data Manipulation

15. SQL INSERT INTO

INSERT INTO Customers (CustomerName, Country) VALUES ('Alfred', 'USA');

16. SQL UPDATE

UPDATE Customers SET City='Berlin' WHERE CustomerID=1;

17. SQL DELETE

DELETE FROM Customers WHERE CustomerID=1;

Functions

18. SQL MIN() and MAX()

```
SELECT MIN(Price), MAX(Price) FROM Products;
```

19. SQL COUNT(), AVG(), SUM()

```
SELECT COUNT(CustomerID), AVG(Price), SUM(Price) FROM Products;
```

Joins

20. SQL INNER JOIN

```
SELECT Orders.OrderID, Customers.CustomerName

FROM Orders

INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;
```

21. SQL LEFT JOIN

```
SELECT Customers.CustomerName, Orders.OrderID

FROM Customers

LEFT JOIN Orders ON Customers.CustomerID=Orders.CustomerID;
```

22. SQL RIGHT JOIN

```
SELECT Customers.CustomerName, Orders.OrderID

FROM Customers

RIGHT JOIN Orders ON Customers.CustomerID=Orders.CustomerID;
```

23. SQL FULL JOIN

```
SELECT Customers.CustomerName, Orders.OrderID

FROM Customers

FULL JOIN Orders ON Customers.CustomerID=Orders.CustomerID;
```

Advanced Queries

24. SQL GROUP BY

```
SELECT Country, COUNT(CustomerID)
FROM Customers
GROUP BY Country;
```

25. SQL HAVING

```
SELECT Country, COUNT(CustomerID)
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

26. SQL EXISTS

```
SELECT CustomerName
FROM Customers
WHERE EXISTS (SELECT OrderID FROM Orders WHERE Customers.CustomerID=Orders.CustomerID);
```

Set Operations

27. SQL UNION

```
SELECT City FROM Customers
UNION
SELECT City FROM Suppliers;
```

28. SQL UNION ALL

```
SELECT City FROM Customers
UNION ALL
SELECT City FROM Suppliers;
```

Data Definition

29. SQL CREATE DATABASE

```
CREATE DATABASE myDatabase;
```

30. SQL CREATE TABLE

```
CREATE TABLE Customers (
CustomerID int,
CustomerName varchar(255),
Country varchar(255)
);
```

31. SQL ALTER TABLE

```
ALTER TABLE Customers ADD Email varchar(255);
```

32. SQL DROP TABLE

```
DROP TABLE Customers;
```

33. SQL DROP DATABASE

```
DROP DATABASE myDatabase;
```

Indexes

34. SQL CREATE INDEX

```
CREATE INDEX idx_customername ON Customers(CustomerName);
```

35. SQL DROP INDEX

```
DROP INDEX idx_customername;
```

Views

36. SQL CREATE VIEW

```
CREATE VIEW [Current Customers] AS

SELECT CustomerName, Country FROM Customers WHERE Country='USA';
```

37. SQL DROP VIEW

```
DROP VIEW [Current Customers];
```

Triggers

38. SQL CREATE TRIGGER

```
CREATE TRIGGER after_insert

AFTER INSERT ON Customers

FOR EACH ROW

BEGIN

INSERT INTO Logs(description) VALUES('New customer added');

END;
```

39. SQL DROP TRIGGER

```
DROP TRIGGER after_insert;
```

Stored Procedures

40. SQL CREATE PROCEDURE

```
CREATE PROCEDURE GetCustomerByCountry (IN CountryName VARCHAR(255))

BEGIN

SELECT * FROM Customers WHERE Country = CountryName;

END;
```

41. SQL CALL PROCEDURE

```
CALL GetCustomerByCountry('USA');
```

42. SQL DROP PROCEDURE

```
DROP PROCEDURE GetCustomerByCountry;
```

More Topics (to complete all 137 topics)

... (You can continue adding additional topics and examples systematically)

Constraints

43. SQL PRIMARY KEY

```
CREATE TABLE Customers (

CustomerID int PRIMARY KEY,

CustomerName varchar(255)

);
```

44. SQL FOREIGN KEY

```
CREATE TABLE Orders (
OrderID int,
CustomerID int,
FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

45. SQL NOT NULL

```
CREATE TABLE Customers (
CustomerID int NOT NULL,
CustomerName varchar(255) NOT NULL
);
```

46. SQL UNIQUE

```
CREATE TABLE Customers (
CustomerID int UNIQUE,
CustomerName varchar(255)
);
```

47. SQL DEFAULT

```
CREATE TABLE Customers (
CustomerID int,
Country varchar(255) DEFAULT 'USA'
);
```

48. SQL CHECK

```
CREATE TABLE Products (
```

ProductID int,

Price decimal CHECK (Price >= 0)

);

Auto Increment

49. SQL AUTO INCREMENT

CREATE TABLE Customers (

CustomerID int AUTO_INCREMENT PRIMARY KEY,

CustomerName varchar(255)

);

Advanced Functions

50. SQL CASE

SELECT OrderID,

CASE

WHEN Quantity > 30 THEN 'High'

ELSE 'Low'

END AS QuantityLevel

FROM Orders;

51. SQL CAST

SELECT CAST(Price AS int) FROM Products;

52. SQL CONVERT

```
SELECT CONVERT(VARCHAR, GETDATE(), 23);
```

Subqueries

53. SQL Subqueries in SELECT

```
SELECT CustomerName,
(SELECT COUNT(OrderID) FROM Orders WHERE Orders.CustomerID = Customers.CustomerID)
AS OrderCount
FROM Customers;
```

54. SQL Subqueries in WHERE

```
SELECT * FROM Customers WHERE Country = (SELECT Country FROM Suppliers WHERE
SupplierID = 1);
```

Transactions

55. SQL Transactions

```
BEGIN TRANSACTION;
UPDATE Accounts SET Balance = Balance - 100 WHERE AccountID = 1;
UPDATE Accounts SET Balance = Balance + 100 WHERE AccountID = 2;
COMMIT;
```

Window Functions

56. SQL RANK

```
SELECT CustomerID,
RANK() OVER (PARTITION BY Country ORDER BY Sales DESC) AS Rank
```

```
FROM Customers;
```

57. SQL ROW_NUMBER

```
SELECT CustomerID,
ROW_NUMBER() OVER (ORDER BY Country) AS RowNumber
FROM Customers;
```

Recursive Queries

58. SQL WITH Recursive

```
WITH EmployeeHierarchy AS (
SELECT EmployeeID, ManagerID
FROM Employees
WHERE ManagerID IS NULL
UNION ALL
SELECT e.EmployeeID, e.ManagerID
FROM Employees e
INNER JOIN EmployeeHierarchy eh ON e.ManagerID = eh.EmployeeID
)
SELECT * FROM EmployeeHierarchy;
```

JSON Data

59. SQL JSON Data

```
SELECT ProductID, ProductName, ProductDetails
```

FROM Products

WHERE JSON_VALUE(ProductDetails, '$.Color') = 'Red';

Pivot Tables

60. SQL Pivot

SELECT *

FROM (SELECT Year, Sales FROM SalesData) AS SourceTable

PIVOT (SUM(Sales) FOR Year IN ([2019], [2020], [2021])) AS PivotTable;

61. SQL CROSS JOIN

Example:

sql

Copy code

SELECT Employees.Name, Departments.DepartmentName

FROM Employees

CROSS JOIN Departments;

62. SQL Self Join

Example:

sql

Copy code

```
SELECT A.EmployeeName AS Employee, B.EmployeeName AS Manager
FROM Employees A, Employees B
WHERE A.ManagerID = B.EmployeeID;
```

63. SQL CTE (Common Table Expressions)

Example:

```
sql
Copy code
WITH EmployeeCTE AS (
SELECT EmployeeID, ManagerID
FROM Employees
WHERE ManagerID IS NOT NULL
)
SELECT * FROM EmployeeCTE;
```

64. SQL Merge Statements

Example:

```
sql
Copy code
MERGE INTO TargetTable AS Target
USING SourceTable AS Source
ON Target.ID = Source.ID
```

WHEN MATCHED THEN

UPDATE SET Target.Name = Source.Name

WHEN NOT MATCHED THEN

INSERT (ID, Name) VALUES (Source.ID, Source.Name);

65. SQL Temporary Tables

Example:

sql

Copy code

CREATE TEMPORARY TABLE TempOrders (OrderID int, Amount decimal);

INSERT INTO TempOrders VALUES (1, 100.00);

66. SQL Data Import/Export

Example:

sql

Copy code

-- Export data to a CSV file

SELECT * FROM Customers INTO OUTFILE '/tmp/customers.csv'

FIELDS TERMINATED BY ',' ENCLOSED BY '"'

LINES TERMINATED BY '\n';

67. SQL Backup and Restore

Example:

sql

Copy code

```
-- Backup database
BACKUP DATABASE myDatabase TO DISK = 'C:\\backup\\myDatabase.bak';
-- Restore database
RESTORE DATABASE myDatabase FROM DISK = 'C:\\backup\\myDatabase.bak';
```

68. SQL Data Encryption

Example:

sql

Copy code

```
CREATE CERTIFICATE MyCertificate WITH SUBJECT = 'Encryption';
CREATE SYMMETRIC KEY MyKey WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE MyCertificate;
```

69. SQL User-Defined Functions

Example:

sql

Copy code

```
CREATE FUNCTION GetDiscount (@TotalAmount decimal)
RETURNS decimal AS
```

BEGIN

RETURN (@TotalAmount * 0.10);

END;

70. SQL Error Handling

Example:

sql

Copy code

BEGIN TRY

INSERT INTO Customers (CustomerID, Name) VALUES (1, 'John Doe');

END TRY

BEGIN CATCH

SELECT ERROR_MESSAGE();

END CATCH;

71. SQL Full-Text Search

Example:

sql

Copy code

SELECT * FROM Articles WHERE CONTAINS(ArticleText, 'database');

72. SQL Hierarchical Queries

Example:

sql

Copy code

START WITH EmployeeID = 1

CONNECT BY PRIOR ManagerID = EmployeeID;

73. SQL Unpivot

Example:

sql

Copy code

SELECT Year, Quarter, Sales

FROM (SELECT * FROM SalesData) AS SourceTable

UNPIVOT (Sales FOR Quarter IN ([Q1], [Q2], [Q3], [Q4])) AS Unpivoted;

74. SQL Bitwise Operators

Example:

sql

Copy code

SELECT EmployeeID, Permissions & 1 AS CanEdit FROM Employees;

75. SQL Permissions and Security

Example:

sql

Copy code

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

REVOKE DELETE ON Customers FROM User1;
```

76. SQL Linked Servers

Example:

sql

Copy code

```
EXEC sp_addlinkedserver 'RemoteServer';

SELECT * FROM RemoteServer.DatabaseName.SchemaName.TableName;
```

77. SQL Profiling and Optimization

Example:

sql

Copy code

```
SET STATISTICS IO ON;

SELECT * FROM LargeTable WHERE ColumnName = 'Value';

SET STATISTICS IO OFF;
```

78. SQL Indexing Best Practices

Example:

sql

Copy code

```
CREATE INDEX idx_customer_name ON Customers (CustomerName);
```

79. SQL Performance Tuning

Example:

sql

Copy code

```
UPDATE STATISTICS Customers;

DBCC DROPCLEANBUFFERS;
```

80. SQL Data Types

Example:

sql

Copy code

```
CREATE TABLE DataTypesExample (

ID INT,

Name VARCHAR(50),

BirthDate DATE

);
```

81. SQL Operators (Arithmetic, Comparison, Logical)

Example:

sql

Copy code

```
SELECT * FROM Products WHERE Price > 50 AND Stock < 10;
```

82. SQL Metadata Queries

Example:

sql

Copy code

```
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE';
```

83. SQL Data Masking

Example:

sql

Copy code

```
CREATE TABLE Employees (SSN VARCHAR(11) MASKED WITH (FUNCTION = 'default()'));
```

84. SQL Partitioning

Example:

sql

Copy code

CREATE PARTITION FUNCTION PartitionFunction (INT)

AS RANGE LEFT FOR VALUES (1, 100, 1000);

85. SQL Advanced Joins (Anti Joins, Semi Joins)

Example:

sql

Copy code

-- Anti Join Example

SELECT * FROM Customers

WHERE NOT EXISTS (SELECT 1 FROM Orders WHERE Customers.CustomerID = Orders.CustomerID);

SQL Topics (86-137) with Examples

86. SQL Table Variables

Example:

```sql

DECLARE @TableVariable TABLE (ID INT, Name NVARCHAR(50));

INSERT INTO @TableVariable VALUES (1, 'Alice'), (2, 'Bob');

SELECT * FROM @TableVariable;

```

87. SQL Dynamic Queries

Example:

```sql
DECLARE @SQL NVARCHAR(MAX);
SET @SQL = 'SELECT * FROM Employees WHERE DepartmentID = 1';
EXEC sp_executesql @SQL;
```

88. SQL Cursor

Example:

```sql
DECLARE EmployeeCursor CURSOR FOR
SELECT Name FROM Employees;
OPEN EmployeeCursor;
FETCH NEXT FROM EmployeeCursor;
WHILE @@FETCH_STATUS = 0
BEGIN
-- Process each row
FETCH NEXT FROM EmployeeCursor;
END;
CLOSE EmployeeCursor;
```

DEALLOCATE EmployeeCursor;

```

#### 89. SQL Temporary Stored Procedures

**Example:**

```sql

CREATE PROCEDURE #TempProc

AS

BEGIN

SELECT GETDATE() AS CurrentDate;

END;

EXEC #TempProc;

DROP PROCEDURE #TempProc;

```

90. SQL Data Validation

Example:

```sql

CREATE TABLE Orders (

OrderID INT,

Quantity INT CHECK (Quantity > 0)

);

```
INSERT INTO Orders VALUES (1, 10);

-- This will fail: INSERT INTO Orders VALUES (2, -5);
```

91. SQL Data Quality Checks

Example:

```sql
SELECT * FROM Customers WHERE Email NOT LIKE '%@%';
```

92. SQL Backup Automation

Example:

```sql
USE master;

EXEC sp_add_job @job_name = 'DatabaseBackupJob';

EXEC sp_add_jobstep @job_name = 'DatabaseBackupJob',

@step_name = 'BackupStep',

@command = 'BACKUP DATABASE MyDB TO DISK = ''C:\Backups\MyDB.bak''';
```

93. SQL Report Generation

Example:

```sql

SELECT Department, COUNT(*) AS TotalEmployees

FROM Employees

GROUP BY Department

ORDER BY TotalEmployees DESC;

```

94. SQL Error Logs

Example:

```sql

EXEC sp_readerrorlog;

```

95. SQL Collations

Example:

```sql

SELECT Name COLLATE Latin1_General_CI_AS AS SortedName

FROM Employees

ORDER BY SortedName;

```

96. SQL Database Snapshots

Example:

```sql
CREATE DATABASE SnapshotDB
ON (
NAME = MyDB,
FILENAME = 'C:\Snapshots\MyDB.ss'
)
AS SNAPSHOT OF MyDB;
```

97. SQL Row-Level Security

Example:

```sql
CREATE SECURITY POLICY EmployeePolicy
ADD FILTER PREDICATE dbo.FilterFunction(UserID)
ON Employees;
```

98. SQL Version Control

Example:

```sql
-- Using Git for SQL script management
```

-- Save SQL scripts as .sql files and commit changes to Git.

```

#### 99. SQL Clustered and Non-Clustered Indexes

**Example:**

```sql

CREATE CLUSTERED INDEX idx_Clustered ON Employees(EmployeeID);

CREATE NONCLUSTERED INDEX idx_NonClustered ON Employees(DepartmentID);

```

100. SQL Sparse Columns

Example:

```sql

CREATE TABLE Product

(

ProductID INT,

ProductName NVARCHAR(50),

Description NVARCHAR(MAX) SPARSE

);

```

101. SQL XML Data
```

Example:

```sql
SELECT ProductID, ProductDetails.query('/Product/Description') AS Description
FROM Products;
```

102. SQL Multi-Valued Parameters

Example:

```sql
DECLARE @IDs TABLE (ID INT);
INSERT INTO @IDs VALUES (1), (2), (3);
SELECT * FROM Employees WHERE EmployeeID IN (SELECT ID FROM @IDs);
```

103. SQL Scheduling Jobs

Example:

```sql
EXEC sp_add_job @job_name = 'DailyReport';
EXEC sp_add_jobschedule @job_name = 'DailyReport',
@schedule_name = 'DailySchedule',
@freq_type = 4, -- Daily
@freq_interval = 1;
```

```

#### 104. SQL Materialized Views

**Example:**

```sql

CREATE MATERIALIZED VIEW EmployeeSummary AS

SELECT DepartmentID, COUNT(*) AS TotalEmployees

FROM Employees

GROUP BY DepartmentID;

```

105. SQL Encryption by Keys

Example:

```sql

CREATE SYMMETRIC KEY SymKey

WITH ALGORITHM = AES_256

ENCRYPTION BY PASSWORD = 'StrongPassword';

OPEN SYMMETRIC KEY SymKey DECRYPTION BY PASSWORD = 'StrongPassword';

```

106. SQL Custom Error Messages

Example:
```

```sql

RAISERROR ('Invalid input detected', 16, 1);

```

107. SQL Plan Cache

Example:

```sql

SELECT * FROM sys.dm_exec_cached_plans;

```

108. SQL Query Store

Example:

```sql

ALTER DATABASE MyDB SET QUERY_STORE = ON;

SELECT * FROM sys.query_store_query;

```

109. SQL Partition Switching

Example:

```sql

ALTER TABLE Orders SWITCH PARTITION 1 TO ArchiveOrders PARTITION 1;

```

110. SQL Sharding

Example:

```sql

-- Shard key example

CREATE TABLE Orders_1 (OrderID INT PRIMARY KEY, CustomerID INT);

CREATE TABLE Orders_2 (OrderID INT PRIMARY KEY, CustomerID INT);

```

.

SQL Topics (111-137) with Examples

111. SQL Multi-Tenant Databases

Example:

```sql

CREATE SCHEMA Tenant1;

CREATE SCHEMA Tenant2;

CREATE TABLE Tenant1.Users (UserID INT, UserName NVARCHAR(50));

CREATE TABLE Tenant2.Users (UserID INT, UserName NVARCHAR(50));

```

112. SQL Schema Management

Example:

```sql
ALTER SCHEMA Sales TRANSFER Orders.OldTableName;

DROP SCHEMA ObsoleteSchema;
```

113. SQL Collation Conflicts

Example:

```sql
SELECT Name COLLATE Latin1_General_BIN AS SortedName
FROM Employees
ORDER BY SortedName;
```

114. SQL Extended Properties

Example:

```sql
EXEC sp_addextendedproperty @name = 'Documentation',
@value = 'This table stores customer data',
@level0type = 'SCHEMA', @level0name = 'dbo',
@level1type = 'TABLE', @level1name = 'Customers';
```

115. SQL Data Annotations

Example:

```sql

-- Not natively supported but use extended properties

EXEC sp_addextendedproperty @name = 'ValidationRule',

@value = 'Must be non-negative',

@level0type = 'SCHEMA', @level0name = 'dbo',

@level1type = 'TABLE', @level1name = 'Products',

@level2type = 'COLUMN', @level2name = 'Price';

```

116. SQL Recursive CTEs

Example:

```sql

WITH EmployeeHierarchy AS (

SELECT EmployeeID, ManagerID, Name, 1 AS Level

FROM Employees

WHERE ManagerID IS NULL

UNION ALL

SELECT e.EmployeeID, e.ManagerID, e.Name, Level + 1

FROM Employees e
```

INNER JOIN EmployeeHierarchy eh ON e.ManagerID = eh.EmployeeID

)

SELECT * FROM EmployeeHierarchy;

```

#### 117. SQL Server Agent

**Example:**

```sql

EXEC msdb.dbo.sp_add_job @job_name = 'DailyDataCleanup';

EXEC msdb.dbo.sp_add_jobstep @job_name = 'DailyDataCleanup',

@step_name = 'CleanupStep',

@command = 'DELETE FROM TempData WHERE CreatedDate < GETDATE() - 30';

```

118. SQL Replication

Example:

```sql

-- Configure replication

EXEC sp_addpublication @publication = 'SalesData';

EXEC sp_addsubscription @publication = 'SalesData', @subscriber = 'SubscriberServer';

```
```

119. SQL Always-On Availability Groups

Example:

```sql
CREATE AVAILABILITY GROUP [AG1]
FOR DATABASE [MyDatabase]
REPLICA ON
N'PrimaryServer' WITH (ROLE = PRIMARY),
N'SecondaryServer' WITH (ROLE = SECONDARY);
```

120. SQL Distributed Queries

Example:

```sql
SELECT * FROM OPENQUERY(RemoteServer, 'SELECT * FROM Customers WHERE Country = "USA"');
```

121. SQL Blockchain Tables

Example:

```sql
CREATE TABLE LedgerTable
(
```

LedgerID INT PRIMARY KEY,

LedgerValue NVARCHAR(50)

) WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.LedgerHistory));

```

#### 122. SQL In-Memory OLTP

**Example:**

```sql

CREATE TABLE InMemoryTable (

ID INT NOT NULL PRIMARY KEY NONCLUSTERED HASH WITH (BUCKET_COUNT = 1000000),

Name NVARCHAR(50) NOT NULL

) WITH (MEMORY_OPTIMIZED = ON);

```

123. SQL Temporal Tables

Example:

```sql

CREATE TABLE TemporalTable (

ID INT PRIMARY KEY,

Name NVARCHAR(50),

ValidFrom DATETIME2 GENERATED ALWAYS AS ROW START,

ValidTo DATETIME2 GENERATED ALWAYS AS ROW END

) WITH (SYSTEM_VERSIONING = ON);

```

124. SQL JSON_MODIFY

Example:

```sql

DECLARE @json NVARCHAR(MAX) = '{"Name": "Alice", "Age": 25}';

SET @json = JSON_MODIFY(@json, '$.Age', 26);

SELECT @json;

```

125. SQL Graph Databases

Example:

```sql

CREATE TABLE Persons (ID INT PRIMARY KEY) AS NODE;

CREATE TABLE WorksAt AS EDGE;

INSERT INTO WorksAt VALUES ((SELECT $node_id FROM Persons WHERE ID = 1), (SELECT $node_id FROM Companies WHERE ID = 10));

```

126. SQL Geospatial Data

Example:
```

```sql

CREATE TABLE Locations (

ID INT PRIMARY KEY,

GeoLocation GEOGRAPHY

);

INSERT INTO Locations VALUES (1, GEOGRAPHY::Point(47.6062, -122.3321, 4326));

```

127. SQL Data Virtualization

Example:

```sql

SELECT * FROM ExternalTable;

-- Data resides in a remote data source but is queried like a local table.

```

128. SQL Virtual Tables

Example:

```sql

CREATE VIEW ActiveCustomers AS

SELECT * FROM Customers WHERE IsActive = 1;

```

129. SQL Query Hints

Example:

```sql
SELECT * FROM Orders WITH (NOLOCK);
```

130. SQL SET Operators (INTERSECT, EXCEPT)

Example:

```sql
SELECT Name FROM Customers
INTERSECT
SELECT Name FROM Employees;

SELECT Name FROM Customers
EXCEPT
SELECT Name FROM Employees;
```

131. SQL Analytic Functions (Lag, Lead)

Example:

```sql
SELECT Name,
LAG(Salary) OVER (ORDER BY Name) AS PreviousSalary,
```

LEAD(Salary) OVER (ORDER BY Name) AS NextSalary

FROM Employees;

```

#### 132. SQL Inline Table-Valued Functions

**Example:**

```sql

CREATE FUNCTION GetEmployeesByDepartment(@DeptID INT)

RETURNS TABLE

AS

RETURN (SELECT * FROM Employees WHERE DepartmentID = @DeptID);

```

133. SQL System Functions

Example:

```sql

SELECT GETDATE() AS CurrentDate, SYSTEM_USER AS CurrentUser;

```

134. SQL CLR Integration

Example:

```sql

```
CREATE ASSEMBLY MyCLRAssembly

FROM 'C:\MyCLRAssembly.dll'

WITH PERMISSION_SET = SAFE;
```

135. SQL Big Data Clusters

Example:

```sql
-- Integration with Hadoop or Spark for big data processing.

-- Query big data clusters using PolyBase:

SELECT * FROM ExternalBigDataTable;
```

136. SQL Graph Queries

Example:

```sql
SELECT p.Name

FROM Persons p, WorksAt w

WHERE MATCH(p-(w)->c) AND c.Name = 'TechCorp';
```

137. SQL Machine Learning Services

Example:

```sql
EXEC sp_execute_external_script
@language = N'Python',
@script = N'print("Hello from SQL ML Services!")';
```