

# SQL

## Notes for Professionals

### Chapter 21: CREATE TABLE

**Parameter:** tablename The name of the table.  
columns Contains an 'enumeration' of all the columns that the table have. See: [Create a New Table](#).  
The CREATE TABLE statement is used to create a new table in the database. A table definition consists of columns, their types, and any integrity constraints.

#### Section 21.1: Create Table From Select

You may want to create a duplicate of a table.

```
CREATE TABLE CloneOfEmployees AS SELECT * FROM Employees;
```

You can use any of the other features of a SELECT statement to modify the data before it. The columns of the new table are automatically created according to the selected rows.

#### Section 21.2: Create a New Table

A basic Employees table, containing an ID, and the employee's first and last name as can be created using

```
CREATE TABLE Employees (  
  id int identity(1,1) primary key not null,  
  first_name varchar(20) not null,  
  last_name varchar(20) not null,  
  phone_number varchar(15) not null  
);
```

This example is specific to [Microsoft SQL Server](#).

CREATE TABLE creates a new table in the database, followed by the table name.

This is then followed by the list of column names and their properties, such as

id int identity(1,1) not null	Meaning
Value	the column's name
id	is the data type.
not null	states that columns will have auto-generated values starting from 1.
identity(1,1)	states that all values in this column will have unique values.
primary key	states that this column cannot have null values.
not null	states that this column cannot have null values.

#### Section 21.3: CREATE TABLE with FOREIGN KEY

Below you could find the table Employees with a reference to the table Customers.

### Chapter 42: Functions (Aggregate)

#### Section 42.1: Conditional aggregation

Payments Table

Customer Payment\_type Amount

Peter	Credit	100
Peter	Credit	300
John	Credit	1000
John	Debit	500

```
select customer,  
sum(case when payment_type = 'credit' then amount else 0 end) as credit,  
sum(case when payment_type = 'debit' then amount else 0 end) as debit  
from payments  
group by customer;
```

Result:

Customer Credit Debit

Peter	400	0
John	1000	500

```
select customer,  
sum(case when payment_type = 'credit' then 1 else 0 end) as credit_transaction_count,  
sum(case when payment_type = 'debit' then 1 else 0 end) as debit_transaction_count  
from payments  
group by customer;
```

Result:

Customer credit\_transaction\_count debit\_transaction\_count

Peter	2	0
John	1	1

#### Section 42.2: List Concatenation

Partial credit to the SO answer.

List Concatenation aggregates a column or expression by combining the values into a single string for each group string to delimit each value (either blank or a comma when omitted) and the order of the values in the result is specified. While it is not part of the SQL standard, every major relational database vendor supports it in their way.

MySQL

```
SELECT ColumnA  
FROM TableAname  
GROUP BY ColumnA  
ORDER BY ColumnA;
```

Oracle & DB2

```
SELECT ColumnA  
LISTAGG(ColumnB, ',' ) WITHIN GROUP (ORDER BY ColumnB) AS ColumnB  
FROM TableAname;
```

SQL Notes for Professionals

### Chapter 52: Subqueries

#### Section 52.1: Subquery in FROM clause

A subquery in a FROM clause acts similarly to a temporary table that is generated during the execution of a query and used afterwards.

```
SELECT Managers.Id, Employees.Salary  
FROM (  
  SELECT Id  
  FROM Employees  
  WHERE ManagerId IS NULL  
 ) AS Managers  
JOIN Employees ON Managers.Id = Employees.Id;
```

#### Section 52.2: Subquery in SELECT clause

```
SELECT  
  Id,  
  Name,  
  Salary  
FROM Customers  
WHERE CustomerId = (SELECT MAX(Salary) FROM Employees);
```

#### Section 52.3: Subquery in WHERE clause

Use a subquery to filter the result set. For example this will return all employees with a salary equal to the highest paid employees.

```
SELECT *  
FROM Employees  
WHERE Salary = (SELECT MAX(Salary) FROM Employees);
```

#### Section 52.4: Correlated Subqueries

Correlated (also known as Synchronized or Coordinated) Subqueries are nested queries that make references to the current row of their outer query.

```
SELECT EmployeeId  
FROM Employee AS whiter  
WHERE Salary = (  
  SELECT MAX(Salary)  
  FROM Employee AS Greer  
  WHERE Greer.DepartmentId = whiter.DepartmentId  
);
```

#### Section 52.5: Filter query results using query on different table

This query selects all employees not on the Supervisors table.

```
SELECT *
```

# Contents

<b>About</b>	1
<b>Chapter 1: Getting started with SQL</b>	2
Section 1.1: Overview	2
<b>Chapter 2: Identifier</b>	3
Section 2.1: Unquoted identifiers	3
<b>Chapter 3: Data Types</b>	4
Section 3.1: DECIMAL and NUMERIC	4
Section 3.2: FLOAT and REAL	4
Section 3.3: Integers	4
Section 3.4: MONEY and SMALLMONEY	4
Section 3.5: BINARY and VARBINARY	4
Section 3.6: CHAR and VARCHAR	5
Section 3.7: NCHAR and NVARCHAR	5
Section 3.8: UNIQUEIDENTIFIER	5
<b>Chapter 4: NULL</b>	6
Section 4.1: Filtering for NULL in queries	6
Section 4.2: Nullable columns in tables	6
Section 4.3: Updating fields to NULL	6
Section 4.4: Inserting rows with NULL fields	7
<b>Chapter 5: Example Databases and Tables</b>	8
Section 5.1: Auto Shop Database	8
Section 5.2: Library Database	10
Section 5.3: Countries Table	13
<b>Chapter 6: SELECT</b>	14
Section 6.1: Using the wildcard character to select all columns in a query	14
Section 6.2: SELECT Using Column Aliases	15
Section 6.3: Select Individual Columns	18
Section 6.4: Selecting specified number of records	19
Section 6.5: Selecting with Condition	20
Section 6.6: Selecting with CASE	20
Section 6.7: Select columns which are named after reserved keywords	21
Section 6.8: Selecting with table alias	21
Section 6.9: Selecting with more than 1 condition	22
Section 6.10: Selecting without Locking the table	23
Section 6.11: Selecting with Aggregate functions	23
Section 6.12: Select with condition of multiple values from column	24
Section 6.13: Get aggregated result for row groups	24
Section 6.14: Selection with sorted Results	25
Section 6.15: Selecting with null	25
Section 6.16: Select distinct (unique values only)	25
Section 6.17: Select rows from multiple tables	26
<b>Chapter 7: GROUP BY</b>	27
Section 7.1: Basic GROUP BY example	27
Section 7.2: Filter GROUP BY results using a HAVING clause	28
Section 7.3: USE GROUP BY to COUNT the number of rows for each unique entry in a given column	28
Section 7.4: ROLAP aggregation (Data Mining)	29

<b>Chapter 8: ORDER BY</b>	31
Section 8.1: Sorting by column number (instead of name)	31
Section 8.2: Use ORDER BY with TOP to return the top x rows based on a column's value	31
Section 8.3: Customized sorting order	32
Section 8.4: Order by Alias	32
Section 8.5: Sorting by multiple columns	33
<b>Chapter 9: AND &amp; OR Operators</b>	34
Section 9.1: AND OR Example	34
<b>Chapter 10: CASE</b>	35
Section 10.1: Use CASE to COUNT the number of rows in a column match a condition	35
Section 10.2: Searched CASE in SELECT (Matches a boolean expression)	36
Section 10.3: CASE in a clause ORDER BY	36
Section 10.4: Shorthand CASE in SELECT	36
Section 10.5: Using CASE in UPDATE	37
Section 10.6: CASE use for NULL values ordered last	37
Section 10.7: CASE in ORDER BY clause to sort records by lowest value of 2 columns	38
<b>Chapter 11: LIKE operator</b>	39
Section 11.1: Match open-ended pattern	39
Section 11.2: Single character match	40
Section 11.3: ESCAPE statement in the LIKE-query	40
Section 11.4: Search for a range of characters	41
Section 11.5: Match by range or set	41
Section 11.6: Wildcard characters	41
<b>Chapter 12: IN clause</b>	43
Section 12.1: Simple IN clause	43
Section 12.2: Using IN clause with a subquery	43
<b>Chapter 13: Filter results using WHERE and HAVING</b>	44
Section 13.1: Use BETWEEN to Filter Results	44
Section 13.2: Use HAVING with Aggregate Functions	45
Section 13.3: WHERE clause with NULL/NOT NULL values	45
Section 13.4: Equality	46
Section 13.5: The WHERE clause only returns rows that match its criteria	46
Section 13.6: AND and OR	46
Section 13.7: Use IN to return rows with a value contained in a list	47
Section 13.8: Use LIKE to find matching strings and substrings	47
Section 13.9: Where EXISTS	48
Section 13.10: Use HAVING to check for multiple conditions in a group	48
<b>Chapter 14: SKIP TAKE (Pagination)</b>	50
Section 14.1: Limiting amount of results	50
Section 14.2: Skipping then taking some results (Pagination)	50
Section 14.3: Skipping some rows from result	51
<b>Chapter 15: EXCEPT</b>	52
Section 15.1: Select dataset except where values are in this other dataset	52
<b>Chapter 16: EXPLAIN and DESCRIBE</b>	53
Section 16.1: EXPLAIN Select query	53
Section 16.2: DESCRIBE tablename:	53
<b>Chapter 17: EXISTS CLAUSE</b>	54
Section 17.1: EXISTS CLAUSE	54
<b>Chapter 18: JOIN</b>	55

Section 18.1: Self Join .....	55
Section 18.2: Differences between inner/outer joins .....	56
Section 18.3: JOIN Terminology: Inner, Outer, Semi, Anti. ....	59
Section 18.4: Left Outer Join .....	69
Section 18.5: Implicit Join .....	70
Section 18.6: CROSS JOIN .....	71
Section 18.7: CROSS APPLY & LATERAL JOIN .....	72
Section 18.8: FULL JOIN .....	73
Section 18.9: Recursive JOINS .....	74
Section 18.10: Basic explicit inner join .....	74
Section 18.11: Joining on a Subquery .....	75
<b>Chapter 19: UPDATE</b> .....	76
Section 19.1: UPDATE with data from another table .....	76
Section 19.2: Modifying existing values .....	77
Section 19.3: Updating Specified Rows .....	77
Section 19.4: Updating All Rows .....	77
Section 19.5: Capturing Updated records .....	77
<b>Chapter 20: CREATE Database</b> .....	78
Section 20.1: CREATE Database .....	78
<b>Chapter 21: CREATE TABLE</b> .....	79
Section 21.1: Create Table From Select .....	79
Section 21.2: Create a New Table .....	79
Section 21.3: CREATE TABLE With FOREIGN KEY .....	79
Section 21.4: Duplicate a table .....	80
Section 21.5: Create a Temporary or In-Memory Table .....	80
<b>Chapter 22: CREATE FUNCTION</b> .....	82
Section 22.1: Create a new Function .....	82
<b>Chapter 23: TRY/CATCH</b> .....	83
Section 23.1: Transaction In a TRY/CATCH .....	83
<b>Chapter 24: UNION / UNION ALL</b> .....	84
Section 24.1: Basic UNION ALL query .....	84
Section 24.2: Simple explanation and Example .....	85
<b>Chapter 25: ALTER TABLE</b> .....	86
Section 25.1: Add Column(s) .....	86
Section 25.2: Drop Column .....	86
Section 25.3: Add Primary Key .....	86
Section 25.4: Alter Column .....	86
Section 25.5: Drop Constraint .....	86
<b>Chapter 26: INSERT</b> .....	87
Section 26.1: INSERT data from another table using SELECT .....	87
Section 26.2: Insert New Row .....	87
Section 26.3: Insert Only Specified Columns .....	87
Section 26.4: Insert multiple rows at once .....	87
<b>Chapter 27: MERGE</b> .....	88
Section 27.1: MERGE to make Target match Source .....	88
Section 27.2: MySQL: counting users by name .....	88
Section 27.3: PostgreSQL: counting users by name .....	88
<b>Chapter 28: cross apply, outer apply</b> .....	90
Section 28.1: CROSS APPLY and OUTER APPLY basics .....	90

<b>Chapter 29: DELETE</b>	92
Section 29.1: DELETE all rows	92
Section 29.2: DELETE certain rows with WHERE	92
Section 29.3: TRUNCATE clause	92
Section 29.4: DELETE certain rows based upon comparisons with other tables	92
<b>Chapter 30: TRUNCATE</b>	94
Section 30.1: Removing all rows from the Employee table	94
<b>Chapter 31: DROP Table</b>	95
Section 31.1: Check for existence before dropping	95
Section 31.2: Simple drop	95
<b>Chapter 32: DROP or DELETE Database</b>	96
Section 32.1: DROP Database	96
<b>Chapter 33: Cascading Delete</b>	97
Section 33.1: ON DELETE CASCADE	97
<b>Chapter 34: GRANT and REVOKE</b>	99
Section 34.1: Grant/revoke privileges	99
<b>Chapter 35: XML</b>	100
Section 35.1: Query from XML Data Type	100
<b>Chapter 36: Primary Keys</b>	101
Section 36.1: Creating a Primary Key	101
Section 36.2: Using Auto Increment	101
<b>Chapter 37: Indexes</b>	102
Section 37.1: Sorted Index	102
Section 37.2: Partial or Filtered Index	102
Section 37.3: Creating an Index	102
Section 37.4: Dropping an Index, or Disabling and Rebuilding it	103
Section 37.5: Clustered, Unique, and Sorted Indexes	103
Section 37.6: Rebuild index	104
Section 37.7: Inserting with a Unique Index	104
<b>Chapter 38: Row number</b>	105
Section 38.1: Delete All But Last Record (1 to Many Table)	105
Section 38.2: Row numbers without partitions	105
Section 38.3: Row numbers with partitions	105
<b>Chapter 39: SQL Group By vs Distinct</b>	106
Section 39.1: Difference between GROUP BY and DISTINCT	106
<b>Chapter 40: Finding Duplicates on a Column Subset with Detail</b>	107
Section 40.1: Students with same name and date of birth	107
<b>Chapter 41: String Functions</b>	108
Section 41.1: Concatenate	108
Section 41.2: Length	108
Section 41.3: Trim empty spaces	109
Section 41.4: Upper & lower case	109
Section 41.5: Split	109
Section 41.6: Replace	110
Section 41.7: REGEXP	110
Section 41.8: Substring	110
Section 41.9: Stuff	110
Section 41.10: LEFT - RIGHT	110

[Click here to download full PDF material](#)