Introduction of SQL

SQL(Structured Query Language) ये RDBMS(Relational Database Management System) Language; data को store करने के लिए काफी मददगार Language है |

SQL में data को manipulate और retrieve किया जाता है |

SQL ये open-source language है |

SQL में User द्वारा कुछ records create किये जाते है | इस records को कुछ clauses और expression का इस्तेमाल करके modify, update, delete किया जाता है |

SQL ये RDBMS data को अपने अन्दर store करता है |

SQL की queries को MySQL, SQL Server, PostGre SQL, MS Access, Oracle इन किसी भी databases में store किया जाता है |

History of SQL

SQL का अविष्कार 1970 के आसपास में IBM के Donald Chamberlin और Raymond Boyce ने किया था |

जब SQL को बनाया गया था तब इसका नाम 'SEQUEL'(Structured English Query Language) रखा गया था | लेकिन 'SEQUEL' ये UK के aircraft company का नाम पहले से ही था | उसके बाद में 'SEQUEL' ये नाम बदलकर 'SQL' रखा गया |

**What is RDBMS ?**

RDBMS ये Relational DataBase Management System का संक्षिप्त रूप है |

MySQL, SQL Server, PostGre SQL, MS Access और Oracle ये सब database systems RDBMS के आधार पर बनाये गए है |

RDBMS data को tables, records और fields में structured किया जाता है |

RDBMS के database के tables में fields के rows/records होते है |

बहुत सारे RDBMS; SQL इस query language का इस्तेमाल करते है |

RDBMS ये कुछ operators का इस्तेमाल table के rows या data को manipulate करने के लिए किया जाता है |

**RDBMS Table**

RDBMS में databases होते है | उन हर database में एक या एक से ज्यादा tables होते है | इन tables में data को store किया जाता है |

Database Name : student\_info

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name : Student | | | |
| **id** | **first\_name** | **last\_name** | **percentages** |
| 1 | Ram | Raman | 83.25 |
| 2 | Shyam | Joshi | 56.85 |
| 3 | Madhulika | Tare | 92.45 |

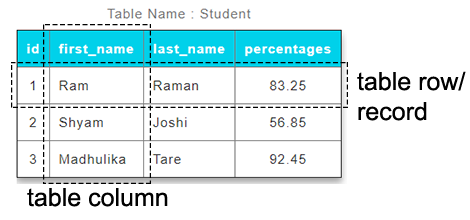
**RDBMS Field**

tables में data को store करने के लिए fields को set करना पड़ता है | निचे tables के fields दिए गए है |

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **first\_name** | **last\_name** | **Percentages** |

**RDBMS Rows/Records and Colums**

tables के data को rows और columns में store किया जाता है |



# Introduction to MySQL

MySQL एक RDBMS है जो कि तेज, और उपयोग करने में आसान है। यह कई छोटे बड़े उद्योगों द्वारा उपयोग किया जाता है। MySQL का निर्माण, MySQL AB नामक Swedish कम्‍पनी ने किया था।

MySQL की Popularity के पीछे के कारण –

1. MySQL कई सारे Operating Systems पर काम कर सकता है तथा कई सारी Programming Language के साथ काम कर सकता है जैसे PHP, PERL, C;, C++, JAVA etc.
2. MySQL बहुत बड़े डाटाबेस जिनमें 50 Million या उस से भी ज्‍यादा Rows हो को Support करता है।
3. यह तेजी के साथ काम करता है।
4. यह एक Standard SQL उपयोग करता है।
5. यह विशाल डाटा समूह के साथ भी अच्‍छे से काम कर लेता है।
6. यह पूरी तरह से मुफ्त हैं।

MySQL C और C++ से बनाया गया है। इसमें डाटा Tables में Store होता है जिसमें Columns और Rows होती है।

किसी Information को Categorized Form में Store करने में Database की महत्‍वपूर्ण भूमिका है।

MySQL दो अलग-अलग editions में उपलब्‍ध होता है :- Open Source MySQL Community Server और Proprietary Enterprise Server.

Database Management System एक Software होता है जिससे हम डाटा को स्‍टोर या mange कर सकते हैं। एक database में data tables के रूप में स्‍टोर होता है। यह data कैसा भी हो सकता है जैसे किसी का name या age या फिर किसी student की information जैसे roll no. id वगैरह।

चूँकि MySQL एक relational database हैं, इसलिए यह database के पूरे data को एक अकेली table के बजाए अलग अलग tables में स्टोर कर सकता है। इन tables के बीच में हम different relations भी establish कर सकते हैं, तथा अपनी जरूरत के अनुसार अलग अलग queries इन tables पर execute करके data को fetch, insert, या update कर सकते हैं।

MySQL से Communicate करने के लिए एक query language हैं SQL, जो almost हर database के साथ काम कर सकती हैं। SQL को हम directly भी और किसी scripting/programming language के साथ भी use कर सकते हैं।

**Feature of MySQL :-**

1) MySQL कई सारी programming और scripting language के साथ काम कर सकता हैं।

For e.g. php, JAVA

2) MySQL portable हैं।

3) इसमें हम दूसरे storage engines आसानी से add कर सकते हैं।

4) इसमें अलग-अलग Platforms पर काम किया जा सकता है।

5) MySQL का storage engine बहुत unique है और इसलिए इसका performance भी बहुत high होता है।

### SQL - What is SQL

SQL ये Structured Query Language का संक्षिप्त रूप है |

SQL ये RDBMS की बहुत ही महत्वपूर्ण language है |

SQL में उसके clauses के साथ Statements का इस्तेमाल data को store, update, delete और manipulate करने के लिए किया जाता है |

SQL में create, select, insert, update, delete और drop और आदि statements का इस्तेमाल किया जाता है |

#### SQL Important Statements

SQL statements आसानी से समझ आते है क्योंकि वो plain english जैसे होते है |

**create :** नया database या table create किया जाता है |

**select :** table को select किया जाता है |

**insert :** table में rows/records को insert किया जाता है |

**update :** table में rows/records को update किया जाता है |

**delete :** table में से rows/records को delete किया जाता है |

**drop :** table को delete किया जाता है |

### SQL - SQL Syntax

Database, table और उनके rows और columns के data को manipulate करने के लिए syntax को समझना बहुत ही जरुरी होता है |

SQL में statement के शुरुआत पर select, create, insert, delete, update, alter या drop जैसे keywords का इस्तेमाल किया जाता है |

SQL में सभी keywords case-insensitive होते है For Example, SELECT, CREATE इत्यादी | लेकिन सामान्यतः सभी keywords को uppercase में ही लिखा जाता है |

SQL में हर statement के आखिर पर semi-colon(;) को दिया जाता है |

**Sample Table**

निचे 'Student' नाम का table दिया गया है |

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name : Student | | | |
| **Id** | **first\_name** | **last\_name** | **percentages** |
| 1 | Ram | Raman | 83.25 |
| 2 | Shyam | Joshi | 56.85 |
| 3 | Madhulika | Tare | 92.45 |

**SQL Syntax**

SELECT column1, column2,..columnN FROM table\_Name

or

SELECT column1, column2,..columnN

FROM table\_Name

**Example**

SELECT \* FROM Student;

Output :

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **first\_name** | **last\_name** | **percentages** |
| 1 | Ram | Raman | 83.25 |
| 2 | Shyam | Joshi | 56.85 |
| 3 | Madhulika | Tare | 92.45 |

SQL के statements और clauses के syntax हर topic में दिए गए है |

### SQL - Data Types

SQL अलग-अलग Data Types रखता है जैसे कि, Numeric, Characters, Binary, date और time जैसे Data Types होते है |

जब SQL में table को create करना पड़ता है तब table के हर column के लिए उसका data type देना पड़ता है |

User अपने जरुरत के मुताबिक data type को चुन सकता है |

**Numeric Data Types in SQL**

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Syntax** | **Range** |
| Integer | INT | -231 to +231-1(-2,14,74,83,648 to +2,14,74,83,647) |
| Small Integer | SMALLINT | -215 to +215-1(-32768 to +32767) |
| Big Integer | BIGINT | -263 to +263-1(92,23,37,20,36,85,47,75,808 to +92,23,37,20,36,85,47,75,807) |
| Tiny Integer | TINYINT | 0 to +28-1(0 to 255) |
| Bit | BIT | 0 or 1 |
| Numeric(Precision, Scale) | NUMERIC(Precision, Scale) | -1038+1 to +1038-1(-1.e+38 to +1.e+38) |
| Decimal or Dec | DECIMAL(Precision, Scale) or DEC(Precision, Scale) | -1038+1 to +1038-1(-1.e+38 to +1.e+38) |
| Float | FLOAT(Precision) | -1.79E+308 to +1.79E+308 |
| Real | REAL | -3.40E + 38 to +3.40E + 38 |

**Character Data Types**

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Syntax** | **Range** |
| Character String | CHAR | Maximum 8,000 characters |
| Character String | VARCHAR | Maximum 8,000 characters |
| Unicode Character String | NCHAR | Maximum 4,000 characters |
| Unicode Character String | NVARCHAR | Maximum 4,000 characters |
| Character String | TEXT | Maximum 231 = 2,147,483,647 characters |
| Unicode Character String | NTEXT | Maximum 230 = 1,073,741,823 characters |

**Binary Data Types**

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Syntax** | **Range** |
| Binary Value | BINARY | Maximum 8,000 Bytes |
| Binary Value | VARBINARY | Maximum 8,000 Bytes |
| Image | IMAGE | Maximum 231 = 2,147,483,647 Bytes |

**Date and Time Data Types**

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Syntax** | **Example** |
| date | DATE | For Example, Feb 27, 2015(Format : YYYY-MM-DD) |
| time | TIME | For Example, 2:15 A.M.(format : ) |
| datetime | DATETIME | Range : January 1, 1753 to December 31, 9999(format : YYYY-MM-DD HH:MI:SS) |
| smalldatetime | SMALLDATETIME | Range : January 1, 1900 to June 6, 2079(format : YYYY-MM-DD HH:MI:SS) |
| timestamp | TIMESTAMP | format : YYYY-MM-DD HH:MI:SS (store year, month, day, hour, minute and second) |

### SQL - Arithmetic Operators

SQL के Statements में कई बार कुछ keywords और character का Operators के रूप में इस्तेमाल किया जाता है |

Arithmetic Opertaors का इस्तेमाल numeric operands पर arithmetical operation करने के लिए किया जाता है |

|  |  |  |
| --- | --- | --- |
| **Operator** | **Meaning** | **Example** |
| + | Addition | 10 + 4 = 14 |
| - | Subtraction | 10 - 4 = 6 |
| \* | Multiplication | 10 \* 4 = 40 |
| / | Division | 10 / 4 = 2.5 |
| % | Modulus | 10 % 4 = 2 |

Table Name : Appliances

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+----------+-----------------+-------+------+

**Example For '+' Operator in SQL**

Example पर सभी columns select(\*) करके और एक ''Cost+GST' नाम का column कुछ समय के लिए add किया गया है | इस column से Cost और GST को add किया जायेगा |

Source Code :

SELECT \*, (Cost+GST) FROM Appliances;

Output :

+----+----------+-----------------+-------+------+------------+

| id | Dealer | Appliances | Cost | GST | (Cost+GST) |

+----+----------+-----------------+-------+------+------------+

| 1 | Balaji | T.V. | 19000 | 3420 | 22420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 | 19200 |

| 3 | Electra | Refrigerator | 10000 | 2800 | 12800 |

| 4 | K.K. | Cooler | 18000 | 5040 | 23040 |

| 5 | Ganesh | A.C. | 35000 | 9800 | 44800 |

| 6 | Rajaram | Iron | 1500 | 420 | 1920 |

| 7 | Ravi | Laptop | 30000 | 8400 | 38400 |

+----+----------+-----------------+-------+------+------------+

**Example For '-' Operator in SQL**

Example पर सभी columns select(\*) करके और एक ''Cost-GST' नाम का column कुछ समय के लिए add किया गया है | इस column से Cost में से GST को subtract किया जायेगा |

Source Code :

SELECT \*, (Cost-GST) FROM Appliances;

Output :

+----+----------+-----------------+-------+------+------------+

| id | Dealer | Appliances | Cost | GST | (Cost-GST) |

+----+----------+-----------------+-------+------+------------+

| 1 | Balaji | T.V. | 19000 | 3420 | 15580 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 | 10800 |

| 3 | Electra | Refrigerator | 10000 | 2800 | 7200 |

| 4 | K.K. | Cooler | 18000 | 5040 | 12960 |

| 5 | Ganesh | A.C. | 35000 | 9800 | 25200 |

| 6 | Rajaram | Iron | 1500 | 420 | 1080 |

| 7 | Ravi | Laptop | 30000 | 8400 | 21600 |

+----+----------+-----------------+-------+------+------------+

**Example For '\*' Operator in SQL**

Example पर सभी columns select(\*) करके और एक ''Cost\*GST' नाम का column कुछ समय के लिए add किया गया है | इस column से Cost और GST को multiply किया जायेगा |

Source Code :

SELECT \*, (Cost\*GST) FROM Appliances;

Output :

+----+----------+-----------------+-------+------+------------+

| id | Dealer | Appliances | Cost | GST | (Cost\*GST) |

+----+----------+-----------------+-------+------+------------+

| 1 | Balaji | T.V. | 19000 | 3420 | 64980000 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 | 63000000 |

| 3 | Electra | Refrigerator | 10000 | 2800 | 28000000 |

| 4 | K.K. | Cooler | 18000 | 5040 | 90720000 |

| 5 | Ganesh | A.C. | 35000 | 9800 | 343000000 |

| 6 | Rajaram | Iron | 1500 | 420 | 630000 |

| 7 | Ravi | Laptop | 30000 | 8400 | 252000000 |

+----+----------+-----------------+-------+------+------------+

**Example For '/' Operator in SQL**

Example पर सभी columns select(\*) करके और एक ''Cost\*GST' नाम का column कुछ समय के लिए add किया गया है | इस column से Cost में से GST को divide किया जायेगा |

Source Code :

SELECT \*, (Cost\*GST) FROM Appliances;

Output :

+----+----------+-----------------+-------+------+------------+

| id | Dealer | Appliances | Cost | GST | (Cost/GST) |

+----+----------+-----------------+-------+------+------------+

| 1 | Balaji | T.V. | 19000 | 3420 | 5.5556 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 | 3.5714 |

| 3 | Electra | Refrigerator | 10000 | 2800 | 3.5714 |

| 4 | K.K. | Cooler | 18000 | 5040 | 3.5714 |

| 5 | Ganesh | A.C. | 35000 | 9800 | 3.5714 |

| 6 | Rajaram | Iron | 1500 | 420 | 3.5714 |

| 7 | Ravi | Laptop | 30000 | 8400 | 3.5714 |

+----+----------+-----------------+-------+------+------------+

**Example For '%' Operator in SQL**

Example पर सभी columns select(\*) को select करके '7%3' मतलब '1' इस id return किया गया है |

Source Code :

SELECT \* FROM Appliances WHERE id = 7%3;

Output :

+----+--------+------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+--------+------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

+----+--------+------------+-------+------+

**Try Also :**

SELECT \*, (Cost+GST) FROM Appliances WHERE (Cost+GST)<19000;

SELECT \*, (Cost-GST) FROM Appliances WHERE (Cost-GST)<19000;

अगर कौनसे appliance पर कितनी percentages; GST है वो देखने के लिए,

SELECT \*, (GST\*100/Cost) FROM Appliances WHERE (Cost-GST)<19000;

**Note :** ऊपर दिए गए Examples सिर्फ User को Arithmetic operators का इस्तेमाल कैसे किया जा सकता है इसलिए दिए गए है |

### SQL - Comparison Operators

SQL के Statements में कई बार कुछ keywords और character का Operators के रूप में इस्तेमाल किया जाता है |

Comparison Operator से दो expression को compare करता है |

|  |  |  |
| --- | --- | --- |
| **Operator** | **Meaning** | **Example** |
| < | Less than | 10 < 4 return False |
| > | Greater than | 10 > 4 return True |
| <= | Less than or Equal to | 10 <= 4 return False |
| >= | Greater than or Equal to | 10 >= 4 return True |
| = | Equal to | 10 = 4 return False |
| != | Not Equal to | 10 != 4 return True |
| <> | Not Equal to | 10 <> 4 return True |

Table Name : Appliances

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+----------+-----------------+-------+------+

**Example For '<' Operator in SQL**

Example पर सभी columns(\*) को select करके Cost इस column में जहा पर 19000 से कम हो वो सभी rows return किये गए है |

Source Code :

SELECT \* FROM Appliances WHERE Cost<19000;

Output :

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 6 | Rajaram | Iron | 1500 | 420 |

+----+----------+-----------------+-------+------+

**Example For '>' Operator in SQL**

Example पर सभी columns(\*) को select करके Cost इस column में जहा पर 19000 से ज्यादा हो वो सभी rows return किये गए है |

Source Code :

SELECT \* FROM Appliances WHERE Cost>19000;

Output :

+----+--------+------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+--------+------------+-------+------+

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+--------+------------+-------+------+

**Example For '<=' Operator in SQL**

Example पर सभी columns(\*) को select करके Cost इस column में जहा पर 10000 या 10000 से कम हो वो सभी rows return किये गए है |

Source Code :

SELECT \* FROM Appliances WHERE Cost<=10000;

Output :

+----+---------+--------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+---------+--------------+-------+------+

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 6 | Rajaram | Iron | 1500 | 420 |

+----+---------+--------------+-------+------+

**Example For '>=' Operator in SQL**

Example पर सभी columns(\*) को select करके Cost इस column में जहा पर 10000 या 10000 से ज्यादा हो वो सभी rows return किये गए है |

Source Code :

SELECT \* FROM Appliances WHERE Cost>=10000;

Output :

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+----------+-----------------+-------+------+

**Example For '=' Operator in SQL**

Example पर सभी columns(\*) को select करके Cost इस column में जहा पर 10000 के बराबर हो वो सभी rows return किये गए है |

Source Code :

SELECT \* FROM Appliances WHERE Cost=10000;

Output :

+----+---------+--------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+---------+--------------+-------+------+

| 3 | Electra | Refrigerator | 10000 | 2800 |

+----+---------+--------------+-------+------+

**Example For '!=' Operator in SQL**

Example पर सभी columns(\*) को select करके Cost इस column में जहा पर 10000 के बराबर ना हो वो सभी rows return किये गए है |

!= और <> ये दोनों operators एक जैसे है |

Source Code :

SELECT \* FROM Appliances WHERE Cost!=10000;

Output :

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+----------+-----------------+-------+------+

**Example For '<>' Operator in SQL**

Example पर सभी columns(\*) को select करके Cost इस column में जहा पर 10000 से कम हो या ज्यादा हो लेकिन बराबर ना हो वो सभी rows return किये गए है |

<> और != ये दोनों operators एक जैसे है |

Source Code :

SELECT \* FROM Appliances WHERE Cost<>10000;

Output :

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+----------+-----------------+-------+------+

### SQL - Logical Operators

SQL के Statements में कई बार कुछ keywords और character का Operators के रूप में इस्तेमाल किया जाता है |

Comparison Operator से दो expression को compare करता है |

|  |  |
| --- | --- |
| **Operator** | **Description** |
| ALL | सभी subquery values जब condition से मिलती है तो True return होता है | |
| AND | एक से ज्यादा conditions दी जाती है | हर condition को AND से seperate किया जाता है अगर हर condition True होती है तो True return होता है | |
| ANY | कोई भी subquery value जब condition से मिलती है तो True return होता है | |
| BETWEEN | AND Operator के साथ दिए गए दो values के बीच के records को select करता है | |
| EXISTS | Subquery में record मौजूद है या नहीं ये check किया जाता है | |
| IN | column के अनुसार दिए गए values को चुनता है | |
| LIKE | column के अनुसार दिए गए pattern से match करता है | |
| NOT | जब condition दी जाती है तब उसके विरुद्ध records display किये जाते है | |
| OR | एक से ज्यादा conditions दी जाती है | हर condition को OR से seperate किया जाता है अगर एक भी condition True होती है तो True return होता है | |
| IS NULL | column के अनुसार NULL value को ढूंढा जाता है | |

इस दिए गए दो tables की मदद से logical operation करेंगे |

Table Name : Appliances

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+----------+-----------------+-------+------+

**Example For 'ALL' Operator in SQL**

Example पर subqueries के मदद से Appliances table के सभी columns select(\*) करके सभी Cost '15000' से ज्यादा होने वाले rows return किये गए है |

Source Code :

SELECT \* FROM Appliances

WHERE Cost > ALL(SELECT cost FROM Appliances

WHERE Cost=15000);

Output :

+----+--------+------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+--------+------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+--------+------------+-------+------+

**Example For 'AND' Operator in SQL**

Example पर सभी columns select(\*) करके Cost 30000 से कम और Cost 15000 से ज्यादा होने वाले सभी records को return किया गया है |

Source Code :

SELECT \* FROM Appliances WHERE Cost<30000 AND Cost>15000;

Output :

+----+--------+------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+--------+------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 4 | K.K. | Cooler | 18000 | 5040 |

+----+--------+------------+-------+------+

**Example For 'ANY' Operator in SQL**

Example पर subqueries के मदद से Appliances table के सभी columns select(\*) करके कोई भी Cost '15000' से बराबर होने वाले rows return किये गए है |

Source Code :

SELECT \* FROM Appliances

WHERE Cost = ANY(SELECT cost FROM Appliances

WHERE Cost=15000);

Output :

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

**Example For 'BETWEEN' Operator in SQL**

Example पर सभी columns select करके Cost 15000 और Cost 30000 के बीच के सभी records return किये गए है |

Source Code :

SELECT \* FROM Appliances

WHERE Cost BETWEEN 15000 AND 30000;

Output :

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+----------+-----------------+-------+------+

**Example For 'EXISTS' Operator in SQL**

**Sample Tables**

Table Name : Appliances

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+----------+-----------------+-------+------+

Table Name : Employee

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 7 | Raj Raman | 25000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | 25000 |

+--------+-----------------+------------+

Example पर Appliances के सभी columns select(\*) करके जो employee में id है वो सभी Appliances में exist id के अनुसार records को return किया गया है |

Source Code :

SELECT \* FROM Appliances WHERE EXISTS(SELECT \* FROM Employee WHERE emp\_id = Appliances.id);

Output :

+----+---------+--------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+---------+--------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+---------+--------------+-------+------+

**Example For 'IN' Operator in SQL**

Example पर सभी columns को select(\*) करके 2, 3 और 7 इस id के records को return किया गया है |

Source Code :

SELECT \* FROM Appliances WHERE id IN(2,3,7);

Output :

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+----------+-----------------+-------+------+

**Example For 'LIKE' Operator in SQL**

Example पर सभी columns select करके r से start होनेवाले सभी emp\_name records return किया गया है |

Source Code :

SELECT \* FROM Employee WHERE emp\_name LIKE 'r%';

Output :

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 3 | Rupal Patil | 30000 |

| 7 | Raj Raman | 25000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | 25000 |

+--------+-----------------+------------+

**Example For 'NOT' Operator in SQL**

Example पर सभी columns select(\*) करके जो Cost 15000 के बराबर नहीं है वो सभी rows return किये गए है |

Source Code :

SELECT \* FROM Appliances WHERE NOT Cost=15000;

Output :

+----+---------+--------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+---------+--------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+---------+--------------+-------+------+

**Example For 'OR' Operator in SQL**

Example पर सभी columns select करके Cost 15000 से कम या Cost 50000 ज्यादा होने वाले सभी records को return किया गया है |

Source Code :

SELECT \* FROM Appliances WHERE Cost<15000 OR Cost>50000;

Output :

+----+---------+--------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+---------+--------------+-------+------+

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 6 | Rajaram | Iron | 1500 | 420 |

+----+---------+--------------+-------+------+

**Example For 'IS NULL' Operator in SQL**

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | NULL |

| 7 | Raj Raman | 25000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | NULL |

+--------+-----------------+------------+

Example पर सभी columns को select(\*) करके NULL emp\_salary वाले सभी records को return किया गया है |

Source Code :

SELECT \* FROM Employee WHERE emp\_salary IS NULL;

Output :

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 4 | Shankar Mane | NULL |

| 15 | Ramesh Mohane | NULL |

+--------+---------------+------------+

### SQL - SELECT Statement

SQL में ज्यादातर SELECT Statement का इस्तेमाल किया जाता है |

SELECT Statement का इस्तेमाल database के table से data पाने(retrieve) के लिए किया जाता है |

जो returned हुआ data होता है वो table जैसा होता है और उस returned हुए data को result-set कहा जाता है |

**Syntax for SELECT Statement in SQL**

**For Single or Multiple Columns in SQL**

SELECT column1,column2,..,columnN

FROM table\_name;

**For All Columns using \* in SQL**

SELECT \*

FROM table\_name;

**Sample Table**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+----------+-----------------+-------+------+

**Example For SELECT Statement with Single Column in SQL**

Example पर Appliances इस table का सिर्फ Dealer ये column select किया गया है |

Source Code :

SELECT Dealer FROM Appliances;

Output :

+----------+

| Dealer |

+----------+

| Balaji |

| Ravikant |

| Electra |

| K.K. |

| Ganesh |

| Rajaram |

| Ravi |

+----------+

**Example For SELECT Statement with Multiple Columns in SQL**

Example पर Appliances इस table के Dealer, Appliances और GST ये columns select किये गए है |

Source Code :

SELECT Dealer,Appliances,GST FROM Appliances;

Output :

+----------+-----------------+------+

| Dealer | Appliances | GST |

+----------+-----------------+------+

| Balaji | T.V. | 3420 |

| Ravikant | Washing Machine | 4200 |

| Electra | Refrigerator | 2800 |

| K.K. | Cooler | 5040 |

| Ganesh | A.C. | 9800 |

| Rajaram | Iron | 420 |

| Ravi | Laptop | 8400 |

+----------+-----------------+------+

**Example For SELECT Statement with All(\*) Columns in SQL**

Example पर Appliances इस table के सभी columns select किये गए है |

Source Code :

SELECT \* FROM Appliances;

Output :

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+----------+-----------------+-------+------+

### SQL - SELECT DISTINCT Statement

SELECT\_DISTINCT Statement का इस्तेमाल सिर्फ Unique values; return करने के लिए किया जाता है |

कई बार Database के table के columns में duplicate values होती है | इन duplicate values को remove करने के लिए SELECT के साथ DISTINCT keyword का इस्तेमाल किया जाता है |

**Syntax for SELECT Statement in SQL**

**For Single or Multiple Columns in SQL**

SELECT DISTINCT column1,column2,..,columnN

FROM table\_name;

**For Select all Columns using \* in SQL**

SELECT DISTINCT \*

FROM table\_name;

**Sample Table**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

**Example For SELECT\_DISTINCT Statement in SQL**

Example पर Appliances table का 'Appliances' ये column select करके उसमे से unique values को return किया गया है |

Source Code :

SELECT DISTINCT Appliances FROM Appliances;

Output :

+-----------------+

| Appliances |

+-----------------+

| T.V. |

| Washing Machine |

| Refrigerator |

| Cooler |

| A.C. |

| Iron |

| Laptop |

+-----------------+

### SQL - WHERE Clause

WHERE Clause का इस्तेमाल ज्यादातर SELECT, UPDATE और DELETE Statement के साथ किया जाता है |

WHERE Clause ये दिए गए condition के मुताबिक data को retrieve, delete या update किया जाता है |

WHERE Clause की condition अगर True होती है तो result-set return होता है |

**Syntax for WHERE Clause in SQL**

**For Single or Multiple Columns in SQL**

SELECT column1, column2,.., column3

FROM table\_name

WHERE condition;

**For All Columns in SQL**

SELECT \*

FROM table\_name

WHERE condition;

**Sample Table**

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | NULL |

| 7 | Raj Raman | 25000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | NULL |

+--------+-----------------+------------+

**Example for WHERE Clause using SELECT Statement in SQL**

Example पर सभी columns को select करके Where clause की मदद से 7 इस emp\_id से कम ids को return किया गया है |

Source Code :

SELECT \* FROM Employee WHERE emp\_id<7;

Output :

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | NULL |

+--------+---------------+------------+

**Example for WHERE Clause using UPDATE Statement in SQL**

Example पर जहा पर id 7 है वहा के emp\_name पर Shreya Bhalekar और emp\_salary पर 40000 set करके सभी records को return किया गया है |

Source Code :

UPDATE Employee

SET emp\_name='Shreya Bhalekar', emp\_salary='40000'

WHERE emp\_id=7;

SELECT \* FROM Employee;

Output :

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | NULL |

| 7 | Shreya Bhalekar | 40000 | <---------- Updated record

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | NULL |

+--------+-----------------+------------+

**Example for WHERE Clause using DELETE Statement in SQL**

Example पर जहा पर id 7 है वो id delete किया गया है और Employee इस table के result-set को return किया गया है |

Source Code :

DELETE FROM Employee

WHERE emp\_id=7;

SELECT \* FROM Employee;

Output :

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | NULL |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | NULL |

+--------+-----------------+------------+

**Note :** WHERE Clause के condition पर <, >, <=, >=, =, !=, <>, BETWEEN, IN, LIKE इन operators का इस्तेमाल किया जा सकता है |

### SQL - INSERT INTO Statement

INSERT INTO Statement का इस्तेमाल Table पर नया record insert करने के लिए किया जाता है |

**Syntax for INSERT INTO Statement in SQL**

**With Columns Name**

INSERT INTO table\_name(column1,column2,..,columnN)

VALUES (value1, value2,..,valueN);

**Without Column Name**

यहाँ पर दिए गए table के columns की सभी values दी जाती है |

INSERT INTO table\_name

VALUES (value1, value2,..,valueN);

**Sample Table**

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | 30000 |

+--------+-----------------+------------+

**Example for INSERT INTO Statement With Multiple Column Names in SQL**

Example पर Employee इस table पर 'Shreya Bhalekar' ये emp\_name और '40000' ये emp\_salary insert की गयी है और पूरे result-set को return किया गया है |

Source Code :

INSERT INTO Employee(emp\_name, emp\_salary)

VALUES ('Shreya Bhalekar', '40000');

SELECT \* FROM Employee;

Output :

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | 30000 |

| 16 | Shreya Bhalekar | 40000 |

+--------+-----------------+------------+

**Example for INSERT INTO Statement With Single Column Names in SQL**

**Sample Table**

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | 30000 |

| 16 | Shreya Bhalekar | 40000 |

+--------+-----------------+------------+

Example पर सिर्फ emp\_name 'Ravi Rajan' ये insert किया गया है | लेकिन emp\_salary ये column की value दी नहीं गयी है | Column create करते वक्त जब default value दी जाती है वो default value वहा पर दी जाती है | For Example, NULL, NOT NULL, AUTO\_INCREMENT

Source Code :

INSERT INTO Employee(emp\_name)

VALUES ('Ravi Rajan');

SELECT \* FROM Employee;

Output :

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | 30000 |

| 22 | Shreya Bhalekar | 40000 |

| 23 | Ravi Rajan | NULL |

+--------+-----------------+------------+

**Example For INSERT INTO Statement Without Columns in SQL**

**Sample Table**

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | 30000 |

| 22 | Shreya Bhalekar | 40000 |

| 23 | Ravi Rajan | NULL |

+--------+-----------------+------------+

Example पर Employee इस table पर बिना columns दिए record को insert किया गया है | अगर columns दिए नहीं जाते है तो सभी columns की values दी जाती है |

INSERT INTO Employee

VALUES(20, 'Rahul Kapoor', '27000');

SELECT \* FROM Employee;

Output :

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 27000 |

| 22 | Shreya Bhalekar | 40000 |

| 23 | Ravi Rajan | NULL |

+--------+-----------------+------------+

**Sample Tables**

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 10000 |

| 20 | Rahul Kapoor | 27000 |

| 21 | Shankar Mane | 20000 |

| 22 | Maria Shaikh | 30000 |

| 23 | Vikram Kank | 40000 |

| 24 | Vaibhav Vichare | 10000 |

| 25 | Shweta Pandit | 25000 |

+--------+-----------------+------------+

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

**'INSERT INTO Statement' With 'SELECT Statement' in SQL**

Example पर Appliances table के Appliances इस column को select करके Employee table के emp\_name इस column पर copy किये गए है |

Source Code :

INSERT INTO Employee(emp\_name)

SELECT Appliances

FROM Appliances;

Output :

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 10000 |

| 20 | Rahul Kapoor | 27000 |

| 21 | Shankar Mane | 20000 |

| 22 | Maria Shaikh | 30000 |

| 23 | Vikram Kank | 40000 |

| 24 | Vaibhav Vichare | 10000 |

| 25 | Shweta Pandit | 25000 |

| 26 | T.V. | 0 |

| 27 | Washing Machine | 0 |

| 28 | Refrigerator | 0 |

| 29 | Cooler | 0 |

| 30 | A.C. | 0 |

| 31 | Iron | 0 |

| 32 | Laptop | 0 |

| 33 | Washing Machine | 0 |

+--------+-----------------+------------+

### SQL - INSERT INTO Statement Multiple Records

INSERT INTO Statement का इस्तेमाल Table पर नया record insert करने के लिए किया जाता है |

**Syntax for INSERT INTO Statement in SQL**

**Insert Multiple records with Columns Name**

INSERT INTO table\_name(column1,column2,..,columnN)

VALUES (value1, value2,..,valueN),

(value1, value2,..,valueN);

**Without Column Name**

यहाँ पर दिए गए table के columns की सभी values दी जाती है |

**Insert Multiple records without Columns Name**

INSERT INTO table\_name

VALUES (value1, value2,..,valueN),

(value1, value2,..,valueN);

**Sample Table**

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 27000 |

| 22 | Shreya Bhalekar | 40000 |

| 23 | Ravi Rajan | NULL |

+--------+-----------------+------------+

**Example for INSERT INTO Statement With Multiple Column Names and Multiple Records in SQL**

Example पर Employee इस table पर multiple records insert किये गए है |

Source Code :

INSERT INTO Employee(emp\_name, emp\_salary)

VALUES ('Umesh Patel', '100000'),

('Iran Shaikh', '13000');

SELECT \* FROM Employee;

Output :

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 27000 |

| 22 | Shreya Bhalekar | 40000 |

| 23 | Ravi Rajan | NULL |

| 24 | Umesh Patel | 100000 |

| 25 | Iran Shaikh | 13000 |

+--------+-----------------+------------+

**Example for INSERT INTO Statement Without Column Names with Multiple Records in SQL**

Example पर Employee table में बिना columns दिए multiple records को insert किया गया है |

Source Code :

INSERT INTO Employee

VALUES (30,'Mahesh Marane', '26000'),

(31,'Lavanya Kumari', '37000');

SELECT \* FROM Employee;

Output :

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 27000 |

| 22 | Shreya Bhalekar | 40000 |

| 23 | Ravi Rajan | NULL |

| 24 | Umesh Patel | 100000 |

| 25 | Iran Shaikh | 13000 |

| 30 | Mahesh Marane | 26000 |

| 31 | Lavanya Kumari | 37000 |

+--------+-----------------+------------+

### SQL - UPDATE Statement

UPDATE Statement का इस्तेमाल मौजुदे records को update करने के लिए किया जाता है |

UPDATE Statement द्वारा एक से ज्यादा भी records को update किया जाता है |

**Syntax for UPDATE Statement in SQL**

**Without WHERE Clause(For All Rows)**

UPDATE table\_name

SET coloum1=value1,column2=value2,...,column3=value3;

**With WHERE Clause(For Specific Row)**

UPDATE table\_name

SET coloum1=value1,column2=value2,...,column3=value3

WHERE condition;

**Sample Table**

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 27000 |

+--------+-----------------+------------+

**Example for UPDATE Statement Without WHERE Clause in SQL**

Example पर WHERE Clause के बिना UPDATE Statement को दिया गया है | अगर WHERE Clause की condition दी नहीं जाती है तो SET पर दिए column की values सभी row पर Update होती है |

Source Code :

UPDATE Employee

SET emp\_salary=30000;

SELECT \* FROM Employee;

Output :

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 30000 |

| 12 | Rajesh Mujumdar | 30000 |

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 30000 |

+--------+-----------------+------------+

**Example for UPDATE Statement With WHERE Clause in SQL**

**Sample Table**

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 20000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 27000 |

+--------+-----------------+------------+

Example पर Employee table के 1 इस emp\_id के emp\_salary इस column के value को update किया गया है |

Source Code :

UPDATE Employee

SET emp\_salary=30000

WHERE emp\_id=1;

SELECT \* FROM Employee;

Output :

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 27000 |

+--------+-----------------+------------+

**Another Example for UPDATE Statement(Multiple Columns) With WHERE Clause in SQL**

**Sample Table**

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 12 | Rajesh Mujumdar | 26000 |

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 27000 |

+--------+-----------------+------------+

Exmple पर Employee इस table में जहा पर emp\_id 12 हो उस record के emp\_name और emp\_salary को update किया गया है |

Source Code :

UPDATE Employee

SET emp\_name='Raj Mehra', emp\_salary='10000'

WHERE emp\_id=12;

SELECT \* FROM Employee;

Output :

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 12 | Raj Mehra | 10000 |

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 27000 |

+--------+---------------+------------+

### SQL - DELETE Statement

DELETE Statement से table से मौजुदे record या पूरे table के records को delete/remove किया जाता है |

**Syntax for DELETE Statement in Python**

**For Specific Record**

DELETE FROM table\_name

WHERE condition;

**For All Records**

DELETE FROM table\_name;

or

DELETE \* FROM table\_name;

**Sample Table**

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 12 | Raj Mehra | 10000 |

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 27000 |

+--------+---------------+------------+

**Example for DELETE Statement with WHERE Clause in SQL**

Example में Employee इस table में जहा पर emp\_name 'Raj Mehra' है वो record table से delete किया गया है |

Source Code :

DELETE FROM Employee

WHERE emp\_name='Raj Mehra';

SELECT \* FROM Employee;

Output :

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 27000 |

+--------+---------------+------------+

**DELETE All Records From Table 'Employee'**

Source Code :

DELETE FROM Employee;

SELECT \* FROM Employee;

Output :

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

+--------+---------------+------------+

### SQL - ORDER BY Clause

ORDER BY Clause का इस्तेमाल result-set के records को ascending या descending में sort करने के लिये किया जाता है |

ORDER BY Clause जब दिया जाता है तब by default; records; ascending order में sort किया जाता है |

**Syntax for ORDER BY Clause in SQL**

SELECT column1,column2,..,columnN

FROM table\_name

ORDER BY column1,column2,..,columnN ASC/DESC;

**Samplle Table**

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 27000 |

+--------+---------------+------------+

**Example for ORDER BY Clause(Ascending Order) in SQL**

Example पर Employee इस table के emp\_name इस column के अनुसार by default; ascending order में sort किया गया है |

Source Code :

SELECT \* FROM Employee ORDER BY emp\_name;

OR

SELECT \* FROM Employee ORDER BY emp\_name ASC;

Output :

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 20 | Rahul Kapoor | 27000 |

| 15 | Ramesh Mohane | 30000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 1 | Shweta Pandit | 30000 |

+--------+---------------+------------+

**Example for ORDER BY Clause(Descending Order) in SQL**

Example पर Employee इस table के emp\_name इस column के अनुसार descending order में sort किया गया है |

Source Code :

SELECT \* FROM Employee ORDER BY emp\_name DESC;

Output :

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 1 | Shweta Pandit | 30000 |

| 4 | Shankar Mane | 15000 |

| 3 | Rupal Patil | 30000 |

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 27000 |

+--------+---------------+------------+

### SQL - SELECT LIMIT Statement

SELECT LIMIT Statement का इस्तेमाल जितने number दिया जाता है उतने records return किये जाते है |

**Syntax for SELECT LIMIT Statement in SQL**

**Without WHERE Clause**

SELECT \*

FROM table\_name

LIMIT number;

**With WHERE Clause**

SELECT \*

FROM table\_name

WHERE condition

LIMIT number;

**Sample Table**

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 27000 |

+--------+---------------+------------+

**Example for SELECT LIMIT Statement in SQL**

Example पर Employee इस table से पहले तीन records को return किया गया है |

Source Code :

SELECT \*

FROM Employee

LIMIT 3;

Output :

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

+--------+---------------+------------+

**Example for SELECT LIMIT Statement With WHERE Clause in SQL**

Example पर Employee इस table में emp\_id 8 से कम 2 records को return किया गया है |

Source Code :

SELECT \*

FROM Employee

WHERE emp\_id<8 LIMIT 2;

Output :

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 30000 |

+--------+---------------+------------+

### SQL - AND, OR, NOT Operator

AND, OR और NOT ये Logical Operators है |

**Sample Table**

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 27000 |

+--------+---------------+------------+

**AND Logical Operator in SQL**

AND Operator के साथ एक से ज्यादा conditions दी जाती है | हर condition को AND से seperate किया जाता है | अगर हर condition True होती है तो True return करता है |

**Syntax for AND Operator in SQL**

SELECT \*

FROM table\_name

WHERE condition1 AND condition2 AND conditionN;

**Example for AND Operator in SQL**

Example पर Employee इस table में emp\_salary 15000 से ज्यादा और emp\_salary 30000 से कम होने वाले records को return किया गया है |

Source Code :

SELECT \*

FROM Employee

WHERE emp\_salary>15000 AND emp\_salary<30000;

Output :

+--------+--------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+--------------+------------+

| 20 | Rahul Kapoor | 27000 |

+--------+--------------+------------+

**OR Logical Operator in SQL**

OR Operator के साथ एक से ज्यादा conditions दी जाती है | हर condition को OR से seperate किया जाता है | अगर एक भी condition True होती है तो True return करता है |

**Syntax for OR Operator in SQL**

SELECT \*

FROM table\_name

WHERE condition1 OR condition2 OR conditionN;

**Example for OR Operator in SQL**

Example पर Employee इस table के records में जहा पर emp\_salary 15000 या 30000 हो वो सभी records return किये गए है |

Source Code :

SELECT \*

FROM Employee

WHERE emp\_salary=15000 OR emp\_salary=30000;

Output :

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 30000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 30000 |

+--------+---------------+------------+

**NOT Logical Operator in SQL**

जब condition दी जाती है तब उसके विरुद्ध records condition के मुताबिक return किये जाते है |

**Syntax for NOT Operator in SQL**

SELECT \*

FROM table\_name

WHERE NOT condition;

**Example for NOT Operator in SQL**

Example पर Employee इस table से जो emp\_id 10 से कम records को return नहीं किया गया है |

Source Code :

SELECT \*

FROM Employee

WHERE NOT emp\_id<10;

Output :

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 15 | Ramesh Mohane | 30000 |

| 20 | Rahul Kapoor | 27000 |

+--------+---------------+------------+

### SQL - LIKE Operator

LIKE Operator का इस्तेमाल WHERE Clause के साथ किया जाता है |

LIKE Operator का इस्तेमाल दिए गए pattern को WHERE clause में दिए गए column में search करता है |

**Syntax for LIKE Operator**

SELECT \*

FROM table\_name

WHERE column\_name LIKE pattern;

**Wildcard Characters in SQL**

* **% :** String से एक या एक से ज्यादा characters से match करता है |
* **\_ :** एक ही character से match करता है |

**Sample Table**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

**Example for LIKE Operator(Ending With letter 'r') in SQL**

Example पर Appliances table के Appliances इस column में 'r' इस character से end होने वाले appliances को return किया गया है |

Source Code :

SELECT \*

FROM Appliances

WHERE Appliances LIKE '%r';

Output :

+----+---------+--------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+---------+--------------+-------+------+

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

+----+---------+--------------+-------+------+

**Example for LIKE Operator(Starting With letter 'r') in SQL**

Example पर Appliances table के Dealer इस column में 'r' इस character से start होने वाले dealers को return किया गया है |

Source Code :

SELECT \*

FROM Appliances

WHERE Appliances LIKE 'r%';

Output :

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

**Example for LIKE Operator(Containing character 'vi') in SQL**

Example पर Appliances table के Dealer इस column में 'vi' ये characters सम्मिलित होने वाले Dealers को return किया गया है |

Source Code :

SELECT \*

FROM Appliances

WHERE Appliances LIKE '%vi%';

Output :

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

**Example for LIKE Operator(Not Containing character 'vi') in SQL**

Example पर Appliances table के Dealer इस column में 'vi' ये characters सम्मिलित नहीं होने वाले Dealers को return किया गया है |

Source Code :

SELECT \*

FROM Appliances

WHERE Appliances NOT LIKE '%vi%';

Output :

+----+---------+--------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+---------+--------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

+----+---------+--------------+-------+------+

**Another Example for LIKE Operator**

Source Code :

SELECT \* FROM Appliances WHERE Dealer LIKE 'Rav\_';

Output :

+----+--------+------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+--------+------------+-------+------+

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+--------+------------+-------+------+

**Try Also**

SELECT \* FROM Appliances WHERE Dealer LIKE 'Ra\_\_';

SELECT \* FROM Appliances WHERE Dealer LIKE '\_av\_';

SELECT \* FROM Appliances WHERE Dealer LIKE '\_avikant';

### SQL - BETWEEN Operator

BETWEEN Operator में AND Operator क साथ values की range दी जाती है | ये values कोई भी हो सकती है |

BETWEEN Operator; AND Operator के साथ दिए गए दो values के बीच के records को select करता है |

**Syntax for BETWEEN Operator in SQL**

SELECT \*

FROM table\_name

WHERE column\_name BETWEEN value1 AND value2;

**Sample Table**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

**Example for BETWEEN Operator in SQL**

Example पर Appliances इस table में id 3 और id 7 के साथ उनके बीच के records को return किया गया है |

Source Code :

SELECT \*

FROM Appliances

WHERE id BETWEEN 3 AND 7;

Output :

+----+---------+--------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+---------+--------------+-------+------+

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+---------+--------------+-------+------+

**Example for NOT BETWEEN Operator in SQL**

Example पर Appliances इस table में id 3 और id 7 के साथ उनके बीच के records को return नहीं किया गया है |

Source Code :

SELECT \*

FROM Appliances

WHERE id NOT BETWEEN 3 AND 7;

Output :

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

**Another Example for BETWEEN Operator in SQL**

Example पर Appliances इस table में id 3 और id 7 के साथ उनके बीच के records को return नहीं किया गया है |

Source Code :

SELECT \*

FROM Appliances

WHERE id BETWEEN 3 AND 7;

Output :

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

### SQL - IN Operator

IN Operator का इस्तेमाल दिए गए column के एक या एक से ज्यादा values के अनुसार records को return करता है |

**Syntax for IN Operator in SQL**

SELECT \*

FROM table\_name

WHERE column\_name IN(value1,value2,..,valueN);

**Sample Table**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

**Example for IN Operator in SQL**

Example पर Appliances इस table के Dealer इस column के 'Balaji' और 'Electra' इस values के records को return किया गया है |

Source Code :

SELECT \*

FROM Appliances

WHERE Dealer IN('Balaji','Electra');

Output :

+----+---------+--------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+---------+--------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

+----+---------+--------------+-------+------+

### SQL - IS NULL Operator

IS NULL Operator column के अनुसार NULL value को ढूंढने के लिए इस्तेमाल किया जाता है |

अगर column पर कोई default value set नहीं की जाती है और वो empty होता है तो वहा पर NULL value set हो जाती है |

**Syntax for 'IS NULL' Operator in SQL**

SELECT \* FROM Employee

WHERE column\_name IS NULL;

SELECT \* FROM Employee

WHERE column\_name IS NOT NULL;

**Sample Table**

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | NULL |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | NULL |

| 20 | Rahul Kapoor | 27000 |

+--------+---------------+------------+

**Example for 'IS NULL' Operator in SQL**

Exampe पर Employee इस table के emp\_salary इस column में जहा पर NULL value होगी उस records को return किया गया है |

Source Code :

SELECT \*

FROM Employee

WHERE emp\_salary IS NULL;

Output :

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 3 | Rupal Patil | NULL |

| 15 | Ramesh Mohane | NULL |

+--------+---------------+------------+

**Example for 'IS NOT NULL' Operator in SQL**

Exampe पर Employee इस table के emp\_salary इस column में जहा पर NULL value नहीं होगी उस records को return किया गया है |

Source Code :

SELECT \*

FROM Employee

WHERE emp\_salary IS NOT NULL;

Output :

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 1 | Shweta Pandit | 30000 |

| 4 | Shankar Mane | 15000 |

| 20 | Rahul Kapoor | 27000 |

+--------+---------------+------------+

### SQL - Aliases

SQL Aliases में table या उसके column के लिए कुछ वक्त के लिए अलग से alias दिया जाता है |

Alias ये SQL queries की length को घटा देता है |

SQL Alias; query जब तक होती है तबतक ही मौजूद रहता है |

**Syntax for Alias**

**for Columns**

SELECT column\_name(s) AS alias\_name

FROM table\_name;

**for Table**

SELECT column\_name(s)

FROM table\_name AS alias\_name;

**Sample Table**

Table Name : Appliances

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

Table Name : Employee

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | NULL |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | NULL |

| 20 | Rahul Kapoor | 27000 |

+--------+---------------+------------+

**Example for Alias(Column) in SQL**

Example पर Appliances इस table के Cost और GST का addition करके उनको total ये alias name दिया गया है |

Source Code :

SELECT Appliances,Cost+GST AS Total

FROM Appliances;

Output :

+-----------------+-------+

| Appliances | Total |

+-----------------+-------+

| T.V. | 22420 |

| Washing Machine | 19200 |

| Refrigerator | 12800 |

| Cooler | 23040 |

| A.C. | 44800 |

| Iron | 1920 |

| Laptop | 38400 |

| Washing Machine | 19200 |

+-----------------+-------+

**Another Example for Alias(Column) in SQL**

Example पर Appliances इस table के सभी columns concat करके उसके 'app\_all' ये alias name दिया गया है |

Source Code :

SELECT CONCAT(id,', ',Appliances,', ',Cost,', ',GST) AS app\_all

FROM Appliances;

Output :

+---------------------------------+

| app\_all |

+---------------------------------+

| 1, T.V., 19000, 3420 |

| 2, Washing Machine, 15000, 4200 |

| 3, Refrigerator, 10000, 2800 |

| 4, Cooler, 18000, 5040 |

| 5, A.C., 35000, 9800 |

| 6, Iron, 1500, 420 |

| 7, Laptop, 30000, 8400 |

| 8, Washing Machine, 15000, 4200 |

+---------------------------------+

**Example for Alias(Table) in SQL**

Example पर Appliances इस table को 'a' ये alias name दिया गया है |

Source Code :

SELECT a.Dealer, a.Appliances

FROM Appliances AS a;

Output :

+----------+-----------------+

| Dealer | Appliances |

+----------+-----------------+

| Balaji | T.V. |

| Ravikant | Washing Machine |

| Electra | Refrigerator |

| K.K. | Cooler |

| Ganesh | A.C. |

| Rajaram | Iron |

| Ravi | Laptop |

| Ravikant | Washing Machine |

+----------+-----------------+

**Same As**

**Another Example for Without Alias(Table) in SQL**

ये example ऊपर दिए हुए example से बिलकुल समान है लेकिन यहाँ पर table को alias name नहीं दिया गया है |

Source Code :

SELECT Appliances.Dealer, Appliances.Appliances

FROM Appliances;

Output :

+----------+-----------------+

| Dealer | Appliances |

+----------+-----------------+

| Balaji | T.V. |

| Ravikant | Washing Machine |

| Electra | Refrigerator |

| K.K. | Cooler |

| Ganesh | A.C. |

| Rajaram | Iron |

| Ravi | Laptop |

| Ravikant | Washing Machine |

+----------+-----------------+

**Another Example for Alias(Columns and Tables) in SQL**

Example पर Appliances को 'a' और Employee को 'e' ये alias name दिया गया है | Appliances table के Dealer और Appliances इस column को और Employee table के emp\_name इस column को select किया गया है | उसके बाद जहा पर Appliances का id; Employee के emp\_id से बराबर होगा वही records return किये गए है |

Source Code :

SELECT a.Dealer, a.Appliances, e.emp\_name

FROM Appliances AS a, Employee AS e

WHERE a.id=e.emp\_id;

Output :

+---------+--------------+---------------+

| Dealer | Appliances | emp\_name |

+---------+--------------+---------------+

| Balaji | T.V. | Shweta Pandit |

| Electra | Refrigerator | Rupal Patil |

| K.K. | Cooler | Shankar Mane |

+---------+--------------+---------------+

उपर के example के मुताबिक returned हुए records को लाल रंग में निर्देशित किया गया है जो id और emp\_id से बराबर है |

Table Name : Appliances

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

Table Name : Employee

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | NULL |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | NULL |

| 20 | Rahul Kapoor | 27000 |

+--------+---------------+------------+

### SQL - GROUP BY Statement

GROUP BY Statement का इस्तेमाल एक या एक से ज्यादा columns के आधार पर records grouped करके retrieve किये जाते है |

SELECT Statement पर aggregate function का इस्तेमाल GROUP BY Statement के साथ किया जा सकता है |

**Aggregate Functions :** AVG(),COUNT(),MAX(),MIN(),SUM()

**Syntax for GROUP BY Statement in SQL**

**Syntax1 :**

SELECT column\_name(s)

FROM table\_name

GROUP BY column\_name(s);

**Syntax2 :**

SELECT column\_name(s)

FROM table\_name

GROUP BY column\_name(s)

ORDER BY column\_name(s);

**Syntax3 : Without Sort**

SELECT column\_name(s)

FROM table\_name

WHERE condition

GROUP BY column\_name(s);

**Syntax4 : With Sort**

SELECT column\_name(s)

FROM table\_name

WHERE condition

GROUP BY column\_name(s)

ORDER BY column\_name(s);

**Syntax5 : Without Sort, With Function**

SELECT column\_name(s),function(column\_name)

FROM table\_name

WHERE condition

GROUP BY column\_name(s);

**Syntax6 : With Sort, Function**

SELECT column\_name(s),function(column\_name)

FROM table\_name

WHERE condition

GROUP BY column\_name(s)

ORDER BY column\_name(s);

**Sample Table**

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 10000 |

| 20 | Rahul Kapoor | 27000 |

| 21 | Shankar Mane | 20000 |

| 22 | Maria Shaikh | 30000 |

| 23 | Vikram Kank | 40000 |

| 24 | Vaibhav Vichare | 10000 |

| 25 | Shweta Pandit | 25000 |

+--------+-----------------+------------+

**Example for GROUP BY Statement in SQL**

Example पर emp\_salary द्वारा grouped करके सभी columns को return किया गया है |

यहाँ पर duplicate salaries को remove किया हुआ और emp\_salary के अनुसार ascending order में sort किया हुआ result-set दिखाई देता है |

Source Code :

SELECT \* FROM Employee

GROUP BY emp\_salary;

Output :

+--------+---------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+---------------+------------+

| 15 | Ramesh Mohane | 10000 |

| 4 | Shankar Mane | 15000 |

| 21 | Shankar Mane | 20000 |

| 25 | Shweta Pandit | 25000 |

| 20 | Rahul Kapoor | 27000 |

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 23 | Vikram Kank | 40000 |

+--------+---------------+------------+

**Example for GROUP BY Statement With Function and ORDER BY Statement in SQL**

Example पर Employee के सभी columns को select(\*) करके उसके साथ COUNT(emp\_salary) को select किया गया है | उसके बाद emp\_salary द्वारा grouped करके COUNT(emp\_salary) से descending order में sort किया गया है |

Source Code :

SELECT \*,COUNT(emp\_salary)

FROM Employee

GROUP BY emp\_salary

ORDER BY COUNT(emp\_salary) DESC;

Output :

+--------+---------------+------------+-------------------+

| emp\_id | emp\_name | emp\_salary | COUNT(emp\_salary) |

+--------+---------------+------------+-------------------+

| 1 | Shweta Pandit | 30000 | 2 |

| 15 | Ramesh Mohane | 10000 | 2 |

| 4 | Shankar Mane | 15000 | 1 |

| 20 | Rahul Kapoor | 27000 | 1 |

| 23 | Vikram Kank | 40000 | 1 |

| 3 | Rupal Patil | 35000 | 1 |

| 21 | Shankar Mane | 20000 | 1 |

| 25 | Shweta Pandit | 25000 | 1 |

+--------+---------------+------------+-------------------+

**Example for GROUP BY Statement With Function, WHERE Clause and ORDER BY Statement in SQL**

Example पर Employee के सभी columns को select(\*) करके उसके साथ COUNT(emp\_salary) को select किया गया है | उसके बाद emp\_id 15 से कम होने वाले emp\_salary द्वारा grouped करके COUNT(emp\_salary) से descending order में sort किया गया है |

Source Code :

SELECT \*,COUNT(emp\_salary)

FROM Employee

WHERE emp\_id<15

GROUP BY emp\_salary

ORDER BY COUNT(emp\_salary) DESC;

Output :

+--------+---------------+------------+-------------------+

| emp\_id | emp\_name | emp\_salary | COUNT(emp\_salary) |

+--------+---------------+------------+-------------------+

| 3 | Rupal Patil | 35000 | 1 |

| 1 | Shweta Pandit | 30000 | 1 |

| 4 | Shankar Mane | 15000 | 1 |

+--------+---------------+------------+-------------------+

**Example for GROUP BY Statement With COUNT() Function in SQL**

Example पर emp\_salary के records को count किया गया है |

For Example, 10000 ये 2 बार, 15000 ये 1 बार , 30000 ये 2 बार emp\_salary इस column पर मौजूद है |

Source Code :

SELECT emp\_salary,COUNT(emp\_salary)

FROM Employee

GROUP BY emp\_salary;

Output :

+------------+-------------------+

| emp\_salary | COUNT(emp\_salary) |

+------------+-------------------+

| 10000 | 2 |

| 15000 | 1 |

| 20000 | 1 |

| 25000 | 1 |

| 27000 | 1 |

| 30000 | 2 |

| 35000 | 1 |

| 40000 | 1 |

+------------+-------------------+

**Try Without GROUP BY Statement With COUNT() Function in SQL**

अगर GROUP BY Statement के बिना column के record को count किया जाता है तो हर query को values के साथ seperately लिखना पड़ता है | जब GROUP BY Statement का इस्तेमाल किया जाता है तो एक ही query में सब records एक ही result-set पर count हो जाते है |

Source Code :

SELECT emp\_salary,COUNT(emp\_salary)

FROM Employee

WHERE emp\_salary=30000;

Output :

+------------+-------------------+

| emp\_salary | COUNT(emp\_salary) |

+------------+-------------------+

| 30000 | 2 |

+------------+-------------------+

### SQL - HAVING Clause

HAVING Clause को GROUP BY Statement के साथ दिया जाता है |

GROUP BY Statement पर Aggregate function के साथ WHERE Clause के साथ इस्तेमाल नहीं किया जा सकता है | लेकिन HAVING Clause का इस्तेमाल GROUP BY Statement पर इस्तेमाल किया जा सकता है |

**Aggregate Functions :** AVG(),COUNT(),MAX(),MIN(),SUM()

**Syntax for GROUP BY Statement in SQL**

**Syntax1 :**

SELECT column\_name(s)

FROM table\_name

GROUP BY column\_name(s);

HAVING condition

**Syntax2 :**

SELECT column\_name(s)

FROM table\_name

GROUP BY column\_name(s)

HAVING condition

ORDER BY column\_name(s);

**Syntax3 : Without Sort**

SELECT column\_name(s)

FROM table\_name

WHERE condition

GROUP BY column\_name(s);

HAVING condition

**Syntax4 : With Sort**

SELECT column\_name(s)

FROM table\_name

WHERE condition

GROUP BY column\_name(s)

HAVING condition

ORDER BY column\_name(s);

**Syntax5 : Without Sort, With Function**

SELECT column\_name(s),function(column\_name)

FROM table\_name

WHERE condition

GROUP BY column\_name(s);

HAVING condition

**Syntax6 : With Sort, Function**

SELECT column\_name(s),function(column\_name)

FROM table\_name

WHERE condition

GROUP BY column\_name(s)

HAVING condition

ORDER BY column\_name(s);

**Sample Table**

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 10000 |

| 20 | Rahul Kapoor | 27000 |

| 21 | Shankar Mane | 20000 |

| 22 | Maria Shaikh | 30000 |

| 23 | Vikram Kank | 40000 |

| 24 | Vaibhav Vichare | 10000 |

| 25 | Shweta Pandit | 25000 |

+--------+-----------------+------------+

**Example for HAVING Clause in SQL**

Example पर सभी columns के साथ COUNT(emp\_salary) इस column को select करके और emp\_salary ये grouped करके 2 से कम counts return किये गए है |

Source Code :

SELECT \*,COUNT(emp\_salary)

FROM Employee

GROUP BY emp\_salary

HAVING COUNT(emp\_salary)<2;

Output :

+--------+---------------+------------+-------------------+

| emp\_id | emp\_name | emp\_salary | COUNT(emp\_salary) |

+--------+---------------+------------+-------------------+

| 4 | Shankar Mane | 15000 | 1 |

| 21 | Shankar Mane | 20000 | 1 |

| 25 | Shweta Pandit | 25000 | 1 |

| 20 | Rahul Kapoor | 27000 | 1 |

| 3 | Rupal Patil | 35000 | 1 |

| 23 | Vikram Kank | 40000 | 1 |

+--------+---------------+------------+-------------------+

### SQL - JOIN

JOIN keyword का इस्तेमाल column की मदद से दो या दो से ज्यादा tables को जोड़ने के लिए किया जाता है |

**Syntax for JOIN in SQL**

**Without WHERE Clause**

SELECT column\_name(s)

FROM table\_name1 JOIN table\_name2

ON table\_name1.column\_name = table\_name2.column\_name;

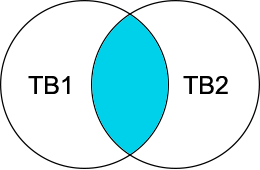
**With WHERE Clause**

SELECT column\_name(s)

FROM table\_name1 JOIN table\_name2

ON table\_name1.column\_name = table\_name2.column\_name

WHERE condition;

**JOIN/INNER JOIN**  


**Sample Tables**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 10000 |

| 20 | Rahul Kapoor | 27000 |

| 21 | Shankar Mane | 20000 |

| 22 | Maria Shaikh | 30000 |

| 23 | Vikram Kank | 40000 |

| 24 | Vaibhav Vichare | 10000 |

| 25 | Shweta Pandit | 25000 |

+--------+-----------------+------------+

**Example for JOIN in SQL**

Example पर Employee table के emp\_id, emp\_name और Appliances table के Appliances(Purchased\_Appliances) और Cost इन columns को select करके Employee और Appliances इन दो tables को join किया गया है | उन दो tables के बीच में Appliances का id और Employee का emp\_id बराबर होनेवाले records को return किया गया है |

Source Code :

SELECT emp\_id, emp\_name, Appliances AS Purchased\_Appliances, Appliances.Cost

FROM Employee JOIN Appliances

ON Employee.emp\_id=Appliances.id;

Output :

+--------+---------------+----------------------+-------+

| emp\_id | emp\_name | Purchased\_Appliances | Cost |

+--------+---------------+----------------------+-------+

| 1 | Shweta Pandit | T.V. | 19000 |

| 3 | Rupal Patil | Refrigerator | 10000 |

| 4 | Shankar Mane | Cooler | 18000 |

+--------+---------------+----------------------+-------+

**Example for JOIN with WHERE Clause in SQL**

Example पर Employee table के emp\_id, emp\_name और Appliances table के Appliances(Purchased\_Appliances) और Cost इन columns को select करके Employee और Appliances इन दो tables को join किया गया है | उन दो tables के बीच में Appliances का id और Employee का emp\_id बराबर होनेवाले और emp\_id 4 से कम वाले records को return किया गया है |

Source Code :

SELECT emp\_id, emp\_name, Appliances AS Purchased\_Appliances, Appliances.Cost

FROM Employee JOIN Appliances

ON Employee.emp\_id=Appliances.id

WHERE Employee.emp\_id<4;

Output :

+--------+---------------+----------------------+-------+

| emp\_id | emp\_name | Purchased\_Appliances | Cost |

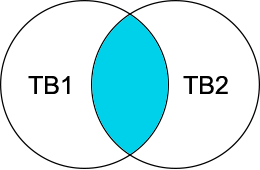
+--------+---------------+----------------------+-------+

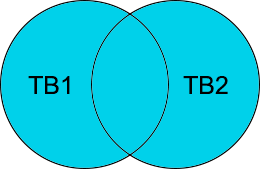
| 1 | Shweta Pandit | T.V. | 19000 |

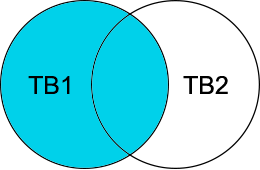
| 3 | Rupal Patil | Refrigerator | 10000 |

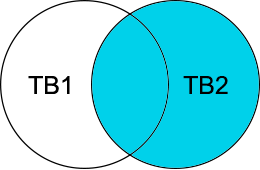
+--------+---------------+----------------------+-------+

**All Types of JOIN**

**JOIN/INNER JOIN**  


**FULL JOIN**  


**LEFT JOIN**  


**RIGHT JOIN**  


मान लो, table1 ये **LEFT table** है और table2 ये **RIGHT table** है |

* **JOIN/INNER JOIN :** दिए गए दोनों tables के values को match करके records return करता है |
* **FULL JOIN :** जब LEFT या RIGHT में match होता है तो सभी records return करता है |
* **LEFT JOIN :** RIGHT table से match करके LEFT table के सभी records को return करता है |
* **RIGHT JOIN :** LEFT table से match करके RIGHT table के सभी records को return करता है |

### SQL - INNER JOIN

INNER JOIN के साथ दिए गए दोनों tables के values को match करके records return करता है |

JOIN keyword और INNER JOIN ये दोनों एक जैसे है |

**Syntax for INNER JOIN in SQL**

**Without WHERE Clause**

SELECT column\_name(s)

FROM table\_name1 INNER JOIN table\_name2

ON table\_name1.column\_name = table\_name2.column\_name;

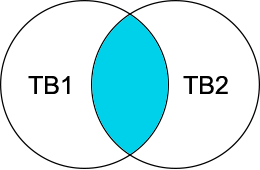
**With WHERE Clause**

SELECT column\_name(s)

FROM table\_name1 INNER JOIN table\_name2

ON table\_name1.column\_name = table\_name2.column\_name

WHERE condition;

**INNER JOIN**  


**Sample Tables**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 10000 |

| 20 | Rahul Kapoor | 27000 |

| 21 | Shankar Mane | 20000 |

| 22 | Maria Shaikh | 30000 |

| 23 | Vikram Kank | 40000 |

| 24 | Vaibhav Vichare | 10000 |

| 25 | Shweta Pandit | 25000 |

+--------+-----------------+------------+

**Example for INNER JOIN in SQL**

Example पर Employee table के emp\_id, emp\_name और Appliances table के Appliances(Purchased\_Appliances) और Cost इन columns को select करके Employee और Appliances इन दो tables को join किया गया है | उन दो tables के बीच में Appliances का id और Employee का emp\_id बराबर होनेवाले और emp\_id 4 से कम वाले records को return किया गया है |

Source Code :

SELECT emp\_id, emp\_name, Appliances AS Purchased\_Appliances, Appliances.Cost

FROM Employee

INNER JOIN Appliances

ON Employee.emp\_id=Appliances.id;

Output :

+--------+---------------+----------------------+-------+

| emp\_id | emp\_name | Purchased\_Appliances | Cost |

+--------+---------------+----------------------+-------+

| 1 | Shweta Pandit | T.V. | 19000 |

| 3 | Rupal Patil | Refrigerator | 10000 |

| 4 | Shankar Mane | Cooler | 18000 |

+--------+---------------+----------------------+-------+

**Returned Records**

ऊपर के Example द्वारा returned हुए records को लाल रंग में निर्देशित किया गया है |

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 10000 |

| 20 | Rahul Kapoor | 27000 |

| 21 | Shankar Mane | 20000 |

| 22 | Maria Shaikh | 30000 |

| 23 | Vikram Kank | 40000 |

| 24 | Vaibhav Vichare | 10000 |

| 25 | Shweta Pandit | 25000 |

+--------+-----------------+------------+

**Example for INNER JOIN with WHERE Clause in SQL**

Example पर Employee table के emp\_id, emp\_name और Appliances table के Appliances(Purchased\_Appliances) और Cost इन columns को select करके Employee और Appliances इन दो tables को join किया गया है | उन दो tables के बीच में Appliances का id और Employee का emp\_id बराबर होनेवाले और emp\_id 4 से कम वाले records को return किया गया है |

Source Code :

SELECT emp\_id, emp\_name, Appliances AS Purchased\_Appliances, Appliances.Cost

FROM Employee INNER JOIN Appliances

ON Employee.emp\_id=Appliances.id

WHERE Employee.emp\_id<4;

Output :

+--------+---------------+----------------------+-------+

| emp\_id | emp\_name | Purchased\_Appliances | Cost |

+--------+---------------+----------------------+-------+

| 1 | Shweta Pandit | T.V. | 19000 |

| 3 | Rupal Patil | Refrigerator | 10000 |

+--------+---------------+----------------------+-------+

**Returned Records**

ऊपर के Example द्वारा returned हुए records को लाल रंग में निर्देशित किया गया है |

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 10000 |

| 20 | Rahul Kapoor | 27000 |

| 21 | Shankar Mane | 20000 |

| 22 | Maria Shaikh | 30000 |

| 23 | Vikram Kank | 40000 |

| 24 | Vaibhav Vichare | 10000 |

| 25 | Shweta Pandit | 25000 |

+--------+-----------------+------------+

### SQL - LEFT JOIN

LEFT JOIN में अगर RIGHT table से match करके LEFT table के सभी records return किये जाते है | अगर RIGHT table से match नहीं किया जाता है तो 'NULL'(no value) return होता है |

**Syntax for LEFT JOIN in SQL**

**Without WHERE Clause**

SELECT column\_name(s)

FROM table\_name1 LEFT JOIN table\_name2

ON table\_name1.column\_name = table\_name2.column\_name;

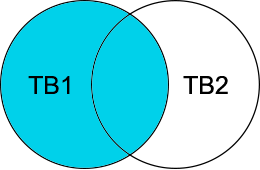
**With WHERE Clause**

SELECT column\_name(s)

FROM table\_name1 LEFT JOIN table\_name2

ON table\_name1.column\_name = table\_name2.column\_name

WHERE condition;

**LEFT JOIN**  


**Sample Tables**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 10000 |

| 20 | Rahul Kapoor | 27000 |

| 21 | Shankar Mane | 20000 |

| 22 | Maria Shaikh | 30000 |

| 23 | Vikram Kank | 40000 |

| 24 | Vaibhav Vichare | 10000 |

| 25 | Shweta Pandit | 25000 |

+--------+-----------------+------------+

**Example for LEFT JOIN in SQL**

Example पर Employee table के emp\_id, emp\_name और Appliances table के Appliances(Purchased\_Appliances) और Cost इन columns को select करके Employee और Appliances इन दो tables को join किया गया है | उन दो tables के बीच में Appliances का id और Employee का emp\_id बराबर होनेवाले records और right table से match न होने वाले सभी LEFT table के records भी return किये गए है |

Source Code :

SELECT emp\_id, emp\_name, Appliances AS Purchased\_Appliances, Appliances.Cost

FROM Employee

LEFT JOIN Appliances

ON Employee.emp\_id=Appliances.id;

Output :

+--------+-----------------+----------------------+-------+

| emp\_id | emp\_name | Purchased\_Appliances | Cost |

+--------+-----------------+----------------------+-------+

| 1 | Shweta Pandit | T.V. | 19000 |

| 3 | Rupal Patil | Refrigerator | 10000 |

| 4 | Shankar Mane | Cooler | 18000 |

| 15 | Ramesh Mohane | NULL | NULL |

| 20 | Rahul Kapoor | NULL | NULL |

| 21 | Shankar Mane | NULL | NULL |

| 22 | Maria Shaikh | NULL | NULL |

| 23 | Vikram Kank | NULL | NULL |

| 24 | Vaibhav Vichare | NULL | NULL |

| 25 | Shweta Pandit | NULL | NULL |

+--------+-----------------+----------------------+-------+

**Example for LEFT JOIN with WHERE Clause in SQL**

Example पर Employee table के emp\_id, emp\_name और Appliances table के Appliances(Purchased\_Appliances) और Cost इन columns को select करके Employee और Appliances इन दो tables को join किया गया है | उन दो tables के बीच में Appliances का id और Employee का emp\_id बराबर होनेवाले और emp\_id 20 से कम होनेवाले records और right table से match न होने वाले सभी LEFT table के records भी return किये गए है |

Source Code :

SELECT emp\_id, emp\_name, Appliances AS Purchased\_Appliances, Appliances.Cost

FROM Employee

LEFT JOIN Appliances

ON Employee.emp\_id=Appliances.id

WHERE Employee.emp\_id<20;

Output :

+--------+---------------+----------------------+-------+

| emp\_id | emp\_name | Purchased\_Appliances | Cost |

+--------+---------------+----------------------+-------+

| 1 | Shweta Pandit | T.V. | 19000 |

| 3 | Rupal Patil | Refrigerator | 10000 |

| 4 | Shankar Mane | Cooler | 18000 |

| 15 | Ramesh Mohane | NULL | NULL |

+--------+---------------+----------------------+-------+

### SQL - RIGHT JOIN

RIGHT JOIN में अगर LEFT table से match करके RIGHT table के सभी records return किये जाते है | अगर LEFT table से match नहीं किया जाता है तो 'NULL'(no value) return होता है |

**Syntax for RIGHT JOIN in SQL**

**Without WHERE Clause**

SELECT column\_name(s)

FROM table\_name1 RIGHT JOIN table\_name2

ON table\_name1.column\_name = table\_name2.column\_name;

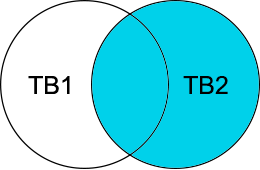
**With WHERE Clause**

SELECT column\_name(s)

FROM table\_name1 RIGHT JOIN table\_name2

ON table\_name1.column\_name = table\_name2.column\_name

WHERE condition;

**RIGHT JOIN**  


**Sample Tables**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 10000 |

| 20 | Rahul Kapoor | 27000 |

| 21 | Shankar Mane | 20000 |

| 22 | Maria Shaikh | 30000 |

| 23 | Vikram Kank | 40000 |

| 24 | Vaibhav Vichare | 10000 |

| 25 | Shweta Pandit | 25000 |

+--------+-----------------+------------+

**Example for RIGHT JOIN in SQL**

Example पर Employee table के emp\_id, emp\_name और Appliances table के Appliances(Purchased\_Appliances) और Cost इन columns को select करके Employee और Appliances इन दो tables को join किया गया है | उन दो tables के बीच में Appliances का id और Employee का emp\_id बराबर होनेवाले records और LEFT table से match न होने वाले सभी RIGHT table के records भी return किये गए है |

Source Code :

SELECT emp\_id, emp\_name, Appliances AS Purchased\_Appliances, Appliances.Cost

FROM Employee

RIGHT JOIN Appliances

ON Employee.emp\_id=Appliances.id;

Output :

+--------+---------------+----------------------+-------+

| emp\_id | emp\_name | Purchased\_Appliances | Cost |

+--------+---------------+----------------------+-------+

| 1 | Shweta Pandit | T.V. | 19000 |

| NULL | NULL | Washing Machine | 15000 |

| 3 | Rupal Patil | Refrigerator | 10000 |

| 4 | Shankar Mane | Cooler | 18000 |

| NULL | NULL | A.C. | 35000 |

| NULL | NULL | Iron | 1500 |

| NULL | NULL | Laptop | 30000 |

| NULL | NULL | Washing Machine | 15000 |

+--------+---------------+----------------------+-------+

**Example for RIGHT JOIN with WHERE Clause in SQL**

Example पर Employee table के emp\_id, emp\_name और Appliances table के Appliances(Purchased\_Appliances) और Cost इन columns को select करके Employee और Appliances इन दो tables को join किया गया है | उन दो tables के बीच में Appliances का id और Employee का emp\_id बराबर होनेवाले और id 20 से कम होनेवाले records और LEFT table से match न होने वाले सभी RIGHT table के records भी return किये गए है |

Source Code :

SELECT emp\_id, emp\_name, Appliances AS Purchased\_Appliances, Appliances.Cost

FROM Employee

RIGHT JOIN Appliances

ON Employee.emp\_id=Appliances.id

WHERE Appliances.id<20;

Output :

+--------+---------------+----------------------+-------+

| emp\_id | emp\_name | Purchased\_Appliances | Cost |

+--------+---------------+----------------------+-------+

| 1 | Shweta Pandit | T.V. | 19000 |

| NULL | NULL | Washing Machine | 15000 |

| 3 | Rupal Patil | Refrigerator | 10000 |

| 4 | Shankar Mane | Cooler | 18000 |

| NULL | NULL | A.C. | 35000 |

| NULL | NULL | Iron | 1500 |

| NULL | NULL | Laptop | 30000 |

| NULL | NULL | Washing Machine | 15000 |

+--------+---------------+----------------------+-------+

### SQL - FULL JOIN

FULL JOIN में LEFT table और RIGHT table में matches हो या ना हो ये दोनों table के सभी records को return करता है अगर match नहीं होता है तो 'NULL'(no value) return होती है |

FULL OUTER JOIN और FULL JOIN ये एक जैसे ही होते है |

FULL JOIN में काफी बड़े result-set को return करने की संभावना होती है |

**Syntax for FULL JOIN in SQL**

**Without WHERE Clause**

SELECT column\_name(s)

FROM table\_name1 FULL JOIN table\_name2

ON table\_name1.column\_name = table\_name2.column\_name;

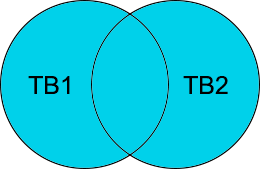
**With WHERE Clause**

SELECT column\_name(s)

FROM table\_name1 FULL JOIN table\_name2

ON table\_name1.column\_name = table\_name2.column\_name

WHERE condition;

**FULL JOIN**  


**Sample Tables**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 10000 |

| 20 | Rahul Kapoor | 27000 |

| 21 | Shankar Mane | 20000 |

| 22 | Maria Shaikh | 30000 |

| 23 | Vikram Kank | 40000 |

| 24 | Vaibhav Vichare | 10000 |

| 25 | Shweta Pandit | 25000 |

+--------+-----------------+------------+

**Example for FULL JOIN with WHERE Clause in SQL**

Source Code :

SELECT e.emp\_id, e.emp\_name, a.Appliances AS Purchased\_Appliances, a.Cost

FROM Employee AS e FULL OUTER JOIN Appliances AS a

ON e.emp\_id=a.id;

### SQL - Self Join

Self Join ये regular join है | इस join के लिए कोई keyword नहीं होता है | यहाँ पे table होता है वो table खुद से ही join होता है |

**Syntax for Self Join in SQL**

**Without WHERE Clause**

SELECT column\_name(s)

FROM table\_name1 t1, table\_name2 t2;

**With WHERE Clause**

SELECT column\_name(s)

FROM table\_name1 t1, table\_name2 t2

WHERE condition

**Sample Tables**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 10000 |

| 20 | Rahul Kapoor | 27000 |

| 21 | Shankar Mane | 20000 |

| 22 | Maria Shaikh | 30000 |

| 23 | Vikram Kank | 40000 |

| 24 | Vaibhav Vichare | 10000 |

| 25 | Shweta Pandit | 25000 |

+--------+-----------------+------------+

**Example for Self Join in SQL**

Example में Employee और Appliances table के सभी columns को select किया गया है और जहा पर emp\_id और id बराबर होने वाले सभी records को return किया गया है |

Source Code :

SELECT \*

FROM Employee e, Appliances a

WHERE e.emp\_id=a.id;

Output :

+--------+---------------+------------+----+---------+--------------+-------+------+

| emp\_id | emp\_name | emp\_salary | id | Dealer | Appliances | Cost | GST |

+--------+---------------+------------+----+---------+--------------+-------+------+

| 1 | Shweta Pandit | 30000 | 1 | Balaji | T.V. | 19000 | 3420 |

| 3 | Rupal Patil | 35000 | 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | Shankar Mane | 15000 | 4 | K.K. | Cooler | 18000 | 5040 |

+--------+---------------+------------+----+---------+--------------+-------+------+

### SQL - UNION Clause

UNION Clause दो या दो से ज्यादा SELECT Statements के result-sets को जोड़ता है |

UNION Clause में जब SELECT Statements दिए जाते है तब उन SELECT Statements पर एक जैसे columns की संख्या होती है |

UNION Clause में सिर्फ unique values को return किया जाता है |

**Syntax for UNION Clause**

SELECT column\_name(s) FROM table\_name1

UNION

SELECT column\_name(s) FROM table\_name2

**Sample Tables**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 10000 |

| 20 | Rahul Kapoor | 27000 |

| 21 | Shankar Mane | 20000 |

| 22 | Maria Shaikh | 30000 |

| 23 | Vikram Kank | 40000 |

| 24 | Vaibhav Vichare | 10000 |

| 25 | Shweta Pandit | 25000 |

+--------+-----------------+------------+

**Example for UNION Clause in SQL**

Example पर Employee table का emp\_salary और Appliances table का Cost इन columns का union बनाके result-set return किया गया है |

Source Code :

SELECT emp\_salary

FROM Employee

UNION

SELECT Cost FROM Appliances;

Output :

+------------+

| emp\_salary |

+------------+

| 30000 |

| 35000 |

| 15000 |

| 10000 |

| 27000 |

| 20000 |

| 40000 |

| 25000 |

| 19000 |

| 18000 |

| 1500 |

+------------+

### SQL - UNION ALL Clause

UNIQUE ALL Clause दो या दो से ज्यादा SELECT Statements के result-sets को जोड़ता है |

UNIQUE ALL Clause में जब SELECT Statements दिए जाते है तब उन SELECT Statements पर एक जैसे columns की संख्या होती है |

UNIQUE ALL Clause में duplicate values को भी return किया जाता है |

**Syntax for UNIQUE ALL Clause**

SELECT column\_name(s) FROM table\_name1

UNION ALL

SELECT column\_name(s) FROM table\_name2

**Sample Tables**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 10000 |

| 20 | Rahul Kapoor | 27000 |

| 21 | Shankar Mane | 20000 |

| 22 | Maria Shaikh | 30000 |

| 23 | Vikram Kank | 40000 |

| 24 | Vaibhav Vichare | 10000 |

| 25 | Shweta Pandit | 25000 |

+--------+-----------------+------------+

**Example for UNIQUE ALL Clause in SQL**

Example पर Employee table का emp\_salary और Appliances table का Cost इन columns का union बनाके result-set return किया गया है |

Source Code :

SELECT emp\_salary

FROM Employee

UNION ALL

SELECT Cost FROM Appliances;

Output :

+------------+

| emp\_salary |

+------------+

| 30000 |

| 35000 |

| 15000 |

| 10000 |

| 27000 |

| 20000 |

| 30000 |

| 40000 |

| 10000 |

| 25000 |

| 19000 |

| 15000 |

| 10000 |

| 18000 |

| 35000 |

| 1500 |

| 30000 |

| 15000 |

+------------+

### SQL - EXISTS Statement

EXISTS Statement का इस्तेमाल दिए हुए subquery पर records पहले से ही है या नहीं ये check करता है | अगर पहले से ही record(s) होता है तो True return होता है और अगर नहीं होता है तो False return होता है |

**Syntax for EXISTS Statement in SQL**

SELECT column\_name(s)

FROM table\_name

WHERE EXISTS(SELECT column\_name

FROM table\_name

WHERE condition);

**Sample Tables**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 10000 |

| 20 | Rahul Kapoor | 27000 |

| 21 | Shankar Mane | 20000 |

| 22 | Maria Shaikh | 30000 |

| 23 | Vikram Kank | 40000 |

| 24 | Vaibhav Vichare | 10000 |

| 25 | Shweta Pandit | 25000 |

+--------+-----------------+------------+

**Example for EXISTS Statement in SQL**

Example पर Employee table के सभी columns select किये गए है | उसके बाद subquery पर Appliances table के Cost इस column को select करके जहा पर Employee table का emp\_salary और Appliances table का Cost इन columns की values बराबर है सिर्फ वो records ही return किये गए है |

Source Code :

SELECT \*

FROM Employee

WHERE EXISTS(SELECT Cost

FROM Appliances

WHERE emp\_salary=Appliances.Cost);

Output :

+--------+-----------------+------------+

| emp\_id | emp\_name | emp\_salary |

+--------+-----------------+------------+

| 1 | Shweta Pandit | 30000 |

| 3 | Rupal Patil | 35000 |

| 4 | Shankar Mane | 15000 |

| 15 | Ramesh Mohane | 10000 |

| 22 | Maria Shaikh | 30000 |

| 24 | Vaibhav Vichare | 10000 |

+--------+-----------------+------------+

### SQL - ANY, ALL Operators

ANY और ALL Operators का इस्तेमाल WHERE या HAVING Clause के साथ किया जाता है |

ANY Operator में सभी subquery values जब condition से मिलती है तब True return किया जाता है |

ALL Operator में कोई भी subquery value जब condition से मिलती है तब True return किया जाता है |

**Syntax for ANY Operator in SQL**

SELECT column\_name(s)

FROM table\_name

WHERE column\_name operator ANY(SELECT column\_name

FROM table\_name

WHERE condition);

**Syntax for ALL Operator in SQL**

SELECT column\_name(s)

FROM table\_name

WHERE column\_name operator ALL(SELECT column\_name

FROM table\_name

WHERE condition);

**Sample Tables**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

**Example for ANY Operator in SQL**

Example पर subqueries के मदद से Appliances table के सभी columns select(\*) करके कोई भी Cost '15000' से बराबर होने वाले rows return किये गए है |

Source Code :

SELECT \*

FROM Appliances

WHERE Cost > ANY(SELECT cost

FROM Appliances

WHERE Cost=15000);

Output :

+----+--------+------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+--------+------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+--------+------------+-------+------+

**Example for ALL Operator in SQL**

Example पर subqueries के मदद से Appliances table के सभी columns select(\*) करके सभी Cost '15000' से ज्यादा होने वाले rows return किये गए है |

Source Code :

SELECT \*

FROM Appliances

WHERE Cost > ALL(SELECT cost

FROM Appliances

WHERE Cost=15000);

Output :

+----+--------+------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+--------+------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Ravi | Laptop | 30000 | 8400 |

+----+--------+------------+-------+------+

### SQL - CREATE DATABASE

SQL में Database create करने के लिए CREATE DATABASE Statement का इस्तेमाल किया जाता है |

**Syntax for CREATE DATABASE in SQL**

CREATE DATABASE database\_name;

**Example for CREATE DATABASE in SQL**

Example पर mydatabase नाम का database create किया गया है |

Source Code :

CREATE DATABASE mydatabase;

### SQL - USE Database

SQL में बहुत सारे Databases होते है लेकिन कौनसा database इस्तेमाल करना है ये बताने के लिए USE DATABASE Statement का इस्तेमाल किया जाता है |

**Syntax for USE Database in SQL**

USE database\_name;

**Example for USE Database in SQL**

Example पर mydatabase नाम का database इस्तेमाल किया गया है |

Source Code :

USE mydatabase;

### SQL - DROP DATABASE

SQL में database को delete करने के लिए DROP DATABASE Statement का इस्तेमाल किया जाता है |

**Syntax for DROP DATABASE in SQL**

DROP DATABASE database\_name;

**Example for DROP DATABASE in SQL**

Example पर mydatabase नाम के database को delete किया गया है |

Source Code :

DROP DATABASE mydatabase;

### SQL - CREATE TABLE

SQL मे data insert करने से पहले database को create करने के बाद उसको use करके table को create करना पड़ता है |

CREATE TABLE Statement से new table को create किया जाता है |

**Syntax for CREATE TABLE Statement in SQL**

CREATE TABLE table\_name (

column\_name1 datatype,

column\_name2 datatype,

.........,

column\_nameN datatype

);

|  |  |
| --- | --- |
| CREATE TABLE | table को create करने के लिए CREATE और TABLE इन दोनों keyword का इस्तेमाल किया है | |
| table\_name( | जिस table को create करना है उस table का दिया जाता है | यहाँ पर column के लिए parenthesis को open(() किया गया है | |
| column\_name1 datatype, column\_name2 datatype, ........., column\_nameN datatype | यहाँ पर एक या एक से ज्यादा columns दिए जाते है | हर column name; unique होता है लेकिन column name के साथ data type देना भी जरुरी होता है | हर column और उसके data type को (,) operator को seperate किया जाता है | |
| ); | यहाँ पर parenthesis को close()) करके semi-colon(;) दिया गया है | semi-colon(;) को देना अनिवार्य होता है | |

**Example for CREATE TABLE Statement in SQL**

Example पर 'mytable' नाम के table को create किया गया है | उस table में id column का data type INT है और उसकीं length 6 है | उसके बाद first\_name और last\_name इन columns का data type VARCHAR और उसकी length 100 दी गयी है |

Source Code :

CREATE TABLE mytable(

id INT(6),

first\_name VARCHAR(100),

last\_name VARCHAR(100));

Output :

Query OK, 0 rows affected (0.47 sec)

**Example for Get Created Table Structure in SQL**

Example पर 'DESC'(description) इस Statement की मदद से table के structure को return किया गया है |

Source Code :

DESC mytable;

Output :

+------------+--------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+------------+--------------+------+-----+---------+-------+

| id | int(6) | YES | | NULL | |

| first\_name | varchar(100) | YES | | NULL | |

| last\_name | varchar(100) | YES | | NULL | |

+------------+--------------+------+-----+---------+-------+

### SQL - Create Table From Existing

अगर Existing table के columns और data किसी दुसरे new table में copy करने के लिए CREATE Statement के साथ SELECT Statement का इस्तेमाल किया जाता है |

**Syntax for Creating Table From Existing Table in SQL**

CREATE TABLE old\_table AS

SELECT column\_name(s)

FROM mytable;

**Sample Table 'mytable'**

+------+------------+-----------+

| id | first\_name | last\_name |

+------+------------+-----------+

| 1 | Akshay | Korke |

+------+------------+-----------+

**Example for Creating Table From Existing Table in SQL**

Example पर mytable का सभी data; mytable1 इस table को create करके उसमे copy किया गया है |

Source Code :

CREATE TABLE mytable1 AS

SELECT \*

FROM mytable;

Output :

Query OK, 1 row affected (0.40 sec)

Records: 1 Duplicates: 0 Warnings: 0

**Table 'mytable1'**

+------+------------+-----------+

| id | first\_name | last\_name |

+------+------------+-----------+

| 1 | Akshay | Korke |

+------+------------+-----------+

### SQL - DROP TABLE

DROP TABLE Statement का इस्तेमाल बनाये हुए table को delete करने के लिये किया जाता है |

**Syntax for DROP TABLE Statement**

DROP TABLE table\_name;

**Example for DROP TABLE Statement in SQL**

Example पर 'mytable' नाम के table को delete किया गया है |

Source Code :

DROP TABLE mytable;

### SQL - TRUNCATE TABLE

TRUNCATE TABLE Statement का इस्तेमाल दिए गए table के data को delete करने के लिए किया जाता है |

इस Statement से table को delete नहीं किया जाता है |

**Syntax for TRUNCATE TABLE Statement**

TRUNCATE TABLE table\_name;

**Example for TRUNCATE TABLE Statement in SQL**

Example पर 'mytable' नाम के table के data को delete किया गया है |

Source Code :

TRUNCATE TABLE mytable;

### SQL - ALTER TABLE

ALTER TABLE Statement का इस्तेमाल दिए गए table के column(s) को add, drop या modify करने के लिए किया जाता है |

**Syntax for ADD Column in a Table Statement**

**For Single Column**

ALTER TABLE table\_name

ADD column\_name data\_type;

**For Multiple Columns**

ALTER TABLE table\_name

ADD column\_name1 data\_type,

ADD column\_name2 data\_type,

ADD .......,

ADD column\_nameN data\_type;

**Example for ADD Column in a Table in SQL**

**Before Adding Column**

Table name : mytable

+------+------------+-----------+

| id | first\_name | last\_name |

+------+------------+-----------+

| 1 | Rakesh | Raman |

+------+------------+-----------+

Example पर 'mytable' नाम के table पर 'marks' नाम का column add किया गया है |

Source Code :

ALTER TABLE mytable;

ADD marks INT(4);

**After Adding Column**

Table name : mytable

+------+------------+-----------+-------+

| id | first\_name | last\_name | marks |

+------+------------+-----------+-------+

| 1 | Rakesh | Raman | NULL |

+------+------------+-----------+-------+

**Add More than One Columns in SQL**

**Before Adding Column**

Table name : mytable

+------+------------+-----------+

| id | first\_name | last\_name |

+------+------------+-----------+

| 1 | Rakesh | Raman |

+------+------------+-----------+

Example पर mytable इस table पर 'Physics', 'Chemistry' और 'Maths' इन तीन columns को add किया गया है |

Source Code :

ALTER TABLE mytable

ADD Physics INT(3),

ADD Chemistry INT(3),

ADD Maths INT(3);

**After Adding Column**

Table name : mytable

+------+------------+-----------+---------+-----------+-------+

| id | first\_name | last\_name | Physics | Chemistry | Maths |

+------+------------+-----------+---------+-----------+-------+

| 1 | Rakesh | Raman | NULL | NULL | NULL |

+------+------------+-----------+---------+-----------+-------+

**DROP Column in a table in SQL**

ALTER TABLE Statement का इस्तेमाल table से column DROP करने के लिए भी किया जाता है |

**Syntax for DROP Column in a Table in SQL**

**for Single Column**

ALTER TABLE table\_name

DROP COLUMN column\_name;

**for Multiple Columns**

ALTER TABLE table\_name

DROP COLUMN column\_name1,

DROP COLUMN column\_name2,

........,

DROP COLUMN column\_nameN;

**Example for DROP Column in a table in SQL**

**Before Deleting Column**

Table name : mytable

+------+------------+-----------+---------+-----------+-------+

| id | first\_name | last\_name | Physics | Chemistry | Maths |

+------+------------+-----------+---------+-----------+-------+

| 1 | Rakesh | Raman | NULL | NULL | NULL |

+------+------------+-----------+---------+-----------+-------+

Example पर mytable इस table के 'Maths' इस column को delete किया गया है |

Source Code :

ALTER TABLE mytable

DROP COLUMN Maths;

**After Deleting Column**

Table name : mytable

+------+------------+-----------+---------+-----------+

| id | first\_name | last\_name | Physics | Chemistry |

+------+------------+-----------+---------+-----------+

| 1 | Rakesh | Raman | NULL | NULL |

+------+------------+-----------+---------+-----------+

**Example for DROP More than One Columns in a table in SQL**

**Before Deleting Column**

Table name : mytable

+------+------------+-----------+---------+-----------+

| id | first\_name | last\_name | Physics | Chemistry |

+------+------------+-----------+---------+-----------+

| 1 | Rakesh | Raman | NULL | NULL |

+------+------------+-----------+---------+-----------+

Example पर mytable इस table के 'Physics' और 'Chemistry' इन columns को delete किया गया है |

Source Code :

ALTER TABLE mytable

DROP COLUMN Physics,

DROP COLUMN Chemistry;

**After Deleting Column**

Table name : mytable

+------+------------+-----------+

| id | first\_name | last\_name |

+------+------------+-----------+

| 1 | Rakesh | Raman |

+------+------------+-----------+

**MODIFY Column in a table in SQL**

ALTER TABLE Statement का इस्तेमाल column को modify करने के लिए भी किया जाता है |

**Syntax for MODIFY Column in a Table in SQL**

**for Single Column**

ALTER TABLE table\_name

MODIFY column\_name column\_type;

**for Multiple Columns**

ALTER TABLE table\_name

MODIFY column\_name1 data\_type;

MODIFY column\_name2 data\_type;

........,

MODIFY column\_nameN data\_type;

Example पर mytable इस table के last\_name इस column का data type modify किया गया है |

Source Code :

ALTER TABLE mytable

MODIFY last\_name TEXT;

### SQL - Renaming Table

SQL में table को rename करने के लिए ALTER TABLE Statement के साथ RENAME TO Statement का इस्तेमाल किया जाता है |

**Syntax for Renaming Table in SQL**

ALTER TABLE table\_name

RENAME TO new\_table\_name;

**Example for Renaming Table in SQL**

Example पर 'mytable' नाम के table 'renamedtable' इस नाम में rename किया गया है |

Source Code :

ALTER TABLE mytable

RENAME TO renamedtable;

### SQL - Constraints

SQL Constraints का इस्तेमाल table के columns के लिए कुछ rules देने के लिए किया जाता है |

Constraints को table CREATE करते वक्त और table ALTER करते वक्त दिया जाता है |

SQL Constraints ये Coloumn level हो सकते है या Table level हो सकते है | यहाँ पर Coloumn level Constraints सिर्फ single column पर ही apply होते है और Table level Constraints ये पूरे table पर apply होते है |

**Syntax for Creating Constraints in SQL**

CREATE TABLE table\_name(

column\_name1 data\_type constraint,

column\_name2 data\_type constraint,

........,

column\_nameN data\_type constraint

);

**Types of Constraints**

* **NOT NULL :** Coloum पर कोई NULL Value नहीं होती है |
* **DEFAULT :** अगर Column पर कोई value दी नहीं जाती है तो यहाँ पर दी हुई default value set की जाती है |
* **PRIMARY KEY :** table में हर record की विशिष्ट रूप से पहचान होती है | अगर ये दिया जाता है तो NOT NULL और UNIQUE constraints देने की जरुरत नहीं पड़ती है |
* **FOREIGN KEY :** दिए गए किसी भी दुसरे table के record की विशिष्ट रूप से पहचान होती है |
* **UNIQUE :** table के column की हर value unique या अलग-अलग होती है |
* **CHECK :** दिए गए table के column की सभी values विशिष्ट condition से समाधानकारक है या नहीं ये सुनिश्चित करता है |
* **INDEX :** जलद गति से data को create और retrieve करने के लिए इस्तेमाल किया जाता है |

### SQL - NOT NULL

SQL के table के हर column अगर कोई value दी नहीं जाती है तो by default 'NULL'(no value) होती है | अगर चाहते हो कि NULL set ना हो तो उस column के लिए constraint देना जरुरी होता है |

Column पर जब NOT NULL constraint दिया जाता है तो by default '0' ये value set की जाती है |

**Example for 'NOT NULL' Constraint in SQL**

Example पर Table1 इस table को create किया गया है | वहा पर id और marks इन columns पर 'NOT NULL' constraint को set किया गया है |

Source Code :

CREATE TABLE Table1(

id INT(2) NOT NULL,

name VARCHAR(100),

marks INT(3) NOT NULL);

### SQL - DEFAULT

SQL के column पर अगर कोई value या constraint दिया नहीं जाता है तो by default NULL value set हो जाती है |

अगर column के value पर कोई भी default value set करना चाहे तो DEFAULT constraint का इस्तेमाल किया जाता है |

अगर column पर कोई value दी नहीं जाती है तो DEFAULT constraint के साथ दी हुई value set हो जाती है |

**Example for 'DEFAULT' Constraint in SQL**

Example पर 'Table2' इस table को create किया गया है | इस table में first\_name और last\_name column पर कोई value दी नहीं जाती है तो 'NONAME' ये default value set हो जाती है |

Source Code :

CREATE TABLE Table2(

id INT(6) NOT NULL,

first\_name VARCHAR(100) DEFAULT 'NONAME',

last\_name VARCHAR(100) DEFAULT 'NONAME',

marks INT(3) NOT NULL);

**Deleting DEFAULT Constraint in SQL**

Example पर Table2 इस table के last\_name इस column से 'DEFAULT' constraint को delete कर दिया गया है |

Source Code :

ALTER TABLE Table2

ALTER COLUMN last\_name DROP DEFAULT;

### SQL - PRIMARY KEY

SQL PRIMARY KEY table के record की विशिष्ट प्रकार की पहचान बनाता है |

PRIMARY KEY ये UNIQUE और NOT NULL इस दोनों constraints का combination होता है |

Database के एक table के लिए सिर्फ एक ही PRIMARY KEY दी जा सकती है | वो एक PRIMARY KEY एक या एक से ज्यादा columns के लिए भी हो सकती है |

PRIMARY KEY ये column-level(single columnn) पर भी दी जाती है और table-level(multiple columns) पर भी दी जाती है |

**Example for 'PRIMARY KEY'(Column Level) Constraint in SQL**

**Syntax for Column-level PRIMARY KEY**

column\_name data\_type PRIMARY\_KEY

Example पर Table3 इस table को create किया गया है | इस table पर id इस column के लिए PRIMARY KEY set की गयी है |

Source Code :

CREATE TABLE Table3(

id INT(2) PRIMARY KEY,

name VARCHAR(100) NOT NULL);

**Example for 'PRIMARY KEY'(Table Level) Constraint in SQL**

**Syntax for Table-level PRIMARY KEY**

...,

CONSTRAINT constraint\_name PRIMARY\_KEY(column\_name1,column\_name2,...,column\_nameN)

Example पर Table4 इस table को create किया गया है | इस create किये हुए table के id और first\_name इन दोनों column पर PRIMARY KEY set की हुई है | जब इनपर कोई record दिया जाता है तब हर record unique होना चाहिए |

Source Code :

CREATE TABLE Table4(

id INT(2),

first\_name VARCHAR(100),

CONSTRAINT PK\_mytable PRIMARY KEY(id, first\_name));

चाहे तो ALTER TABLE से column के PRIMARY KEY को add या drop भी किया जा सकता है |

**Syntax for ADD PRIMARY KEY**

ALTER TABLE table\_name

ADD PRIMARY KEY(column\_name);

**Syntax for DROP PRIMARY KEY**

ALTER TABLE table\_name

DROP PRIMARY KEY;

### SQL - FOREIGN KEY

FOREIGN KEY का इस्तेमाल दिए गए एक table के column का वर्णन किसी दूसरे table के column पर करने के लिए किया जाता है |

SQL में FOREIGN KEY का इस्तेमाल किसी एक table जो दूसरे table के PRIMARY KEY वाले column से सम्बन्ध रखता है |

जिस table पर FOREIGN KEY होती है उसे 'child table' कहा जाता है और जिस table से सम्बन्ध रखा जाता है उसे 'parent table' कहा जाता है |

**Sample Table Description**

Table name : Dealer

+----------------+--------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------------+--------------+------+-----+---------+-------+

| package\_id | int(6) | NO | | NULL | |

| cust\_id | int(6) | NO | PRI | NULL | |

| Address | varchar(200) | NO | | NULL | |

| package\_amount | float | NO | | NULL | |

+----------------+--------------+------+-----+---------+-------+

returned by this query:

desc Dealer;

**Example for FOREIGN KEY(Column Level) in SQL**

Example पर customers इस table के cust\_id इस column से Dealer इस table के cust\_id को link किया गया है | Dealer के cust\_id इस column की value; Customers का cust\_id access कर सकता है | Customers का cust\_id; Dealer के cust\_id के अलावा दूसरा कोई भी value नहीं दे सकता |

Source Code :

CREATE TABLE customers(

cust\_id INT(6) NOT NULL PRIMARY KEY,

cust\_name VARCHAR(100) NOT NULL,

Address VARCHAR(200) NOT NULL,

FOREIGN KEY(cust\_id)

REFERENCES Dealer (cust\_id));

**Same AS**

**Example for FOREIGN KEY(Table Level) in SQL**

Example पर customers इस table के cust\_id इस column से Dealer इस table के cust\_id को link किया गया है |

Source Code :

CREATE TABLE customers(

cust\_id INT(6) NOT NULL PRIMARY KEY,

cust\_name VARCHAR(100) NOT NULL,

Address VARCHAR(200) NOT NULL,

CONSTRAINT fk\_mytable FOREIGN KEY (cust\_id)

REFERENCES Dealer (cust\_id));

**Example for ADD FOREIGN KEY with ALTER TABLE Statement in SQL**

**Syntax :**

ALTER TABLE table\_name

ADD FOREIGN KEY (column\_name1) REFERENCES anotherTable (column\_name1);

Example पर अगर customers पर कोई FOREIGN KEY दी नहीं जाती है और उसे अलग से देना चाहते हो तो ALTER TABLE Statement के साथ उसका इस्तेमाल किया जाता है |

Source Code :

ALTER TABLE customers

ADD FOREIGN KEY (cust\_id) REFERENCES Dealer (cust\_id);

**Example for DROP FOREIGN KEY with ALTER TABLE Statement in SQL**

**Syntax :**

ALTER TABLE table\_name

DROP FOREIGN KEY constraint\_name

Exmaple पर customers इस table के fk\_mytable इस FOREIGN KEY constraint को drop किया गया है |

Source Code :

ALTER TABLE customers

DROP FOREIGN KEY fk\_mytable;

### SQL - UNIQUE

SQL में कई बार column की हर values अलग-अलग या unique होने की आवश्यकता होती है | इस आवश्यकता को पूरा करने के लिए 'UNIQUE' constraint का इस्तेमाल किया जाता है |

UNIQUE constraint का इस्तेमाल column पर सभी values अलग-अलग देने के लिए किया जाता है | लेकिन NULL value के लिए कोई मर्यादा नहीं होती है |

UNIQUE constraint के बजाय PRIMARY KEY का भी इस्तेमाल किया जा सकता है | लेकिन UNIQUE constraint को एक table में एक से ज्यादा बार इस्तेमाल किया जा सकता है | पर PRIMARY KEY को एक table में सिर्फ एक बार ही इस्तेमाल किया जा सकता है |

**Syntax for UNIQUE constraint in SQL**

column\_name data\_type UNIQUE

**Example for UNIQUE constraint(Column Level) in SQL**

Example पर 'Mytable' इस table को create किया गया है और उस table के id इस column के लिए UNIQUE constraint दिया गया है | id column के लिए सिर्फ unique value दी जा सकती है |

Source Code :

CREATE TABLE Mytable(

id INT(4) NOT NULL UNIQUE,

name VARCHAR(100) NOT NULL,

marks INT(2) NOT NULL);

**Example for UNIQUE constraint(Table Level) in SQL**

Example पर 'Mytable1' नाम का table create किया गया है | इस table के id और name इस table के लिए UNIQUE constraint set किया गया है |

Source Code :

CREATE TABLE Mytable1(

id INT(6) NOT NULL,

name VARCHAR(100) NOT NULL,

CONSTRAINT u\_mytable UNIQUE (id,name));

**Example for ADD UNIQUE constraint(Column Level) with ALTER TABLE Statement in SQL**

Example पर 'Mytable1' इस table के column 'name' पर UNIQUE constraint को add किया गया है |

Source Code :

ALTER TABLE Mytable1

ADD UNIQUE (name);

**Example for ADD UNIQUE constraint(Table Level) with ALTER TABLE Statement in SQL**

Example पर 'Mytable2' इस table के column 'id' और 'name' पर UNIQUE constraint को add किया गया है |

Source Code :

ALTER TABLE Mytable2

ADD CONSTRAINT u\_table UNIQUE (id,name);

**Example for DROP UNIQUE constraint with ALTER TABLE Statement in SQL**

Source Code :

ALTER TABLE Mytable2

DROP INDEX u\_table;

### SQL - CHECK

CHECK constraint का इस्तेमाल column पर दिए जानेवाले values के लिए limit देने के लिए किया जाता है |

CHECK constraint जिस column पर दिया जाता है उस condition के मुताबिक column पर data insert किया जाता है |

CHECK constraint column-level(single column) और table-level(multiple columns) होता है |

**Example for CHECK Constraint(Column Level) in SQL**

**Syntax :**

column\_name data\_type CHECK(condition)

Example पर 'Mytable6' इस table को create किया गया है | यहाँ पर marks इस column में सिर्फ 40 इस value से बड़ी values देने की अनुमति दी गयी है |

Source Code :

CREATE TABLE Mytable6(

id INT(2) NOT NULL,

name VARCHAR(100),

marks INT(2) CHECK(marks>40));

**Example for CHECK Constraint(Table Level) in SQL**

**Syntax :**

....,

CONSTRAINT constraint\_name CHECK (condition1 AND condition2)

Example पर 'Mytable7' इस table को create किया गया है | यहाँ पर marks इस column में सिर्फ 40 इस value से बड़ी values देने की और id इस column पर 10 से बड़ी values देने की अनुमति दी हुई है |

Source Code :

CREATE TABLE Mytable7(

id INT(2) NOT NULL,

name VARCHAR(100),

marks INT(2),

CONSTRAINT ck\_mytable CHECK(marks>40 AND id>10));

### SQL - INDEX Constraint

CREATE INDEX Statement का इस्तेमाल tables में indexes create करने के लिए किया जाता है |

INDEX constraint index create करने के साथ-साथ data retrieve भी करता है |

SQL में table के single या multiple columns coulmns का उपयोग करके INDEX को create किया जाता है |

SQL Indexes ये गति प्रदान करने की क्षमता रखता है | Database में हर INDEX name अलग-अलग या unique होता है |

**Syntax for CREATE INDEX Statement in SQL**

CREATE INDEX index\_name

ON table\_name(column\_name1, column\_name2,..., column\_nameN);

**for unique indexes**

CREATE UNIQUE INDEX index\_name

ON table\_name(column\_name1, column\_name2,..., column\_nameN);

**Example for CREATE INDEX Statement(Single Column) in SQL**

Example पर Appliances इस table के id column के लिए index को create किया गया है |

Source Code :

CREATE INDEX myindex\_id

ON Appliances(id);

**Example for CREATE INDEX Statement(Multiple Columns) in SQL**

Example पर Appliances इस table के Appliances और Cost इन columns के लिए index को create किया गया है |

Source Code :

CREATE INDEX myindex\_appCost

ON Appliances(Appliances, Cost);

**Example for DROP INDEX in SQL**

Example पर Appliances इस table के myindex\_id इस index को drop किया गया है |

Source Code :

ALTER TABLE Appliances

DROP INDEX myindex\_id;

### SQL - AUTO INCREMENT

AUTO\_INCREMENT Field का इस्तेमाल automatic नयी numeric values generate करने के लिए किया जाता है |

Table के किसी numeric data type के column के साथ AUTO\_INCREMENT ये specifier दिया जाता है तब उस column पर automatically unique value दी जाती है |

AUTO\_INCREMENT में numeric value की शुरुआत 1 से होती है | जैसे जैसे table पर records insert किये जाते है वैसे वैसे AUTO\_INCREMENT दिए हुए column की values 1-1 से बढती जाती है |

AUTO\_INCREMENT जी column पर दिया जाता है उस column पर PRIMARY KEY या UNIQUE constraint होना जरुरी होता है |

**Syntax for AUTO\_INCREMENT Specifier in SQL**

column\_name data\_type AUTO\_INCREMENT

**Example for AUTO\_INCREMENT Specifier in SQL**

Example पर Student ये table create किया गया है | इस table के id इस column पर AUTO\_INCREMENT ये specifier दिया गया है |

CREATE TABLE Student(

id INT(2) PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL);

**Try to Inserting Record in SQL**

Example पर name इस column पर 'Rakesh' और Mangesh ये values insert की गयी है लेकिन वहा पर id ये भी column है | इस id column पर AUTO\_INCREMENT होने के कारण id पर automatically value insert होगी |

Source Code :

INSERT INTO Student (name) VALUES ('Rakesh'),('Mangesh');

SELECT \* FROM Student;

Output :

+------+------------+

| id | name |

+------+------------+

| 1 | Rakesh |

+------+------------+

| 2 | Mangesh |

+------+------------+

### SQL - VIEW

SQL में CREATE VIEW का इस्तेमाल दिए गए query से virtual table को create करने के लिए किया जाता है |

SQL VIEW में table के records को result-set जैसे ही return करता है |

**Syntax for CREATE VIEW Statement in SQL**

CREATE VIEW view\_name AS

SELECT column\_name1,column\_name2,...,column\_nameN

FROM table\_name

WHERE condition;

**Sample Table**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

**Example for CREATE VIEW Statement in SQL**

Example पर v\_appliances इस VIEW(virtual table) को create किया गया है | इस view में Appliances table के id इस column से 4 से कम वाले सभी columns को set किया गया है |

Source Code :

CREATE VIEW v\_appliances AS

SELECT \* FROM Appliances

WHERE id<4;

**Return Records From VIEW in SQL**

Example पर virtual table(VIEW) 'v\_appliances' से records को result-set में return किया गया है |

Source Code :

SELECT \* FROM v\_appliances;

Output :

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

+----+----------+-----------------+-------+------+

**Return All VIEWS in SQL**

Example पर सभी SQL VIEWs को return किया गया है |

Source Code :

SELECT \*

FROM INFORMATION\_SCHEMA.VIEWS;

Output :

+---------------+---------------------+--------------+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+--------------+--------------+---------+---------------+----------------------+----------------------+-----------+

| TABLE\_CATALOG | TABLE\_SCHEMA | TABLE\_NAME | VIEW\_DEFINITION | CHECK\_OPTION | IS\_UPDATABLE | DEFINER | SECURITY\_TYPE | CHARACTER\_SET\_CLIENT | COLLATION\_CONNECTION | ALGORITHM |

+---------------+---------------------+--------------+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+--------------+--------------+---------+---------------+----------------------+----------------------+-----------+

| def | electronic\_products | v\_appliances | select `electronic\_products`.`appliances`.`id` AS `id`,`electronic\_products`.`appliances`.`Dealer` AS `Dealer`,`electronic\_products`.`appliances`.`Appliances` AS `Appliances`,`electronic\_products`.`appliances`.`Cost` AS `Cost`,`electronic\_products`.`appliances`.`GST` AS `GST` from `electronic\_products`.`appliances` where (`electronic\_products`.`appliances`.`id` < 4) | NONE | YES | @ | DEFINER | cp850 | cp850\_general\_ci | UNDEFINED |

+---------------+---------------------+--------------+----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+--------------+--------------+---------+---------------+----------------------+----------------------+-----------+

**Syntax for CREATE OR REPLACE VIEW(Updating View) in SQL**

अगर VIEW को update करना हो तो इस syntax का इस्तेमाल किया जाता है |

Source Code :

CREATE OR REPLACE VIEW view\_name AS

SELECT column\_name1,column\_name2,...,column\_nameN

FROM table\_name

WHERE condition;

**Syntax for DROP VIEW in SQL**

यहाँ पर VIEW को delete करने का syntax दिया गया है |

Source Code :

DROP VIEW view\_name;

NUMERIC FUNCTIONS

### SQL - ABS

ABS() SQL Function दिए गए Number की absolute value को return करता है |

**Syntax for ABS() SQL Function**

ABS(X)

**Parameter :**

**X :** यहाँ पर number दिया जाता है |

**Example for ABS() SQL Function**

Source Code :

SELECT ABS(-2.5), ABS(2.5);

Output :

+-----------+----------+

| ABS(-2.5) | ABS(2.5) |

+-----------+----------+

| 2.5 | 2.5 |

+-----------+----------+

### SQL - AVG

AVG() SQL Function दिए गए expression की average value return करता है |

AVG() Function में numeric expressions का इस्तेमाल किया जाता है |अगर नहीं किया जाता है तो '0' return होता है |

**AVG() = SUM() / COUNT()**

**Syntax for AVG() SQL Function**

AVG(expression)

**Parameter :**

**expression :** यहाँ पर expression दिया जाता है |

**Sample Table**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Balaji | T.V. | 19000 | 3420 |

| 2 | Ravikant | Washing Machine | 15000 | 4200 |

| 3 | Electra | Refrigerator | 10000 | 2800 |

| 4 | K.K. | Cooler | 18000 | 5040 |

| 5 | Ganesh | A.C. | 35000 | 9800 |

| 6 | Rajaram | Iron | 1500 | 420 |

| 7 | Ravi | Laptop | 30000 | 8400 |

| 8 | Ravikant | Washing Machine | 15000 | 4200 |

+----+----------+-----------------+-------+------+

**Example for AVG() SQL Function**

Source Code :

SELECT AVG(Cost)

FROM Appliances;

Output :

+------------+

| AVG(Cost) |

+------------+

| 17937.5000 |

+------------+

**Same AS**  
  
Source Code :

SELECT (SUM(Cost)/COUNT(\*))

FROM Appliances;

Output :

+----------------------+

| (SUM(Cost)/COUNT(\*)) |

+----------------------+

| 17937.5000 |

+----------------------+

### SQL - CEIL

CEIL() SQL Function दिए गए Floating-Point Number से बड़ी Integer value को या दी हुई Integer Number से equal value को return करता है |

**Syntax for CEIL() SQL Function**

CEIL(X)

**Parameter :**

**X :** यहाँ पर number दिया जाता है |

CEIL() और CEILING() ये दोनों एक जैसे Functions है |

**Example for CEIL() SQL Function**

Source Code :

SELECT CEIL(2.50),CEIL(2.51),CEIL(2.49);

Output :

+------------+------------+------------+

| CEIL(2.50) | CEIL(2.51) | CEIL(2.49) |

+------------+------------+------------+

| 3 | 3 | 3 |

+------------+------------+------------+

### SQL - CEILING

CEILING() SQL Function दिए गए Floating-Point Number से बड़ी Integer value को या दी हुई Integer Number से equal value को return करता है |

**Syntax for CEILING() SQL Function**

CEILING(X)

**Parameter :**

**X :** यहाँ पर number दिया जाता है |

CEILING() और CEILING() ये दोनों एक जैसे Functions है |

**Example for CEILING() SQL Function**

Source Code :

SELECT CEILING(2.50),CEILING(2.51),CEILING(2.49);

Output :

+---------------+---------------+---------------+

| CEILING(2.50) | CEILING(2.51) | CEILING(2.49) |

+---------------+---------------+---------------+

| 3 | 3 | 3 |

+---------------+---------------+---------------+

### SQL - COUNT

COUNT() SQL Function दिए गए expression से records की संख्या को return करता है |

**Syntax for COUNT() SQL Function**

COUNT(expression)

**Parameter :**

**expression :** यहाँ पर expression दिया जाता है |

COUNT() Function में NULL value को count नहीं किया जाता है |

**Sample Table**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Ravikant | Washing Machine | 15000 | 4200 |

| 2 | Balaji | T.V. | 19000 | 3420 |

| 3 | Ravikant | Washing Machine | 15000 | 4200 |

| 4 | Electra | Refrigerator | 10000 | 2800 |

| 5 | K.K. | Cooler | 18000 | 5040 |

| 6 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Rajaram | Iron | 1500 | 420 |

| 8 | Ravi | Laptop | 30000 | NULL |

+----+----------+-----------------+-------+------+

**Example for COUNT() SQL Function**

Example पर Appliances table के GST इस column के जरिये records की संख्या को return किया गया है |

Source Code :

SELECT COUNT(GST)

FROM Appliances;

Output :

+------------+

| COUNT(GST) |

+------------+

| 7 |

+------------+

**Another Example for COUNT() SQL Function**

Example पर Appliances table के सभी columns के जरिये records की संख्या को return किया गया है |

Source Code :

SELECT COUNT(GST)

FROM Appliances;

Output :

+----------+

| COUNT(\*) |

+----------+

| 8 |

+----------+

### SQL - FLOOR

FLOOR() SQL Function दिए गए Floating-Point Number से छोटी Integer value को या दी हुई Integer Number से equal value को return करता है |

**Syntax for FLOOR() SQL Function**

FLOOR(X)

**Parameter :**

**X :** यहाँ पर number दिया जाता है |

**Example for FLOOR() SQL Function**

Source Code :

SELECT FLOOR(2.50),FLOOR(2.51),FLOOR(2.49),FLOOR(2.99);

Output :

+-------------+-------------+-------------+-------------+

| FLOOR(2.50) | FLOOR(2.51) | FLOOR(2.49) | FLOOR(2.99) |

+-------------+-------------+-------------+-------------+

| 2 | 2 | 2 | 2 |

+-------------+-------------+-------------+-------------+

### SQL - GREATEST

GREATEST() SQL Function दिए गए expressions के list से greatest value को return करता है |

GREATEST() Function में दिए गए पहले expression का जो data type होता है वही data type उसका return type होता है | ये expressions को एक दूसरे से compare करता है |

GREATEST() Function ये comparison से पहले जो exp1 का data type होता है उसमे बाकी के expressions को convert किया जाता है |

**Syntax for GREATEST() SQL Function**

GREATEST(exp1, exp2, ..., expN)

**Parameter :**

**exp1, exp2, ..., expN :** यहाँ पर expressions दिये जाते है |

**Example for GREATEST() SQL Function**

Source Code :

SELECT GREATEST("5", "10", "8"), GREATEST(5, 10, 8);

Output :

+--------------------------+--------------------+

| GREATEST("5", "10", "8") | GREATEST(5, 10, 8) |

+--------------------------+--------------------+

| 8 | 10 |

+--------------------------+--------------------+

**Another Example for GREATEST() SQL Function**

Source Code :

SELECT GREATEST("Hello", "World", "Hiii");

Output :

+------------------------------------+

| GREATEST("Hello", "World", "Hiii") |

+------------------------------------+

| World |

+------------------------------------+

### SQL - LEAST

LEAST() SQL Function दिए गए expressions के list से smallest value को return करता है |

LEAST() Function ये comparison से पहले जो exp1 का data type होता है उसमे बाकी के expressions को convert किया जाता है |

अगर Expression के रूप में NULL value होती है तो LEAST() Function द्वारा NULL return किया जाता है |

**Syntax for LEAST() SQL Function**

LEAST(exp1, exp2, ..., expN)

**Parameter :**

**exp1, exp2, ..., expN :** यहाँ पर expressions दिये जाते है |

**Example for LEAST() SQL Function**

Source Code :

SELECT LEAST("5", "10", "8"), LEAST(5, 10, 8);

Output :

+-----------------------+-----------------+

| LEAST("5", "10", "8") | LEAST(5, 10, 8) |

+-----------------------+-----------------+

| 10 | 5 |

+-----------------------+-----------------+

**Another Example for LEAST() SQL Function**

Source Code :

SELECT LEAST("Hello", "World", "Hiii");

Output :

+---------------------------------+

| LEAST("Hello", "World", "Hiii") |

+---------------------------------+

| Hello |

+---------------------------------+

**Another Example for LEAST() SQL Function**

Source Code :

SELECT LEAST("Hello", "World", NULL, "Hiii");

Output :

+---------------------------------------+

| LEAST("Hello", "World", NULL, "Hiii") |

+---------------------------------------+

| NULL |

+---------------------------------------+

### SQL - MAX

MAX() SQL Function दिए गए expression से maximum value को return करता है |

**Syntax for MAX() SQL Function**

MAX(expression)

**Parameter :**

**expression :** यहाँ पर expression दिया जाता है |

**Sample Table**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Ravikant | Washing Machine | 15000 | 4200 |

| 2 | Balaji | T.V. | 19000 | 3420 |

| 3 | Ravikant | Washing Machine | 15000 | 4200 |

| 4 | Electra | Refrigerator | 10000 | 2800 |

| 5 | K.K. | Cooler | 18000 | 5040 |

| 6 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Rajaram | Iron | 1500 | 420 |

| 8 | Ravi | Laptop | 30000 | NULL |

+----+----------+-----------------+-------+------+

**Example for MAX() SQL Function**

Example पर Appliances table के Appliances इस column के maximum value को return किया गया है |

Source Code :

SELECT MAX(Appliances) FROM Appliances;

Output :

+-----------------+

| MAX(Appliances) |

+-----------------+

| Washing Machine |

+-----------------+

**Another Example for MAX() SQL Function**

Example पर Appliances table के Cost इस column के maximum value को मतलब सबसे महंगे Appliance की cost को return किया गया है |

Source Code :

SELECT MAX(Cost) FROM Appliances;

Output :

+-----------+

| MAX(Cost) |

+-----------+

| 35000 |

+-----------+

### SQL - MIN

MIN() SQL Function दिए गए expression से minimum value को return करता है |

**Syntax for MIN() SQL Function**

MIN(expression)

**Parameter :**

**expression :** यहाँ पर expression दिया जाता है |

**Sample Table**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Ravikant | Washing Machine | 15000 | 4200 |

| 2 | Balaji | T.V. | 19000 | 3420 |

| 3 | Ravikant | Washing Machine | 15000 | 4200 |

| 4 | Electra | Refrigerator | 10000 | 2800 |

| 5 | K.K. | Cooler | 18000 | 5040 |

| 6 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Rajaram | Iron | 1500 | 420 |

| 8 | Ravi | Laptop | 30000 | NULL |

+----+----------+-----------------+-------+------+

**Example for MIN() SQL Function**

Example पर Appliances table के Appliances इस column के maximum value को return किया गया है |

Source Code :

SELECT MIN(Appliances) FROM Appliances;

Output :

+-----------------+

| MIN(Appliances) |

+-----------------+

| A.C. |

+-----------------+

**Another Example for MIN() SQL Function**

Example पर Appliances table के Cost इस column के minimum value को मतलब सबसे सस्ते Appliance की cost को return किया गया है |

Source Code :

SELECT MIN(Cost) FROM Appliances;

Output :

+-----------+

| MIN(Cost) |

+-----------+

| 1500 |

+-----------+

### SQL - MOD

MOD() SQL Function दिए गए denominator और numerator के जरिये remainder को return करता है |

**Syntax for MOD() SQL Function**

MOD(dividend/denominator, divisor/numerator)

**Parameter :**

**dividend/denominator :** यहाँ पर number दिया जाता है |

**divisor/numerator :** यहाँ पर number दिया जाता है |

**Example for MOD() SQL Function**

Source Code :

SELECT MOD(10,3),MOD(8,4),MOD(8,3);

Output :

+-----------+----------+----------+

| MOD(10,3) | MOD(8,4) | MOD(8,3) |

+-----------+----------+----------+

| 1 | 0 | 2 |

+-----------+----------+----------+

### SQL - PI

PI() SQL Function ये PI की value को return करता है |

**Syntax for PI() SQL Function**

PI()

**Parameter :**

PI() Function में कोई parameter नहीं होता है |

**Example for PI() SQL Function**

Source Code :

SELECT PI();

Output :

+----------+

| PI() |

+----------+

| 3.141593 |

+----------+

### SQL - POW

POW() SQL Function दिए गए base और exponent को calculate करके value को return करता है |

**Syntax for POW() SQL Function**

POW(base, exp)

**Parameter :**

**base :** यहाँ पर base value दी जाती है |

**exp :** यहाँ पर exponent value दी जाती है |

अगर base negative और exponent floating-point number होता है तो 'out of range' error return होता है |

**baseexp**

POW() और POWER() ये दोनों एक जैसे functions है |

**Example for POW() SQL Function**

Source Code :

SELECT POW(10,2),POW(-2,-10),POW(2,2.5);

Output :

+-----------+--------------+-------------------+

| POW(10,2) | POW(-2,-10) | POW(2,2.5) |

+-----------+--------------+-------------------+

| 100 | 0.0009765625 | 5.656854249492381 |

+-----------+--------------+-------------------+

### SQL - POWER

POWER() SQL Function ये दिए गए base और exponent को calculate करके value को return करता है |

**Syntax for POWER() SQL Function**

POWER(base, exp)

**Parameter :**

**base :** यहाँ पर base value दी जाती है |

**exp :** यहाँ पर exponent value दी जाती है |

अगर base negative और exponent floating-point number होता है तो 'out of range' error return होता है |

**baseexp**

POW() और POWER() ये दोनों एक जैसे functions है |

**Example for POWER() SQL Function**

Source Code :

SELECT POWER(10,2),POWER(-2,-10),POWER(2,2.5);

Output :

+-----------+--------------+-------------------+

| POWER(10,2) | POWER(-2,-10) | POWER(2,2.5) |

+-----------+--------------+-------------------+

| 100 | 0.0009765625 | 5.656854249492381 |

+-----------+--------------+-------------------+

### SQL - RAND

RAND() SQL Function ये random number को return या दिए गए range से random number को return करता है |

**Syntax for RAND() SQL Function**

RAND(seed)

**Parameter :**

**seed :** Optional. अगर दिया जाता है तो ये random numbers का repeatable sequence produce करता है |

**Example for RAND() SQL Function**

Example पर RAND() Function में कोई parameter नहीं दिया गया है | लेकिन हर वक्त नया random Number generate होगा |

Source Code :

SELECT RAND();

Output :

+--------------------+

| RAND() |

+--------------------+

| 0.3450670981268092 |

+--------------------+

**Example for RAND() With Parameter in SQL Function**

Example पर RAND() Function में parameter दिया गया है | लेकिन हर वक्त एक ही random Number generate होगा |

Source Code :

SELECT RAND(6);

Output :

+--------------------+

| RAND(6) |

+--------------------+

| 0.6563190842571847 |

+--------------------+

**Set Range for RAND() Function in SQL**

**Syntax :**

RAND()\*(to-from)+from);

Example पर 10 से 12 के बीच के random number को return किया गया है |

Source Code :

SELECT RAND()\*(12-10)+10;

Output :

+--------------------+

| RAND()\*(12-10)+10 |

+--------------------+

| 11.370565797547405 |

+--------------------+

अगर चाहते हो कि Integer Random Number return हो तो RAND() के साथ CEIL() या FLOOR() Function का इस्तेमाल किया जाता है |

**Try :**

यहाँ पर 10(exclusive) और 30(inclusive) के बीच के integer random number को return किया जायेगा |

SELECT CEIL(RAND()\*(30-10)+10);

यहाँ पर 10(inclusive) और 30(exclusive) के बीच के integer random number को return किया जायेगा |

SELECT FLOOR(RAND()\*(30-10)+10);

### SQL - ROUND

ROUND() SQL Function दिए गए Floating-point Number और decimal\_places के जरिये rounded value को return करता है |

**Syntax for ROUND() SQL Function**

ROUND(X, decimal\_places)

**Parameter :**

**x :** यहाँ पर Floating-point number दिया जाता है |

**decimal\_places :** दिए गए Floating-point Number से decimal point के बाद कितने digits चाहिए वो positive value यहाँ पर दी जाती है |

ROUNG() और TRUNCATE() ये दोनों विरुद्ध functions है |

**Example for ROUND() SQL Function**

Source Code :

SELECT ROUND(1.124,0),ROUND(-1.124,1),ROUND(1.124,2);

Output :

+----------------+-----------------+----------------+

| ROUND(1.124,0) | ROUND(-1.124,1) | ROUND(1.124,2) |

+----------------+-----------------+----------------+

| 1 | -1.1 | 1.12 |

+----------------+-----------------+----------------+

### SQL - SQRT

SQRT() SQL Function दिए गए number का square root return करता है |

**Syntax for SQRT() SQL Function**

SQRT(X)

**Parameter :**

**X :** यहाँ पर positive number दिया जाता है | अगर negative number दिया जाता है तो 'NULL' return होता है |

**Example for SQRT() SQL Function**

Source Code :

SELECT SQRT(8),SQRT(4),SQRT(16),SQRT(-16);

Output :

+--------------------+---------+----------+-----------+

| SQRT(8) | SQRT(4) | SQRT(16) | SQRT(-16) |

+--------------------+---------+----------+-----------+

| 2.8284271247461903 | 2 | 4 | NULL |

+--------------------+---------+----------+-----------+

### SQL - SUM

SUM() SQL Function दिए गए expression से sum की हुई value को return करता है |

**Syntax for SUM() SQL Function**

SUM(expression)

**Parameter :**

**expression :** यहाँ पर expression दिया जाता है |

**Sample Table**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Ravikant | Washing Machine | 15000 | 4200 |

| 2 | Balaji | T.V. | 19000 | 3420 |

| 3 | Ravikant | Washing Machine | 15000 | 4200 |

| 4 | Electra | Refrigerator | 10000 | 2800 |

| 5 | K.K. | Cooler | 18000 | 5040 |

| 6 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Rajaram | Iron | 1500 | 420 |

| 8 | Ravi | Laptop | 30000 | NULL |

+----+----------+-----------------+-------+------+

**Example for SUM() SQL Function**

Source Code :

SELECT SUM(Cost)

FROM Appliances;

Output :

+-----------+

| SUM(Cost) |

+-----------+

| 143500 |

+-----------+

### SQL - TRUNCATE

TRUNCATE() SQL Function दिए गए floating-point number से दिए गए decimal\_places को truncate करके value return करता है |

**Syntax for TRUNCATE() SQL Function**

TRUNCATE(X, decimal\_places)

**Parameter :**

**X :** यहाँ पर floating-point number दिया जाता है |

**decimal\_places :** यहाँ पर जितनी value दी जाती है उतने floating-point number के decimal point के बाद digits remove/truncate किये जाते है |

TRUNCATE() और ROUND() ये दोनों विरुद्ध functions है |

**Example for TRUNCATE() SQL Function**

Source Code :

SELECT TRUNCATE(125.1458,2),ROUND(125.1458,3);

Output :

+----------------------+-------------------+

| TRUNCATE(125.1458,2) | ROUND(125.1458,3) |

+----------------------+-------------------+

| 125.14 | 125.146 |

+----------------------+-------------------+

STRING FUNCTION

### SQL - ASCII

ASCII() Function दिए गए character का ASCII Code return करता है |

**Syntax for ASCII() String SQL Function**

ASCII(char)

**Parameter :**

**char :** यहाँ पर कोई भी character दिया जाता है |

**Example for ASCII() String Function in SQL**

Source Code :

SELECT ASCII('A'),ASCII('Akshara'),ASCII(1);

Output :

+------------+------------------+----------+

| ASCII('A') | ASCII('Akshara') | ASCII(1) |

+------------+------------------+----------+

| 65 | 65 | 49 |

+------------+------------------+----------+

**Another Example for ASCII() String Function in SQL**

**Sample Table**

Table Name : Appliances

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Ravikant | Washing Machine | 15000 | 4200 |

| 2 | Balaji | T.V. | 19000 | 3420 |

| 3 | Ravikant | Washing Machine | 15000 | 4200 |

| 4 | Electra | Refrigerator | 10000 | 2800 |

| 5 | K.K. | Cooler | 18000 | 5040 |

| 6 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Rajaram | Iron | 1500 | 420 |

| 8 | Ravi | Laptop | 30000 | NULL |

+----+----------+-----------------+-------+------+

Source Code :

SELECT ASCII(Appliances) FROM Appliances;

Output :

+-------------------+

| ASCII(Appliances) |

+-------------------+

| 87 | - W

| 84 | - T

| 87 | - W

| 82 | - R

| 67 | - C

| 65 | - A

| 73 | - I

| 76 | - L

+-------------------+

### SQL - CHAR LENGTH

CHAR\_LENGTH() Function दिए गए string की length को return करता है |

**Syntax for CHAR\_LENGTH() String SQL Function**

CHAR\_LENGTH(str)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

CHARACTER\_LENGTH() और CHAR\_LENGTH() ये दोनों एक जैसे functions है |

**Example for CHAR\_LENGTH() String Function in SQL**

Source Code :

SELECT CHAR\_LENGTH('Hello'),CHAR\_LENGTH('Hii');

Output :

+----------------------+--------------------+

| CHAR\_LENGTH('Hello') | CHAR\_LENGTH('Hii') |

+----------------------+--------------------+

| 5 | 3 |

+----------------------+--------------------+

**Another Example for CHAR\_LENGTH() String Function in SQL**

**Sample Table**

Table Name : Appliances

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Ravikant | Washing Machine | 15000 | 4200 |

| 2 | Balaji | T.V. | 19000 | 3420 |

| 3 | Ravikant | Washing Machine | 15000 | 4200 |

| 4 | Electra | Refrigerator | 10000 | 2800 |

| 5 | K.K. | Cooler | 18000 | 5040 |

| 6 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Rajaram | Iron | 1500 | 420 |

| 8 | Ravi | Laptop | 30000 | NULL |

+----+----------+-----------------+-------+------+

Source Code :

SELECT CHAR\_LENGTH(Appliances) FROM Appliances;

Output :

+-------------------------+

| CHAR\_LENGTH(Appliances) |

+-------------------------+

| 15 | - Washing Machine

| 4 | - T.V.

| 15 | - Washing Machine

| 12 | - Refrigerator

| 6 | - Cooler

| 4 | - A.C.

| 4 | - Iron

| 6 | - Laptop

+-------------------------+

### SQL - CHARACTER LENGTH

CHARACTER\_LENGTH() Function दिए गए string की length को return करता है |

**Syntax for CHARACTER\_LENGTH() String SQL Function**

CHARACTER\_LENGTH(str)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

CHARACTER\_LENGTH() और CHAR\_LENGTH() ये दोनों एक जैसे functions है |

**Example for CHARACTER\_LENGTH() String Function in SQL**

Source Code :

SELECT CHARACTER\_LENGTH('Hello'),CHARACTER\_LENGTH('Hii');

Output :

+---------------------------+-------------------------+

| CHARACTER\_LENGTH('Hello') | CHARACTER\_LENGTH('Hii') |

+---------------------------+-------------------------+

| 5 | 3 |

+---------------------------+-------------------------+

**Another Example for CHARACTER\_LENGTH() String Function in SQL**

**Sample Table**

Table Name : Appliances

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Ravikant | Washing Machine | 15000 | 4200 |

| 2 | Balaji | T.V. | 19000 | 3420 |

| 3 | Ravikant | Washing Machine | 15000 | 4200 |

| 4 | Electra | Refrigerator | 10000 | 2800 |

| 5 | K.K. | Cooler | 18000 | 5040 |

| 6 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Rajaram | Iron | 1500 | 420 |

| 8 | Ravi | Laptop | 30000 | NULL |

+----+----------+-----------------+-------+------+

Source Code :

SELECT CHARACTER\_LENGTH(Appliances) FROM Appliances;

Output :

+------------------------------+

| CHARACTER\_LENGTH(Appliances) |

+------------------------------+

| 15 | - Washing Machine

| 4 | - T.V.

| 15 | - Washing Machine

| 12 | - Refrigerator

| 6 | - Cooler

| 4 | - A.C.

| 4 | - Iron

| 6 | - Laptop

+------------------------------+

### SQL - CONCAT

CONCAT() Function दिए गए strings को concatenates करके return करता है |

**Syntax for CONCAT() String SQL Function**

CONCAT(str1, str2, ..., str3)

**Parameter :**

**str1, str2, ..., str3 :** यहाँ पर एक या एक से ज्यादा string को दिया जाता है |

**Example for CONCAT() String Function in SQL**

Source Code :

SELECT CONCAT("Hello", "World");

Output :

+--------------------------+

| CONCAT("Hello", "World") |

+--------------------------+

| HelloWorld |

+--------------------------+

**Another Example for CONCAT() String Function in SQL**

**Sample Table**

Table Name : Appliances

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Ravikant | Washing Machine | 15000 | 4200 |

| 2 | Balaji | T.V. | 19000 | 3420 |

| 3 | Ravikant | Washing Machine | 15000 | 4200 |

| 4 | Electra | Refrigerator | 10000 | 2800 |

| 5 | K.K. | Cooler | 18000 | 5040 |

| 6 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Rajaram | Iron | 1500 | 420 |

| 8 | Ravi | Laptop | 30000 | NULL |

+----+----------+-----------------+-------+------+

Example पर Appliances table के Dealer , semi-colon(,) और Appliances इन columns को concatenates किया गया है |

Source Code :

SELECT CONCAT(Dealer,', ',Appliances) FROM Appliances;

Output :

+--------------------------------+

| CONCAT(Dealer,', ',Appliances) |

+--------------------------------+

| Ravikant, Washing Machine |

| Balaji, T.V. |

| Ravikant, Washing Machine |

| Electra, Refrigerator |

| K.K., Cooler |

| Ganesh, A.C. |

| Rajaram, Iron |

| Ravi, Laptop |

+--------------------------------+

### SQL - CONCAT WS

CONCAT\_WS() Function दिए गए strings को seperator के साथ concatenates करके return करता है |

**Syntax for CONCAT\_WS() String SQL Function**

CONCAT\_WS(sep, str1, str2, ..., str3)

**Parameter :**

**sep :** यहाँ पर seperator दिया जाता है |

**str1, str2, ..., str3 :** यहाँ पर एक या एक से ज्यादा string को दिया जाता है |

**Example for CONCAT\_WS() String Function in SQL**

Source Code :

SELECT CONCAT\_WS(", ", "Hello", "World");

Output :

+-----------------------------------+

| CONCAT\_WS(", ", "Hello", "World") |

+-----------------------------------+

| Hello, World |

+-----------------------------------+

**Another Example for CONCAT\_WS() String Function in SQL**

**Sample Table**

Table Name : Appliances

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Ravikant | Washing Machine | 15000 | 4200 |

| 2 | Balaji | T.V. | 19000 | 3420 |

| 3 | Ravikant | Washing Machine | 15000 | 4200 |

| 4 | Electra | Refrigerator | 10000 | 2800 |

| 5 | K.K. | Cooler | 18000 | 5040 |

| 6 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Rajaram | Iron | 1500 | 420 |

| 8 | Ravi | Laptop | 30000 | NULL |

+----+----------+-----------------+-------+------+

Example पर Appliances table के Dealer और Appliances इन columns को concatenates करके comma(,) से seperate किया गया है |

Source Code :

SELECT CONCAT\_WS(", ", Dealer, Appliances) FROM Appliances;

Output :

+-------------------------------------+

| CONCAT\_WS(", ", Dealer, Appliances) |

+-------------------------------------+

| Ravikant, Washing Machine |

| Balaji, T.V. |

| Ravikant, Washing Machine |

| Electra, Refrigerator |

| K.K., Cooler |

| Ganesh, A.C. |

| Rajaram, Iron |

| Ravi, Laptop |

+-------------------------------------+

### SQL - FIELD

FIELD() Function दिए गए value को दिए गए अगले values पर ढूंढकर position को Integer value में return किया जाता है |

**Syntax for FIELD() String SQL Function**

FIELD(findValue, val1, val2, ..., valN)

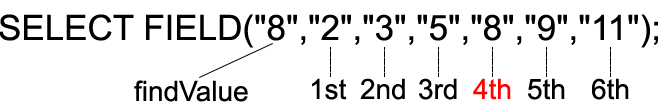
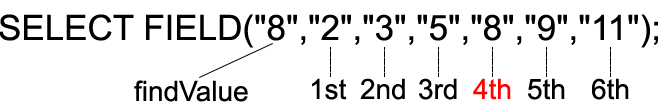
**Parameter :**

**findValue :** इस value को दिए गए अगले value से ढूंढा जाता है |

**val1, val2, ..., valN :** यहाँ पर दिए गए findValue को ढूंढा जाता है |

अगर value को ढूंढा नहीं जाता है तो 0 return किया जाता है |

**Example for FIELD() String Function in SQL**



Example पर 8 इस value को अगले values पर ढूंढ के उसकी position को return किया गया है |

Source Code :

SELECT FIELD("8","2","3","5","8","9","11");

Output :

+-------------------------------------+

| FIELD("8","2","3","5","8","9","11") |

+-------------------------------------+

| 4 |

+-------------------------------------+

**Another Example for FIELD() String Function in SQL**

**Sample Table**

Table Name : Appliances

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Ravikant | Washing Machine | 15000 | 4200 |

| 2 | Balaji | T.V. | 19000 | 3420 |

| 3 | Ravikant | Washing Machine | 15000 | 4200 |

| 4 | Electra | Refrigerator | 10000 | 2800 |

| 5 | K.K. | Cooler | 18000 | 5040 |

| 6 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Rajaram | Iron | 1500 | 420 |

| 8 | Ravi | Laptop | 30000 | NULL |

+----+----------+-----------------+-------+------+

Example पर Appliances table के Appliances इस column पर 'Iron' इस value की position को return किया गया है |

Source Code :

SELECT FIELD("Iron", Appliances) FROM Appliances;

Output :

+---------------------------+

| FIELD("Iron", Appliances) |

+---------------------------+

| 0 |

| 0 |

| 0 |

| 0 |

| 0 |

| 0 |

| 1 | - Iron

| 0 |

+---------------------------+

### SQL - FIND IN SET

FIND\_IN\_SET() Function दिए गए value को दिए गए values के set पर ढूंढकर position को Integer value में return किया जाता है |

**Syntax for FIND\_IN\_SET() String SQL Function**

FIND\_IN\_SET(findValue, "setOfValue")

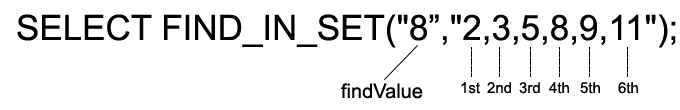
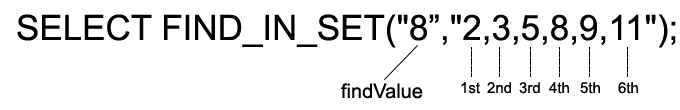
**Parameter :**

**findValue :** इस value को दिए गए अगले setOfValue पर ढूंढा जाता है |

**"setOfValue" :** यहाँ पर दिए गए values के set में findValue को ढूंढा जाता है | setOfValue के हर value को comma(,) से seperate किया जाता है |

अगर ढूंढा नहीं जाता तो 0 return होता है |

**Example for FIND\_IN\_SET() String Function in SQL**



Example पर 8 इस value को अगले values के set पर ढूंढ के उसकी position को return किया गया है |

Source Code :

SELECT FIND\_IN\_SET(8, "2,3,5,8,9,11");

Output :

+--------------------------------+

| FIND\_IN\_SET(8, "2,3,5,8,9,11") |

+--------------------------------+

| 4 |

+--------------------------------+

### SQL - FORMAT

FORMAT() Function दिए गए Floating-point Number और decimal\_places से value को return करता है |

**Syntax for FORMAT() String SQL Function**

FORMAT(X, decimal\_places)

**Parameter :**

**X :** यहाँ पर number दिया जाता है |

**decimal\_places :** दिए गए Floating-point Number से decimal point के बाद कितने digits चाहिए वो positive value यहाँ पर दी जाती है |

FORMAT() और ROUND() ये numeric function दोनों एक जैसे functions है |

**Example for FORMAT() String Function in SQL**

Source Code :

SELECT FORMAT(124.12587, 2), FORMAT(124.12587,0);

Output :

+----------------------+---------------------+

| FORMAT(124.12587, 2) | FORMAT(124.12587,0) |

+----------------------+---------------------+

| 124.13 | 124 |

+----------------------+---------------------+

### SQL - INSERT

INSERT() Function दिए गए old string पर दिए गए substring को insert करके replaced string को return करता है |

**Syntax for INSERT() String SQL Function**

INSERT(oldStr, pos, len, newSubStr)

**Parameter :**

**oldStr :** यहाँ पर old string दिया जाता है |

**pos :** old string से कौनसे position से new string को insert करना है वो position यहाँ पर दी जाती है |

**len :** दिए हुए position से कितने characters remove करने है वो length यहाँ पर दी जाती है |

**newSubStr :** insert करने के लिए यहाँ पर substring दिया जाता है |

String की position 1 से शुरू होती है |

अगर position invalid दी जाती है तो old string return होता है |

**Example for INSERT() String Function in SQL**

Example पर MyWeb इस old string में position 1 से 2 characters remove करके substring को insert करके replaced string को return किया जाता है |

Source Code :

SELECT INSERT("MyWeb", 1, 2, "Your");

Output :

+-------------------------------+

| INSERT("MyWeb", 1, 2, "Your") |

+-------------------------------+

| YourWeb |

+-------------------------------+

### SQL - INSTR

INSTR() Function दिए गए string पर दिए गए substring को search करके उसकी position को return करता है |

**Syntax for INSTR() String SQL Function**

INSTR(str, subStr)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

**subStr :** यहाँ पर substring दिया जाता है |

INSTR() Function case-insensitive होता है |

अगर string पर substring को ढूंढा नहीं जाता तो '0' return होता है |

Position की शुरुआत 0 से होती है |

**Example for INSTR() String Function in SQL**

Example पर HelloWorld इस string से 'w' इस substring को ढूंढकर उसकी position को return किया गया है |

Source Code :

SELECT INSTR("HelloWorld", 'w');

Output :

+--------------------------+

| INSTR("HelloWorld", 'w') |

+--------------------------+

| 6 |

+--------------------------+

**Another Example for INSTR() String Function in SQL**

Example पर Appliances table के Appliances इस column के values पर 'l' इस substring को ढूंढके उसकी position को return किया गया है |

Source Code :

SELECT INSTR(Appliances, 'l')

FROM Appliances;

Output :

+------------------------+

| INSTR(Appliances, 'l') |

+------------------------+

| 0 | - Washing Machine

| 0 | - T.V.

| 0 | - Washing Machine

| 0 | - Refrigerator

| 4 | - Cooler

| 0 | - A.C.

| 0 | - Iron

| 1 | - Laptop

+------------------------+

### SQL - LCASE

LCASE() Function दिए गए string को lowercase में convert करके return करता है |

**Syntax for LCASE() String SQL Function**

LCASE(str)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

LCASE() और LOWER() ये दोनों एक जैसे funtions है |

**Example for LCASE() String Function in SQL**

Example पर दिए string को lowercase में convert करके return किया गया है |

Source Code :

SELECT LCASE("HEllo WORld");

Output :

+----------------------+

| LCASE("HEllo WORld") |

+----------------------+

| hello world |

+----------------------+

**Another Example for LCASE() String Function in SQL**

Example पर Appliances इस table के Appliances इस column की सभी values को lowercase में convert किया गया है |

Source Code :

SELECT LCASE(Appliances)

FROM Appliances;

Output :

+-------------------+

| LCASE(Appliances) |

+-------------------+

| washing machine |

| t.v. |

| washing machine |

| refrigerator |

| cooler |

| a.c. |

| iron |

| laptop |

+-------------------+

### SQL - LEFT

LEFT() Function दिए गए string के left से दिए गए length तक character(s) को return करता है |

**Syntax for LEFT() String SQL Function**

LEFT(str, length)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

**length :** यहाँ पर length दी जाती है |

LEFT() Function दिए जानेवाले length तक string के left से character(s) को लेकर substring को return किया जाता है |

**Example for LEFT() String Function in SQL**

Example पर string के पहले 3 characters को return किया गया है |

Source Code :

SELECT LEFT("HelloWorld",3);

Output :

+----------------------+

| LEFT("HelloWorld",3) |

+----------------------+

| Hel |

+----------------------+

**Example for LEFT() String Function in SQL**

Example पर Appliances table के Appliances इस column के values से पहले 3 characters को return किया गया है |

Source Code :

SELECT LEFT(Appliances,3)

FROM Appliances;

Output :

+--------------------+

| LEFT(Appliances,3) |

+--------------------+

| Was |

| T.V |

| Was |

| Ref |

| Coo |

| A.C |

| Iro |

| Lap |

+--------------------+

### SQL - LENGTH

LENGTH() Function दिए गए string की length को return करता है |

**Syntax for LENGTH() String SQL Function**

LENGTH(str)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

LENGTH(), CHAR\_LENGTH() और CHARACTER\_LENGTH() ये Functions एक जैसे है |

**Example for LENGTH() String Function in SQL**

Example पर दिए गए string की length को return किया गया है |

Source Code :

SELECT LENGTH("Hello World"),CHAR\_LENGTH("Hello World"),CHARACTER\_LENGTH("Hello World");

Output :

+-----------------------+----------------------------+---------------------------------+

| LENGTH("Hello World") | CHAR\_LENGTH("Hello World") | CHARACTER\_LENGTH("Hello World") |

+-----------------------+----------------------------+---------------------------------+

| 11 | 11 | 11 |

+-----------------------+----------------------------+---------------------------------+

**Another Example for LENGTH() String Function in SQL**

**Sample Table**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Ravikant | Washing Machine | 15000 | 4200 |

| 2 | Balaji | T.V. | 19000 | 3420 |

| 3 | Ravikant | Washing Machine | 15000 | 4200 |

| 4 | Electra | Refrigerator | 10000 | 2800 |

| 5 | K.K. | Cooler | 18000 | 5040 |

| 6 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Rajaram | Iron | 1500 | 420 |

| 8 | Ravi | Laptop | 30000 | NULL |

+----+----------+-----------------+-------+------+

Example पर Appliances table के Appliances column के सभी values की length को return किया गया है |

Source Code :

SELECT LENGTH(Appliances)

FROM Appliances;

Output :

+--------------------+

| LENGTH(Appliances) |

+--------------------+

| 15 |

| 4 |

| 15 |

| 12 |

| 6 |

| 4 |

| 4 |

| 6 |

+--------------------+

### SQL - LOCATE

LOCATE() Function दिए गए string के substring के पहले occurrence की position को return करता है |

**Syntax for LOCATE() String SQL Function**

LOCATE(subStr, str, start)

**Parameter :**

**subStr :** यहाँ पर substring दिया जाता है | इस substring को दिए गए string पर ढूंढा जायेगा |

**str :** यहाँ पर string दिया जाता है | इस string पर substring को ढूंढा जायेगा |

**start :** Optional. यहाँ पर starting position दी जाती है | अगर दिया नहीं जाता है तो '1' default होता है | कहा से ढूंढना start करना है वो position यहाँ पर दी जाती है |

String की position 1 से शुरू होती है |

LOCATE() ये function case-insensitive होता है |

अगर string पर substring नहीं होता है तो '0' return होता है |

**Example for LOCATE() String Function in SQL**

Source Code :

SELECT LOCATE("w", "HelloWorldHelloWorld");

Output :

+-------------------------------------+

| LOCATE("w", "HelloWorldHelloWorld") |

+-------------------------------------+

| 6 |

+-------------------------------------+

### SQL - LOWER

LOWER() Function दिए गए string को lowercase में convert करके return करता है |

**Syntax for LOWER() String SQL Function**

LOWER(str)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

LOWER() और LCASE() ये दोनों एक जैसे funtions है |

**Example for LOWER() String Function in SQL**

Example पर दिए string को lowercase में convert करके return किया गया है |

Source Code :

SELECT LOWER("HEllo WORld");

Output :

+----------------------+

| LOWER("HEllo WORld") |

+----------------------+

| hello world |

+----------------------+

**Another Example for LOWER() String Function in SQL**

Example पर Appliances इस table के Appliances इस column की सभी values को lowercase में convert किया गया है |

Source Code :

SELECT LOWER(Appliances)

FROM Appliances;

Output :

+-------------------+

| LOWER(Appliances) |

+-------------------+

| washing machine |

| t.v. |

| washing machine |

| refrigerator |

| cooler |

| a.c. |

| iron |

| laptop |

+-------------------+

### SQL - LPAD

LPAD() Function दिए गए length तक string को left-pad string देकर padded किये हुए string को return किया जाता है |

**Syntax for LPAD() String SQL Function**

LPAD(str, len, padStr)

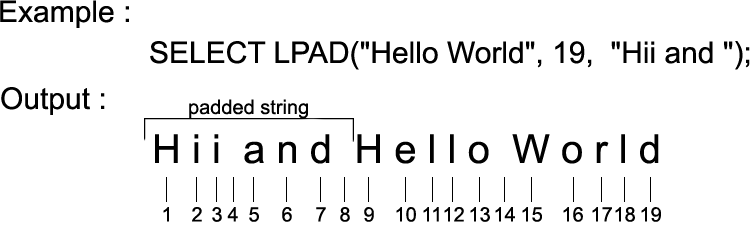
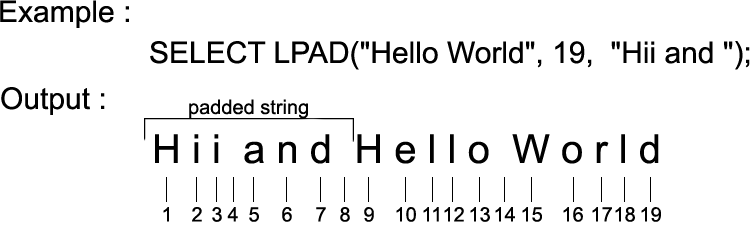
**Parameter :**

**str :** यहाँ पर string दिया जाता है | इस string पर left से padStr दिया जायेगा |

**len :** यहाँ पर length दी जाती है | अगर length str से ज्यादा होती है तो padStr को left पर दिया जाता है और अगर length str से कम होती है तो दिए हुए length तक ही str को return किया जाता है |

**padStr :** यहाँ पर padded string दिया जाता है | इस string को मुख्य string के left से दिया जाता है |

**Example for LPAD() String Function in SQL**



Source Code :

SELECT LPAD("Hello World", 19, "Hii and ");

Output :

+--------------------------------------+

| LPAD("Hello World", 19, "Hii and ") |

+--------------------------------------+

| Hii and Hello World |

+--------------------------------------+

### SQL - LTRIM

LTRIM() Function दिए गए string के शुरुआत के spaces को remove करके string को return करता है |

**Syntax for LTRIM() String SQL Function**

LTRIM(str)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

**Example for LTRIM() String Function in SQL**

Example पर string के leading spaces को remove किया गया है |

Source Code :

SELECT LTRIM(" Hello World");

Output :

+----------------------------+

| LTRIM(" Hello World") |

+----------------------------+

| Hello World |

+----------------------------+

### SQL - MID

MID() Function दिए गए string को starting position से दिए हुए length तक substring को return करता है |

**Syntax for MID() String SQL Function**

MID(str, start, len)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

**start :** यहाँ पर starting position दी जाती है |

**len :** Optional. starting position से दिए हुए length जितने characters return किये जाते है | अगर दिया नहीं जाता है तो starting position से पूरा string return किया जाता है |

string की position 0 से शुरू होती है |

अगर starting position पर positive value दी जाती है तो string के शुरुआत से start किया जाता है |

अगर starting position पर negative value दी जाती है तो string के आखिर से start किया जाता है |

MID(), SUBSTR() और SUBSTRING() ये तीनों Functions एक जैसे है |

**Example for MID() String Function in SQL**

Example पर दिए string पर दूसरे position से 6 characters को return किया गया है |

Source Code :

SELECT MID("Hello World", 2, 6);

Output :

+--------------------------+

| MID("Hello World", 2, 6) |

+--------------------------+

| ello W |

+--------------------------+

### SQL - POSITION

POSITION() Function दिए गए string के substring के पहले occurrence की position को return करता है |

**Syntax for POSITION() String SQL Function**

POSITION(subStr in str)

**Parameter :**

**subStr :** यहाँ पर substring दिया जाता है |

**str :** यहाँ पर string दिया जाता है |

POSITION() Function case-insensitive होता है |

String की position '1' से शुरू होती है |

अगर substring को ढूंढा नहीं जाता है तो '0' return होता है |

**Example for POSITION() String Function in SQL**

Example पर String पर substring पर ढूंढके उसकी position को return किया गया है |

Source Code :

SELECT POSITION("w" IN "HelloWorld");

Output :

+-------------------------------+

| POSITION("w" IN "HelloWorld") |

+-------------------------------+

| 6 |

+-------------------------------+

**Another Example for POSITION() String Function in SQL**

**Sample Table :**

+----+----------+-----------------+-------+------+

| id | Dealer | Appliances | Cost | GST |

+----+----------+-----------------+-------+------+

| 1 | Ravikant | Washing Machine | 15000 | 4200 |

| 2 | Balaji | T.V. | 19000 | 3420 |

| 3 | Ravikant | Washing Machine | 15000 | 4200 |

| 4 | Electra | Refrigerator | 10000 | 2800 |

| 5 | K.K. | Cooler | 18000 | 5040 |

| 6 | Ganesh | A.C. | 35000 | 9800 |

| 7 | Rajaram | Iron | 1500 | 420 |

| 8 | Ravi | Laptop | 30000 | NULL |

+----+----------+-----------------+-------+------+

Example पर Appliances table के Appliances इस column के values से 'w' इस substring के पहले occurrence की positions return की गयी है |

Source Code :

SELECT POSITION("w" IN Appliances)

FROM Appliances;

Output :

+-----------------------------+

| POSITION("w" IN Appliances) |

+-----------------------------+

| 1 | - Washing Machine

| 0 | - T.V.

| 1 | - Washing Machine

| 0 | - Refrigerator

| 0 | - Cooler

| 0 | - A.C.

| 0 | - Iron

| 0 | - Laptop

+-----------------------------+

### SQL - REPEAT

REPEAT() Function दिए गए string को दिए गए number तक repeat करके return किया जाता है |

**Syntax for REPEAT() String SQL Function**

REPEAT(str, numberOfRepeat)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

**numberOfRepeat :** string को कितनी बार repeat करना है वो संख्या यहाँ पर दी जाती है | अगर 0 या 0 से कम value दी जाती है तो कोई भी string return नहीं किया जाता है |

**Example for REPEAT() String Function in SQL**

Source Code :

select REPEAT("Hello World", 3);

Output :

+-----------------------------------+

| REPEAT("Hello World", 3) |

+-----------------------------------+

| Hello WorldHello WorldHello World |

+-----------------------------------+

### SQL - REPLACE

REPLACE() Function दिए गए string से और उसके substring को दिए गए नए substring से replace करके नया string return करता है |

यहाँ पर दिए गए सभी substrings के occurrences को replace किया जाता है |

**Syntax for REPLACE() String SQL Function**

REPLACE(str, fromSubStr, toSubStr)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

**fromSubStr :** यहाँ पर दिए गए string का substring दिया जाता है |

**toSubStr :** यहाँ पर नया substring दिया जाता है |

REPLACE() Function case-sensitive होता है |

अगर दिया जानेवाला substring; string पर नहीं होता है तो original string को return किया जाता है |

**Example for REPLACE() String Function in SQL**

Source Code :

select REPLACE("Hello World Hello World", "World", "Friends");

Output :

+--------------------------------------------------------+

| REPLACE("Hello World Hello World", "World", "Friends") |

+--------------------------------------------------------+

| Hello Friends Hello Friends |

+--------------------------------------------------------+

**Try Another Examples**

select REPLACE("Hello World Hello world", "World", "Friends");

select REPLACE("Hello WorlD Hello World", "D", "d");

### SQL - REVERSE

REVERSE() Function दिए गए string को reverse करके return करता है |

**Syntax for REVERSE() String SQL Function**

REVERSE(str)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

**Example for REVERSE() String Function in SQL**

Source Code :

SELECT REVERSE("Hello World");

Output :

+------------------------+

| REVERSE("Hello World") |

+------------------------+

| dlroW olleH |

+------------------------+

### SQL - RIGHT

RIGHT() Function दिए गए string के right से दिए गए length तक character(s) को return करता है |

**Syntax for RIGHT() String SQL Function**

RIGHT(str, length)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

**length :** यहाँ पर length दी जाती है |

RIGHT() Function दिए जानेवाले length तक string के right से character(s) को लेकर substring को return किया जाता है |

**Example for RIGHT() String Function in SQL**

Example पर string के आखिर के 3 characters को return किया गया है |

Source Code :

SELECT RIGHT("HelloWorld",3);

Output :

+-----------------------+

| RIGHT("HelloWorld",3) |

+-----------------------+

| rld |

+-----------------------+

**Example for RIGHT() String Function in SQL**

Example पर Appliances table के Appliances इस column के values से आखिर के 3 characters को return किया गया है |

Source Code :

SELECT RIGHT(Appliances,3) FROM Appliances;

Output :

+---------------------+

| RIGHT(Appliances,3) |

+---------------------+

| ine |

| .V. |

| ine |

| tor |

| ler |

| .C. |

| ron |

| top |

+---------------------+

### SQL - RPAD

RPAD() Function दिए गए length तक string को right-pad string देकर padded किये हुए string को return किया जाता है |

**Syntax for RPAD() String SQL Function**

RPAD(str, len, padStr)

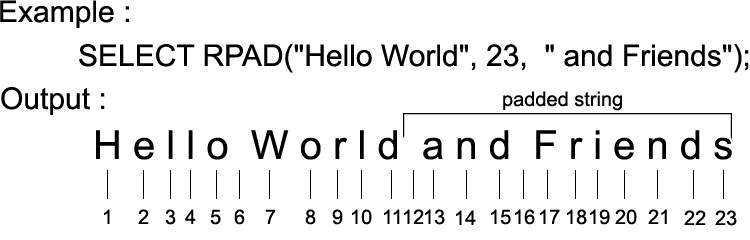
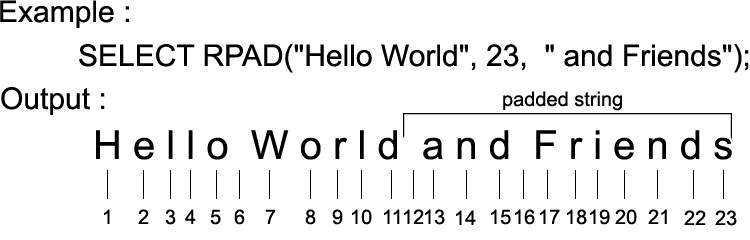
**Parameter :**

**str :** यहाँ पर string दिया जाता है | इस string पर right से padStr दिया जायेगा |

**len :** यहाँ पर length दी जाती है | अगर length str से ज्यादा होती है तो padStr को right पर दिया जाता है और अगर length str से कम होती है तो दिए हुए length तक ही str को return किया जाता है |

**padStr :** यहाँ पर padded string दिया जाता है | इस string को मुख्य string के right से दिया जाता है |

**Example for RPAD() String Function in SQL**



Source Code :

SELECT RPAD("Hello World", 23, " and Friends");

Output :

+------------------------------------------+

| RPAD("Hello World", 23, " and Friends") |

+------------------------------------------+

| Hello World and Friends |

+------------------------------------------+

### SQL - RTRIM

RTRIM() Function दिए गए string के आखिर के spaces को remove करके string को return करता है |

**Syntax for RTRIM() String SQL Function**

RTRIM(str)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

**Example for RTRIM() String Function in SQL**

Example पर string के trailing spaces को remove किया गया है |

Source Code :

SELECT RTRIM("Hello World ");

Output :

+---------------------------------+

| RTRIM("Hello World ") |

+---------------------------------+

| Hello World |

+---------------------------------+

### SQL - SPACE

SPACE() Function दिए गए number जितने space का string return करता है |

**Syntax for SPACE() String SQL Function**

SPACE(X)

**Parameter :**

**X :** यहाँ पर spaces की संख्या दी जाती है |

**Example for SPACE() String Function in SQL**

Source Code :

SELECT CONCAT("Hello",SPACE(20),"World");

Output :

+------------------------------------------+

| CONCAT("Hello",SPACE(30),"World") |

+------------------------------------------+

| Hello World |

+------------------------------------------+

### SQL - STRCMP

STRCMP() Function दिए गए दो string को एक दूसरे से compare करके 0, 1 या -1 return होता है |

**Syntax for STRCMP() String SQL Function**

STRCMP(str1, str2)

**Parameter :**

**str1 :** यहाँ पर पहला string दिया जाता है |

**str2 :** यहाँ पर दूसरा string दिया जाता है |

STRCMP() Function case-insensitive होता है |

अगर str1 और str2 एक जैसे होते है तो 0 return होता है |

अगर str1; str2 से बड़ा होता है या एक जैसे नहीं होते है तो 1 return होता है |

अगर str1; str2 से छोटा होता है या एक जैसे नहीं होते है तो -1 return होता है |

**Example for STRCMP() String Function in SQL**

Source Code :

SELECT STRCMP("Hello World", "Hello World");

Output :

+--------------------------------------+

| STRCMP("Hello World", "Hello World") |

+--------------------------------------+

| 0 |

+--------------------------------------+

**Try Another Examples**

SELECT STRCMP("Hello Worl", "Hello World");

SELECT STRCMP("Hello World", "Hello Worl");

### SQL - SUBSTR

SUBSTR() Function दिए गए string को starting position से दिए हुए length तक substring को return करता है |

**Syntax for SUBSTR() String SQL Function**

SUBSTR(str, start, len)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

**start :** यहाँ पर starting position दी जाती है |

**len :** starting position से दिए हुए length जितने characters return किये जाते है | अगर दिया नहीं जाता है तो starting position से पूरा string return किया जाता है |

SUBSTR(), SUBSTRING() और MID() ये तीनों Functions एक जैसे है |

string की position 0 से शुरू होती है |

अगर starting position पर positive value दी जाती है तो string के शुरुआत से start किया जाता है |

अगर starting position पर negative value दी जाती है तो string के आखिर से start किया जाता है |

**Example for SUBSTR() String Function in SQL**

Example पर दिए string पर दूसरे position से 6 characters को return किया गया है |

Source Code :

SELECT SUBSTR("Hello World", 2, 6);

Output :

+-----------------------------+

| SUBSTR("Hello World", 2, 6) |

+-----------------------------+

| ello W |

+-----------------------------+

### SQL - SUBSTRING

SUBSTRING() Function दिए गए string को starting position से दिए हुए length तक substring को return करता है |

**Syntax for SUBSTRING() String SQL Function**

SUBSTRING(str, start, len)

**OR**

SUBSTRING(str FROM start FOR len)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

**start :** यहाँ पर starting position दी जाती है |

**len :** starting position से दिए हुए length जितने characters return किये जाते है | अगर दिया नहीं जाता है तो starting position से पूरा string return किया जाता है |

SUBSTRING(), SUBSTR() और MID() ये तीनों Functions एक जैसे है |

string की position 0 से शुरू होती है |

अगर starting position पर positive value दी जाती है तो string के शुरुआत से start किया जाता है |

अगर starting position पर negative value दी जाती है तो string के आखिर से start किया जाता है |

**Example for SUBSTRING() String Function in SQL**

Example पर दिए string पर दूसरे position से 6 characters को return किया गया है |

Source Code :

SELECT SUBSTRING("Hello World", 2, 6);

Output :

+--------------------------------+

| SUBSTRING("Hello World", 2, 6) |

+--------------------------------+

| ello W |

+--------------------------------+

### SQL - SUBSTRING INDEX

SUBSTRING\_INDEX() Function दिए गए string पर delimiter की मदद से दिए हुए delimiter के number तक substring को return करता है |

**Syntax for SUBSTRING\_INDEX() String SQL Function**

SUBSTRING\_INDEX(str, delimiter, number)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

**delimiter :** यहाँ पर delimiter दिया जाता है | ये delimiter; str पर मौजूद होता है |

**number :** ये delimiter की संख्या होती है | इस delimiter की संख्या तक string में से substring को return किया जाता है |

**Example for SUBSTRING\_INDEX() String Function in SQL**

Source Code :

SELECT SUBSTRING\_INDEX("My IP 127.0.0.1", ".", 1);

SELECT SUBSTRING\_INDEX("My IP 127.0.0.1", ".", 2);

SELECT SUBSTRING\_INDEX("My IP 127.0.0.1", ".", 3);

Output :

+--------------------------------------------+

| SUBSTRING\_INDEX("My IP 127.0.0.1", ".", 1) |

+--------------------------------------------+

| My IP 127 |

+--------------------------------------------+

+--------------------------------------------+

| SUBSTRING\_INDEX("My IP 127.0.0.1", ".", 2) |

+--------------------------------------------+

| My IP 127.0 |

+--------------------------------------------+

+--------------------------------------------+

| SUBSTRING\_INDEX("My IP 127.0.0.1", ".", 3) |

+--------------------------------------------+

| My IP 127.0.0 |

+--------------------------------------------+

### SQL - TRIM

TRIM() Function दिए गए string से शुरुआत के और आखिर के spaces को remove करके string को return करता है |

**Syntax for TRIM() String SQL Function**

TRIM(str)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

**Example for TRIM() String Function in SQL**

Example पर string के leading और trailing spaces को remove किया गया है |

Source Code :

SELECT CONCAT(TRIM(" Hello "),"World");

Output :

+-------------------------------------+

| CONCAT(TRIM(" Hello "),"World") |

+-------------------------------------+

| HelloWorld |

+-------------------------------------+

### SQL - UCASE

UCASE() Function दिए गए string को uppercase में convert करके return करता है |

**Syntax for UCASE() String SQL Function**

UCASE(str)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

UCASE() और UPPER() ये दोनों एक जैसे funtions है |

**Example for UCASE() String Function in SQL**

Example पर दिए string को uppercase में convert करके return किया गया है |

Source Code :

SELECT UCASE("HEllo WORld");

Output :

+----------------------+

| UCASE("HEllo WORld") |

+----------------------+

| HELLO WORLD |

+----------------------+

**Another Example for UCASE() String Function in SQL**

Example पर Appliances इस table के Appliances इस column की सभी values को uppercase में convert किया गया है |

Source Code :

SELECT UCASE(Appliances)

FROM Appliances;

Output :

+-------------------+

| UCASE(Appliances) |

+-------------------+

| WASHING MACHINE |

| T.V. |

| WASHING MACHINE |

| REFRIGERATOR |

| COOLER |

| A.C. |

| IRON |

| LAPTOP |

+-------------------+

### SQL - UPPER

UPPER() Function दिए गए string को uppercase में convert करके return करता है |

**Syntax for UPPER() String SQL Function**

UPPER(str)

**Parameter :**

**str :** यहाँ पर string दिया जाता है |

UPPER() और UCASE() ये दोनों एक जैसे funtions है |

**Example for UPPER() String Function in SQL**

Example पर दिए string को uppercase में convert करके return किया गया है |

Source Code :

SELECT UPPER("HEllo WORld");

Output :

+----------------------+

| UPPER("HEllo WORld") |

+----------------------+

| HELLO WORLD |

+----------------------+

**Another Example for UPPER() String Function in SQL**

Example पर Appliances इस table के Appliances इस column की सभी values को uppercase में convert किया गया है |

Source Code :

SELECT UPPER(Appliances)

FROM Appliances;

Output :

+-------------------+

| UPPER(Appliances) |

+-------------------+

| WASHING MACHINE |

| T.V. |

| WASHING MACHINE |

| REFRIGERATOR |

| COOLER |

| A.C. |

| IRON |

| LAPTOP |

+-------------------+

**DATE/TIME Functions**

### SQL - ADDDATE

ADDDATE() DATE/TIME Function दिए गए date या time और interval के हिसाब से date को return करता है |

**Syntax for ADDDATE() DATE/TIME Function in SQL**

**Syntax1 :**

ADDDATE(date, INTERVAL exp unit)

**Parameter for Syntax1 :**

**date :** यहाँ पर निचे दिए हुए format जैसी date दी जाती है |  
YYYY-MM-DD HH:MM:SS For Example, 2016-05-26 03:25:57  
YYYY-MM-DD For Example, 2016-05-26  
YY-MM-DD HH:MM:SS For Example, 16-05-26 03:25:57   
YY-MM-DD For Example, 16-05-26

**INTERVAL exp :** INTERVAL keyword के साथ expression पर unit के मुताबिक positive(add date) या negative(subtract date) value दी जाती है |

**unit :** unit पर निचे दी हुई values दी जा सकती है |   
MICROSECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER YEAR, SECOND\_MICROSECOND, MINUTE\_MICROSECOND, MINUTE\_SECOND, HOUR\_MICROSECOND, HOUR\_SECOND, HOUR\_MINUTE, DAY\_MICROSECOND, DAY\_SECOND, DAY\_MINUTE, DAY\_HOUR, YEAR\_MONTH

**Syntax2 :**

ADDDATE(date, days)

**Parameter for Syntax2 :**

**date :** यहाँ पर निचे दिए हुए format जैसी date दी जाती है |  
YYYY-MM-DD HH:MM:SS For Example, 2016-05-26 03:25:57  
YYYY-MM-DD For Example, 2016-05-26

**days :** यहाँ पर days को दिए हुए positive(add date) या negative(subtract date) value के जरिये date पर days add या subtract किया जाता है |

DATE\_ADD() और ADDDATE() ये दोनों Functions एक जैसे होते है |

**Example for ADDDATE()(According to Syntax1) DATE/TIME Function in SQL**

Example पर 2016-05-26 03:25:57 इस Date/Time पर 1 second को add करके date return की गयी है |

Source Code :

SELECT ADDDATE("2016-05-26 03:25:57", INTERVAL 1 SECOND);

Output :

+---------------------------------------------------+

| ADDDATE("2016-05-26 03:25:57", INTERVAL 1 SECOND) |

+---------------------------------------------------+

| 2016-05-26 03:25:58 |

+---------------------------------------------------+

**Another Example for ADDDATE()(According to Syntax1) DATE/TIME Function in SQL**

Example पर 2016-05-26 इस Date पर 2 days को add करके date return की गयी है |

Source Code :

SELECT ADDDATE("2016-05-26", INTERVAL 2 DAY);

Output :

+---------------------------------------+

| ADDDATE("2016-05-26", INTERVAL 2 DAY) |

+---------------------------------------+

| 2016-05-28 |

+---------------------------------------+

**Another Example for ADDDATE()(According to Syntax1) DATE/TIME Function in SQL**

Example पर 2016-05-26 इस Date पर 2 days को subtract करके date return की गयी है |

Source Code :

SELECT ADDDATE("2016-05-26", INTERVAL -2 DAY);

Output :

+----------------------------------------+

| ADDDATE("2016-05-26", INTERVAL -2 DAY) |

+----------------------------------------+

| 2016-05-24 |

+----------------------------------------+

**Another Example for ADDDATE()(According to Syntax2) DATE/TIME Function in SQL**

Example पर 2016-05-26 इस Date पर 4 days को add करके date return की गयी है |

Source Code :

SELECT ADDDATE("2016-05-26", 4);

Output :

+--------------------------+

| ADDDATE("2016-05-26", 4) |

+--------------------------+

| 2016-05-30 |

+--------------------------+

**Another Example for ADDDATE()(According to Syntax2) DATE/TIME Function in SQL**

Example पर 2016-05-26 इस Date पर 4 days को subtract करके date return की गयी है |

Source Code :

SELECT ADDDATE("2016-05-26", -4);

Output :

+---------------------------+

| ADDDATE("2016-05-26", -4) |

+---------------------------+

| 2016-05-22 |

+---------------------------+

### SQL - ADDTIME

ADDTIME() DATE/TIME Function दिए गए time या date time पर time को add करके time या datetime को return करता है |

**Syntax for ADDTIME() DATE/TIME Function in SQL**

SELECT ADDTIME(time/datetime, time)

**Parameter :**

**time/datetime :** यहाँ पर time या datetime दिया जाता है |

**time :** यहाँ पर time दिया जाता है |

**Example for ADDTIME() DATE/TIME Function in SQL**

Example पर दिए हुए datetime पर 2 seconds को add किया गया है |

Source Code :

SELECT ADDTIME("2016-05-26 03:25:57", "2");

Output :

+-------------------------------------+

| ADDTIME("2016-05-26 03:25:57", "2") |

+-------------------------------------+

| 2016-05-26 03:25:59 |

+-------------------------------------+

**Another Example for ADDTIME() DATE/TIME Function in SQL**

Example पर दिए हुए time पर 2 seconds को add किया गया है |

Source Code :

SELECT ADDTIME("03:25:57", "2");

Output :

+-------------------------------------+

| ADDTIME("03:25:57", "2") |

+-------------------------------------+

| 2016-05-26 03:25:59 |

+-------------------------------------+

**Another Example for ADDTIME() DATE/TIME Function in SQL**

Example पर दिए हुए datetime पर 2 days और 1:12:59 इस time को add किया गया है |

Source Code :

SELECT ADDTIME("2016-05-26 03:25:57", "2 1:12:59");

Output :

+---------------------------------------------+

| ADDTIME("2016-05-26 03:25:57", "2 1:12:59") |

+---------------------------------------------+

| 2016-05-28 04:38:56 |

+---------------------------------------------+

**Another Example for ADDTIME() DATE/TIME Function in SQL**

Example पर दिए हुए datetime पर 2 days और 1:12:59 इस time को subtract किया गया है |

Source Code :

SELECT ADDTIME("2016-05-26 03:25:57", "-2 1:12:59");

Output :

+----------------------------------------------+

| ADDTIME("2016-05-26 03:25:57", "-2 1:12:59") |

+----------------------------------------------+

| 2016-05-24 02:12:58 |

+----------------------------------------------+

### SQL - CURDATE

CURDATE() DATE/TIME Function Current Date को return करता है |

**Syntax for CURDATE() DATE/TIME Function in SQL**

CURDATE()

**Parameter :**

CURDATE() Function के लिए कोई parameter नहीं होता है |

Return होनेवाली date 'YYYY-MM-DD' इस format में return होती है |

CURRENT\_DATE() और CURDATE() ये दोनों Functions एक जैसे है |

**Example for CURDATE() DATE/TIME Function in SQL**

Example पर Current date को return किया गया है |

Source Code :

SELECT CURDATE();

Output :

+------------+

| CURDATE() |

+------------+

| 2017-07-22 |

+------------+

### SQL - CURRENT DATE

CURRENT\_DATE() DATE/TIME Function Current Date को return करता है |

**Syntax for CURRENT\_DATE() DATE/TIME Function in SQL**

CURRENT\_DATE()

**Parameter :**

CURRENT\_DATE() Function के लिए कोई parameter नहीं होता है |

Return होनेवाली date 'YYYY-MM-DD' इस format में return होती है |

CURRENT\_DATE() और CURDATE() ये दोनों Functions एक जैसे है |

**Example for CURRENT\_DATE() DATE/TIME Function in SQL**

Example पर Current date को return किया गया है |

Source Code :

SELECT CURRENT\_DATE();

Output :

+----------------+

| CURRENT\_DATE() |

+----------------+

| 2017-07-22 |

+----------------+

### SQL - CURRENT TIME

CURRENT\_TIME() DATE/TIME Function Current Time को return करता है |

**Syntax for CURRENT\_TIME() DATE/TIME Function in SQL**

CURRENT\_TIME()

**Parameter :**

CURRENT\_TIME() Function के लिए कोई parameter नहीं होता है |

Return होनेवाला time 'HH:MM:SS' इस format में return होती है |

CURRENT\_TIME() और CURTIME() ये दोनों Functions एक जैसे है |

**Example for CURRENT\_TIME() DATE/TIME Function in SQL**

Example पर Current time को return किया गया है |

Source Code :

SELECT CURRENT\_TIME();

Output :

+----------------+

| CURRENT\_TIME() |

+----------------+

| 11:15:31 |

+----------------+

### SQL - CURRENT TIMESTAMP

CURRENT\_TIMESTAMP() DATE/TIME Function current timestamp मतलब date और time को return करता है |

**Syntax for CURRENT\_TIMESTAMP() DATE/TIME Function in SQL**

CURRENT\_TIMESTAMP()

**Parameter :**

CURRENT\_TIMESTAMP() Function के लिए कोई parameter नहीं होता है |

Return होनेवाला datetime 'YYYY-MM-DD HH:MM:SS' इस format में return होती है |

NOW(), LOCALTIME(), LOCALTIMESTAMP() और CURRENT\_TIMESTAMP() ये सभी Functions एक जैसे है |

**Example for CURRENT\_TIMESTAMP() DATE/TIME Function in SQL**

Example पर Current time को return किया गया है |

Source Code :

SELECT CURRENT\_TIMESTAMP();

Output :

+---------------------+

| CURRENT\_TIMESTAMP() |

+---------------------+

| 2017-07-22 11:22:24 |

+---------------------+

### SQL - CURTIME

CURTIME() DATE/TIME Function Current Time को return करता है |

**Syntax for CURTIME() DATE/TIME Function in SQL**

CURTIME()

**Parameter :**

CURTIME() Function के लिए कोई parameter नहीं होता है |

Return होनेवाला time 'HH:MM:SS' इस format में return होती है |

CURTIME() और CURTIME() ये दोनों Functions एक जैसे है |

**Example for CURTIME() DATE/TIME Function in SQL**

Example पर Current time को return किया गया है |

Source Code :

SELECT CURTIME();

Output :

+-----------+

| CURTIME() |

+-----------+

| 11:20:16 |

+-----------+

### SQL - DATE

DATE() DATE/TIME Function दिए गए date या datetime से सिर्फ date को return करता है |

**Syntax for DATE() DATE/TIME Function in SQL**

DATE(date/datetime)

अगर invalid format में date या datetime दिया जाता है तो 'NULL' return होता है |

**Parameter :**

**date/datetime :** यहाँ पर date या datetime दिया जाता है | यहाँ पर निचे दिए हुए Format में date या datetime दिया जाता है |  
YYYY-MM-DD HH:MM:SS For Example, 2016-05-26 03:25:57  
YYYY-MM-DD For Example, 2016-05-26  
YY-MM-DD HH:MM:SS For Example, 16-05-26 03:25:57  
YY-MM-DD For Example, 16-05-26

**Example for DATE() DATE/TIME Function in SQL**

Source Code :

SELECT DATE("2016-05-26 03:25:57");

Output :

+-----------------------------+

| DATE("2016-05-26 03:25:57") |

+-----------------------------+

| 2016-05-26 |

+-----------------------------+

### SQL - DATEDIFF

DATEDIFF() DATE/TIME Function दिए गए दो dates के बीच का difference; negative या positive days में return करता है |

**Syntax for DATEDIFF() DATE/TIME Function in SQL**

DATEDIFF(date1, date2)

**Parameter :**

**date1 :** यहाँ पर 'YYYY-MM-DD' या 'YY-MM-DD' इस format में date दी जाती है |

**date2 :** यहाँ पर 'YYYY-MM-DD' या 'YY-MM-DD' इस format में date दी जाती है |

अगर date1 ये date2 से छोटी होती है तो negative days return किये जाते है |

अगर date1 ये date2 से बड़ी होती है तो positive days return किये जाते है |

**Example for DATEDIFF() DATE/TIME Function in SQL**

Source Code :

SELECT DATEDIFF("2016-05-26", "2015-05-26");

Output :

+--------------------------------------+

| DATEDIFF("2016-05-26", "2015-05-26") |

+--------------------------------------+

| 366 |

+--------------------------------------+

**Another Example for DATEDIFF() DATE/TIME Function in SQL**

Source Code :

SELECT DATEDIFF("2015-05-26", "2016-05-26");

Output :

+--------------------------------------+

| DATEDIFF("2015-05-26", "2016-05-26") |

+--------------------------------------+

| -366 |

+--------------------------------------+

### SQL - DATE ADD

DATE\_ADD() DATE/TIME Function दिए गए date या time और interval के हिसाब से date को return करता है |

**Syntax for DATE\_ADD() DATE/TIME Function in SQL**

DATE\_ADD(date, INTERVAL exp unit)

**Parameter :**

**date :** यहाँ पर निचे दिए हुए format जैसी date दी जाती है |  
YYYY-MM-DD HH:MM:SS For Example, 2016-05-26 03:25:57  
YYYY-MM-DD For Example, 2016-05-26  
YY-MM-DD HH:MM:SS For Example, 16-05-26 03:25:57   
YY-MM-DD For Example, 16-05-26

**INTERVAL exp :** INTERVAL keyword के साथ expression पर unit के मुताबिक positive(add date) या negative(subtract date) value दी जाती है |

**unit :** unit पर निचे दी हुई values दी जा सकती है |   
MICROSECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER YEAR, SECOND\_MICROSECOND, MINUTE\_MICROSECOND, MINUTE\_SECOND, HOUR\_MICROSECOND, HOUR\_SECOND, HOUR\_MINUTE, DAY\_MICROSECOND, DAY\_SECOND, DAY\_MINUTE, DAY\_HOUR, YEAR\_MONTH

DATE\_ADD() और DATE\_SUB() ये दोनों Functions विरुद्ध होते है |

**Example for DATE\_ADD() DATE/TIME Function in SQL**

Example पर 2016-05-26 03:25:57 इस Date/Time पर 1 second को add करके date return की गयी है |

Source Code :

SELECT DATE\_ADD("2016-05-26 03:25:57", INTERVAL 1 SECOND);

Output :

+---------------------------------------------------+

| DATE\_ADD("2016-05-26 03:25:57", INTERVAL 1 SECOND)|

+---------------------------------------------------+

| 2016-05-26 03:25:58 |

+---------------------------------------------------+

**Another Example for DATE\_ADD() DATE/TIME Function in SQL**

Example पर 2016-05-26 इस Date पर 2 days को add करके date return की गयी है |

Source Code :

SELECT DATE\_ADD("2016-05-26", INTERVAL 2 DAY);

Output :

+---------------------------------------+

| DATE\_ADD("2016-05-26", INTERVAL 2 DAY)|

+---------------------------------------+

| 2016-05-28 |

+---------------------------------------+

**Another Example for DATE\_ADD() DATE/TIME Function in SQL**

Example पर 2016-05-26 इस Date पर 2 days को subtract करके date return की गयी है |

Source Code :

SELECT DATE\_ADD("2016-05-26", INTERVAL -2 DAY);

Output :

+----------------------------------------+

| DATE\_ADD("2016-05-26", INTERVAL -2 DAY)|

+----------------------------------------+

| 2016-05-24 |

+----------------------------------------+

### SQL - DATE FORMAT

DATE\_FORMAT() DATE/TIME Function दिए गए date या datetime से दिए गए format जैसे return करता है |

**Syntax for DATE\_FORMAT() DATE/TIME Function in SQL**

DATE\_FORMAT(date/datetime, format)

अगर invalid format में date या datetime दिया जाता है तो 'NULL' return होता है |

**Parameter :**

**date/datetime :** यहाँ पर date या datetime दिया जाता है | यहाँ पर निचे दिए हुए Format में date या datetime दिया जाता है |  
YYYY-MM-DD HH:MM:SS For Example, 2016-05-26 03:25:57  
YYYY-MM-DD For Example, 2016-05-26  
YY-MM-DD HH:MM:SS For Example, 16-05-26 03:25:57  
YY-MM-DD For Example, 16-05-26

**format :** निचे सभी formats दिए गए है | format में दिए हुए सभी examples '2016-05-26 03:25:57 AM' इस datetime के अनुसार दिए गए है |

|  |  |  |
| --- | --- | --- |
| **Directive** | **Description** | **Example** |
| %a | abbreviated weekday name | Thu |
| %b | abbreviated month name | May |
| %c | month in number | 5 |
| %d | day of the month(01 to 31) | 26 |
| %D | day of the month(1 to 31) with suffix | 26th |
| %e | day of the month(01 to 31) same as %d | 26 |
| %f | Microseconds((000000 to 999999)) | 000000 |
| %h | hour(12-hour clock)(01 to 12) | 03 |
| %H | hour(24-hour clock)(00 to 23) | 03 |
| %i | Minutes(00 to 59) | 25 |
| %I | hour(12-hour clock)(01 to 12) Same as %h | 03 |
| %j | day of the year(001 to 366) | 147 |
| %k | hour(24-hour clock)(0 to 23) | 3 |
| %l | hour(12-hour clock)(1 to 12) | 3 |
| %m | month in number(01 to 12) | 05 |
| %M | full month name | May |
| %p | AM or PM according to time | AM |
| %r | only time with (12-hour clock)AM/PM | 03:25:57 AM |
| %S | second in number(00 to 59) | 57 |
| %s | second in number(00 to 59) | 57 |
| %T | full time(24-hour clock) without AM/PM | 03:25:57 |
| %U | week number of the year(00 to 53)(sunday is the first day of the week) | 21 |
| %u | week number of the year(00 to 53)(monday is the first day of the week) | 21 |
| %V | week number of the year(01 to 53)(sunday is the first day of the week) | 21 |
| %v | week number of the year(01 to 53)(monday is the first day of the week) | 21 |
| %w | weekday in number(0 to 6)(sunday is the first day of the week) | 4 |
| %W | weekday full name | Thursday |
| %x | Same as %v | 21 |
| %X | Same as %V | 21 |
| %y | two digit year | 16 |
| %Y | four digit year | 2016 |

DATE\_FORMAT() और STR\_TO\_DATE() ये दोनों एक जैसे functions है |

**Example for DATE\_FORMAT() DATE/TIME Function in SQL**

Source Code :

SELECT DATE\_FORMAT("2016-05-26 03:25:57 AM", "%Y-%m-%d %l:%i:%S %p");

Output :

+---------------------------------------------------------------+

| DATE\_FORMAT("2016-05-26 03:25:57 AM", "%Y-%m-%d %l:%i:%S %p") |

+---------------------------------------------------------------+

| 2016-05-26 3:25:57 AM |

+---------------------------------------------------------------+

### SQL - DATE SUB

DATE\_SUB() DATE/TIME Function दिए गए date या time और interval के हिसाब से date को return करता है |

**Syntax for DATE\_SUB() DATE/TIME Function in SQL**

DATE\_SUB(date, INTERVAL exp unit)

**Parameter :**

**date :** यहाँ पर निचे दिए हुए format जैसी date दी जाती है |  
YYYY-MM-DD HH:MM:SS For Example, 2016-05-26 03:25:57  
YYYY-MM-DD For Example, 2016-05-26  
YY-MM-DD HH:MM:SS For Example, 16-05-26 03:25:57   
YY-MM-DD For Example, 16-05-26

**INTERVAL exp :** INTERVAL keyword के साथ expression पर unit के मुताबिक positive(subtract date) या negative(add date) value दी जाती है |

**unit :** unit पर निचे दी हुई values दी जा सकती है |   
MICROSECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER YEAR, SECOND\_MICROSECOND, MINUTE\_MICROSECOND, MINUTE\_SECOND, HOUR\_MICROSECOND, HOUR\_SECOND, HOUR\_MINUTE, DAY\_MICROSECOND, DAY\_SECOND, DAY\_MINUTE, DAY\_HOUR, YEAR\_MONTH

DATE\_SUB() और DATE\_ADD() ये दोनों Functions विरुद्ध होते है |

SUBDATE() और DATE\_SUB() ये दोनों functions एक जैसे है |

**Example for DATE\_SUB() DATE/TIME Function in SQL**

Example पर 2016-05-26 03:25:57 इस Date/Time पर 1 second को subtract करके date return की गयी है |

Source Code :

SELECT DATE\_SUB("2016-05-26 03:25:57", INTERVAL 1 SECOND);

Output :

+----------------------------------------------------+

| DATE\_SUB("2016-05-26 03:25:57", INTERVAL 1 SECOND) |

+----------------------------------------------------+

| 2016-05-26 03:25:56 |

+----------------------------------------------------+

**Another Example for DATE\_SUB() DATE/TIME Function in SQL**

Example पर 2016-05-26 इस Date पर 2 days को subtract करके date return की गयी है |

Source Code :

SELECT DATE\_SUB("2016-05-26", INTERVAL 2 DAY);

Output :

+----------------------------------------+

| DATE\_SUB("2016-05-26", INTERVAL 2 DAY) |

+----------------------------------------+

| 2016-05-24 |

+----------------------------------------+

**Another Example for DATE\_SUB() DATE/TIME Function in SQL**

Example पर 2016-05-26 इस Date पर 2 days को add करके date return की गयी है |

Source Code :

SELECT DATE\_SUB("2016-05-26", INTERVAL -2 DAY);

Output :

+-----------------------------------------+

| DATE\_SUB("2016-05-26", INTERVAL -2 DAY) |

+-----------------------------------------+

| 2016-05-28 |

+-----------------------------------------+

### SQL - DAY

DAY() DATE/TIME Function दिए गए date या datetime से सिर्फ day(1 to 31) को return करता है |

**Syntax for DAY() DATE/TIME Function in SQL**

DAY(date/datetime)

अगर invalid format में date या datetime दिया जाता है तो 'NULL' return होता है |

**Parameter :**

**date/datetime :** यहाँ पर date या datetime दिया जाता है | यहाँ पर निचे दिए हुए Format में date या datetime दिया जाता है |  
YYYY-MM-DD HH:MM:SS For Example, 2016-05-26 03:25:57  
YYYY-MM-DD For Example, 2016-05-26  
YY-MM-DD HH:MM:SS For Example, 16-05-26 03:25:57  
YY-MM-DD For Example, 16-05-26

DAY() और DAYOFMONTH() ये दोनों Functions एक जैसे है |

**Example for DAY() DATE/TIME Function in SQL**

Source Code :

SELECT DAY("2016-05-26 03:25:57");

Output :

+----------------------------+

| DAY("2016-05-26 03:25:57") |

+----------------------------+

| 26 |

+----------------------------+

### SQL - DAYNAME

DAYNAME() DATE/TIME Function दिए गए date या datetime से सिर्फ weekday name को return करता है |

**Syntax for DAYNAME() DATE/TIME Function in SQL**

DAYNAME(date/datetime)

अगर invalid format में date या datetime दिया जाता है तो 'NULL' return होता है |

**Parameter :**

**date/datetime :** यहाँ पर date या datetime दिया जाता है | यहाँ पर निचे दिए हुए Format में date या datetime दिया जाता है |  
YYYY-MM-DD HH:MM:SS For Example, 2016-05-26 03:25:57  
YYYY-MM-DD For Example, 2016-05-26  
YY-MM-DD HH:MM:SS For Example, 16-05-26 03:25:57  
YY-MM-DD For Example, 16-05-26

**Example for DAYNAME() DATE/TIME Function in SQL**

Source Code :

SELECT DAYNAME("2016-05-26 03:25:57");

Output :

+--------------------------------+

| DAYNAME("2016-05-26 03:25:57") |

+--------------------------------+

| Thursday |

+--------------------------------+

### SQL - DAYOFMONTH

DAYOFMONTH() DATE/TIME Function दिए गए date या datetime से सिर्फ month के day(1 to 31) को return करता है |

**Syntax for DAYOFMONTH() DATE/TIME Function in SQL**

DAYOFMONTH(date/datetime)

अगर invalid format में date या datetime दिया जाता है तो 'NULL' return होता है |

**Parameter :**

**date/datetime :** यहाँ पर date या datetime दिया जाता है | यहाँ पर निचे दिए हुए Format में date या datetime दिया जाता है |  
YYYY-MM-DD HH:MM:SS For Example, 2016-05-26 03:25:57  
YYYY-MM-DD For Example, 2016-05-26  
YY-MM-DD HH:MM:SS For Example, 16-05-26 03:25:57  
YY-MM-DD For Example, 16-05-26

DAY() और DAYOFMONTH() ये दोनों Functions एक जैसे है |

**Example for DAYOFMONTH() DATE/TIME Function in SQL**

Source Code :

SELECT DAYOFMONTH("2016-05-26 03:25:57");

Output :

+-----------------------------------+

| DAYOFMONTH("2016-05-26 03:25:57") |

+-----------------------------------+

| 26 |

+-----------------------------------+

### SQL - DAYOFWEEK

DAYOFWEEK() DATE/TIME Function दिए गए date या datetime से सिर्फ week number(1=Sunday to 7=Saturday) को return करता है |

**Syntax for DAYOFWEEK() DATE/TIME Function in SQL**

DAYOFWEEK(date/datetime)

अगर invalid format में date या datetime दिया जाता है तो 'NULL' return होता है |

**Parameter :**

**date/datetime :** यहाँ पर date या datetime दिया जाता है | यहाँ पर निचे दिए हुए Format में date या datetime दिया जाता है |  
YYYY-MM-DD HH:MM:SS For Example, 2016-05-26 03:25:57  
YYYY-MM-DD For Example, 2016-05-26  
YY-MM-DD HH:MM:SS For Example, 16-05-26 03:25:57  
YY-MM-DD For Example, 16-05-26

**Example for DAYOFWEEK() DATE/TIME Function in SQL**

Source Code :

SELECT DAYOFWEEK("2016-05-26 03:25:57");

Output :

+----------------------------------+

| DAYOFWEEK("2016-05-26 03:25:57") |

+----------------------------------+

| 5 |

+----------------------------------+

### SQL - DAYOFYEAR

DAYOFYEAR() DATE/TIME Function दिए गए date या datetime से year के day(1 to 366) को return करता है |

**Syntax for DAYOFYEAR() DATE/TIME Function in SQL**

DAYOFYEAR(date/datetime)

अगर invalid format में date या datetime दिया जाता है तो 'NULL' return होता है |

**Parameter :**

**date/datetime :** यहाँ पर date या datetime दिया जाता है | यहाँ पर निचे दिए हुए Format में date या datetime दिया जाता है |  
YYYY-MM-DD HH:MM:SS For Example, 2016-05-26 03:25:57  
YYYY-MM-DD For Example, 2016-05-26  
YY-MM-DD HH:MM:SS For Example, 16-05-26 03:25:57  
YY-MM-DD For Example, 16-05-26

**Example for DAYOFYEAR() DATE/TIME Function in SQL**

Source Code :

SELECT DAYOFYEAR("2016-05-26 03:25:57");

Output :

+----------------------------------+

| DAYOFYEAR("2016-05-26 03:25:57") |

+----------------------------------+

| 147 |

+----------------------------------+

### SQL - EXTRACT

EXTRACT() DATE/TIME Function दिए गए date या datetime से दिए गए unit को return करता है |

**Syntax for EXTRACT() DATE/TIME Function in SQL**

EXTRACT(unit FROM date/datetime)

**Parameter :**

**unit :** unit पर निचे दी हुई values दी जा सकती है |   
MICROSECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER YEAR, SECOND\_MICROSECOND, MINUTE\_MICROSECOND, MINUTE\_SECOND, HOUR\_MICROSECOND, HOUR\_SECOND, HOUR\_MINUTE, DAY\_MICROSECOND, DAY\_SECOND, DAY\_MINUTE, DAY\_HOUR, YEAR\_MONTH

**date/datetime :** यहाँ पर date या datetime दिया जाता है |

अगर invalid format में date या datetime और invalid unit दिया जाता है तो 'NULL' return होता है |

अगर दिया हुआ unit; date पर नहीं होता है तो 'NULL' return होता है |

**Example for EXTRACT() DATE/TIME Function in SQL**

Source Code :

SELECT EXTRACT(YEAR FROM "2016-05-26 03:25:57");

Output :

+------------------------------------------+

| EXTRACT(YEAR FROM "2016-05-26 03:25:57") |

+------------------------------------------+

| 2016 |

+------------------------------------------+

**Another Example for EXTRACT() DATE/TIME Function in SQL**

Example पर MONTH ये unit दिया गया है लेकिन date पर वो unit मौजूद न होने के कारण 'NULL' return होता है |

Source Code :

SELECT EXTRACT(MONTH FROM "03:25:57");

Output :

+--------------------------------+

| EXTRACT(MONTH FROM "03:25:57") |

+--------------------------------+

| NULL |

+--------------------------------+

**Another Example for EXTRACT() DATE/TIME Function in SQL**

Source Code :

SELECT EXTRACT(MINUTE FROM "03:25:57");

Output :

+---------------------------------+

| EXTRACT(MINUTE FROM "03:25:57") |

+---------------------------------+

| 25 |

+---------------------------------+

### SQL - FROM DAYS

FROM\_DAYS() DATE/TIME Function दिए गए days को 'YYYY-MM-DD' इस format में convert करके date को return करता है |

**Syntax for FROM\_DAYS() DATE/TIME Function in SQL**

FROM\_DAYS(days)

**Parameter :**

**days :** यहाँ पर days की संख्या दी जाती है |

FROM\_DAYS() और TO\_DAYS() ये दोनों विरूद्ध Functions है |

**Example for FROM\_DAYS() DATE/TIME Function in SQL**

Source Code :

SELECT FROM\_DAYS(366);

Output :

+----------------+

| FROM\_DAYS(366) |

+----------------+

| 0001-01-01 |

+----------------+

**Another Example for FROM\_DAYS() DATE/TIME Function in SQL**

Example पर cuurent date को

Source Code :

SELECT FROM\_DAYS(736897);

Output :

+-------------------+

| FROM\_DAYS(736897) |

+-------------------+

| 2017-07-22 |

+-------------------+

### SQL - HOUR

HOUR() DATE/TIME Function दिए गए date से hour(0 to 23) को return करता है |

**Syntax for HOUR() DATE/TIME Function in SQL**

HOUR(date/datetime)

**Parameter :**

**date/datetime :** यहाँ पर date दी जाती है |

**Example for HOUR() DATE/TIME Function in SQL**

Source Code :

SELECT HOUR("2016-05-26 03:25:57");

Output :

+-----------------------------+

| HOUR("2016-05-26 03:25:57") |

+-----------------------------+

| 3 |

+-----------------------------+

### SQL - LAST DAY

LAST\_DAY() DATE/TIME Function दिए गए date के month के आखिर के day को return करता है |

**Syntax for LAST\_DAY() DATE/TIME Function in SQL**

LAST\_DAY(date/datetime)

**Parameter :**

**date/datetime :** यहाँ पर date दी जाती है |

**Example for LAST\_DAY() DATE/TIME Function in SQL**

Source Code :

SELECT LAST\_DAY("2016-05-26");

Output :

+------------------------+

| LAST\_DAY("2016-05-26") |

+------------------------+

| 2016-05-31 |

+------------------------+

### SQL - LOCALTIME

LOCALTIME() DATE/TIME Function current localtime मतलब date और time को return करता है |

**Syntax for LOCALTIME() DATE/TIME Function in SQL**

LOCALTIME()

**Parameter :**

LOCALTIME() Function के लिए कोई parameter नहीं होता है |

Return होनेवाला datetime 'YYYY-MM-DD HH:MM:SS' इस format में return होती है |

NOW(), LOCALTIME(), LOCALTIMESTAMP() और CURRENT\_TIMESTAMP() ये सभी Functions एक जैसे है |

**Example for LOCALTIME() DATE/TIME Function in SQL**

Example पर Current time को return किया गया है |

Source Code :

SELECT LOCALTIME();

Output :

+---------------------+

| LOCALTIME() |

+---------------------+

| 2017-07-22 11:22:24 |

+---------------------+

### SQL - LOCALTIMESTAMP

LOCALTIMESTAMP() DATE/TIME Function current local timestamp मतलब date और time को return करता है |

**Syntax for LOCALTIMESTAMP() DATE/TIME Function in SQL**

LOCALTIMESTAMP()

**Parameter :**

LOCALTIMESTAMP() Function के लिए कोई parameter नहीं होता है |

Return होनेवाला datetime 'YYYY-MM-DD HH:MM:SS' इस format में return होती है |

NOW(), LOCALTIME(), LOCALTIMESTAMP() और CURRENT\_TIMESTAMP() ये सभी Functions एक जैसे है |

**Example for LOCALTIMESTAMP() DATE/TIME Function in SQL**

Example पर Current time को return किया गया है |

Source Code :

SELECT LOCALTIMESTAMP();

Output :

+---------------------+

| LOCALTIMESTAMP() |

+---------------------+

| 2017-07-22 11:22:24 |

+---------------------+

### SQL - MICROSECOND

MICROSECOND() DATE/TIME Function दिए गए date से microseconds(0 to 999999) को return करता है |

**Syntax for MICROSECOND() DATE/TIME Function in SQL**

MICROSECOND(date/datetime)

**Parameter :**

**date/datetime :** यहाँ पर date दी जाती है |

अगर date पर microseconds दिए नहीं जाते है तो 0 return होता है |

**Example for MICROSECOND() DATE/TIME Function in SQL**

Source Code :

SELECT MICROSECOND("2016-05-26 03:25:57.000756");

Output :

+-------------------------------------------+

| MICROSECOND("2016-05-26 03:25:57.000756") |

+-------------------------------------------+

| 756 |

+-------------------------------------------+

**Another Example for MICROSECOND() DATE/TIME Function in SQL**

Source Code :

SELECT MICROSECOND("2016-05-26 03:25:57");

Output :

+------------------------------------+

| MICROSECOND("2016-05-26 03:25:57") |

+------------------------------------+

| 0 |

+------------------------------------+

### SQL - MINUTE

MINUTE() DATE/TIME Function दिए गए date से minute(0 to 59) को return करता है |

**Syntax for MINUTE() DATE/TIME Function in SQL**

MINUTE(date/datetime)

**Parameter :**

**date/datetime :** यहाँ पर date दी जाती है |

**Example for MINUTE() DATE/TIME Function in SQL**

Source Code :

SELECT MINUTE("2016-05-26 03:25:57");

Output :

+-------------------------------+

| MINUTE("2016-05-26 03:25:57") |

+-------------------------------+

| 25 |

+-------------------------------+

**Another Example for MINUTE() DATE/TIME Function in SQL**

Source Code :

SELECT MINUTE("03:25:57");

Output :

+--------------------+

| MINUTE("03:25:57") |

+--------------------+

| 25 |

+--------------------+

### SQL - MONTH

MONTH() DATE/TIME Function दिए गए date से month(1 to 12) को return करता है |

**Syntax for MONTH() DATE/TIME Function in SQL**

MONTH(date/datetime)

**Parameter :**

**date/datetime :** यहाँ पर date दी जाती है |

**Example for MONTH() DATE/TIME Function in SQL**

Source Code :

SELECT MONTH("2016-05-26 03:25:57");

Output :

+------------------------------+

| MONTH("2016-05-26 03:25:57") |

+------------------------------+

| 5 |

+------------------------------+

### SQL - MONTHNAME

MONTHNAME() DATE/TIME Function दिए गए date से month name को return करता है |

**Syntax for MONTHNAME() DATE/TIME Function in SQL**

MONTHNAME(date/datetime)

**Parameter :**

**date/datetime :** यहाँ पर date दी जाती है |

**Example for MONTHNAME() DATE/TIME Function in SQL**

Source Code :

SELECT MONTHNAME("2016-05-26 03:25:57");

Output :

+----------------------------------+

| MONTHNAME("2016-05-26 03:25:57") |

+----------------------------------+

| May |

+----------------------------------+

### SQL - NOW

NOW() DATE/TIME Function current date और time को return करता है |

**Syntax for NOW() DATE/TIME Function in SQL**

NOW()

**Parameter :**

NOW() Function के लिए कोई parameter नहीं होता है |

Return होनेवाला datetime 'YYYY-MM-DD HH:MM:SS' इस format में return होती है |

NOW(), LOCALTIME(), LOCALTIMESTAMP() और CURRENT\_TIMESTAMP() ये सभी Functions एक जैसे है |

**Example for NOW() DATE/TIME Function in SQL**

Example पर Current time को return किया गया है |

Source Code :

SELECT NOW();

Output :

+---------------------+

| NOW() |

+---------------------+

| 2017-07-22 11:22:24 |

+---------------------+

### SQL - PERIOD ADD

PERIOD\_ADD() DATE/TIME Function दिए गए period पर month की संख्या को add करके period को return करता है |

**Syntax for PERIOD\_ADD() DATE/TIME Function in SQL**

PERIOD\_ADD(period, numberOfMonth)

**Parameter :**

**period :** यहाँ पर YYMM या YYYYMM इस format में period दिया जाता है |

**numberOfMonth :** यहाँ पर month की संख्या को दिया जाता है | इन month की संख्या को period पर add किया जाता है | अगर negative; month की संख्या दी जाती है तो period से month को subtract किया जाता है |

**Example for PERIOD\_ADD() DATE/TIME Function in SQL**

Source Code :

SELECT PERIOD\_ADD("201606",1);

Output :

+------------------------+

| PERIOD\_ADD("201606",1) |

+------------------------+

| 201607 |

+------------------------+

**Example for PERIOD\_ADD() DATE/TIME Function in SQL**

Source Code :

SELECT PERIOD\_ADD("201606",-1);

Output :

+------------------------+

| PERIOD\_ADD("201606",-1)|

+------------------------+

| 201605 |

+------------------------+

### SQL - PERIOD DIFF

PERIOD\_DIFF() DATE/TIME Function दिए गए periods के बीच का differrence; negative या positive month की संख्या में return करता है |

**Syntax for PERIOD\_DIFF() DATE/TIME Function in SQL**

PERIOD\_DIFF(period1, period2)

**Parameter :**

**period1 :** यहाँ पर YYMM या YYYYMM इस format में period दिया जाता है |

**period2 :** यहाँ पर YYMM या YYYYMM इस format में period दिया जाता है |

अगर period1 ये period2 से छोटा होता है तो negative month return होता है |

अगर period1 ये period2 से बड़ा होता है तो positive month return होता है |

**Example for PERIOD\_DIFF() DATE/TIME Function in SQL**

Source Code :

SELECT PERIOD\_DIFF("201606",1);

Output :

+------------------------+

| PERIOD\_DIFF("201606",1) |

+------------------------+

| 201607 |

+------------------------+

**Example for PERIOD\_DIFF() DATE/TIME Function in SQL**

Source Code :

SELECT PERIOD\_DIFF("201606",-1);

Output :

+------------------------+

| PERIOD\_DIFF("201606",-1)|

+------------------------+

| 201605 |

+------------------------+

### SQL - QUARTER

QUARTER() DATE/TIME Function दिए गए date के quarter(1 to 4) को return करता है |

अगर date 01 JAN से 31 MAR के बीच की हो तो 1 return होता है |

अगर date 01 APR से 30 JUN के बीच की हो तो 2 return होता है |

अगर date 01 JUL से 30 SEP के बीच की हो तो 3 return होता है |

अगर date 01 OCT से 31 DEC के बीच की हो तो 4 return होता है |

**Syntax for QUARTER() DATE/TIME Function in SQL**

QUARTER(date)

**Parameter :**

**date :** यहाँ पर date दी जाती है |

**Example for QUARTER() DATE/TIME Function in SQL**

Source Code :

SELECT QUARTER("2016-12-30");

Output :

+-----------------------+

| QUARTER("2016-12-30") |

+-----------------------+

| 4 |

+-----------------------+

### SQL - SECOND

SECOND() DATE/TIME Function दिए गए date से second(0 to 59) को return करता है |

**Syntax for SECOND() DATE/TIME Function in SQL**

SECOND(date/datetime)

**Parameter :**

**date/datetime :** यहाँ पर date दी जाती है |

**Example for SECOND() DATE/TIME Function in SQL**

Source Code :

SELECT SECOND("2016-05-26 03:25:57");

Output :

+-------------------------------+

| SECOND("2016-05-26 03:25:57") |

+-------------------------------+

| 57 |

+-------------------------------+

**Another Example for SECOND() DATE/TIME Function in SQL**

Source Code :

SELECT SECOND("03:25:57");

Output :

+--------------------+

| SECOND("03:25:57") |

+--------------------+

| 57 |

+--------------------+

### SQL - SEC TO TIME

SEC\_TO\_TIME() DATE/TIME Function दिए गए seconds को time में convert करके return करता है |

**Syntax for SEC\_TO\_TIME() DATE/TIME Function in SQL**

SEC\_TO\_TIME(second)

**Parameter :**

**second :** यहाँ पर seconds की संख्या दी जाती है |

**Example for SEC\_TO\_TIME() DATE/TIME Function in SQL**

Source Code :

SELECT SEC\_TO\_TIME(120);

Output :

+------------------+

| SEC\_TO\_TIME(120) |

+------------------+

| 00:02:00 |

+------------------+

### SQL - STR TO DATE

STR\_TO\_DATE() DATE/TIME Function दिए गए date या datetime से दिए गए format जैसे return करता है |

**Syntax for STR\_TO\_DATE() DATE/TIME Function in SQL**

STR\_TO\_DATE(date/datetime, format)

अगर invalid format में date या datetime दिया जाता है तो 'NULL' return होता है |

**Parameter :**

**date/datetime :** यहाँ पर date या datetime दिया जाता है | यहाँ पर निचे दिए हुए Format में date या datetime दिया जाता है |  
YYYY-MM-DD HH:MM:SS For Example, 2016-05-26 03:25:57  
YYYY-MM-DD For Example, 2016-05-26  
YY-MM-DD HH:MM:SS For Example, 16-05-26 03:25:57  
YY-MM-DD For Example, 16-05-26

**format :** निचे सभी formats दिए गए है | format में दिए हुए सभी examples '2016-05-26 03:25:57 AM' इस datetime के अनुसार दिए गए है |

|  |  |  |
| --- | --- | --- |
| **Directive** | **Description** | **Example** |
| %a | abbreviated weekday name | Thu |
| %b | abbreviated month name | May |
| %c | month in number | 5 |
| %d | day of the month(01 to 31) | 26 |
| %D | day of the month(1 to 31) with suffix | 26th |
| %e | day of the month(01 to 31) same as %d | 26 |
| %f | Microseconds((000000 to 999999)) | 000000 |
| %h | hour(12-hour clock)(01 to 12) | 03 |
| %H | hour(24-hour clock)(00 to 23) | 03 |
| %i | Minutes(00 to 59) | 25 |
| %I | hour(12-hour clock)(01 to 12) Same as %h | 03 |
| %j | day of the year(001 to 366) | 147 |
| %k | hour(24-hour clock)(0 to 23) | 3 |
| %l | hour(12-hour clock)(1 to 12) | 3 |
| %m | month in number(01 to 12) | 05 |
| %M | full month name | May |
| %p | AM or PM according to time | AM |
| %r | only time with (12-hour clock)AM/PM | 03:25:57 AM |
| %S | second in number(00 to 59) | 57 |
| %s | second in number(00 to 59) | 57 |
| %T | full time(24-hour clock) without AM/PM | 03:25:57 |
| %U | week number of the year(00 to 53)(sunday is the first day of the week) | 21 |
| %u | week number of the year(00 to 53)(monday is the first day of the week) | 21 |
| %V | week number of the year(01 to 53)(sunday is the first day of the week) | 21 |
| %v | week number of the year(01 to 53)(monday is the first day of the week) | 21 |
| %w | weekday in number(0 to 6)(sunday is the first day of the week) | 4 |
| %W | weekday full name | Thursday |
| %x | Same as %v | 21 |
| %X | Same as %V | 21 |
| %y | two digit year | 16 |
| %Y | four digit year | 2016 |

STR\_TO\_DATE() और DATE\_FORMAT() ये दोनों Functions एक जैसे है |

**Example for STR\_TO\_DATE() DATE/TIME Function in SQL**

Source Code :

SELECT STR\_TO\_DATE("2016-05-26 03:25:57 AM", "%Y-%m-%d %l:%i:%S %p");

Output :

+---------------------------------------------------------------+

| STR\_TO\_DATE("2016-05-26 03:25:57 AM", "%Y-%m-%d %l:%i:%S %p") |

+---------------------------------------------------------------+

| 2016-05-26 03:25:57 |

+---------------------------------------------------------------+

### SQL - SUBDATE

SUBDATE() DATE/TIME Function दिए गए date या time और interval के हिसाब से date को return करता है |

**Syntax for SUBDATE() DATE/TIME Function in SQL**

SUBDATE(date, INTERVAL exp unit)

**Parameter :**

**date :** यहाँ पर निचे दिए हुए format जैसी date दी जाती है |  
YYYY-MM-DD HH:MM:SS For Example, 2016-05-26 03:25:57  
YYYY-MM-DD For Example, 2016-05-26  
YY-MM-DD HH:MM:SS For Example, 16-05-26 03:25:57   
YY-MM-DD For Example, 16-05-26

**INTERVAL exp :** INTERVAL keyword के साथ expression पर unit के मुताबिक positive(subtract date) या negative(add date) value दी जाती है |

**unit :** unit पर निचे दी हुई values दी जा सकती है |   
MICROSECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER YEAR, SECOND\_MICROSECOND, MINUTE\_MICROSECOND, MINUTE\_SECOND, HOUR\_MICROSECOND, HOUR\_SECOND, HOUR\_MINUTE, DAY\_MICROSECOND, DAY\_SECOND, DAY\_MINUTE, DAY\_HOUR, YEAR\_MONTH

SUBDATE() और DATE\_ADD() ये दोनों Functions विरुद्ध होते है |

SUBDATE() और SUBDATE() ये दोनों functions एक जैसे है |

**Example for SUBDATE() DATE/TIME Function in SQL**

Example पर 2016-05-26 03:25:57 इस Date/Time पर 1 second को subtract करके date return की गयी है |

Source Code :

SELECT SUBDATE("2016-05-26 03:25:57", INTERVAL 1 SECOND);

Output :

+----------------------------------------------------+

| SUBDATE("2016-05-26 03:25:57", INTERVAL 1 SECOND) |

+----------------------------------------------------+

| 2016-05-26 03:25:56 |

+----------------------------------------------------+

**Another Example for SUBDATE() DATE/TIME Function in SQL**

Example पर 2016-05-26 इस Date पर 2 days को subtract करके date return की गयी है |

Source Code :

SELECT SUBDATE("2016-05-26", INTERVAL 2 DAY);

Output :

+----------------------------------------+

| SUBDATE("2016-05-26", INTERVAL 2 DAY) |

+----------------------------------------+

| 2016-05-24 |

+----------------------------------------+

**Another Example for SUBDATE() DATE/TIME Function in SQL**

Example पर 2016-05-26 इस Date पर 2 days को add करके date return की गयी है |

Source Code :

SELECT SUBDATE("2016-05-26", INTERVAL -2 DAY);

Output :

+-----------------------------------------+

| SUBDATE("2016-05-26", INTERVAL -2 DAY) |

+-----------------------------------------+

| 2016-05-28 |

+-----------------------------------------+

### SQL - SUBTIME

SUBTIME() DATE/TIME Function दिए गए date या datetime से time को subtract करके datetime return करता है |

**Syntax for SUBTIME() DATE/TIME Function in SQL**

SUBTIME(date/datetime, time)

**Parameter :**

**date/datetime :** यहाँ पर date दी जाती है |

**time :** यहाँ पर time दिया जाता है | अगर positive दिया जाता है तो subtract किया जाता है और अगर negative दिया जाता है तो add किया जाता है |

**Example for SUBTIME() DATE/TIME Function in SQL**

Example पर दिए गए datetime से 1 second को subtract किया गया है |

Source Code :

SELECT SUBTIME("2016-05-26 03:25:57", "1");

Output :

+-------------------------------------+

| SUBTIME("2016-05-26 03:25:57", "1") |

+-------------------------------------+

| 2016-05-26 03:25:56 |

+-------------------------------------+

**Another Example for SUBTIME() DATE/TIME Function in SQL**

Example पर दिए गए datetime में 1 second को add किया गया है |

Source Code :

SELECT SUBTIME("2016-05-26 03:25:57", "-1");

Output :

+--------------------------------------+

| SUBTIME("2016-05-26 03:25:57", "-1") |

+--------------------------------------+

| 2016-05-26 03:25:58 |

+--------------------------------------+

### SQL - SYSDATE

SYSDATE() DATE/TIME Function system की current date और time को return करता है |

**Syntax for SYSDATE() DATE/TIME Function in SQL**

SYSDATE()

**Parameter :**

SYSDATE() इस Function के लिए कोई parameter नहीं होता है |

**Example for SYSDATE() DATE/TIME Function in SQL**

Source Code :

SELECT SYSDATE();

Output :

+---------------------+

| SYSDATE() |

+---------------------+

| 2017-07-22 17:05:49 |

+---------------------+

### SQL - TIME

TIME() DATE/TIME Function दिए गए datetime से time को return करता है |

**Syntax for TIME() DATE/TIME Function in SQL**

TIME(datetime)

**Parameter :**

**datetime :** यहाँ पर datetime दिया जाता है |

**Example for TIME() DATE/TIME Function in SQL**

Source Code :

SELECT TIME("2016-05-26 03:25:57");

Output :

+-----------------------------+

| TIME("2016-05-26 03:25:57") |

+-----------------------------+

| 03:25:57 |

+-----------------------------+

### SQL - TIME FORMAT

TIME\_FORMAT() DATE/TIME Function दिए गए date या datetime से दिए गए format जैसे return करता है |

**Syntax for TIME\_FORMAT() DATE/TIME Function in SQL**

TIME\_FORMAT(date/datetime, format)

**Parameter :**

**date/datetime :** यहाँ पर date या datetime दिया जाता है | यहाँ पर निचे दिए हुए Format में date या datetime दिया जाता है |  
YYYY-MM-DD HH:MM:SS For Example, 2016-05-26 03:25:57  
YYYY-MM-DD For Example, 2016-05-26  
YY-MM-DD HH:MM:SS For Example, 16-05-26 03:25:57  
YY-MM-DD For Example, 16-05-26

**format :** निचे सभी formats दिए गए है | format में दिए हुए सभी examples '2016-05-26 03:25:57 AM' इस datetime के अनुसार दिए गए है |

|  |  |  |
| --- | --- | --- |
| **Directive** | **Description** | **Example** |
| %a | abbreviated weekday name | Thu |
| %b | abbreviated month name | May |
| %c | month in number | 5 |
| %d | day of the month(01 to 31) | 26 |
| %D | day of the month(1 to 31) with suffix | 26th |
| %e | day of the month(01 to 31) same as %d | 26 |
| %f | Microseconds((000000 to 999999)) | 000000 |
| %h | hour(12-hour clock)(01 to 12) | 03 |
| %H | hour(24-hour clock)(00 to 23) | 03 |
| %i | Minutes(00 to 59) | 25 |
| %I | hour(12-hour clock)(01 to 12) Same as %h | 03 |
| %j | day of the year(001 to 366) | 147 |
| %k | hour(24-hour clock)(0 to 23) | 3 |
| %l | hour(12-hour clock)(1 to 12) | 3 |
| %m | month in number(01 to 12) | 05 |
| %M | full month name | May |
| %p | AM or PM according to time | AM |
| %r | only time with (12-hour clock)AM/PM | 03:25:57 AM |
| %S | second in number(00 to 59) | 57 |
| %s | second in number(00 to 59) | 57 |
| %T | full time(24-hour clock) without AM/PM | 03:25:57 |
| %U | week number of the year(00 to 53)(sunday is the first day of the week) | 21 |
| %u | week number of the year(00 to 53)(monday is the first day of the week) | 21 |
| %V | week number of the year(01 to 53)(sunday is the first day of the week) | 21 |
| %v | week number of the year(01 to 53)(monday is the first day of the week) | 21 |
| %w | weekday in number(0 to 6)(sunday is the first day of the week) | 4 |
| %W | weekday full name | Thursday |
| %x | Same as %v | 21 |
| %X | Same as %V | 21 |
| %y | two digit year | 16 |
| %Y | four digit year | 2016 |

TIME\_FORMAT() और DATE\_FORMAT() ये दोनों एक जैसे functions है |

**Example for TIME\_FORMAT() DATE/TIME Function in SQL**

Source Code :

SELECT TIME\_FORMAT("2016-05-26 03:25:57 AM", "%Y-%m-%d %l:%i:%S %p");

Output :

+---------------------------------------------------------------+

| TIME\_FORMAT("2016-05-26 03:25:57 AM", "%Y-%m-%d %l:%i:%S %p") |

+---------------------------------------------------------------+

| 2016-05-26 3:25:57 AM |

+---------------------------------------------------------------+

### SQL - TIME TO SEC

TIME\_TO\_SEC() DATE/TIME Function दिए गए time को seconds में convert करके return करता है |

**Syntax for TIME\_TO\_SEC() DATE/TIME Function in SQL**

TIME\_TO\_SEC(time)

**Parameter :**

**time :** यहाँ पर time दिया जाता है |

**Example for TIME\_TO\_SEC() DATE/TIME Function in SQL**

Source Code :

SELECT TIME\_TO\_SEC("03:25:57");

Output :

+-------------------------+

| TIME\_TO\_SEC("03:25:57") |

+-------------------------+

| 12357 |

+-------------------------+

**Another Example for TIME\_TO\_SEC() DATE/TIME Function in SQL**

Source Code :

SELECT TIME\_TO\_SEC("-03:25:57");

Output :

+--------------------------+

| TIME\_TO\_SEC("-03:25:57") |

+--------------------------+

| -12357 |

+--------------------------+

### SQL - TIMEDIFF

TIMEDIFF() DATE/TIME Function दिए गए दो time के बीच का differrencetime में return करता है |

**Syntax for TIMEDIFF() DATE/TIME Function in SQL**

TIMEDIFF(time1, time2)

**Parameter :**

**time1 :** यहाँ पर time दिया जाता है |

**time2 :** यहाँ पर time दिया जाता है |

अगर time1 ये time2 से छोटा होता है तो negative time को return किया जाता है |

अगर time1 ये time2 से बड़ा होता है तो positive time को return किया जाता है |

**Example for TIMEDIFF() DATE/TIME Function in SQL**

Source Code :

SELECT TIMEDIFF("04:25:58","03:25:58");

Output :

+---------------------------------+

| TIMEDIFF("04:25:58","03:25:58") |

+---------------------------------+

| 01:00:00 |

+---------------------------------+

**Another Example for TIMEDIFF() DATE/TIME Function in SQL**

Source Code :

SELECT TIMEDIFF("03:25:58","04:25:58");

Output :

+---------------------------------+

| TIMEDIFF("03:25:58","04:25:58") |

+---------------------------------+

| -01:00:00 |

+---------------------------------+

### SQL - TIMESTAMP

TIMESTAMP() DATE/TIME Function दिए गए date या datetime को datetime में convert करके return करता है |

**Syntax for TIMESTAMP() DATE/TIME Function in SQL**

TIMESTAMP(exp)

**Parameter :**

**exp :** यहाँ पर date या datetime दिया जाता है |

**Example for TIMESTAMP() DATE/TIME Function in SQL**

Source Code :

SELECT TIMESTAMP("2016-05-26");

Output :

+-------------------------+

| TIMESTAMP("2016-05-26") |

+-------------------------+

| 2016-05-26 00:00:00 |

+-------------------------+

### SQL - TO DAYS

TO\_DAYS() DATE/TIME Function दिए गए date को days में convert करके return करता है |

**Syntax for TO\_DAYS() DATE/TIME Function in SQL**

TO\_DAYS(date/datetime)

**Parameter :**

**date/datetime :** यहाँ पर date दी जाती है |

TO\_DAYS() और TO\_DAYS() ये दोनों विरूद्ध Functions है |

**Example for TO\_DAYS() DATE/TIME Function in SQL**

Source Code :

SELECT TO\_DAYS("0001-01-01");

Output :

+-----------------------+

| TO\_DAYS("0001-01-01") |

+-----------------------+

| 366 |

+-----------------------+

**Another Example for TO\_DAYS() DATE/TIME Function in SQL**

Example पर cuurent date को

Source Code :

SELECT TO\_DAYS("2017-07-22");

Output :

+-----------------------+

| TO\_DAYS("2017-07-22") |

+-----------------------+

| 736897 |

+-----------------------+

### SQL - WEEK

WEEK() DATE/TIME Function दिए गए date के year का week return करता है |

**Syntax for WEEK() DATE/TIME Function in SQL**

WEEK(date)

**Parameter :**

**date :** यहाँ पर date दी जाती है |

WEEK() और WEEKOFYEAR() ये दोनों एक जैसे functions है |

**Example for WEEK() DATE/TIME Function in SQL**

Source Code :

SELECT WEEK("2016-05-26");

Output :

+--------------------+

| WEEK("2016-05-26") |

+--------------------+

| 21 |

+--------------------+

### SQL - WEEKDAY

WEEKDAY() DATE/TIME Function दिए गए date के weekday को number में return करता है |

**Syntax for WEEKDAY() DATE/TIME Function in SQL**

WEEKDAY(date)

**Parameter :**

**date :** यहाँ पर date दी जाती है |

**Example for WEEKDAY() DATE/TIME Function in SQL**

Source Code :

SELECT WEEKDAY("2016-05-26");

Output :

+-----------------------+

| WEEKDAY("2016-05-26") |

+-----------------------+

| 3 |

+-----------------------+

### SQL - WEEKOFYEAR

WEEKOFYEAR() DATE/TIME Function दिए गए date के year का week return करता है |

**Syntax for WEEKOFYEAR() DATE/TIME Function in SQL**

WEEKOFYEAR(date)

**Parameter :**

**date :** यहाँ पर date दी जाती है |

WEEK() और WEEKOFYEAR() ये दोनों एक जैसे functions है |

**Example for WEEKOFYEAR() DATE/TIME Function in SQL**

Source Code :

SELECT WEEKOFYEAR("2016-05-26");

Output :

+--------------------------+

| WEEKOFYEAR("2016-05-26") |

+--------------------------+

| 21 |

+--------------------------+

### SQL - YEAR

YEAR() DATE/TIME Function दिए गए date से year को return करता है |

**Syntax for YEAR() DATE/TIME Function in SQL**

YEAR(date)

**Parameter :**

**date :** यहाँ पर date दी जाती है |

**Example for YEAR() DATE/TIME Function in SQL**

Source Code :

SELECT YEAR("2016-05-26");

Output :

+--------------------+

| YEAR("2016-05-26") |

+--------------------+

| 2016 |

+--------------------+

### SQL - YEARWEEK

YEARWEEK() DATE/TIME Function दिए गए date से year और week को return करता है |

**Syntax for YEARWEEK() DATE/TIME Function in SQL**

YEARWEEK(date)

**Parameter :**

**date :** यहाँ पर date दी जाती है |

**Example for YEARWEEK() DATE/TIME Function in SQL**

Example के 201621 इस output पर 2016 ये year है और 21 ये year का week है |

Source Code :

SELECT YEARWEEK("2016-05-26");

Output :

+------------------------+

| YEARWEEK("2016-05-26") |

+------------------------+

| 201621 |

+------------------------+