

# **SQL COMMANDS**

**ATUL KUMAR ([LINKEDIN](#))**

## CONTENTS

- 1.FUNDAMENTALS OF SQL**
- 2.FILTERING COMMANDS**
- 3.ORDERING COMMANDS**
- 4.ALIAS**
- 5.AGGREGATE COMMANDS**
- 6.GROUP BY COMMANDS**
- 7.CONDITIONAL STATEMENT**
- 8.JOINS**
- 9.SUBQUERY**
- 10.VIEW & INDEX**
- 11.STRING FUNCTIONS**
- 12.MATHEMATICAL FUNCTIONS**
- 13. DATE-TIME FUNCTIONS**
- 14.PATTERN MATCHING(regex)**
- 15.DATA TYPE CONVERSION FUNCTIONS**

DQL(Data Query Language)

To fetch the data from the database

Example: SELECT

DML(Data Manipulation Language)-

To modify the database objects

Example: INSERT,UPDATE,DELETE

DDL(Data Definition Language)

To create & modify database  
objects

Example: CREATE,DROP,ALTER,TRUNCATE

## 1. FUNDAMENTAL OF SQL

### CREATE

CREATE statement is used to create a table

### Syntax:

```
CREATE TABLE "TABLE_NAME"(  
    "COLUMN1"  "DATA_TYPE"  CONSTRAINTS,  
    "COLUMN2"  "DATA_TYPE"  CONSTRAINTS,  
    "COLUMN3"  "DATA_TYPE"  CONSTRAINTS,  
    .....  
    "COLUMN N"  "DATA_TYPE"  CONSTRAINTS  
);
```

### INSERT

INSERT statement is used insert new data into the table

## Syntax:

```
INSERT INTO  
"TABLE_NAME" (COL1, COL2, .....COL_N)  
VALUES (Col_val_1, Col_val_2, ..... Col_val_N);
```

## Import data from file(PostgreSQL)

### For csv file

```
COPY TABLE_NAME(column1, column2,... ) FROM  
FILE_PATH DELIMITER ',' CSV HEADER;
```

### For txt file

```
COPY TABLE_NAME(column1, column2,... ) FROM  
FILE_PATH DELIMITER ',';
```

## SELECT

SELECT statement is used to retrieve data from the table

## Syntax

```
SELECT * FROM "TABLE_NAME";
```

For select one column

```
SELECT "COLUMN_NAME" FROM "TABLE_NAME";
```

For select multiple columns

```
SELECT "COLUMN1,COLUMN2..."  
FROM "TABLE_NAME";
```

For select all columns

```
SELECT * FROM "TABLE_NAME";
```

## DISTINCT

DISTINCT keyword is used to eliminate all duplicate records & fetch only unique records

## Syntax:

```
SELECT DISTINCT(*) FROM "TABLE_NAME";
```

## WHERE

WHERE clause is used to filter a records

## Syntax:

```
SELECT "COLUMN_NAME(S)"  
FROM "TABLE_NAME"
```

WHERE CONDITION;

AND/OR

The AND/OR is used to combine multiple conditions

Syntax:

```
SELECT "COLUMN_NAMES(s)"  
FROM "TABLE_NAME"  
WHERE CONDITION AND/OR CONDITION;
```

UPDATE

It is used to modify the existing data in the table

Syntax:

```
UPDATE "TABLE_NAME"  
SET COL_1=VAL_1,COL_2=VAL_2,...  
WHERE CONDITION;
```

DELETE

It is used to delete existing records in the table

## Syntax:

For delete all rows

```
DELETE FROM "TABLE_NAME";
```

For delete single/multiple row(s)

```
DELETE FROM "TABLE_NAME"  
WHERE CONDITION;
```

## ALTER

It is used to change the definition or structure of the table

### Syntax:

#### ADD COLUMN

```
ALTER TABLE "TABLE_NAME"  
ADD "COLUMN_NAME" "DATA_TYPE";
```

#### DROP COLUMN

```
ALTER TABLE "TABLE_NAME"  
DROP "COLUMN_NAME";
```

#### MODIFY DATA TYPE

```
ALTER TABLE "TABLE_NAME"  
ALTER COLUMN "COL_NAME" TYPE
```

NEW\_DATA\_TYPE;

### RENAME COLUMN

ALTER TABLE "TABLE\_NAME"

RENAME COLUMN "COL\_NAME" TO "NEW\_NAME";

### ADD CONSTRAINTS

ALTER TABLE "TABLE\_NAME"

ADD CONSTRAINT COL\_NAME CHECK  
CONDITION;

## 2. FILTERING COMMANDS

### IN

Used to reduce multiple OR logical operator in  
SELECT,DELETE,INSERT & UPDATE statements

### Syntax:

SELECT "COL\_NAME" FROM "TABLE\_NAME"

WHERE "COL\_NAME" IN ('VAL1', 'VAL2',...);

### BETWEEN

Used to retrieve data within a given range

### Syntax:

```
SELECT "COL_NAME(S)" FROM "TABLE_NAME"  
WHERE "COL_NAME" BETWEEN "VAL1" AND  
"VAL2";
```

LIKE

Used to perform pattern matching/regex using wildcards(%, \_)

% - match any string of any length

\_ - match on a single character

### Syntax:

```
SELECT "COL_NAME" FROM "TABLE_NAME"  
WHERE "COL_NAME" LIKE 'PATTERN';
```

## 3. Ordering Commands

**ORDER BY**

Used to sort the data & it is only used in  
SELECT statement

**Syntax:**

```
SELECT "COL_NAME(s)" FROM "TABLE_NAME"  
ORDER BY "COL_NAME" ASC/DESC;
```

**LIMIT**

Used to limit the number of records based on a  
given limit

**Syntax:**

```
SELECT "COL_NAME(S)" FROM "TABLE_NAME"  
[WHERE & ORDER BY – Optional]  
LIMIT "LIMIT_VALUE";
```

## 4. ALIAS

**AS**

Used to assign an alias to the column

## Syntax:

```
SELECT "COL_NAME" as "COL_ALIAS"  
FROM "TABLE_NAME";
```

## 5.AGGREGATE COMMANDS

### COUNT

Used to count the expression

## Syntax:

```
SELECT COUNT(COL_NAME) FROM "TABLE_NAME";
```

### SUM

Used to sum the expression

## Syntax:

```
SELECT SUM(COL_NAME) FROM "TABLE_NAME";
```

AVG

Used to average the expression

Syntax:

```
SELECT AVG(COL_NAME) FROM "TABLE_NAME";
```

MIN

Used to retrieve the minimum value

Syntax:

```
SELECT MIN(COL_NAME) FROM "TABLE_NAME";
```

MAX

Used to retrieve the maximum value

Syntax:

```
SELECT MAX(COL_NAME) FROM "TABLE_NAME";
```

## 6.GROUP BY COMMANDS

### GROUP BY

GROUP BY clause is used to group the results by one or more columns

#### Syntax:

```
SELECT "COL_1", "COL_2",..... FROM  
"TABLE_NAME"  
  
GROUP BY "COL_NAME";
```

### HAVING

HAVING clause is added to SQL because the WHERE keyword cannot be used with aggregate functions

#### Syntax:

```
SELECT "COL_1", "COL_2",..... FROM  
"TABLE_NAME"  
  
GROUP BY "COL_NAME"  
  
HAVING 'CONDITION';
```

## 7. CONDITIONAL STATEMENT

CASE

CASE expression is a conditional expression

Syntax:

CASE

    WHEN CONDITION THEN RESULT

    [WHEN CONDITION THEN RESULT]

    [WHEN CONDITION THEN RESULT]

    ELSE RESULT

END

## 8.JOINS

JOINS used to fetch data from multiple tables

TYPES:

INNER JOIN

    INNER JOIN produces only the set of records that match in table A and table B

## Syntax:

```
SELECT COL1,COL2,....  
FROM "TABLE_1"  
INNER JOIN "TABLE_2"  
ON TABLE_1. COMMON_COL = TABLE_2.  
COMMON_COL;
```

## LEFT JOIN

LEFT JOIN returns all the rows in the table A(Left),even if there is no matches in the table B(Right)

## Syntax:

```
SELECT COL_1,COL_2,...  
FROM "TABLE_1"  
LEFT JOIN "TABLE_2"  
ON TABLE_1. COMMON_COL = TABLE_2.  
COMMON_COL;
```

## RIGHT JOIN

RIGHT JOIN returns all the rows in the table B(Right),even if there is no matches in the table

A(left)

### Syntax:

```
SELECT COL_1,COL_2,...  
FROM "TABLE_1"  
RIGHT JOIN "TABLE_2"  
ON TABLE_1.COMMON_COL = TABLE_2.  
COMMON_COL;
```

### **FULL JOIN**

FULL JOIN combines the results of both right & left join

### Syntax:

```
SELECT COL_1,COL_2,...  
FROM "TABLE_1"  
FULL JOIN "TABLE_2"  
ON TABLE_1.COMMON_COL = TABLE_2.  
COMMON_COL;
```

### **CROSS JOIN**

CROSS JOIN creates Cartesian product

between two sets

### Syntax:

```
SELECT TAB1.COL,TAB2.COL,....  
FROM "TABLE_1", "TABLE_2",.....
```

EXCEPT

Used to fetch all the data from table A except that matches with table B

### Syntax:

```
SELECT COL1,COL2,.....  
FROM TABLE_A [WHERE]  
EXCEPT  
SELECT COL_1,COL_2,.....  
FROM TABLE_B [WHERE];
```

UNION

Used to combine two or more SELECT statements

### Syntax:

```
SELECT COL1,COL2,.....  
FROM TABLE_A [WHERE]
```

UNION

SELECT COL\_1,COL\_2,.....

FROM TABLE\_B [WHERE];

SUBQUERY

SUBQUERY is a query within a query

Syntax:

SUBQUERY is in WHERE clause

SELECT "COL\_1" FROM "TABLE\_NAME\_1"

WHERE "COL\_2" [operator]

(SELECT "COL\_3" FROM "TABLE\_NAME\_2"

WHERE CONDITION);

VIEW

VIEW is a virtual table created by a query joining one or more tables

Syntax:

CREATE[OR RESPONSE] view\_name AS

SELECT "COL\_NAME(S)"

FROM "TABLE\_NAME"

## INDEX

An INDEX creates an entry for each value that appears in the indexed column

### Syntax:

```
CREATE[UNIQUE] INDEX "index_name"  
ON "TABLE_NAME"  
(index_col1 [ASC/DESC],.....)
```

## 11.STRING FUNCTIONS

### LENGTH:

LENGTH function retrieves the length of the specified string

### Syntax:

```
LENGTH('string')
```

### UPPER/LOWER

UPPER/LOWER function converts all the characters in the specified string to uppercase/lowercase

### Syntax:

`upper('string')`

`lower('string')`

## REPLACE

REPLACE function replaces all the occurrences of the specified string

### Syntax:

`REPLACE('string', 'from string', to string')`

## TRIM

TRIM function removes all specified characters either from beginning or end of the string or both

### Syntax:

`TRIM( [Leading|Trailing|Both] [trim char] from string)`

## RTRIM

RTRIM function removes all specified characters from RHS of the string

### Syntax:

`RTRIM('string', trim char)`

## LTRIM

LTRIM function removes all specified characters from LHS of the string

### Syntax:

LTRIM('string', trim char)

## CONCATENATION

|| operator used to concatenate two or more strings

### Syntax:

'string\_1' || 'string\_2' || 'string\_3'

## SUBSTRING

SUBSTRING function used to extract substring from a string

### Syntax:

SUBSTRING('string' [start position]  
[substring length]);

## STRING\_AGG

String aggregate function concatenates input values into a string, separated by a delimiter

### Syntax

STRING\_AGG('expression', delimiter)

## 12.MATHEMATICAL FUNCTIONS

### CEIL

CEIL function returns the smallest integer value which is greater than or equal to a number

#### Syntax:

CEIL(number)

### FLOOR

FLOOR function returns the largest integer value which is less than or equal to a number

#### Syntax:

FLOOR(number)

## RANDOM

RANDOM function used to generate random number between 0 & 1 (1 will be excluded)

### Syntax:

RANDOM( );

## SETSEED

SETSEED function used to set a seed for the next time that we call the RANDOM function

### Syntax:

SETSEED(seed)

[seed can have a value between 1 and -1(both are inclusive)]

## ROUND

ROUND function rounds a number to a specified number of decimal places

### Syntax:

ROUND(number)

## POWER

POWER function returns m raised to the nth power

Syntax:

POWER(m,n)

## 13. DATE-TIME FUNCTIONS

### CURRENT\_DATE

CURRENT\_DATE function returns the current date

Syntax:

CURRENT\_DATE()

### CURRENT\_TIME

CURRENT\_TIME function returns the current time with the time zone

Syntax:

CURRENT\_TIME()

## CURRENT\_TIMESTAMP

CURRENT\_TIMESTAMP function returns the current date & current time with the time zone

### Syntax:

CURRENT\_TIMESTAMP()

## AGE

AGE function returns the difference between two dates

### Syntax:

AGE(date1,date2)

## EXTRACT

EXTRACT function extract specified parts from date

### Syntax:

EXTRACT('unit' FROM 'date')

[unit will be day,month,year,doy,decade,hour,minute,second,etc..]

## 14. PATTERN MATCHING

There are three different approaches to pattern matching

- Using LIKE
- Using SIMILAR TO
- Using Regular Expression

- | denotes alternation (either of two alternatives).
- \* denotes repetition of the previous item zero or more times.
- + denotes repetition of the previous item one or more times.
- ? denotes repetition of the previous item zero or one time.
- {*m*} denotes repetition of the previous item exactly *m* times.
- {*m*,*n*} denotes repetition of the previous item *m* or more times.
- {*m,n*} denotes repetition of the previous item at least *m* and not more than *n* times.
- Parentheses () can be used to group items into a single logical item.
- A bracket expression [...] specifies a character class,

## 15. DATA TYPE CONVERSION FUNCTIONS

### TO\_CHAR

TO\_CHAR function converts number/date to String

Syntax:

**TO\_CHAR(value,format-mask)**

**TO\_DATE**

**TO\_DATE** function converts string to date

Syntax:

**TO\_DATE(string,format-mask)**

**TO\_NUMBER**

**TO\_NUMBER** function converts string to date

Syntax:

**TO\_NUMBER(string,format-mask)**

Format	Description
9	Numeric value with the specified number of digits
0	Numeric value with leading zeros
.	(period) decimal point
D	decimal point that uses locale
,	group (thousand) separator (comma)

Format	Description
FM	Fill mode, which suppresses padding blanks and leading zeroes.
PR	Negative value in angle brackets.
S	Sign anchored to a number that uses locale
L	Currency symbol that uses locale
G	Group separator that uses locale
MI	Minus sign in the specified position for numbers that are less than 0.
PL	Plus sign in the specified position for numbers that are greater than 0.
SG	Plus / minus sign in the specified position
RN	Roman numeral that ranges from 1 to 3999
TH or th	Upper case or lower case ordinal number suffix

Pattern	Description
Y,YYY	year in 4 digits with comma
YYYY	year in 4 digits
YY	last 2 digits of year
YYY	last 3 digits of year

Y	The last digit of year
IYYY	ISO 8601 week-numbering year (4 or more digits)
IYY	Last 3 digits of ISO 8601 week-numbering year
IY	Last 2 digits of ISO 8601 week-numbering year
I	Last digit of ISO 8601 week-numbering year
BC, bc, AD or ad	Era indicator without periods
B.C., b.c., A.D. ora.d.	Era indicator with periods
MONTH	English month name in uppercase
Month	Full capitalized English month name
Month	Full lowercase English month name
MON	Abbreviated uppercase month name e.g., JAN, FEB, etc.
Mon	Abbreviated capitalized month name e.g., Jan, Feb, etc.
Mon	Abbreviated lowercase month name e.g., jan, feb, etc.
MM	month number from 01 to 12
DAY	Full uppercase day name
Day	Full capitalized day name
Day	Full lowercase day name

DY	Abbreviated uppercase day name
Dy	Abbreviated capitalized day name
Dy	Abbreviated lowercase day name
DDD	Day of year (001-366)
IDDD	Day of ISO 8601 week-numbering year (001-371; day 1 of the year is Monday of the first ISO week)
DD	Day of month (01-31)
D	Day of the week, Sunday (1) to Saturday (7)
ID	ISO 8601 day of the week, Monday (1) to Sunday (7)
W	Week of month (1-5) (the first week starts on the first day of the month)
WW	Week number of year (1-53) (the first week starts on the first day of the year)
IW	Week number of ISO 8601 week-numbering year (01-53; the first Thursday of the year is in week 1)
CC	Century e.g, 21, 22, etc.
J	Julian Day (integer days since November 24, 4714 BC at midnight UTC)
RM	Month in upper case Roman numerals (I-XII; >
Rm	Month in lowercase Roman numerals (i-xii; >

HH	Hour of day (0-12)
HH12	Hour of day (0-12)
HH24	Hour of day (0-23)
MI	Minute (0-59)
SS	Second (0-59)
MS	Millisecond (000-999)
US	Microsecond (000000-999999)
SSSS	Seconds past midnight (0-86399)
AM, am, PM or pm	Meridiem indicator (without periods)
A.M., a.m., P.M. or p.m.	Meridiem indicator (with periods)