50+ Basic SQL Commands you should know!

Table of Contents

Introduction	3
Category A	4
Category B	8
Category C	10
Category D	13
Category E	16
Category F	19
Advanced Category	21
Conclusion	24
Reference	24

Introduction

Database management is done using a language called SQL, or Structured Query Language. SQL is made up of declarative statements and commands that give the database instructions so it can carry out tasks.

You may construct tables in databases, add to and modify enormous amounts of data, search through it to discover a specific piece of information quickly, or completely remove tables using SQL instructions.

In this article, we'll examine 50+ SQL statements for beginners and show you how to use them to successfully query a database, or ask for a specific piece of data.

Category A

CREATE

Create is used for creating the new table in the dataset.

BEGIN/ END

They are used for writing compound statements.

INSERT

it is used to insert one or more rows to the table in the database.

SELECT*

It selects all the elements present in the table.

```
SELECT * FROM Employee_Info;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONENUMBER	HIRE_DATE	JOB_ID	SALARY	COMISSION_PT	MANAGER_ID	DEPARTMENT_I
1	john	parker	jp@gmail.com	12345	03-JAN-22	AI1011	50000	2	501	1001
2	mary	gold	mg@gmail.com	23412	08-FEB-22	AI1012	52000	3	502	1002
3	binu	antony	ba@gmail.com	48643	27-FEB-22	AI1012	68000	7	505	1008
4	jojo	depp	jd@gmail.com	73643	21-MAR-22	AI1014	47000	5	508	1003
5	jinu	cina	jc@gmail.com	39584	01-APR-22	AI1015	72000	2	506	1009
6	alex	jane	aj@gmail.com	82732	18-APR-22	AI1013	41000	8	503	1005

We can also select multiple columns by using column names.

SELECT Employee_ID, First_Name
FROM Employee_Info;

EMPLOYEE_ID	FIRST_NAME
1	john
2	mary
3	binu
4	jojo
5	jinu

Computed Columns (* , /)

Following operations can be performed using the operators.

*

SELECT Employee_ID, First_Name, Salary, (Comission_pt*2) FROM Employee_Info WHERE (Comission_pt*2)>10;

EMPLOYEE_ID	FIRST_NAME	SALARY	(COMISSION_PT*2)
3	binu	68000	14
6	alex	41000	16

```
1
```

```
SELECT Employee_ID, First_Name,
Salary,(Salary/30)
FROM Employee_Info
WHERE Salary >=40000;
```

EMPLOYEE_ID	FIRST_NAME	SALARY	(SALARY/30)
1	john	50000	1666.6666666666666666666666666666666666
2	mary	52000	1733.3333333333333333333333333333333333
3	binu	68000	2266.6666666666666666666666666666666666
4	jojo	47000	1566.6666666666666666666666666666666666
5	jinu	72000	2400
6	alex	41000	1366.6666666666666666666666666666666666

NULL

It is used to check whether the row contains any null value or not.

```
Select Employee_ID, First_Name from Employee_Info where Comission_pt IS NULL
```

no data found

CONCAT

It is used to concatenate two string values in the table.

```
SELECT CONCAT(First_Name, Last_Name) AS FullName FROM Employee_Info;
```



DISTINCT

It is used to ensure that the values in the table are unique and not duplicated.

SELECT DISTINCT PhoneNumber FROM Employee_Info;

PHONI	ENUMBER
1234	5
2341	2
4864	3
82732	2
39584	4
73643	3

WHERE NOT

NOT is used just to negate any given statement.

SELECT First_Name FROM Employee_Info WHERE NOT job_ID='AI1012';

FIRST_NAME
john
jojo
jinu
alex

Category B

BETWEEN

It is used to specify the area or range between two values.

SELECT Employee_ID,First_Name FROM Employee_Info WHERE Salary BETWEEN 40000 AND 60000;

EMPLOYEE_ID	FIRST_NAME
1	john
2	mary
4	jojo
6	alex

IN

In is used to specify that the given targeted value is present in the set or not.

```
SELECT Employee_ID,First_Name FROM Employee_Info
WHERE job_ID IN ('AI1012', 'AI1013');
```

EMPLOYEE_ID	FIRST_NAME
2	mary
3	binu
6	alex

LIKE

It is used to match the pattern in the table.

```
SELECT Email, PhoneNumber FROM Employee_Info WHERE First_Name LIKE 'j%';
```

EMAIL	PHONENUMBER
jp@gmail.com	12345
jd@gmail.com	73643
jc@gmail.com	39584

COMPARISON OPERATORS (=,>)

They are used to perform various comparisons in the table using the conditions.

=

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONENUMBER	HIRE_DATE	JOB_ID	SALARY	COMISSION_PT	MANAGER_ID	DEPARTMENT_ID
2	mary	gold	mg@gmail.com	23412	08-FEB-22	AI1012	52000	3	502	1002

>

SELECT Employee_ID,First_Name,Last_Name from Employee_Info where Salary > 55000

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
3	binu	antony
5	jinu	cina

Category C

AND

It is used to combine the boolean expressions and executes if both of them are true.

```
SELECT Employee_ID, First_Name, Last_Name FROM Employee_Info WHERE Salary = 50000 AND PhoneNumber = '12345'
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
1	j <mark>ohn</mark>	parker

OR

It is used to test various conditions and execute even if any one condition is true.

```
SELECT Employee_ID, First_Name, Last_Name FROM Employee_Info WHERE Manager ID = 506 OR Comission pt = 7
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
3	binu	antony
5	jinu	cina

NOT

NOT is used just to negate any given statement.

```
SELECT First_Name FROM Employee_Info WHERE NOT job_ID='AI1012';
```

Combination of logical operators

```
SELECT Employee_ID, First_Name, Last_Name, Email FROM Employee_Info
WHERE NOT job_ID='AI1012' AND ( Salary = 50000 OR PhoneNumber = '82732')
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL
1	john	parker	jp@gmail.com
6	alex	jane	aj@gmail.com

ORDER BY

It is used to arrange the elements of the column in ascending or decending order.

SELECT Employee_ID,First_Name,Salary,Department_ID FROM Employee_Info
ORDER BY Salary ASC, Department ID ASC;

EMPLOYEE_ID	FIRST_NAME	SALARY	DEPARTMENT_ID
6	alex	41000	1005
4	jojo	47000	1003
1	john	50000	1001
2	mary	52000	1002
3	bin <mark>u</mark>	68000	1008
5	jinu	72000	1009

MIN

It is used to find the minimum value for the particular column.



MAX

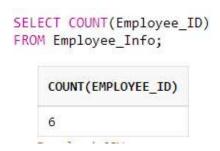
It is used to find the maximum value for the particular column.

SELECT MAX(Salary) AS LargestFees FROM Employee_Info;



COUNT

It is used to count the number of elements in the specific column.



SUM

It executes the sum of the elements present in the column

SELECT SUM(Salary) AS SALARYSUM FROM Employee_Info;



Category D

GROUP BY

It is used to group the elements which have the same values in one or more columns.

```
SELECT COUNT(Employee_ID), job_ID
FROM Employee_Info
GROUP BY job_ID;
```

COUNT(EMPLOYEE_ID)	JOB_ID
1	AI1011
2	AI1012
1	AI1014
1	AI1015
1	AI1013

GROUP BY HAVING

It is used to group the elements which have the same values in one or more columns followed by a condition which needs to be true.

```
SELECT COUNT(Employee_ID),job_ID
FROM Employee_Info
GROUP BY job_ID
HAVING COUNT(Employee_ID) < 3
```

no data found

Build In Functions

UPPER

It converts all the letters to the uppercase.

```
SELECT UPPER(First_Name) AS UppercaseFirstName FROM Employee_Info;
```



LOWER

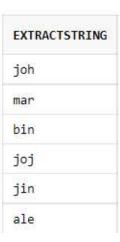
It converts all the letters to the lowercase.

```
SELECT UPPER(Last_Name) AS LowercaseLatName FROM Employee_Info;
```

SUBSTR

It executes the position of the character present in the string.

```
SELECT SUBSTR(First_Name, 1, 3) AS ExtractString FROM Employee_Info;
```



LISTAGG

It uses delimiter to convert multiple row data to the single list of the values.

```
SELECT LISTAGG(First_Name, '\') WITHIN GROUP (ORDER BY First_Name)Agg_Name FROM Employee_Info;
```

AGG_NAME alex\binu\jinu\john\jojo\mary

INITCAP

It is used to capatilize the first letter of the string.

Select INITCAP(First_name) from Employee_Info;

INITCA	AP(FIRST_NAME)
John	
Mary	
Binu	
Jojo	
Jinu	
Alex	

CEIL

It is used to round off the value and use the smallest or equal round of that value.

Select CEIL(Comission_pt) from Employee_Info;



Category E

CREATING TABLE 2 FOR JOIN OPERATIONS

```
CREATE TABLE AdditionalInfo (
refID number(6,0),
Dept_ID number(6,0),
HouseNo varchar2(50 byte),
city varchar2(50 byte),
state varchar2(50 byte)
);
```

REFID	DEPT_ID	HOUSENO	CITY	STATE
151	1001	742	New Delhi	Delhi
152	1002	848	Bhopal	MP
153	1008	472	Lucknow	UP
154	1009	374	Kolkata	West bengal

LEFT OUTER JOIN

It returns all the records of the left (table 1) and matching record from table2(right)

```
SELECT Employee_Info.Employee_ID, Employee_Info.First_Name, AdditionalInfo.city
FROM Employee_Info
LEFT OUTER JOIN AdditionalInfo
ON Employee_Info.Department_ID = AdditionalInfo.Dept_ID;
```

EMPLOYEE_ID	FIRST_NAME	CITY
1	john	New Delhi
2	mary	Bhopal
3	binu	Lucknow
5	jinu	Kolkata
6	alex	=
4	jojo	2

RIGHT OUTER JOIN

It returns all the records of the right (table 2) and matching record from table1(left)

```
SELECT AdditionalInfo.refID, AdditionalInfo.HouseNo, Employee_Info.First_Name FROM Employee_Info
RIGHT OUTER JOIN AdditionalInfo
ON Employee_Info.Department_ID = AdditionalInfo.Dept_ID;
```

REFID	HOUSENO	FIRST_NAME
151	742	john
152	848	mary
153	472	binu
154	374	jinu

FULL JOIN

It is used to join the results obtained from right and left joins.

```
SELECT Employee_Info.Employee_ID, Employee_Info.First_Name, AdditionalInfo.state
FROM Employee_Info
FULL OUTER JOIN AdditionalInfo
ON Employee_Info.Department_ID = AdditionalInfo.Dept_ID;
```

EMPLOYEE_ID	FIRST_NAME	STATE
1	john	Delhi
2	mary	MP
3	binu	UP
4	jojo	×
5	jinu	West bengal
6	alex	-

CROSS JOIN

It is used to execute the cartesian product of the rows from the table.

SELECT Employee_Info.Employee_ID,Employee_Info.First_Name, AdditionalInfo.city,AdditionalInfo.state FROM Employee_Info CROSS JOIN AdditionalInfo;

EMPLOYEE_ID	FIRST_NAME	CITY	STATE
1	john	New Delhi	Delhi
2	mary	New Delhi	Delhi
3	binu	New Delhi	Delhi
4	jojo	New Delhi	Delhi
5	jinu	New Delhi	Delhi
6	alex	New Delhi	Delhi
1	john	Bhopal	MP
2	mary	Bhopal	MP
3	binu	Bhopal	MP
4	jojo	Bhopal	MP
5	jinu	Bhopal	MP
6	alex	Bhopal	MP
1	john	Lucknow	UP
2	mary	Lucknow	UP
3	binu	Lucknow	UP
4	jojo	Lucknow	UP
5	jinu	Lucknow	UP
6	alex	Lucknow	UP
1	john	Kolkata	West bengal
2	mary	Kolkata	West bengal
3	binu	Kolkata	West bengal
4	jojo	Kolkata	West bengal
5	jinu	Kolkata	West bengal

Category F

Single row subqueries

```
SELECT First_Name, Employee_ID, PhoneNumber
FROM Employee_Info
WHERE Employee_ID =
(SELECT Employee_ID
FROM Employee_Info
WHERE First_Name = 'alex');
```

FIRST_NAME	EMPLOYEE_ID	PHONENUMBER
alex	6	82732

EXISTS

It is used to check whether the record is present in the subquery or not.

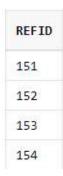
```
SELECT First_Name
FROM Employee_Info
WHERE EXISTS (SELECT First_Name FROM Employee_Info WHERE Employee_Id = 01 AND Department_ID = 1001);
```



```
SELECT refID

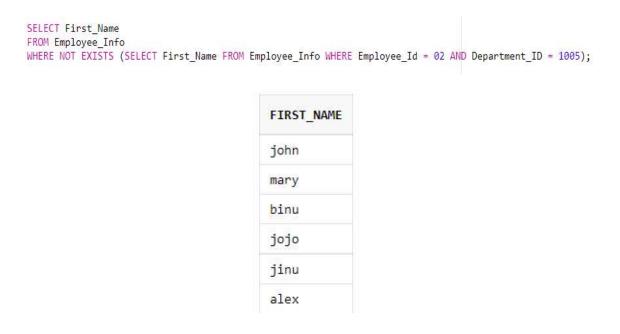
FROM AdditionalInfo

WHERE EXISTS (SELECT refID FROM AdditionalInfo WHERE city = 'Lucknow' AND state = 'UP');
```



NOT EXISTS

It is the opposite of EXISTS. It is used to check whether the record is absent in the subquery. It returns true if the record is not present.



```
SELECT refID
FROM AdditionalInfo
WHERE NOT EXISTS (SELECT refID FROM AdditionalInfo WHERE city = 'Bhopal' AND state = 'MP');
```

no data found

Advanced Category

UNION

It is used to combine two or more selected elements together.

```
SELECT Employee_ID,PhoneNumber,Email FROM Employee_Info;
UNION
SELECT refID,city,state FROM AdditionalInfo;
```

EMPLOYEE_ID	PHONENUMBER	EMAIL		
1	12345	jp@gmail.com		
2	23412	mg@gmail.com		
3	48643	ba@gmail.com		
4	73643	jd@gmail.com		
5	39584	jc@gmail.com		
6	82732	aj@gmail.com		

REFID	CITY	STATE			
151	New Delhi	Delhi			
152	Bhopal	MP			
153	Lucknow	UP			
154	Kolkata	West bengal			

ANY

It is used to specify the value which is less or greater than the values in the subquery.

```
SELECT First_Name
FROM Employee_Info
WHERE Department_ID = ANY
(SELECT refID
FROM AdditionalInfo
WHERE state = 'Delhi');
```

no data found

COMMIT

It stores all the changes that are performed by any transaction in a database.

COMMIT;

Statement processed.

ROLLBACK

It is used to undo the transactions in the database which are not saved in it.

ROLLBACK;

Statement processed.

NESTED SUBQUERIES

1.

```
FROM Employee_Info
WHERE Employee_ID IN (SELECT Employee_ID
FROM Employee_Info
WHERE Salary > 45000);
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONENUMBER	HIRE_DATE	JOB_ID	SALARY	COMISSION_PT	MANAGER_ID	DEPARTMENT_ID
1	john	parker	jp@gmail.com	12345	03-JAN-22	AI1011	50000	2	501	1001
2	mary	gold	mg@gmail.com	23412	08-FEB-22	AI1012	52000	3	502	1002
3	binu	antony	ba@gmail.com	48643	27-FEB-22	AI1012	68000	7	505	1008
4	jojo	depp	jd@gmail.com	73643	21-MAR-22	AI1014	47000	5	508	1003
5	jinu	cina	jc@gmail.com	39584	01-APR-22	AI1015	72000	2	506	1009

2.

```
UPDATE Employee_Info
SET Salary = Salary * 0.25
WHERE Department_ID IN (SELECT Dept_ID FROM AdditionalInfo
    WHERE Dept_ID = 1002);
select * from Employee_Info;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONENUMBER	HIRE_DATE	JOB_ID	SALARY	COMISSION_PT	MANAGER_ID	DEPARTMENT_ID
1	john	parker	jp@gmail.com	12345	03-JAN-22	AI1011	50000	2	501	1001
2	mary	gold	mg@gmail.com	23412	08-FEB-22	AI1012	13000	3	502	1002
3	binu	antony	ba@gmail.com	48643	27-FEB-22	AI1012	68000	7	505	1008
4	jojo	depp	jd@gmail.com	73643	21-MAR-22	AI1014	47000	5	508	1003
5	jinu	cina	jc@gmail.com	39584	01-APR-22	AI1015	72000	2	506	1009
6	alex	jane	aj@gmail.com	82732	18-APR-22	AI1013	41000	8	503	1005

INDEX

It is created to find the exact location of the data without consuming much time.

```
CREATE INDEX index_
ON Employee_Info (Employee_ID);
```

Index created.

ALTER

It is used to add/delete/modify the columns in the table.

```
ALTER TABLE Employee_Info DROP COLUMN Comission_pt;
```

Table altered.

EMPLOYEE_ID	FIRST_NAME	EMAIL	PHONENUMBER	HIRE_DATE	JOB_ID	SALARY	COMISSION_PT	MANAGER_ID	DEPARTMENT_ID
1	john	jp@gmail.com	12345	03-JAN-22	AI1011	50000	2	501	1001
2	mary	mg@gmail.com	23412	08-FEB-22	AI1012	52000	3	502	1002
3	binu	ba@gmail.com	48643	27-FEB-22	AI1012	68000	7	505	1008
4	jojo	jd@gmail.com	73643	21-MAR-22	AI1014	47000	5	508	1003
5	jinu	jc@gmail.com	39584	01-APR-22	AI1015	72000	2	506	1009
6	alex	aj@gmail.com	82732	18-APR-22	AI1013	41000	8	503	1005

TO DELETE ALL ROWS

```
delete from Employee_Info;
6 row(s) deleted.
```

TO DELETE TABLE

```
DROP TABLE Employee_Info;
```

Table dropped.

Conclusion

Sql is commonly used everywhere in the industry. We can write queries in order to perform operations with the data stored in the database.

In this project 50+ queries are performed for different data manipulation purposes.

Reference

1. Oracle Help Centre - DataBase SQL Language Reference