

SQL Commands

By :

Dr. Rinkle Aggarwal

Associate Professor, CSED

TIET, Patiala

The INSERT Statement Syntax

- Add new rows to a table by using the INSERT statement.

```
INSERT INTO  table [(column [, column...])]  
VALUES      (value [, value...]);
```

- Only one row is inserted at a time with this syntax.

Inserting New Rows

- **Insert a new row containing values for each column.**
- **List values in the default order of the columns in the table.**
- **Optionally, list the columns in the INSERT clause.**

```
INSERT INTO departments (department_id, department_name,  
                        manager_id, location_id)  
VALUES      (70, 'Public Relations', 100, 1700);  
1 row created.
```

- **Enclose character and date values within single quotation marks.**

Inserting Rows with Null Values

- **Implicit method: Omit the column from the column list.**

```
INSERT INTO departments (department_id,  
                        department_name  )  
VALUES (30, 'Purchasing');  
1 row created.
```

- **Explicit method: Specify the NULL keyword in the VALUES clause.**

```
INSERT INTO departments  
VALUES (100, 'Finance', , );  
1 row created.
```

Tables Used in the Course

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95	ST_CLERK
142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97	ST_CLERK
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700

0.121.2004	09-JUL-98	ST_CLERK
1.44.1344.429018	29-JAN-00	SA_MAN
1.44.1644.129067	11-MAY-06	SA_DEP
1.44.1644.129067		
GRA	LOWEST_SAL	H
A	1000	
B	3000	
C	6000	
D	10000	
E	15000	

Capabilities of SQL SELECT Statements

Projection

A diagram illustrating the Projection operation. It shows a 10x5 grid representing a table. The second and fourth columns are highlighted in pink, indicating they are the selected attributes. The other columns are yellow.

Table 1

Selection

A diagram illustrating the Selection operation. It shows a 10x5 grid representing a table. The first, third, and fourth rows are highlighted in pink, indicating they are the selected rows. The other rows are yellow.

Table 1

A diagram representing Table 1 for the Join operation. It is a 10x5 grid where the last column is highlighted in pink, and the other columns are yellow.

Table 1

Join



A diagram representing Table 2 for the Join operation. It is a 10x6 grid where the first column is highlighted in pink, and the other columns are yellow.

Table 2

Selecting All Columns

```
SELECT *  
FROM departments;
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

Selecting Specific Columns

```
SELECT department_id, location_id  
FROM departments;
```

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

Writing SQL Statements

- **SQL statements are not case sensitive.**
- **SQL statements can be on one or more lines.**
- **Keywords cannot be abbreviated or split across lines.**
- **Clauses are usually placed on separate lines.**
- **Indents are used to enhance readability.**

Arithmetic Expressions

Create expressions with number and date data by using arithmetic operators.

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

Using Arithmetic Operators

```
SELECT last_name, salary, salary + 300  
FROM   employees;
```

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300

...

Hartstein	13000	13300
Fay	6000	6300
Higgins	12000	12300
Gietz	8300	8600

20 rows selected.

Operator Precedence



- **Multiplication and division take priority over addition and subtraction.**
- **Operators of the same priority are evaluated from left to right.**
- **Parentheses are used to force prioritized evaluation and to clarify statements.**

Operator Precedence

```
SELECT last_name, salary, 12*salary+100
FROM   employees;
```

LAST_NAME	SALARY	12*SALARY+100
King	24000	288100
Kochhar	17000	204100
De Haan	17000	204100
Hunold	9000	108100
Ernst	6000	72100

■ ■ ■

Hartstein	13000	156100
Fay	6000	72100
Higgins	12000	144100
Gietz	8300	99700

20 rows selected.

Using Parentheses

```
SELECT last_name, salary, 12*(salary+100)
FROM   employees;
```

LAST_NAME	SALARY	12*(SALARY+100)
King	24000	289200
Kochhar	17000	205200
De Haan	17000	205200
Hunold	9000	109200
Ernst	6000	73200

...

Hartstein	13000	157200
Fay	6000	73200
Higgins	12000	145200
Gietz	8300	100800

20 rows selected.

Defining a Null Value

- A null is a value that is unavailable, unassigned, unknown, or inapplicable.
- A null is not the same as zero or a blank space.

```
SELECT last_name, job_id, salary, commission_pct  
FROM employees;
```

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
King	AD_PRES	24000	
Kochhar	AD_VP	17000	

...

Zlotkey	SA_MAN	10500	.2
Abel	SA_REP	11000	.3
Taylor	SA_REP	8600	.2

...

Gietz	AC_ACCOUNT	8300	
-------	------------	------	--

20 rows selected.

Null Values in Arithmetic Expressions

Arithmetic expressions containing a null value evaluate to null.

```
SELECT last_name, 12*salary*commission_pct  
FROM   employees;
```

LAST_NAME	12*SALARY*COMMISSION_PCT
King	
Kochhar	
...	
Zlotkey	25200
Abel	39600
Taylor	20640
...	
Gietz	

20 rows selected

Defining a Column Alias

A column alias:

- **Renames a column heading**
- **Is useful with calculations**
- **Immediately follows the column name - there can also be the optional AS keyword between the column name and alias**
- **Requires double quotation marks if it contains spaces or special characters or is case sensitive**

Using Column Aliases

```
SELECT last_name AS name, commission_pct comm
FROM employees;
```

NAME	COMM
King	
Kochhar	
De Haan	

20 rows selected.

```
SELECT last_name "Name", salary*12 "Annual Salary"
FROM employees;
```

Name	Annual Salary
King	288000
Kochhar	204000
De Haan	204000

20 rows selected.

Concatenation Operator

A concatenation operator:

- **Concatenates columns or character strings to other columns**
- **Is represented by two vertical bars (||)**
- **Creates a resultant column that is a character expression**

Using the Concatenation Operator

```
SELECT    last_name||job_id AS "Employees"  
FROM      employees;
```

Employees	
KingAD_PRE	
KochharAD_VP	
De HaanAD_VP	
HunoldIT_PROG	
ErnstIT_PROG	
LorentzIT_PROG	
MourgosST_MAN	
RajsST_CLERK	

■ ■ ■

20 rows selected.

Literal Character Strings

- A literal is a character, a number, or a date included in the `SELECT` list.
- Date and character literal values must be enclosed within single quotation marks.
- Each character string is output once for each row returned.

Using Literal Character Strings

```
SELECT last_name || ' is a ' || job_id  
       AS "Employee Details"  
FROM   employees;
```

Employee Details
King is a AD_PRE
Kochhar is a AD_VP
De Haan is a AD_VP
Hunold is a IT_PROG
Ernst is a IT_PROG
Lorentz is a IT_PROG
Mourgos is a ST_MAN
Rajs is a ST_CLERK

20 rows selected.

Duplicate Rows

The default display of queries is all rows, including duplicate rows.

```
SELECT department_id  
FROM   employees;
```

DEPARTMENT_ID	
	90
	90
	90
	60
	60
	60
	50
	50
	50

Eliminating Duplicate Rows

Eliminate duplicate rows by using the **DISTINCT** keyword in the **SELECT** clause.

```
SELECT DISTINCT department_id  
FROM employees;
```

DEPARTMENT_ID	
	10
	20
	50
	60
	80
	90
	110

8 rows selected.

Displaying Table Structure

Use the *iSQL*Plus* DESCRIBE command to display the structure of a table.

```
DESC[RIBE] tablename
```

Displaying Table Structure

```
DESCRIBE employees
```

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

Limiting Rows Using a Selection

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90
103	Hunold	IT_PROG	60
104	Ernst	IT_PROG	60
107	Lorentz	IT_PROG	60
124	Mourgos	ST_MAN	50

...

20 rows selected.

**“retrieve all
employees
in department 90”**



EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

Limiting the Rows Selected

- Restrict the rows returned by using the **WHERE** clause.

```
SELECT    * | { [DISTINCT] column | expression [alias], ... }  
FROM      table  
[WHERE    condition(s)];
```

- The **WHERE** clause follows the **FROM** clause.

Using the WHERE Clause

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90 ;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

Character Strings and Dates

- Character strings and date values are enclosed in single quotation marks.
- Character values are case sensitive, and date values are format sensitive.
- The default date format is DD-MON-RR.

```
SELECT last_name, job_id, department_id  
FROM employees  
WHERE last_name = 'Whalen';
```

Comparison Conditions

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

Using Comparison Conditions

```
SELECT last_name, salary
FROM   employees
WHERE  salary <= 3000;
```

LAST_NAME	SALARY
Matos	2600
Vargas	2500

Other Comparison Conditions

Operator	Meaning
BETWEEN ...AND...	Between two values (inclusive),
IN (set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

Using the BETWEEN Condition

Use the **BETWEEN** condition to display rows based on a range of values.

```
SELECT last_name, salary
FROM employees
WHERE salary BETWEEN 2500 AND 3500;
```



Lower limit

Upper limit

LAST_NAME	SALARY
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500

Using the IN Condition

Use the **IN** membership condition to test for values in a list.

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201);
```

EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
202	Fay	6000	201
200	Whalen	4400	101
205	Higgins	12000	101
101	Kochhar	17000	100
102	De Haan	17000	100
124	Mourgos	5800	100
149	Zlotkey	10500	100
201	Hartstein	13000	100

8 rows selected.

Using the LIKE Condition

- Use the **LIKE** condition to perform wildcard searches of valid search string values.
- Search conditions can contain either literal characters or numbers:
 - **%** denotes zero or many characters.
 - **_** denotes one character.

```
SELECT    first_name
FROM      employees
WHERE     first_name LIKE 'S%';
```

Using the LIKE Condition

- You can combine pattern-matching characters.

```
SELECT last_name  
FROM employees  
WHERE last_name LIKE '_o%';
```

LAST_NAME
Kochhar
Lorentz
Mourgos

- You can use the `ESCAPE` identifier to search for the actual `%` and `_` symbols.

Using the NULL Conditions

Test for nulls with the IS NULL operator.

```
SELECT last_name, manager_id  
FROM employees  
WHERE manager_id IS NULL;
```

LAST_NAME	MANAGER_ID
King	

Logical Conditions

Operator	Meaning
AND	Returns TRUE if <i>both</i> component conditions are true
OR	Returns TRUE if <i>either</i> component condition is true
NOT	Returns TRUE if the following condition is false

Using the AND Operator

AND requires both conditions to be true.

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >=10000
AND    job_id LIKE '%MAN%';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
149	Zlotkey	SA_MAN	10500
201	Hartstein	MK_MAN	13000

Using the OR Operator

OR requires either condition to be true.

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
100	King	AD_PRES	24000
101	Kochhar	AD_VP	17000
102	De Haan	AD_VP	17000
124	Mourgos	ST_MAN	5800
149	Zlotkey	SA_MAN	10500
174	Abel	SA_REP	11000
201	Hartstein	MK_MAN	13000
205	Higgins	AC_MGR	12000

8 rows selected.

Using the NOT Operator

```
SELECT last_name, job_id
FROM employees
WHERE job_id
      NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP');
```

LAST_NAME	JOB_ID
King	AD_PRES
Kochhar	AD_VP
De Haan	AD_VP
Mourgos	ST_MAN
Zlotkey	SA_MAN
Whalen	AD_ASST
Hartstein	MK_MAN
Fay	MK_REP
Higgins	AC_MGR
Gietz	AC_ACCOUNT

10 rows selected.

Rules of Precedence

Order Evaluated	Operator
1	Arithmetic operators
2	Concatenation operator
3	Comparison conditions
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	NOT logical condition
7	AND logical condition
8	OR logical condition

Override rules of precedence by using parentheses.

Rules of Precedence

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id = 'SA_REP'
OR job_id = 'AD_PRES'
AND salary > 15000;
```

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000
Abel	SA_REP	11000
Taylor	SA_REP	8600
Grant	SA_REP	7000

Rules of Precedence

Use parentheses to force priority.

```
SELECT last_name, job_id, salary
FROM employees
WHERE (job_id = 'SA_REP'
OR job_id = 'AD_PRES')
AND salary > 15000;
```

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000

ORDER BY Clause

- Sort rows with the ORDER BY clause
 - ASC: ascending order, default
 - DESC: descending order
- The ORDER BY clause comes last in the SELECT statement.

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY hire_date ;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
King	AD_PRES	90	17-JUN-87
Whalen	AD_ASST	10	17-SEP-87
Kochhar	AD_VP	90	21-SEP-89
Hunold	IT_PROG	60	03-JAN-90
Ernst	IT_PROG	60	21-MAY-91

■ ■ ■

20 rows selected.

Sorting in Descending Order

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY hire_date DESC ;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
Zlotkey	SA_MAN	80	29-JAN-00
Mourgos	ST_MAN	50	16-NOV-99
Grant	SA_REP		24-MAY-99
Lorentz	IT_PROG	60	07-FEB-99
Vargas	ST_CLERK	50	09-JUL-98
Taylor	SA_REP	80	24-MAR-98
Matos	ST_CLERK	50	15-MAR-98
Fay	MK_REP	20	17-AUG-97
Davies	ST_CLERK	50	29-JAN-97

■ ■ ■

20 rows selected.

Sorting by Column Alias

```
SELECT employee_id, last_name, salary*12 annsal  
FROM employees  
ORDER BY annsal;
```

EMPLOYEE_ID	LAST_NAME	ANNSAL
144	Vargas	30000
143	Matos	31200
142	Davies	37200
141	Rajs	42000
107	Lorentz	50400
200	Whalen	52800
124	Mourgos	69600
104	Ernst	72000
202	Fay	72000
178	Grant	84000

■ ■ ■

20 rows selected.

Sorting by Multiple Columns

- The order of ORDER BY list is the order of sort.

```
SELECT last_name, department_id, salary
FROM employees
ORDER BY department_id, salary DESC;
```

LAST_NAME	DEPARTMENT_ID	SALARY
Whalen	10	440
Hartstein	20	1300
Fay	20	600
Mourgos	50	580
Rajs	50	350
Davies	50	310
Matos	50	260
Vargas	50	250

...

20 rows selected.

- You can sort by a column that is not in the