



SQL



SQL

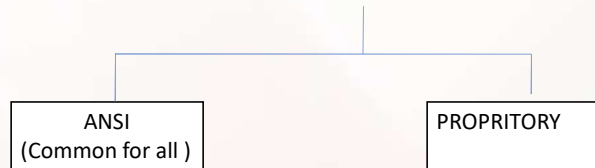


- SQL stands for **S**tructured **Q**uery **L**anguage
- SQL is the language used to communicate with the database (Oracle MS-ACCES, MS SQL Server) to access, manipulate, and control data.

SQL



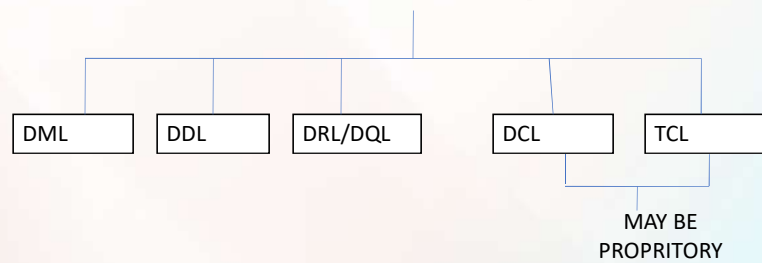
SQL TYPE



SQL



SQL Command(categories)



SQL Reports/outputs

COMPLETE
DETAILS(FILTER OR
WHERE)COMPLETE
SUMMARY(GROUP BY
CLAUSE)COMPLETE
DETAILS
+
SUMMARY

(SUBTOTAL)

• Capabilities of SQL

Projection

Selection

Join

SQL



- DRL/DQL
- DRL is **D**ata **R**etrieval **L**anguage(**D**ata **Q**uery **L**anguage)
- Used to retrieve or query the database and fetch selected data which matches the criteria
- It is done by using a 'SELECT' Statement

Select Statement in Sql



- It helps to fetch data
- It is 'SELECTION' Type of query .
- Basic Select Statement:-

```
SELECT *|{[DISTINCT] column|expression [alias],...}  
FROM table;
```

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.

SQL example

- Example:-
- `Select * from emp;`
- `Select ename,sal from emp;`

Arithmetic Expressions

- You can use arithmetic Expressions within select statements

Operator	Description
+	Add
-	subtract
*	multiply
/	divide

Oprt. example

```
SELECT ename, sal, sal + 300  
FROM emp;
```

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.

SQL Example of Use of Parenthesis

```
SELECT ename, sal, 12*sal+100  
FROM emp;
```

```
SELECT ename, sal, 12*(sal+100)  
FROM emp;
```


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 ename, job, sal, comm  
FROM emp;
```

- Arithmetic expressions containing NULL value evaluate to NULL.

Defining 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

Example Using Column Aliases

- ```
SELECT ename AS name,
 comm comm_given
FROM emp;
```
- ```
SELECT ename "Name",  
       sal*12 "Annual Salary"  
FROM emp;
```

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

```
SELECT ename || job AS "Employees"  
FROM emp;
```

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.
- `SELECT ename || ' is a ' || job`
- `AS "Employee Details"`
`FROM emp;`

Using 'Distinct' in Select Stmtnt

- `SELECT deptno`
`FROM emp;`
- `SELECT distinct deptno`
`FROM emp;`

Will this work?

```
SELECT DISTINCT deptno, job FROM emp;
```

SQL REPORTS OR OUTPUT



Complete details

Restricting number of rows



- Restrict the rows returned by using the WHERE clause.
- The WHERE clause follows the FROM clause.
- `SELECT *|{[DISTINCT] column|expression
[alias],...}
FROM table
[WHERE condition(s)];`

Where clause in 'Select Statement'



- `Select dname,loc`
 - `From dept`
 - `Where deptno=30`
-
- `Select dname,loc`
`From dept`
`Where deptno=50;`

Character Strings And Dates



- Character strings and date values are enclosed in single quotation marks.
- Characters are case sensitive
- dates are format sensitive

Comparison Conditions

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than equal to
<	Less than
<=	Less than equal to
!= / <> / ^=	NOT Equal to

Other Comparison Conditions

Operator	Meaning
• BETWEEN ...AND...	Between two values(inclusive), (NOT BETWEEN.....AND)
• IN(set) / NOT IN	Match any of a list of values
• LIKE / NOT LIKE	Match a character pattern
• IS NULL / • IS NOT NULL	Is a null value

Using the between condition

- Use the BETWEEN condition to display rows based on a range of values. This range includes Lower as well as Upper Limits

```
• SELECT ename, sal
  FROM emp
 WHERE sal BETWEEN 2500 AND 3500;
                Lower limit      Upper limit
```

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);
```

* Maximum values which can be provided to the set of IN operator is 1024 for comparison

Using the like operator

- 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.

Using 'Escape' Contd.

- **The ESCAPE Option**
- When you need to have an exact match for the actual % and _ characters, use the *ESCAPE* option.
- This option specifies what the escape character is. If you want to search for strings that contain 'SA_',

Using Like Operator with Escape

- you can use the following SQL statement:
- ```
SELECT employee_id, last_name, job_id
FROM employees
WHERE job_id LIKE '%SA_%' ESCAPE '\';
```

## Using NULL

- **IS NULL operator:-**
- ```
SELECT last_name, manager_id
FROM employees
WHERE manager_id IS NULL;
```

Rules Of Precedence

```
SELECT ename, job, sal
FROM emp
WHERE job = 'SALESMAN'
OR job = 'PRESIDENT'
AND sal > =5000;
* (job_id = 'SA_REP'
OR job_id = 'AD_PRES')
```

SORTING DATA

- It is displaying data in 'Ascending' or 'Descending' manner.

ORDER BY Clause

- Sort rows with the ORDER BY clause
 - ASC: ascending order, default
 - DESC: descending order
- You can sort Data on Multiple columns too
 - *The ORDER BY clause comes last in the SELECT statement.

Which Clause Of The Query Executes First

```
Select *  
From emp  
Where sal>4000;
```

Ways Of Sorting

```
Select ename,sal+100 as Total_Sal, job  
  From emp  
  Order by job;
```

```
Select ename,sal+100 as Total_Sal, job  
  From emp  
  Order by job Total_Sal;
```

Ways Of Sorting

```
Select ename,sal+100 as Total_Sal, job  
  From emp  
  Order by 3;
```

/

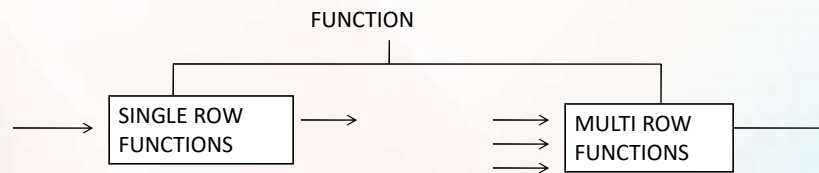
- Max Number of col. Tht can be used in Order By Clause are - 1024.
- **B**est Practices are use of aliases to expression in order by clause

- SUMMARY QUERIES

Aggregate Functions

- The summary queries are those queries which use different Functions.
- Perform calculations on data
- Modify individual data items
- Manipulate output for groups of rows
- Format dates and numbers for display
- Convert column data types
- SQL functions sometimes take arguments and always return a value.

Two Types of SQL Functions



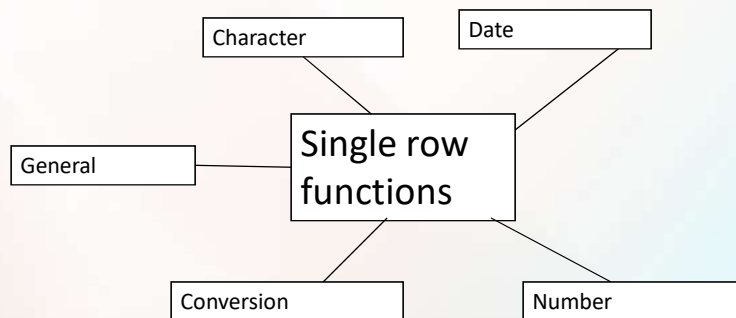
SUMMARY QUERIES



Single row functions

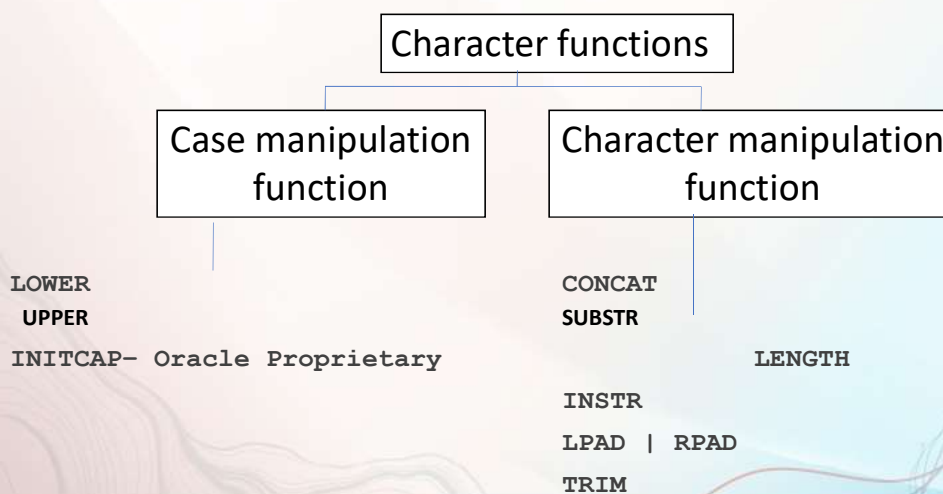
- Manipulate data items
 - Accept arguments and return one value
 - Act on each row returned
 - Return one result per row
 - May modify the data type
 - Can be nested
 - Accept arguments which can be a column or an
- Expression `function_name [(arg1, arg2,...)]`

Single Row Function



Character functions

accept character data as input and can return both character and numeric values.



Character functions

Function	Result
<code>CONCAT('Hello', 'World')</code>	HelloWorld
<code>SUBSTR('HelloWorld',1,5)</code>	Hello
<code>LENGTH('HelloWorld')</code>	10
<code>INSTR('HelloWorld', 'W')</code>	6
<code>LPAD(salary,10,'*')</code>	*****24000
<code>RPAD(salary, 10, '*')</code>	24000*****
<code>TRIM('H' FROM 'HelloWorld')</code>	elloWorld

Character functions

e.g.

`SUBSTR(column|expression,m[,n])`

Returns specified characters from character value starting at character position *m*, *n* characters long.

If m is negative, the count starts from the end of the character value.

If n is omitted, all characters to the end of the string are returned.

Example



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> set lin 100;
SQL>
SQL> select substr('HELLOWORLD',1,5) from DUAL;

SUBST
-----
HELLO
SQL> |
```

Character function



LPAD/RPAD

- **SELECT** LPAD(sal, 10, '*') **FROM** emp;
- **SELECT** RPAD(sal, 10, '*') **FROM** emp;
- **SELECT** sal,sal/100, rpad('o',sal/100,'o') as histogram
from emp
where deptno = 30;

•

Character functions

Simple 'TRIM' function:-

```
SELECT TRIM (0 FROM 067270676800) "TRIM Example"  
FROM DUAL;
```

```
SELECT TRIM('S' FROM 'STEVENS') AS TRIM  
FROM Dual
```

```
SELECT TRIM(TRAILING 'S' FROM 'STEVENS' ) AS TRIM  
FROM Dual
```

```
SELECT TRIM(LEADING 'S' FROM 'STEVENS' ) AS TRIM  
FROM Dual
```

```
SELECT TRIM (both ' ' from ' String with blanks ' )  
FROM dual
```

Number Functions

- Number functions accept numeric input and return numeric values.
- Round()
- Trunc()
- Mod()

Number function

ROUND ()

- select round(788.447,2) from dual

Examples	Result
ROUND(748.58, -1)	750.00
ROUND(748.58, -2)	700.00
ROUND(748.58, -3)	1000.00

NUMBER FUNCTIONS

TRUNC ()

- The TRUNC function truncates the column, expression, or value to *n decimal places*

```
SELECT TRUNC(45.923,2), TRUNC(45.923),
TRUNC(45.923,-2)
FROM DUAL;
```

```
SQL>
SQL>
SQL> SELECT TRUNC(45.923,2), TRUNC(45.923),
2 TRUNC(45.923,-2)
3 FROM DUAL;

TRUNC(45.923,2) TRUNC(45.923) TRUNC(45.923,-2)
-----
45.92          45              0

SQL>
```

Number Function

- The MOD function finds the remainder of value1 divided by value2.

```
SELECT ename, sal, MOD(sal, 5000)
FROM emp
WHERE job= 'SALESMAN';
```

Using Date Function

- Oracle database stores dates in an internal numeric format: century, year, month, day, hours, minutes, seconds.
- The default date display format is DD-MON-RR.
SYSDATE is a function that returns:
 - Date
 - Time

Date Function.

- e.g.

```
SELECT ename, hiredate
FROM emp
WHERE ename like 'A%';
```

ENAME	HIREDATE
ALLEN	20-FEB-81
ADAMS	23-MAY-87

SQL>

Arithmetic with Dates

- Add or subtract a number to or from a date for a resultant date value.
- Subtract two dates to find the number of days between those dates.
- Add hours to a date by dividing the number of hours by 24.

Arithmetic with Dates

- Since the database stores dates as numbers, you can perform calculations using arithmetic operators
- such as addition and subtraction. You can add and subtract number constants as well as dates

Operation	Result	Description
date + number	Date	Adds a number of days to a date
date - number	Date	Subtracts a number of days from a date
date - date	Number of days	Subtracts one date from another
date + number/24	Date	Adds a number of hours to a date

Date Functions

Function	Description
MONTHS_BETWEEN	Number of months between two dates
ADD_MONTHS	Add calendar months to Date
NEXT_DAY	Next day of the date Specified
LAST_DAY	Last day of the month
ROUND	Round date
TRUNC	Truncate date

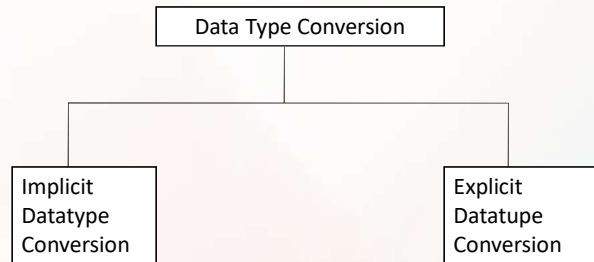
Using Date Function

- `MONTHS_BETWEEN ('01-SEP-95','11-JAN-94') - 19.6774194`
- `ADD_MONTHS ('11-JAN-94',6) - '11-JUL-94'`
- `NEXT_DAY ('01-SEP-95','FRIDAY') - '08-SEP-95'`
- `LAST_DAY ('01-FEB-95') - '28-FEB-95'`

ROUND & TRUNC With Date

- `ROUND (SYSDATE,'MONTH') 01-AUG-95`
- `TRUNC (SYSDATE , 'YEAR') 01-JAN-95`

Conversion function



- CHAR TO NUMBER
- CHAR TO DATE
- DATE TO CHAR
- NUMBR TO CHAR

Explicit Data Type Conversions



- SQL provides three functions to convert a value from one data type to another:
- TO_NUMBER
- TO_CHAR
- TO_DATE

Explicit Datatype Conversion

- The format model:
- **TO_CHAR(*date*, '*format_model*')**
- Must be enclosed in single quotation marks and is case sensitive
- Can include any valid date format element
- Is separated from the date value by a comma

TO_CHAR() With date datatype

```
• SELECT empno, TO_CHAR(hiredate, 'MM/YY')  
  FROM emp  
 WHERE ename = 'SMITH';
```

- | | |
|---------|--|
| • YYYY | Full year in numbers |
| • MONTH | Full name of the month |
| • DD | Numeric day of the month |
| • DY | Name of day; three-letter abbreviation |

Elements of the Date Format Model



- Time elements format the time portion of the date.
- Add character strings by enclosing them in double quotation marks.
- Number suffixes spell out numbers.
- HH24:MI:SS AM 15:45:32 PM
- DD "of" MONTH 12 of OCTOBER
ddspth fourteenth

TO_CHAR() With Number Datatype



- `TO_CHAR(number, 'format_model')`
- These are some of the format elements you can use with the `TO_CHAR` function to display a number value as a character:
- `TO_CHAR(number, 'format_model')`

TO_CHAR() With Number

9	Represents a number
0	Forces a zero to be displayed
\$	Places a floating dollar sign
L	Uses the floating local currency symbol
.	Prints a decimal point
,	Prints a thousand indicator
B	Display zero values as blank , not zero

Using the TO_NUMBER and TO_DATE Functions

- Convert a character string to a number format using the TO_NUMBER function:

```
TO_NUMBER(char[, 'format_model'])
```

- Convert a character string to a date format using the TO_DATE function:

```
TO_DATE(char[, 'format_model'])
```

TO_NUMBER() Datatype

- `SELECT TO_NUMBER('- $12,345.67', '$99,999.99') FROM dual;`

- `TO_DATE()` Datatype

- e.g. 22-july-2009

```
Select to char(to date('22-jul-2009','dd-mon
yyyy'),'day-mon-yy') from dual;
```

```
SELECT TO DATE('061167','MMDDYY') "Birthday"
from DUAL
```

Nesting Functions

- Single-row functions can be nested to any level.
- Nested functions are evaluated from deepest level to the least deep level.

- `F3(F2(F1(col,arg1),arg2),arg3)`

General Functions

- These functions work with any data type and pertain to using nulls.
- NVL (expr1, expr2)
- NVL2 (expr1, expr2, expr3)
- NULLIF (expr1, expr2)
- COALESCE (expr1, expr2, ..., exprn)

General function

Function	Description
NVL	Converts a null value to an actual value
NVL2	If expr1 is not null, NVL2 returns expr2. If expr1 is null, NVL2 returns expr3. The argument expr1 can have any data type.
NULLIF	Compares two expressions and returns null if they are equal, or the first expression if they are not equal
COALESCE	Returns the first non-null expression in the expression list

NVL Function

- Converts a null to an actual value.
- Data types that can be used are date, character, and number.
- e.g.

```
NVL(comm, 0)
```

NVL2 function

- ```
SELECT ename, sal, comm,
 NVL2(comm, 'SAL+COMM', 'SAL') income
FROM emp WHERE deptno IN (20, 30)
```
- ```
SELECT ename, sal, comm,  
       NVL2(comm, SAL+COMM, SAL) income  
FROM emp WHERE deptno IN (20, 30)
```

Using the NULLIF Function

- `SELECT first_name, LENGTH(first_name)"expr1",
last_name, LENGTH(last_name)"expr2",
NULLIF(LENGTH(first_name),LENGTH(last_name)) result
FROM employees;`

Using the COALESCE Function

- The advantage of the COALESCE function over the NVL function is that the COALESCE function can take multiple alternate values.
- If the first expression is not null, it returns that expression; otherwise, it does a COALESCE of the remaining expressions.

Using COALESCE Functions

```
1. SELECT COALESCE (NULL, NULL , 'NOT NULL' , NULL )  
   test from dual;
```

```
TEST  
-----  
NOT NULL
```

```
2. SELECT ename,  
   COALESCE(comm, sal, 10) comm  
   FROM emp  
   ORDER BY comm;
```

Conditional Expressions

- Provide the use of IF-THEN-ELSE logic within a SQL statement
- Use two methods:
 - - CASE expression
 - - DECODE function

The CASE Expression

- Facilitates conditional inquiries by doing the work of an IF-THEN-ELSE statement:

```
CASE expr WHEN Compare _expr1 THEN  
    return_expr1  
    [WHEN Compare _expr2 THEN return_expr2  
    WHEN Compare _exprn THEN return_exprn  
    ELSE else_expr]  
END
```

Case

```
SELECT last_name, job_id, salary,  
CASE job_id WHEN 'IT_PROG' THEN 1.10*salary  
    WHEN 'ST_CLERK' THEN 1.15*salary  
    WHEN 'SA_REP' THEN 1.20*salary  
    ELSE salary  
END "REVISED_SALARY"  
FROM employees
```

Decode Function

The DECODE function decodes *expression* after *comparing it to each search value*.

If the expression is the same as *search*, *result* is returned.

If the default value is omitted, a null value is returned where a search value does not match any of the result values.

Decode Function

```
• SELECT last_name, job_id, salary,  
  DECODE(job_id, 'IT_PROG', 1.10*salary,  
          'ST_CLERK', 1.15*salary,  
          'SA_REP', 1.20*salary,  
          salary) REVISED_SALARY  
FROM employees;
```

Aggregating data Using group functions

Group Function

- Group functions operate on sets of rows to give one result per group
- EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
	7000
10	4400

The maximum
salary in
the EMPLOYEES
table

24000

Types of Group Functions

- AVG-Number Type
- SUM-Number Type
- COUNT
- MAX
- MIN

FOR ANY DATA TYPE

- These functions accept single parameter & are not overloaded

Types Of Group Functions

Function	Description
AVG ([DISTINCT ALL] <i>n</i>)	Average value of <i>n</i> , ignoring null values
COUNT ({* [DISTINCT ALL] <i>expr</i> })	Number of rows, where <i>expr</i> evaluates to something other than null (count all selected rows using *, including duplicates and row with nulls)
SUM ([DISTINCT ALL] <i>n</i>)	Sum values of <i>n</i> , ignoring null values
MAX ([DISTINCT ALL] <i>expr</i>)	Maximum value of <i>expr</i> , ignoring null values
MIN ([DISTINCT ALL] <i>expr</i>)	Minimum value of <i>expr</i> , ignoring null values

Syntax

```
SELECT [column,] group_function(column), ...  
FROM table  
[WHERE condition]  
[GROUP BY column]  
[ORDER BY column];
```

Tips About Groups

- The Oracle server on it's own arranges output in ascending order when group by clause
- All group functions ignore NULL values except COUNT with *

Example

```
SELECT MIN(sal), SUM(sal)
FROM emp
  where job LIKE '%SA%'
/
```

Or

```
MIN(distinct sal),MAX(distinct sal);
COUNT(*),COUNT(),COUNT(DISTINCT expr);
```

Types Of Group Function

- You can nest a Function within a Function.
- ```
SELECT AVG(NVL(commission_pct, 0))
 FROM employees;
```

## Creating Groups of Data

- Until now, all group functions have treated the table as one large group of information.
- At times, you need to divide the table of information into smaller groups.
- This can be done by using the GROUP BY clause.

## Syntax Of Group By

```
SELECT column, group_function(column)
FROM table
[WHERE condition]
[GROUP BY group_by_expression]
[ORDER BY column]
```

## Using the GROUP BY Clause

- The GROUP BY column does not have to be in the SELECT list.

```
SELECT AVG(sal)
FROM emp
GROUP BY deptno;
```

\* But vice versa not allowed

## Using the GROUP BY Clause Contd.

```
• SELECT deptno, AVG(sal)
FROM emp
GROUP BY deptno;
```

Will this work ?

```
SELECT department_id, COUNT(last_name)
FROM employees;
```



## Grouping by More Than One Column

```
SQL> select deptno,job,avg(sal)
 from emp
 group by deptno,job;
```

## Illegal Queries Using Group Functions

- You cannot use group functions in the WHERE clause.
- You use the HAVING clause to restrict groups.

```
• SELECT empdeptno,MIN(sal)
 FROM emp
 WHERE MIN(sal)>1250
 GROUP BY deptno
```

## Using Having Clause

- The Oracle server performs the following steps when you use the HAVING clause:
  1. Rows are grouped.
  2. The group function is applied to the group.
  3. The groups that match the criteria in the HAVING clause are displayed.
- The HAVING clause can precede the GROUP BY clause

## Using Having Clause

You can Even use subquery in Having.

e.g. Find Max of Salary of those Emp whose Max sal is less than Maximum Sal from Dept 20

```

Oracle SQL*Plus
File Edit Search Options Help
SQL> ed
Write file afiedt.buf

1 select deptno,max(sal)
2 from emp
3 group by deptno
SQL> /

DEPTNO MAX(SAL)

10 5000
20 3000
30 2850

SQL> select max(sal)
2 from emp
3 having max(sal)>1000
4 /

MAX(SAL)

5000

SQL> ed
Write file afiedt.buf

1 select max(sal)
2 from emp
3 having max(sal)<3000
SQL> /

no rows selected

SQL> select deptno,max(sal)
2 from emp
3 group by deptno
4 having max(sal)<(select max(sal) from emp where deptno=20);

DEPTNO MAX(SAL)

30 2850

SQL>

```

- Manipulating Data

## Data Manipulation Language

- A DML statement is executed when you:
  - Add new rows to a table
  - Modify existing rows in a table
  - Remove existing rows from a table

Logical unit of DML is a 'Transaction'

## Insert

- To add a new row to the table 'Insert' Statement is issued.
- The INSERT statement syntax -

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

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

## Guidelines

- 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.
- Enclose character and date values within single quotation marks.
- Explicitly use NULL Or '' if no col. names are provided

## Caution

- Be careful about inserting data with these constraints:-
- Mandatory value missing for a NOT NULL column
- Duplicate value violates uniqueness constraint
- Foreign key constraint violated
- CHECK constraint violated
- Data type mismatch
- Value too wide to fit in column

## Inserting Specific Date Values

```
INSERT INTO emp
VALUES (114, 'Den', 'ACCOUNT', 7902
, TO_DATE('FEB 3, 1999', 'MON DD, YYYY')
, 1000, 100, 30)
/
```

## Using Substitution Variable

- Use & substitution in a SQL statement to prompt for values.
- & is a placeholder for the variable value

```
SQL> insert into dept(deptno,dname,loc)
 2 values(&deptno,'&dname','&loc');
```

## Copying Rows from Another Table

- Write your INSERT statement with a subquery.
- Do not use the VALUES clause.
- Match the number of columns in the INSERT clause to those in the subquery.

## Inserting Based On Another Table



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> INSERT INTO EMPHISTORY(EMPNO,ENAME,JOB,SAL,COMM)
2 SELECT EMPNO,ENAME,JOB,SAL,COMM
3 FROM EMP
4 WHERE SAL>2000;

6 rows created.

SQL>
```

## ? Subquery In INSERT Statement



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> set line 160
SQL> select *
2 from emp;

 EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

7369 SMITH CLERK 7902 17-DEC-80 800 0 20
7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30
7521 WARD SALESMAN 7698 22-FEB-81 1250 0 30
7566 JONES MANAGER 7839 02-APR-81 2975 0 20
7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30
7698 BLAKE MANAGER 7839 01-MAY-81 2850 0 30
7782 CLARK MANAGER 7839 09-JUN-81 2450 0 10
7788 SCOTT ANALYST 7566 19-APR-87 3000 0 30
7839 KING PRESIDENT 7839 17-MAY-81 5000 0 10
7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30
7876 ADAMS CLERK 7788 23-MAY-87 1100 0 20

 EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

7900 JONES CLERK 7698 03-DEC-81 950 0 30
7902 FORD ANALYST 7566 03-DEC-81 3000 0 30
7934 MILLER CLERK 7782 23-JAN-82 1300 0 10

14 rows selected.

SQL> INSERT INTO
2 /
3 /
4 /
5 /
6 /
7 /
8 /
9 /
10 /
11 /
12 /
13 /
14 /
15 /
16 /
17 /
18 /
19 /
20 /
21 /
22 /
23 /
24 /
25 /
26 /
27 /
28 /
29 /
30 /
31 /
32 /
33 /
34 /
35 /
36 /
37 /
38 /
39 /
40 /
41 /
42 /
43 /
44 /
45 /
46 /
47 /
48 /
49 /
50 /
51 /
52 /
53 /
54 /
55 /
56 /
57 /
58 /
59 /
60 /
61 /
62 /
63 /
64 /
65 /
66 /
67 /
68 /
69 /
70 /
71 /
72 /
73 /
74 /
75 /
76 /
77 /
78 /
79 /
80 /
81 /
82 /
83 /
84 /
85 /
86 /
87 /
88 /
89 /
90 /
91 /
92 /
93 /
94 /
95 /
96 /
97 /
98 /
99 /
100 /
101 /
102 /
103 /
104 /
105 /
106 /
107 /
108 /
109 /
110 /
111 /
112 /
113 /
114 /
115 /
116 /
117 /
118 /
119 /
120 /
121 /
122 /
123 /
124 /
125 /
126 /
127 /
128 /
129 /
130 /
131 /
132 /
133 /
134 /
135 /
136 /
137 /
138 /
139 /
140 /
141 /
142 /
143 /
144 /
145 /
146 /
147 /
148 /
149 /
150 /
151 /
152 /
153 /
154 /
155 /
156 /
157 /
158 /
159 /
160 /
161 /
162 /
163 /
164 /
165 /
166 /
167 /
168 /
169 /
170 /
171 /
172 /
173 /
174 /
175 /
176 /
177 /
178 /
179 /
180 /
181 /
182 /
183 /
184 /
185 /
186 /
187 /
188 /
189 /
190 /
191 /
192 /
193 /
194 /
195 /
196 /
197 /
198 /
199 /
200 /
201 /
202 /
203 /
204 /
205 /
206 /
207 /
208 /
209 /
210 /
211 /
212 /
213 /
214 /
215 /
216 /
217 /
218 /
219 /
220 /
221 /
222 /
223 /
224 /
225 /
226 /
227 /
228 /
229 /
230 /
231 /
232 /
233 /
234 /
235 /
236 /
237 /
238 /
239 /
240 /
241 /
242 /
243 /
244 /
245 /
246 /
247 /
248 /
249 /
250 /
251 /
252 /
253 /
254 /
255 /
256 /
257 /
258 /
259 /
260 /
261 /
262 /
263 /
264 /
265 /
266 /
267 /
268 /
269 /
270 /
271 /
272 /
273 /
274 /
275 /
276 /
277 /
278 /
279 /
280 /
281 /
282 /
283 /
284 /
285 /
286 /
287 /
288 /
289 /
290 /
291 /
292 /
293 /
294 /
295 /
296 /
297 /
298 /
299 /
300 /
301 /
302 /
303 /
304 /
305 /
306 /
307 /
308 /
309 /
310 /
311 /
312 /
313 /
314 /
315 /
316 /
317 /
318 /
319 /
320 /
321 /
322 /
323 /
324 /
325 /
326 /
327 /
328 /
329 /
330 /
331 /
332 /
333 /
334 /
335 /
336 /
337 /
338 /
339 /
340 /
341 /
342 /
343 /
344 /
345 /
346 /
347 /
348 /
349 /
350 /
351 /
352 /
353 /
354 /
355 /
356 /
357 /
358 /
359 /
360 /
361 /
362 /
363 /
364 /
365 /
366 /
367 /
368 /
369 /
370 /
371 /
372 /
373 /
374 /
375 /
376 /
377 /
378 /
379 /
380 /
381 /
382 /
383 /
384 /
385 /
386 /
387 /
388 /
389 /
390 /
391 /
392 /
393 /
394 /
395 /
396 /
397 /
398 /
399 /
400 /
401 /
402 /
403 /
404 /
405 /
406 /
407 /
408 /
409 /
410 /
411 /
412 /
413 /
414 /
415 /
416 /
417 /
418 /
419 /
420 /
421 /
422 /
423 /
424 /
425 /
426 /
427 /
428 /
429 /
430 /
431 /
432 /
433 /
434 /
435 /
436 /
437 /
438 /
439 /
440 /
441 /
442 /
443 /
444 /
445 /
446 /
447 /
448 /
449 /
450 /
451 /
452 /
453 /
454 /
455 /
456 /
457 /
458 /
459 /
460 /
461 /
462 /
463 /
464 /
465 /
466 /
467 /
468 /
469 /
470 /
471 /
472 /
473 /
474 /
475 /
476 /
477 /
478 /
479 /
480 /
481 /
482 /
483 /
484 /
485 /
486 /
487 /
488 /
489 /
490 /
491 /
492 /
493 /
494 /
495 /
496 /
497 /
498 /
499 /
500 /
501 /
502 /
503 /
504 /
505 /
506 /
507 /
508 /
509 /
510 /
511 /
512 /
513 /
514 /
515 /
516 /
517 /
518 /
519 /
520 /
521 /
522 /
523 /
524 /
525 /
526 /
527 /
528 /
529 /
530 /
531 /
532 /
533 /
534 /
535 /
536 /
537 /
538 /
539 /
540 /
541 /
542 /
543 /
544 /
545 /
546 /
547 /
548 /
549 /
550 /
551 /
552 /
553 /
554 /
555 /
556 /
557 /
558 /
559 /
560 /
561 /
562 /
563 /
564 /
565 /
566 /
567 /
568 /
569 /
570 /
571 /
572 /
573 /
574 /
575 /
576 /
577 /
578 /
579 /
580 /
581 /
582 /
583 /
584 /
585 /
586 /
587 /
588 /
589 /
590 /
591 /
592 /
593 /
594 /
595 /
596 /
597 /
598 /
599 /
600 /
601 /
602 /
603 /
604 /
605 /
606 /
607 /
608 /
609 /
610 /
611 /
612 /
613 /
614 /
615 /
616 /
617 /
618 /
619 /
620 /
621 /
622 /
623 /
624 /
625 /
626 /
627 /
628 /
629 /
630 /
631 /
632 /
633 /
634 /
635 /
636 /
637 /
638 /
639 /
640 /
641 /
642 /
643 /
644 /
645 /
646 /
647 /
648 /
649 /
650 /
651 /
652 /
653 /
654 /
655 /
656 /
657 /
658 /
659 /
660 /
661 /
662 /
663 /
664 /
665 /
666 /
667 /
668 /
669 /
670 /
671 /
672 /
673 /
674 /
675 /
676 /
677 /
678 /
679 /
680 /
681 /
682 /
683 /
684 /
685 /
686 /
687 /
688 /
689 /
690 /
691 /
692 /
693 /
694 /
695 /
696 /
697 /
698 /
699 /
700 /
701 /
702 /
703 /
704 /
705 /
706 /
707 /
708 /
709 /
710 /
711 /
712 /
713 /
714 /
715 /
716 /
717 /
718 /
719 /
720 /
721 /
722 /
723 /
724 /
725 /
726 /
727 /
728 /
729 /
730 /
731 /
732 /
733 /
734 /
735 /
736 /
737 /
738 /
739 /
740 /
741 /
742 /
743 /
744 /
745 /
746 /
747 /
748 /
749 /
750 /
751 /
752 /
753 /
754 /
755 /
756 /
757 /
758 /
759 /
760 /
761 /
762 /
763 /
764 /
765 /
766 /
767 /
768 /
769 /
770 /
771 /
772 /
773 /
774 /
775 /
776 /
777 /
778 /
779 /
780 /
781 /
782 /
783 /
784 /
785 /
786 /
787 /
788 /
789 /
790 /
791 /
792 /
793 /
794 /
795 /
796 /
797 /
798 /
799 /
800 /
801 /
802 /
803 /
804 /
805 /
806 /
807 /
808 /
809 /
810 /
811 /
812 /
813 /
814 /
815 /
816 /
817 /
818 /
819 /
820 /
821 /
822 /
823 /
824 /
825 /
826 /
827 /
828 /
829 /
830 /
831 /
832 /
833 /
834 /
835 /
836 /
837 /
838 /
839 /
840 /
841 /
842 /
843 /
844 /
845 /
846 /
847 /
848 /
849 /
850 /
851 /
852 /
853 /
854 /
855 /
856 /
857 /
858 /
859 /
860 /
861 /
862 /
863 /
864 /
865 /
866 /
867 /
868 /
869 /
870 /
871 /
872 /
873 /
874 /
875 /
876 /
877 /
878 /
879 /
880 /
881 /
882 /
883 /
884 /
885 /
886 /
887 /
888 /
889 /
890 /
891 /
892 /
893 /
894 /
895 /
896 /
897 /
898 /
899 /
900 /
901 /
902 /
903 /
904 /
905 /
906 /
907 /
908 /
909 /
910 /
911 /
912 /
913 /
914 /
915 /
916 /
917 /
918 /
919 /
920 /
921 /
922 /
923 /
924 /
925 /
926 /
927 /
928 /
929 /
930 /
931 /
932 /
933 /
934 /
935 /
936 /
937 /
938 /
939 /
940 /
941 /
942 /
943 /
944 /
945 /
946 /
947 /
948 /
949 /
950 /
951 /
952 /
953 /
954 /
955 /
956 /
957 /
958 /
959 /
960 /
961 /
962 /
963 /
964 /
965 /
966 /
967 /
968 /
969 /
970 /
971 /
972 /
973 /
974 /
975 /
976 /
977 /
978 /
979 /
980 /
981 /
982 /
983 /
984 /
985 /
986 /
987 /
988 /
989 /
990 /
991 /
992 /
993 /
994 /
995 /
996 /
997 /
998 /
999 /
1000 /
1001 /
1002 /
1003 /
1004 /
1005 /
1006 /
1007 /
1008 /
1009 /
1010 /
1011 /
1012 /
1013 /
1014 /
1015 /
1016 /
1017 /
1018 /
1019 /
1020 /
1021 /
1022 /
1023 /
1024 /
1025 /
1026 /
1027 /
1028 /
1029 /
1030 /
1031 /
1032 /
1033 /
1034 /
1035 /
1036 /
1037 /
1038 /
1039 /
1040 /
1041 /
1042 /
1043 /
1044 /
1045 /
1046 /
1047 /
1048 /
1049 /
1050 /
1051 /
1052 /
1053 /
1054 /
1055 /
1056 /
1057 /
1058 /
1059 /
1060 /
1061 /
1062 /
1063 /
1064 /
1065 /
1066 /
1067 /
1068 /
1069 /
1070 /
1071 /
1072 /
1073 /
1074 /
1075 /
1076 /
1077 /
1078 /
1079 /
1080 /
1081 /
1082 /
1083 /
1084 /
1085 /
1086 /
1087 /
1088 /
1089 /
1090 /
1091 /
1092 /
1093 /
1094 /
1095 /
1096 /
1097 /
1098 /
1099 /
1100 /
1101 /
1102 /
1103 /
1104 /
1105 /
1106 /
1107 /
1108 /
1109 /
1110 /
1111 /
1112 /
1113 /
1114 /
1115 /
1116 /
1117 /
1118 /
1119 /
1120 /
1121 /
1122 /
1123 /
1124 /
1125 /
1126 /
1127 /
1128 /
1129 /
1130 /
1131 /
1132 /
1133 /
1134 /
1135 /
1136 /
1137 /
1138 /
1139 /
1140 /
1141 /
1142 /
1143 /
1144 /
1145 /
1146 /
1147 /
1148 /
1149 /
1150 /
1151 /
1152 /
1153 /
1154 /
1155 /
1156 /
1157 /
1158 /
1159 /
1160 /
1161 /
1162 /
1163 /
1164 /
1165 /
1166 /
1167 /
1168 /
1169 /
1170 /
1171 /
1172 /
1173 /
1174 /
1175 /
1176 /
1177 /
1178 /
1179 /
1180 /
1181 /
1182 /
1183 /
1184 /
1185 /
1186 /
1187 /
1188 /
1189 /
1190 /
1191 /
1192 /
1193 /
1194 /
1195 /
1196 /
1197 /
1198 /
1199 /
1200 /
1201 /
1202 /
1203 /
1204 /
1205 /
1206 /
1207 /
1208 /
1209 /
1210 /
1211 /
1212 /
1213 /
1214 /
1215 /
1216 /
1217 /
1218 /
1219 /
1220 /
1221 /
1222 /
1223 /
1224 /
1225 /
1226 /
1227 /
1228 /
1229 /
1230 /
1231 /
1232 /
1233 /
1234 /
1235 /
1236 /
1237 /
1238 /
1239 /
1240 /
1241 /
1242 /
1243 /
1244 /
1245 /
1246 /
1247 /
1248 /
1249 /
1250 /
1251 /
1252 /
1253 /
1254 /
1255 /
1256 /
1257 /
1258 /
1259 /
1260 /
1261 /
1262 /
1263 /
1264 /
1265 /
1266 /
1267 /
1268 /
1269 /
1270 /
1271 /
1272 /
1273 /
1274 /
1275 /
1276 /
1277 /
1278 /
1279 /
1280 /
1281 /
1282 /
1283 /
1284 /
1285 /
1286 /
1287 /
1288 /
1289 /
1290 /
1291 /
1292 /
1293 /
1294 /
1295 /
1296 /
1297 /
1298 /
1299 /
1300 /
1301 /
1302 /
1303 /
1304 /
1305 /
1306 /
1307 /
1308 /
1309 /
1310 /
1311 /
1312 /
1313 /
1314 /
1315 /
1316 /
1317 /
1318 /
1319 /
1320 /
1321 /
1322 /
1323 /
1324 /
1325 /
1326 /
1327 /
1328 /
1329 /
1330 /
1331 /
1332 /
1333 /
1334 /
1335 /
1336 /
1337 /
1338 /
1339 /
1340 /
1341 /
1342 /
1343 /
1344 /
1345 /
1346 /
1347 /
1348 /
1349 /
1350 /
1351 /
1352 /
1353 /
1354 /
1355 /
1356 /
1357 /
1358 /
1359 /
1360 /
1361 /
1362 /
1363 /
1364 /
1365 /
1366 /
1367 /
1368 /
1369 /
1370 /
1371 /
1372 /
1373 /
1374 /
1375 /
1376 /
1377 /
1378 /
1379 /
1380 /
1381 /
1382 /
1383 /
1384 /
1385 /
1386 /
1387 /
1388 /
1389 /
1390 /
1391 /
1392 /
1393 /
1394 /
1395 /
1396 /
1397 /
1398 /
1399 /
1400 /
1401 /
1402 /
1403 /
1404 /
1405 /
1406 /
1407 /
1408 /
1409 /
1410 /
1411 /
1412 /
1413 /
1414 /
1415 /
1416 /
1417 /
1418 /
1419 /
1420 /
1421 /
1422 /
1423 /
1424 /
1425 /
1426 /
1427 /
1428 /
1429 /
1430 /
1431 /
1432 /
1433 /
1434 /
1435 /
1436 /
1437 /
1438 /
1439 /
1440 /
1441 /
1442 /
1443 /
1444 /
1445 /
1446 /
1447 /
1448 /
1449 /
1450 /
1451 /
1452 /
1453 /
1454 /
1455 /
1456 /
1457 /
1458 /
1459 /
1460 /
1461 /
1462 /
1463 /
1464 /
1465 /
1466 /
1467 /
1468 /
1469 /
1470 /
1471 /
1472 /
1473 /
1474 /
1475 /
1476 /
1477 /
1478 /
1479 /
1480 /
1481 /
1482 /
1483 /
1484 /
1485 /
1486 /
1487 /
1488 /
1489 /
1490 /
1491 /
1492 /
1493 /
1494 /
1495 /
1496 /
1497 /
1498 /
1499 /
1500 /
1501 /
1502 /
1503 /
1504 /
1505 /
1506 /
1507 /
1508 /
1509 /
1510 /
1511 /
1512 /
1513 /
1514 /
1515 /
1516 /
1517 /
1518 /
1519 /
1520 /
1521 /
1522 /
1523 /
1524 /
1525 /
1526 /
1527 /
1528 /
1529 /
1530 /
1531 /
1532 /
1533 /
1534 /
1535 /
1536 /
1537 /
1538 /
1539 /
1540 /
1541 /
1542 /
1543 /
1544 /
1545 /
1546 /
1547 /
1548 /
1549 /
1550 /
1551 /
1552 /
1553 /
1554 /
1555 /
1556 /
1557 /
1558 /
1559 /
1560 /
1561 /
1562 /
1563 /
1564 /
1565 /
1566 /
1567 /
1568 /
1569 /
1570 /
1571 /
1572 /
1573 /
1574 /
1575 /
1576 /
1577 /
1578 /
1579 /
1580 /
1581 /
1582 /
1583 /
1584 /
1585 /
1586 /
1587 /
1588 /
1589 /
1590 /
1591 /
1592 /
1593 /
1594 /
1595 /
1596 /
1597 /
1598 /
1599 /
1600 /
1601 /
1602 /
1603 /
1604 /
1605 /
1606 /
1607 /
1608 /
1609 /
1610 /
1611 /
1612 /
1613 /
1614 /
1615 /
1616 /
1617 /
1618 /
1619 /
1620 /
1621 /
1622 /
1623 /
1624 /
1625 /
1626 /
1627 /
1628 /
1629 /
1630 /
1631 /
1632 /
1633 /
1634 /
1635 /
1636 /
1637 /
1638 /
1639 /
1640 /
1641 /
1642 /
1643 /
1644 /
1645 /
1646 /
1647 /
1648 /
1649 /
1650 /
1651 /
1652 /
1653 /
1654 /
1655 /
1656 /
1657 /
1658 /
1659 /
1660 /
1661 /
1662 /
1663 /
1664 /
1665 /
1666 /
1667 /
1668 /
1669 /
1670 /
1671 /
1672 /
1673 /
1674 /
1675 /
1676 /
1677 /
1678 /
1679 /
1680 /
1681 /
1682 /
1683 /
1684 /
1685 /
1686 /
1687 /
1688 /
1689 /
1690 /
1691 /
1692 /
1693 /
1694 /
1695 /
1696 /
1697 /
1698 /
1699 /
1700 /
1701 /
1702 /
1703 /
1704 /
1705 /
1706 /
1707 /
1708 /
1709 /
1710 /
1711 /
1712 /
1713 /
1714 /
1715 /
1716 /
1717 /
1718 /
1719 /
1720 /
1721 /
1722 /
1723 /
1724 /
1725 /
1726 /
1727 /
1728 /
1729 /
1730 /
1731 /
1732 /
1733 /
1734 /
1735 /
1736 /
1737 /
1738 /
1739 /
1740 /
1741 /
1742 /
1743 /
1744 /
1745 /
1746 /
1747 /
1748 /
1749 /
1750 /
1751 /
1752 /
1753 /
1754 /
1755 /
1756 /
1757 /
1758 /
1759 /
1760 /
1761 /
1762 /
1763 /
1764 /
1765 /
1766 /
1767 /
1768 /
1769 /
1770 /
1771 /
1772 /
1773 /
1774 /
1775 /
1776 /
1777 /
1778 /
1779 /
1780 /
1781 /
1782 /
1783 /
1784 /
1785 /
1786 /
1787 /
1788 /
1789 /
1790 /
1791 /
1792 /
1793 /
1794 /
1795 /
1796 /
1797 /
1798 /
1799 /
1800 /
1801 /
1802 /
1803 /
1804 /
1805 /
1806 /
1807 /
1808 /
1809 /
1810 /
1811 /
1812 /
1813 /
1814 /
1815 /
1816 /
1817 /
1818 /
1819 /
1820 /
1821 /
1822 /
1823 /
1824 /
1825 /
1826 /
1827 /
1828 /
1829 /
1830 /
1831 /
1832 /
1833 /
1834 /
1835 /
1836 /
1837 /
1838 /
1839 /
1840 /
1841 /
1842 /
1843 /
1844 /
1845 /
1846 /
1847 /
1848 /
1849 /
1850 /
1851 /
1852 /
1853 /
1854 /
1855 /
1856 /
1857 /
1858 /
1859 /
1860 /
1861 /
1862 /
1863 /
1864 /
1865 /
1866 /
1867 /
1868 /
1869 /
1870 /
1871 /
1872 /
1873 /
1874 /
1875 /
1876 /
1877 /
1878 /
1879 /
1880 /
1881 /
1882 /
1883 /
1884 /
1885 /
1886 /
1887 /
1888 /
1889 /
1890 /
1891 /
1892 /
1893 /
1894 /
1895 /
1896 /
1897 /
1898 /
1899 /
1900 /
1901 /
1902 /
1903 /
1904 /
1905 /
1906 /
1907 /
1908 /
1909 /
1910 /
1911 /
1912 /
1913 /
1914 /
1915 /
1916 /
1917 /
1918 /
1919 /
1920 /
1921 /
1922 /
1923 /
1924 /
1925 /
1926 /
1927 /
1928 /
1929 /
1930 /
1931 /
1932 /
1933 /
1934 /
1935 /
1936 /
1937 /
1938 /
1939 /
1940 /
1941 /
1942 /
1943 /
1944 /
1945 /
1946 /
1947 /
1948 /
1949 /
1950 /
1951 /
1952 /
1953 /
1954 /
1955 /
1956 /
1957 /
1958 /
1959 /
1960 /
1961 /
1962 /
1963 /
1964 /
1965 /
1966 /
1967 /
1968 /
1969 /
1970 /
1971 /
1972 /
1973 /
1974 /
1975 /
1976 /
1977 /
1978 /
1979 /
1980 /
1981 /
1982 /
1983 /
1984 /
1985 /
1986 /
1987 /
1988 /
1989 /
1990 /
1991 /
1992 /
1993 /
1994 /
1995 /
1996 /
1997 /
1998 /
1999 /
2000 /
2001 /
2002 /
2003 /
2004 /
2005 /
2006 /
2007 /
2008 /
2009 /
2010 /
2011 /
2012 /
2013 /
2014 /
2015 /
2016 /
2017 /
2018 /
2019 /
2020 /
2021 /
2022 /
2023 /
2024 /
2025 /
2026 /
2027 /
2028 /
2029 /
2030 /
2031 /
2032 /
2033 /
2034 /
2035 /
2036 /
2037 /
2038 /
2039 /
2040 /
2041 /
2042 /
2043 /
2044 /
2045 /
2046 /
2047 /
2048 /
2049 /
2050 /
2051 /
2052 /
2053 /
2054 /
2055 /
2056 /
2057 /
2058 /
2059 /
2060 /
2061 /
2062 /
2063 /
2064 /
2065 /
2066 /
2067 /
2068 /
2069 /
2070 /
2071 /
2072 /
2073 /
2074 /
2075 /
2076 /
2077 /
2078 /
2079 /
2080 /
2081 /
2082 /
2083 /
2084 /
2085 /
2086 /
2087 /
2088 /
2089 /
2090 /
2091 /
2092 /
2093 /
2094 /
2095 /
2096 /
2097 /
2
```



## Using the WITH CHECK OPTION Keyword on DML Statements

- A subquery is used to identify the table and columns of the DML statement.
- The WITH CHECK OPTION keyword prohibits you from changing rows that are not in the subquery.

## Example



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> INSERT INTO
2 (SELECT EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO
3 FROM EMP
4 WHERE DEPTNO=10 WITH CHECK OPTION)
5 VALUES(100,'SYDNEY','CLERK',7782,'17-NOV-99',999,10,10)
6 /
1 row created.
SQL> INSERT INTO
2 (SELECT EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO
3 FROM EMP
4 WHERE DEPTNO=10 WITH CHECK OPTION)
5 VALUES(100,'SYDNEY','CLERK',7782,'17-NOV-99',999,10,20)
6 /
ERROR at line 5:
ORA-01402: view WITH CHECK OPTION where-clause violation
SQL> |
```



## Update

- Modify existing rows with the UPDATE statement.

- UPDATE *table*

```
SET column = value [, column = value, ...]
[WHERE condition];
```



## Update

- Updating single row:-

```
SQL> UPDATE DEPT
2 SET DNAME='MAHA_SALES'
3 WHERE DEPTNO=20;
```

1 row updated.



## Updating All Rows In The Table

- SQL> update emphistory  
2 set comm =NULL;

3 rows updated.

## Updating Specific Columns With Subquery

- SQL> update emp  
set job=(select job from emp  
where empid=7902),  
sal=(select sal from emp  
where empno=7876)  
where ename='ADAMS';

## Updating Rows Based on Another Table

```
UPDATE leave_emp
SET tenure= (SELECT sysdate_hiredate
 FROM emp
 WHERE empno= 7902)

/
```

## Updating Rows: Integrity Constraint Error

```
UPDATE emp SET deptno= 9
WHERE department_id = 10;
```

```
update dept set deptno=9
*
```

ERROR at line 1:

```
ORA-02292: integrity constraint (SCOTT.FK_DEPTNO)
violated - child record found
```

## Using Default Values In INSERT & UPDATE

- With the explicit default feature, you can use the DEFAULT keyword as a column value where the column default is desired.
- Takes Default col. Value if no value is supplied to specified column.
- Can be used for INSERT & UPDATE Stmt

## Default

• DEFAULT with INSERT:

```
INSERT INTO dept (depto, dename, loc)
VALUES (300, 'Engineering', DEFAULT);
```

DEFAULT with UPDATE:

```
UPDATE dept SET loc=DEFAULT
WHERE deptno=90;
```

## Removing a Row from a Table

- One or more rows can be deleted by using DELETE Statement
- You can remove existing rows from a table by using the DELETE statement.

```
DELETE [FROM] table
[WHERE condition];
```

\* If no rows are deleted, a message "0 rows deleted." is returned:

## Deleting Rows from a Table

- Specific rows are deleted if you specify the WHERE clause.
- Deleting Rows from a Table

```
DELETE FROM emp
WHERE ename = 'CLARK';
```

1 row deleted.

## Deleting Rows From Table

- All rows in the table are deleted if you omit the WHERE clause.
- e.g- `DELETE FROM emp;`

## Deleting Rows Based on Another Table

- Use subqueries in DELETE statements to remove
- rows from a table based on values from another table.
- `DELETE FROM emp  
WHERE deptno=  
      (SELECT deptno  
      FROM dept  
      WHERE dname LIKE '%ING%');`

[illegible]

## The 'Merge' Statement

- Provides the ability to conditionally update or insert data into a database table
- Performs an UPDATE if the row exists, and an INSERT if it is a new row:
  - - Avoids separate updates
  - - Increases performance and ease of use
  - - Is useful in data warehousing applications

## Merge-Syntax

```
MERGE INTO table_name table_alias
 USING (table|view|sub_query) alias
 ON (join condition)
 WHEN MATCHED THEN
 UPDATE SET
 col1 = col_val1,
 col2 = col2_val
 WHEN NOT MATCHED THEN
 INSERT (column_list)
 VALUES (column_values);
```

## Example

```
MERGE INTO MERG_EMP M
 USING EMP E
 ON (M.EMPNO=E.EMPNO)
 WHEN MATCHED THEN
 UPDATE SET
 M.ENAME=E.ENAME,
 M.JOB=E.JOB,
 M.MGR=E.MGR,
 M.HIREDATE=E.HIREDATE,
 M.SAL=E.SAL,
 M.COMM=E.COMM,
 M.DEPTNO=E.DEPTNO
 WHEN NOT MATCHED THEN
 INSERT VALUES (E.EMPNO,E.ENAME,E.JOB,E.MGR,E.HIREDATE,E.SAL,E.COMM,E.DEPTNO);
```



# Database Transactions

- A transaction is a logical unit of related SQL Statements.
- The DBMS must ensure that a Transaction either completes successfully or not at all
- It must not be intermediate.

## Transaction

Begin when the first DML SQL statement is Executed

End with one of the following events:

- A COMMIT or ROLLBACK statement is issued
- A DDL or DCL statement executes (automatic commit)
- The user exits *iSQL\*Plus*
- The system crashes

## Statements Used

- Commit
- Rollback
- Savepoint

## TCL-Transaction Control Language



- Commit-Makes changes made to the data, Permanent.
- Rollback-Used to undo the changes made to the database till last commit was fired
- Savepoint-Creating a mark for previous action taken on database.

## Advantages of COMMIT and ROLLBACK Statements



With COMMIT and ROLLBACK statements, you can:

- Ensure data consistency
- Preview data changes before making changes permanent
- Group logically related operations

- You can control transaction by using
- Commit/Rollback/Savepoint depending upon whether to make SQL Statement permanent or not.

## Transaction

| Statement                         | Description                                                                                                                                                                                                                                                                                                                                                                        |
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| COMMIT                            | Ends the current transaction by making all pending data changes Permanent                                                                                                                                                                                                                                                                                                          |
| SAVEPOINT <i>name</i>             | Marks a savepoint within the current transaction                                                                                                                                                                                                                                                                                                                                   |
| ROLLBACK                          | ROLLBACK ends the current transaction by discarding all pending data changes                                                                                                                                                                                                                                                                                                       |
| ROLLBACK TO <i>SAVEPOINT name</i> | ROLLBACK TO SAVEPOINT rolls back the current transaction to the specified savepoint, thereby discarding any changes and or savepoints created after the savepoint to which you are rolling back. If you omit the TO SAVEPOINT clause, the ROLLBACK statement rolls back the entire transaction. As savepoints are logical, there is no way to list the savepoints you have created |

## Rolling Back Changes to a Marker

UPDATE...

SAVEPOINT update\_done;

Savepoint created.

INSERT...

ROLLBACK TO update\_done;

Rollback complete.

## Implicit Transaction Processing

- An automatic commit occurs under the following
- circumstances:
  - DDL statement is issued
  - DCL statement is issued
  - Normal exit from *iSQL\*Plus*, without explicitly issuing COMMIT or ROLLBACK statements
- An automatic rollback occurs under an abnormal termination of *iSQL\*Plus* or a system failure.

## State of the Data Before COMMIT or ROLLBACK

- The previous state of the data can be recovered.
- The current user can review the results of the DML operations by using the SELECT statement.
- Other users *cannot view the results of the DML statements* by the current user.
- The affected rows are *locked*; other users *cannot change* the data within the affected rows.

## State of the Data after COMMIT

- Data changes are made permanent in the database.
- The previous state of the data is permanently lost.
- All users can view the results.
- Locks on the affected rows are released; those rows
- are available for other users to manipulate.
- All savepoints are erased.

## Committing Data



```
COMMIT;
Commit complete.
```

## State of the Data After ROLLBACK



- Discard all pending changes by using the ROLLBACK statement:
- Data changes are undone.
- Previous state of the data is restored.
- Locks on the affected rows are released.

```
• DELETE FROM copy_emp;
 22 rows deleted.
ROLLBACK;
Rollback complete
```

## Statement-Level Rollback

- If a single DML statement fails during execution, only that statement is rolled back.
- The Oracle server implements an implicit savepoint.
- All other changes are retained.
- The user should terminate transactions explicitly by executing a COMMIT or ROLLBACK statement.

## Read Consistency

- Read consistency guarantees a consistent view of the data at all times.
- Changes made by one user do not conflict with
- Changes made by another user.
- Read consistency ensures that on the same data:
  - Readers do not wait for writers.
  - Writers do not wait for readers



## Locking

- In an Oracle database, locks:
- Prevent destructive interaction between concurrent transactions
- Require no user action
- Automatically use the lowest level of restrictiveness
- Are held for the duration of the transaction
- Are of two types: explicit locking and implicit locking

## Implicit Locking

- Two lock modes:
  - Exclusive: Locks out other users
  - Share: Allows other users to access
- High level of data concurrency:
  - DML: Table share, row exclusive
  - Queries: No locks required
  - DDL: Protects object definitions
- Locks held until commit or rollback

# DCL

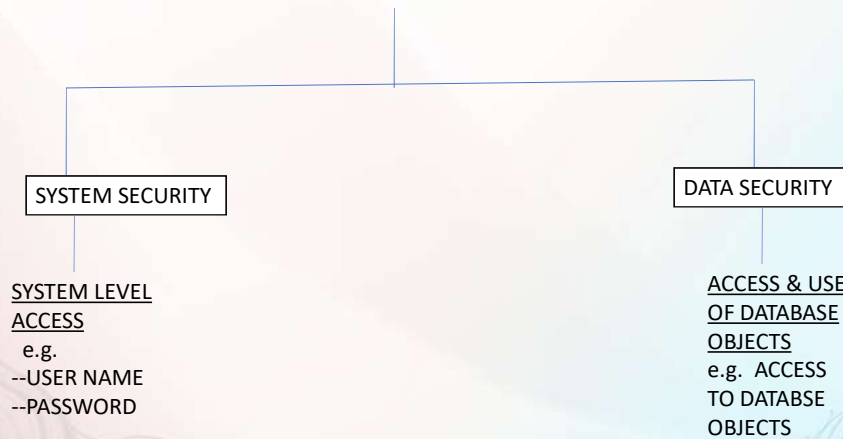
## Data Control Language

- In a multiple-user environment, you want to maintain security of the database access and use.
- Oracle server database can give you security for
  - Access of data.
  - Giving privileges on the data to user or group
  - Revoking privileges back from the user

# Controlling User Access



## Database Security



## System Privileges



- The database administrator has high-level system privileges for tasks such as:
- Typical DBA Privileges

| System Privilege s | Operations Authorized                                                        |
|--------------------|------------------------------------------------------------------------------|
| CREATE USER        | Grantee can create other Oracle users (a privilege required for a DBA role). |
| DROP USER.         | Grantee can drop another user                                                |
| DROP ANY TABLE.    | Grantee can drop a table in any schema                                       |
| BACKUP ANY TABLE   | Grantee can back up any table in any schema with the export utility.         |
| SELECT ANY TABLE   | Grantee can query tables, views, or snapshots in any schema.                 |
| CREATE ANY TABLE   | Grantee can create tables in any schema                                      |

## Creating Users

- The DBA creates users by using the CREATE USER statement.
- User created.  

```
CREATE USER user
IDENTIFIED BY password;
```

Example:-

```
CREATE USER scott
IDENTIFIED BY tiger;
```

## User System Privileges

- Once a user is created, the DBA can grant specific system privileges to a user.
- An application developer, for example, may have the following system privileges:
  - CREATE SESSION
  - CREATE TABLE
  - CREATE SEQUENCE
  - CREATE VIEW
  - CREATE PROCEDURE

## Granting System Privileges

- The DBA can grant a user specific system privileges.
- `GRANT create session, create table, create sequence, create view TO scott;`

Grant succeeded.

## Role

- A role is a named group of related privileges that can be granted to the user.
- This method makes it easier to revoke and maintain privileges.
- A user can have access to several roles, and several users can be assigned the same role.

## Creating and Granting Privileges to a Role

- Create a role  
`CREATE ROLE manager;`
- Grant privileges to a role  
`GRANT create table, create view  
TO manager;`
- Grant a role to users  
`GRANT manager TO DEHAAN, KOCHHAR;`

## Changing Your Password

- The DBA creates your user account and initializes your password.
- You can change your password by using the
- `ALTER USER` statement.

```
ALTER USER scott
IDENTIFIED BY lion;
```

## OBJECT PRIVILEGES

### Object Privileges

- An *object privilege* is a privilege or right to perform a particular action on a specific table, view, sequence, or procedure.

## Object Privileges



| Object Privilege | Table | View | Sequence | Procedure |
|------------------|-------|------|----------|-----------|
|                  | ✓     |      | ✓        |           |
| ALTER            | ✓     | ✓    |          |           |
| DELETE           |       |      |          | ✓         |
| EXECUTE          | ✓     |      |          |           |
| INDEX            | ✓     | ✓    |          |           |
| INSERT           |       |      |          |           |
| REFERENCES       | ✓     | ✓    |          |           |
|                  | ✓     | ✓    | ✓        |           |
| SELECT           | ✓     | ✓    |          |           |
| UPDATE           |       |      |          |           |

## Object Privileges



- Object privileges vary from object to object.
- An owner has all the privileges on the object.
- An owner can give specific privileges on that owner's object.
- ```
GRANT object_priv [(columns)]  
ON object  
TO {user|role|PUBLIC}  
[WITH GRANT OPTION];
```


Granting Object Privileges

- Grant query privileges on the EMPLOYEES table.

```
GRANT select
ON employees
TO sue, rich;
Grant succeeded.
```

- Grant privileges to update specific columns to users and roles.

```
GRANT update (department_name, location_id)
ON departments
TO scott, manager;
Grant succeeded.
```

Using the WITH GRANT OPTION and PUBLIC Keywords

- Give a user authority to pass along privileges.

```
GRANT select, insert
ON departments
TO scott
WITH GRANT OPTION;
Grant succeeded.
```

- Allow all users on the system to query data from Alice's DEPARTMENTS table.

```
GRANT select
ON alice.departments
TO PUBLIC;
Grant succeeded.
```

Using Data Dictionary For Checking Privileges

- `ROLE_SYS_PRIVS` System privileges granted to roles
- `ROLE_TAB_PRIVS` Table privileges granted to roles
- `USER_ROLE_PRIVS` Roles accessible by the user
- `USER_TAB_PRIVS_MADE` Object privileges granted on the user's objects
- `USER_TAB_PRIVS_RECD` Object privileges granted to the user
- `USER_COL_PRIVS_MADE` Object privileges granted on the columns of the user's objects
- `USER_COL_PRIVS_RECD` Object privileges granted to the user on specific columns
- `USER_SYS_PRIVS` Lists system privileges granted to the user

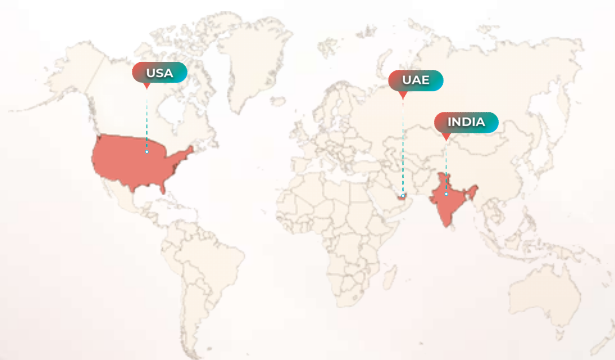
How to Revoke Object Privileges


- You use the `REVOKE` statement to revoke privileges granted to other users.
- Privileges granted to others through the `WITH GRANT OPTION` clause are also revoked.
- `REVOKE {privilege [, privilege...]|ALL}`
`ON object`
`FROM {user[, user...]|role|PUBLIC}`
`[CASCADE CONSTRAINTS];`


Example


- As user Alice, revoke the SELECT and INSERT privileges given to user Scott on the DEPARTMENTS table.
- REVOKE select, insert
ON departments
FROM scott;
Revoke succeeded.


Physical Presence





www.vinsys.com


enquiry@vinsys.com

 Shivaji Niketan, Tejas Society,
Behind Kothrud Bus Stand, Near
Mantri Park, Kothrud, Pune -
411029.

 304, City Tower 2, Near Crown
Plaza, Sheikh Zayed Road. Dubai,
UAE.
PO.Box - 213279

 132 West 31st Street, First
Floor, New York, 10001, USA