VINSYS®

DEliVERiNGSKiLLS
DRiViNGSUCCESS

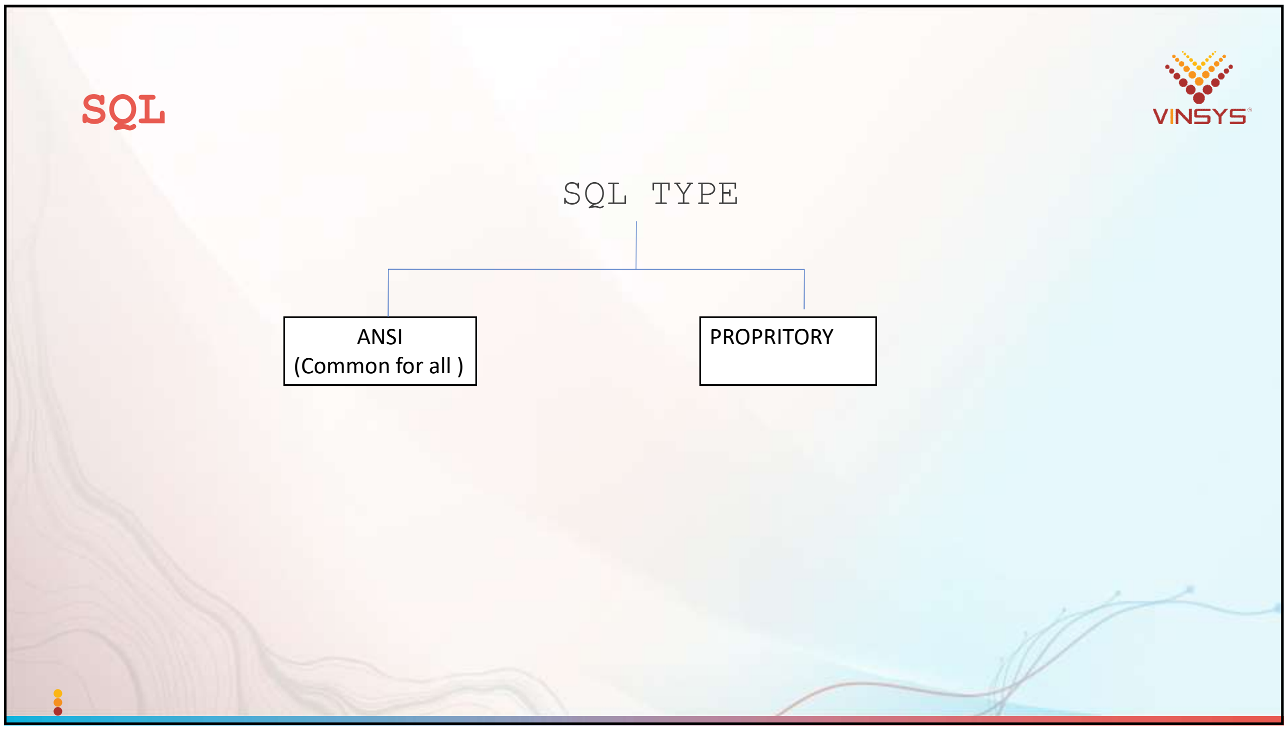
www.VINSYS.COM

VINSYS®

SQL

SQL

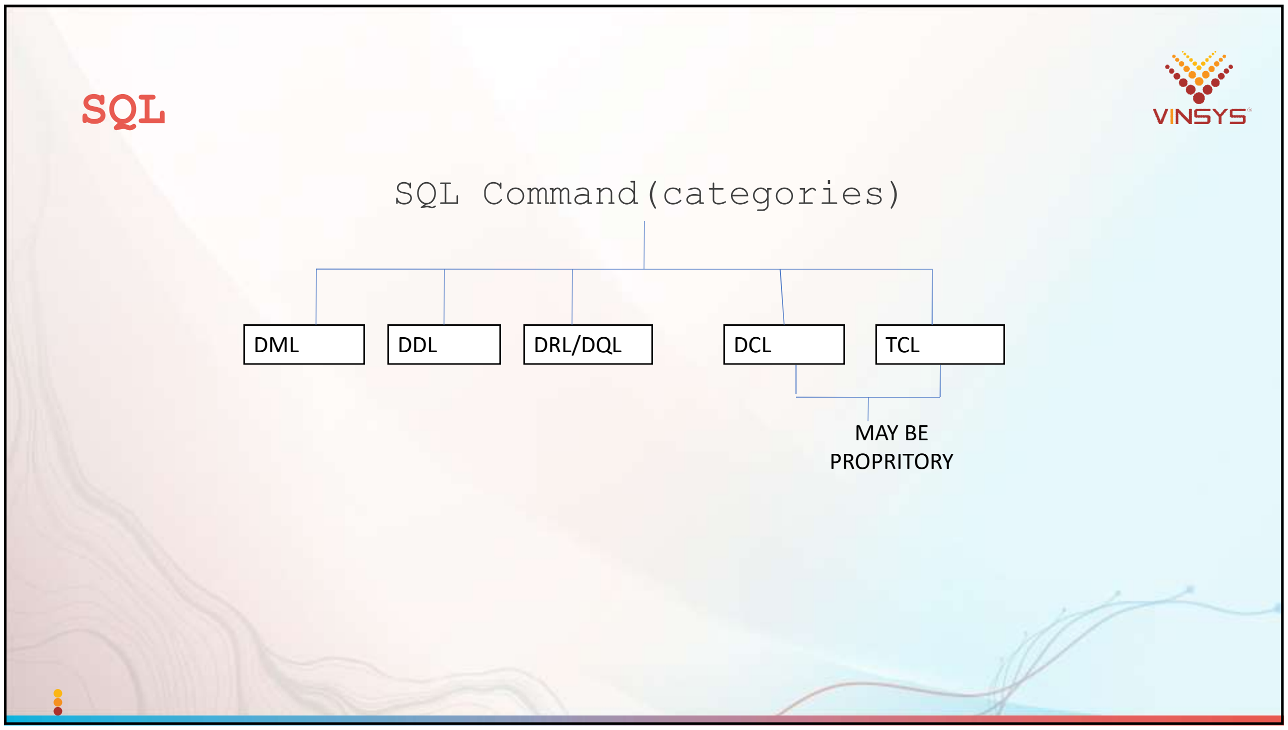
- SQL stands for **S**tructured **Q**uery **L**anguage
- SQL is the language used to communicate with the database(Oracle MS-ACCES,MS SQL Server) to access, manipulate, and control data.



SQL

SQL TYPE

- ANSI (Common for all)
- PROPRITORY



SQL

SQL Command(categories)

- DML
- DDL
- DRL/DQL
- DCL
- TCL

DCL, TCL: MAY BE PROPRITORY

SQL

VINSYS

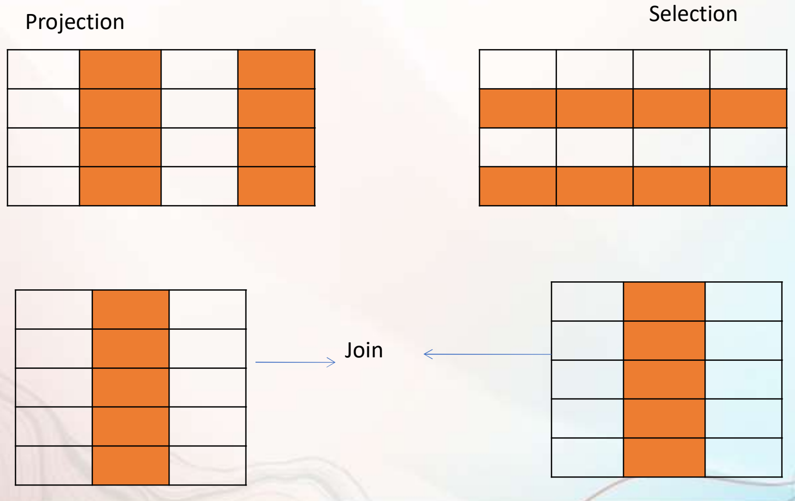
SQL Reports/outputs

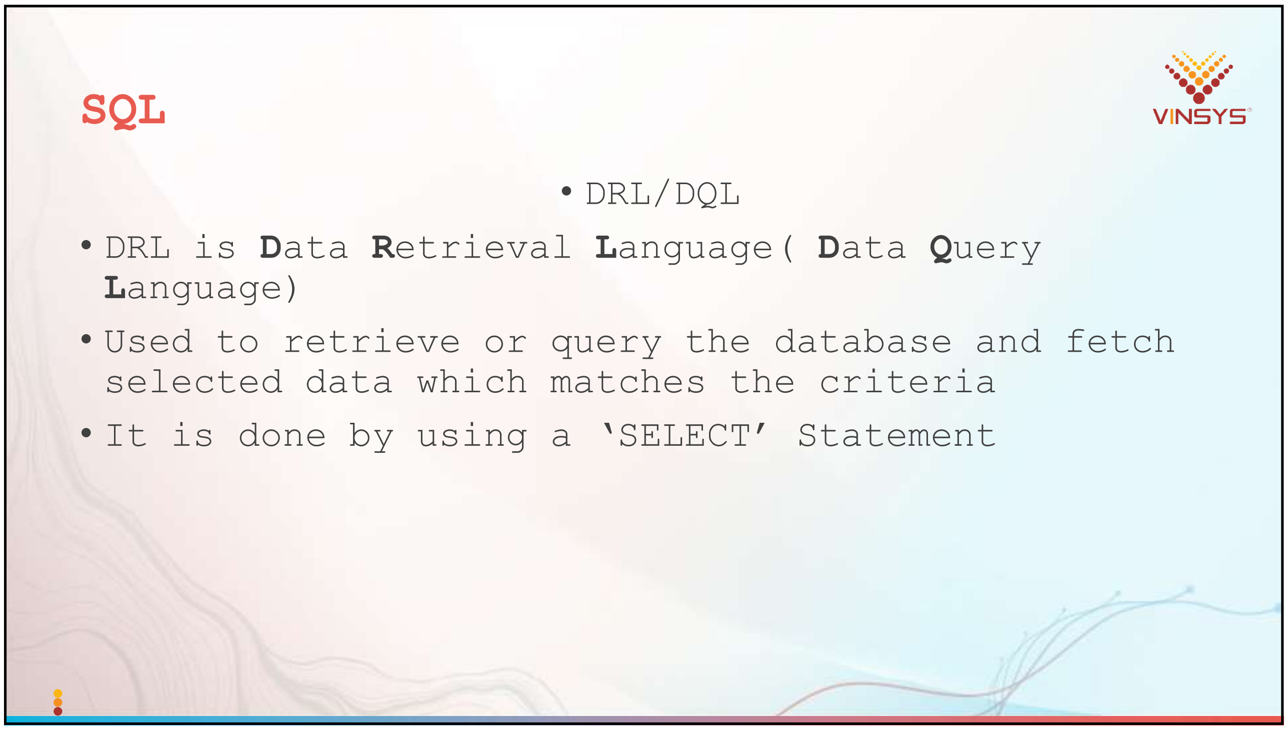
COMPLETE DETAILS	COMPLETE SUMMARY	COMPLETE DETAILS + SUMMARY
(FILTER OR WHERE)	(GROUP BY CLAUSE)	(SUBTOTAL)

SQL

VINSYS

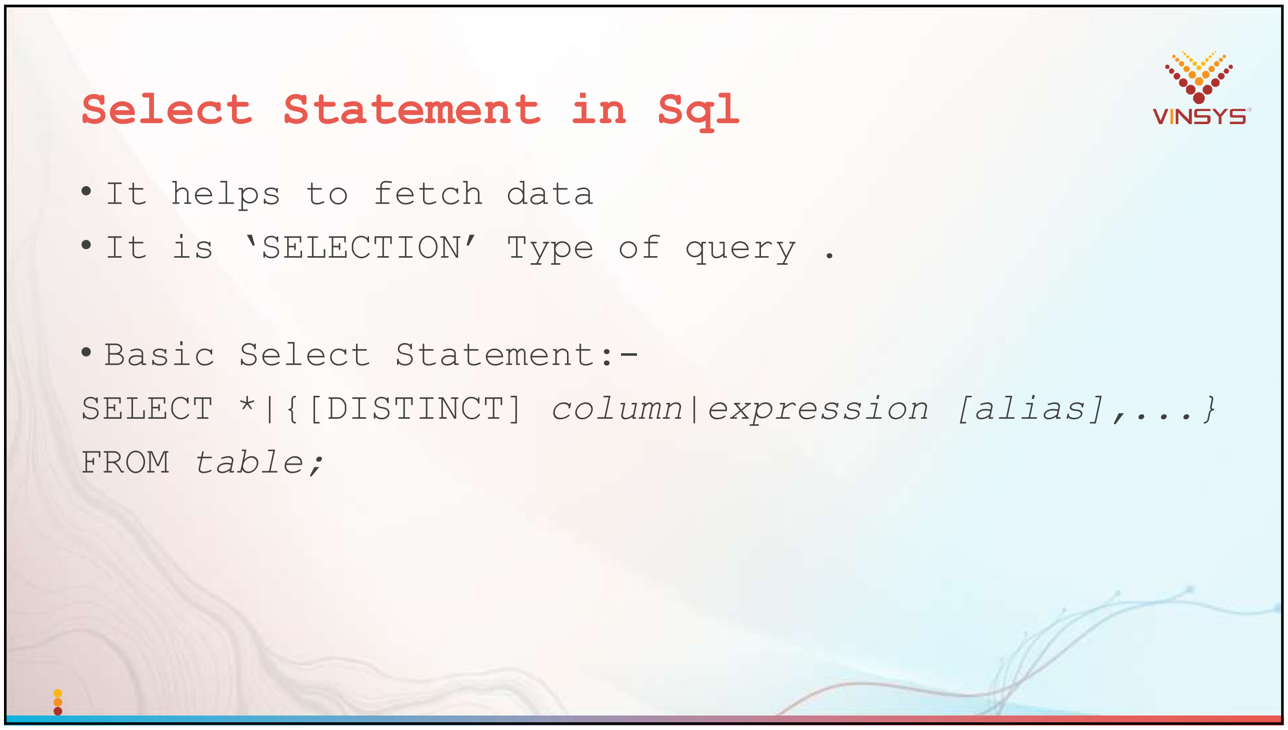
- Capabilities of SQL





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

VINSYS

- 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

VINSYS

- 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;
```

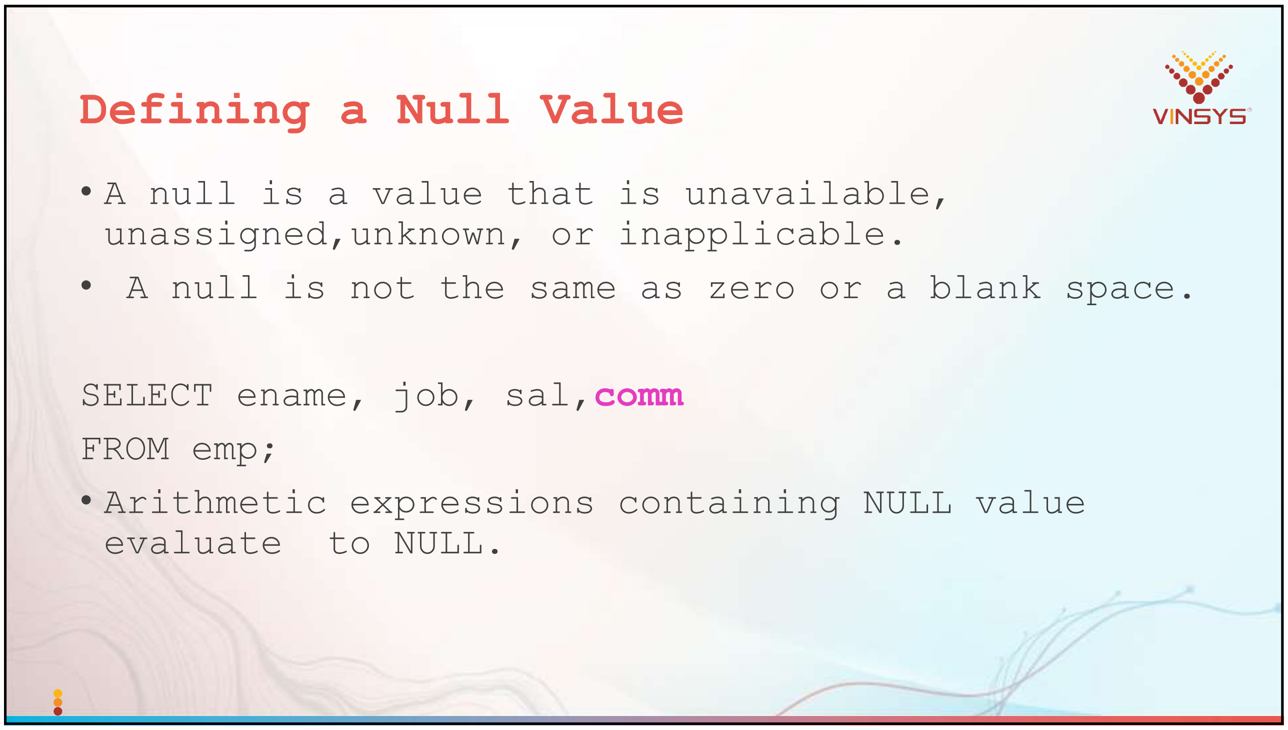
Operator Precedence

*   /   +   -

- 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;
```

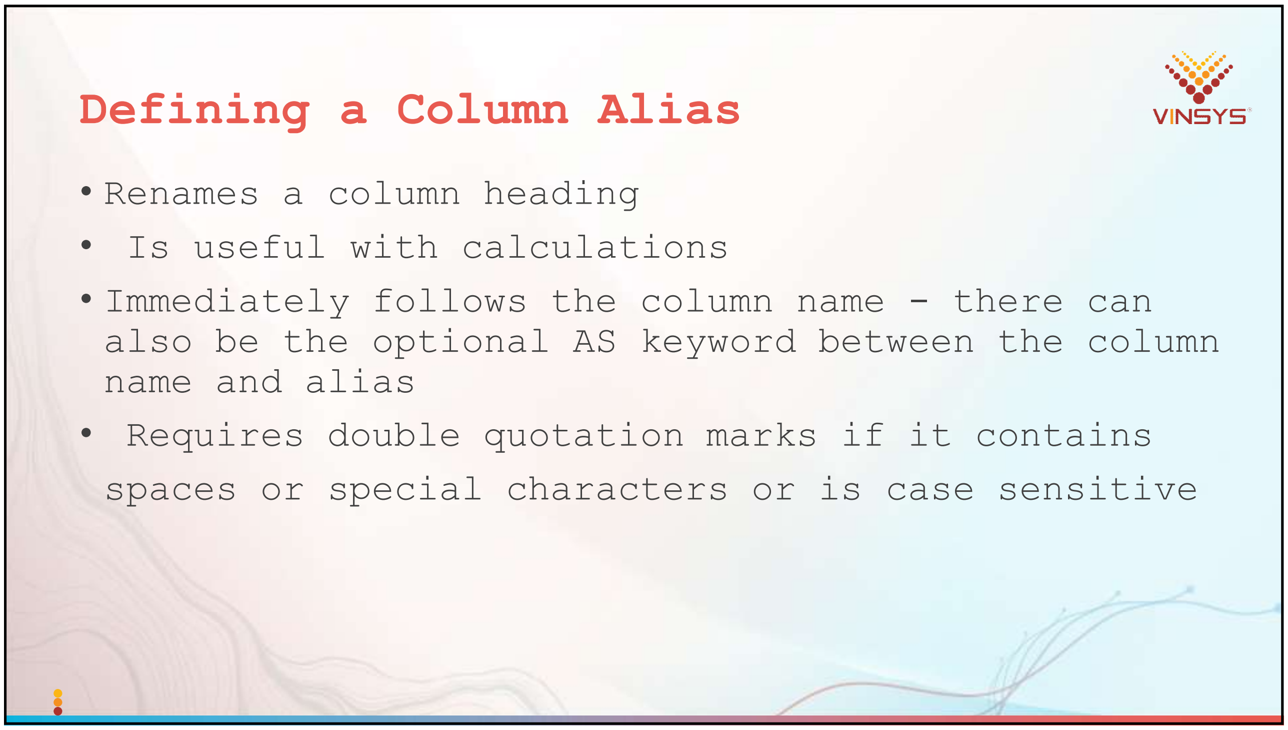


Defining a Null Value

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

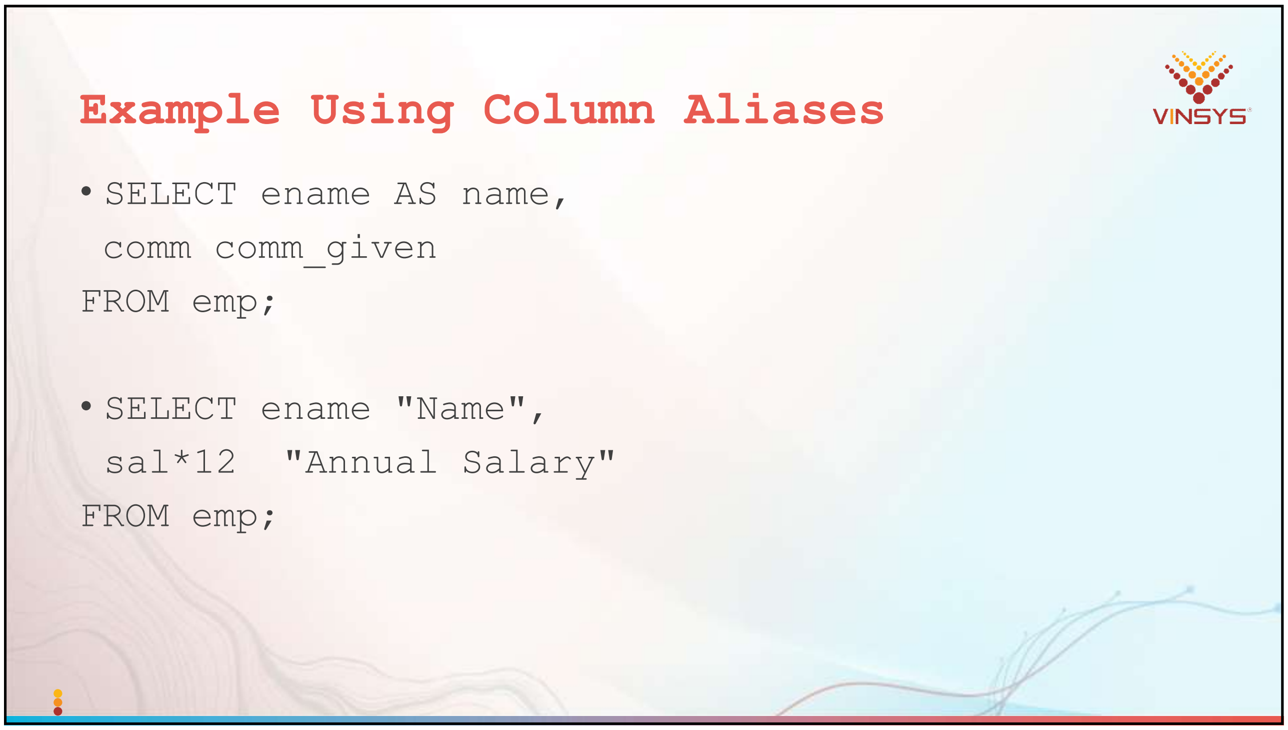
```
SELECT ename, job, sal,comm
FROM emp;
```

- 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

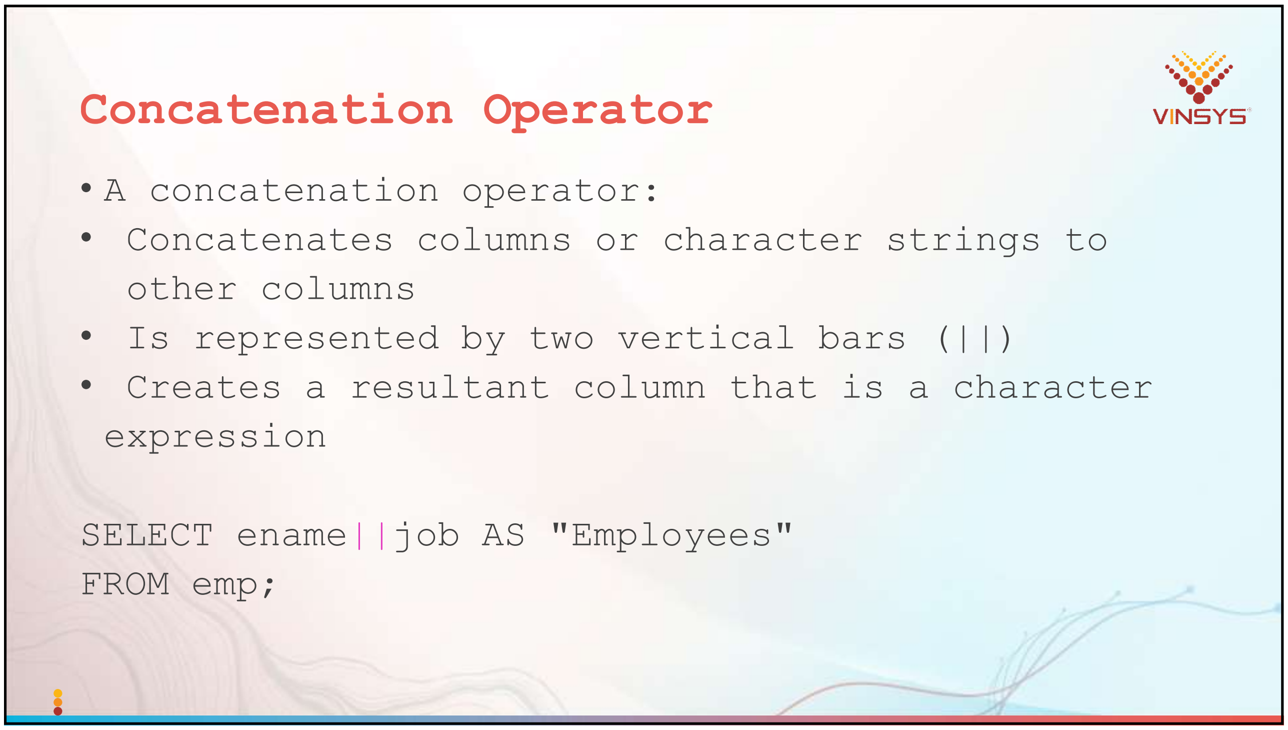


Example Using Column Aliases

- SELECT ename AS name,
  comm comm_given
  FROM emp;

- SELECT ename "Name",
  sal*12 "Annual Salary"
  FROM emp;

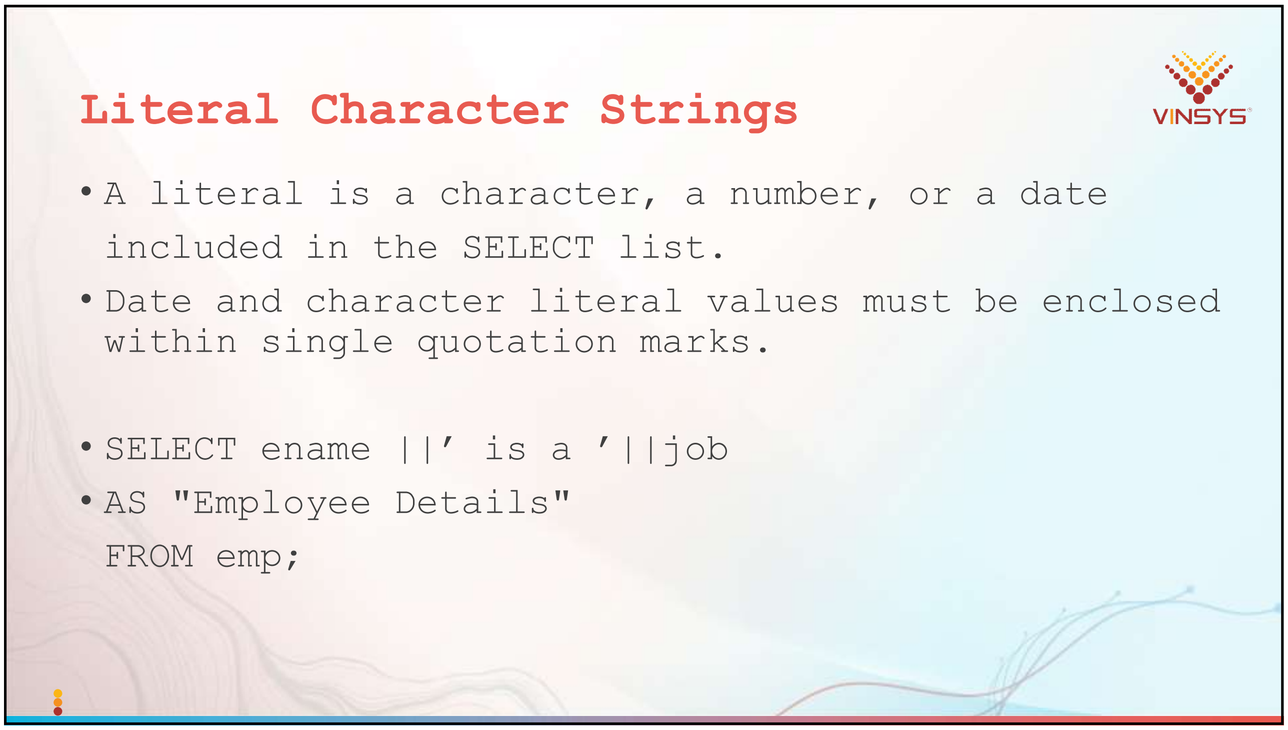
```
SELECT ename AS name,
 comm comm_given
FROM emp;

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

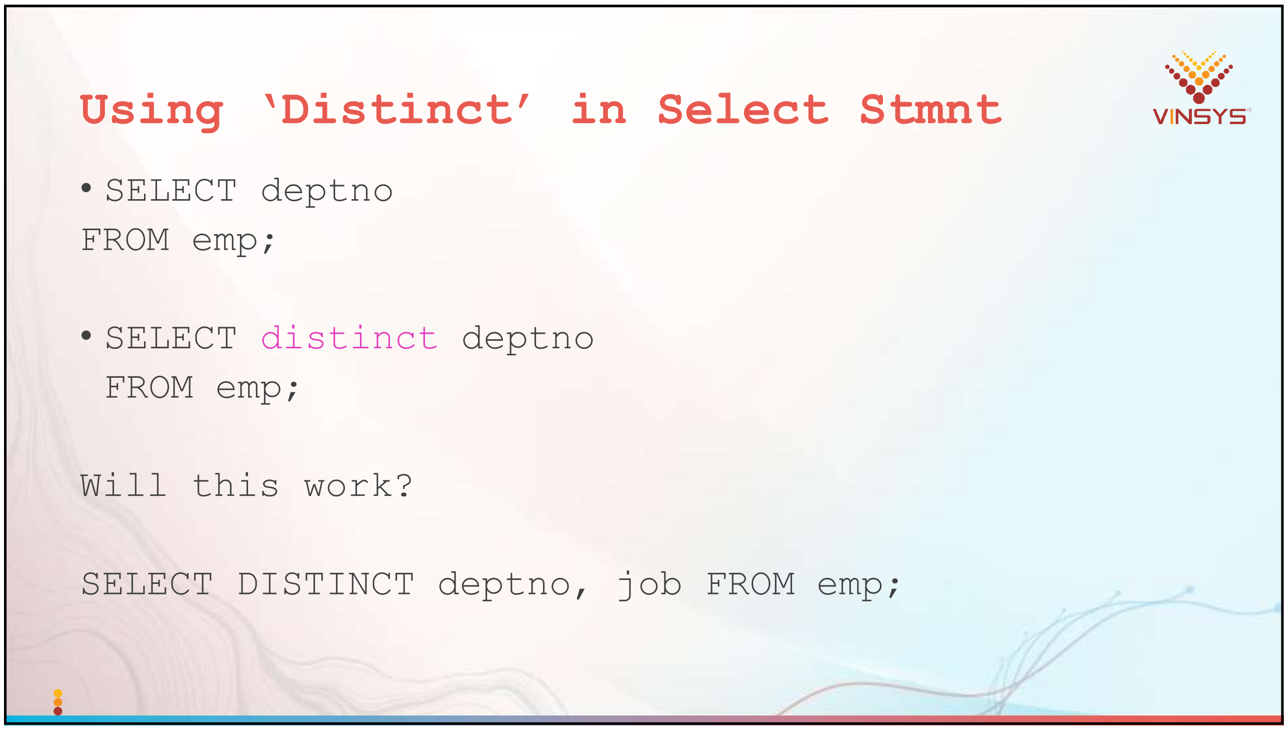
```
SELECT ename||job AS "Employees"
FROM emp;
```



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 Stmnt

- 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)*];

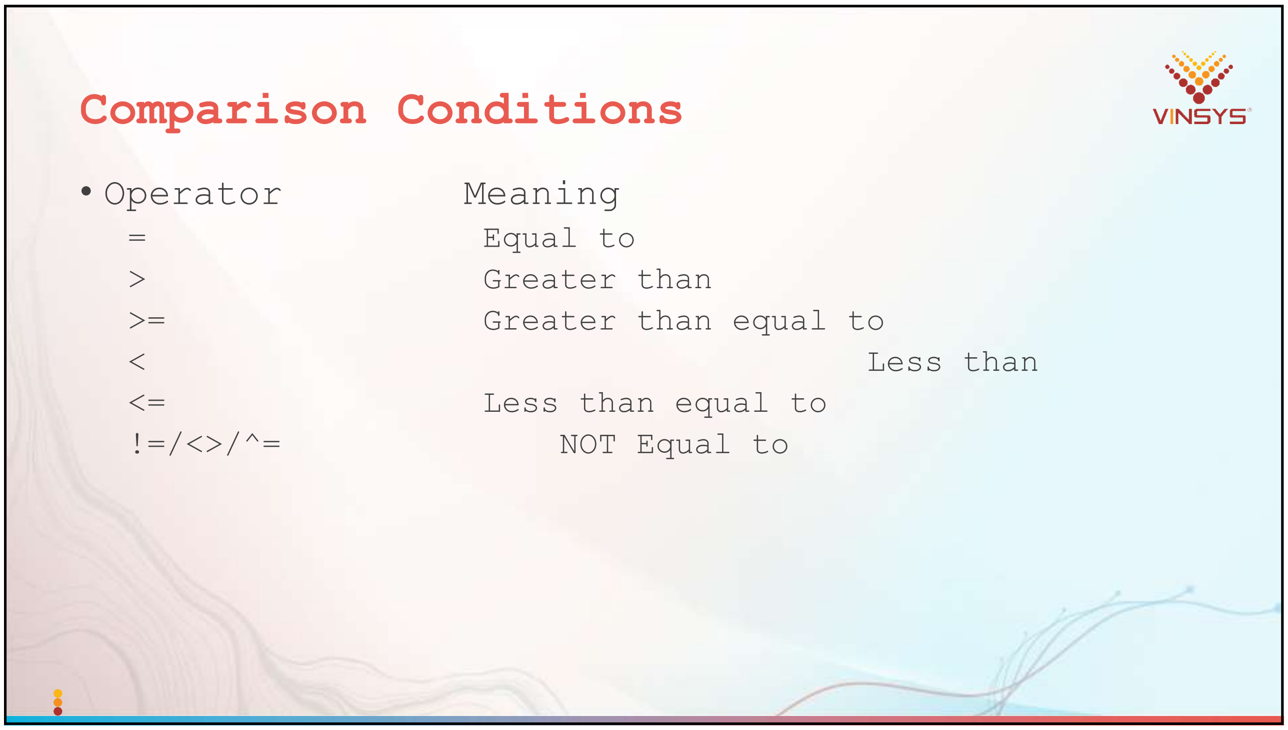
Where clause in 'Select Statement'

- Select dname,loc
- From dept
- Where deptno=30

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

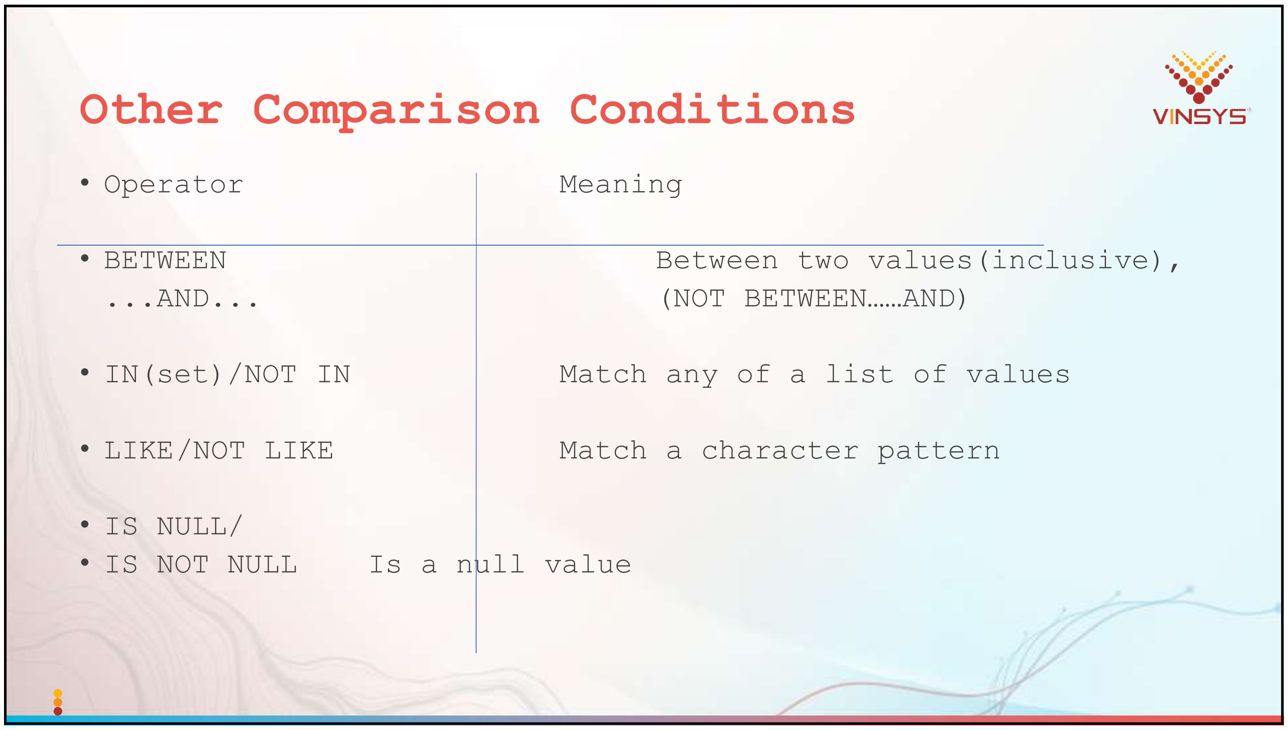
Character Strings And Dates

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



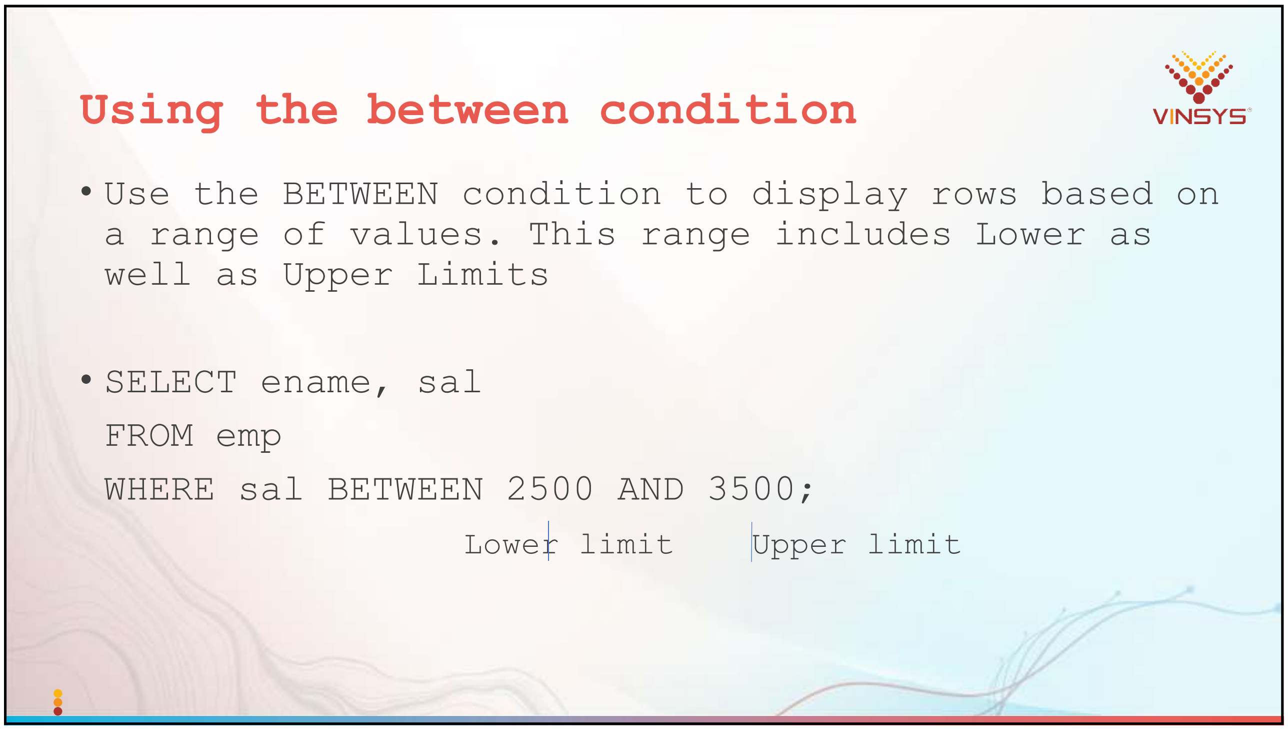
Comparison Conditions

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

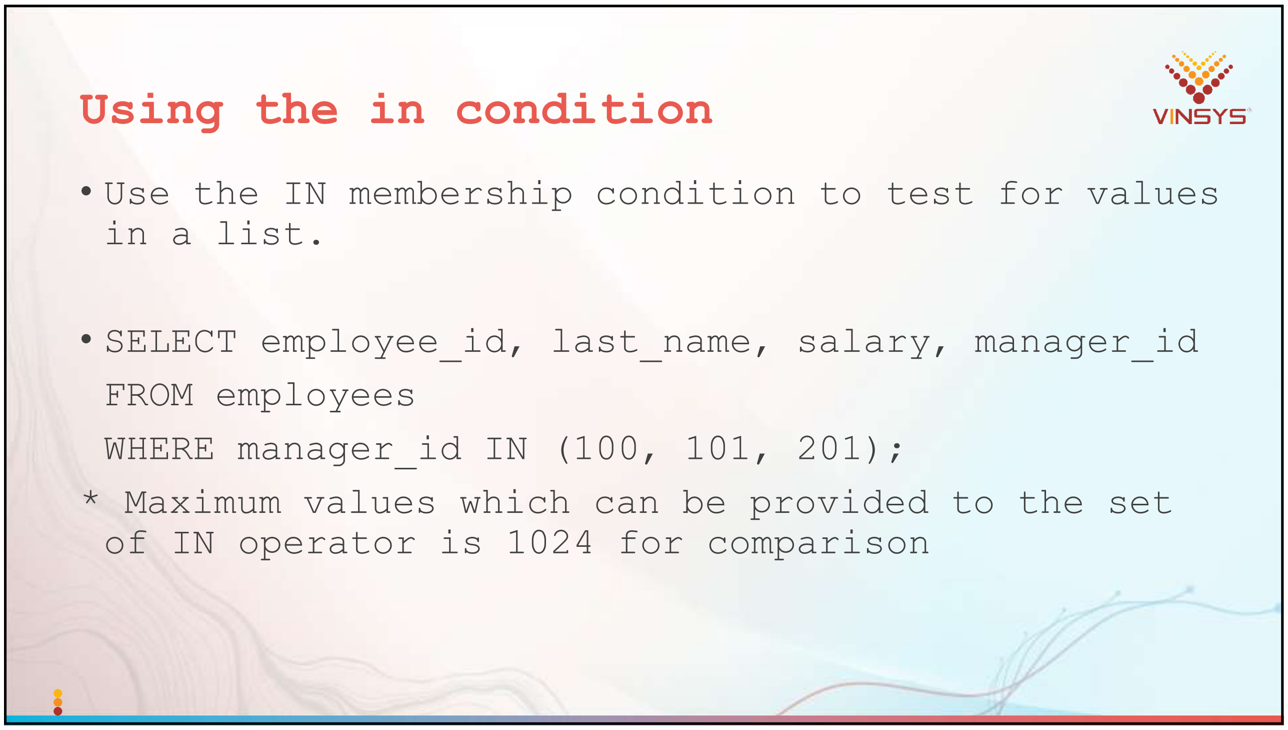


Using the between condition

- 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);
  ```

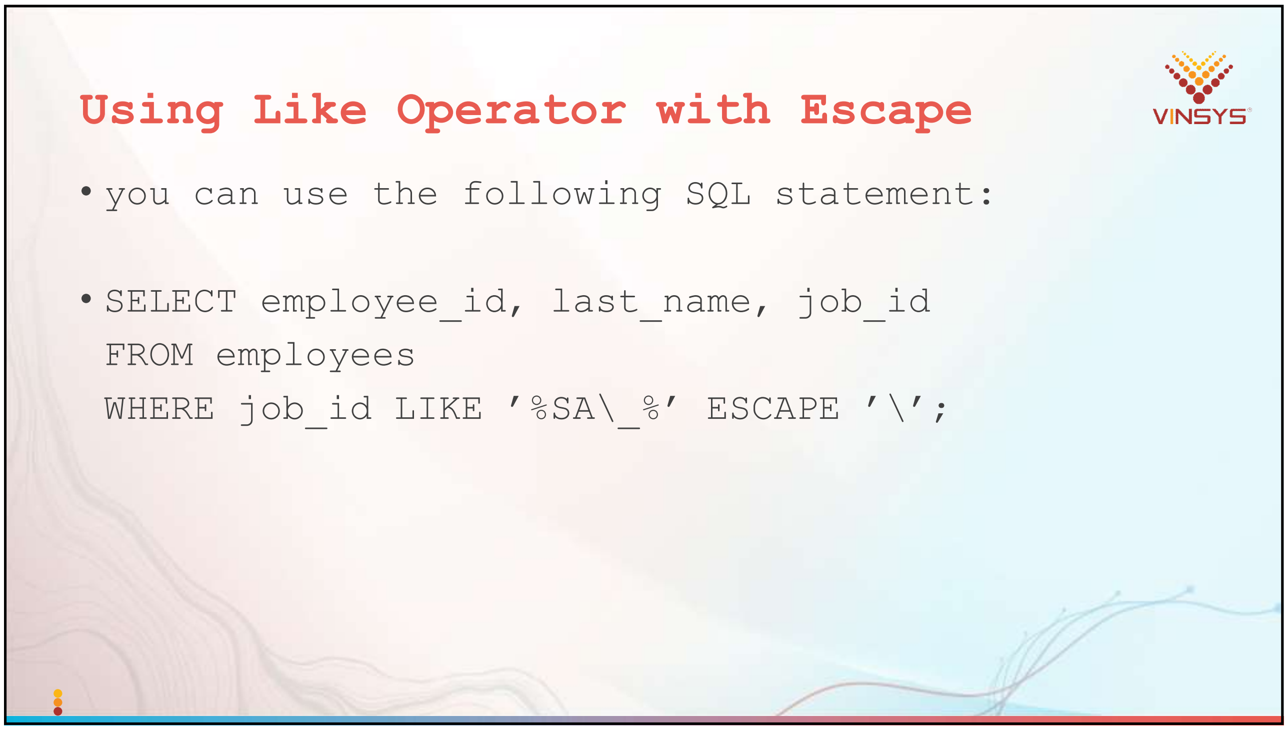
* Maximum values which can be provided to the set of IN operator is 1024 for comparison

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.

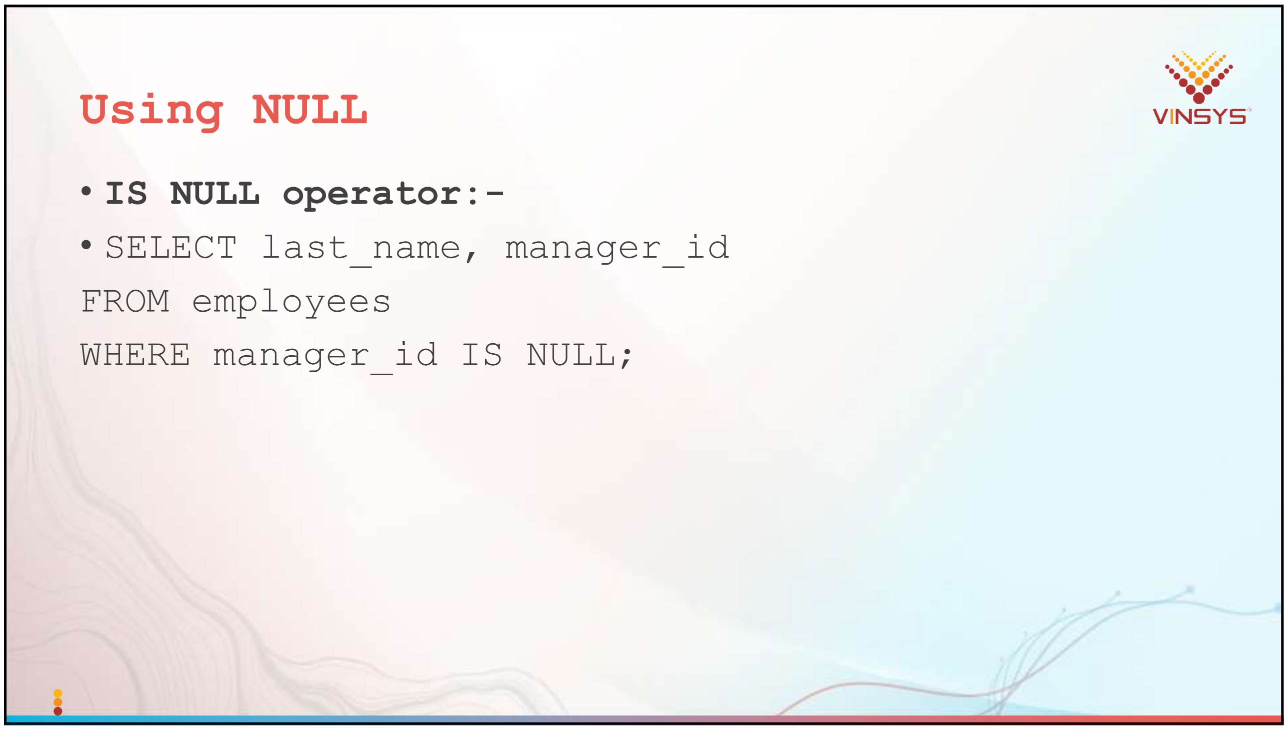
Using 'Escape' Contd.

- **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 ’\’;

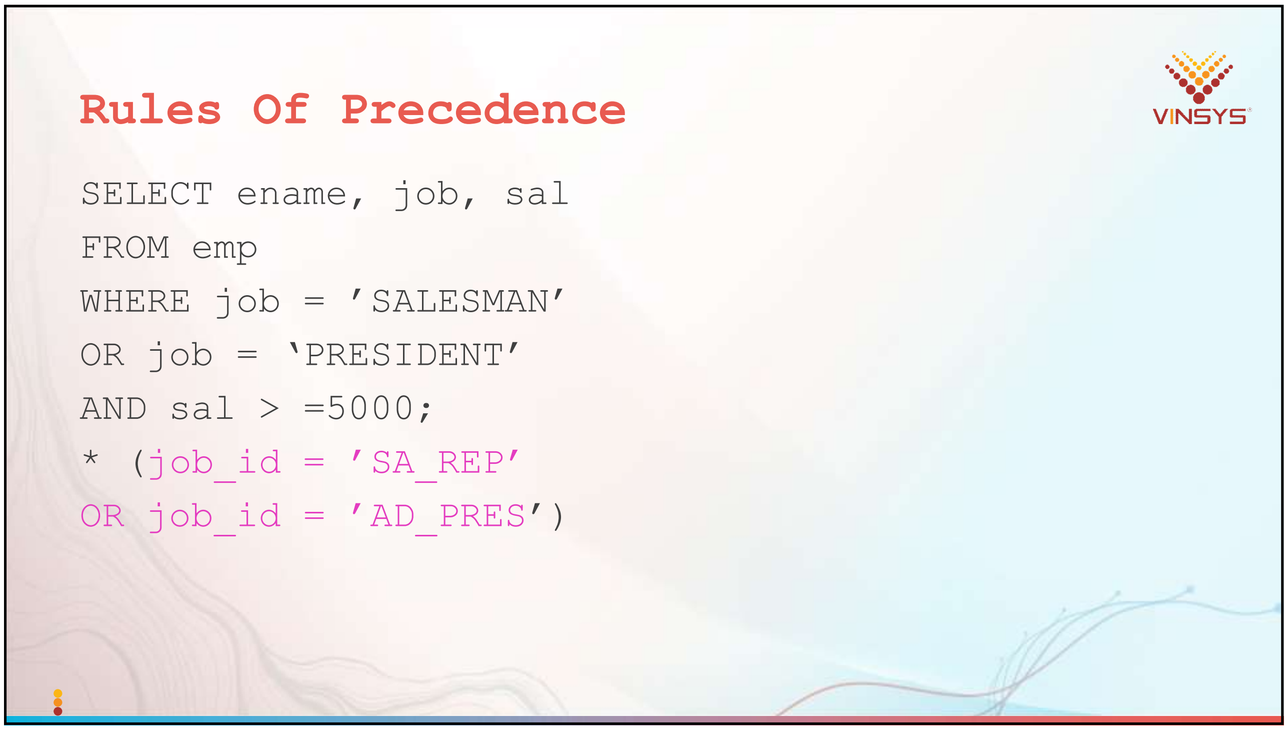


Using NULL

- **IS NULL operator:-**
- SELECT last_name, manager_id

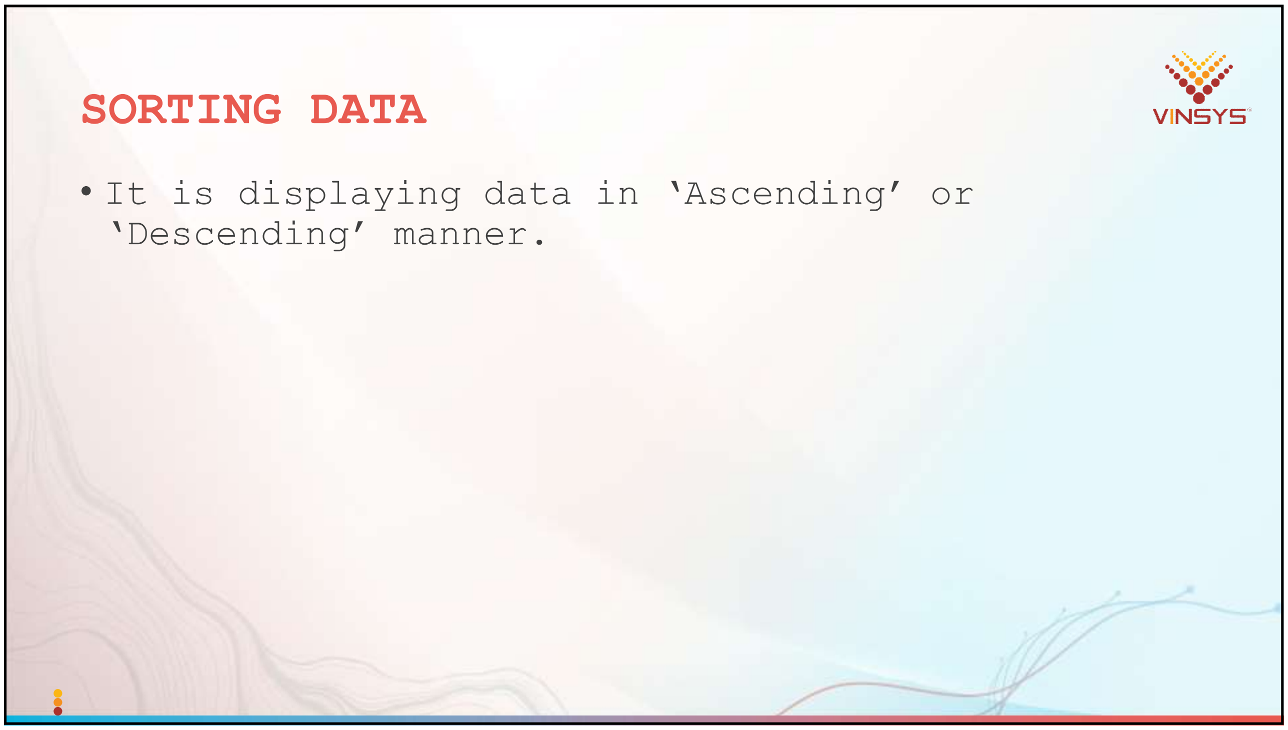
FROM employees

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;
/
```

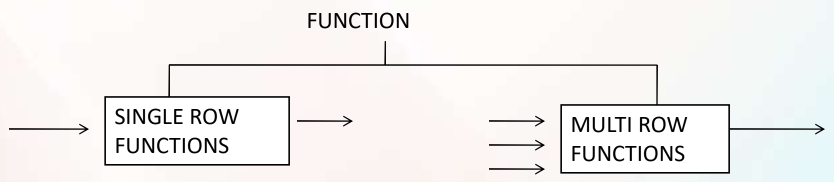
- Max Number of col. Tht can be used in Order By Clause are – 1024.
- **B**est Practices are use of aliases to expression in order by clause

- 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

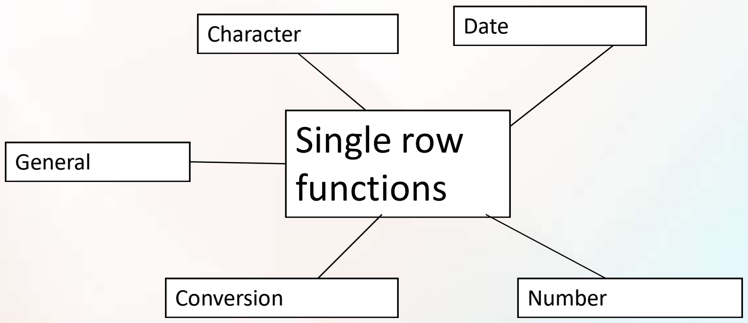


SUMMARY QUERIES

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,...)]*

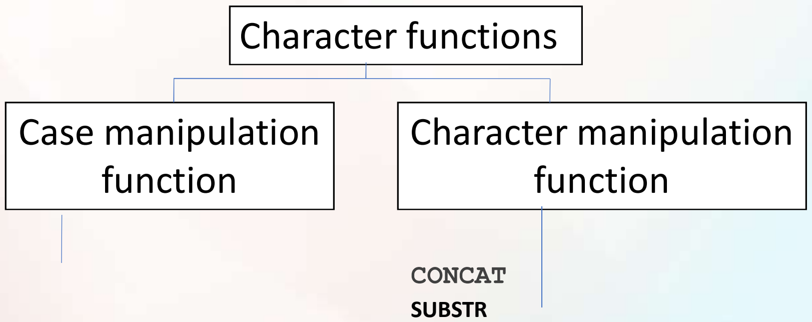
Single Row Function



Character functions

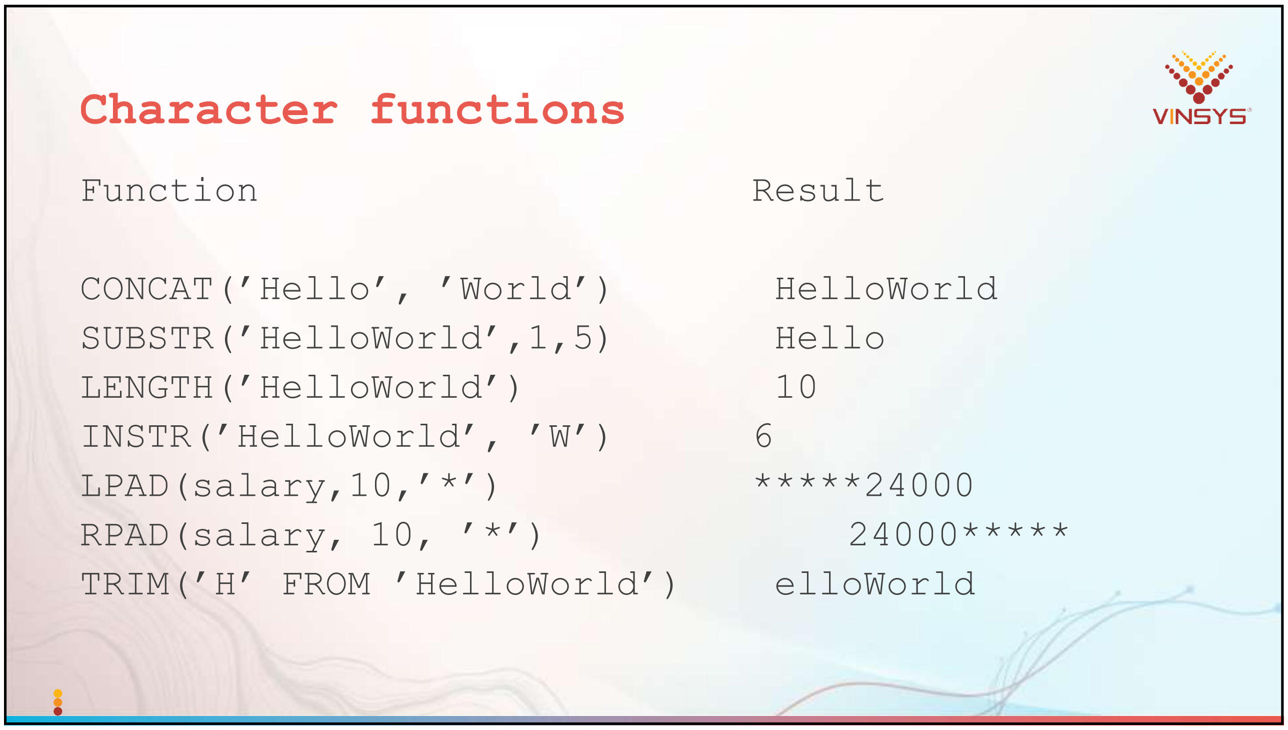
accept character data as input and can return both character and

numeric values.



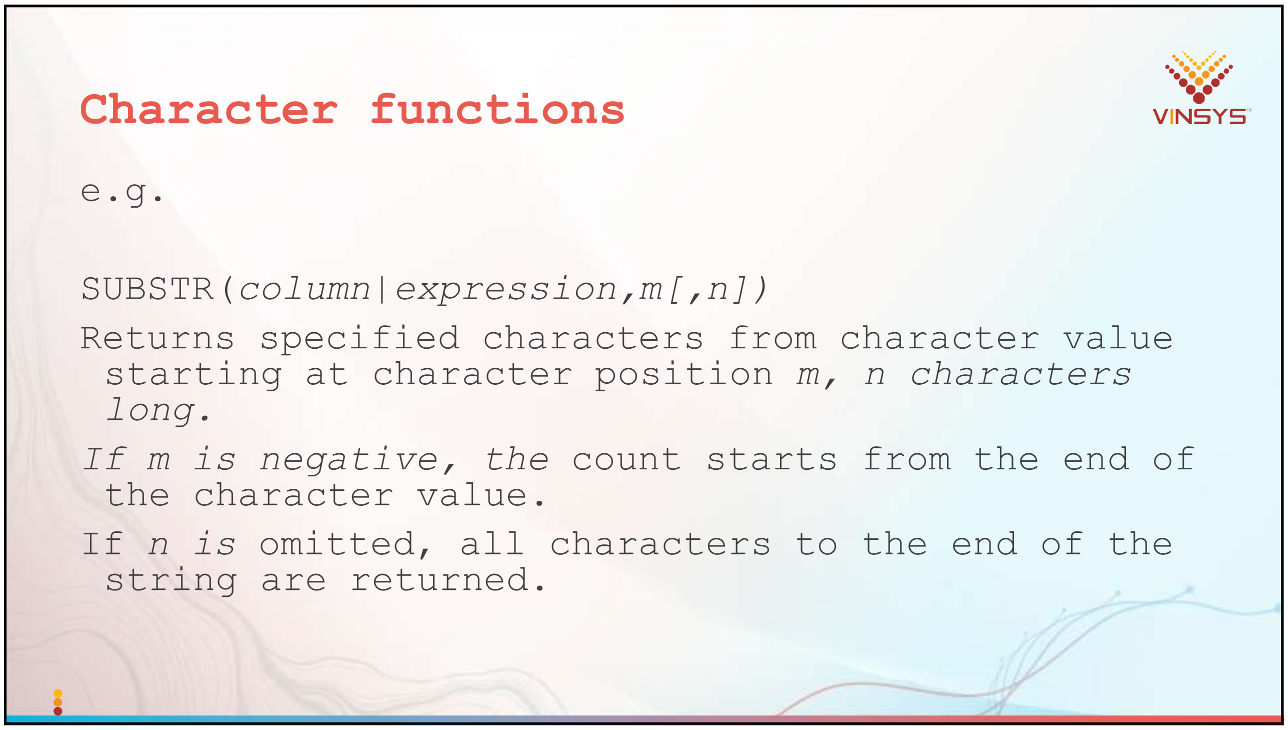
LOWER
UPPER
INITCAP- Oracle Proprietary

CONCAT
SUBSTR
LENGTH
INSTR
LPAD | RPAD
TRIM



Character functions

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



Character functions

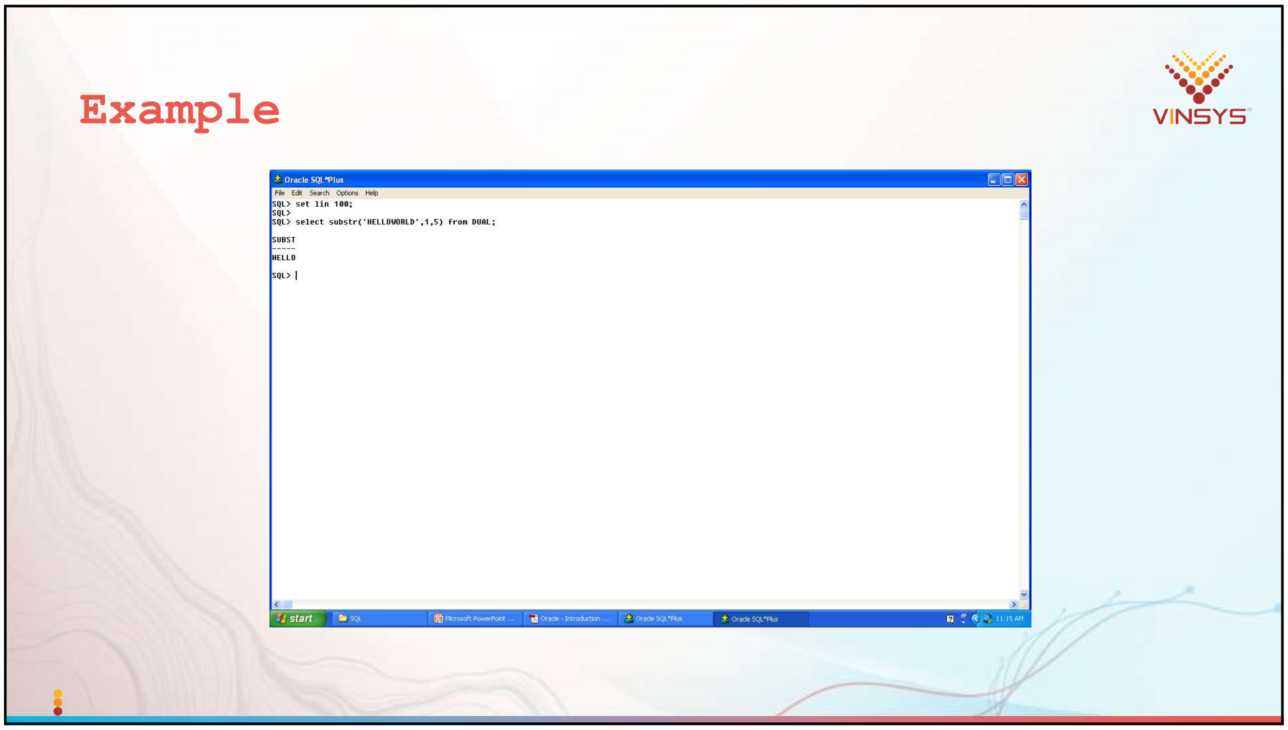
e.g.

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

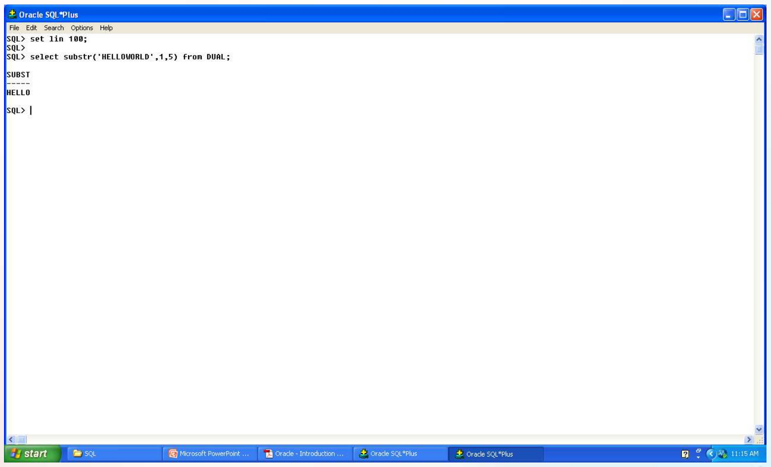
Returns specified characters from character value starting at character position *m*, *n characters long.*

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

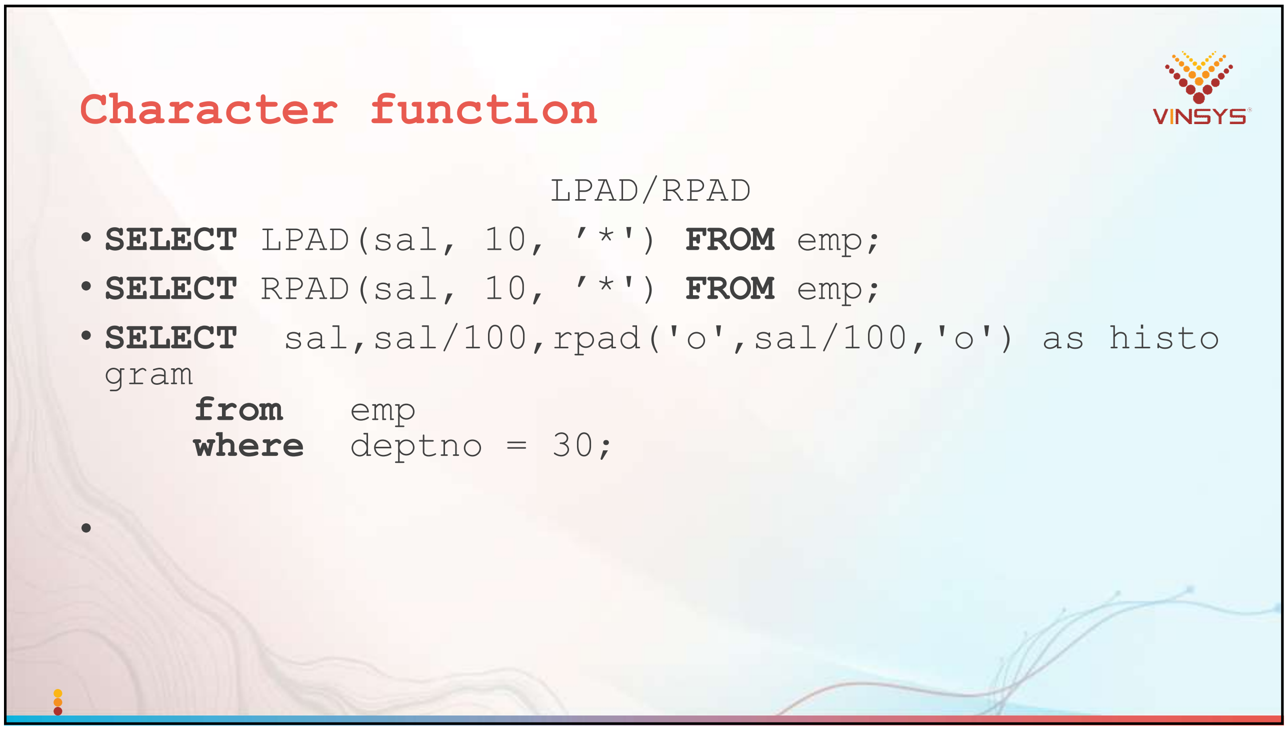
If *n is* omitted, all characters to the end of the string are returned.



Example



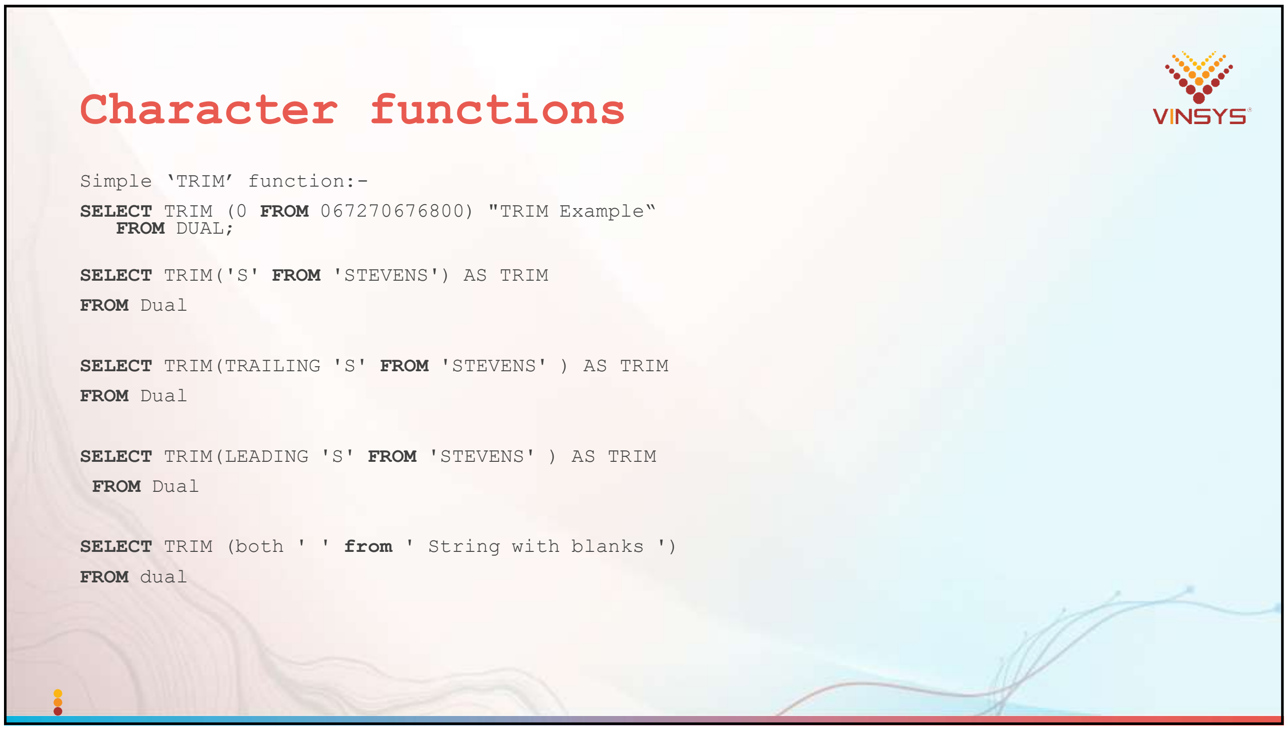
```
SQL> set lin 100;
SQL>
SQL> select substr('HELLOWORLD',1,5) from DUAL;

SUBST
-----
HELLO

SQL> |
```



Character function

LPAD/RPAD

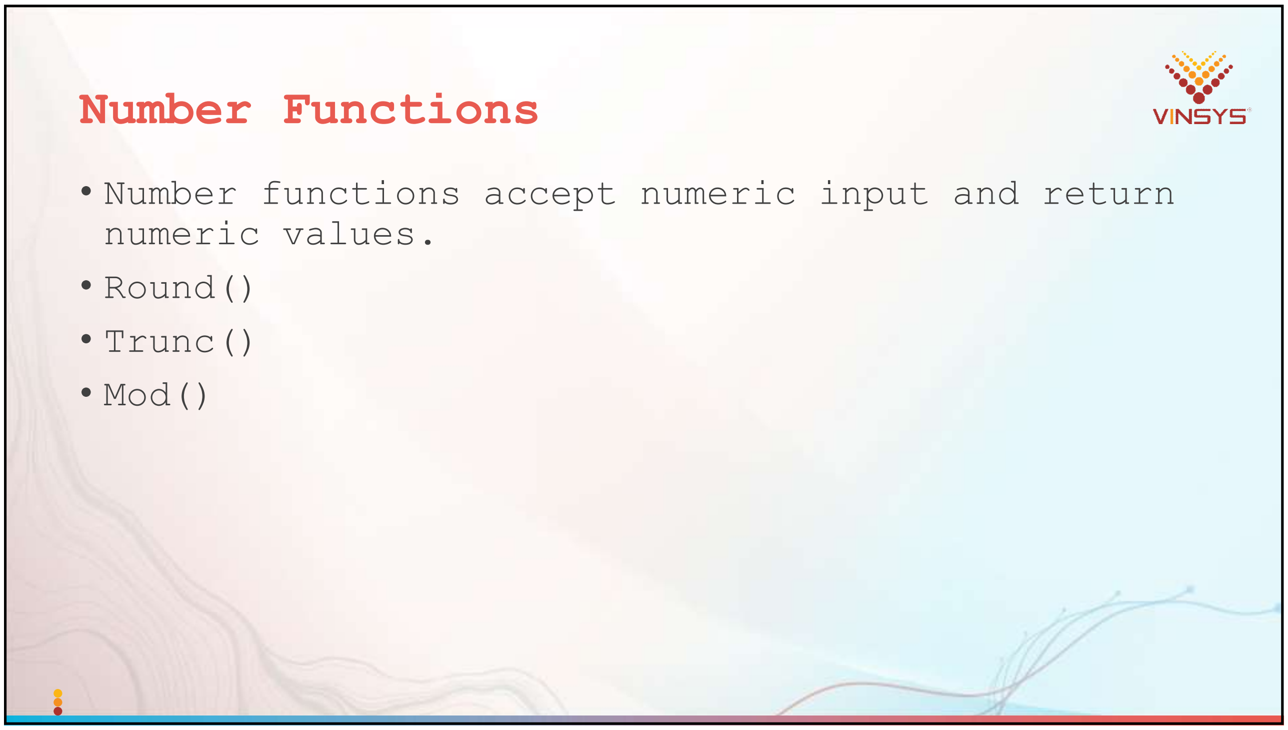
- **SELECT** LPAD(sal, 10, '*') **FROM** emp;
- **SELECT** RPAD(sal, 10, '*') **FROM** emp;
- **SELECT** sal,sal/100,rpad('o',sal/100,'o') as histo gram
  from emp
  where deptno = 30;
-



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()

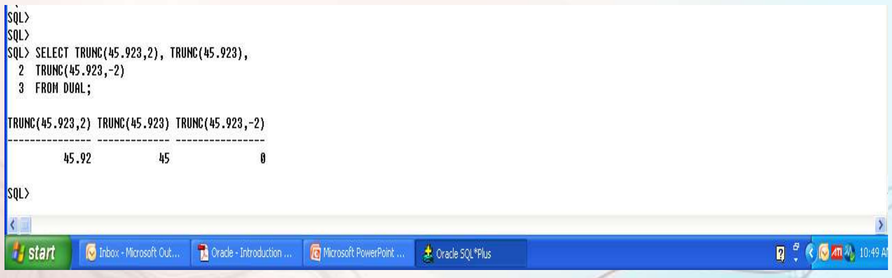
- select round(788.447,2) from dual

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.

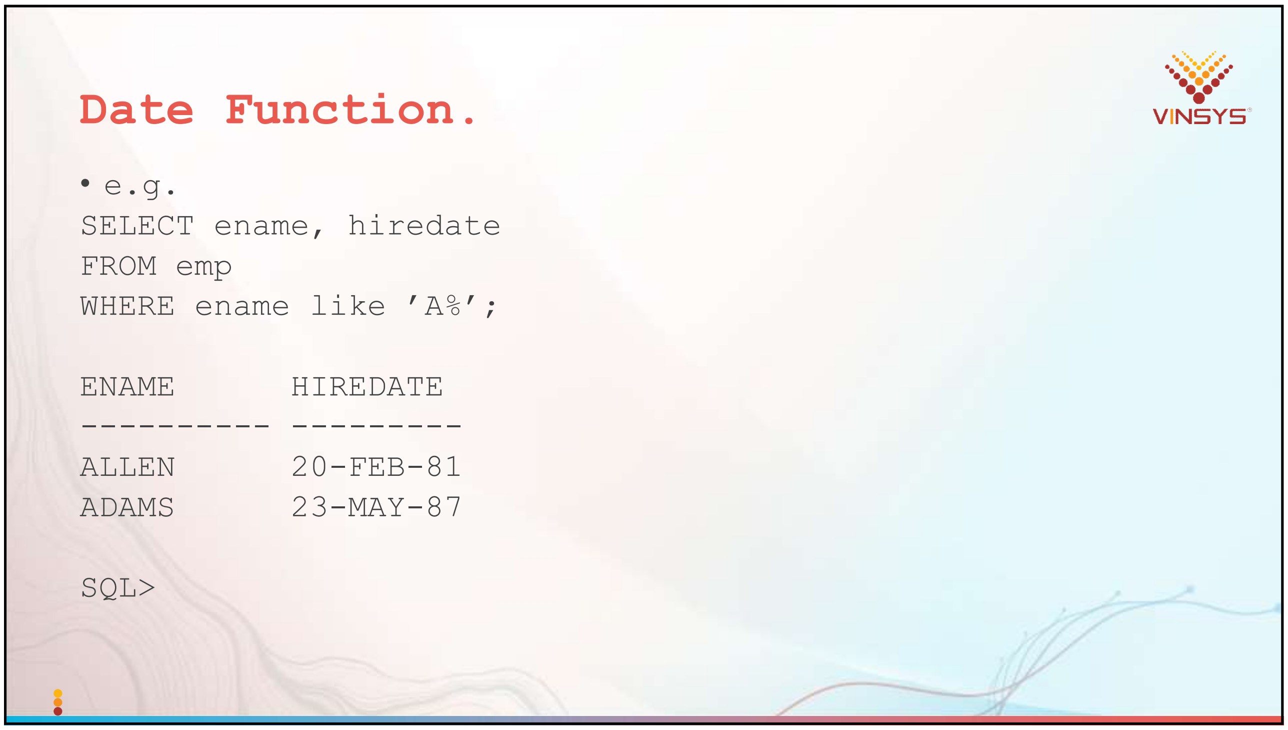
```
SELECT ename, sal, MOD(sal, 5000)
FROM emp
WHERE job= 'SALESMAN';
```

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:

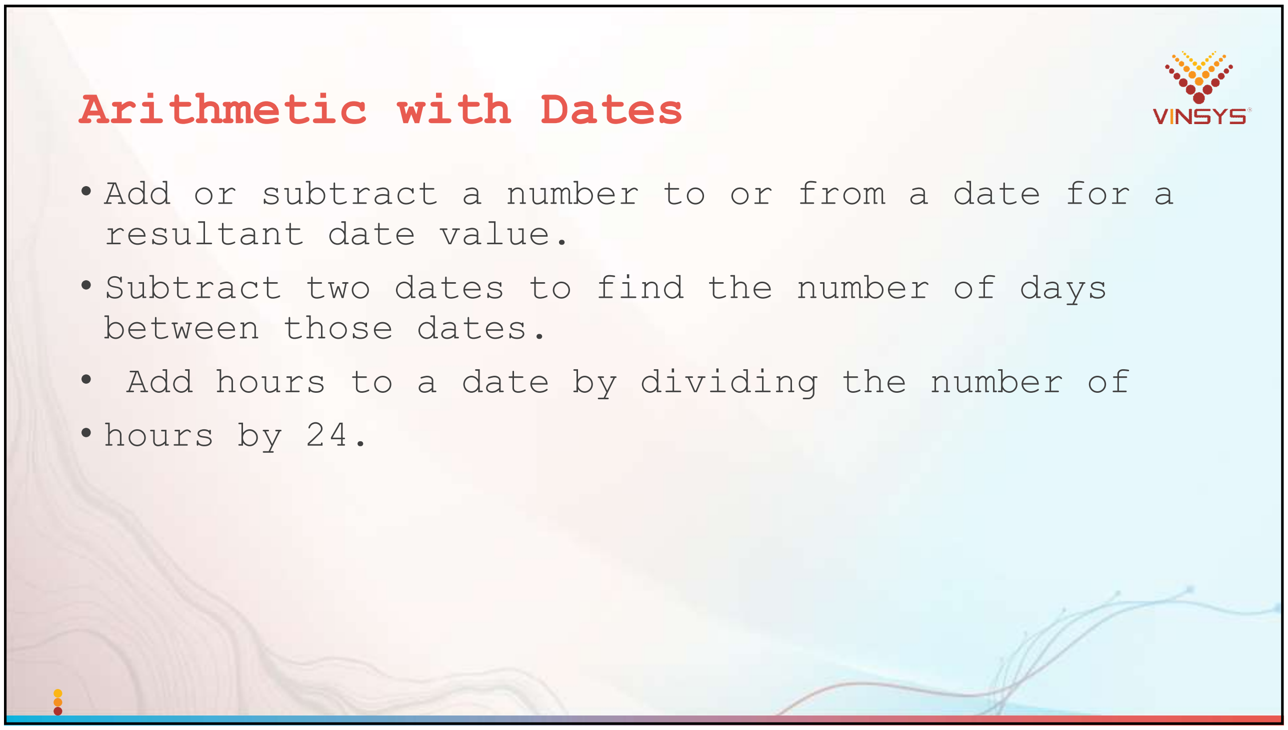
- Date
- Time



Date Function.

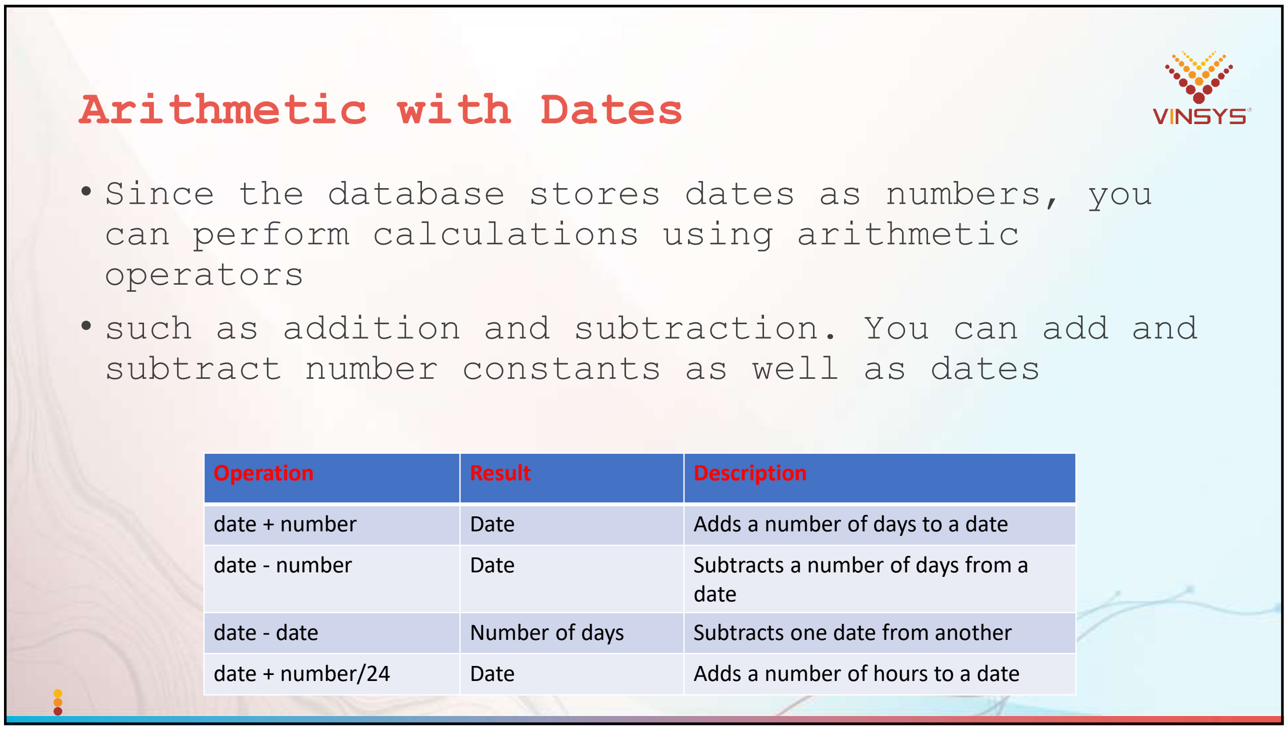
- e.g.

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

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

SQL>
```



Arithmetic with Dates

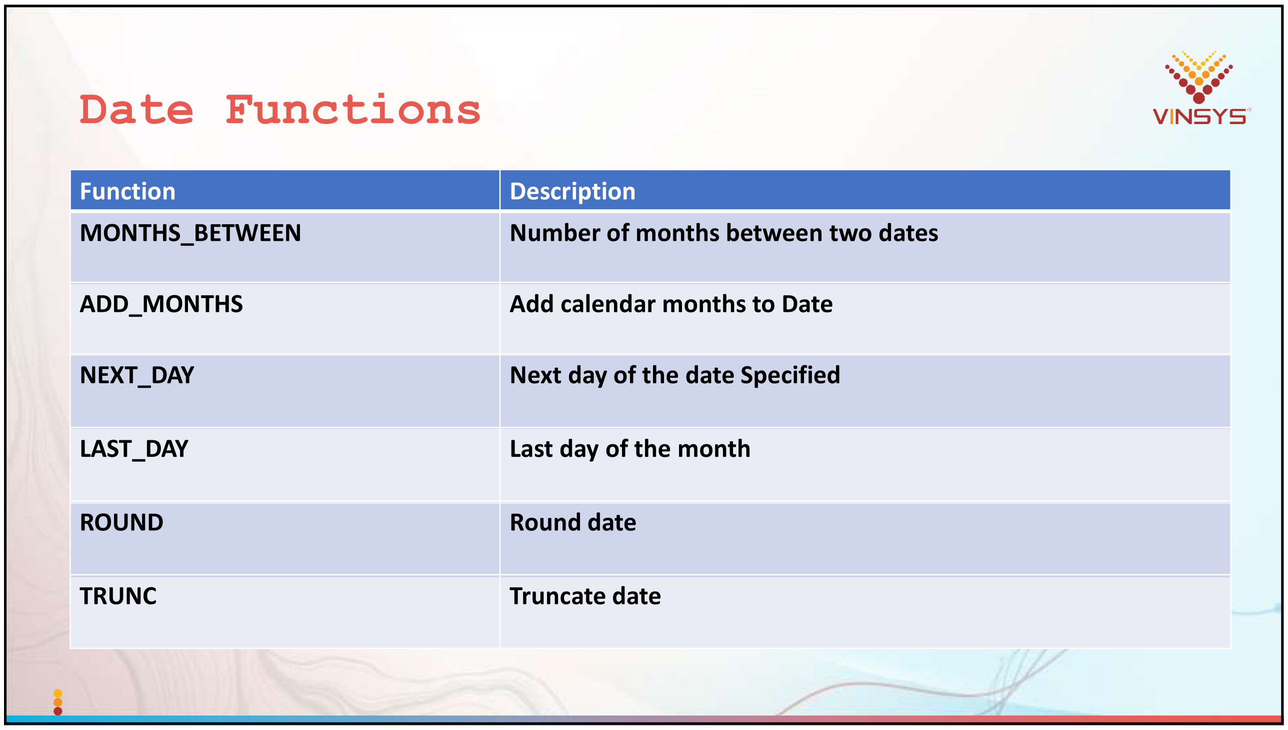
- Add or subtract a number to or from a date for a resultant date value.
- Subtract two dates to find the number of days between those dates.
- Add hours to a date by dividing the number of
- hours by 24.



Arithmetic with Dates

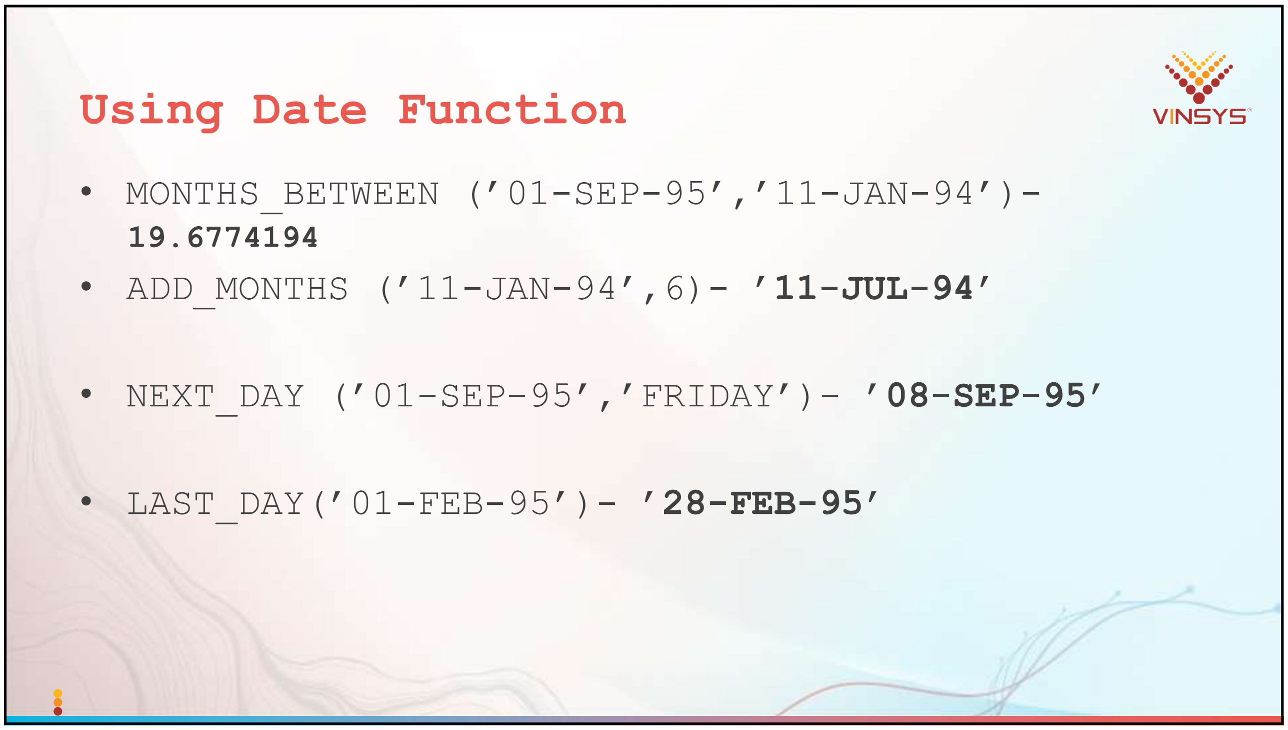
- Since the database stores dates as numbers, you can perform calculations using arithmetic operators
- such as addition and subtraction. You can add and subtract number constants as well as dates

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



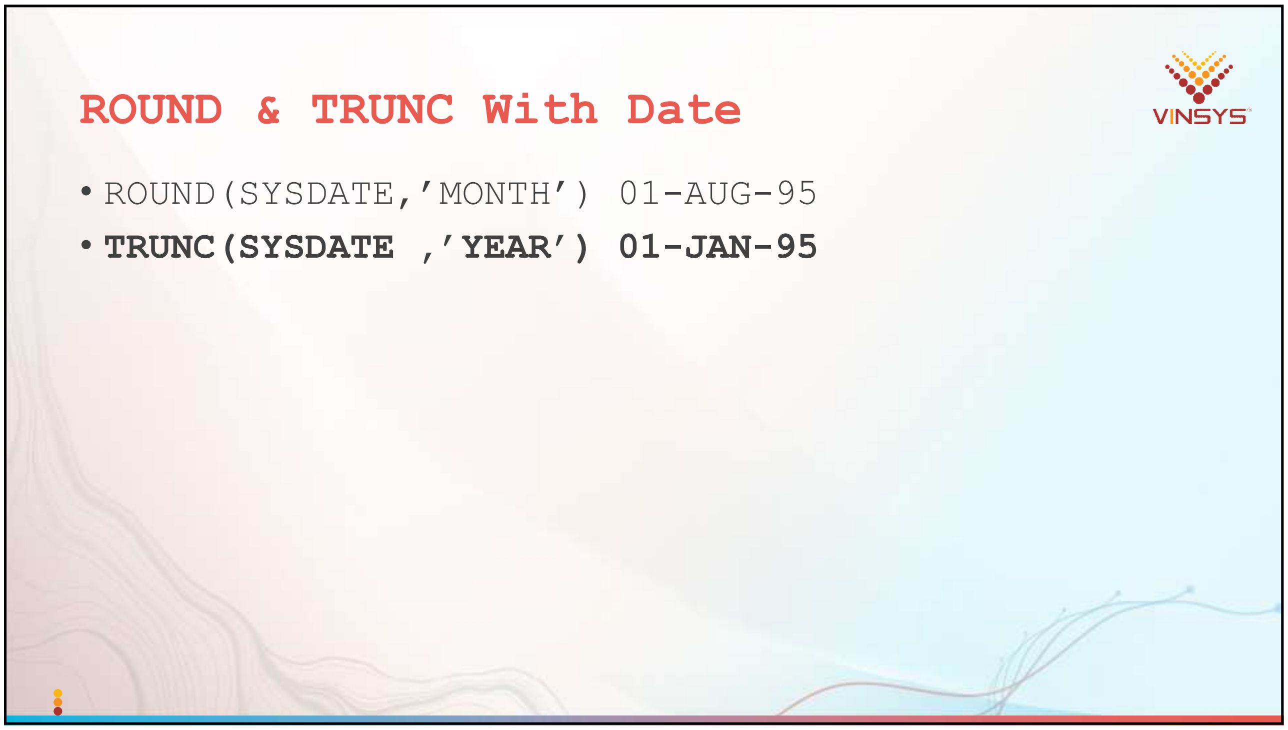
Date Functions

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

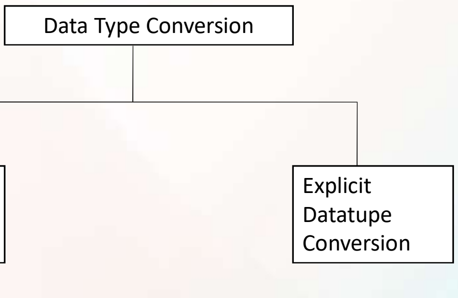
- MONTHS_BETWEEN ('01-SEP-95','11-JAN-94')- **19.6774194**
- ADD_MONTHS ('11-JAN-94',6)- **'11-JUL-94'**
- NEXT_DAY ('01-SEP-95','FRIDAY')- **'08-SEP-95'**
- LAST_DAY('01-FEB-95')- **'28-FEB-95'**



ROUND & TRUNC With Date

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

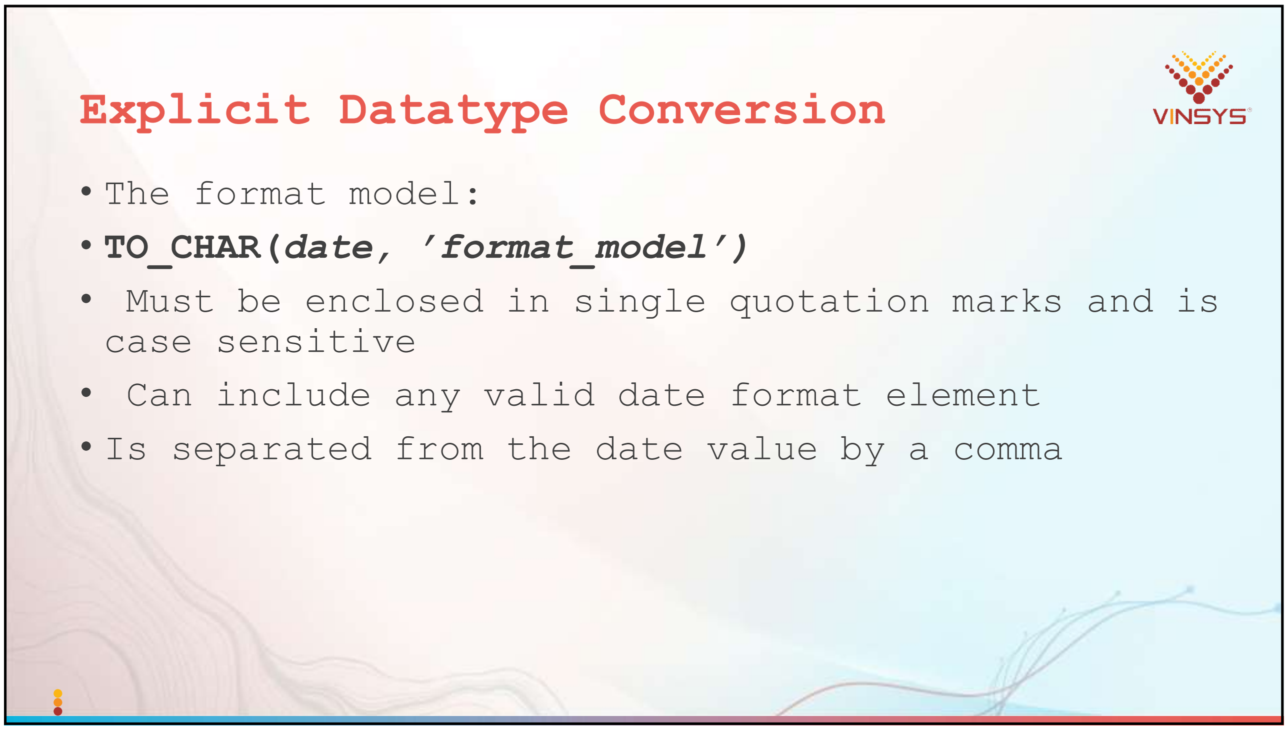
Conversion function



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

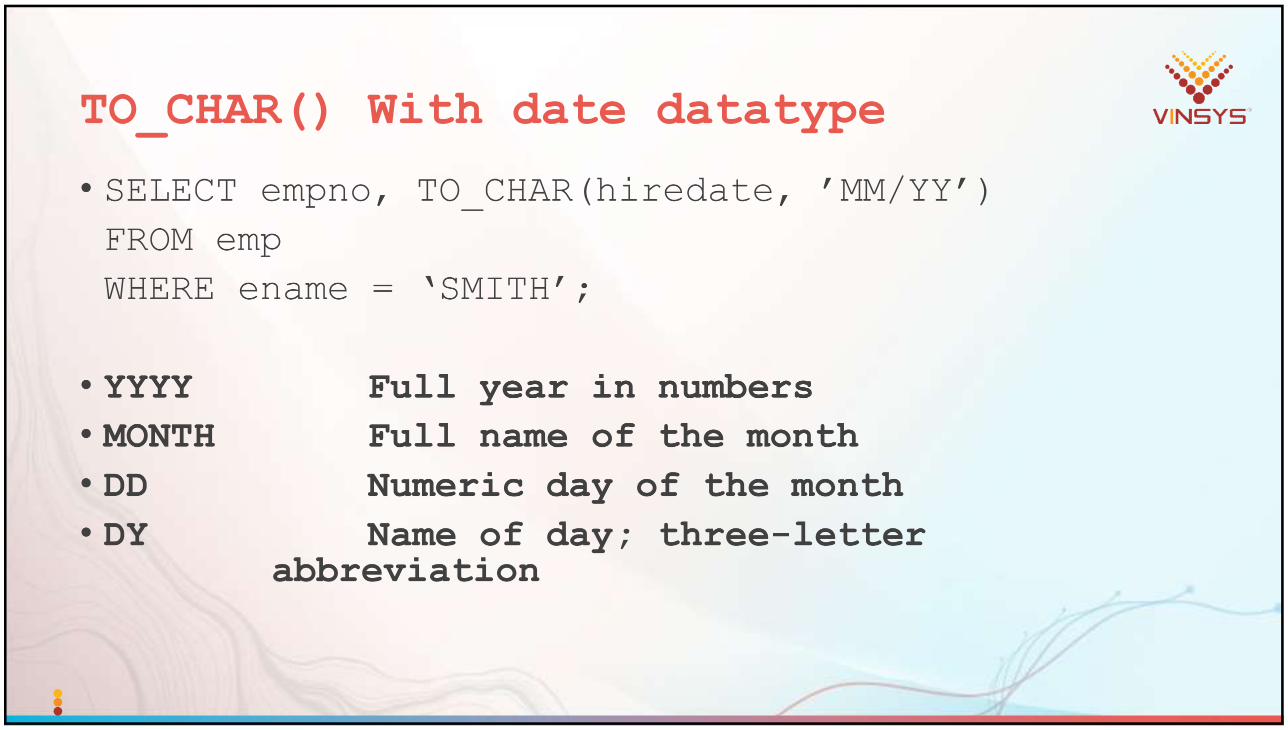
Explicit Data Type Conversions

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



Explicit Datatype Conversion

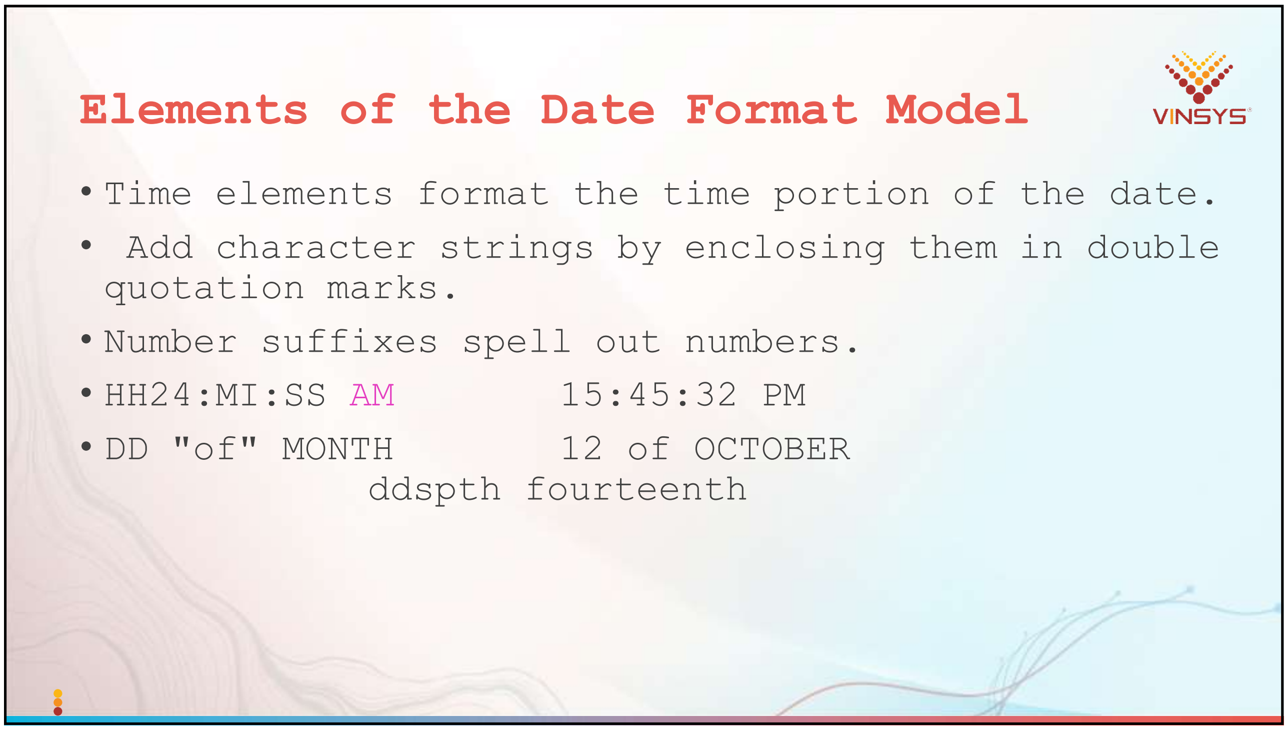
- The format model:
- **TO_CHAR(*date, 'format_model'*)**
- Must be enclosed in single quotation marks and is case sensitive
- Can include any valid date format element
- Is separated from the date value by a comma



TO_CHAR() With date datatype

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

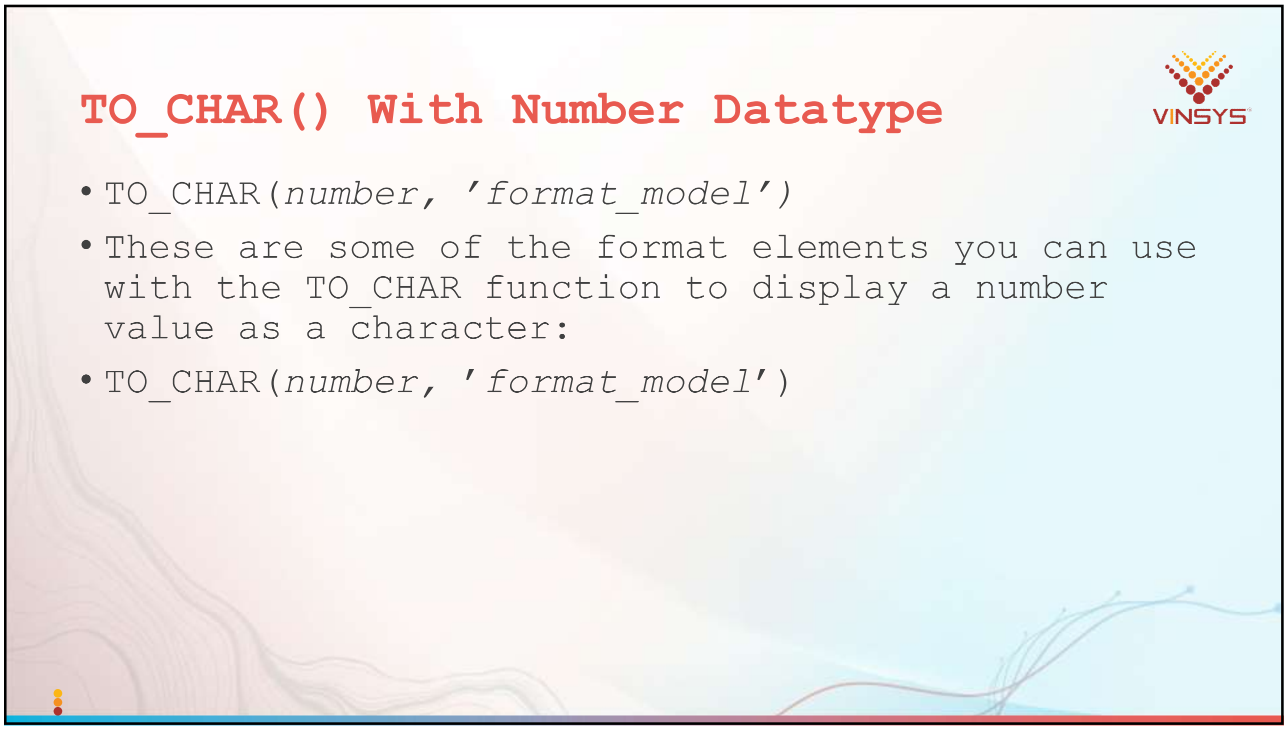
• **YYYY**	**Full year in numbers**
• **MONTH**	**Full name of the month**
• **DD**	**Numeric day of the month**
• **DY**	**Name of day; three-letter abbreviation**



Elements of the Date Format Model

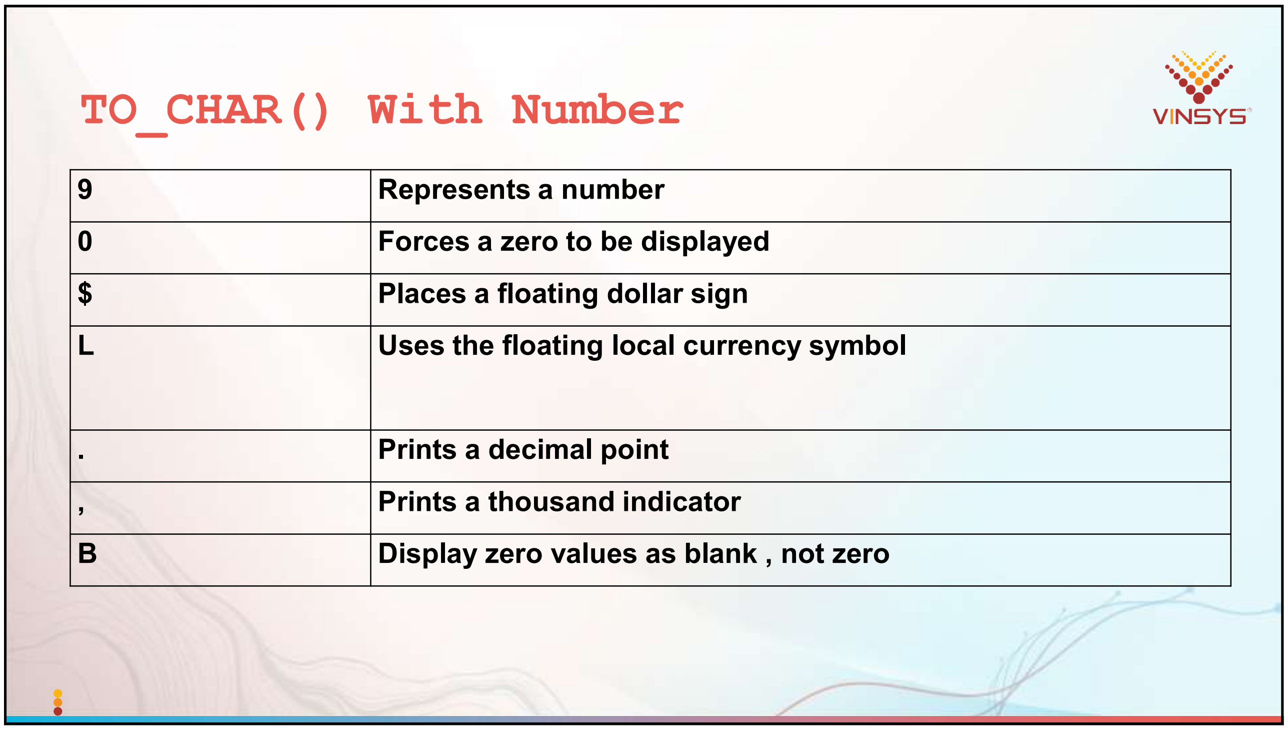
- Time elements format the time portion of the date.
- Add character strings by enclosing them in double quotation marks.
- Number suffixes spell out numbers.

HH24:MI:SS AM	15:45:32 PM
DD "of" MONTH	12 of OCTOBER
ddspth	fourteenth



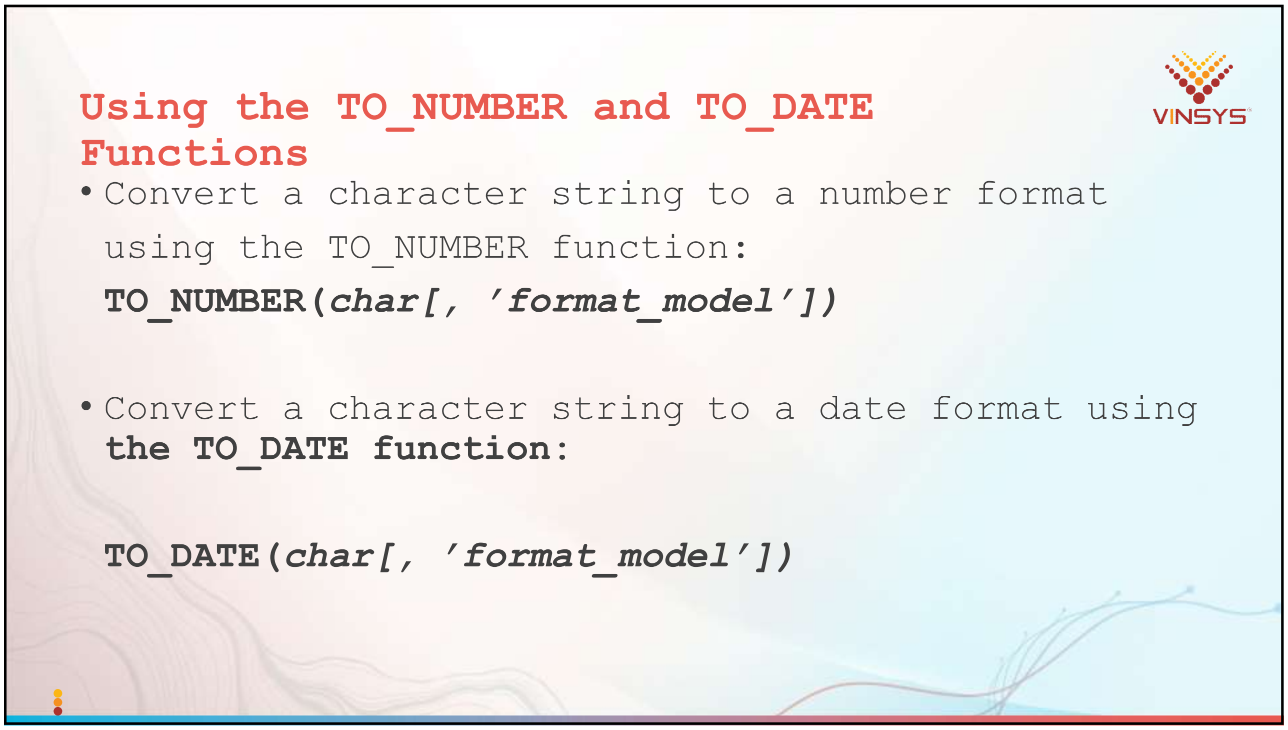
TO_CHAR() With Number Datatype

- 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

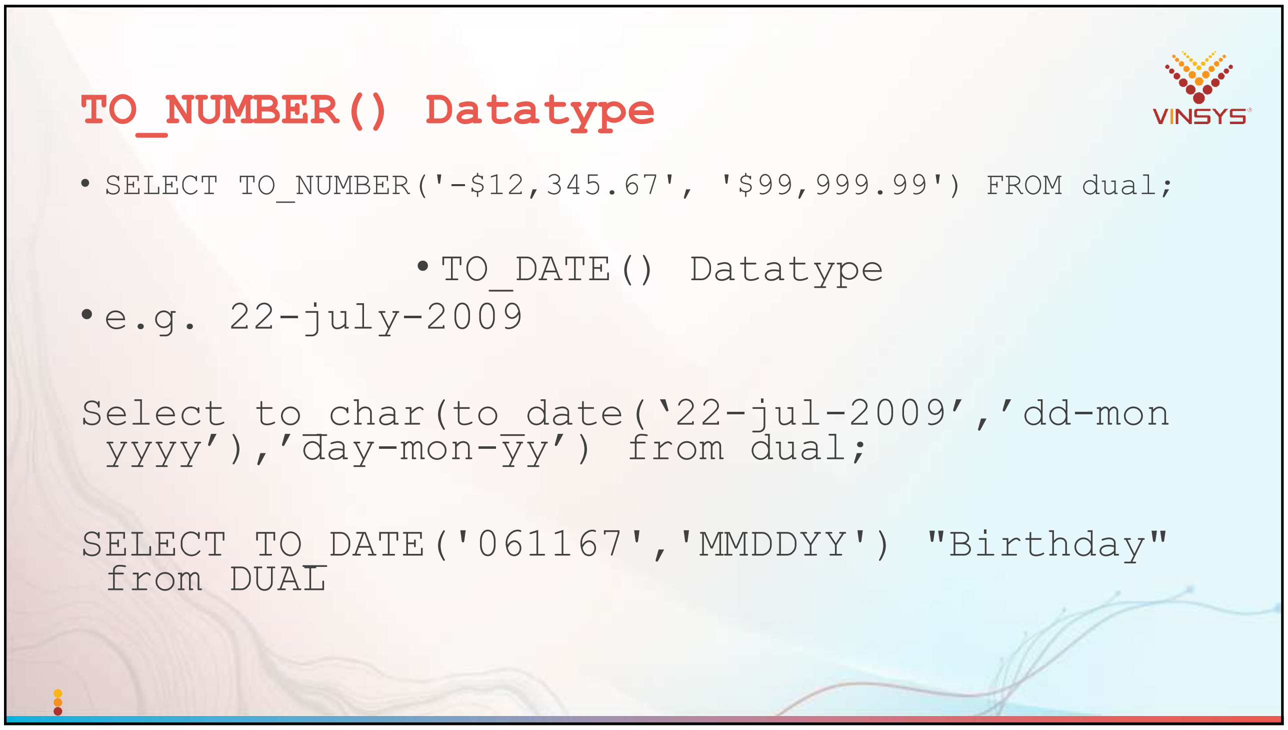


Using the TO_NUMBER and TO_DATE Functions

- 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']*)



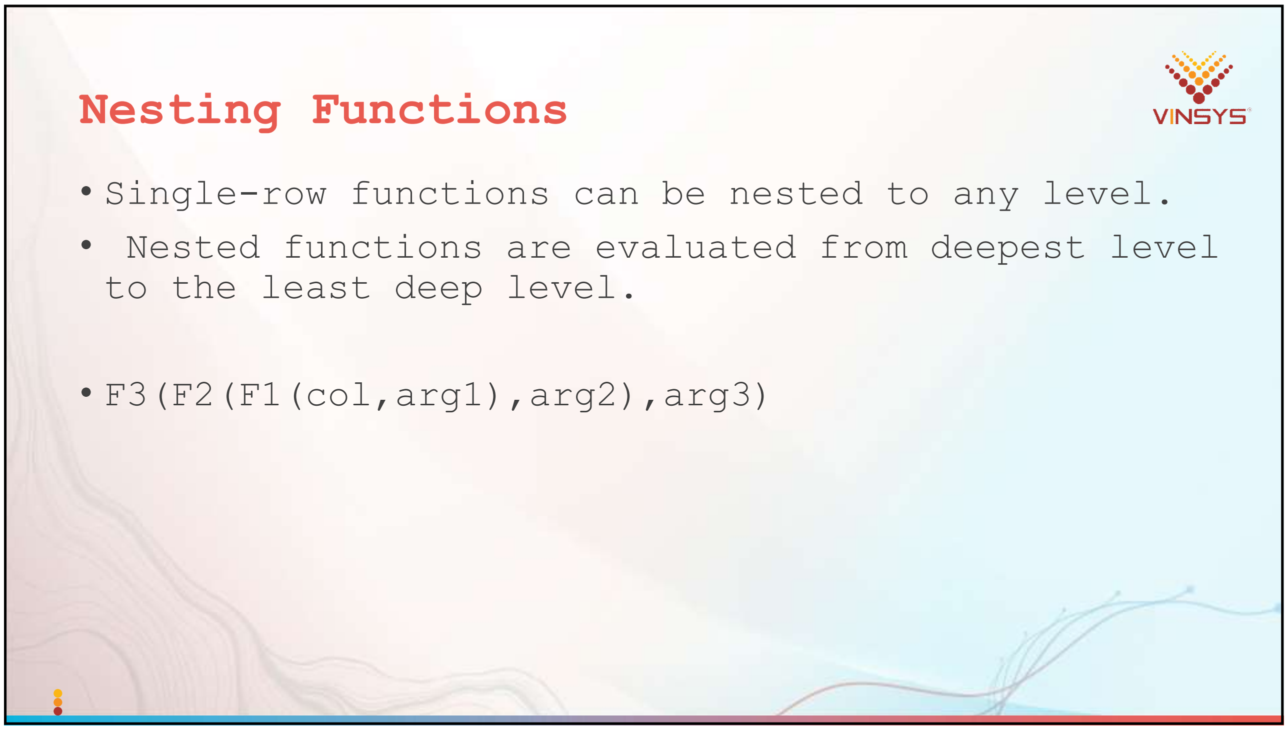
TO_NUMBER() Datatype

- 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, ..., expr*n*)

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 expr1can 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

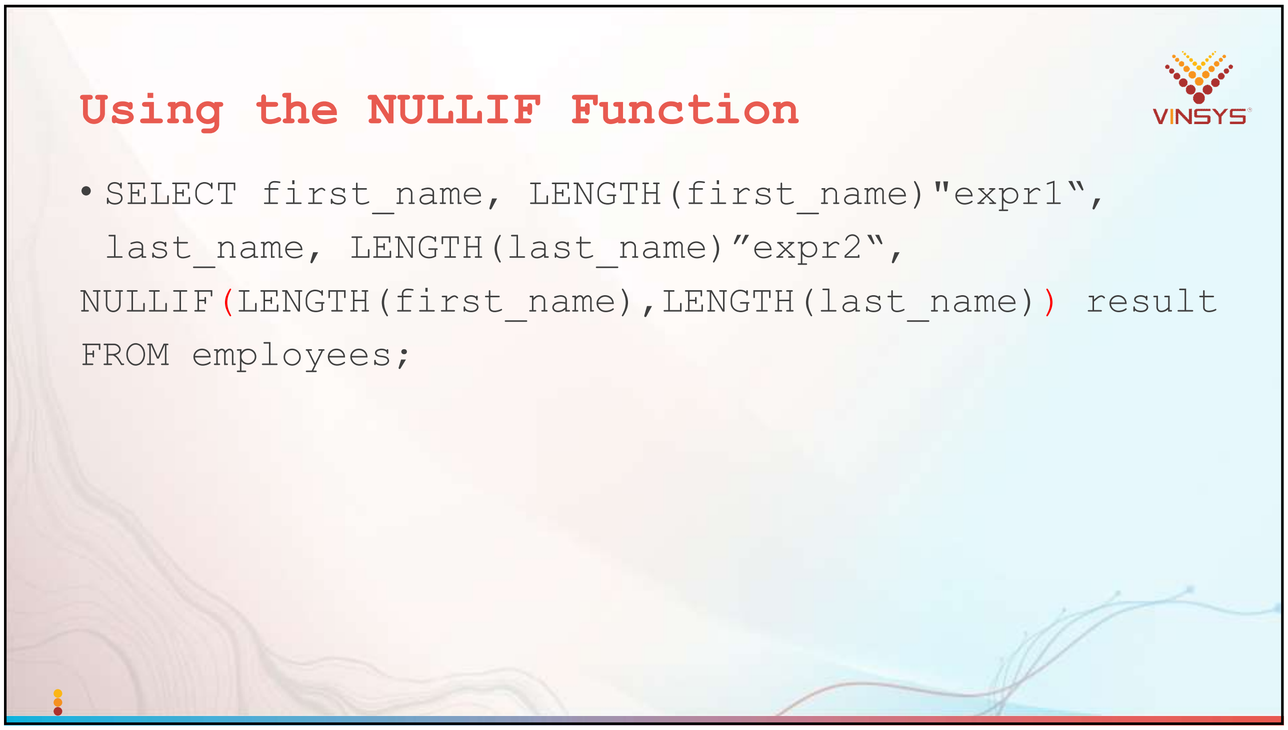
- Converts a null to an actual value.
- Data types that can be used are date, character,and number.
- e.g.

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

NVL2 function

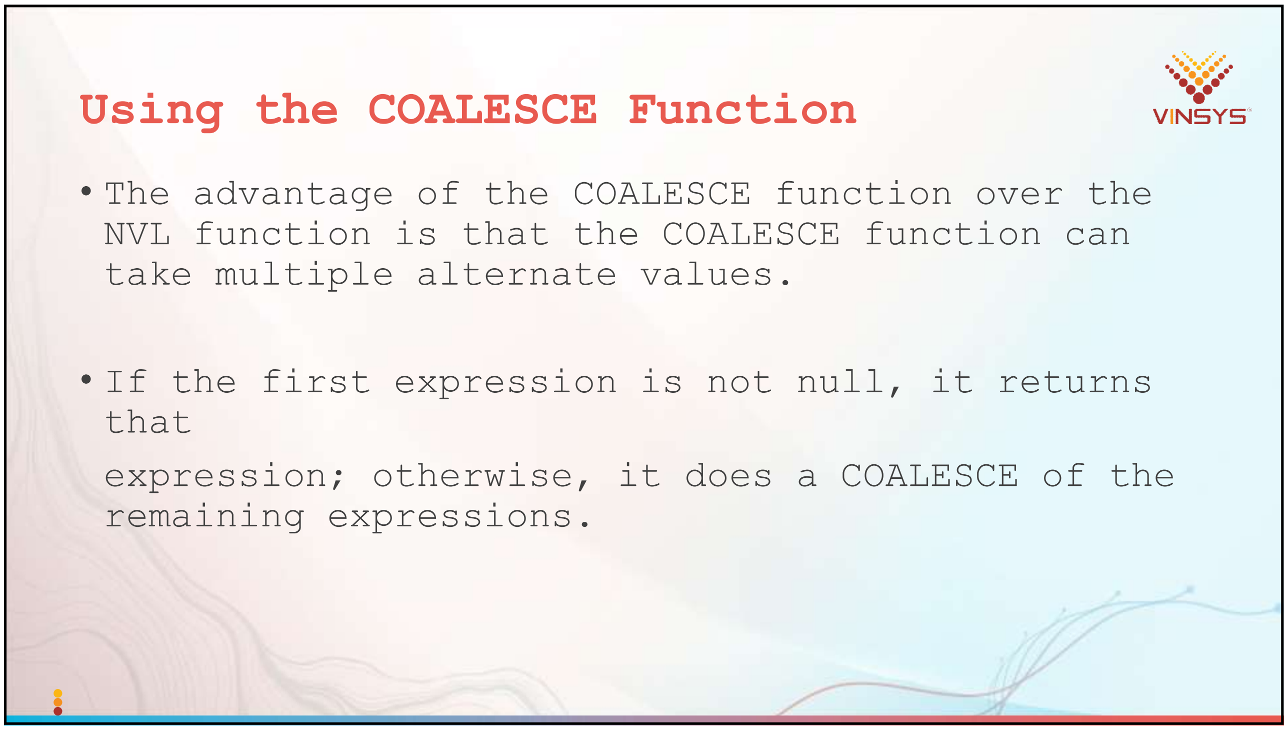
-
  ```
  SELECT ename, sal, comm,
     NVL2(comm,'SAL+COMM', 'SAL') income
     FROM emp WHERE deptno IN (20, 30)
  ```

-
  ```
  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.

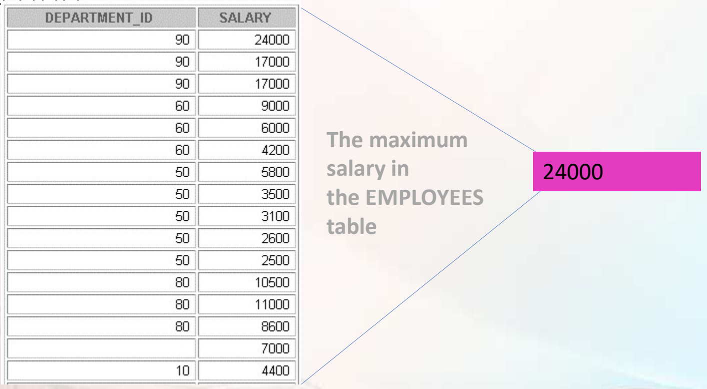
Decode Function

```
• 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

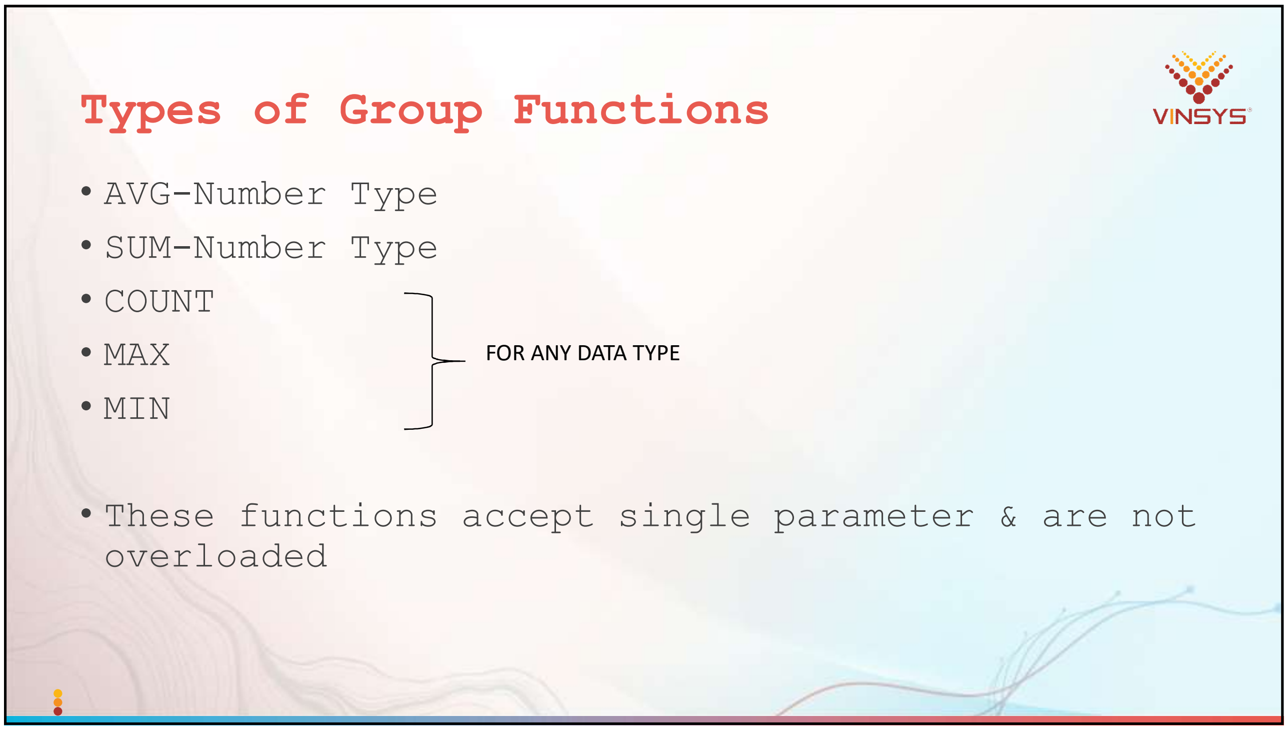
Group Function

- 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

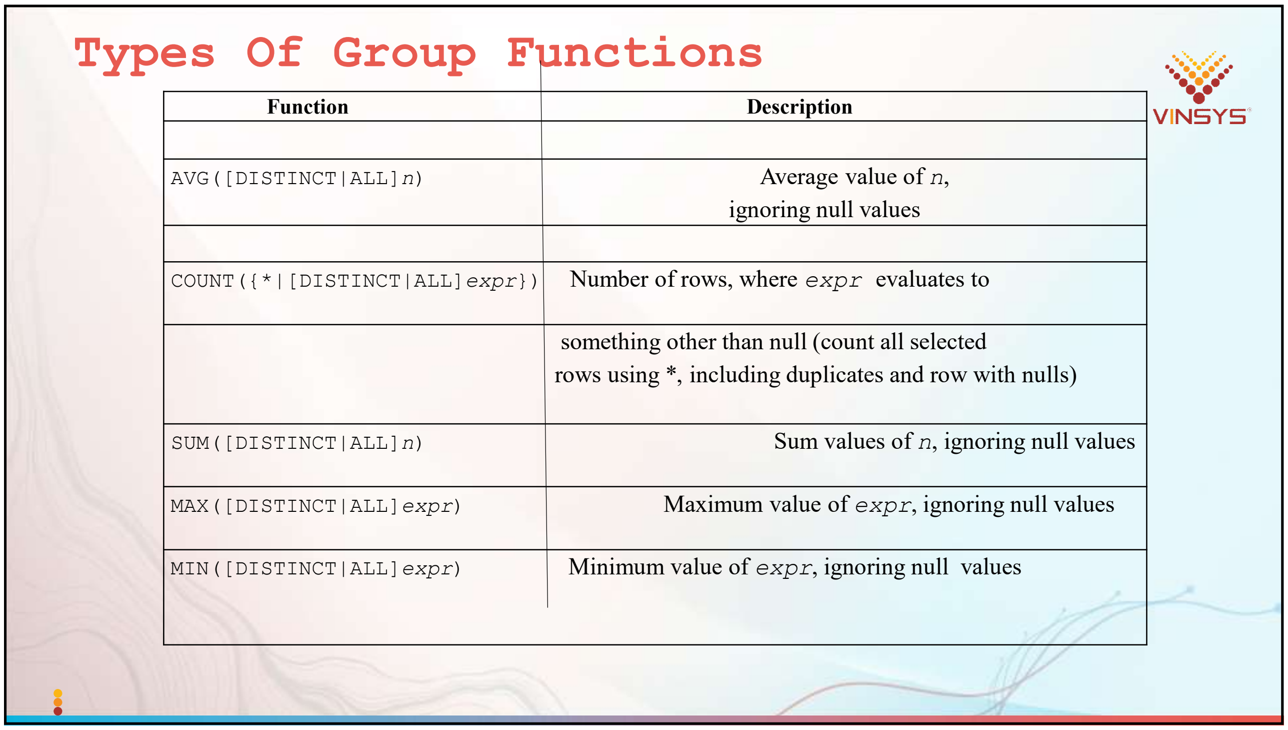


Types of Group Functions

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

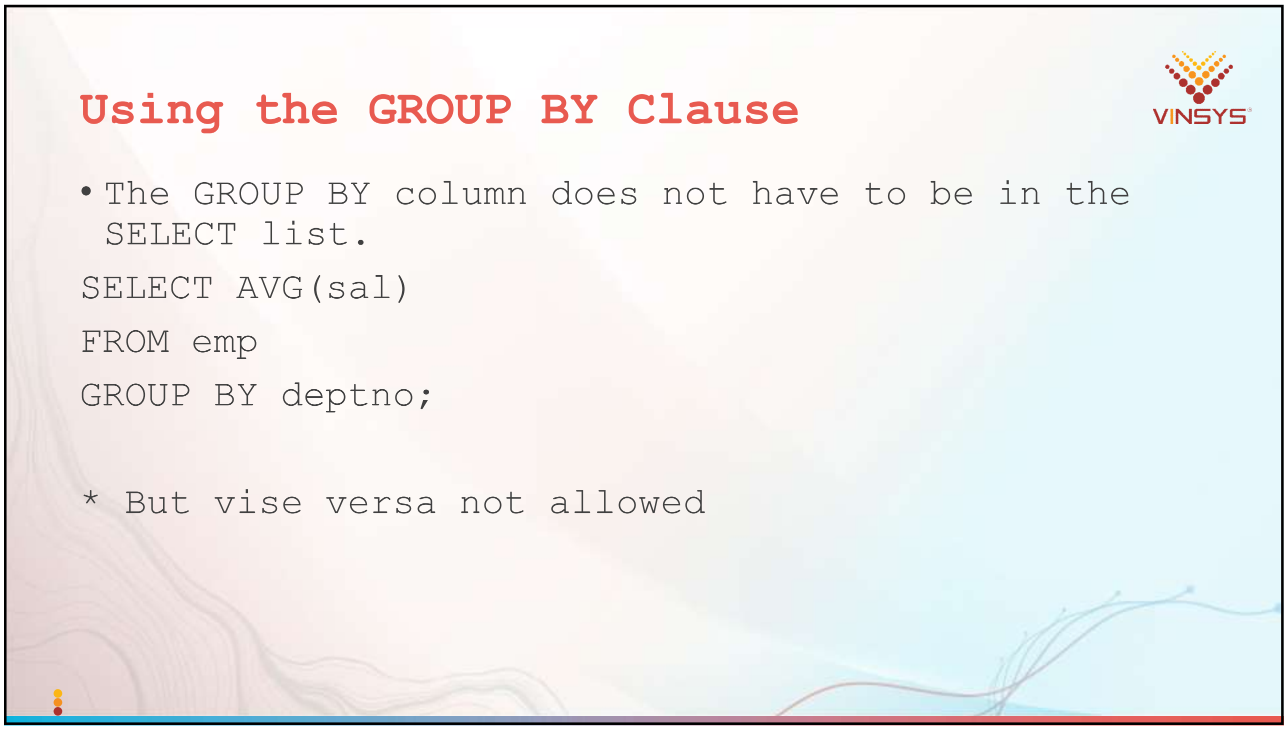
```
     FROM employees;
```

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]
```

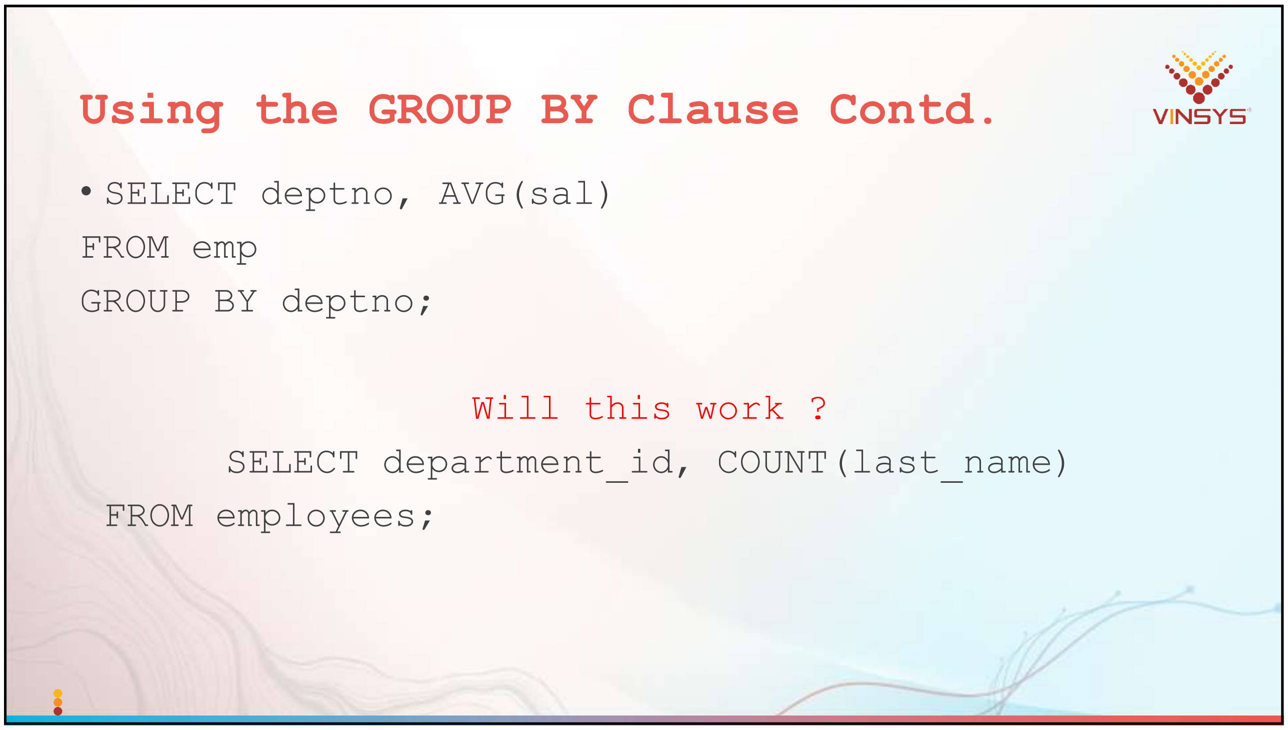


Using the GROUP BY Clause

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

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

* But vise 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

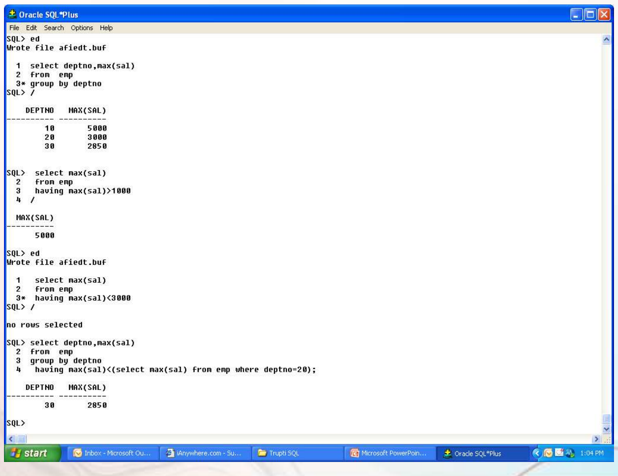
- The Oracle server performs the following steps when you use the HAVING clause:

- 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



```
SQL> ed
Wrote 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
Wrote 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>
```

VINSYS

• Manipulating Data

VINSYS

Data Manipulation Language

- A DML statement is executed when you:

o Add new rows to a table

o Modify existing rows in a table

o Remove existing rows from a table

Logical unit of DML is a ‘Transaction’

Insert

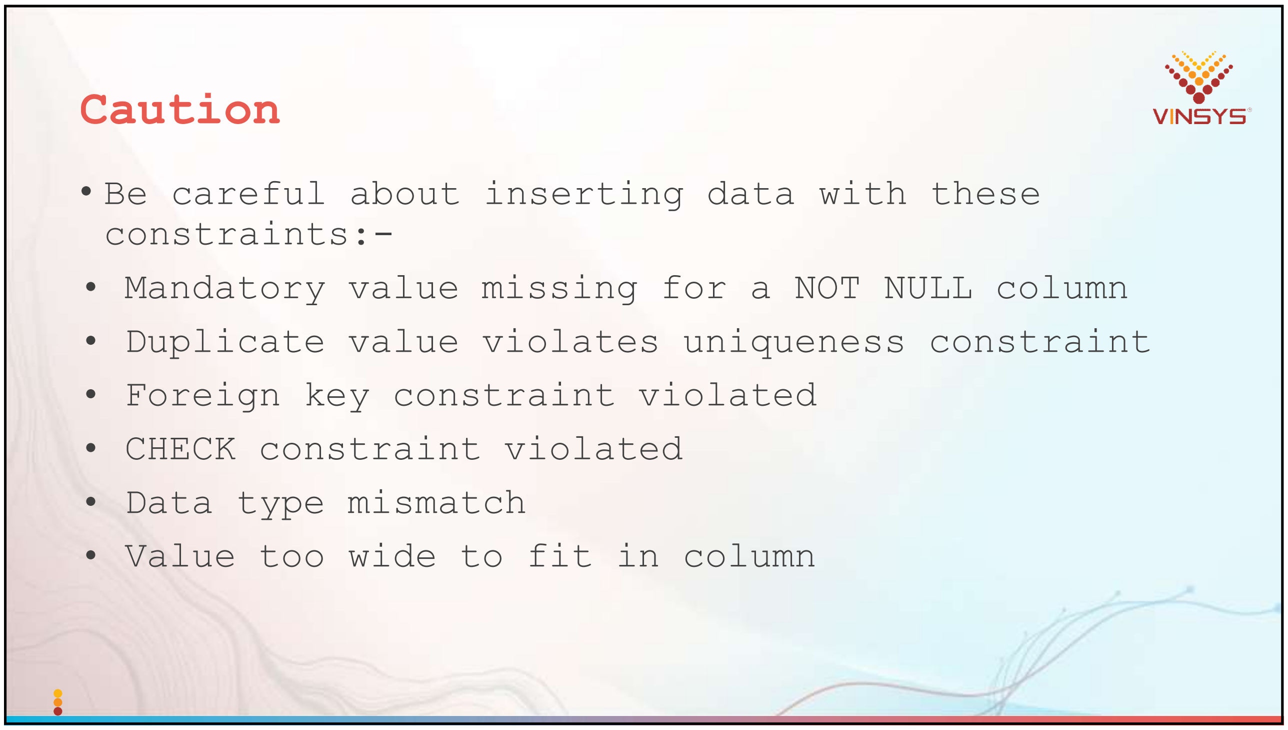
- To add a new row to the table ‘Insert’ Statement is issued.
- The INSERT statement syntax –

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

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

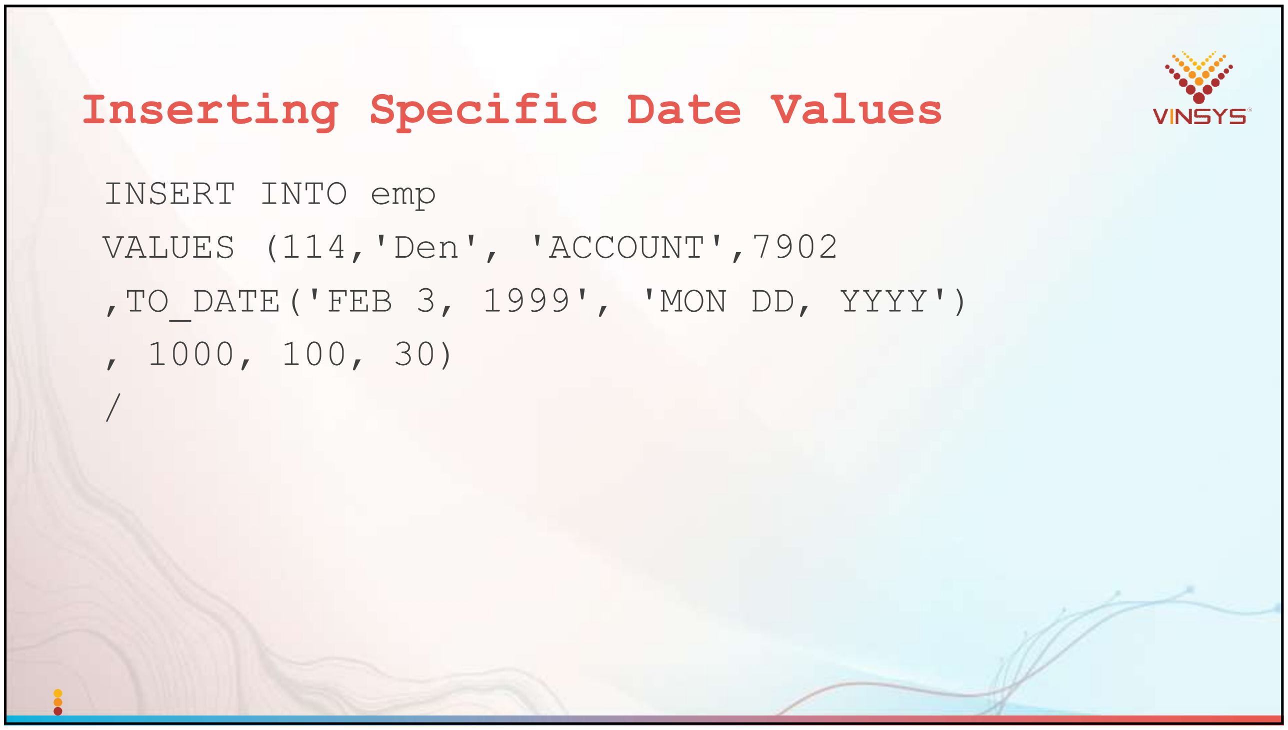
Guidelines

- 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

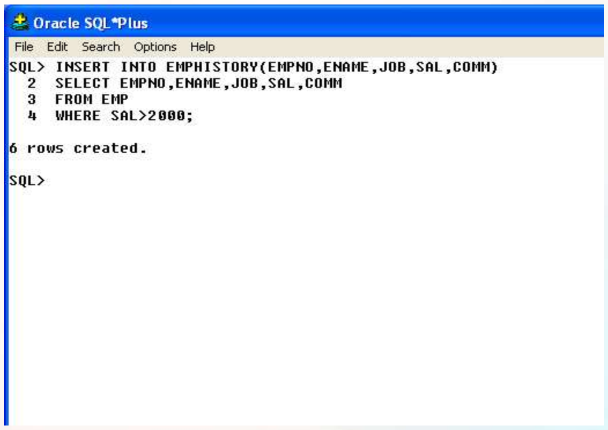
- Use & substitution in a SQL statement to prompt for values.
- & is a placeholder for the variable value

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

Copying Rows from Another Table

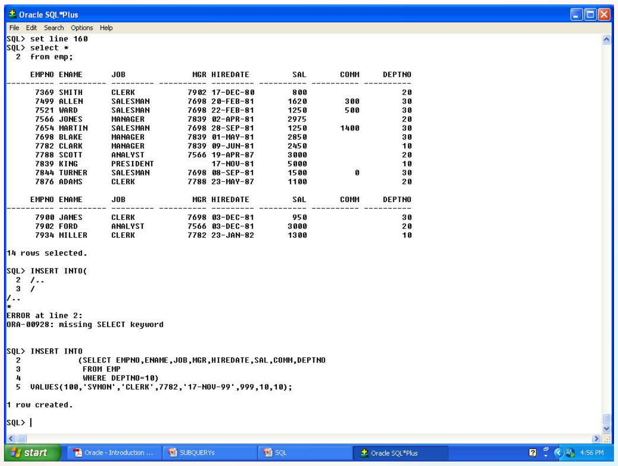
- Write your INSERT statement with a subquery.
- Do not use the VALUES clause.
- Match the number of columns in the INSERT clause to those in the subquery.

Inserting Based On Another Table



```
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



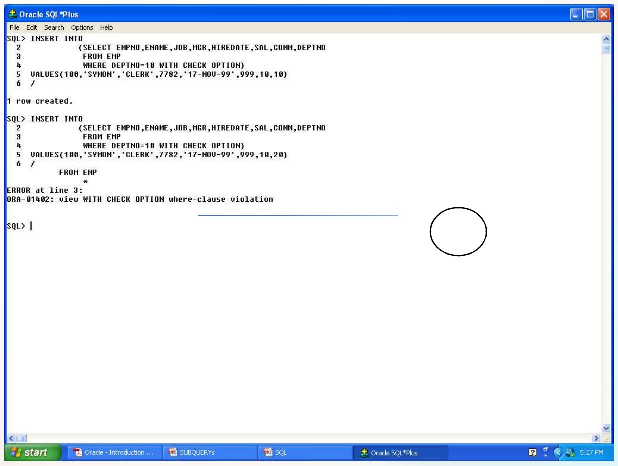
```
SQL> set line 160
SQL> select *
  2  from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1620        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.

SQL> INSERT INTO(
  2  /..
  3  /
/..
*
ERROR at line 2:
ORA-00928: missing SELECT keyword


SQL> INSERT INTO
  2            (SELECT EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO
  3             FROM EMP
  4             WHERE DEPTNO=10)
  5  VALUES(100,'SYMON','CLERK',7782,'17-NOV-99',999,10,10);

1 row created.

SQL> |
```

YOU HAVE TO INCLUDE ALL THE COL. FROM THE TABLE USING SUCH AN INSERT STATEMENT

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



```
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,'SYMON','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,'SYMON','CLERK',7782,'17-NOV-99',999,10,20)
  6  /
           FROM EMP
                *
ERROR at line 3:
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 Stmnt

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.

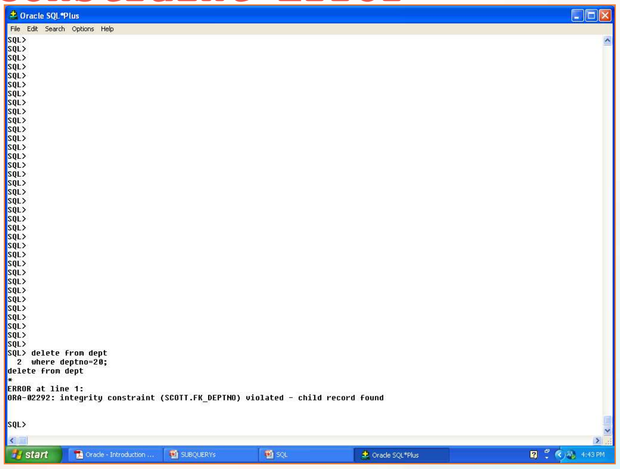
- e.g- DELETE FROM emp;

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%');
```

Deleting Rows: Integrity Constraint Error



```
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL> delete from dept
  2  where deptno=20;
delete from dept
*
ERROR at line 1:
ORA-02292: integrity constraint (SCOTT.FK_DEPTNO) violated - child record found


SQL>
```

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,Permenent.
- 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` *name*	Marks a savepoint within the current transaction
`ROLLBACK`	`ROLLBACK` ends the current transaction by discarding all pending data changes
`ROLLBACK TO` *SAVEPOINT name*	`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

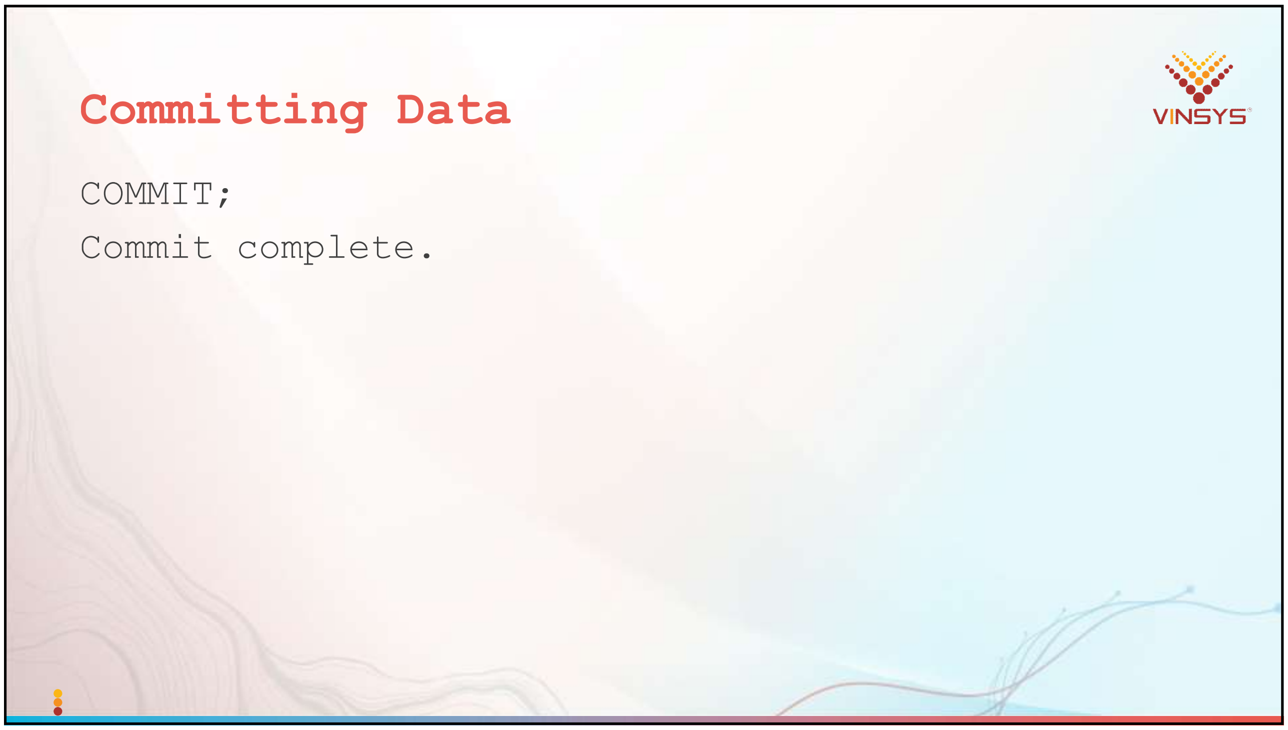
- An automatic commit occurs under the following
- circumstances:
  - DDL statement is issued
  - DCL statement is issued
  - Normal exit from *iSQL*Plus, without explicitly* issuing COMMIT or ROLLBACK statements
- An automatic rollback occurs under an abnormal termination of *iSQL*Plus or a system failure.*

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.

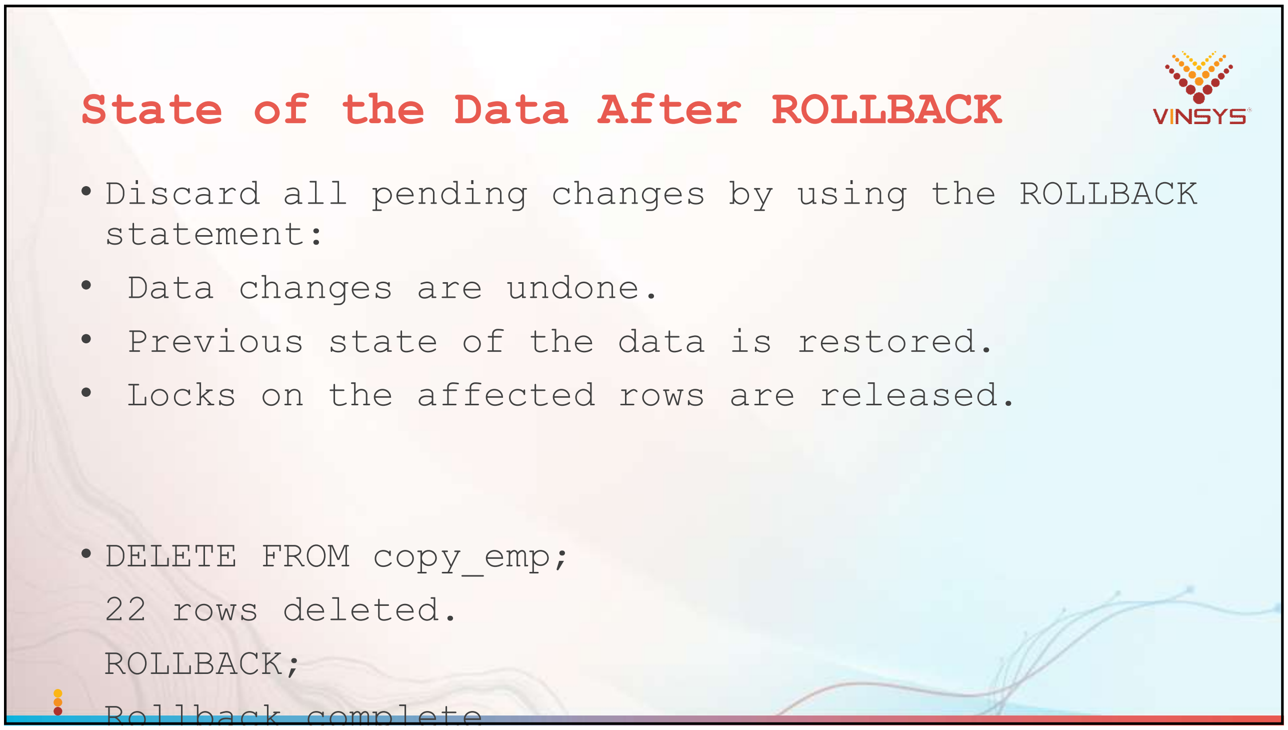
State of the Data after COMMIT

- 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

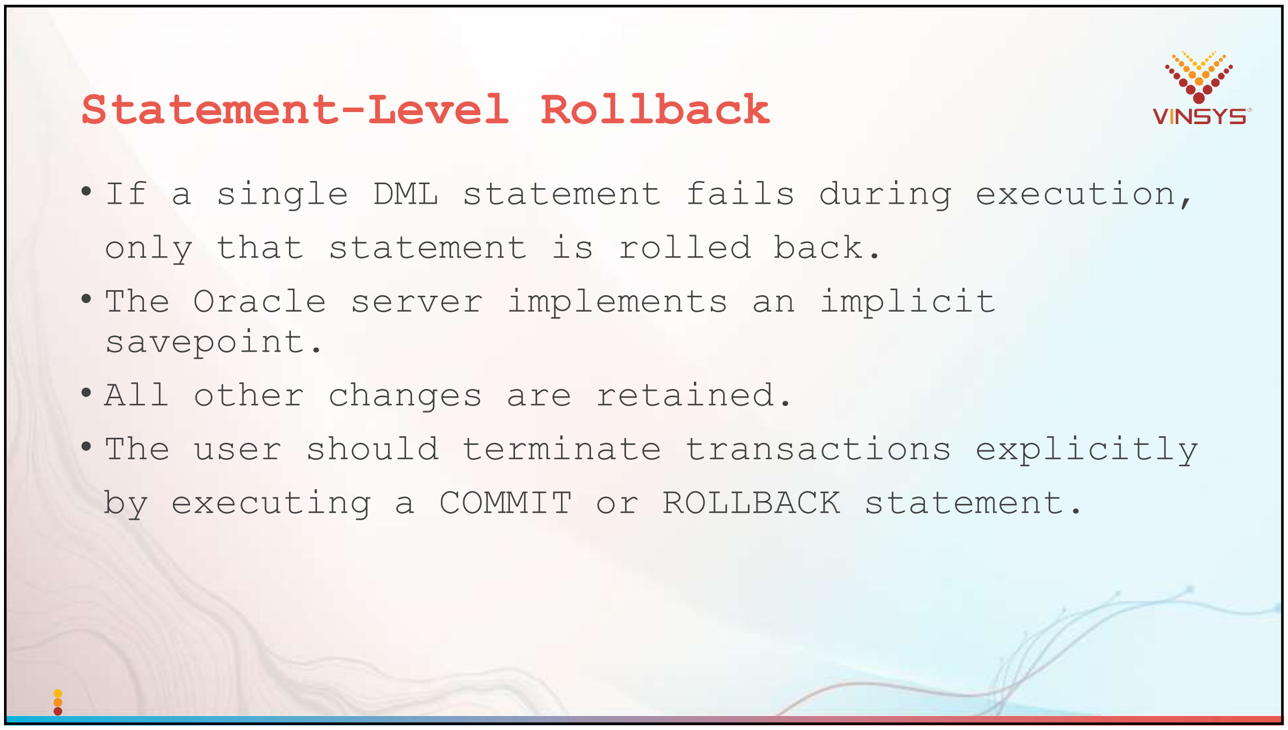
```
COMMIT;
Commit complete.
```



State of the Data After ROLLBACK

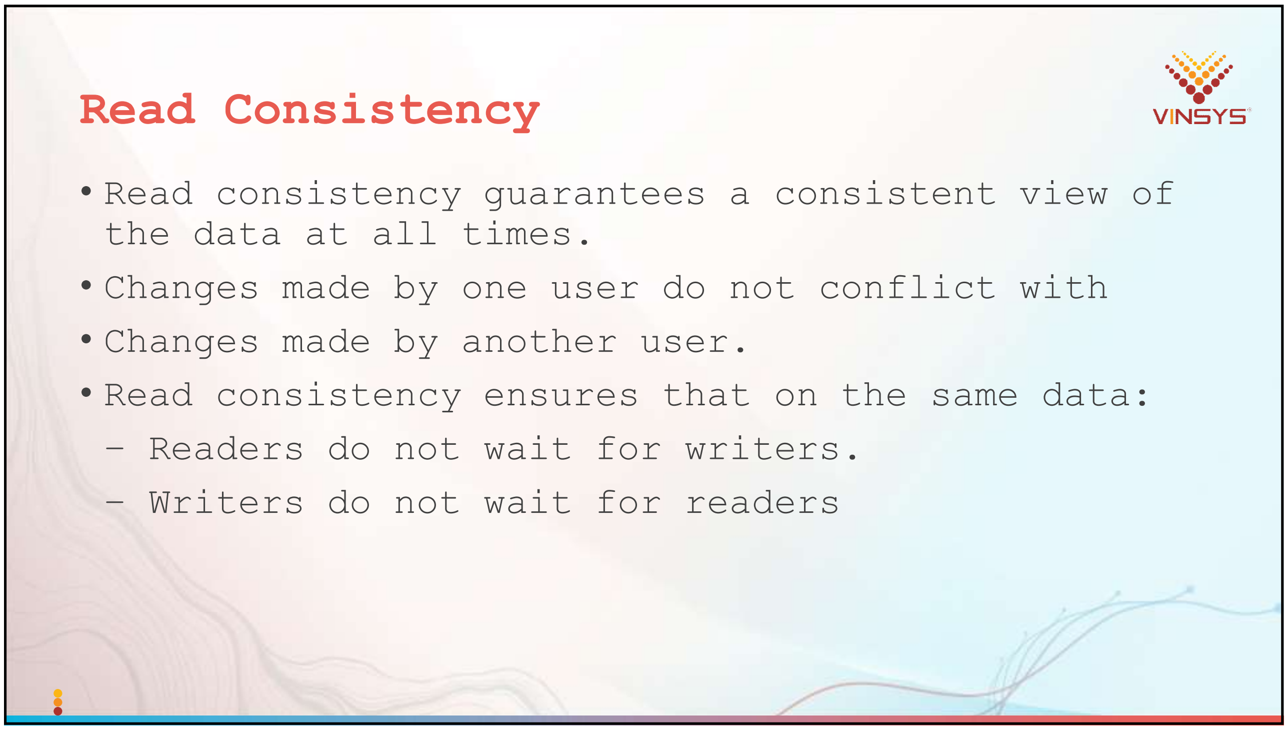
- Discard all pending changes by using the ROLLBACK statement:
- Data changes are undone.
- Previous state of the data is restored.
- Locks on the affected rows are released.

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



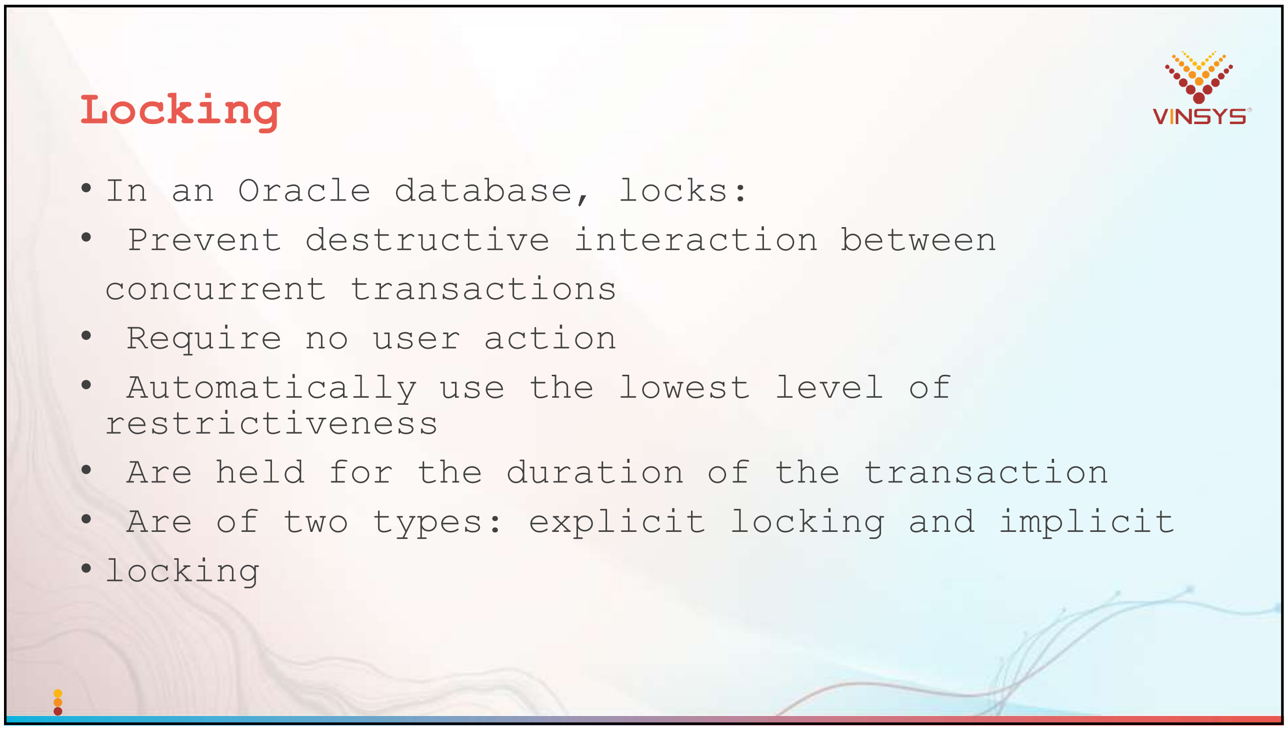
Statement-Level Rollback

- 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.



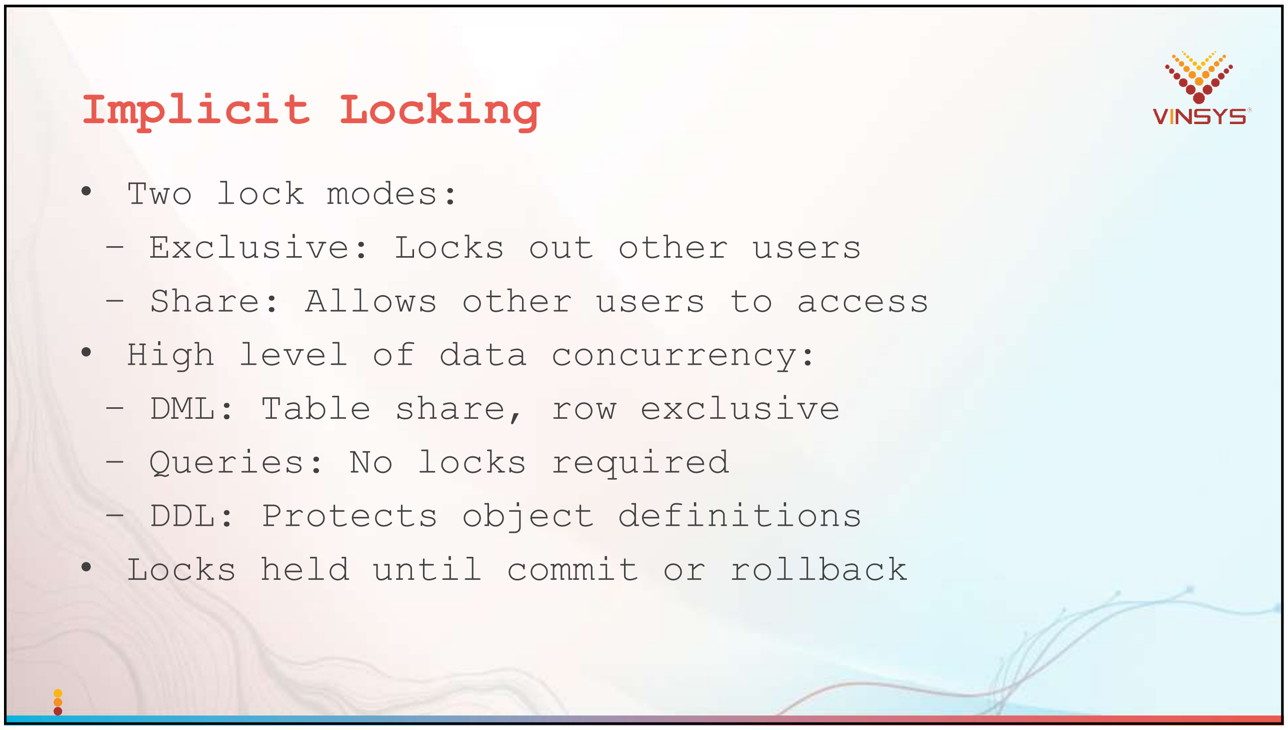
Read Consistency

- 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

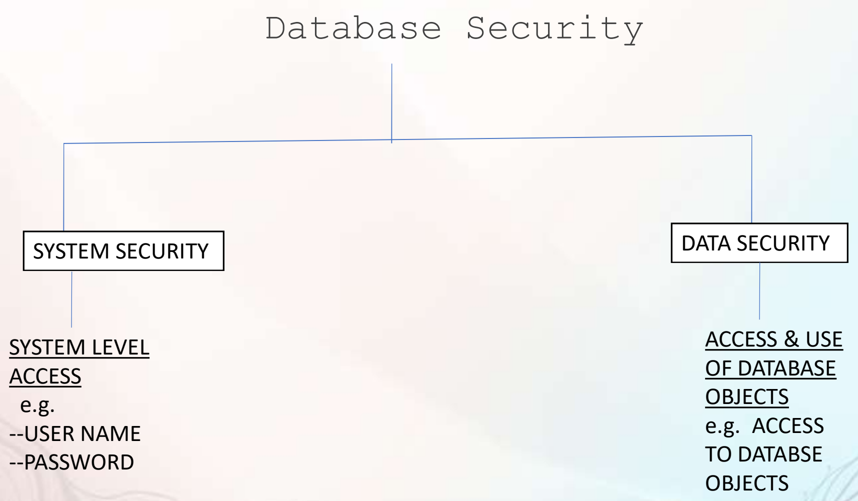
- Two lock modes:
  - Exclusive: Locks out other users
  - Share: Allows other users to access
- High level of data concurrency:
  - DML: Table share, row exclusive
  - Queries: No locks required
  - DDL: Protects object definitions
- Locks held until commit or rollback

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

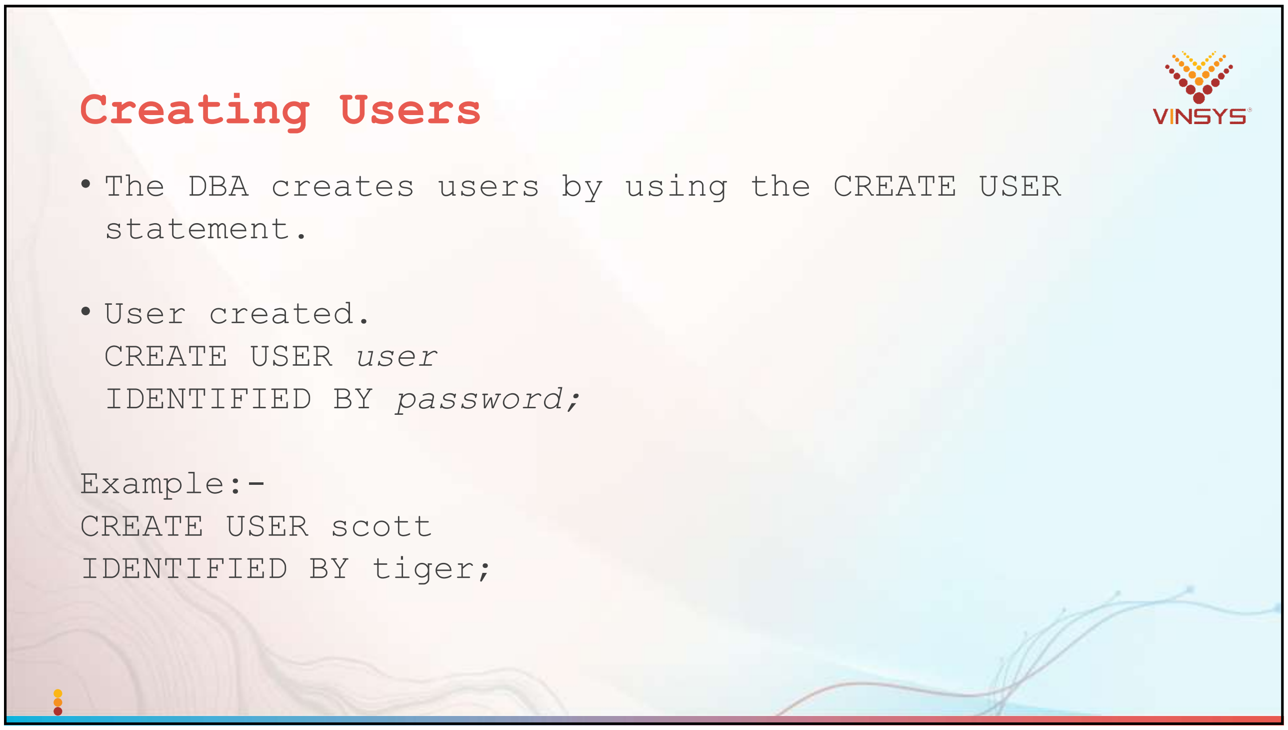
Controlling User Access



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

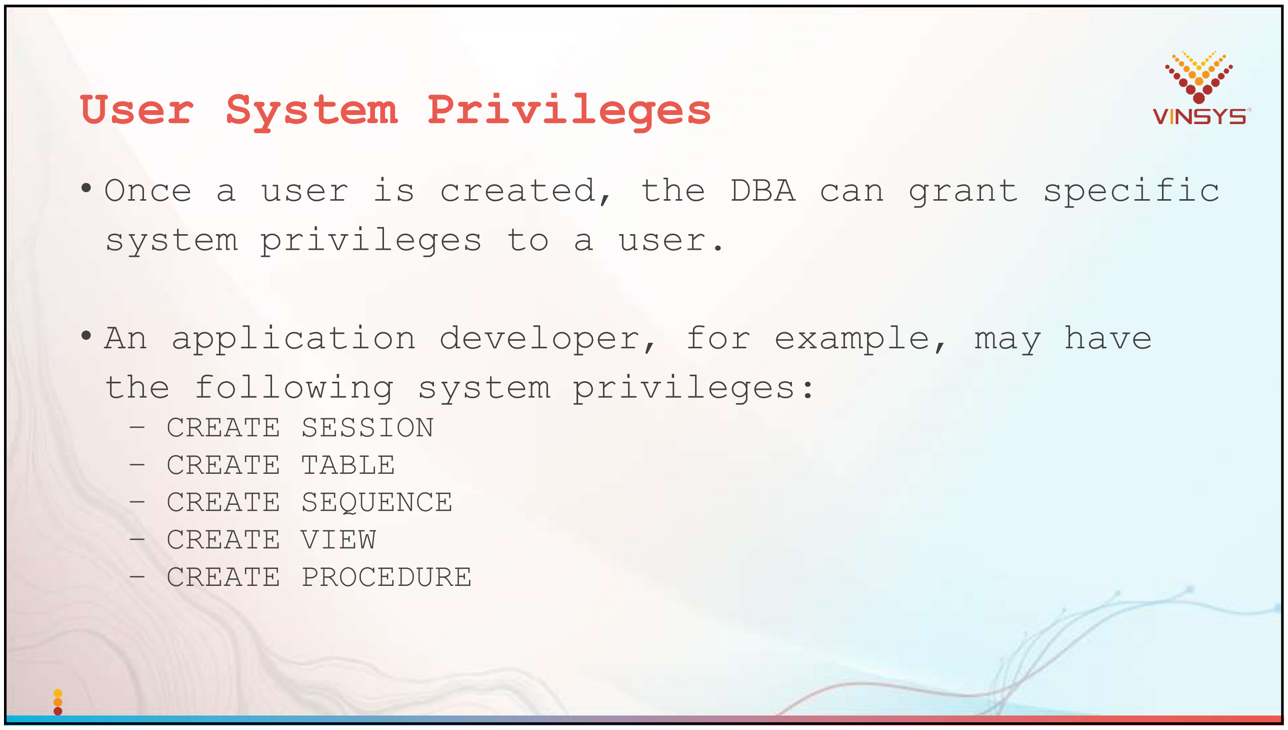


Creating Users

- 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.

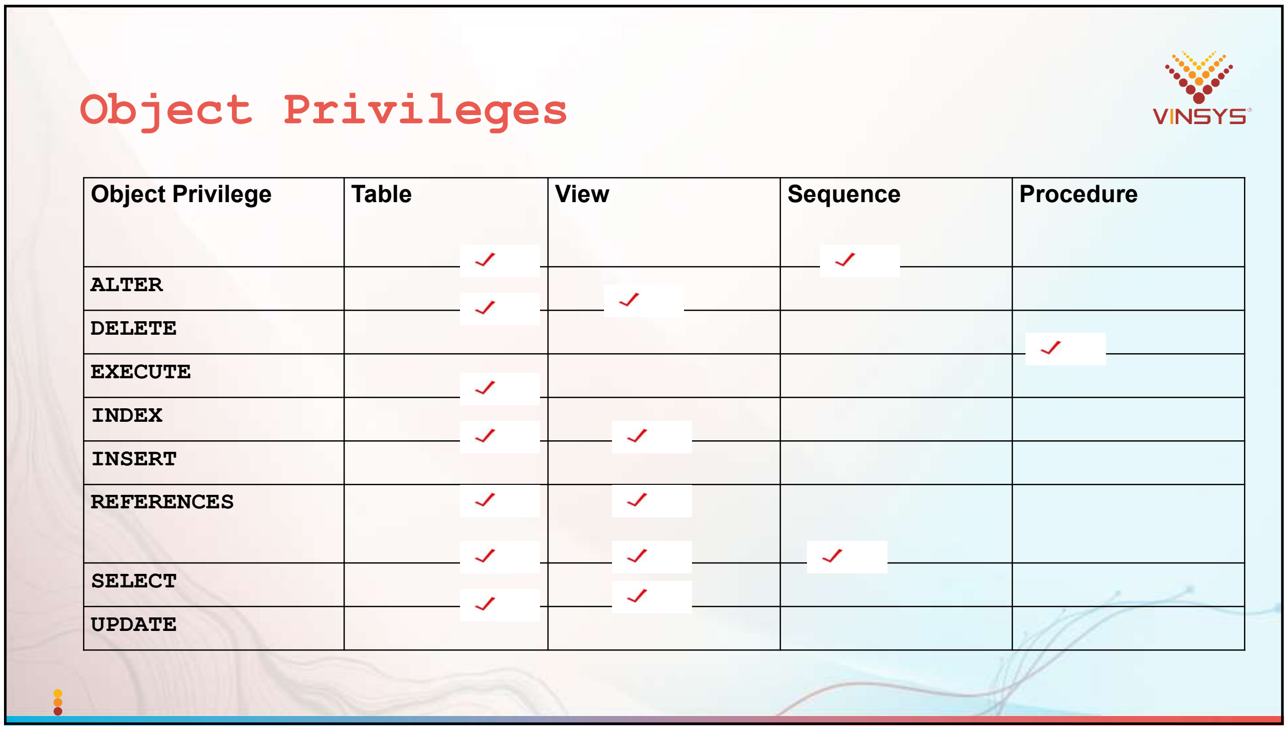
- You can change your password by using the
- ALTER USER statement.

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

OBJECT PRIVILEGES

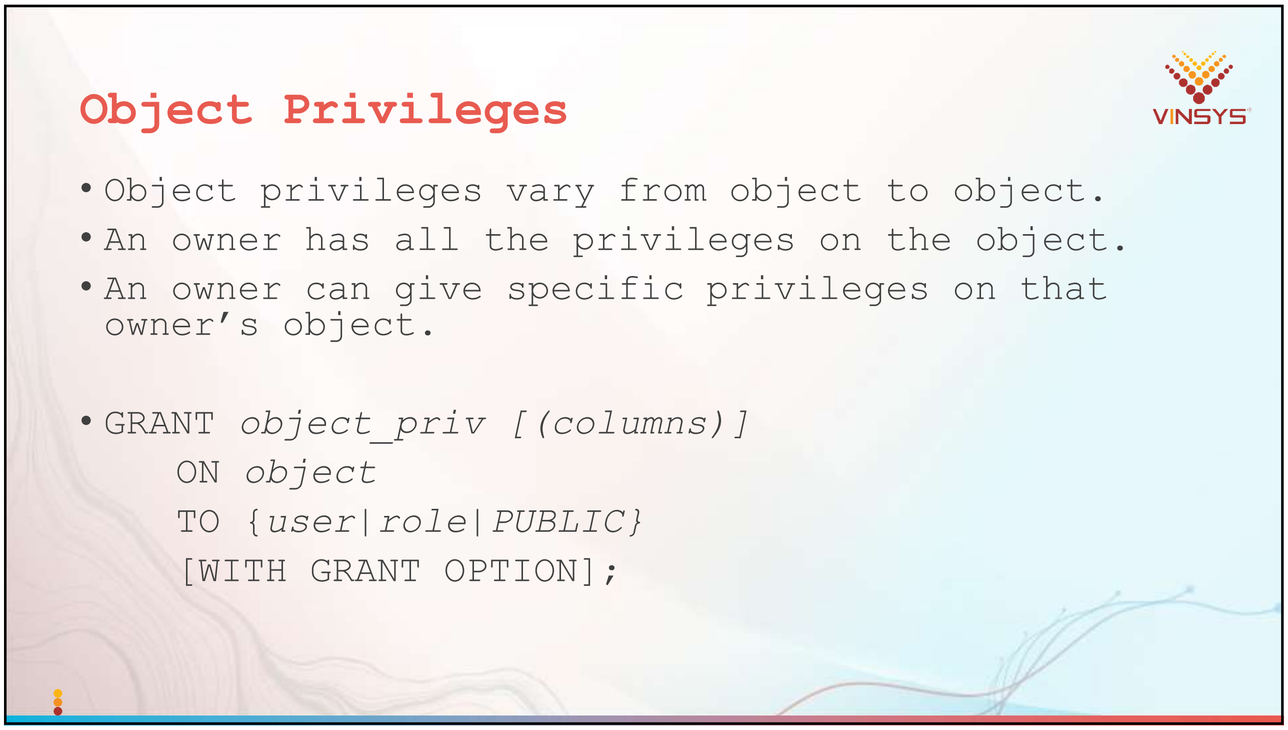
Object Privileges

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



Object Privileges

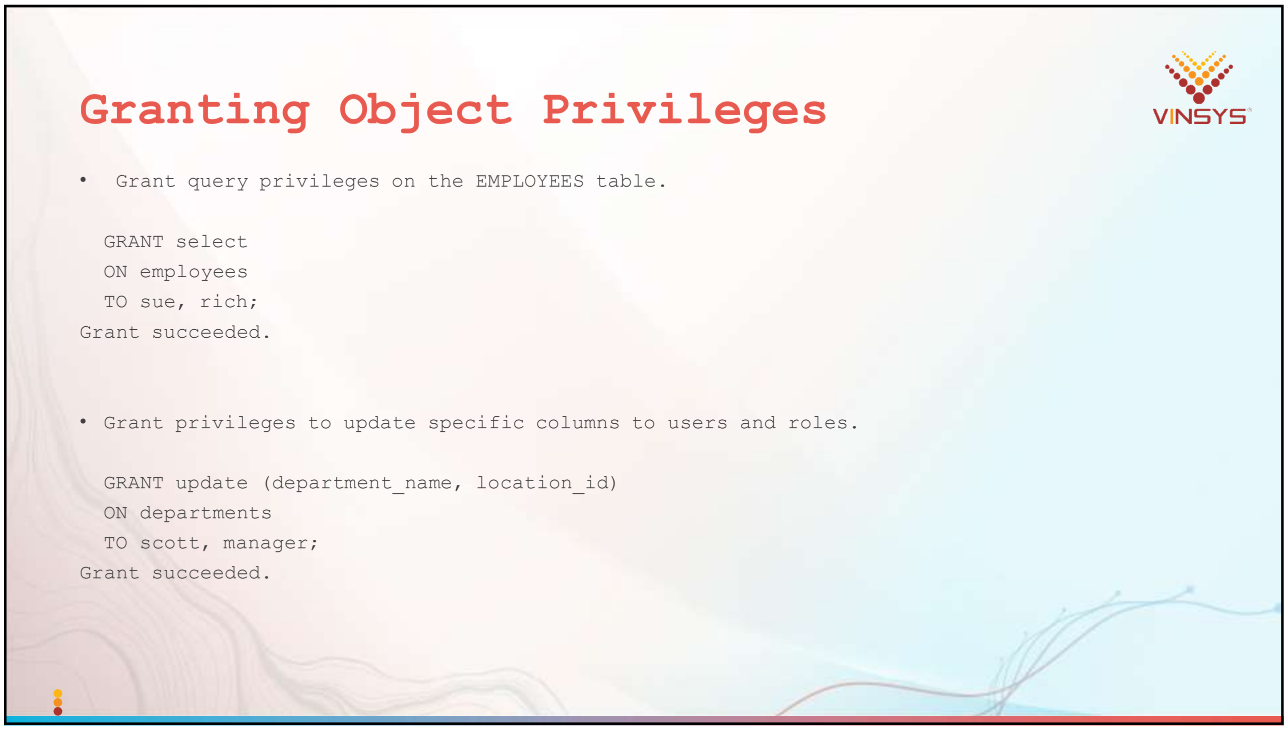
Object Privilege	Table	View	Sequence	Procedure
ALTER	✓		✓	
DELETE	✓	✓		
EXECUTE				✓
INDEX	✓			
INSERT	✓	✓		
REFERENCES	✓	✓		
SELECT	✓	✓	✓	
UPDATE	✓	✓		



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];



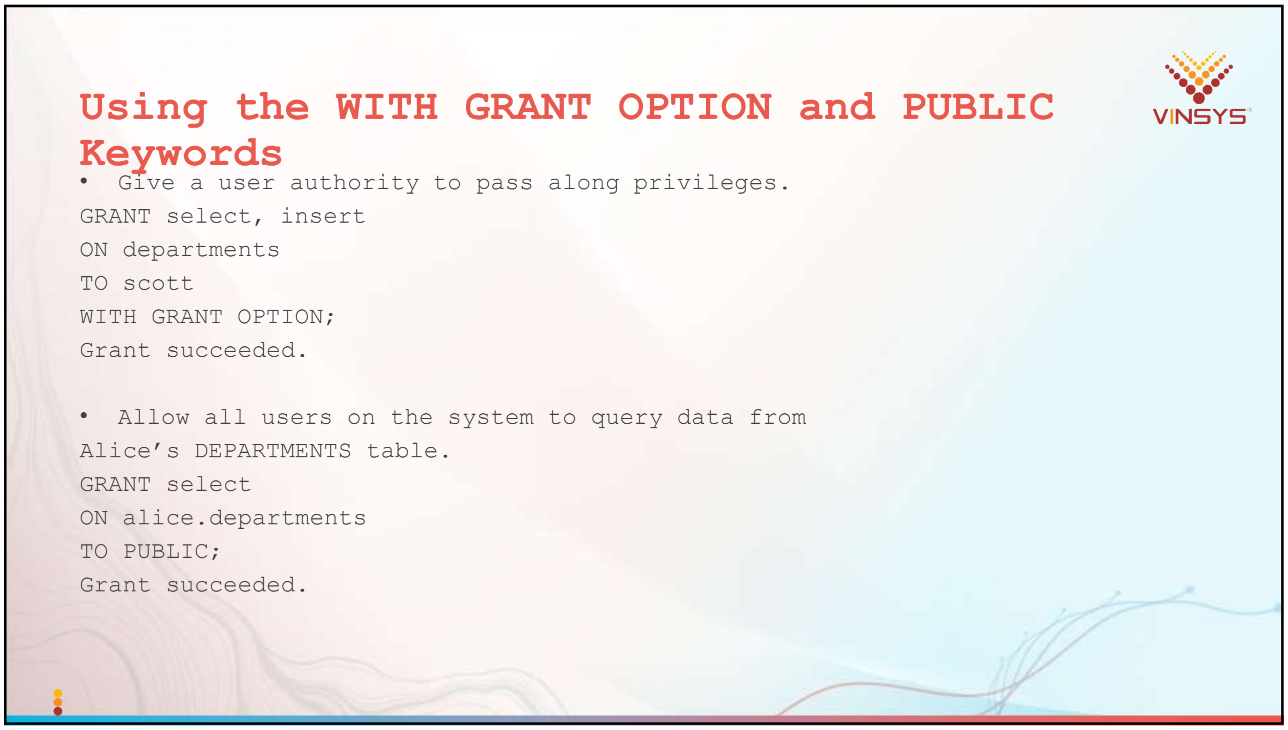
Granting Object Privileges

- 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];

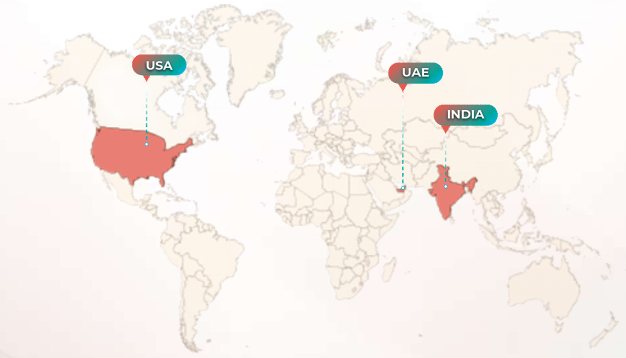
Example

- 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