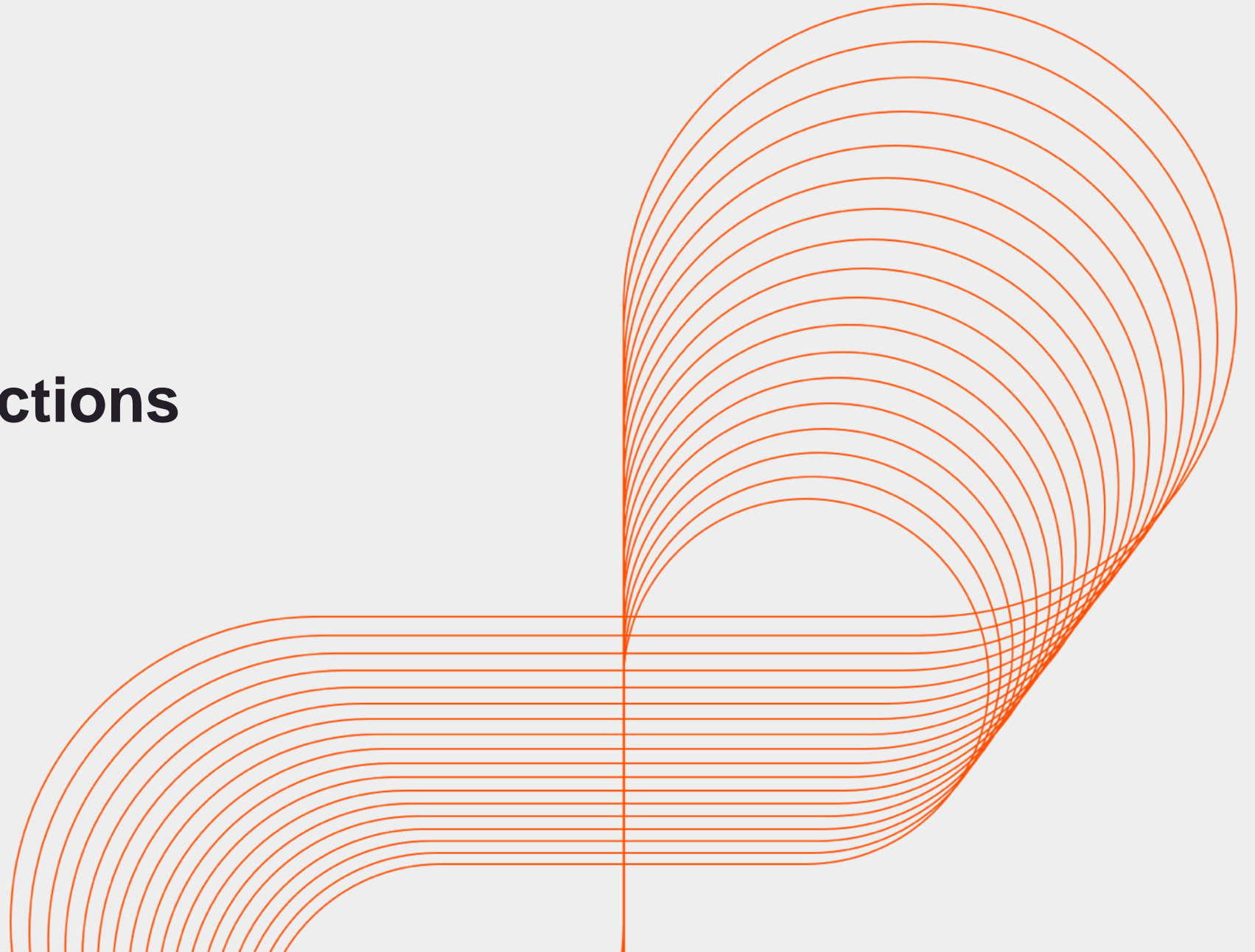




Nugget 5: Functions

Persistent University



Key Learning Points

- 1. Single row Functions**
- 2. Conditional Expressions**
- 3. Aggregate Functions**
- 4. Group by and Having clauses**

Sample Data

Table Name : EMPLOYEE

EmployeeId	FirstName	LastName	Email	PhoneNumber	HireDate	JobId	Salary	CommisionPct	ManagerId	DepartmentId
1	John	Demn	JohnD@yahoo.com	9898780979	1/10/2001	IT_PROF	70000	0.5	NULL	10
2	Ken	Dale	kendaleD@gmail.com	7877787655	4/1/2001	SALES_HEAD	50000	NULL	NULL	10
3	James	Walton	JW@yahoo.com	5787887888	1/1/2001	IT_REP	30000	0.2	1	20
4	robin	sngal	robin@gmail.com	4990988839	5/1/2001	SALES_REP	40000	0.3	2	20
5	ajay	ghosala	ghosala@hotmail.com	9809888898	6/10/2002	SALES_REP	30000	0.4	2	20

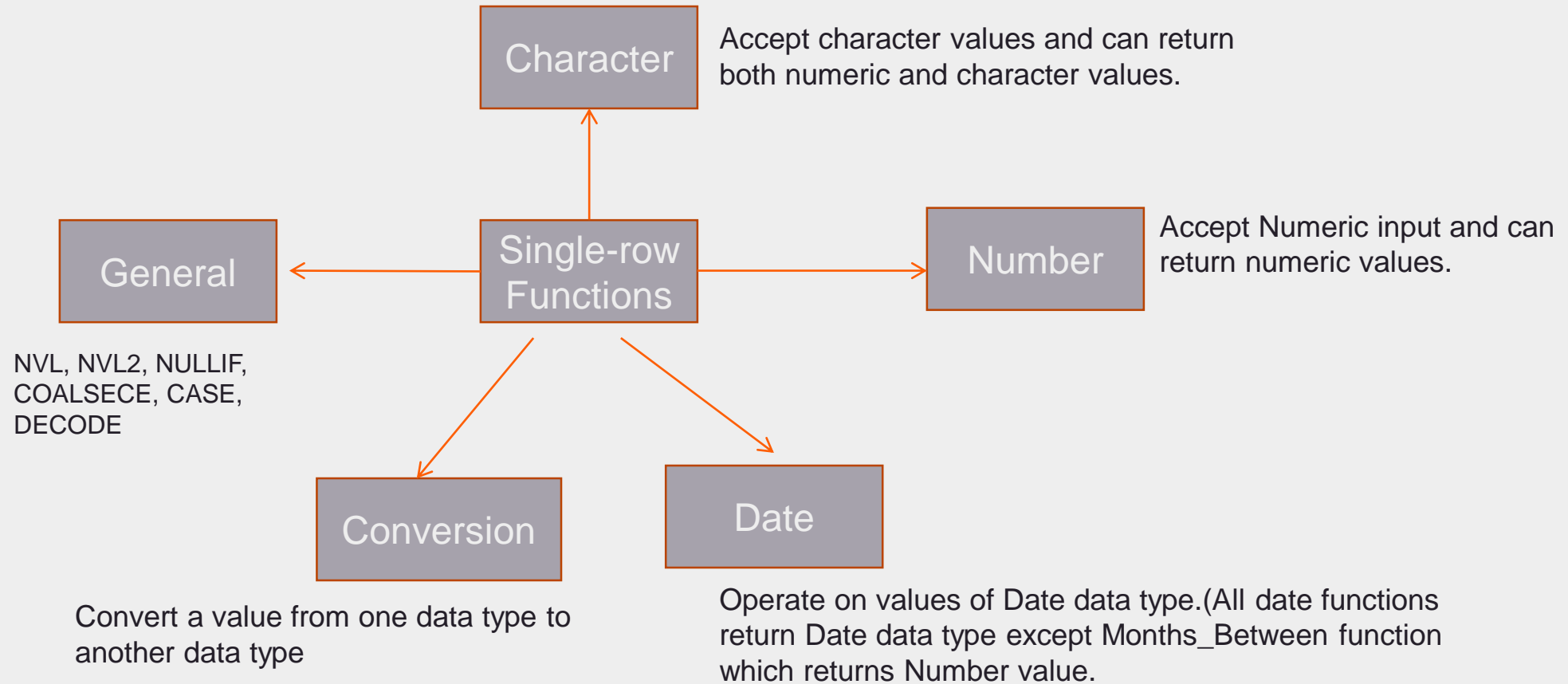
Functions

- Functions are very powerful feature of SQL and can be used to do the following:
 - Perform calculation on data.
 - Modify individual data items.
 - Manipulate output of group of rows.
 - Format date and numbers for display.
 - Convert column data types.
- SQL functions sometimes takes argument and always returns value.
- There are two distinct types of SQL functions
 - Single Row Functions
 - Multiple Row Functions

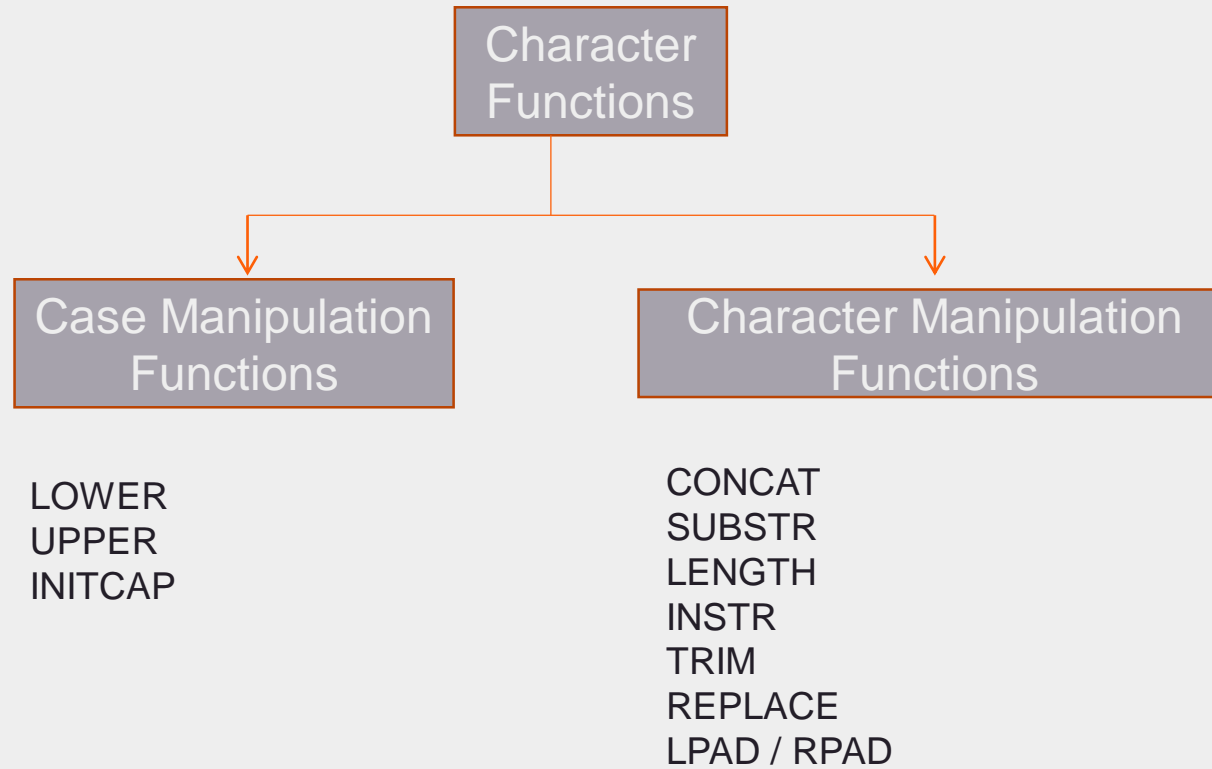
Single Row Functions

- Single Row Functions
 - Manipulate data items
 - Accept argument and returns value
 - Act on each row returned
 - Return one result per row
 - May Modify the datatype
 - Can be nested
 - Accept arguments which can be: constant, variable, expression or column name
 - FunctionName [(args1, args2,...)]

Single Row Functions



Character Functions



Character Functions

- LOWER(Column|Expression) : Converts alpha character values to lower case.
- UPPER(Column|Expression) : Converts alpha character values to upper case.
- INITCAP(Column|Expression) : Converts alpha character values to upper case for the first letter of each word, all other letters in lower case
- CONCAT(Column1|Expression1, Column2|Expression2) : concatenate the first character value to the second character value; equivalent to concatenation operator '||' .
- SUBSTR(Column|Expression, m[, n]) : Returns the string from Character value starting at character position m, n characters long. (If m is negative, the counting starts from end of the character value. If n is omitted all the characters till the end of the string are returned.)

Character Functions

- LENGTH(Column|Expression) : Returns the number of characters in the expression.
- INSTR(Column|Expression, 'string',[m],[n]) : Returns the numeric position of the named string in expression. m: position to start searching. n: nth occurrence of the string specified. m and n are optional. m and n defaults to 1.
- REPLACE(test, search_string,replace_string) : search_string, if appears in the TEXT will be replaced with replace_string.
- LPAD(Column|Expression, n, 'string') : Pads the string to the left of the character expression to make a total width of n.
- RPAD(Column|Expression, n, 'string') : Pads the string to the right of character expression to make a total width of n.
- TRIM(trimcharacter From Trimsource) : Enables user to trim the leading and trailing characters from a character string. If trimcharacter and trimsource is a character literal, enclose it in single quotations.

Character Function examples

- Select LOWER(FirstName) From Employee;
- Select LOWER ('HELLO World') From Dual; Output: - 'hello world'
- Select INITCAP('HELLO World') From Dual; Output: - 'Hello World'
- select CONCAT (FirstName, ' is a Employee') from Employee;
- Select SUBSTR('Please check it',8,5) from Dual; Output:- 'check'
- Select SUBSTR('Please check it',8) from Dual; Output:- 'check it'
- Select SUBSTR('Please check it',-8,8) from Dual; Output:- 'check it'
- Select Email, SUBSTR(EMAIL,4) from Employee;
- Select REPLACE('It is raining','is','was') from dual; Output:- 'It was raining'

Note: Dual table is owned by User SYS and can be accessed by all the users. It contains one column DUMMY and one row with value. The DUAL table is generally used for SELECT syntax completeness, because both SELECT and FROM clauses are compulsory, and several calculation do not need to select from actual tables.

Character Function examples

- Select Email, LENGTH(Email) from Employee;
- Select INSTR('it is really nice', 'r') from dual; Output: 7
-- Position of 1st occurrence of character 'r' starting from 1st position.
- Select INSTR('it is really nice', 'i', 3) from dual; Output: 4
-- Position of 1st occurrence of character 'i' starting from 3rd position.
- Select INSTR('it is really nice', 'i', 1, 3) from dual; Output: 15
-- Position of 3rd occurrence of character 'i' starting from 1st position.
- Select email, INSTR(email, 'com') from employee;
- Select LPAD('IT_REP', 10, '*') from dual; Output: '****IT_REP'
- Select RPAD('SALES_REP', 15, '*') from dual; Output: 'SALES_REP*****'

Number Functions

- ROUND(Column|Expression, n) : Rounds the column or expression/value to n decimal places If n is omitted, no decimal places. If n is negative, numbers to the left of decimal places are rounded
- TRUNC(Column|Expression, n) :Truncates the column or expression/value to n decimal places If n is omitted, it defaults to 0.
- MOD (m,n) : Returns of reminder of m divided by n

Examples

- Select ROUND(45.923,2), ROUND(45.928,2),ROUND(45.923,0)
from dual;
- Select TRUNC(45.923,2), TRUNC(45.928,2),TRUNC(45.923,0)
from dual;
- Select lastName, salary, MOD(salary,3000) from Employee;

ROUND(45.923,2)	ROUND(45.928,2)	ROUND(45.923,0)
45.92	45.93	46

TRUNC(45.923,2)	TRUNC(45.928,2)	TRUNC(45.923,0)
45.92	45.92	45

LASTNAME	SALARY	MOD(SALARY,3000)
Demn	70000	1000
Dale	50000	2000
Walton	30000	0
sngal	40000	1000
ghosala	30000	0

Working with Dates

- Oracle Default Date format is DD-MON-RR
- SYSDATE is a function that returns current database server date and time.
- Arithmetic with Dates:
 - Add or subtract a number to or from a Date
Date + number, date – number
 - Subtract 2 dates to find the number of days between those dates
Date – Date
 - Add hours to a date by dividing the number of hours by 24.
date + number/24

Date Functions

- MONTHS_BETWEEN(date1, date2): Finds the number of months between 2 dates. If date1 is later than date2 then result is positive, else it is negative.
- ADD_MONTHS(date, n): Adds n number of calendar months to date. N must be an integer and can be negative.
- NEXT_DAY(date, 'char'): Finds the date of the next specified day of the week following the 'date'.
- LAST_DAY(date): Find the date of the last day of the month that contains the input date.
- TRUNC(date, 'fmt'): Returns the date truncated to the unit specified by format model. If format model is not specified, it is truncated to nearest day.
- ROUND(date, 'fmt'): Returns the date rounded to the unit specified by format model. If format model is not specified, it is rounded to nearest day.

Date Function Examples

- Select MONTHS_BETWEEN('01-MAR-2013', '01-MAR-2009') from dual; Output: 48
- Select MONTHS_BETWEEN('01-MAR-2013', '01-MAR-2014') from dual; Output: -12
- Select NEXT_DAY('04-FEB-2013','Friday') from dual Output: 08-FEB-13
- Select LAST_DAY('01-FEB-2013') from dual; Output: 28-FEB-13
- Select employeeid, hireDate,

```
trunc(months_between(sysdate, hireDate)) TENURE,  
add_months(hiredate,6) REVIEW, Next_day(hiredate, 'friday'),  
Last_day(hiredate) from employee  
where trunc(months_between(sysdate, hireDate))=150;
```

EMPLOYEEID	HIREDATE	TENURE	REVIEW	NEXT_DAY(HIREDATE, 'FRIDAY')	LAST_DAY(HIREDATE)
1	10-Jan-01	150	10-Jul-01	12-Jan-01	31-Jan-01
3	1-Jan-01	150	1-Jul-01	5-Jan-01	31-Jan-01

Date Function Examples

- `Select sysdate, round(sysdate,'MONTH'), round(sysdate, 'YEAR'),
trunc(sysdate,'MONTH'), trunc(sysdate, 'YEAR')`
`from dual;`

Note: SYSDATE is a date function which retrieves current system date and time.

If sysdate is 16th Jul 2013 then Output is as below

SYSDATE	ROUND(SYSDATE,'MONTH')	ROUND(SYSDATE, 'YEAR')	TRUNC(SYSDATE,'MONTH')	TRUNC(SYSDATE, 'YEAR')
16-JUL-13	01-AUG-13	01-JAN-14	01-JUL-13	01-JAN-13

If

SYSDATE	ROUND(SYSDATE,'MONTH')	ROUND(SYSDATE, 'YEAR')	TRUNC(SYSDATE,'MONTH')	TRUNC(SYSDATE, 'YEAR')
06-JUN-13	01-JUN-13	01-JAN-13	01-JUN-13	01-JAN-13

General Functions

General functions can work with any data type. They are basically related to usage of NULL.

- NVL(expr1, expr2) : Converts a null value to a actual value.
- NVL2(expr1, expr2, expr3) : if expr1 is not null, then it returns expr2, if expr1 is null then it returns expr3.
- NULLIF(expr1, expr2) : Compares 2 expressions and returns null if they are equal, or the first expression if they are not equal.
- COALESCE(expr1, expr2....,exprn): Returns the first non null expression in the list.

General Function Examples

- Select lastname, commitionpct, nvl(commitionpct,0),hiredate, nvl(hiredate,'01-Feb-12'),managerid, nvl(managerid,0)
from employee;

LASTNAME	COMMITIONPCT	NVL(COMMITIONPCT,0)	HIREDATE	NVL(HIREDATE,'01-FEB-12')	MANAGERID	NVL(MANAGERID,0)
DEMN	0.5	0.5	10-JAN-01	10-JAN-01	(NULL)	0
DALE	(NULL)	0	1-APR-01	1-APR-01	(NULL)	0
WALTON	0.2	0.2	1-JAN-01	1-JAN-01	1	1
SNGAL	0.3	0.3	1-MAY-01	1-MAY-01	2	2
GHOSALA	0.4	0.4	10-JUN-02	10-JUN-02	2	2

- Select lastname, salary, commitionpct,
nvl2(commitionpct, salary*commitionpct/100, 0) calculated_comm
from employee;

LASTNAME	SALARY	COMMITIONPCT	CALCULATED_COMM
Dale	50000	(null)	0
sngal	40000	0.3	120

Conversion Functions

- IMPLICIT Data type conversion: The Oracle server can automatically convert the following
 - From Varchar2/ CHAR To NUMBER
 - From Varchar2/ CHAR To Date
 - From Number to Varchar2
 - From Date to Varchar2

For Example:

```
Test_var varchar2(20)
```

```
Test_var=10
```

```
Test_var number
```

```
Test_var='10'
```

Conversion Functions

- EXPLICIT Data type Conversion: SQL provides three functions to convert a value from one data type to another.
 - To_char
 - To_number
 - To_Date
- To_char(Date, 'format_model'): Convert a date value to varchar2 character string with format model given.
 - Format model :
 - must be enclosed in single quotes
 - is separated from date by comma
 - can include any valid data format element.
 - uses fm element to remove padding blanks or leading 0s

Conversion Functions

- Some of the Elements of Date format model

YEAR HH

YYYY HH24

YY MI

MONTH SS

MON AM/PM

MM DAY

DD DY

WW

- Some suffixes in format model

- TH: Ordinal number (For eg. DDTH for 4th)

- SP: Spelled Out number (For eg DDSP for FOUR)

- SPTH or THSP : Spelled out ordinal numbers (For eg. DDSPTH for Fourth)

Conversion Functions

- To_char(Number, 'format_model'): Convert a numeric value to varchar2 character string with format model given.
 - Format model
 - must be enclosed in single quotes
 - is separated from number by comma
 - can include any valid Number format element.
 - Some of the Elements of Number format model
 - 9 : Represents a number (Used to specify the width)
 - 0 : Forces to display 0.
 - \$: Places a dollar sign
 - L : Uses the local currency symbol
 - . : Prints decimal point
 - , : Prints a thousand indicator

Note: Oracle server displays a string of # sign in place of whole number if number of digits exceed the number of digits provided in the format model. Oracle server rounds the stored decimal value to the number of decimal spaces provided in the format model.

Conversion Functions

- To_number(char [, 'format_model']): Converts a character string to a number format
- To_date(char[, 'format_model']): Converts a character string to a date format

Note: Format elements are same as seen in To_char function.

Conversion Function examples

• Select sysdate, to_char(sysdate, 'fmDD MONTH YYYY') from dual;

SYSDATE	TO_CHAR(SYSDATE,'FMDDMONTHYYYY')
16-JUL-13	16 JULY 2013

• Select sysdate, to_char(sysdate, 'fmDDth " of " MONTH YYYY HH24:MI:SS') from dual;

SYSDATE	TO_CHAR(SYSDATE,'FMDDTH"OF"MONTHYYYYHH24:MI:SS')
16-JUL-13	16TH of JULY 2013 15:02:43

• Select to_char(salary, '\$99,999.00') salary from employee;

SALARY
\$70,000.00
\$50,000.00
\$30,000.00
\$40,000.00
\$30,000.00

• Select to_number('10,000.00','99,999.00') "Numeric value" from dual;

Numeric value
10000

• Select lastname,hiredate from employee

where hiredate=to_date('Jun 10, 2002','Mon DD, YYYY');

LASTNAME	HIREDATE
ghosala	10-JUN-02

Nesting Functions

- Single Row Functions can be nested to any level.
- Nested functions are evaluated from innermost level to outermost.

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

- Example:

```
select replace(lpad(substr('I am best', 3),10,'*'),'*','<') from dual;
```

Output of substr('I am best', 3)	:	'am best'	..Which will go as input to lpad
Output of lpad('am best',10,'*')	:	'***am best'	..Which will go as input to replace
Output of Replace('***am best','*','<')	:	'<<<am best'	

Conditional Expressions

- Conditional expressions provides the use of IF-Then-Else logic within a SQL statement.

There are two types of conditional expressions.

CASE Expression

DECODE Function

Conditional Expressions

- CASE Syntax:

```
CASE Expr          WHEN comparison_expr1 THEN return_expr1
                    [WHEN comparison_expr2 THEN return_expr2
                     WHEN comparison_exprn THEN return_exprn
                     ELSE else_expr]
END
```

- Oracle searched for the first WHEN ..THEN pair for which expr is equal to comparison_expr and returns the corresponding result_expr. If none of WHEN....THEN pair meet the condition and if ELSE clause exists, it returns else_expr.
- User can not specify literal NULL for the return_expr or else_expr.
- Expressions expr and comparison_expr must be of same data type; which can be CHAR, varchar2, number etc

Conditional Expressions

- DECODE Function Syntax:

```
DECODE(col/expression,  
        search1, result1  
        [search2, result2,.....  
        searchn, resultn]  
        [,default] )
```

- The DECODE function decodes an expression in IF-THEN-ELSE Logic.
- It decodes expression after comparing it with each search value. If the expr is same as search then result is returned.
- If default value is omitted, a null value is returned where expression does not match any of the search values.

Conditional Expression examples

- `Select employeeid, lastname, jobid, salary, Case Jobid when 'SALES_REP' then '1.05*salary'`
`when 'SALES_HEAD' then '1.12*salary'`

Else 'salary'

END "Calculation for Revised Salary"

From Employee;

- ```
Select employeeid, lastname, jobid, salary, DECODE(Jobid, 'SALES_REP', '1.05*salary',
 'SALES_HEAD', '1.12*salary',
 Salary) "Calculation for Revised Salary"
```

From Employee;

-- Output of above both the queries is same.

| EMPLOYEEID | LASTNAME | JOBID      | SALARY | Calculation for Revised Salary |
|------------|----------|------------|--------|--------------------------------|
| 1          | Demn     | IT_PROF    | 70000  | salary                         |
| 2          | Dale     | SALES_HEAD | 50000  | 1.12*salary                    |
| 3          | Walton   | IT_REP     | 30000  | salary                         |
| 4          | sngal    | SALES_REP  | 40000  | 1.05*salary                    |
| 5          | ghosala  | SALES_REP  | 30000  | 1.05*salary                    |

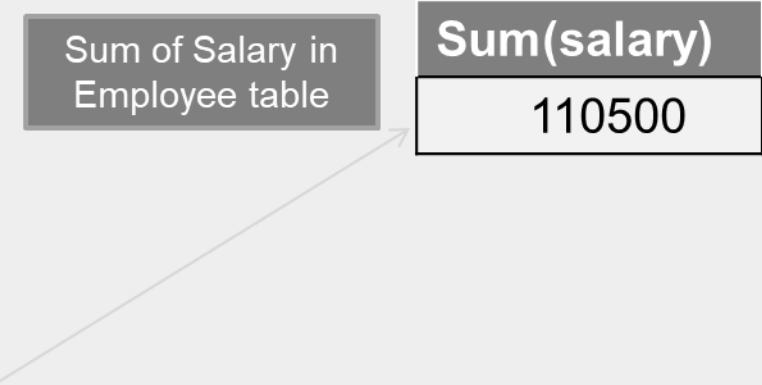
## Multiple row Functions/ Group Functions

- Group Functions operate on sets of rows to return one result per group.
- These sets may be whole table or table split into groups based on criteria.

| Emp_id | Salary |
|--------|--------|
| 1      | 5000   |
| 2      | 4000   |
| 3      | 50000  |
| 4      | 14500  |
| 5      | 23000  |
| 6      | 14000  |

Sum of Salary in Employee table

| Sum(salary) |
|-------------|
| 110500      |



## Types of Group functions

- Avg([Distinct/All] n): Average value of n; ignoring null values.
- Max([Distinct/All] expr): Maximum value of expression; ignoring null values.
- Min([Distinct/All] expr): Minimum value of expression; ignoring null values.
- Sum([Distinct/All] n): Sum values of n; ignoring null values.
- Count({\*| [Distinct/All] expr}) : Number of rows, where expr evaluates to something other than null. Count(\*) counts all rows including null and duplicates.
- STDDEV([Distinct/All] n)
- Variance([Distinct/All] n)

# Group Functions

Guidelines for using Group Functions:

- DISTINCT makes the function consider only non duplicate values; ALL makes it consider every value including duplicates. The default is ALL.
- Functions with the expr argument can accept CHAR, VARCHAR2, NUMBER and Date data type.
- All group functions ignore null values by default.



## Using Group functions

- Using AVG and SUM Functions : Can be used only against NUMERIC Data type.

```
SELECT avg(salary) ,sum(salary) from Employee;
```

| AVG(SALARY) | SUM(SALARY) |
|-------------|-------------|
| 44000       | 220000      |

```
SELECT avg(salary), round(avg(commitionpct),2) avg_comm from employee;
```

| AVG(SALARY) | AVG_COMM |
|-------------|----------|
| 44000       | 0.35     |

- Using MIN and MAX Functions : Can be used against any data type.

```
SELECT MIN(salary) ,MAX(salary) from Employee;
```

| MIN(SALARY) | MAX(SALARY) |
|-------------|-------------|
| 30000       | 70000       |

```
SELECT MIN(hiredate), MAX(hiredate) from employee;
```

| MIN(HIREDATE) | MAX(HIREDATE) |
|---------------|---------------|
| 01-JAN-01     | 10-JUN-02     |

```
SELECT MIN(firstname), MAX(firstname) from employee;
```

| MIN(FIRSTNAME) | MAX(FIRSTNAME) |
|----------------|----------------|
| James          | Robin          |

## Using Group Functions

- Using the COUNT Function : Can be used with any data type. It has 3 formats:
  - Count(\*): Returns the number of rows in a table including duplicate and rows containing null values in any of the columns.
  - Count(expr): Returns the number of non null values in the column specified by *expr*.
  - Count(distinct expr): Returns the number of unique, non null values in the column specified by *expr*.

### Examples:

- |                                                            |           |                                 |
|------------------------------------------------------------|-----------|---------------------------------|
| - SELECT count(*) from employee;                           | Output: 5 |                                 |
| - SELECT count(*) from employee where firstname like 'J%'; | Output: 2 |                                 |
| - SELECT count(commitionpct) from employee;                | Output: 4 | :- ignores null values          |
| - SELECT count(jobid) from employee;                       | Output: 5 | :- Considers duplicates as well |
| - SELECT count(distinct jobid) from employee;              | Output: 4 |                                 |

## Grouping Data using Group by clause

- Group By Clause is used to divide the rows in the table into groups. We can then use group functions to return the summary information per group.
- Syntax :

SELECT [column1, ] **group\_function(Column),...**

From table

[where Condition]

**[Group by group\_by\_expression]**

[Order by column];

## Grouping Data using Group by clause

- Guidelines:

- When group function is present in the select list, user cannot select individual rows as well; unless the individual column appears in the group by clause.
- Using WHERE clause, rows can be excluded before dividing them into groups.
- Column names must be included in the Group By clause. Column alias is not allowed in Group by clause.
- All columns in the SELECT list; that are not in group function must be in a group by clause.

## Excluding Group Results

- WHERE clause to restrict the rows that you select. In the same way, HAVING clause is used to restrict the groups.
- Syntax:  
Select [column1, ] **group\_function(Column),...**  
  
From table  
  
[where Condition]  
  
**[Group by group\_by\_expression]**  
  
**[Having group\_condition]**  
  
[Order by column];
- Oracle server performs the following steps:
  - Rows are grouped
  - Group function is applied to the group
  - Groups that match the criteria in the having clause are displayed.

## Using Group By and Having clauses

- SELECT departmentid, sum(salary)

FROM employee

GROUP BY departmentid;

| DEPARTMENTID | SUM(SALARY) |
|--------------|-------------|
| 10           | 120000      |
| 20           | 100000      |

- SELECT departmentid, jobid, sum(salary)

FROM employee

GROUP BY departmentid, jobid;

| DEPARTMENTID | JOBID      | SUM(SALARY) |
|--------------|------------|-------------|
| 10           | SALES_HEAD | 50000       |
| 10           | IT_PROF    | 70000       |
| 20           | IT_REP     | 30000       |
| 20           | SALES_REP  | 70000       |

-- Grouping data based on multiple columns

- SELECT departmentid,max(salary)

FROM employee

GROUP BY departmentid

HAVING max(salary) >45000;

| DEPARTMENTID | MAX(SALARY) |
|--------------|-------------|
| 10           | 70000       |

## Sequence of all clauses in SELECT statement

- **SELECT** jobid, SUM(salary)  
**FROM** employee  
**WHERE** jobid NOT LIKE '%PROF%'  
**GROUP BY** jobid  
**HAVING** SUM(salary)>=50000  
**ORDER BY** SUM(salary)

| JOBID      | SUM(SALARY) |
|------------|-------------|
| SALES_HEAD | 120000      |
| SALES_REP  | 100000      |

## Illegal queries using Group by clause

- `SELECT departmentId, sum(salary) FROM Employee;`

Error:- ORA 00937 : Not a single group group function

Note: All columns/expressions in the SELECT list that is not in aggregate function must be in the group by clause.

- `SELECT departmentid, AVG(salary)`  
`FROM employee`  
`WHERE AVG(salary) > 10000`  
`GROUP BY departmentid;`

Error:- ORA 00934 : Group function is not allowed here.

Note: Group function is not allowed in the Where clause and Where clause can not be used to restrict the groups. There is a Having clause for this purpose.



## Session 5: Summary

With this we have come to an end of our fifth session where we discussed various types of Functions.

- At the end of Nugget 5, we see that you are now able to answer following questions:
  - Explain “Various Character functions”.
  - Explain “Date functions in detail”
  - Explain “Various Numeric Functions”
  - Explain “Conversion functions in detail”
  - Explain “Various Group functions with examples”
  - Explain the importance of “GROUP BY and Having clause”.

## Reference Material: Sites

<http://beginner-sql-tutorial.com/oracle-functions.htm>

[http://www.oracle-dba-online.com/sql/oracle\\_sql\\_functions.htm](http://www.oracle-dba-online.com/sql/oracle_sql_functions.htm)

[http://docs.oracle.com/cd/E11882\\_01/server.112/e26088/functions002.htm](http://docs.oracle.com/cd/E11882_01/server.112/e26088/functions002.htm)

[http://vpf-web.harvard.edu/applications/ad\\_hoc/key\\_functions\\_in\\_oracle\\_sql.pdf](http://vpf-web.harvard.edu/applications/ad_hoc/key_functions_in_oracle_sql.pdf)

[http://psoug.org/reference/group\\_by.html](http://psoug.org/reference/group_by.html)



**Persistent**

**Thank you!**

Persistent University

