Lab 04

Single row and multiple row Functions

SQL Functions

Introduction

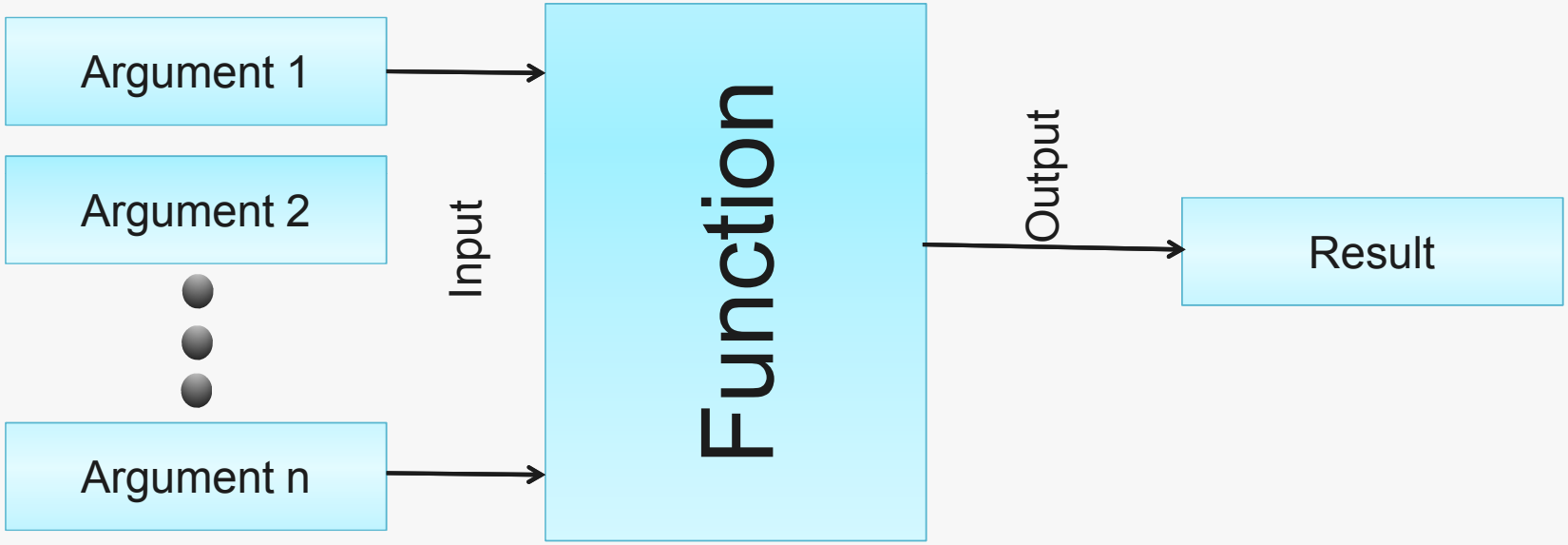
Functions are used to perform following tasks:

- ✓Perform calculation on data
- ✓Modify(DML) individual data items
- ✓Manipulate output for Group of rows
- ✓Format Date and numbers for display
- ✓Convert column data types

Function takes arguments and **must always** return a value

SQL Functions

Functions Working



Output = Function (argument1,argument2,…..,argumentn)

Types of SQL Functions

1. Single Row Functions

Function that operate on single row and return one result per row

Examples are:

- ✓ Character function
- ✓ Number function
- ✓ Date function
- ✓ Conversion Function

2. Multiple row functions

These Functions work on group of rows and give result then like aggregate functions(count,Avg,sum)

1

Single-Row Functions

Single Row Functions

1. Character Functions

Single row character functions accept data and can return character & numberic values.character functions can be devided into following:

1. Case Conversion Functions
2. Character Manipulation functions

Single Row Functions

1. Case Conversion Functions

Function	Results
Lower('SQL Course')	Sql course
UPPER('SQL Course'	SQL COURSE
INITCAP('SQL Course')	Sql Course

Example 1: To print an employee name (first letter capital) and job title (lower case)

Text goes here

SELECT 'The job title for ' || INITCAP(ename) || ' is ' || LOWER(job) AS "EMPLOYEE DETAILS" FROM emp;

Single Row Functions

1. Case Conversion Functions(cont)

Example 2: To display the employee number, name (in upper case) and department number for employee *Blake.*

```
SELECT empno, UPPER(ename), deptno
FROM emp
WHERE LOWER(ename) = 'blake';
```

2. Character manipulation functions

Function	Results
CONCAT('Good','String')	GoodString

Single Row Functions

2. Character manipulation functions(cont)

Function	Results
SUBSTR('String',2,4)	trin
LENGTH('String')	6
INSTR('String','r')	3
LPAD(SAL,10,'*')	******5000

Starting from left

It will LPDAD(Left PAD)

Provided characters as '*'

Single Row Functions

2. Character manipulation functions(cont)

Example 2: To display employee name and job joined together, length of employee name, and the numeric position of letter A in the employee name, for all employees who are in *sales.*

SELECT empno, CONCAT(ename, job), LENGTH(ename), INSTR(ename, 'A')

FROM emp WHERE SUBSTR(job, 1, 5) = 'SALES';

We can also join with dept table instead of SUBSTR

```
  1  SELECT empno, CONCAT(ename, job), LENGTH(ename), INSTR(ename, 'A')
  2* FROM emp WHERE SUBSTR(job, 1, 5) = 'SALES'
SQL> /

     EMPNO CONCAT(ENAME,JOB)   LENGTH(ENAME) INSTR(ENAME,'A')
---------- -------------------- ------------- ----------------
      7499 ALLENSALESMAN                    5                1
      7521 WARDSALESMAN                     4                2
      7654 MARTINSALESMAN                   6                2
      7844 TURNERSALESMAN                   6                0
```

What if I want to print space between Ename & Job

Single Row Functions

2. Character manipulation functions(cont)

SELECT empno, CONCAT(ename,' '|| job) , LENGTH(ename), INSTR(ename, 'A') FROM emp WHERE SUBSTR(job, 1, 5) = 'SALES‘;

```
  1* SELECT empno, CONCAT(ename,' '|| job) , LENGTH(ename), INSTR(ename, 'A') FROM emp WHERE SUBSTR(job, 1, 5) = 'SALES'
SQL> /

     EMPNO CONCAT(ENAME,''||JOB LENGTH(ENAME) INSTR(ENAME,'A')
---------- -------------------- ------------- ---------------
      7499 ALLEN SALESMAN                    5               1
      7521 WARD SALESMAN                     4               2
      7654 MARTIN SALESMAN                   6               2
      7844 TURNER SALESMAN                   6               0
```

2. Number Functions

It accepts input and return numeric values

Single Row Functions

1. Round

Syntax: ('Column name' or numeric expression or value , n)

OR

ROUND(input,roundto)

So n is the no of decimal places until which we need to round

Positive n works till nth number after decimal point starting from left
Negative n works for nth number before decimal point starting from Right

Note :Dual is a dummy table one row with value X and 1 column (DUMMY) created by oracle. This table is created by SYS user so accessed by all users(from scott you are viewing)
There is no need of Dual table in SQL Server at all.

Oracle:
select 'something' from dual // ***from*** keyword is compulsory in Oracle

SQL Server:
SELECT 'something'

Single Row Functions

1. Round(cont)

Example 1: SELECT ROUND(3162.845,1) from dual

```
  1* select Round(3162.845,1) from dual
SQL> /

ROUND(3162.845,1)
-----------------
           3162.8
```

1 means after decimal we need 1 number Only. means 8 needs to be present over there Where it is now but should have impact of Removing number on it.

Note: 1-4 will not ceil the number (range 1)

5-9 will ceil the number (range 2)

This rule will implement for 10's 100's & so on

Example 2: SELECT ROUND(3162.8451297,5) from dual

```
  1* select Round(3162.8451297,5) from dual
SQL> /

ROUND(3162.8451297,5)
---------------------
           3162.84513
```

5 means we need 84512 so 9 will make 2 to 3 After rounding

Single Row Functions

1. Round(cont)

Example 3: SELECT ROUND(3162.845,-2) from dual

```
  1* SELECT ROUND(3162.845,-2) from dual
SQL> /

ROUND(3162.845,-2)
------------------
              3200
```

Before decimal we dnt remove the numbers but
We make the numbers in zeros format
Here -2 says I need first right two digits to replace
With 00 so 6 which occurs in **range 2**
will make the 1 to 2 with “62” to “00”

Example 4: SELECT ROUND(142786,-3) from dual

```
  1* SELECT ROUND(142786,-3) from dual
SQL> /

ROUND(142786,-3)
----------------
          143000
```

Here -3 says I need first right three digits to replace with
“000” and 7 which occurs in range 2 will make the 2 to 3
With “786” to “000”

Single Row Functions

1. Round(cont)

Example 5: SELECT ROUND(3162.845,0) from dual

```
  1* SELECT ROUND(3162.845,0) from dual
SQL> /

ROUND(3162.845,0)
-----------------
             3163
```

Here 0 removes all numbers after decimal
And the first **Unit** digit **(like 2 here)**
will remain same if there is no number After decimal
other wise as 8 occurs in **range 2** so it will Ceil 2 to 3
with all decimals places to be remove

Example 6: SELECT ROUND(3162.845) from dual

By Default n= 0 so this query will produce same
Result as above

Example 7: SELECT ROUND(45.927) from dual

```
  1* SELECT ROUND(45.927) from dual
SQL> /

ROUND(45.927)
-------------
           46
```

Single Row Functions

1. Round(cont)

Example 8: SELECT ROUND(45.927,2) from dual

```
  1* SELECT ROUND(45.927,2) from dual
SQL> /

ROUND(45.927,2)
---------------
          45.93
```

2 means after decimal we need 2 number
Only means 92 needs to be present over there
Where it is now but should have impact of
Removing number on it. So 7 will change 2 to 3 as
It occurs in **range 2**

Example 9: SELECT ROUND(45.927,-1) from dual

```
  1* SELECT ROUND(45.927,-1) from dual
SQL> /

ROUND(45.927,-1)
----------------
              50
```

Here -1 says I need first right 1 digits to replace with
“0” and 5 which occurs in range 2 will make the 4 to 5
With “5” to “0”

Single Row Functions

2) Trunc

Syntax: ('Column name' or numeric expression or value , n)

OR

Trunc(input,Trunc to)

Example 1: SELECT Trunc(3162.845,1) from dual

```
  1* SELECT Trunc(3162.845,1) from dual
  2  /

TRUNC(3162.845,1)
-----------------
           3162.8
```

1 means after decimal we need 1 number
Only means 8 needs to be present over there
Where it is now .
Trunc will just erase the other numbers in after
Decimal case so 45 Will remove

Single Row Functions

2) Trunc(cont)

Example 2: SELECT TRUNC(45.927) from dual

```
  1* SELECT Trunc(45.927) from dual
  2  /

TRUNC(45.927)
-------------
           45
```

Example 3: SELECT TRUNC(142786,-3) from dual

Here -3 says I need first right three digits to replace with "000" while other digits will remain same we not change them

```
  1* SELECT TRUNC(142786,-3) from dual
SQL> /

TRUNC(142786,-3)
----------------
          142000
```

Single Row Functions

3) MOD(m,n) Returns the Remainder of *m* devided by *n*

MOD(m,n)

Lets see how we take a remainder MOD(20,3)

This is a remainder and Mod (20,3) will return 2

Example 1: calculate the remainder of the ratio of salary to commission for all employees whose job title is salesman.

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

```
  1  SELECT ename, sal, comm, MOD(sal, comm)
  2  FROM emp
  3* WHERE UPPER(job) = 'SALESMAN'
SQL> /

ENAME             SAL       COMM MOD(SAL,COMM)
---------- ---------- ---------- -------------
ALLEN            1600        300           100
WARD             1250        500           250
MARTIN           1250       1400          1250
TURNER           1500          0          1500
```

Single Row Functions

3. Date Functions

SYSDATE is a date function that returns the current date and time. The current date can be displayed by selecting SYSDATE from a from DUAL. Current date can be displayed by:

```
SELECT SYSDATE
FROM DUAL;
```

Arithmetic with Dates:

We can add or subtract a number from a date to find a resultant date.

Example : display the name and number of days & number of weeks for all employees in department 10.

```
SELECT ename,(SYSDATE - HIREDATE) "Number of day",
(SYSDATE - HIREDATE) / 7 "Number of Weeks"
FROM emp
```

```
  1  SELECT ename,(SYSDATE - HIREDATE) "Number of day",
  2  (SYSDATE - HIREDATE) / 7 "Number of Weeks"
  3  FROM emp
  4* WHERE deptno = 10
SQL> /

ENAME      Number of day Number of Weeks
---------- ------------- ---------------
CLARK         12834.1491      1833.44987
KING          12673.1491      1810.44987
MILLER        12606.1491      1800.87845
```

Single Row Functions

Date functions operate on Oracle dates. All date functions return a value of DATE datatype except MONTHS_BETWEEN, which returns a numeric value. Lets see some date Functions:

Function	Result	Description
MONTHS_BETWEEN('01-SEP-95', '11-JAN-94')	19.6774194	Number of months between two dates
ADD_MONTHS('11-JAN-94', 6)	'11-JUL-94'	Add calendar months to dates
NEXT_DAY('01-SEP-95', 'FRIDAY')	'08-SEP-95'	Next day of the date specified
LAST_DAY('01-SEP-95')	'30-SEP-95'	Last day of the month
ROUND(TO_DATE('25-JUL-95', 'DD-MON-YY'), 'MONTH')	01-AUG-95	Round date
ROUND(TO_DATE('25-JUL-95', 'DD-MON-YY'), 'YEAR')	01-JAN-96	Round date
TRUNC(TO_DATE('25-JUL-95', 'DD-MON-YY'), 'MONTH')	01-JUL-95	Truncate date
TRUNC(TO_DATE('25-JUL-95', 'DD-MON-YY'), 'YEAR')	01-JAN-95	Truncate date

Single Row Functions

Example 1: For all employees employed for fewer than 400 months, display the employee number,hiredate, number of months employed, six-month review date, first Friday after hiredate and last day of the month hired.

After 6 months probation there is a employee review.

```
SELECT empno, hiredate, MONTHS_BETWEEN(SYSDATE, hiredate) AS "TENURE",
ADD_MONTHS(hiredate, 6) as "Review date", NEXT_DAY(hiredate, 'FRIDAY'),
LAST_DAY(hiredate) FROM emp
WHERE MONTHS_BETWEEN(SYSDATE, hiredate) < 400;
```

```
  1  SELECT empno, hiredate, MONTHS_BETWEEN(SYSDATE, hiredate) TENURE,
  2  ADD_MONTHS(hiredate, 6) REVIEW, NEXT_DAY(hiredate, 'FRIDAY'),
  3  LAST_DAY(hiredate) FROM emp
  4* WHERE MONTHS_BETWEEN(SYSDATE, hiredate) < 400
SQL> /

     EMPNO HIREDATE      TENURE REVIEW    NEXT_DAY( LAST_DAY(
---------- --------- ---------- --------- --------- ---------
      7788 19-APR-87 351.329789 19-OCT-87 24-APR-87 30-APR-87
      7876 23-MAY-87 350.200756 23-NOV-87 29-MAY-87 31-MAY-87
```

Single Row Functions

Example 2: Comparing the hire dates for all employees who started in 1981 via displaying the employee number, hiredate, and month started using the ROUND and TRUNC functions.

SELECT empno, hiredate, ROUND(hiredate, ‘MONTH’),TRUNC(hiredate, ‘MONTH’) FROM emp
WHERE hiredate like ‘%81’;

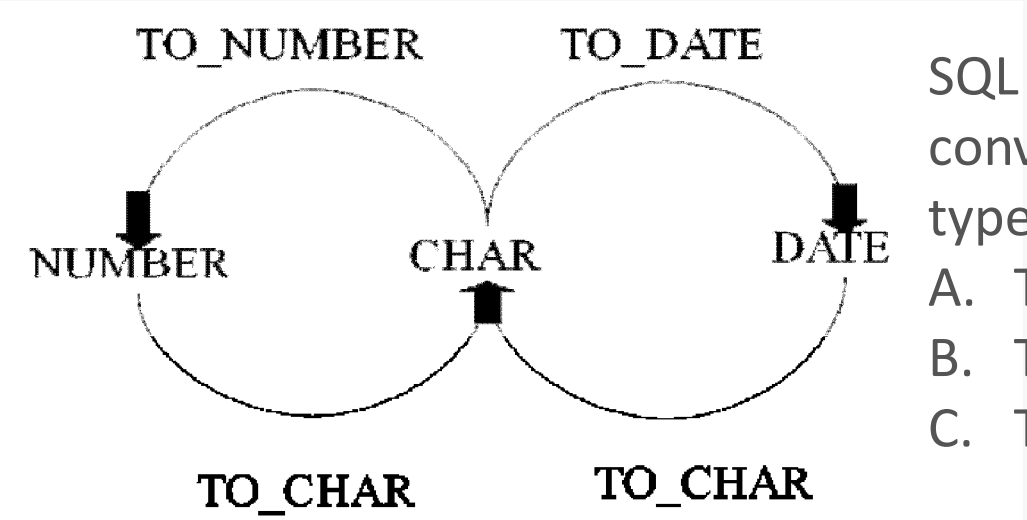
```
  1  SELECT empno, hiredate, ROUND(hiredate, 'MONTH'),TRUNC(hiredate,
  2  'MONTH') FROM emp
  3* WHERE hiredate like '%81'
SQL> /

     EMPNO HIREDATE  ROUND(HIR TRUNC(HIR
---------- --------- --------- ---------
      7499 20-FEB-81 01-MAR-81 01-FEB-81
      7521 22-FEB-81 01-MAR-81 01-FEB-81
      7566 02-APR-81 01-APR-81 01-APR-81
      7654 28-SEP-81 01-OCT-81 01-SEP-81
      7698 01-MAY-81 01-MAY-81 01-MAY-81
      7782 09-JUN-81 01-JUN-81 01-JUN-81
      7839 17-NOV-81 01-DEC-81 01-NOV-81
      7844 08-SEP-81 01-SEP-81 01-SEP-81
      7900 03-DEC-81 01-DEC-81 01-DEC-81
      7902 03-DEC-81 01-DEC-81 01-DEC-81

10 rows selected.
```

Single Row Functions

4. Conversion Functions



SQL provides three functions to convert a value from one data type to another.

A. TO_CHAR
B. TO_NUMBER
C. TO_DATE

Lets take few examples of TO_CHAR function with dates

Example 1: display Employee number,month number of hiredate and year of hiring of Mr Blake

```
SELECT empno, TO_CHAR(hiredate, 'MM/YY')
as "Month_Hired/Year_Hired"
FROM emp
WHERE ename = 'BLAKE'
```

```
  1  SELECT empno,
  2  TO_CHAR(hiredate, 'MM/YY') as "Month_Hired/Year_Hired"
  3  FROM emp
  4* WHERE ename = 'BLAKE'
SQL> /

     EMPNO Month
---------- -----
      7698 05/81
```

Single Row Functions

Example 2: To display the employee name and hiredate for all employees. The hiredate appears as 17 November, 1981.

```
SELECT ename, TO_CHAR(hiredate, 'fmDD
Month YYYY') HIREDATE
FROM emp;
```

The *fm element called Format Mask is used to remove padded blanks or suppress leading zeros. Without fm query is as:*

```
SELECT ename, TO_CHAR(hiredate, 'DD Month
YYYY') HIREDATE
FROM emp;
```

The *fm element called Format Mask is used to remove padded blanks or suppress leading zeros.*

```
ENAME      HIREDATE
---------- ------------------
SMITH      17 December 1980
ALLEN      20 February 1981
WARD       22 February 1981
JONES      2 April 1981
MARTIN     28 September 1981
BLAKE      1 May 1981
CLARK      9 June 1981
SCOTT      19 April 1987
KING       17 November 1981
TURNER     8 September 1981
ADAMS      23 May 1987
JAMES      3 December 1981
FORD       3 December 1981
MILLER     23 January 1982
```

```
ENAME      HIREDATE
---------- ------------------
SMITH      17 December  1980
ALLEN      20 February  1981
WARD       22 February  1981
JONES      02 April     1981
MARTIN     28 September 1981
BLAKE      01 May       1981
CLARK      09 June      1981
SCOTT      19 April     1987
KING       17 November  1981
TURNER     08 September 1981
ADAMS      23 May       1987
JAMES      03 December  1981
FORD       03 December  1981
MILLER     23 January   1982

14 rows selected.
```

Single Row Functions

Example 3: To print the employee name and time of joining in format HH:MI:SS (Assuming that hiredate column were used for storing joining time)

```
SELECT ename, TO_CHAR(hiredate, 'HH:MI:SS') as
Time_of_Joining
FROM emp;
```

ENAME	TIME_OF_
SMITH	12:00:00
ALLEN	12:00:00
WARD	12:00:00
JONES	12:00:00
MARTIN	12:00:00
BLAKE	12:00:00
CLARK	12:00:00
SCOTT	12:00:00
KING	12:00:00
TURNER	12:00:00
ADAMS	12:00:00
JAMES	12:00:00
FORD	12:00:00
MILLER	12:00:00

14 rows selected.

Lets take few examples of TO_CHAR function with Numbers. Use to treat numbers as string

Single Row Functions

Example 4: To display the salary of employee SCOTT with $ sign preceded

```
SELECT TO_CHAR(sal, '$99,999') SALARY
FROM emp
WHERE ename = 'SCOTT'
```

The oracle server displays a string of pound signs (#) in place of a whole number(before decimal number) whose digits exceed the number of digits provided in the format model. You can experience it from this example

```
  1* SELECT TO_CHAR(54000, '99.999') from dual
SQL> /

TO_CHAR
-------
#######
```

```
  1* select to_char(23.456,'99.99') from dual
SQL> /

TO_CHA
------
 23.46
```

After decimal number *exceeding case* will be rounded of by oracle

Single Row Functions

Lets take a examples of TO_NUMBER function which convert a character string with numbers to Numbers

Example : add number 5 into a string of characters **'1000'**

```
SQL> select to_number('1000')+5 from dual;

TO_NUMBER('1000')+5
-------------------
               1005
```

Lets take a examples of TO_DATE function which convert a character string to a specified Date format.

As discussed in lab 01 that whenever

SELECT ename, hiredate
FROM emp
WHERE hiredate = TO_DATE('February 22, 1981', 'Month dd, YYYY');

```
  1  SELECT ename, hiredate
  2  FROM emp
  3* WHERE hiredate = TO_DATE('February 22, 1981', 'Month dd, YYYY')
SQL> /

ENAME      HIREDATE
---------- ---------
WARD       22-FEB-81
```

Same date format as Saved In emp table

function simply

Single Row Functions

Decode Functions

Decode functions works same as Case statements or IF then Else statement in a normal programming language

Syntax: ('Column name',Search1,Result1,search2,Result2,…..
……..Search n , result n, Default)

Example 1 : To print job, salary and revised salary depending on the job.It has been announced that all Analyst got 10% increment and all clerks got 15 percent increment and all managers got 20% increment while rest staff will have same sallary. Please print empno & name also as all same designations don't have same salary due to different experiences

```
SELECT empno,ename,job, sal, DECODE (job,
'ANALYST', SAL*1.1, 'CLERK', SAL*1.15,
'MANAGER', SAL*1.20, SAL)
REVISED_SALARY
FROM emp
```

EMPNO	ENAME	JOB	SAL	REVISED_SALARY
7369	SMITH	CLERK	800	920
7499	ALLEN	SALESMAN	1600	1600
7521	WARD	SALESMAN	1250	1250
7566	JONES	MANAGER	2975	3570
7654	MARTIN	SALESMAN	1250	1250
7698	BLAKE	MANAGER	2850	3420
7782	CLARK	MANAGER	2450	2940
7788	SCOTT	ANALYST	3000	3300
7839	KING	PRESIDENT	5000	5000
7844	TURNER	SALESMAN	1500	1500
7876	ADAMS	CLERK	1100	1265
7900	JAMES	CLERK	950	1092.5
7902	FORD	ANALYST	3000	3300
7934	MILLER	CLERK	1300	1495

Single Row Functions

Above query can be best explained in IF THEN ELSE terms as

IF job = ‘ANALYST’ THEN sal = sal * 1.1
IF job = ‘CLERK’ THEN sal = sal * 1.15
IF job = ‘MANAGER’ THEN sal = sal * 1.20
ELSE sal = sal

```
SELECT empno,ename,job, sal,
CASE WHEN job = 'ANALYST' THEN sal * 1.1
        WHEN job = 'CLERK' THEN sal * 1.15
        WHEN job = 'MANAGER' THEN sal * 1.20
        ELSE sal
END AS "REVISED SALARY" FROM emp
```

EMPNO	ENAME	JOB	SAL	REVISED SALARY
7369	SMITH	CLERK	800	920
7499	ALLEN	SALESMAN	1600	1600
7521	WARD	SALESMAN	1250	1250
7566	JONES	MANAGER	2975	3570
7654	MARTIN	SALESMAN	1250	1250
7698	BLAKE	MANAGER	2850	3420
7782	CLARK	MANAGER	2450	2940
7788	SCOTT	ANALYST	3000	3300
7839	KING	PRESIDENT	5000	5000
7844	TURNER	SALESMAN	1500	1500
7876	ADAMS	CLERK	1100	1265
7900	JAMES	CLERK	950	1092.5
7902	FORD	ANALYST	3000	3300
7934	MILLER	CLERK	1300	1495

Lets see how tax is applicable on a job person. Divide salary from 1000 and if salary is less than 1 than there is no tax other wise for range of **1**’s *tax is* **9%** *for* **2** *tax is* **20 %** *for* **3** *tax is* **30%** *for* **4** *tax is* **40%** *for* **5** *tax is* **42%** *for* **6** *tax is* **44 %** *and for greater than* **6** *it would be* **45 %**.

Single Row Functions

Example 2 : Display the applicable tax rate for each employee in dept 30

```
SELECT ename, sal,
DECODE(TRUNC(sal/1000, 0),
0, 0.00,
1, 0.09,
2, 0.20,
3, 0.30,
4, 0.40,
5, 0.42,
6, 0.44,
0.45) TAX_RATE
FROM emp
WHERE deptno = 30;
```

We have seen in the same lecture that Trunc with zero will remove after decimals numbers.

```
  1  SELECT ename, sal, DECODE(TRUNC(sal/1000, 0),
  2  0, 0.00,
  3  1, 0.09,
  4  2, 0.20,
  5  3, 0.30,
  6  4, 0.40,
  7  5, 0.42,
  8  6, 0.44,
  9  0.45) TAX_RATE
 10  FROM emp
 11* WHERE deptno = 30
SQL> /

ENAME             SAL   TAX_RATE
---------- ---------- ----------
ALLEN            1600        .09
WARD             1250        .09
MARTIN           1250        .09
BLAKE            2850         .2
TURNER           1500        .09
JAMES             950          0
```

Single Row Functions

Nesting Functions

Functions with in the function is called nesting function Single-row functions can be nested to any level.

Example : Display head of company(person with no manager)

```
SELECT ENAME, NVL(TO_CHAR(MGR), 'No Manager')
FROM EMP
WHERE MGR IS NULL;
```

```
  1  SELECT ENAME, NVL(TO_CHAR(MGR), 'No Manager')
  2  FROM EMP
  3* WHERE MGR IS NULL
SQL> /

ENAME      NVL(TO_CHAR(MGR),'NOMANAGER')
---------- -------------------------------------
KING       No Manager
```

We are learning one more thing from this query that is NVL function have both arguments of same data type like NVL(comm,0) so both are numeric similarly we cant write NVL(MGR,'No Manager') because MGR is number.

2

Multiple_Row Functions

These Functions work on group of rows and then give result like aggregate functions(count, Avg, sum)

Syntax

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

One of the basic identity of

Multiple Row Functions

Lets take few examples of Applying multiple row functions to all rows in a table (means no where clause)

Example 1 : To show the **average salary**, **minimum salary**, **maximum salary** and **count of employees in the organization**

ALL

```
SELECT AVG (SAL), MIN(SAL), MAX(SAL), COUNT(*)
FROM EMP;
```

```
  1  SELECT AVG (SAL), MIN(SAL), MAX(SAL), COUNT(*)
  2* FROM EMP
SQL> /

  AVG(SAL)   MIN(SAL)   MAX(SAL)   COUNT(*)
---------- ---------- ---------- ----------
2073.21429        800       5000         14
```

Example 2 : Total number of rows in a table

```
SELECT  COUNT(*)
FROM EMP;
```

Example 3 : Total number of rows in a table that has dept=30

```
SELECT  COUNT(*) FROM EMP
Where deptno = 30;
```

Multiple Row Functions

Note: group functions do not include null values in count sum or average.

Example 4 : Display the count of nonnull commission rows or nonnull rows.

```
SELECT COUNT(comm)
FROM EMP;
```

```
COUNT(COMM)
-----------
          4
```

Showing that there are 4 rows in a employee table that are not null

Example 5 : Find the average of commission values of all employees

~~SELECT AVG(Comm)~~
~~FROM EMP;~~

```
SELECT AVG(NVL(Comm,0))
FROM EMP;
```

```
  1  SELECT  AVG(Comm)
  2* FROM EMP
SQL> /

 AVG(COMM)
----------
       550
AVG(NVL(COMM,0))
----------------
      157.142857
```

Multiple Row Functions

Lets take few example of Multiple row functions to group of Rows

Example 1 : Display the ***Department Wise*** Average salary of all employees

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

```
  1  SELECT deptno, AVG(sal) AVG_SAL
  2  FROM emp
  3* GROUP BY deptno
SQL> /

    DEPTNO    AVG_SAL
---------- ----------
        30 1566.66667
        20       2175
        10 2916.66667
```

NOTE*:** if you are applying a group function on a column or columns then whatever columns you will be displaying in select command other than those must be a part of Group by clause. Like we did ***deptno above.

Multiple Row Functions

Example 2 : Display the ***Job Wise*** total salary for each department

```
SELECT deptno, job, sum(sal)
FROM emp
GROUP BY deptno, job;
```

```
  1  SELECT deptno, job, sum(sal)
  2  FROM emp
  3* GROUP BY deptno,job
SQL> /

    DEPTNO JOB         SUM(SAL)
---------- --------- ----------
        20 CLERK           1900
        30 SALESMAN        5600
        20 MANAGER         2975
        30 CLERK            950
        10 PRESIDENT       5000
        30 MANAGER         2850
        10 CLERK           1300
        10 MANAGER         2450
        20 ANALYST         6000

9 rows selected.
```

Excluding Group Result Or Restrict number of group rows

```
SELECT column, group_function
FROM table
[WHERE condition]
[GROUP BY
group_by_expression]
[HAVING group_condition]
[ORDER BY column];
```

1 2 3 — SEQUENCE

In the same way that we use the WHERE clause to restrict the rows that we select, the HAVING clause is used to restrict groups.

You can use having clause directly after where clause in syntax

Multiple Row Functions

Example 1 : Display the ***Department Wise*** Average and maximum salary in the descending order of average salary for all departments having average salary higher than 2000.

```
SELECT DEPTNO, AVG(SAL), MAX(SAL)
FROM EMP
GROUP BY DEPTNO
HAVING AVG(SAL) > 2000
ORDER BY AVG(SAL) desc
```

```
  1  SELECT DEPTNO, AVG(SAL), MAX(SAL)
  2  FROM EMP
  3  GROUP BY DEPTNO
  4  HAVING AVG(SAL) > 2000
  5* ORDER BY AVG(SAL) desc
  6  /

    DEPTNO   AVG(SAL)   MAX(SAL)
---------- ---------- ----------
        10 2916.66667       5000
        20       2175       3000
```

Example 2 : Display the ***Department Wise*** Average commission and minimum salary in the ascending order of salary for all departments having salary higher than 500 but Average salary lower than 4000

Multiple Row Functions

```
SELECT DEPTNO, AVG(NVL(comm,0)), MIN(SAL)
FROM EMP
WHERE SAL > 500
GROUP BY DEPTNO
HAVING AVG(SAL) < 4000
ORDER BY AVG(SAL)
```

```
  1  SELECT DEPTNO, AVG(NVL(comm,0)), MIN(SAL)
  2  FROM EMP
  3  WHERE SAL > 500
  4  GROUP BY DEPTNO
  5  HAVING AVG(SAL) < 4000
  6* ORDER BY AVG(SAL)
SQL> /

    DEPTNO AVG(NVL(COMM,0))   MIN(SAL)
---------- --------------- ----------
        30       366.666667        950
        20                0        800
        10                0       1300
```

Example 2 : Display the job title (*except Sales department)* and total monthly salary consumed by each job title from a company(payroll) for those Jobs having payroll exceeding 5000.Give this data in ascending order of payroll.

```
SELECT JOB, SUM(SAL) PAYROLL
FROM EMP
WHERE JOB NOT LIKE 'SALES%'
GROUP BY JOB
HAVING SUM(SAL) > 5000
ORDER BY SUM(SAL);
```

```
  1  SELECT JOB, SUM(SAL) PAYROLL
  2  FROM EMP
  3  WHERE JOB NOT LIKE 'SALES%'
  4  GROUP BY JOB
  5  HAVING SUM(SAL) > 5000
  6* ORDER BY SUM(SAL)
  7  /

JOB          PAYROLL
--------- ----------
ANALYST         6000
MANAGER         8275
```

Multiple Row Functions

Nesting Group Functions

Display Group functions with in the functions for example to display the maximum average salary by nesting group functions

```
SELECT max(avg(sal))
FROM emp
GROUP BY deptno;
```

```
  1  SELECT max(avg(sal))
  2  FROM emp
  3* GROUP BY deptno
SQL> /

MAX(AVG(SAL))
-------------
   2916.66667
```