

19/8/25

## EXERCISE-6

### Single Row Functions

#### Objective

After the completion of will be able to do the

- Describe various in SQL
- Use character, in SELECT statement.
- Describe the use

Evaluation Procedure	Marks awarded
Practice Evaluation (5)	
Viva(5)	
Total (10)	
Faculty Signature	

this exercise, the students following types of functions available number and date functions of conversion functions.

#### Single row functions:

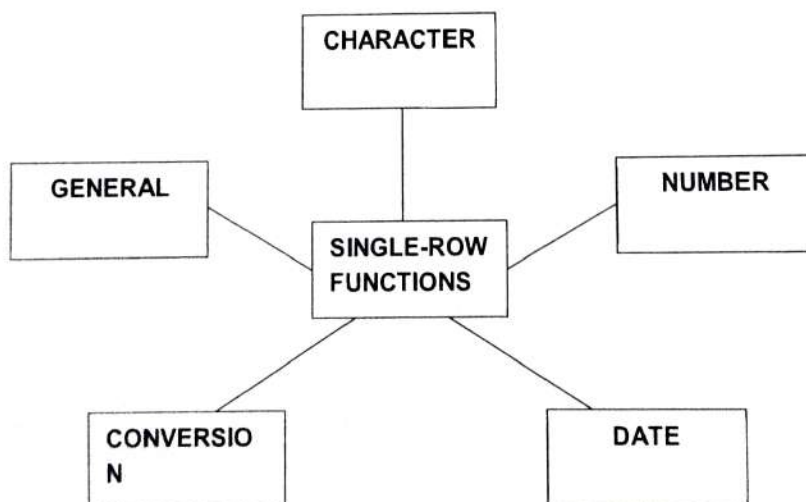
Manipulate data items.  
Accept arguments and return one value.  
Act on each row returned.  
Return one result per row.  
May modify the data type.  
Can be nested.  
Accept arguments which can be a column or an expression

#### Syntax

Function\_name(arg1,...argn)

An argument can be one of the following

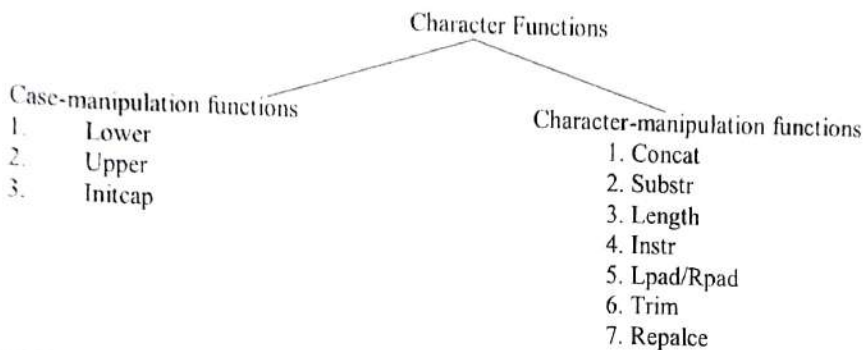
- ✓ User-supplied constant
- ✓ Variable value
- ✓ Column name
- ✓ Expression



- Character Functions: Accept character input and can return both character and number values.

- Number functions: Accept numeric input and return numeric values.
- Date Functions: Operate on values of the DATE data type.
- Conversion Functions: Convert a value from one type to another.

## Character Functions



Function	Purpose
lower(column/expr)	Converts alpha character values to lowercase
upper(column/expr)	Converts alpha character values to uppercase
initcap(column/expr)	Converts alpha character values the to uppercase for the first letter of each word, all other letters in lowercase
concat(column1/expr1, column2/expr2)	Concatenates the first character to the second character
substr(column/expr,m,n)	Returns specified characters from character value starting at character position m, n characters long
length(column/expr)	Returns the number of characters in the expression
instr(column/expr,'string',m,n)	Returns the numeric position of a named string
lpad(column/expr, n,'string')	Pads the character value right-justified to a total width of n character positions
rpadd(column/expr,'string',m,n)	Pads the character value left-justified to a total width of n character positions
trim(leading/trailing/both, trim_character FROM trim_source)	Enables you to trim heading or string. trailing or both from a character
replace(text, search_string, replacement_string)	

### Example:

```
lower('SQL Course')--sql course
upper('SQL Course')--SQL COURSE
initcap('SQL Course')--Sql Course
```

```
SELECT 'The job id for' || upper(last_name || 'is') || lower(job_id) AS "EMPLOYEE DETAILS"
FROM employees;
```

```
SELECT employee_id, last_name, department_id
FROM employees
WHERE LOWER(last_name)='higgins';
```

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

Command	Query	Output
initcap(char);	<i>select initcap('hello') from dual;</i>	Hello
lower(char);	<i>select lower('HELLO') from dual;</i>	Hello
upper(char);	<i>select upper('hello') from dual;</i>	HELLO
ltrim(char,[set]);	<i>select ltrim('cseit', 'cse') from dual;</i>	IT
rtrim(char,[set]);	<i>select rtrim('cseit', 'it') from dual;</i>	CSE
replace(char,search string, replace string);	<i>select replace('jack and jue', 'j', 'bl') from dual;</i>	black and blue
substr(char,m,n);	<i>select substr('information', 3, 4) from dual;</i>	form

### Example:

SELECT employee\_id, CONCAT (first\_name,last\_name) NAME , job\_id,LENGTH(last\_name),  
INSTR(last\_name,'a') "contains'a?"  
FROM employees WHERE SUBSTR(job\_id,4)='ERP';

### NUMBER FUNCTIONS

Function	Purpose
round(column/expr, n)	Rounds the value to specified decimal
trunc(column/expr,n)	Truncates value to specified decimal
mod(m,n)	Returns remainder of division

### Example

Function	Result
round(45.926,2)	45.93
trunc(45.926,2)	45.92
mod(1600,300)	100

SELECT ROUND(45.923,2), ROUND(45.923,0), ROUND(45.923,-1) FROM dual;

**NOTE:** Dual is a dummy table you can use to view results from functions and calculations.

SELECT TRUNC(45.923,2), TRUNC(45.923), TRUNC(45.923,-2) FROM dual;

SELECT last\_name,salary,MOD(salary,5000) FROM employees WHERE job\_id='sa\_rep';

### Working with Dates

The Oracle database stores dates in an internal numeric format: century, year, month, day, hours, minutes, and seconds.

- The default date display format is DD-MON-RR.
- Enables you to store 21st-century dates in the 20th century by specifying only the last two digits of the year
- Enables you to store 20th-century dates in the 21st century in the same way

### Example



SELECT last\_name, hire\_date FROM employees WHERE hire\_date < '01-FEB-88';

### Working with Dates

**SYSDATE** is a function that returns:

- Date
- Time

#### Example

**Display the current date using the DUAL table.**

SELECT SYSDATE FROM DUAL;

### 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

Because 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.

You can perform the following operations:

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

#### Example

SELECT last\_name, (SYSDATE-hire\_date)/7 AS WEEKS  
FROM employees  
WHERE department\_id = 90;

### Date Functions

Function	Result
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

### Date Functions

Date functions operate on Oracle dates. All date functions return a value of DATE data type except MONTHS\_BETWEEN, which returns a numeric value.

- **MONTHS\_BETWEEN(date1, date2)::** Finds the number of months between date1 and date2. The result can be positive or negative. If date1 is later than date2, the result is positive; if date1 is earlier than date2, the result is negative. The noninteger part of the result represents a portion of the month.

- **ADD\_MONTHS(date, n)**:: Adds n number of calendar months to date. The value of n must be an integer and can be negative.

- **NEXT\_DAY(date, 'char')**:: Finds the date of the next specified day of the week ('char') following date. The value of char may be a number representing a day or a character string.

- **LAST\_DAY(date)**:: Finds the date of the last day of the month that contains date

- **ROUND(date, 'fmt')**:: Returns date rounded to the unit that is specified by the format model fmt. If the format model fmt is omitted, date is rounded to the nearest day.

- **TRUNC(date, 'fmt')**:: Returns date with the time portion of the day truncated to the unit that is specified by the format model fmt. If the format model fmt is omitted, date is truncated to the nearest day.

### Using Date Functions

Function	Result
<b>MONTHS_BETWEEN</b> ( '01-SEP-95' , '11-JAN-94' )	19.6774194
<b>ADD_MONTHS</b> ( '11-JAN-94' , 6)	'11-JUL-94'
<b>NEXT_DAY</b> ( '01-SEP-95' , 'FRIDAY' )	'08-SEP-95'
<b>LAST_DAY</b> ( '01-FEB-95' )	'28-FEB-95'

### Example

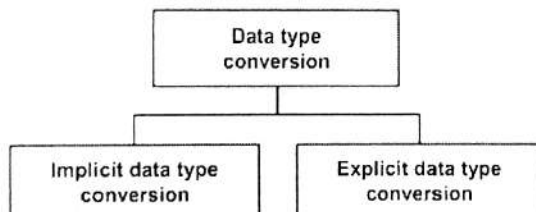
Display the employee number, hire date, number of months employed, sixmonth review date, first Friday after hire date, and last day of the hire month for all employees who have been employed for fewer than 70 months.

```
SELECT employee_id, hire_date, MONTHS_BETWEEN (SYSDATE, hire_date)
TENURE, ADD_MONTHS (hire_date, 6) REVIEW, NEXT_DAY (hire_date, 'FRIDAY'),
LAST_DAY(hire_date)
FROM employees
WHERE MONTHS_BETWEEN (SYSDATE, hire_date) < 70;
```

### Conversion Functions

This covers the following topics:

- Writing a query that displays the current date
- Creating queries that require the use of numeric, character, and date functions
- Performing calculations of years and months of service for an employee



### Implicit Data Type Conversion

For assignments, the Oracle server can automatically convert the following:

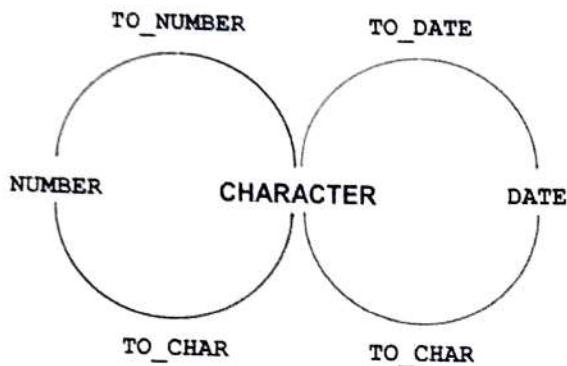
From	To
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

For example, the expression `hire_date > '01-JAN-90'` results in the implicit conversion from the string '01-JAN-90' to a date.

For expression evaluation, the Oracle Server can automatically convert the following:

From	To
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE

### Explicit Data Type Conversion



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

#### Example:

**Using the TO\_CHAR Function with Dates**

`TO_CHAR(date, 'format_model')`

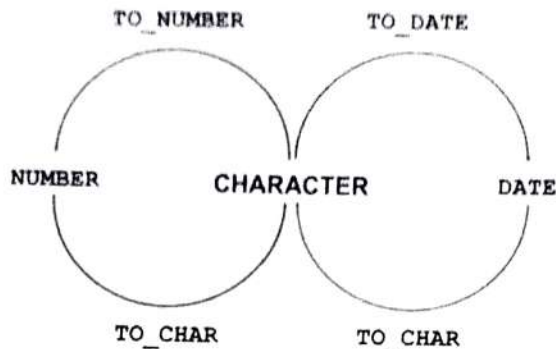
**The format model:**

- Must be enclosed by single quotation marks
- Is case-sensitive
- Can include any valid date format element
- Has an fm element to remove padded blanks or suppress leading zeros
- Is separated from the date value by a comma

```
SELECT employee_id, TO_CHAR(hire_date, 'MM/YY') Month_Hired
FROM employees WHERE last_name = 'Higgins';
```



### Elements of the Date Format Model



### Sample Format Elements of Valid Date

Element	Description
SCC or CC	Century; server prefixes B.C. date with -
Years in dates YYYY or SYYYY	Year; server prefixes B.C. date with -
YYY or YY or Y	Last three, two, or one digits of year
Y,YYY	Year with comma in this position
IYYY, IYY, IY, I	Four-, three-, two-, or one-digit year based on the ISO standard
SYEAR or YEAR	Year spelled out; server prefixes B.C. date with -
BC or AD	Indicates B.C. or A.D. year
B C. or A.D.	Indicates B.C. or A.D. year using periods
Q	Quarter of year
MM	Month: two-digit value
MONTH	Name of month padded with blanks to length of nine characters
MON	Name of month, three-letter abbreviation
RM	Roman numeral month
WW or W	Week of year or month
DDD or DD or D	Day of year, month, or week
DAY	Name of day padded with blanks to a length of nine characters
DY	Name of day; three-letter abbreviation
J	Julian day; the number of days since December 31, 4713 B.C.

### Date Format Elements: Time Formats

Use the formats that are listed in the following tables to display time information and literals and to change numerals to spelled numbers.

Element	Description
AM or PM	Meridian indicator
A.M. or P.M.	Meridian indicator with periods
HH or HH12 or HH24	Hour of day, or hour (1-12), or hour (0-23)
MI	Minute (0-59)
SS	Second (0-59)
SSSSS	Seconds past midnight (0-86399)

#### Other Formats

Element	Description
...	Punctuation is reproduced in the result
"of the"	Quoted string is reproduced in the result

#### Specifying Suffixes to Influence Number Display

Element	Description
TH	Ordinal number (for example, DDTH for 4TH)
SP	Spelled-out number (for example, DDSP for FOUR)
SPTH or THSP	Spelled-out ordinal numbers (for example, DDSPTH for FOURTH)

#### Example

```
SELECT last_name,  
       TO_CHAR(hire_date, 'fmDD Month YYYY') AS HIREDATE  
FROM   employees;
```

Modify example to display the dates in a format that appears as "Seventeenth of June 1987 12:00:00 AM."

```
SELECT last_name,  
       TO_CHAR(hire_date, 'fmDdspth "of" Month YYYY fmHH:MI:SS AM') HIREDATE  
FROM   employees;
```

#### Using the TO\_CHAR Function with Numbers

TO\_CHAR(number, 'format\_model')

These are some of the format elements that you can use with the TO\_CHAR function to display a number value as a character:

Element	Result
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 comma as thousands indicator

#### Number Format Elements

If you are converting a number to the character data type, you can use the following format elements:



Element	Description	Example	Result
9	Numeric position (number of 9s determines display width)	99999	1234
0	Display leading zeros	099999	001234
\$	Floating dollar sign	\$99999	\$1234
£	Floating local currency symbol	£99999	£1234
D	Returns in the specified position the decimal character. The default is a period (.)	99199	99.99
	Decimal point in position specified	99999.99	1234.00
G	Returns the group separator in the specified position. You can specify multiple group separators in a number format model	9,999	9,000
	Comma in position specified	999,999	1,234
MI	Minus signs to right (negative values)	99999MI	1234-
PR	Parenthesize negative numbers	99999PR	<1234>
EEEE	Scientific notation (format must specify four Es)	99,999EEEE	1.234E+05
U	Returns in the specified position the "Euro" (or other) dual currency	U9999	€1234
V	Multiply by 10 <sup>n</sup> times (n = number of 9s after V)	9999V99	123400
S	Returns the negative or positive value	S9999	-1234 or +1234
B	Display zero values as blank, not 0	B9999.99	1234.00

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY
FROM employees
WHERE last_name = 'Ernst';
```

### 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'])
- These functions have an fx modifier. This modifier specifies the exact matching for the character argument and date format model of a TO\_DATE function.

The fx modifier specifies exact matching for the character argument and date format model of a TO\_DATE function:

- Punctuation and quoted text in the character argument must exactly match (except for case) the corresponding parts of the format model.
- The character argument cannot have extra blanks. Without fx, Oracle ignores extra blanks.
- Numeric data in the character argument must have the same number of digits as the corresponding element in the format model. Without fx, numbers in the character argument can omit leading zeros.

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date = TO_DATE('May 24, 1999', 'fxMonth DD, YYYY');
```

Find the Solution for the following:

1. Write a query to display the current date. Label the column Date.

```
SELECT SYSDATE AS "Date" FROM "Dual";
```

2. The HR department needs a report to display the employee number, last name, salary, and increased by 15.5% (expressed as a whole number) for each employee. Label the column New Salary.

```
SELECT employee_id, last_name, salary, ROUND  
(salary * 1.1555) AS "New Salary" FROM employees;
```

3. Modify your query lab\_03\_02.sql to add a column that subtracts the old salary from the new salary. Label the column Increase.

```
SELECT employee_id, last_name, salary, ROUND  
(salary * 1.1555) AS "New Salary", ROUND((salary * 1.1555)  
- salary) AS "Increase" FROM employees;
```

4. Write a query that displays the last name (with the first letter uppercase and all other letters lowercase) and the length of the last name for all employees whose name starts with the letters J, A, or M. Give each column an appropriate label. Sort the results by the employees' last names.

```
SELECT INITCAP (last_name) AS "Last Name", LENGTH  
(last_name) AS "Length" FROM employees WHERE last_name  
LIKE 'J%' OR last_name LIKE 'A%' OR last_name  
LIKE 'M%' ORDER BY last_name;
```

5. Rewrite the query so that the user is prompted to enter a letter that starts the last name. For example, if the user enters H when prompted for a letter, then the output should show all employees whose last name starts with the letter H.

```
SELECT last_name, FROM employees WHERE last_name  
LIKE '&start_letter%';
```

6. The HR department wants to find the length of employment for each employee. For each employee, display the last name and calculate the number of months between today and the date on which the employee was hired. Label the column MONTHS\_WORKED. Order your results by the number of months employed. Round the number of months up to the closest whole number.

```
SELECT last_name, CEIL (MONTHS-BETWEEN(SYSDATE  
- hire_date)) AS "MONTHS-WORKED" FROM employees  
ORDER BY MONTHS-BETWEEN (SYSDATE, hire_date) DESC;
```

Note: Your results will differ.

7. Create a report that produces the following for each employee:

<employee last name> earns <salary> monthly but wants <3 times salary> Label the column Dream Salaries

```
SELECT last_name || ' earns ' || salary || ' monthly but  
wants ' || (salary * 3) AS "Dream Salaries" FROM employees;
```

8. Create a query to display the last name and salary for all employees. Format the salary to be 15 characters long, left-padded with the \$ symbol. Label the column SALARY.

```
SELECT last_name || PADTO_CHAR(salary, 15, '$') AS  
"SALARY" FROM employees;
```

9. Display each employee's last name, hire date, and salary review date, which is the first Monday after six months of service. Label the column REVIEW. Format the dates to appear in the format similar to "Monday, the Thirty-First of July, 2000."

```
SELECT last_name, hire_date TO_CHAR(NEXT_DAY(ADD -  
MONTHS(hire_date, 6) MONDAY), 'Day, "the" DDspth "of"  
Month' YYYY) AS "REVIEW" FROM employees;
```

10. Display the last name, hire date, and day of the week on which the employee started. Label the column DAY. Order the results by the day of the week, starting with Monday.

```
SELECT last_name || hire_date TO_CHAR(hire_date, 'Day')  
AS "DAY" FROM employees ORDER BY TO_CHAR  
(hire_date, 'D');
```

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	5
Total (15)	15
Faculty Signature	T B P



Practice Questions  
Introduction to Functions

1. For each task, choose whether a single-row or multiple row function would be most appropriate:
- a. Showing all of the email addresses in upper case letters Single-row (UPPER)
  - b. Determining the average salary for the employees in the sales department Multiple-row (AVG)
  - c. Showing hire dates with the month spelled out (September 1, 2004) Single-row (TO\_CHAR)
  - d. Finding out the employees in each department that had the most seniority (the earliest hire date) Multiple-row
  - e. Displaying the employees' salaries rounded to the hundreds place Single-row (ROUND) MIN (GROUN)
  - f. Substituting zeros for null values when displaying employee commissions. Single-row (NVL)

2. The most common multiple-row functions are: AVG, COUNT, MAX, MIN, and SUM. Give your own definition for each of these functions.

AVG → Returns the average value of a numeric column

COUNT → Return the number of rows or non-null values if an expression is given.

MAX → return the highest value in a column

MIN → return the lowest value in a column

SUM → Return the total of numeric values in a column.

3. Test your definitions by substituting each of the multiple-row functions in this query.

```
SELECT FUNCTION(salary)
```

```
FROM employees
```

Write out each query and its results.

```
SELECT AVG (salary) AS average - Salary FROM employees;
```

```
SELECT COUNT AS total - employees FROM employees;
```

```
SELECT MAX (salary) AS Max Salary FROM employees;
```

```
SELECT MIN (salary) AS Min - Salary FROM employees;
```

```
SELECT SUM (salary) AS sum - salary FROM employees;
```

## Case and Character Manipulation

1. Using the three separate words "Oracle," "Internet," and "Academy," use one command to produce the following output:  
The Best Class Oracle Internet Academy

```
SELECT the Best Class '||' Oracle '||' '||' Internet '||' '||'  
'Academy' AS output FROM dual;
```

2. Use the string "Oracle Internet Academy" to produce the following output:  
The Net net

```
SELECT 'The' || SUBSTR('Oracle Internet Academy', 8, 3) ||  
' ' || LOWER(SUBSTR('Oracle Internet Academy', 8, 3)) AS  
output FROM dual;
```

3. What is the length of the string "Oracle Internet Academy"?

```
SELECT LENGTH('Oracle Internet Academy') AS length  
FROM dual;
```

4. What's the position of "I" in "Oracle Internet Academy"?

```
SELECT INSTR('Oracle Internet Academy', 'I') AS  
Position FROM dual;
```

5. Starting with the string "Oracle Internet Academy", pad the string to create  
\*\*\*\*Oracle\*\*\*\*Internet\*\*\*\*Academy\*\*\*\*

```
SELECT '****' || REPLACE('Oracle Internet Academy',  
' ' , '****') || '****' AS padded FROM dual;
```

## Number Functions

1. Display Oracle database employee last\_name and salary for employee ids between 100 and 102. Include a third column that divides each salary by 1.55 and rounds the result to two decimal places.

```
-SELECT last_name, salary, ROUND(salary / 1.55, 2) AS  
divided FROM employee WHERE employee_id BETWEEN  
100 AND 102;
```

2. Display employee last\_name and salary for those employees who work in department 80. Give each of them a raise of 5.333% and truncate the result to two decimal places.

```
SELECT last_name, salary, TRUNC(salary * 1.05333, 2)  
AS new_salary FROM employee WHERE department_id = 80;
```

3. Use a MOD number function to determine whether 38873 is an even number or an odd number.

```
SELECT MOD(38873, 2) AS remainder FROM dual;
```

4. Use the DUAL table to process the following numbers:

845.553 - round to one decimal place

30695.348 - round to two decimal places

30695.348 - round to -2 decimal places

2.3454 - truncate the 454 from the decimal place

```
SELECT ROUND(845.553) AS r1,
```

```
ROUND(30695.348, 2) AS r2,
```

```
ROUND(30695.348, -2) AS r3,
```

```
TRUNC(2.3454, 1) AS t1 FROM  
dual;
```

5. Divide each employee's salary by 3. Display only those employees' last names and salaries who earn a salary that is a multiple of 3.

```
SELECT last_name, salary FROM employee WHERE  
MOD(salary, 3) = 0;
```

6. Divide 34 by 8. Show only the remainder of the division. Name the output as EXAMPLE.

```
SELECT MOD(34, 8) AS EXAMPLE FROM dual;
```

7. How would you like your paycheck - rounded or truncated? What if your paycheck was calculated to be \$565.784 for the week, but you noticed that it was issued for \$565.78. The loss of .004 cent would probably make very little difference to you. However, what if this was done to a thousand people, a 100,000 people, or a million people! Would it make a difference then? How much difference?

Rounded  $\rightarrow$   $\text{ROUND}(565.784, 2) \rightarrow 565.78$

Truncated  $\rightarrow$   $\text{TRUNC}(565.784, 2) \rightarrow 565.78$