National University of Computer & Emerging Sciences, Karachi
Computer Science Department
Fall 2024, Lab Manual – 02

Course Code: CL-2005	**Course: Database Systems Lab**
Instructor(s):	**Mehak Mazhar**

Contents:

Simple Select Query:
The purpose of a SELECT statement is to display and retrieve data from one or more database tables. SELECT is the most frequently used command in SQL, and is used to query the database tables. Usually a simple SELECT query involves two more clauses i.e. FROM and WHERE. FROM is used to refer to the tables to retrieve data from.

Syntax: **SELECT <column1>, <column2>…..<column n> FROM <table1>;**

Example: **Display all columns of HR Database's Employee table.**

SELECT employee_id, first_name, last_name, email, phone_number, hire_date, job_id, salary, commission_pct, manager_id, department_id FROM employees;

Task: **Display any two columns from employees table.**

Use of Column Alias:
A Column Alias is used to give column a name that is different than the name it is given in the Table. A user usually provides a Column Alias through a special keyword "AS" to display the column with a changed name i.e. Renamed Column.

Syntax: **SELECT <column1> as "New Column Name" FROM <table1>;**

Example: **Display the column Employee_ID and Phone_Number from Employees table. Display the column Phone Number as Contact Number.**

SELECT employee_id, phone_number as "Contact Number" From employees**;**

Task: **Display Hire_date from employees table, name it as Joining Date.**

Use of Concatenation Operator:
A concatenation operator is denoted by a pipe || which is used to concatenate columns or strings together. The pipe operator is independent of the data type of the column.

Syntax: **SELECT <column1> || <column2> FROM <table1>.**

Example: **Show the First Name and Salary of employees as a single column named "Employees and Salaries".**

SELECT first_name || salary as "Employees and Salaries" FROM employees;

Task**:Display the first_name, last_name of Employees together in one column named "FULL NAME"**

Use of DISTINCT Keyword.

The distinct keyword is used to show unique records of a table. The SELECT does not eliminate duplicate when it projects over one or more column. To eliminate the duplicates, we use DISTINCT keyword**.**

Syntax**: SELECT DISTINCT <column1> FROM <table1>.**

Example**: Show unique departments of Employees Table.**

SELECT DISTINCT(department_id) FROM employees;

Use of ALL Keyword:

A query with keyword ALL display all rows irrespective of the duplicate records found in the table. In general it is the reverse of what DISTINCT keyword does.

Syntax**: SELECT ALL <column1> FROM <table1>.**

Example**: Show all salaries of Employees.**

SELECT ALL(salary) FROM employees;

Use of Asterisk Keyword:

Many SQL retrieval requires all columns to be displayed in the output, While doing so would be hectic, we use Universal (*) operator to do the same. It is used to show all columns of a table at once.

Syntax**: SELECT * FROM <table1>.**

Example**: Show all columns of table DEPARTMENTS.**

SELECT * FROM departments;

Row Selection Using WHERE Clause:

We often need to restrict the number of rows retrieved from the table. This can be done using WHERE clause. The clause uses a search condition or set of search conditions to filter the rows.

Syntax**: SELECT <column1>...<column(n)> FROM <table1> WHERE <column1> =________;**

Example**: Show the first_name, salary of Employee whose employee id is 100.**

SELECT first_name, salary FROM employees WHERE employee_id = 100;

There are Five Possible Types of Search Condition and Operators to be Used:

(a) Comparison Search Condition: The comparison search condition involves comparison between the column's actual value and desired value and returns the results filtered using comparison operators (<, >, <=, >= , <> , !=, =). These conditions may involve the use of Logical Operators (AND, OR, NOT) with parameters if needed to show order of evaluation. A search that involves comparison and logical operators together is called Compound Comparison Search Condition.
Example: **List all employees having monthly salaries greater than 20,000 and deptno: 100.**
SELECT * FROM employees WHERE salary > 20,000 and department_id =100;

(b) Range Search Condition: The range search uses BETWEEN and NOT BETWEEN operators to filter the rows on the basis of range of elements. The Between operator includes the endpoints too for search output**.**

Example: **List the staff with the salary between 20,000 and 30,000.**

SELECT * FROM employees WHERE salary BETWEEN 20,000 and 30,000;

(c) **Set Membership Search Condition:** The set membership test (IN) tests whether a data value matches one of a list of values.
Example: **List the salaries of Sales Manager and Purchasing Manager**.

SELECT * FROM job WHERE job_title IN (‘Sales Manager’, ‘Purchasing Manager’);

(d) **Pattern Match Search Condition:** The search condition involves searching for a particular character or string within a column value. Like Operator with the help of pattern matching symbols (_, %) are used find patterns in the column’s value. **‘_’ represents a single character while ‘%’ represents a sequence of characters.**

Example: List all the employees whose names contains an ‘a’ in their first names.

SELECT * FROM employees WHERE first_name LIKE ‘%a%’**;**

OR **List all employees having L as second letter in their first names.**

SELECT * FROM employees WHERE first_name LIKE ‘_a %’;

(e) **NULL Search Condition**: The NULL Search Condition uses NULL operator to filter fields that have NULL values.
Example: Display all employees whose commission is not null.

SELECT * FROM employees WHERE commission_pct is not null;

Tasks:
1) List all Employees having annual salary greater 20, 000 and lesser than 30,000.
2) List employee_id and first_name of employees from department # 60 to department #100.
3) List all the Employees having an ‘ll’ in their first_names.
4) List all the employees with no commission.

Sorting Rows with Order by Clause:

Generally, the rows of an SQL query result table are not arranged in a particular order, however with the use of Order By clause, the users can arrange the result in a particular ascending or descending order (alphabetical or numerical) of values present in the fields. The Order By uses column identifiers. Either these are column names or column numbers.

Syntax: **SELECT <column1>..<column(n)> FROM <table1> Order By <column identifier>.**

Example: **Show all employees in order of their increasing salaries**.

SELECT * FROM employees ORDER BY salary **asc**;

Task: **List all employees in order of their decreasing salaries.**

DUAL Table in Oracle:

This is a single row and single column dummy table provided by oracle. This is used to perform mathematical calculations without using a table.

Syntax: SELECT * FROM DUAL

Built In Oracle Functions:

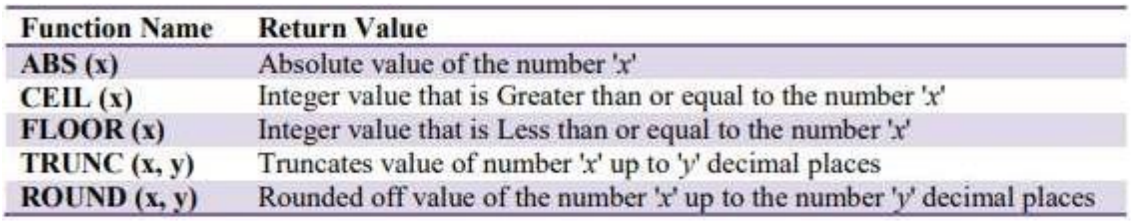
Built In Functions area very powerful feature of SQL capable of: performing calculations, modifying individual data, or output for group of rows, format dates and numbers and conversion of column data types. There are two distinct types of functions:

• **Single-row functions:** Single Row or Scalar Functions return a value for every row that is processed in a query.

• **Aggregate Functions:** The group functions are used to calculate aggregate values like total or average, which return just one total or one average value after processing a group of rows.

There are four types of single row functions. They are:

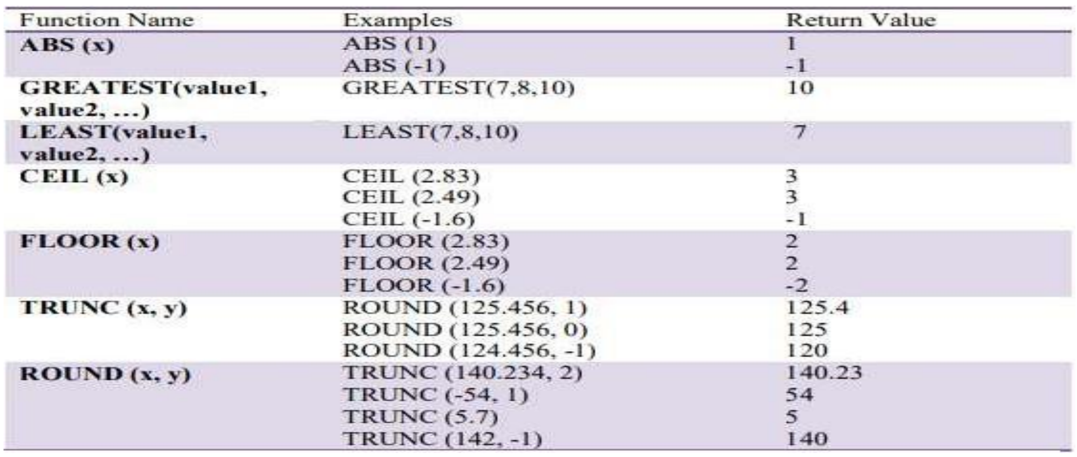
Numeric Functions: These are functions that accept numeric input and return numeric values.



Function Name	Return Value
ABS (x)	Absolute value of the number 'x'
CEIL (x)	Integer value that is Greater than or equal to the number 'x'
FLOOR (x)	Integer value that is Less than or equal to the number 'x'
TRUNC (x, y)	Truncates value of number 'x' up to 'y' decimal places
ROUND (x, y)	Rounded off value of the number 'x' up to the number 'y' decimal places

The Implementation of these Numeric Functions can be understood from following examples:

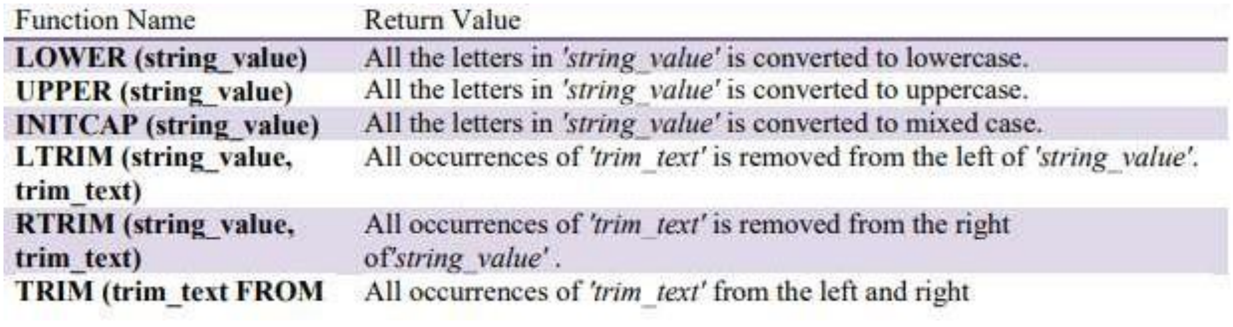
Syntax: SELECT greatest(55,87) from DUAL;



Function Name	Examples	Return Value
ABS (x)	ABS (1) ABS (-1)	1 -1
GREATEST(value1, value2, ...)	GREATEST(7,8,10)	10
LEAST(value1, value2, ...)	LEAST(7,8,10)	7
CEIL (x)	CEIL (2.83) CEIL (2.49) CEIL (-1.6)	3 3 -1
FLOOR (x)	FLOOR (2.83) FLOOR (2.49) FLOOR (-1.6)	2 2 -2
TRUNC (x, y)	ROUND (125.456, 1) ROUND (125.456, 0) ROUND (124.456, -1)	125.4 125 120
ROUND (x, y)	TRUNC (140.234, 2) TRUNC (-54, 1) TRUNC (5.7) TRUNC (142, -1)	140.23 54 5 140

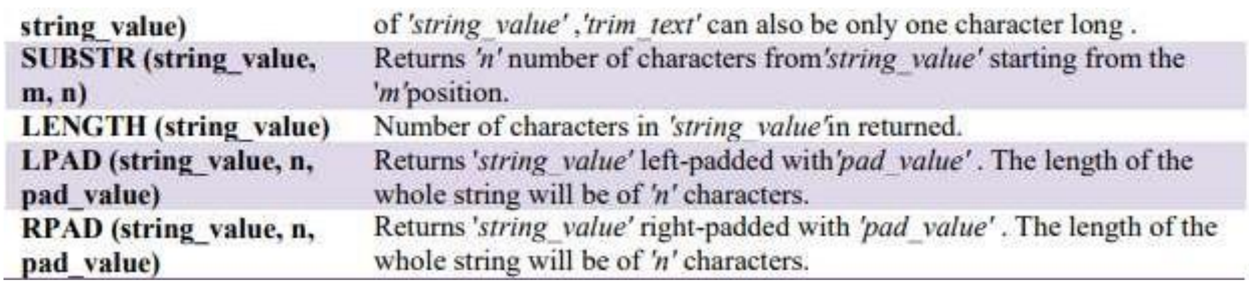
Character or Text Functions: These are functions that accept character input and can return both character and number values. Following are some frequently used char functions:

Syntax: SELECT upper (‘mehak mazhar’) FROM DUAL



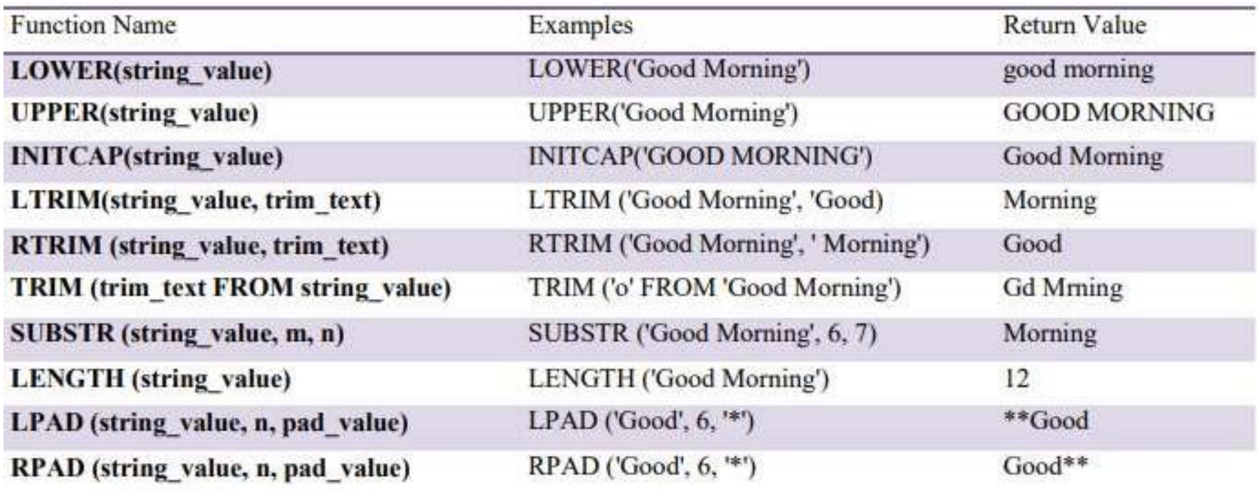
Function Name	Return Value
LOWER (string_value)	All the letters in *'string_value'* is converted to lowercase.
UPPER (string_value)	All the letters in *'string_value'* is converted to uppercase.
INITCAP (string_value)	All the letters in *'string_value'* is converted to mixed case.
LTRIM (string_value, trim_text)	All occurrences of *'trim_text'* is removed from the left of *'string_value'*.
RTRIM (string_value, trim_text)	All occurrences of *'trim_text'* is removed from the right of*'string_value'* .
TRIM (trim_text FROM	All occurrences of *'trim_text'* from the left and right

Syntax: SELECT SUBSTR('Hello World', 1, 3) FROM DUAL;



string_value)	of *'string_value'* ,*'trim_text'* can also be only one character long .
SUBSTR (string_value, m, n)	Returns *'n'* number of characters from*'string_value'* starting from the *'m'*position.
LENGTH (string_value)	Number of characters in *'string_value'*in returned.
LPAD (string_value, n, pad_value)	Returns '*string_value'* left-padded with*'pad_value'* . The length of the whole string will be of *'n'* characters.
RPAD (string_value, n, pad_value)	Returns '*string_value'* right-padded with *'pad_value'* . The length of the whole string will be of *'n'* characters.

Following examples illustrate the usage of these functions.
Syntax: SELECT initcap ('mehak mazhar') FROM DUAL



Function Name	Examples	Return Value
LOWER(string_value)	LOWER('Good Morning')	good morning
UPPER(string_value)	UPPER('Good Morning')	GOOD MORNING
INITCAP(string_value)	INITCAP('GOOD MORNING')	Good Morning
LTRIM(string_value, trim_text)	LTRIM ('Good Morning', 'Good)	Morning
RTRIM (string_value, trim_text)	RTRIM ('Good Morning', ' Morning')	Good
TRIM (trim_text FROM string_value)	TRIM ('o' FROM 'Good Morning')	Gd Mrning
SUBSTR (string_value, m, n)	SUBSTR ('Good Morning', 6, 7)	Morning
LENGTH (string_value)	LENGTH ('Good Morning')	12
LPAD (string_value, n, pad_value)	LPAD ('Good', 6, '*')	**Good
RPAD (string_value, n, pad_value)	RPAD ('Good', 6, '*')	Good**

Date Functions: These are functions that take values that are of data type DATE as input and return values of data type DATE, except for the MONTHS_BETWEEN function, which returns a number.

Functions	Description
ADD_MONTHS(date, n)	Returns a date value after adding 'n' months to date 'x'.
MONTHS_BETWEEN(x1,x2)	Returns the number of months between date 1& date 2
ROUND(x, date_format)	Returns the date 'x' rounded off to the nearest century, year, month, date, hour, minute, or second as specified by the 'date_format'
TRUNC(x, date_format)	Return the date 'x' lesser than or equal to nearest century, year, month, date, hour, minute, or second as specified by the 'date_format'
NEXT_DAY(x, week_day)	Returns the next date of the date 'week_day' on or after the date 'x' occurs.
LAST_DAY(x)	It is used to determine the number of days remaining in a month from the date 'x' specified
SYSDATE()	Returns the systems current date and time.
NEW_TIME(x, zone1, zone2)	Returns the date and time in zone2 if date 'x' represents the time in zone1.

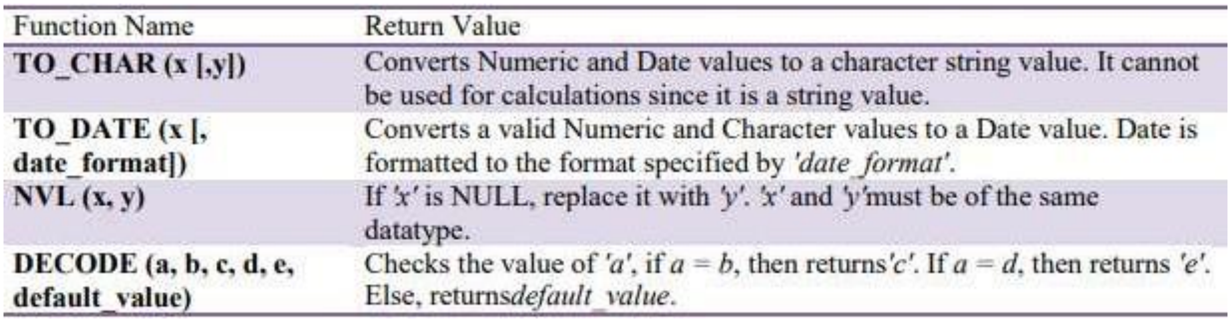
Implementation:

Functions Examples Return Value

ADD_MONTHS()	ADD_MONTHS (‘16-Sep-81’,3)	16-DEC-81
MONTHS_BETWEEN()	MONTHS_BETWEEN(’16-SEP-81’, ’16-DEC-81’)	3
NEXT_DAY()	NEXT_DAY(’01-JUN-08’, ‘Wednesday’)	04-JUN-08
LAST_DAY()	LAST_DAY(’01-JUN-08’)	30-JUN-08
NEW_TIME()	NEW_TIME(’01-JUN-08’, ‘ISL’, ‘EST’)	31-MAY-08

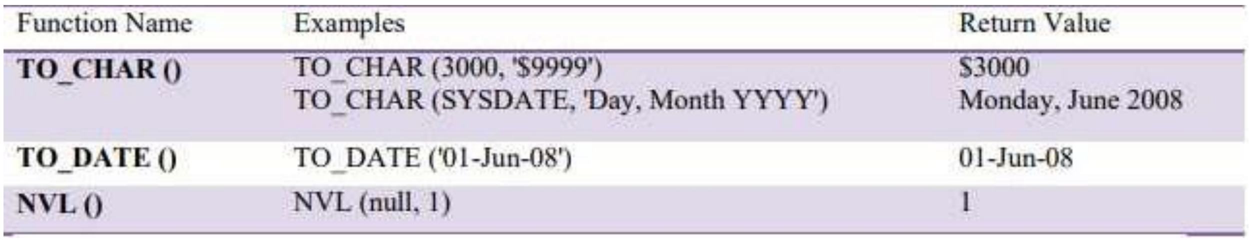
Conversion Functions: These are functions that help us to convert a value in one form to another form. For Example: a null value into an actual value, or a value from one datatype to another datatype like NVL, TO_CHAR, TO_NUMBER, TO_DATE etc.
You can combine more than one function together in an expression. This is known as nesting of functions.

Following are few examples of conversion functions available in oracle.



Function Name	Return Value
TO_CHAR (x [,y])	Converts Numeric and Date values to a character string value. It cannot be used for calculations since it is a string value.
TO_DATE (x [, date_format])	Converts a valid Numeric and Character values to a Date value. Date is formatted to the format specified by *'date_format'*.
NVL (x, y)	If *'x'* is NULL, replace it with *'y'*. *'x'* and *'y'*must be of the same datatype.
DECODE (a, b, c, d, e, default_value)	Checks the value of *'a'*, if *a = b*, then returns*'c'*. If *a = d*, then returns *'e'*. Else, returns*default_value*.

Implementation:



Function Name	Examples	Return Value
TO_CHAR ()	TO_CHAR (3000, '$9999') TO_CHAR (SYSDATE, 'Day, Month YYYY')	$3000 Monday, June 2008
TO_DATE ()	TO_DATE ('01-Jun-08')	01-Jun-08
NVL ()	NVL (null, 1)	1

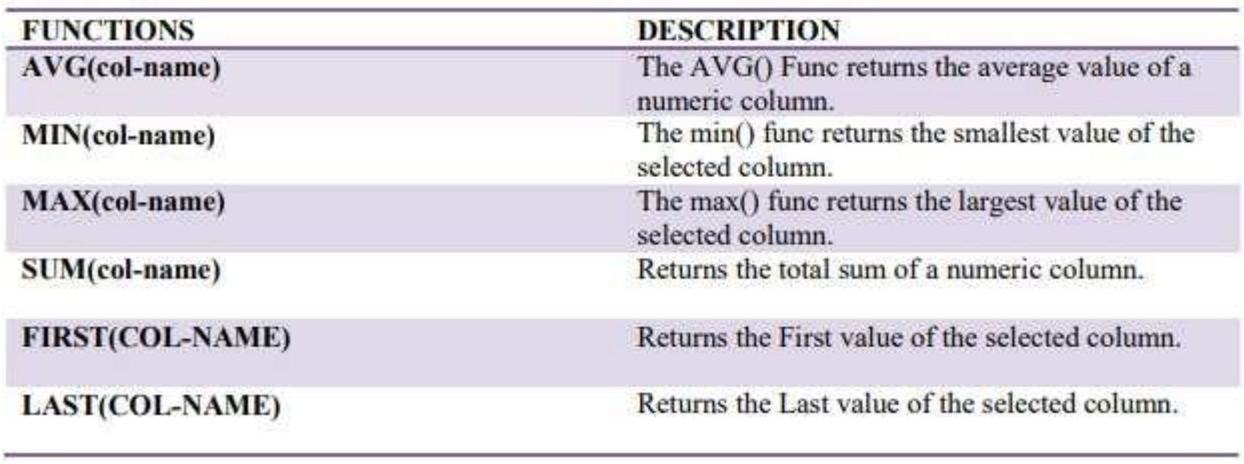
Tasks:

1) Print the full name of employees (first and last name combined) and their department names in uppercase
2) Display employee name and job title together, length of employee name and the numeric position of letter A in employee name, for all employees who are in sales.
   Hint: For finding position you need to use string function **“instr()”,** this function worked as *INSTR(string1, string2)(s1:sreaching string, s2:string/char you’re searching for).*
3) Show the employee ID, name, and job title for employees whose job title ends with the letter 'R'
4) For all employees employed for more than 100 months, display the employee number, hire date, number of months employed, first Friday after hire date and last day of the month hired.
5) Comparing the hire dates for all employees who started in 2003, display the employee number, hire date, and month started using the conversion and date functions.
6) To display the employee number, the month number and year of hiring.
7) To display the employee name and hire date for all employees. The hire date appears as 16 September, 2021.
8) Display the salary of employee STEVEN with $ sign preceded.
9) Find the next ‘Monday’ considering today’s date as date.
10) List all Employees who have an ‘E’ in their last names.

11) Show all employees' last three letters of first name

Aggregate Functions (Group Functions):

A group function is an Oracle SQL function that returns a single result based on many rows, as opposed to single-row functions. These functions are: AVG, COUNT, MIN, MAX, STDDEV, SUM, VARIANCE, etc. Grouping functions may include either of the keywords DISTINCT or ALL.ALL is the default if neither is specified and uses all selected rows in the calculation. DISTINCT uses only one row for each value in the calculations. Note: Group Functions like AVG do not include NULL valued rows. For that we can nest a NULL function into AVG function.
Some Group Functions available in Oracle are:



FUNCTIONS	DESCRIPTION
AVG(col-name)	The AVG() Func returns the average value of a numeric column.
MIN(col-name)	The min() func returns the smallest value of the selected column.
MAX(col-name)	The max() func returns the largest value of the selected column.
SUM(col-name)	Returns the total sum of a numeric column.
FIRST(COL-NAME)	Returns the First value of the selected column.
LAST(COL-NAME)	Returns the Last value of the selected column.

Examples:

1) Show the average salary, minimum salary, maximum salary and count of employees in the organization.
SELECT AVG(salary), MIN(salary), MAX(salary), COUNT(employee_id) FROM employees;

2) Show the earliest and latest hire date of employees.
SELECT MAX(hire_date), MIN(hire_date) FROM employees;

3) Compute the difference between the minimum and maximum salary.
SELECT MAX(salary) - MIN(salary) FROM employees;

4) To show total number of rows in a table.
SELECT COUNT (*) FROM employees;

EXERCISE

1) Display the employee name, job title, and the difference between the maximum and minimum salaries within their department.
2) Display the employee name, salary, and the number of years they have been with the company, rounded to the nearest year.
3) List the employee name and the length of their last name, ordered by the length in descending order.
4) List the employee names with any leading zeros removed from their employee ID
5) Find the names of employees who have been with the company for exactly 5 years and display their employee ID, name, and hire date.
6) List the employee name and the current date formatted as 'DD-MON-YYYY
7) Display the employee name, department ID, and their job title for all employees whose job title contains the letter 'E'.
8) List the employee name, salary, and the number of employees in their department.
9) Display the employee name and the number of days between their hire date and the first day of the following month.
10) Write a query to display the employee name and job title for all employees who have 'Manager' in their job title.
11) Display the employee names with any leading and trailing spaces removed
12) List the employee name, department name, and the difference in days between their hire date and the current date.
13) Write a query to find the employees who were hired on the last day of the month, and display their employee ID, name, and hire date.
14) Write a query to find employee whose ID are greater than 100 and less than 150 and their department_id is greater than 90 and less than 100 along with their first_name, Last_name & Job ID.
15) Display cities in title case (first letter capitalized) where the second character is an “a”