

Name: Shreya Bansal

PRN: 23070521144

Sec: B2

Practical 3.

Part 1.

Write and execute SQL functions- aggregate, numeric, date, string, and conversion.

SQL Aggregate functions

SQL Aggregate Functions are used to perform calculations on a set of rows and return a single value. They are often used with the **GROUP BY** clause in SQL to summarize data for each group. Commonly used aggregate functions include **COUNT()**, **SUM()**, **AVG()**, **MIN()**, and **MAX()**.

Common SQL Aggregate Functions

Count()

- **Count(*)**: Returns the total number of records.
- **Count(salary)**: Return the number of Non-Null values over the column salary.
- **Count(Distinct Salary)**: Return the number of distinct Non-Null values over the column salary.

Sum()

- **sum(salary)**: Sum all Non-Null values of Column salary for example 3120.
- **sum(Distinct salary)**: Sum of all distinct Non-Null values for example 3120.

Avg()

- **Avg(salary)** = $\text{Sum}(\text{salary}) / \text{count}(\text{salary}) = 3120 / 5 = 624$
- **Avg(Distinct salary)** = $\text{sum}(\text{Distinct salary}) / \text{Count}(\text{Distinct Salary}) = 3120 / 5 = 624$

Min()

- **Min(salary)**: Minimum value in the salary column except NULL i.e., 403.

Max():

- **Max(salary)**: Maximum value in the salary i.e., 802.

SQL Aggregate Functions with Examples (Oracle SQL Plus)

Create the Employee table

```
CREATE TABLE Employee (
    Id NUMBER PRIMARY KEY,
    Name CHAR(1),
    Salary NUMBER(10, 2)
);
```

-- Insert data into the Employee table

```
INSERT INTO Employee (Id, Name, Salary)
VALUES (1, 'A', 802);
```

```
INSERT INTO Employee (Id, Name, Salary)
VALUES (2, 'B', 403);
```

```
INSERT INTO Employee (Id, Name, Salary)
VALUES (3, 'C', 604);
```

```
INSERT INTO Employee (Id, Name, Salary)
VALUES (4, 'D', 705);
```

```
INSERT INTO Employee (Id, Name, Salary)
VALUES (5, 'E', 606);
```

```
INSERT INTO Employee (Id, Name, Salary)
VALUES (6, 'F', NULL);
```

-- Commit the changes to make them permanent [OPTIONAL]
COMMIT;

In the following example, we will use multiple aggregate functions on the data.

--Count the number of employees

```
SELECT COUNT(*) AS TotalEmployees FROM Employee;
```

-- Calculate the total salary

```
SELECT SUM(Salary) AS TotalSalary FROM Employee;
```

-- Find the average salary

```
SELECT AVG(Salary) AS AverageSalary FROM Employee;
```

-- Get the highest salary

```
SELECT MAX(Salary) AS HighestSalary FROM Employee;
```

-- Determine the lowest salary

```
SELECT MIN(Salary) AS LowestSalary FROM Employee;
```

Using Aggregate Functions with GROUP BY

GROUP BY allows you to group rows that share a property, enabling you to perform **aggregate calculations** on each group. This is commonly used with the COUNT(), SUM(), AVG(), MIN(), and MAX() functions.

Example: Total Salary by Each Employee

```
SELECT Name, SUM(Salary) AS TotalSalary  
FROM Employee  
GROUP BY Name;
```

Using HAVING with Aggregate Functions

The **HAVING clause** is used to filter results after applying aggregate functions. Unlike WHERE, which filters rows before aggregation, **HAVING filters groups after aggregation.**

Example: Find Employees with Salary Greater Than 600

```
SELECT Name, SUM(Salary) AS TotalSalary  
FROM Employee  
GROUP BY Name  
HAVING SUM(Salary) > 600;
```

SQL | String functions

SQL String Functions are powerful tools that allow you to manipulate, format, and extract specific parts of text data in your database. These functions are essential for tasks like cleaning up data, comparing strings, and combining text fields. Whether you're working with names, addresses, or any form of textual data, mastering SQL string functions is crucial for efficient data handling and analysis.

Common SQL String Functions

String functions are used to perform an operation on input string and return an output string. Following are the string functions defined in SQL:

1. CONCAT(): Concatenate Strings

The CONCAT() function is used to concatenate (combine) two or more strings into one string. It is useful when you want to merge fields like first and last names into a full name.

Query:

```
SELECT CONCAT('John', ' ', 'Doe') AS FullName;
```

Output:

John Doe

2. CHAR_LENGTH() / CHARACTER_LENGTH(): Find String Length

The CHAR_LENGTH() or LENGTH() function returns the length of a string in characters. It's essential for validating or manipulating text data, especially when you need to know how many characters a string contains.

Query:

```
SELECT CHAR_LENGTH('Hello') AS StringLength;
```

Output:

5

3. UPPER() and LOWER(): Convert Text Case

These functions convert the text to uppercase or lowercase, respectively. They are useful for normalizing the case of text in a database.

Query:

```
SELECT UPPER('hello') AS UpperCase;  
SELECT LOWER('HELLO') AS LowerCase;
```

Output:

```
HELLO  
hello
```

4. LENGTH(): Length of String in Bytes

LENGTH() returns the length of a string in bytes. This can be useful for working with multi-byte character sets.

Query:

```
SELECT LENGTH('Hello') AS LengthInBytes;
```

Output:

```
5
```

5. REPLACE(): Replace Substring in String

The REPLACE() function replaces occurrences of a substring within a string with another substring. This is useful for cleaning up data, such as replacing invalid characters or formatting errors.

Query:

```
SELECT REPLACE('Hello World', 'World', 'SQL') AS UpdatedString;
```

Output:

```
Hello SQL
```

6. SUBSTRING() / SUBSTR(): Extract Part of a String

The SUBSTRING() (or SUBSTR()) function is used to extract a substring from a string, starting from a specified position. It is especially useful when you need to extract a specific part of a string, like extracting the domain from an email address.

Query:

```
SELECT SUBSTRING('Hello World', 1, 5) AS SubStringExample;
```

Output:

```
Hello
```

7. LEFT() and RIGHT(): Extract Substring from Left or Right

The LEFT() and RIGHT() functions allow you to extract a specified number of characters from the left or right side of a string, respectively.

Query:

```
SELECT LEFT('Hello World', 5) AS LeftString;  
SELECT RIGHT('Hello World', 5) AS RightString;
```

Output:

```
Hello
```

```
World
```

8. INSTR(): Find Position of Substring

INSTR() finds the position of the first occurrence of a substring in a string. This is useful when you need to locate the position of specific characters within a string.

Query:

```
SELECT INSTR('Hello World', 'World') AS SubstringPosition;
```

Output:

7

9. TRIM(): Remove Leading and Trailing Spaces

INSTR() finds the position of the first occurrence of a substring in a string. This is useful when you need to locate the position of specific characters within a string.

TRIM([[LEADING | TRAILING | BOTH] [character] FROM string])

Query:

```
SELECT TRIM(' ' FROM ' Hello World ') AS TrimmedString;
```

Output:

Hello World

10. REVERSE(): Reverse the String

The REVERSE() function reverses the characters in a string. It's useful in situations where you need to process data backward, such as for password validation or certain pattern matching.

Query:

```
SELECT REVERSE('Hello') AS ReversedString;
```

Output:

olleH

Some Other String Function

These are the some other SQL Functions.

ASCII(): This function is used to find the ASCII value of a character.

Syntax: SELECT ascii('t');

Output: 116

CONCAT_WS(): This function is used to add two words or strings with a symbol as concatenating symbol.

Syntax: SELECT CONCAT_WS('_', 'sitnagpur', 'for', 'sitnagpur');

Output: sitnagpur_for_sitnagpur

FIND_IN_SET(): This function is used to find a symbol from a set of symbols.

Syntax: SELECT FIND_IN_SET('b', 'a, b, c, d, e, f');

Output: 2

FORMAT(): This function is used to display a number in the given format.

Syntax: SELECT FORMAT(0.981 * 100, 'N2') + '%' AS PercentageOutput;

Output: '98.10%'

INSTR(): This function is used to find the occurrence of an alphabet.

Syntax: INSTR('sitnagpur for sitnagpur', 'e');

Output: 2 (the first occurrence of 'e')

LCASE(): This function is used to convert the given string into lower case.

Syntax: LCASE ("SitnagpurFor Sitnagpur To Learn");

Output: sitnagpurforsitnagpur to learn

LOCATE(): This function is used to find the nth position of the given word in a string.

Syntax: SELECT LOCATE('for', 'sitnagpurforsitnagpur', 1);

Output: 6

LPAD(): This function is used to make the given string of the given size by adding the given symbol.

Syntax: LPAD('sitnagpur', 8, '0');

Output:

000sitnagpur

MID(): This function is to find a word from the given position and of the given size.

Syntax: Mid ("sitnagpurforsitnagpur", 6, 2);

Output: for

POSITION(): This function is used to find position of the first occurrence of the given alphabet.

Syntax: SELECT POSITION('e' IN 'sitnagpurforsitnagpur');

Output: 2

REPEAT(): This function is used to write the given string again and again till the number of times mentioned.

Syntax: SELECT REPEAT('sitnagpur', 2);

Output: sitnagpursitnagpur

REPLACE(): This function is used to cut the given string by removing the given sub string.

Syntax: REPLACE('123sitnagpur123', '123');

Output: sitnagpur

RPAD(): This function is used to make the given string as long as the given size by adding the given symbol on the right.

Syntax: RPAD('sitnagpur', 8, '0');

Output: 'sitnagpur000'

RTRIM(): This function is used to cut the given sub string from the original string.

Syntax: RTRIM('sitnagpurxyzyyy', 'xyz');

Output: 'sitnagpur'

SPACE(): This function is used to write the given number of spaces.

Syntax: SELECT SPACE(7);

Output: ' '

STRCMP(): This function is used to compare 2 strings.

Syntax: SELECT STRCMP('google.com', 'sitnagpur.edu.in');

Output: -1

```
SQL> create table employee(
  2  id number primary key,
  3  name char(1),
  4  salary number(10,2)
  5  );

Table created.

SQL> insert into employee(id,name,salary)
  2  values(1,'A',802);

1 row created.

SQL> insert into employee(id,name,salary)
  2  values(2,'B',403);

1 row created.

SQL> insert into employee(id,name,salary)
  2  values(3,'C',604);

1 row created.

SQL> insert into employee(id,name,salary)
  2  values(4,'D',705);

1 row created.

SQL> insert into employee(id,name,salary)
  2  values(5,'E',606);

1 row created.

SQL> insert into employee(id,name,salary)
  2  values(6,'F',NULL);

1 row created.

SQL> COMMIT;

Commit complete.

SQL> select count(*) as totalemployees from employee;

TOTALEMPLOYEES
-----
6

SQL> select sum(salary) as totalsalary from employee;

TOTALSALARY
```



Search

```
SQL> select count(*) as totalemployees from employee;

TOTALEMPLOYEES
-----
6

SQL> select sum(salary) as totalsalary from employee;

TOTALSALARY
-----
3120

SQL> select avg(salary) as averagesalary from employee;

AVERAGESALARY
-----
624

SQL> select max(salary) as highestsalary from employee;

HIGHESTSALARY
-----
802

SQL> select min(salary) as lowestsalary from employee;

LOWESTSALARY
-----
403

SQL> select name,sum(salary) as totalsalary from employee
  2 group by name;

N TOTALSALARY
-----
D      705
A      802
B      403
C      604
E      606
F      606

6 rows selected.

SQL> select name,sum(salary) as totalsalary from employee
  2 group by name
  3 having sum(salary)>600;

N TOTALSALARY
-----
D      705
```



```
SQL> select min(salary) as lowestsalary from employee;
LOWESTSALARY
-----
403

SQL> select name,sum(salary) as totalsalary from employee
2 group by name;

N TOTALSALARY
-
D      705
A      802
B      403
C      604
E      606
F

6 rows selected.

SQL> select name,sum(salary) as totalsalary from employee
2 group by name
3 having sum(salary)>600;

N TOTALSALARY
-
D      705
A      802
C      604
E      606

SQL> select concat("John","Doe") as fullname;
select concat("John","Doe") as fullname
*
ERROR at line 1:
ORA-00923: FROM keyword not found where expected

SQL> select * from employee;

ID N      SALARY
-----
1 A      802
2 B      403
3 C      604
4 D      705
5 E      606
6 F

6 rows selected.
```

86°F
Haze

Search

```
SQL> select concat('John','','Doe') as fullname;
select concat('John','','Doe') as fullname
*
ERROR at line 1:
ORA-00909: invalid number of arguments

SQL> select concat('John','Doe') as fullname;
select concat('John','Doe') as fullname
*
ERROR at line 1:
ORA-00923: FROM keyword not found where expected
```

```
SQL> select concat('John','Doe') as fullname from employee;
```

```
FULLNAM
```

```
-----  
JohnDoe  
JohnDoe  
JohnDoe  
JohnDoe  
JohnDoe  
JohnDoe
```

```
6 rows selected.
```

```
SQL> select * from employee;
```

ID	N	SALARY
1	A	802
2	B	403
3	C	604
4	D	705
5	E	606
6	F	

```
6 rows selected.
```

```
SQL> select char_length('Hello') as stringlength from employees;
select char_length('Hello') as stringlength from employees
*
ERROR at line 1:
ORA-00904: "CHAR_LENGTH": invalid identifier
```

```
SQL> select length('Hello') as stringlength from employees;
```

```
STRINGLENGTH
```



86°F
Haze

Search

```
SQL*Plus  
SUBST  
-----  
Hello  
  
SQL> select left('Hello') as leftstring from employees;  
select left('Hello') as leftstring from employees  
*  
ERROR at line 1:  
ORA-00904: "LEFT": invalid identifier  
  
SQL> select left('Hello World',5) as leftstring from employees;  
select left('Hello World',5) as leftstring from employees  
*  
ERROR at line 1:  
ORA-00904: "LEFT": invalid identifier  
  
SQL> select right('Hello World',5) as rightstring from employees;  
select right('Hello World',5) as rightstring from employees  
*  
ERROR at line 1:  
ORA-00904: "RIGHT": invalid identifier  
  
SQL> select instr('Hello World', 'World') as substringposition from employees;  
SUBSTRINGPOSITION  
-----  
7  
  
SQL> select trim(''FROM'Hello World') as trimmedstring from employees;  
T  
-  
  
SQL> select reverse('Hello') as reversedstring from employee;  
REVER  
-----  
olleH  
olleH  
olleH  
olleH  
olleH  
olleH  
  
6 rows selected.  
SQL> |
```

