PRACTICAL – 5

Aim: To study data communication using SQL functions, focusing on aggregate, numeric, string, conversion, date, and set operations functions, to manipulate and retrieve data effectively from the bank's database.

Constraints –

- Not Null Constraints: Ensure critical fields are not null.
- Unique Constraints: Ensure data integrity by limiting column values.

DESCRIPTION (Theory):

Aggregate Functions: These functions perform calculations on a set of values and return a single value. They include:

- **AVG:** Calculates the average value.
- **MIN:** Finds the minimum value.
- **COUNT:** Counts the number of rows.
- **MAX:** Determines the maximum value.
- **SUM:** Sums up the values.

Numeric Functions: These functions perform mathematical operations on numeric data. They include:

- **ABS:** Returns the absolute value.
- **POWER:** Raises a number to a specified power.
- **ROUND:** Rounds a number to a specified number of decimal places.
- **SQRT:** Computes the square root of a number.
- **String Functions:** These functions manipulate and format string data. They include:
- **LOWER:** Converts text to lowercase.
- **INITCAP:** Capitalizes the first letter of each word.
- **UPPER:** Converts text to uppercase.
- **SUBSTR:** Extracts a substring from a string.
- **LENGTH:** Returns the length of a string.
- **LTRIM:** Removes specified leading characters.
- **RTRIM:** Removes specified trailing characters.
- **LPAD:** Pads the left side of a string with a specified character.
- **RPAD:** Pads the right side of a string with a specified character.

Conversion Functions: These functions convert data from one type to another. They include:

- **TO_NUMBER:** Converts a string to a number.
- **TO_CHAR:** Converts a number to a formatted string.
- **Date Functions:** These functions work with date and time values. They include:
- **TO_DATE:** Converts a string to a date.
- **ADD_MONTHS:** Adds a specified number of months to a date.

- **LAST_DAY:** Returns the last day of the month for a given date.
- **MONTHS_BETWEEN:** Calculates the number of months between two dates.
- **NEXT_DAY:** Returns the next occurrence of a specified day of the week after a given date.

Set Operations: These operations combine results from multiple queries. They include:

- **UNION:** Combines results from multiple queries, removing duplicates.
- **UNION ALL:** Combines results from multiple queries, including duplicates.
- **INTERSECT:** Returns rows common to all queries. • **MINUS:** Returns rows from the first query that are not in the second query.

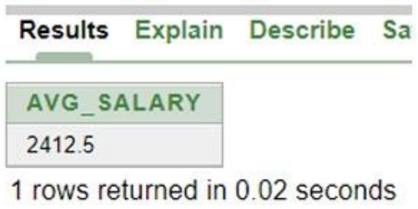
AGGREGATE FUNCTIONS:

1) AVG (DISTINCT | ALL | n)

Test Case: Verify the average salary from the JobProfile table and the distinct average salary.

QUERY:

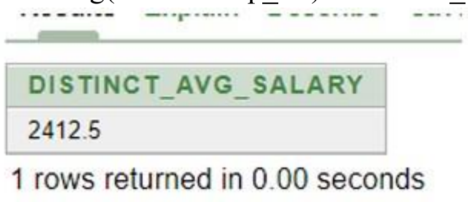
select avg(emp_sal) as avg_salary from employee;



```
AVG_SALARY
2412.5
1 rows returned in 0.02 seconds
```

Explanation: The AVG() function calculates the average of the emp_sal column values. This query returns the average salary of all employees.

select avg(distinct emp_sal) as distinct_avg_salary from employee;



```
DISTINCT_AVG_SALARY
2412.5
1 rows returned in 0.00 seconds
```

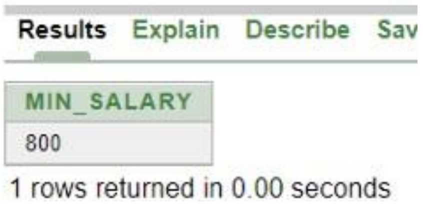
Explanation: The AVG(DISTINCT) function calculates the average salary while considering only unique salary values. This query returns the average salary after eliminating duplicates.

2) MIN (DISTINCT | ALL | expr)

Test Case: Verify the minimum salary from the JobProfile table.

QUERY:

select min(emp_sal) as min_salary from employee;



MIN_SALARY
800

1 rows returned in 0.00 seconds

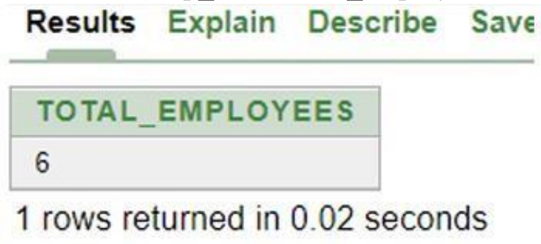
Explanation: The MIN() function finds the minimum value in the emp_sal column. This query retrieves the lowest salary among all employees.

3) COUNT (DISTINCT | ALL | expr)

Test Case: Verify the total count of employees and the count of distinct departments.

QUERY:

select count(emp_no) as total_employees from employee;

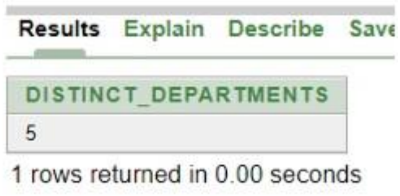


TOTAL_EMPLOYEES
6

1 rows returned in 0.02 seconds

Explanation: The COUNT() function returns the number of rows in the employee table, counting each emp_no. This query returns the total number of employees.

select count(distinct dept_no) as distinct_departments from employee;



DISTINCT_DEPARTMENTS
5

1 rows returned in 0.00 seconds

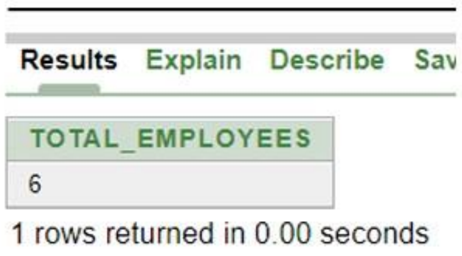
Explanation: The COUNT(DISTINCT) function counts the number of unique dept_no values. This query returns the count of distinct departments.

4) COUNT (*)

Test Case: Verify the total number of employees.

QUERY:

select count(*) as total_employees from employee;



```
TOTAL_EMPLOYEES
6
1 rows returned in 0.00 seconds
```

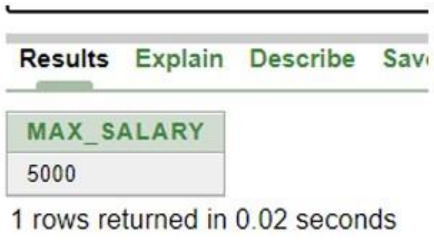
Explanation: The COUNT(*) function returns the total number of rows in the employee table, including those with NULL values. This query counts all employees.

5) MAX (DISTINCT | ALL | expr)

Test Case: Verify the maximum salary from the JobProfile table.

QUERY:

select max(emp_sal) as max_salary from employee;



```
MAX_SALARY
5000
1 rows returned in 0.02 seconds
```

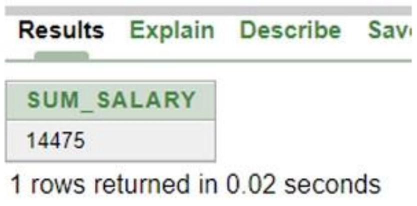
Explanation: The MAX() function finds the highest value in the emp_sal column. This query retrieves the highest salary among all employees.

6) SUM (DISTINCT | ALL | n)

Test Case: Verify the total sum of salaries and the distinct total salary.

QUERY:

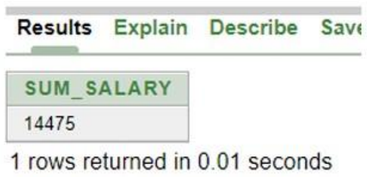
select sum(emp_sal) as sum_salary from employee;



```
SUM_SALARY
14475
1 rows returned in 0.02 seconds
```

Explanation: The SUM() function calculates the total sum of the emp_sal column values. This query returns the total salary of all employees. sql

select sum(distinct emp_sal) as distinct_sum_salary from employee;



```
SUM_SALARY
14475
1 rows returned in 0.01 seconds
```

Explanation: The SUM(DISTINCT) function calculates the sum of unique salary values. This query sums distinct salary values among employees.

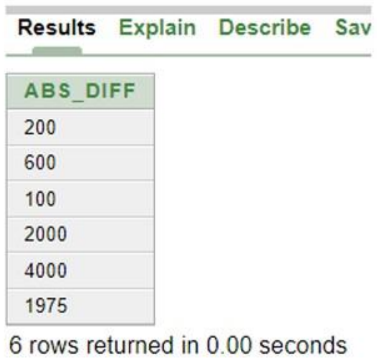
NUMERIC FUNCTIONS:

1) ABS(n)

Test Case: Verify the absolute difference between salary and 1000.

QUERY:

select abs(emp_sal - 1000) as abs_diff from employee;



ABS_DIFF
200
600
100
2000
4000
1975

6 rows returned in 0.00 seconds

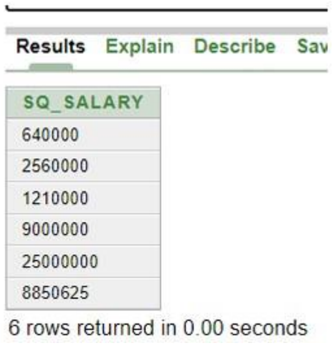
Explanation: The ABS() function returns the absolute value of the difference between the employee's salary and 1000. This query calculates how far each salary is from 1000, without regard to sign.

2) POWER(m, n)

Test Case: Verify the square of the salary.

QUERY:

select power(emp_sal, 2) as sq_salary from employee;



SQ_SALARY
640000
2560000
1210000
9000000
25000000
8850625

6 rows returned in 0.00 seconds

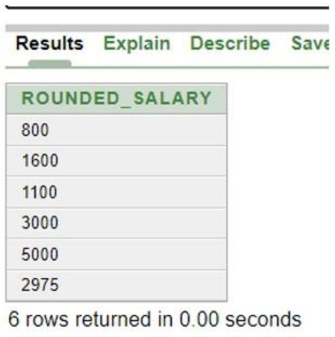
Explanation: The POWER() function raises the salary to the power of 2. This query computes the square of each employee's salary.

3) ROUND(n, m)

Test Case: Verify the salary rounded to 2 decimal places.

QUERY:

select round(emp_sal, 2) as rounded_salary from employee;



```
ROUNDED_SALARY
800
1600
1100
3000
5000
2975
6 rows returned in 0.00 seconds
```

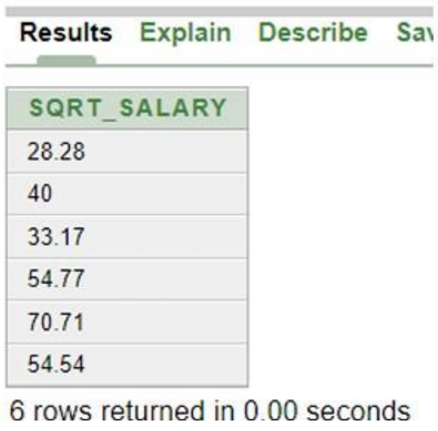
Explanation: The ROUND() function rounds the salary to 2 decimal places. This query provides salaries rounded to the nearest cent.

4) SQRT(n)

   Test Case: Verify the square root of the salary.

   QUERY:

   select round(sqrt(emp_sal), 2) as sqrt_salary from employee;



```
SQRT_SALARY
28.28
40
33.17
54.77
70.71
54.54
6 rows returned in 0.00 seconds
```

   Explanation: The SQRT() function returns the square root of each salary. This query calculates the square root of each salary and rounds it to 2 decimal places.

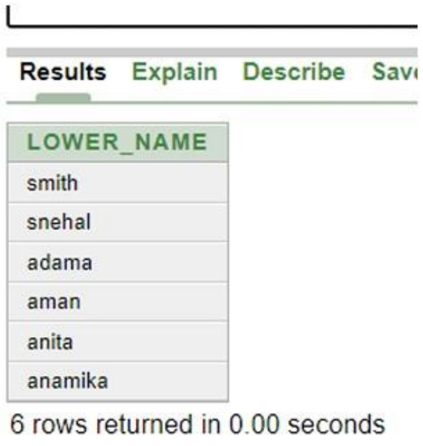
STRING FUNCTIONS:

1) LOWER(char)

   Test Case: Verify the employee first names in lowercase.

   QUERY:

   select lower(emp_name) as lower_name from employee;



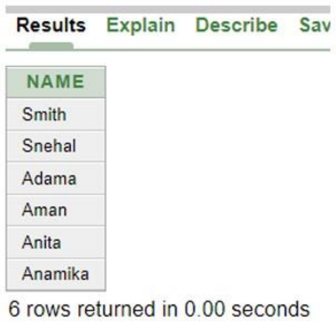
Results Explain Describe Sav

LOWER_NAME
smith
snehal
adama
aman
anita
anamika

6 rows returned in 0.00 seconds

Explanation: The LOWER() function converts all characters in the emp_name column to lowercase. This query transforms employee names to lowercase.

2) INITCAP(char)
Test Case: Verify the employee first names with the initial capital letter.
QUERY:
select initcap(emp_name) as name from employee;



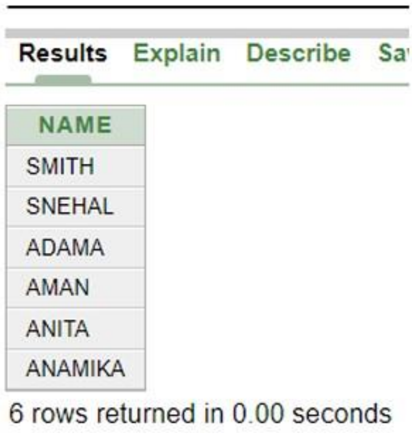
Results Explain Describe Sav

NAME
Smith
Snehal
Adama
Aman
Anita
Anamika

6 rows returned in 0.00 seconds

Explanation: The INITCAP() function capitalizes the first letter of each word in the emp_name column. This query formats employee names with initial capital letters.

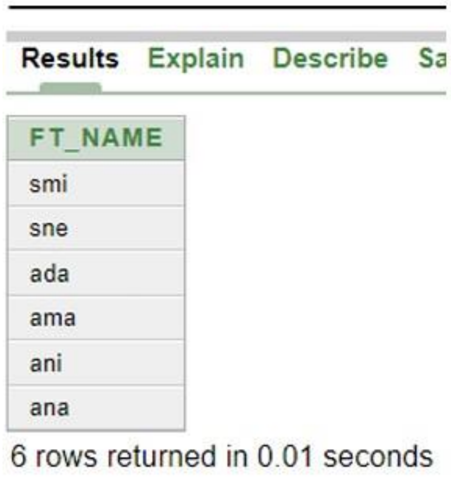
3) UPPER(char)
Test Case: Verify the employee first names in uppercase.
QUERY:
select upper(emp_name) as name from employee;



```
NAME
SMITH
SNEHAL
ADAMA
AMAN
ANITA
ANAMIKA
6 rows returned in 0.00 seconds
```

Explanation: The UPPER() function converts all characters in the emp_name column to uppercase. This query transforms employee names to uppercase.

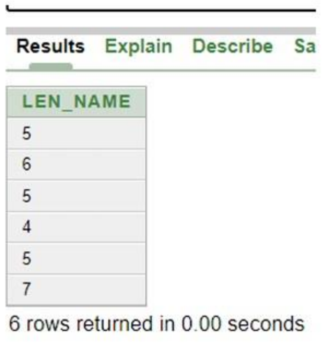
4) SUBSTR(char, m [, n])
   Test Case: Verify the first three characters of the employee first names.
   QUERY:
   select substr(emp_name, 1, 3) as ft_name from employee;



```
FT_NAME
smi
sne
ada
ama
ani
ana
6 rows returned in 0.01 seconds
```

Explanation: The SUBSTR() function extracts a substring from the emp_name column starting at position 1 with a length of 3 characters. This query retrieves the first three characters of employee names.

5) LENGTH(word)
   Test Case: Verify the length of the employee first names.
   QUERY:
   select length(emp_name) as len_name from employee;


Results Explain Describe Sa

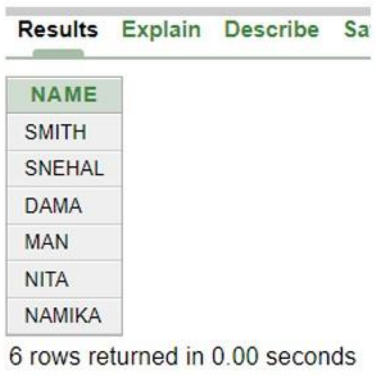
LEN_NAME
5
6
5
4
5
7

6 rows returned in 0.00 seconds

Explanation: The LENGTH() function returns the number of characters in the emp_name column. This query calculates the length of each employee's name.

6) LTRIM(char [, set])
   Test Case: Verify the employee first names with leading 'A' removed.
   QUERY:
   select ltrim(emp_name, 'a') as name from employee;

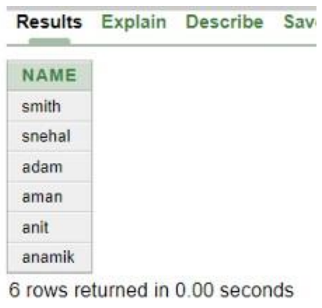
   Explanation: The LTRIM() function removes leading characters specified in the set parameter from the emp_name column. This query removes leading 'a' characters from employee names.


Results Explain Describe Sa

NAME
SMITH
SNEHAL
DAMA
MAN
NITA
NAMIKA

6 rows returned in 0.00 seconds

7) RTRIM(char [, set])
   Test Case: Verify the employee first names with trailing 'a' removed.
   QUERY:
   select rtrim(emp_name, 'a') as name from employee;

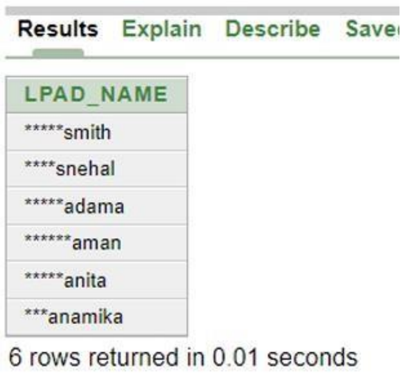


NAME
smith
snehal
adam
aman
anit
anamik

6 rows returned in 0.00 seconds

Explanation: The RTRIM() function removes trailing characters specified in the set parameter from the emp_name column. This query removes trailing 'a' characters from employee names.

8) LPAD(char1, n [, char2])
   Test Case: Verify the employee first names padded to the left with '*' up to a total length of 10.
   QUERY:
   select lpad(emp_name, 10, '*') as lpad_name from employee;

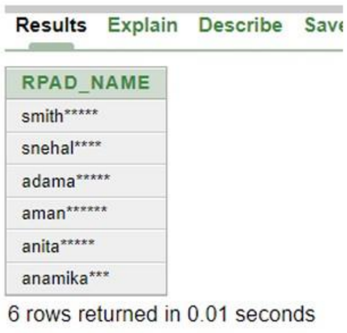


LPAD_NAME
*****smith
****snehal
*****adama
******aman
*****anita
***anamika

6 rows returned in 0.01 seconds

Explanation: The LPAD() function pads the left side of the emp_name column with '' characters until the total length is 10. This query formats employee names to a total length of 10 characters, padding with '' on the left.

9) RPAD(char1, n [, char2])
   Test Case: Verify the employee first names padded to the right with '*' up to a total length of 10.
   QUERY:
   select rpad(emp_name, 10, '*') as rpad_name from employee;



RPAD_NAME
smith*****
snehal****
adama*****
aman******
anita*****
anamika***

6 rows returned in 0.01 seconds

Explanation: The RPAD() function pads the right side of the emp_name column with '' characters until the total length is 10. This query formats employee names to a total length of 10 characters, padding with '' on the right.

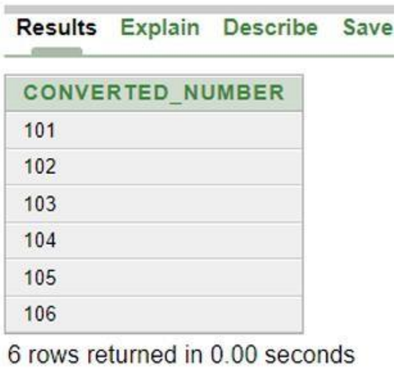
CONVERSION FUNCTIONS:

1) TO_NUMBER(char)

Test Case: Verify the conversion of a string to a number.

QUERY:

select to_number(emp_no) as converted_number from employee;



CONVERTED_NUMBER
101
102
103
104
105
106

6 rows returned in 0.00 seconds

Explanation: The TO_NUMBER() function converts the emp_no column from a string to a numeric value. This query converts employee numbers to numeric format.

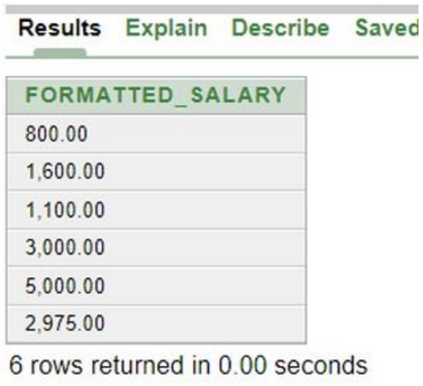
2) TO_CHAR(n [, fmt])

Test Case: Verify the conversion of a number to a formatted string.

QUERY:

select to_char(emp_sal, '999,999.99') as formatted_salary from employee;

Explanation: The TO_CHAR() function converts the numeric emp_sal values into a formatted string according to the specified format '999,999.99'. This format includes commas for thousands and ensures two decimal places. This query displays employee salaries with comma separators and two decimal places.



Results Explain Describe Saved

```
FORMATTED_SALARY
800.00
1,600.00
1,100.00
3,000.00
5,000.00
2,975.00
6 rows returned in 0.00 seconds
```

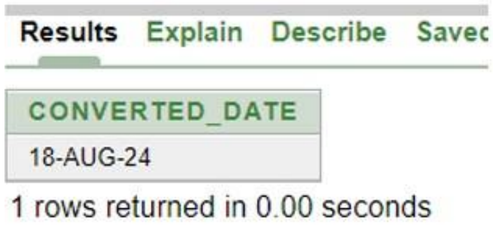
DATE CONVERSION FUNCTIONS:

1) TO_DATE (char [, fmt])

Test Case: Verify the conversion of a string to a date.

QUERY:

select to_date('2024-08-18', 'yyyy-mm-dd') as converted_date from dual;



Results Explain Describe Savec

```
CONVERTED_DATE
18-AUG-24
1 rows returned in 0.00 seconds
```

Explanation: The TO_DATE() function converts the string '2024-08-18' into a date using the specified format 'yyyy-mm-dd'. This query formats the string into a date value.

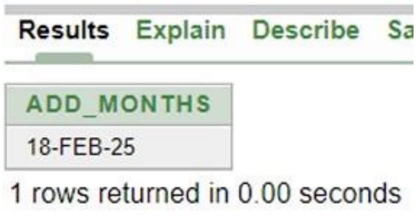
DATE FUNCTIONS:

1) ADD_MONTHS(d, n)

Test Case: Verify the date after adding 6 months to the current date.

QUERY:

select add_months(sysdate, 6) as add_months from dual;



Results Explain Describe Sa

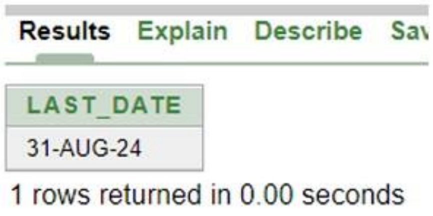
```
ADD_MONTHS
18-FEB-25
1 rows returned in 0.00 seconds
```

Explanation: The ADD_MONTHS() function adds 6 months to the current date (sysdate). This query calculates the date 6 months ahead from today.

2) LAST_DAY(d)

Test Case: Verify the last day of the current month.

QUERY:
select last_day(sysdate) as last_date from dual;

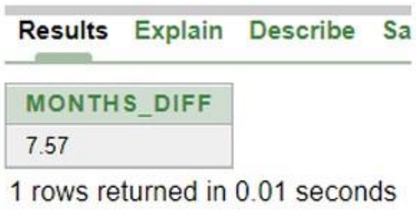


LAST_DATE
31-AUG-24

1 rows returned in 0.00 seconds

Explanation: The LAST_DAY() function returns the last day of the month for the given date (sysdate). This query provides the last day of the current month.

3) MONTHS_BETWEEN(d1, d2)
Test Case: Verify the number of months between two dates.
QUERY:
select round(months_between(sysdate, to_date('2024-01-01', 'yyyy-mm-dd')), 2) as months_diff from dual;

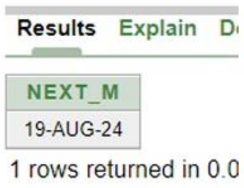


MONTHS_DIFF
7.57

1 rows returned in 0.01 seconds

Explanation: The MONTHS_BETWEEN() function calculates the number of months between sysdate (current date) and '2024-01-01'. This query rounds the result to 2 decimal places to show the difference in months.

4) NEXT_DAY(date, char)
Test Case: Verify the next Monday from the current date.
QUERY:
select next_day(sysdate, 'monday') as next_m from dual;



NEXT_M
19-AUG-24

1 rows returned in 0.0

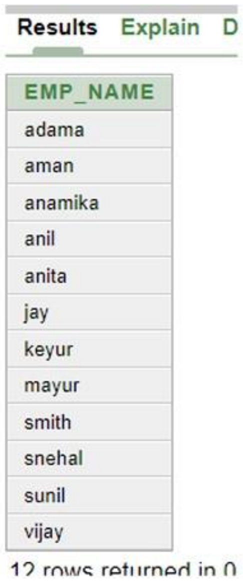
Explanation: The NEXT_DAY() function returns the date of the next occurrence of the specified day of the week ('monday') after the given date (sysdate). This query finds the date of the next Monday from today.

SET OPERATIONS:

1) UNION
Test Case: Verify the union of first names from employees and customers.

QUERY:

select emp_name from employee; UNION
select cname from deposit;



Results Explain D

EMP_NAME
adama
aman
anamika
anil
anita
jay
keyur
mayur
smith
snehal
sunil
vijay

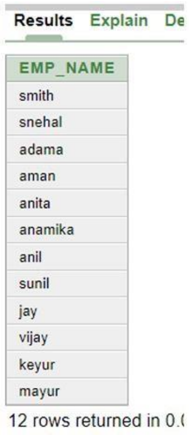
12 rows returned in 0

Explanation: The UNION operator combines the result sets of the two queries and removes duplicate rows. This query merges first names from the employee table and customer names from the deposit table.

2) UNION ALL
Test Case: Verify the union all of first names from employees and customers, including duplicates.

QUERY:

select emp_name from employee UNION
ALL
select cname from deposit;



Results Explain De

EMP_NAME
smith
snehal
adama
aman
anita
anamika
anil
sunil
jay
vijay
keyur
mayur

12 rows returned in 0.(

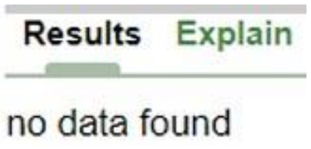
Explanation: The UNION ALL operator combines the result sets of the two queries and includes all rows, including duplicates. This query merges all first names from the employee table with all customer names from the deposit table.

3) INTERSECTION

Test Case: Verify the intersection of first names from employees and customers.

QUERY:

select emp_name from employee INTERSECT
select cname from deposit;


Results Explain

no data found

Explanation: The INTERSECT operator returns only the rows that are common to both queries. This query finds the first names that appear in both the employee and deposit tables. (No data is found as no names are the same in both columns.)

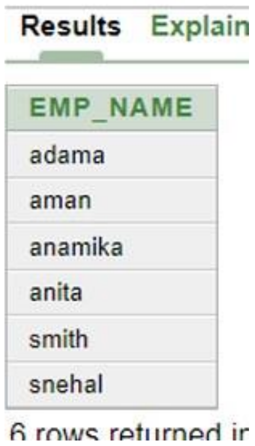
(NO DATA FOUND AS NO NAME IS SAME IN BOTH COLUMNS)

4) MINUS

Test Case: Verify the first names from employees that are not in customers.

QUERY:

select emp_name from employee MINUS
select cname from deposit;


Results Explain

EMP_NAME
adama
aman
anamika
anita
smith
snehal

6 rows returned ir

Explanation: The MINUS operator returns the rows from the first query that are not present in the second query. This query retrieves first names from the employee table that do not appear in the deposit table.

CONCLUSION: Mastering SQL functions and constraints ensures efficient data handling and integrity in databases, helping solve real-world problems and that the solve for the query that. To Create A New like salary calculations and name formatting while maintaining data accuracy.