

Name-Harshit Selarka

Prn-23070521131

Section B

Batch B-2

Practical 3:

Part 4:

SQL Numeric Functions

SQL **Numeric Functions** are essential tools for performing **mathematical** and arithmetic operations on numeric data. These functions allow you to manipulate numbers, perform calculations, and **aggregate data** for reporting and analysis purposes.

Note: In this Lab Manual, examples are based on ORACLE SQLPLUS and MYSQL for your support but you are free to use any platform.

Numeric Functions in SQL*Plus (Oracle) and MySQL

Function	Description
ABS(n)	Returns the absolute value of n
ACOS(n)	Returns the arc cosine (inverse cosine) of n
ASIN(n)	Returns the arc sine (inverse sine) of n
ATAN(n)	Returns the arc tangent (inverse tangent) of n

ATN2(y, x) (SQL Server only)	Returns the angle whose tangent is the quotient of two arguments (Not available in Oracle/MySQL)
-------------------------------------	--

AVG(expression) Returns the average of a set of values

CEILING(n) Returns the smallest integer greater than or equal to **n**

COUNT(expression)	Returns the number of rows matching a condition
COS(n)	Returns the cosine of n (in radians)
COT(n)	Returns the cotangent of n
DEGREES(n)	Converts radians to degrees
EXP(n)	Returns e raised to the power of n
FLOOR(n)	Returns the largest integer less than or equal to n
LOG(n)	Returns the natural logarithm (base e) of n
LOG10(n)	Returns the base-10 logarithm of n
MAX(expression)	Returns the maximum value in a column
MIN(expression)	Returns the minimum value in a column
PI()	Returns the value of π (pi)
POWER(x, y)	Returns x raised to the power of y
RADIANS(n)	Converts degrees to radians
RAND()	Returns a random number between 0 and 1

ROUND(n, d)	Rounds n to d decimal places
--------------------	--

SIGN(n)	Returns -1 , 0 , or 1 depending on the sign of n
SIN(n)	Returns the sine of n (in radians)
SQRT(n)	Returns the square root of n
SQUARE(n)	Returns the square of n (same as POWER(n, 2))

SUM(expression)	Returns the sum of a set of values
TAN(n)	Returns the tangent of n (in radians)

2. Examples in SQL*Plus (Oracle) /skip if you want to use mysql platform

2.1 Absolute Value (ABS)

```
SELECT ABS(-10) FROM dual; -- Result: 10
```

Input

Run SQL

-- Online SQL Editor to Run SQL Online.
-- Use the editor to create new tables, insert data and all other SQL operations.
-- Absolute Value (ABS)
SELECT ABS(-10) AS AbsoluteValue;

Output

AbsoluteValue
10

2.2 Arc Cosine (ACOS)

SELECT ACOS(0.5) FROM dual; -- Result: 1.04719755 (in radians)

The screenshot shows an online SQL editor interface. On the left, there is a schema diagram with three tables: Customers, Orders, and Shippings. The Customers table has columns: customer_id (int), first_name (varchar(100)), last_name (varchar(100)), age (int), and country (varchar(100)). The Orders table has columns: order_id (integer), item (varchar(100)), amount (integer), and customer_id (integer). The Shippings table has columns: shipping_id (integer), status (integer), and customer (integer). The main area is titled 'Input' and contains the following SQL code:

```
-- Online SQL Editor to Run SQL Online.  
-- Use the editor to create new tables, insert data and all other SQL operations.  
-- Arc Cosine (ACOS)  
SELECT ACOS(0.5) AS ArcCosine;
```

Below the input area, there is an 'Output' section showing the result of the query:

ArcCosine
1.0471975511965979

2.3 Arc Sine (ASIN)

SELECT ASIN(0.5) FROM dual; -- Result: 0.523598775 (in radians)

Input

Run SQL

```
-- Online SQL Editor to Run SQL Online.
-- Use the editor to create new tables, insert data and all other SQL operations.
-- Arc Sine (ASIN)
SELECT ASIN(0.5) AS ArcSine;
```

Output

ArcSine
0.5235987755982989

2.4 Arc Tangent (ATAN)

SELECT ATAN(1) FROM dual; -- Result: 0.785398163 (in radians)

The screenshot shows an online SQL editor interface. On the left, there is a database schema with three tables: Customers, Orders, and Shippings. The Customers table has columns: customer_id (int), first_name (varchar(100)), last_name (varchar(100)), age (int), and country (varchar(100)). The Orders table has columns: order_id (integer), item (varchar(100)), amount (integer), and customer_id (integer). The Shippings table has columns: shipping_id (integer), status (integer), and customer (integer). The main area is divided into 'Input' and 'Output' sections. The 'Input' section contains the following SQL code:

```
-- Online SQL Editor to Run SQL Online.  
-- Use the editor to create new tables, insert data and all other SQL operations.  
-- Arc Tangent (ATAN)  
SELECT ATAN(1) AS ArcTangent;
```

The 'Output' section shows the result of the query:

ArcTangent
0.7853981633974483

2.5 Average (AVG)

SELECT AVG(salary) FROM employees; -- Finds the average salary

The screenshot shows an online SQL editor interface. On the left, there is a schema diagram with four tables: Customers, Employees, Orders, and Shippings. The Customers table has columns: customer_id (int), first_name (varchar(100)), last_name (varchar(100)), age (int), and country (varchar(100)). The Employees table has columns: id (int auto_increment), name (varchar(100)), department (varchar(50)), and salary (decimal(10,2)). The Orders table has columns: order_id (integer), item (varchar(100)), amount (integer), and customer_id (integer). The Shippings table has columns: shipping_id (integer), status (integer), and customer (integer). The main editor area is titled 'Input' and contains the following SQL code:

```
-- Online SQL Editor to Run SQL Online.  
-- Use the editor to create new tables, insert data and all other SQL operations.  
-- Insert sample data  
INSERT INTO employees (name, department, salary) VALUES  
( 'Alice', 'Sales', 50000),  
( 'Bob', 'HR', 60000),  
( 'Charlie', 'Sales', 70000),  
( 'David', 'IT', 80000);
```

Below the input area is an 'Output' section which displays the message: "SQL query successfully executed. However, the result set is empty."

2.6 Ceiling (CEILING)

SELECT CEIL(4.2) FROM dual; -- Result: 5

The screenshot shows an online SQL editor interface. On the left, there is a schema section with three tables: Customers, Orders, and Shippings. The Customers table has columns: customer_id (int), first_name (varchar(100)), last_name (varchar(100)), age (int), and country (varchar(100)). The Orders table has columns: order_id (integer), item (varchar(100)), amount (integer), and customer_id (integer). The Shippings table has columns: shipping_id (integer), status (integer), and customer (integer). The main area is titled 'Input' and contains the following SQL code:

```
-- Online SQL Editor to Run SQL Online.  
-- Use the editor to create new tables, insert data and all other SQL operations.  
-- Ceiling (CEIL)  
SELECT CEIL(4.2) AS CeilingValue;
```

Below the input area, there is an 'Output' section showing the result of the query:

CeilingValue
5

2.7 Count (COUNT)

```
SELECT COUNT(*) FROM employees WHERE department =  
'Sales'; -- Counts employees in Sales
```

Employees [-]	
id	[int auto_increment]
name	[varchar(100)]
department	
	[varchar(50)]
salary	[decimal(10,2)]

Orders [-]	
order_id	[integer]
item	[varchar(100)]
amount	[integer]
customer_id	[integer]

Shippings [-]	
shipping_id	[integer]
status	[integer]
customer	[integer]

```
-- Count (COUNT)
SELECT COUNT(*) AS EmployeeCount FROM employees WHERE department = 'Sales';
```

Output

EmployeeCount
6

2.8 Cosine (COS)

```
SELECT COS(0) FROM dual; -- Result: 1
```

Customers [-]

customer_id [int]

first_name [varchar(100)]

last_name [varchar(100)]

age [int]

country [varchar(100)]

Employees [-]

id [int auto_increment]

name [varchar(100)]

department [varchar(50)]

salary [decimal(10,2)]

Orders [-]

order_id [integer]

item [varchar(100)]

amount [integer]

customer_id [integer]

Shipments [-]

shipping_id [integer]

status [integer]

customer [integer]

< input

Run SQL

```
-- Cosine (COS)
SELECT COS(0) AS CosineValue
```

Output

CosineValue
1

2.9 Cotangent (COT)

`SELECT 1/TAN(PI()/4) FROM dual; -- Result: 1`

The screenshot shows a SQL IDE interface. On the left, there is a schema browser with four tables: Customers, Employees, Orders, and Shippings. The main area is titled 'Input' and contains the following SQL query:

```
-- Cotangent (COT)  
SELECT 1/TAN(PI()/4) AS CotangentValue;
```

Below the input area, there is an 'Output' section showing the result of the query:

CotangentValue
1.0000000000000002

2.10 Convert Radians to Degrees (DEGREES)

SELECT DEGREES(PI()/2) FROM dual; -- Result: 90

The screenshot shows a SQL IDE interface. On the left, there is a schema browser with four tables: Customers, Employees, Orders, and Shippings. The main area is divided into 'Input' and 'Output' sections. The 'Input' section contains the following SQL query:

```
-- Convert Radians to Degrees (DEGREES)
SELECT DEGREES(PI()/2) AS DegreesValue;
```

The 'Output' section displays the result of the query in a table with one row:

DegreesValue
90

2.11 Exponential (EXP)

`SELECT EXP(2) FROM dual; -- Result: 7.389056099`

The screenshot shows a SQL IDE interface. On the left, there is a schema explorer with four tables: Customers, Employees, Orders, and Shippings. The main editor area is titled 'Input' and contains the following SQL code:

```
-- Exponential (EXP)
SELECT EXP(2) AS ExponentialValue;
```

Below the editor, the 'Output' section displays the result of the query in a table format:

ExponentialValue
7.38905609893065

2.12 Floor (FLOOR)

SELECT FLOOR(4.8) FROM dual; -- Result: 4

The screenshot shows a SQL IDE interface. On the left, there is a schema explorer with four tables: Customers, Employees, Orders, and Shippings. The main area is divided into 'Input' and 'Output' sections. The 'Input' section contains the SQL query: `SELECT FLOOR(4.8) AS FloorValue;`. The 'Output' section shows the result of the query as a table with one column 'FloorValue' and one row containing the value '4'.

Customers [-]

- customer_id [int]
- first_name [varchar(100)]
- last_name [varchar(100)]
- age [int]
- country [varchar(100)]

Employees [-]

- id [int auto_increment]
- name [varchar(100)]
- department [varchar(50)]
- salary [decimal(10,2)]

Orders [-]

- order_id [integer]
- items [varchar(100)]
- amount [integer]
- customer_id [integer]

Shippings [-]

- shipping_id [integer]
- status [integer]
- customer [integer]

Input

```
SELECT FLOOR(4.8) AS FloorValue;
```

Output

FloorValue
4

2.13 Natural Logarithm (LOG)

`SELECT LOG(2.718281828) FROM dual; -- Result: 1 (since $e^1 = e$)`

Customers [-]

customer_id [int]
first_name [varchar(100)]
last_name [varchar(100)]
age [int]
country [varchar(100)]

Employees [-]

id [int auto_increment]
name [varchar(100)]
department [varchar(50)]
salary [decimal(10,2)]

Orders [-]

order_id [integer]
item [varchar(100)]
amount [integer]
customer_id [integer]

Shipings [-]

shipping_id [integer]
status [integer]
customer [integer]

Input

-- Natural Logarithm (LOG)
SELECT LOG(2.718281828) AS NaturalLog;

Output

NaturalLog
0.99999999998311266

Run SQL

2.14 Logarithm Base 10 (LOG10)

SELECT LOG(10, 100) FROM dual; -- Result: 2 (since $10^2 = 100$)

The screenshot displays a SQL IDE interface. On the left, a database schema is shown with four tables: Customers, Employees, Orders, and Shippings. The Customers table has columns: customer_id (int), first_name (varchar(100)), last_name (varchar(100)), age (int), and country (varchar(100)). The Employees table has columns: id (int auto_increment), name (varchar(100)), department (varchar(50)), and salary (decimal(10,2)). The Orders table has columns: order_id (integer), item (varchar(100)), amount (integer), and customer_id (integer). The Shippings table has columns: shipping_id (integer), status (integer), and customer (integer).

The main window is titled 'Input' and contains the following SQL query:

```
-- Logarithm Base 10 (LOG10)
SELECT LOG10(100) AS LogBase10;
```

The 'Output' section shows the result of the query:

LogBase10
2

2.15 Maximum (MAX)

`SELECT MAX(salary) FROM employees; -- Finds the highest salary`

The screenshot shows a SQL IDE interface. On the left, there is a schema explorer with four tables: Customers, Employees, Orders, and Shippings. The main editor area is titled 'input' and contains the following SQL query:

```
-- Maximum (MAX)
SELECT MAX(salary) AS MaxSalary FROM employees;
```

Below the editor, there is an 'Output' section displaying the result of the query in a table format:

MaxSalary
80000

2.16 Minimum (MIN)

`SELECT MIN(salary) FROM employees; -- Finds the lowest salary`



2.17 Pi (PI)

```
SELECT ACOS(-1) FROM dual; -- Result: 3.14159265
```



2.18 Power (POWER)

```
SELECT POWER(3, 2) FROM dual; -- Result: 9
```



2.19 Convert Degrees to Radians (RADIANS)

```
SELECT RADIANS(180) FROM dual; -- Result: 3.14159265
```



2.20 Random Number (RAND)

```
SELECT DBMS_RANDOM.VALUE FROM dual; -- Returns a random  
number between 0 and 1
```

2.21 Round Number (ROUND)

```
SELECT ROUND(3.14159265, 2) FROM dual; -- Result: 3.14
```



2.22 Sign of Number (SIGN)

```
SELECT SIGN(-10) FROM dual; -- Result: -1
```

```
SELECT SIGN(0) FROM dual; -- Result: 0
```

```
SELECT SIGN(10) FROM dual; -- Result: 1
```



2.23 Sine (SIN)

```
SELECT SIN(PI()/2) FROM dual; -- Result: 1
```




2.24 Square Root (SQRT)

```
SELECT SQRT(16) FROM dual; -- Result: 4
```



2.25 Square (SQUARE)

```
SELECT POWER(4, 2) FROM dual; -- Result: 16
```



2.26 Sum (SUM)

```
SELECT SUM(salary) FROM employees; -- Sum of all salaries
```



2.27 Tangent (TAN)

```
SELECT TAN(PI()/4) FROM dual; -- Result: 1
```



3. Examples in MySQL //SKIP IF DONE WITH ORACLE SQLPLUS

💡💡 The MySQL syntax is almost the same as Oracle, except for some functions.

```
SELECT ABS(-10); -- 10
SELECT ACOS(0.5); -- 1.04719755
SELECT ASIN(0.5); -- 0.523598775
SELECT ATAN(1); -- 0.785398163
SELECT AVG(salary) FROM employees;
SELECT CEIL(4.2); -- 5
SELECT COUNT(*) FROM employees WHERE department =
```

```

'Sales' ;
SELECT COS(0); -- 1
SELECT COT(1); -- 0.6420926159
SELECT DEGREES(PI()/2); -- 90
SELECT EXP(2); -- 7.389056099
SELECT FLOOR(4.8); -- 4
SELECT LOG(2.718281828); -- 1
SELECT LOG10(100); -- 2
SELECT MAX(salary) FROM employees;
SELECT MIN(salary) FROM employees;
SELECT PI(); -- 3.1415926535
SELECT POWER(3, 2); -- 9
SELECT RADIANS(180); -- 3.1415926535
SELECT RAND(); -- Random number
SELECT ROUND(3.14159265, 2); -- 3.14
SELECT SIGN(-10); -- -1
SELECT SIN(PI()/2); -- 1
SELECT SQRT(16); -- 4
SELECT SUM(salary) FROM employees;
SELECT TAN(PI()/4); -- 1

```

Advanced SQL Numeric Function Use Cases (Oracle & MySQL)

Following are **complex queries** using **numeric functions** in **real-world applications** for **financial analysis**, **scientific calculations**, **data analytics** and **system performance monitoring**.

1 Financial Analytics: Compound Interest Calculation Use

Case: Calculate compound interest for a bank's customer accounts.

```

SELECT
  account_id,
  principal,
  interest_rate,
  years,
  ROUND(principal * POWER((1 + interest_rate / 100),
years), 2) AS future_value
FROM savings_accounts;

```

Formula Used:

$$FV = P \times (1 + r/n)^{nt}$$

Where:

- **principal**: Initial deposit
- **interest_rate**: Annual interest rate
- **years**: Time period
- **POWER()** function computes exponentiation

Result Example:

account_id	principal	interest_rate	years	future_value
------------	-----------	---------------	-------	--------------

101 1000 5 10 1628.89



Purchases

Use Case: Find **customer spending variability** to create better promotions.

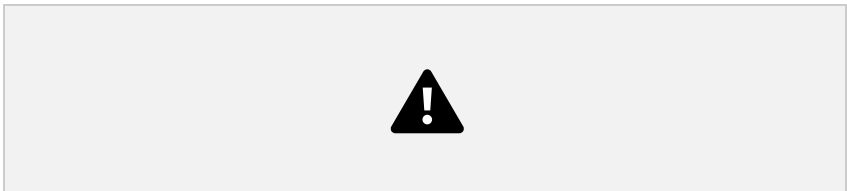
```
SELECT
  customer_id,
  ROUND(AVG(purchase_amount), 2) AS avg_spend,
  ROUND(STDDEV(purchase_amount), 2) AS
  spending_variability
FROM orders
GROUP BY customer_id
HAVING COUNT(*) > 5; -- Filter frequent customers
```

Key Insights:

- Uses `STDDEV()` to measure customer spending consistency.
- Filters for customers with at least **5 purchases** (`HAVING COUNT(*) > 5`).

Result Example:

customer_id	avg_spend	spending_variability
201	250.00	50.25
305	500.00	120.75



Use Case: Compute **CPU load trend** for a cloud server system.

```
SELECT
  server_id,
  ROUND(AVG(cpu_usage), 2) AS avg_cpu,
  ROUND(MAX(cpu_usage), 2) AS max_cpu,
  ROUND(MIN(cpu_usage), 2) AS min_cpu,
  ROUND(SQRT(POWER(MAX(cpu_usage) - MIN(cpu_usage),
2)), 2) AS load_variance
FROM server_logs
WHERE log_date >= SYSDATE - INTERVAL '7' DAY -- Last 7
days
GROUP BY server_id;
```

Key Metrics:

- **AVG()** to measure **average CPU usage**.
- **MAX()** & **MIN()** for **peak & lowest usage**.
- **SQRT(POWER())** to calculate variance in load.

Result Example:

server_id	avg_cpu	max_cpu	min_cpu	load_variance
A123	65.25	92.75	45.50	47.25
B456	40.10	75.00	20.20	54.80



4 Fraud Detection: Identifying Abnormal Transaction Use Case:

Detect transactions that are significantly **higher** than the usual customer behavior.

```
SELECT
    transaction_id,
    customer_id,
    amount,
    (SELECT AVG(amount) FROM transactions WHERE
     customer_id = t.customer_id) AS avg_amount,
    (SELECT STDDEV(amount) FROM transactions WHERE
     customer_id = t.customer_id) AS stddev_amount FROM
    transactions t
WHERE amount > (SELECT AVG(amount) + 2 * STDDEV(amount)
FROM transactions WHERE customer_id = t.customer_id);
```

Logic:

- **Outlier transactions** are those **greater than 2 standard deviations** from the average.
- Uses **AVG()** and **STDDEV()** **per customer** to personalize fraud detection.

Result Example:

transacti on_id	custome r_id	amo un t	avg_am ou nt	stddev_am ou nt
90872	201	12000	3000	4500

💡💡 If a customer usually spends \$3,000 ± \$4,500, a \$12,000 transaction is flagged as suspicious

5 Logistics: Estimating Delivery Time Based on Distance

Use Case: Predict **delivery time (in hours)** for orders based on **distance** and **speed factors**.

```
SELECT
  order_id,
  distance_km,
  ROUND(distance_km / avg_speed, 2) AS
  estimated_delivery_time
FROM (
  SELECT order_id, distance_km,
  CASE
    WHEN distance_km < 50 THEN 60 -- Urban: 60 km/h
    WHEN distance_km BETWEEN 50 AND 200 THEN 80 --
    Suburban: 80 km/h
    ELSE 100 -- Highway: 100 km/h
  END AS avg_speed
  FROM orders
);
```

Business Insight:

- Uses **speed categories** (CASE) to calculate **realistic delivery estimates**.

- Uses `ROUND()` to **format** the estimated time.

Result Example:

order_id	distance_km	estimated_delivery_time (hrs)
101	120	1.50
202	20	0.33

6 Astronomy/Physics: Calculating Earthquake Magnitude (Logarithmic Formula)

Use Case: Compute earthquake **Richter magnitude** based on **seismic wave amplitude**.

```
SELECT
  earthquake_id,
  station_id,
  amplitude,
  ROUND(LOG10(amplitude) + 3, 2) AS magnitude
FROM seismic_readings;
```

Richter Scale Formula:

$$M = \log_{10}(A) + 3 \quad M = \log_{10}(A) + 3$$

- Uses `LOG10()` to calculate **magnitude from amplitude**.

Result Example:

earthquake_id	station_id	amplitude	magnitude
EQ001	ST1001	5000	6.70

7 E-Commerce: Personalized Discount Calculation Use

Case: Apply dynamic **discount rates** based on **spending history**.

```
SELECT
  customer_id,
  total_spent,
  CASE
    WHEN total_spent > 10000 THEN ROUND(total_spent * 0.15,
2)
    WHEN total_spent BETWEEN 5000 AND 10000 THEN
ROUND(total_spent * 0.10, 2)
    ELSE ROUND(total_spent * 0.05, 2)
  END AS discount
FROM (
  SELECT customer_id, SUM(order_value) AS total_spent
FROM orders GROUP BY customer_id
);
```

Discount Strategy:

- **15% off** for VIP customers (> \$10,000)
- **10% off** for mid-level (\$5,000 - \$10,000)
- **5% off** for casual shoppers (< \$5,000)

Result Example:

customer_id	total_spent	discount
101	12000	1800

202	7500	750
-----	------	-----