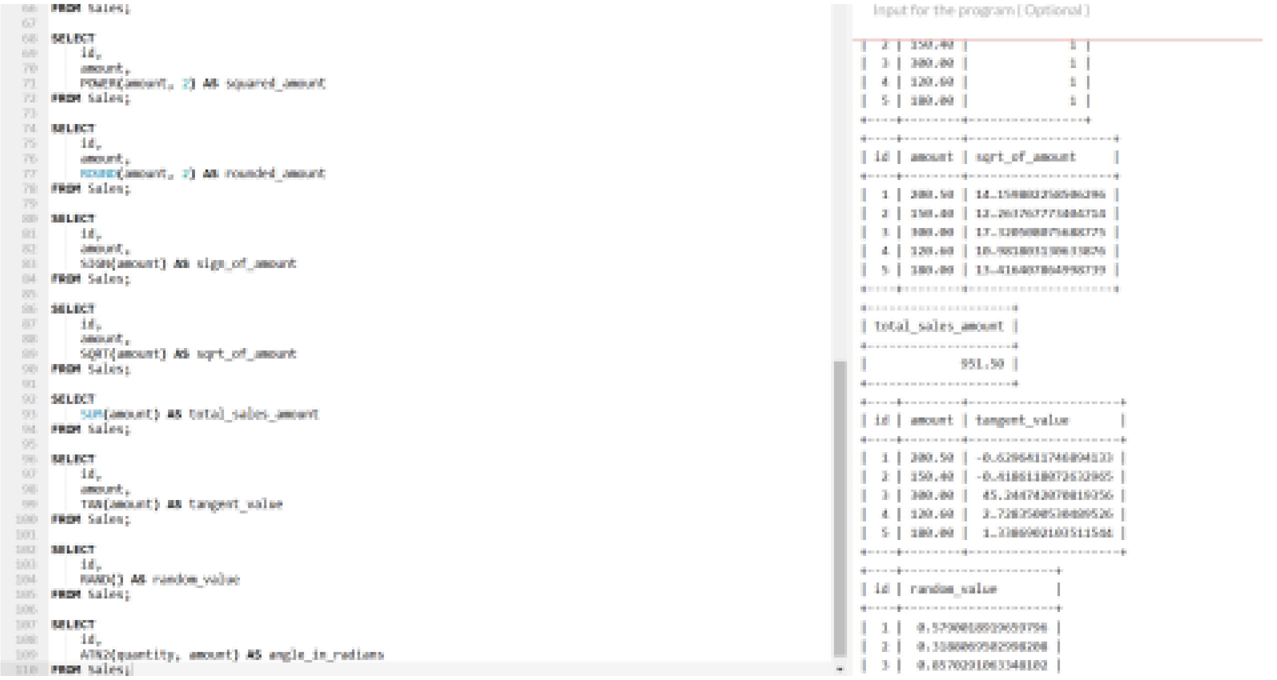
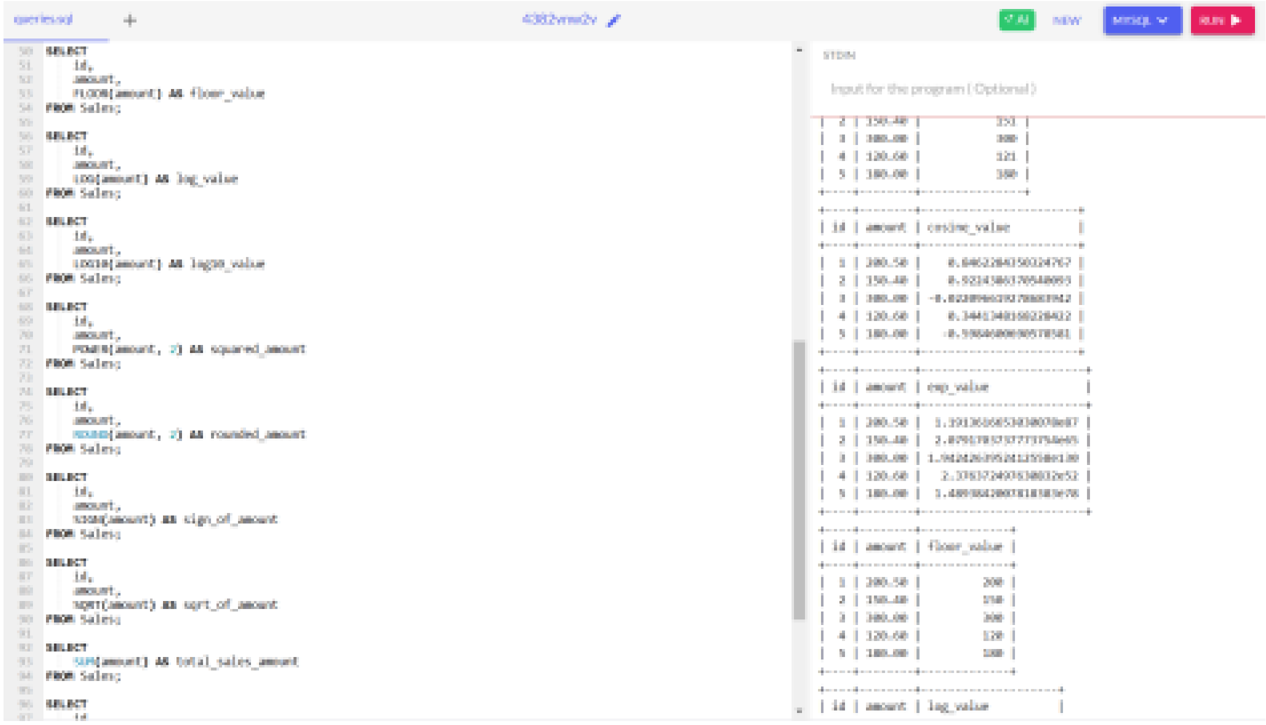


```
FROM Sales;

SELECT
    id,
    amount,
    POWER(amount, 2) AS squared_amount
FROM Sales;

SELECT
    id,
    amount,
    ROUND(amount, 2) AS rounded_amount
FROM Sales;

SELECT
    id,
    amount,
    SIGN(amount) AS sign_of_amount
FROM Sales;

SELECT
    id,
    amount,
    SQRT(amount) AS sqrt_of_amount
FROM Sales;

SELECT
    SUM(amount) AS total_sales_amount
FROM Sales;

SELECT
    id,
    amount,
    TAN(amount) AS tangent_value
FROM Sales;

SELECT
    id,
    RAND() AS random_value
FROM Sales;

SELECT
    id,
    ATN2(quantity, amount) AS angle_in_radians
FROM Sales;
```

Input for the program (Optional)

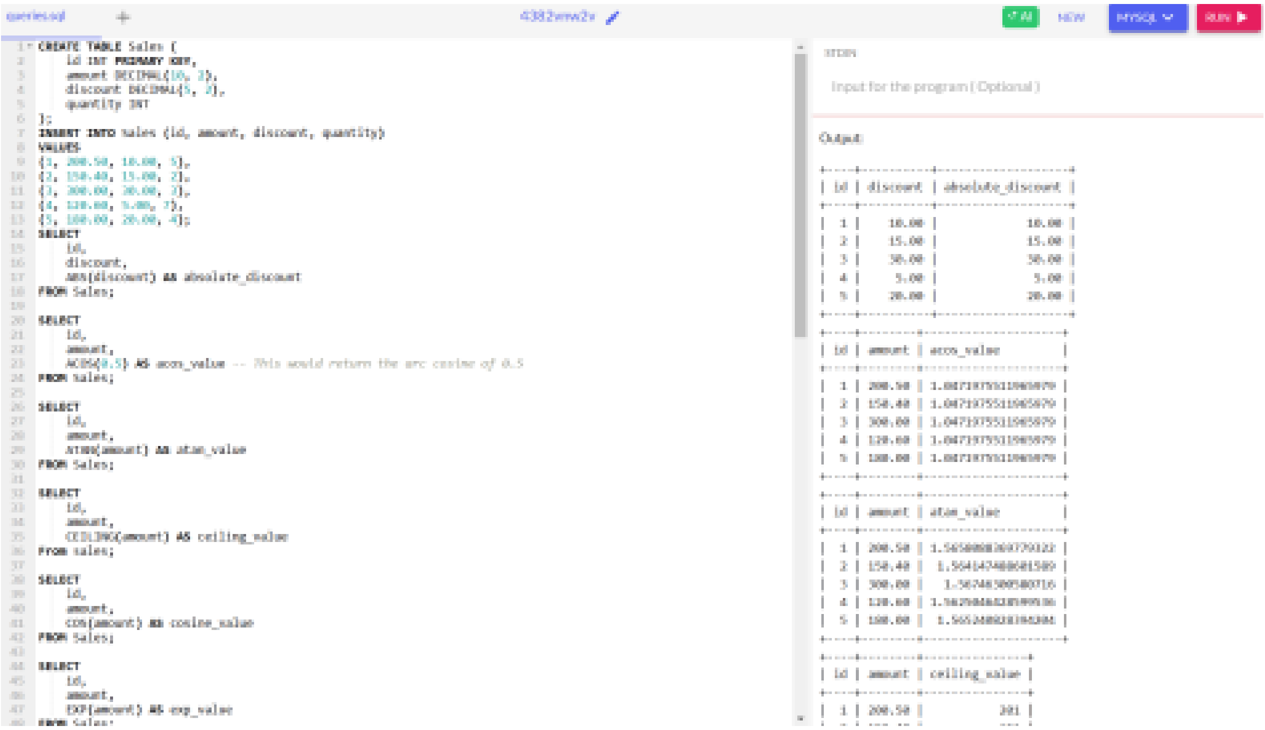
```
|  2 | 150.40 |                1 |
|  3 | 300.00 |                1 |
|  4 | 120.60 |                1 |
|  5 | 180.00 |                1 |
+----+--------+------------------+
+----+--------+--------------------+
| id | amount | sqrt_of_amount     |
+----+--------+--------------------+
|  1 | 200.50 | 14.159802258506296 |
|  2 | 150.40 | 12.263767773404714 |
|  3 | 300.00 | 17.320508075688775 |
|  4 | 120.60 | 10.981803130633826 |
|  5 | 180.00 | 13.416407864998739 |
+----+--------+--------------------+
+--------------------+
| total_sales_amount |
+--------------------+
|             951.50 |
+--------------------+
+----+--------+---------------------+
| id | amount | tangent_value       |
+----+--------+---------------------+
|  1 | 200.50 | -0.6296811746894133 |
|  2 | 150.40 | -0.4186118072632965 |
|  3 | 300.00 |  45.244742070819356 |
|  4 | 120.60 |  2.7283500530489526 |
|  5 | 180.00 |  1.3386902103511544 |
+----+--------+---------------------+
+----+---------------------+
| id | random_value        |
+----+---------------------+
|  1 |  0.5790018819659756 |
|  2 |  0.3188069502998208 |
|  3 |  0.8578291063348102 |
```



```
SELECT
    id,
    amount,
    FLOOR(amount) AS floor_value
FROM Sales;

SELECT
    id,
    amount,
    LOG(amount) AS log_value
FROM Sales;

SELECT
    id,
    amount,
    LOG10(amount) AS log10_value
FROM Sales;

SELECT
    id,
    amount,
    POWER(amount, 2) AS squared_amount
FROM Sales;

SELECT
    id,
    amount,
    ROUND(amount, 2) AS rounded_amount
FROM Sales;

SELECT
    id,
    amount,
    SIGN(amount) AS sign_of_amount
FROM Sales;

SELECT
    id,
    amount,
    SQRT(amount) AS sqrt_of_amount
FROM Sales;

SELECT
    SUM(amount) AS total_sales_amount
FROM Sales;

SELECT
    id
```

Input for the program (Optional)

```
|  2 | 150.40 |            151 |
|  3 | 300.00 |            300 |
|  4 | 120.60 |            121 |
|  5 | 180.00 |            180 |
+----+--------+----------------+
+----+--------+----------------------+
| id | amount | cosine_value         |
+----+--------+----------------------+
|  1 | 200.50 |   0.6452284350324767 |
|  2 | 150.40 |   0.9224306170540093 |
|  3 | 300.00 | -0.022096639278603942 |
|  4 | 120.60 |   0.3441340160228422 |
|  5 | 180.00 |  -0.5984600690578581 |
+----+--------+----------------------+
+----+--------+------------------------+
| id | amount | exp_value              |
+----+--------+------------------------+
|  1 | 200.50 |  1.3913616653030070e87 |
|  2 | 150.40 |  2.0791783737773754e65 |
|  3 | 300.00 | 1.9424263952412555e130 |
|  4 | 120.60 |   2.3763724976530032e52 |
|  5 | 180.00 |  1.4893842007818583e78 |
+----+--------+------------------------+
+----+--------+-------------+
| id | amount | floor_value |
+----+--------+-------------+
|  1 | 200.50 |         200 |
|  2 | 150.40 |         150 |
|  3 | 300.00 |         300 |
|  4 | 120.60 |         120 |
|  5 | 180.00 |         180 |
+----+--------+-------------+
+----+--------+--------------------+
| id | amount | log_value          |
```



```sql
CREATE TABLE Sales (
    id INT PRIMARY KEY,
    amount DECIMAL(10, 2),
    discount DECIMAL(5, 2),
    quantity INT
);
INSERT INTO Sales (id, amount, discount, quantity)
VALUES
(1, 200.50, 10.00, 5),
(2, 150.40, 15.00, 2),
(3, 300.00, 30.00, 3),
(4, 120.00, 5.00, 7),
(5, 180.00, 20.00, 4);
SELECT
    id,
    discount,
    ABS(discount) AS absolute_discount
FROM Sales;

SELECT
    id,
    amount,
    ACOS(0.5) AS acos_value -- This would return the arc cosine of 0.5
FROM Sales;

SELECT
    id,
    amount,
    ATAN(amount) AS atan_value
FROM Sales;

SELECT
    id,
    amount,
    CEILING(amount) AS ceiling_value
From Sales;

SELECT
    id,
    amount,
    COS(amount) AS cosine_value
FROM Sales;

SELECT
    id,
    amount,
    EXP(amount) AS exp_value
FROM Sales;
```

```
+----+----------+-------------------+
| id | discount | absolute_discount |
+----+----------+-------------------+
|  1 |    10.00 |             10.00 |
|  2 |    15.00 |             15.00 |
|  3 |    30.00 |             30.00 |
|  4 |     5.00 |              5.00 |
|  5 |    20.00 |             20.00 |
+----+----------+-------------------+
+----+--------+--------------------+
| id | amount | acos_value         |
+----+--------+--------------------+
|  1 | 200.50 | 1.0471975511965979 |
|  2 | 150.40 | 1.0471975511965979 |
|  3 | 300.00 | 1.0471975511965979 |
|  4 | 120.00 | 1.0471975511965979 |
|  5 | 180.00 | 1.0471975511965979 |
+----+--------+--------------------+
+----+--------+--------------------+
| id | amount | atan_value         |
+----+--------+--------------------+
|  1 | 200.50 | 1.5658088369779122 |
|  2 | 150.40 |  1.564147480868158 |
|  3 | 300.00 |   1.56746300580716 |
|  4 | 120.00 | 1.5629668328589536 |
|  5 | 180.00 |  1.565240828394284 |
+----+--------+--------------------+
+----+--------+---------------+
| id | amount | ceiling_value |
+----+--------+---------------+
|  1 | 200.50 |           201 |
```

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)
LOG10(n)
	Returns the natural logarithm (base e) of
	n Returns the base-10 logarithm of n

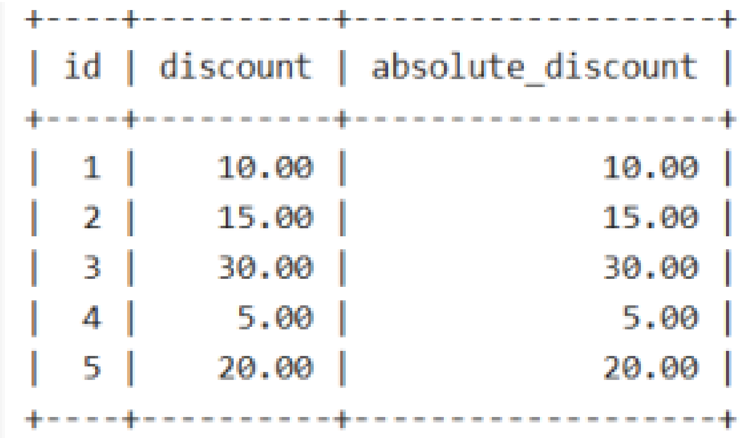
MAX(expression)	Returns the maximum value in a column
MIN(expression)	Returns the minimum value in a column
PI()	Returns the value of π (pi)

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

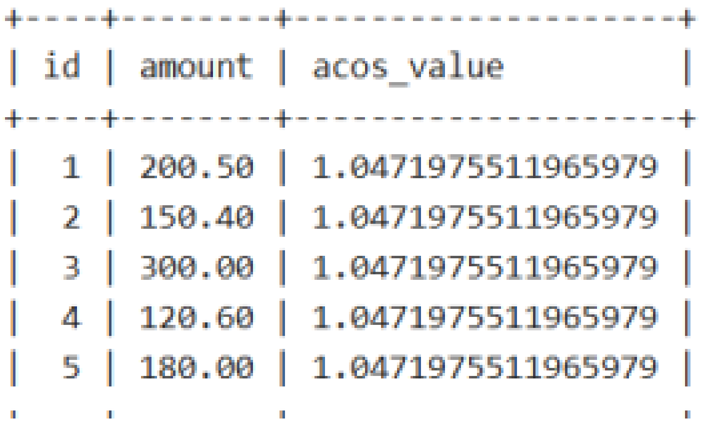
2.1 Absolute Value (ABS)



```
+----+----------+-------------------+
| id | discount | absolute_discount |
+----+----------+-------------------+
|  1 |    10.00 |             10.00 |
|  2 |    15.00 |             15.00 |
|  3 |    30.00 |             30.00 |
|  4 |     5.00 |              5.00 |
|  5 |    20.00 |             20.00 |
+----+----------+-------------------+
```

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

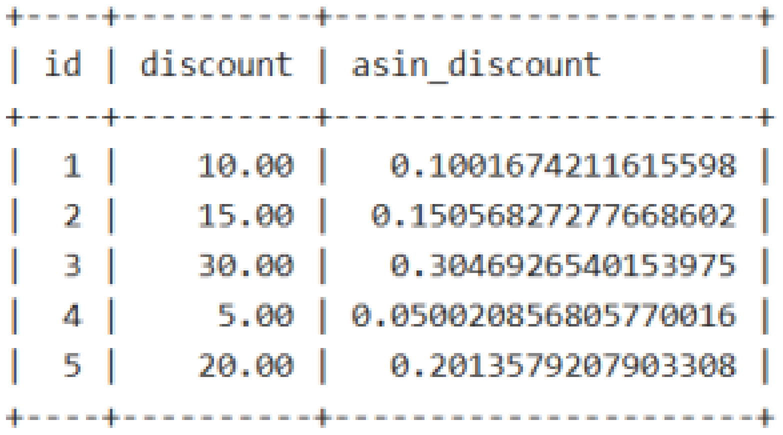
2.2 Arc Cosine (ACOS)



```
+----+--------+--------------------+
| id | amount | acos_value         |
+----+--------+--------------------+
|  1 | 200.50 | 1.0471975511965979 |
|  2 | 150.40 | 1.0471975511965979 |
|  3 | 300.00 | 1.0471975511965979 |
|  4 | 120.60 | 1.0471975511965979 |
|  5 | 180.00 | 1.0471975511965979 |
```

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

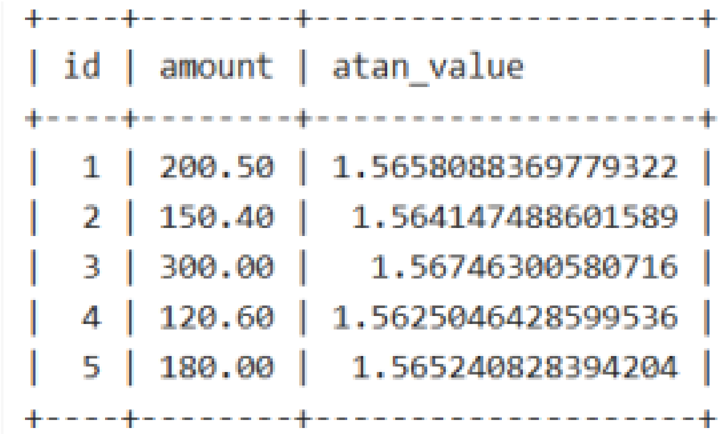
2.3 Arc Sine (ASIN)



```
+----+----------+----------------------+
| id | discount | asin_discount        |
+----+----------+----------------------+
|  1 |    10.00 |   0.1001674211615598 |
|  2 |    15.00 |  0.15056827277668602 |
|  3 |    30.00 |   0.3046926540153975 |
|  4 |     5.00 | 0.050020856805770016 |
|  5 |    20.00 |   0.2013579207903308 |
+----+----------+----------------------+
```

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

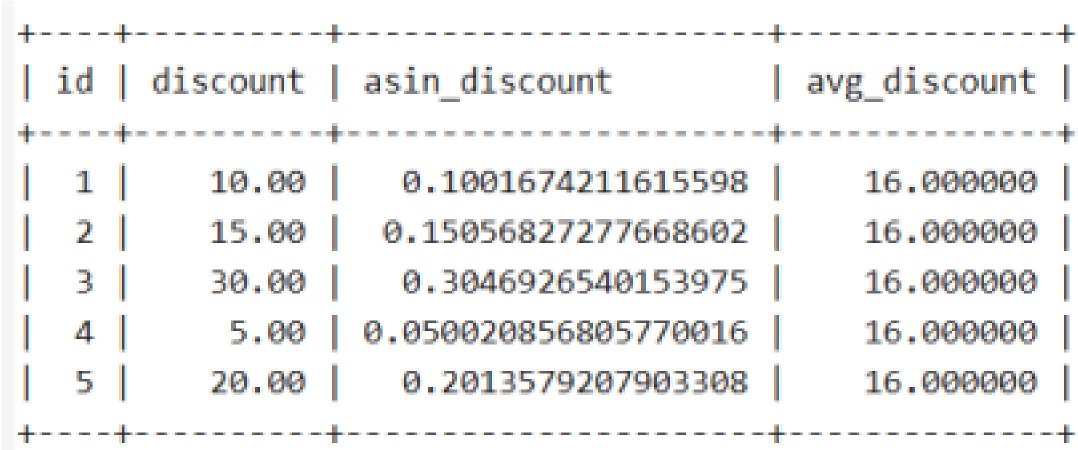
2.4 Arc Tangent (ATAN)



```
+----+--------+--------------------+
| id | amount | atan_value         |
+----+--------+--------------------+
|  1 | 200.50 | 1.5658088369779322 |
|  2 | 150.40 |  1.564147488601589 |
|  3 | 300.00 |   1.56746300580716 |
|  4 | 120.60 | 1.5625046428599536 |
|  5 | 180.00 |  1.565240828394204 |
+----+--------+--------------------+
```

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

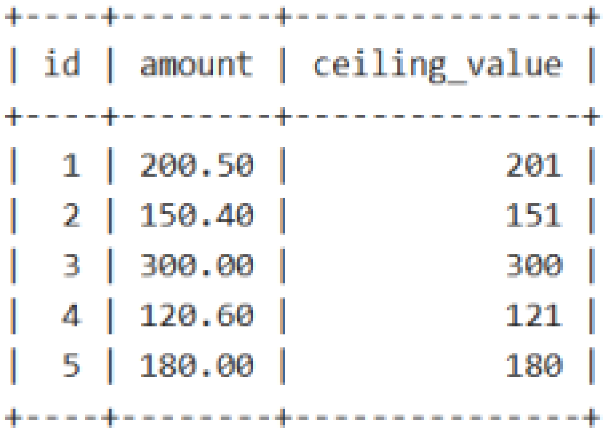
2.5 Average (AVG)



```
+----+----------+----------------------+--------------+
| id | discount | asin_discount        | avg_discount |
+----+----------+----------------------+--------------+
|  1 |    10.00 |   0.1001674211615598 |    16.000000 |
|  2 |    15.00 |  0.15056827277668602 |    16.000000 |
|  3 |    30.00 |   0.3046926540153975 |    16.000000 |
|  4 |     5.00 | 0.050020856805770016 |    16.000000 |
|  5 |    20.00 |   0.2013579207903308 |    16.000000 |
+----+----------+----------------------+--------------+
```

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

2.6 Ceiling (CEILING)



```
+----+--------+---------------+
| id | amount | ceiling_value |
+----+--------+---------------+
|  1 | 200.50 |           201 |
|  2 | 150.40 |           151 |
|  3 | 300.00 |           300 |
|  4 | 120.60 |           121 |
|  5 | 180.00 |           180 |
+----+--------+---------------+
```

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

2.7 Count (COUNT)

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

2.8 Cosine (COS)

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

2.9 Cotangent (COT)

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

2.10 Convert Radians to Degrees (DEGREES)

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

2.11 Exponential (EXP)

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

2.12 Floor (FLOOR)

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

2.13 Natural Logarithm (LOG)

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

2.14 Logarithm Base 10 (LOG10)

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

2.15 Maximum (MAX)

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

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)

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

1️⃣Financial Analytics: Compound Interest Calculati 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×(1+r/n)(nt)FV = P \times (1 + r/n)^{(nt)}FV=P×(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

2️⃣Customer Segmentation: Standard Deviation 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

3️⃣Real-Time Performance Monitoring: CPU Load Analys

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

4️⃣Fraud Detection: Identifying Abnormal Transactio

Use Case: Detect transactions that are significantly **higher** than the usu 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:

transaction_id	**customer_id**	**amount**	**avg_amount**	**stddev_amount**
90872	201	12000	3000	4500

�� If a customer usually spends $3,000 ± $4,500, a $12,000 transaction is flagged as suspiciou

5️⃣Logistics: Estimating Delivery Time Based on Distan

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

6️⃣Astronomy/Physics: Calculating Earthquake Magnitu (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=log10(A)+3M = \log_{10} (A) + 3M=log10(A)+3

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

Result Example:

earthquake_id	station_id	amplitude	magnitude
EQ001	ST1001	5000	6.70

7️⃣E-Commerce: Personalized Discount Calculati 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