Chapter: Unlocking Maths Prowess

Topic: Mathematical Functions in SQL

Section 1: Learn

1.1 Introduction to Mathematical Functions in SQL

SQL is not just a query language — it can also perform **mathematical computations** directly on your database columns. These computations can be used for:

- Generating totals, discounts, or averages
- Performing analytics or statistical calculations
- Formatting and controlling numeric precision in reports

SQL supports both **basic arithmetic** and **advanced math functions** to handle day-to-day data processing needs.

1.2 Basic Arithmetic Operations

You can use arithmetic operators directly in your SQL queries.

Operators:

- `+` : Addition
- `-` : Subtraction
- `*` : Multiplication
- `/` : Division
- `%` : Modulus (remainder)

Example:

```
SELECT price, quantity, price * quantity AS total_cost
FROM products;
```

> Multiplies price and quantity to get the total cost per item.

1.3 ROUND(): Rounding to Specific Decimal Places

Purpose:

Rounds a number to the nearest value based on decimal position.

Syntax:

```
ROUND(number, decimal_places)
```

Example:

```
SELECT ROUND(56.789, 2); -- Result: 56.79
```

> Useful for monetary values.

1.4 CEIL() and FLOOR(): Round Up or Down

CEIL(): Always rounds up to the next whole number.

FLOOR(): Always rounds down to the previous whole number.

Example:

```
SELECT CEIL(5.2), FLOOR(5.8); -- Output: 6 and 5
```

> Billing and resource allocation often use CEIL to avoid undercharging.

1.5 ABS(): Absolute Value

Returns the **non-negative version** of a number.

Example:

```
SELECT ABS(-120); -- Output: 120
```

> Used in financial reports to represent values positively.

1.6 POWER() and SQRT(): Exponentiation and Square Root

POWER(): Raises a number to a given power.

```
SELECT POWER(4, 3); -- Output: 64
```

SQRT(): Calculates the square root of a number.

```
SELECT SQRT(81); -- Output: 9
```

1.7 MOD(): Remainder After Division

Returns the remainder when one number is divided by another.

```
SELECT MOD(11, 4); -- Output: 3
```

> Used to check for even/odd numbers or cyclic patterns.

1.8 SIGN(): Detect Number Sign

Returns:

- 1 for positive

- 0 for zero
- -1 for negative

```
SELECT SIGN(-200); -- Output: -1
```

1.9 TRUNCATE(): Remove Decimal Digits Without Rounding

Syntax:

```
TRUNCATE(number, decimal_places)
```

Example:

```
SELECT TRUNCATE(78.9876, 2); -- Output: 78.98
```

> Keeps fixed decimal values, ideal for tax or accounting systems.

Section 2: Practise

Exercise 1: Calculate Total Cost

```
SELECT item_name, quantity, price, quantity * price AS total_cost
FROM invoice;
```

Exercise 2: Round Discounted Prices

```
SELECT item_name, ROUND(price * 0.85, 2) AS discounted_price
FROM products;
```

Exercise 3: Use CEIL and FLOOR for Allocation

```
SELECT student_name, CEIL(marks/10) AS grade_band
FROM marksheet;
```

Exercise 4: Normalize Values to Positive

```
SELECT transaction_id, ABS(amount) AS normalized_amount
FROM transactions;
```

Exercise 5: Use POWER to Calculate Area of a Square

```
SELECT side_length, POWER(side_length, 2) AS area
FROM square_table;
```

Exercise 6: Find Sign of Revenue Growth

```
SELECT branch_name, revenue_2024 - revenue_2023 AS growth,
    SIGN(revenue_2024 - revenue_2023) AS growth_direction
FROM branches;
```

Section 3: FAQ – Know More

Q1. How is TRUNCATE different from ROUND()?

- ROUND() modifies the number based on rounding rules.
- TRUNCATE() just cuts off digits, no rounding.

Q2. What happens when dividing by zero in SQL?

It causes a runtime error. You should always check before dividing.

```
SELECT CASE
        WHEN divisor = 0 THEN 'Invalid'
        ELSE dividend / divisor
      END AS result
FROM calc_data;
```

Q3. How can I generate a random number?

Use RAND() (in MySQL):

```
SELECT RAND();  -- Returns a number between 0 and 1
```

Q4. Can I combine multiple mathematical functions in one query?

Yes, you can nest functions or use them in expressions.

```
SELECT ROUND(SQRT(price * quantity), 2) AS result
FROM products;
```

Q5. Are mathematical functions usable in WHERE clauses?

Absolutely. For example:

```
SELECT * FROM transactions
WHERE ABS(amount) > 1000;
```

End of Notes for Chapter: Unlocking Maths Prowess