Below are detailed descriptions along with two code examples for each of the three categories: DATE functions, TIME functions, and INTERVALs.

### 1. DATE Functions:

#### a. CURRENT_DATE Function:
The `CURRENT_DATE` function returns the current date.

Example 1:
```sql
SELECT CURRENT_DATE AS current_date;
```

Example 2:
```sql
-- Insert a record with the current date
INSERT INTO orders (order_date) VALUES (CURRENT_DATE);
```

#### b. DATE_ADD Function:
The `DATE_ADD` function is used to add a specified number of days to a date.

Example 1:
```sql
-- Add 7 days to the order_date
SELECT DATE_ADD(order_date, INTERVAL 7 DAY) AS new_date FROM orders;
```

Example 2:
```sql
-- Update the order_date by adding 30 days
UPDATE orders SET order_date = DATE_ADD(order_date, INTERVAL 30 DAY) WHERE
order_id = 1;
```

### 2. TIME Functions:

#### a. CURRENT_TIME Function:
The `CURRENT_TIME` function returns the current time.

Example 1:
```sql
SELECT CURRENT_TIME AS current_time;
```

Example 2:
```sql
-- Insert a record with the current time
INSERT INTO log (log_time) VALUES (CURRENT_TIME);
```

#### b. EXTRACT Function:
The `EXTRACT` function is used to extract a part of a time, such as hours, minutes, or seconds.

Example 1:
```sql
-- Extract the hour from the log_time
SELECT EXTRACT(HOUR FROM log_time) AS hour FROM log;
```

Example 2:
```sql
-- Calculate the average minutes spent on a task
SELECT AVG(EXTRACT(MINUTE FROM task_duration)) AS avg_minutes FROM tasks;
```

### 3. INTERVALs:

#### a. Adding INTERVAL to DATE:
The `INTERVAL` keyword is used to specify a duration to add to a date.

Example 1:
```sql
-- Add 3 months to the subscription expiration date
SELECT subscription_expiration + INTERVAL 3 MONTH AS new_expiration FROM subscriptions;
```

Example 2:
```sql
-- Find orders placed within the last week
SELECT * FROM orders WHERE order_date >= (CURRENT_DATE - INTERVAL 7 DAY);
```

#### b. Calculating Time Difference with INTERVAL:
The `INTERVAL` can be used to calculate the time difference between two timestamps.

Example 1:

```sql
-- Calculate the age of a user in years
SELECT user_id, CURRENT_DATE - birthdate AS age FROM users;
```

Example 2:
```sql
-- Find tasks that took more than 2 hours to complete
SELECT * FROM tasks WHERE task_duration > INTERVAL 2 HOUR;
```

These examples illustrate common use cases for working with dates, times, and intervals in SQL. Keep in mind that specific syntax may vary depending on the SQL database system you are using (e.g., MySQL, PostgreSQL, SQL Server).

Certainly! Let's explore additional examples for each category:

### 1. DATE Functions:

#### c. DATE_FORMAT Function:
The `DATE_FORMAT` function is used to format a date as a string.

Example 1:
```sql
-- Display the order_date in 'YYYY-MM-DD' format
SELECT DATE_FORMAT(order_date, '%Y-%m-%d') AS formatted_date FROM orders;
```

Example 2:
```sql
-- Insert a record with the current date in 'Month Day, Year' format
INSERT INTO events (event_date) VALUES (DATE_FORMAT(CURRENT_DATE, '%M %e, %Y'));
```

#### d. DATEDIFF Function:
The `DATEDIFF` function calculates the difference between two dates.

Example 1:
```sql
-- Calculate the number of days between order_date and today
SELECT DATEDIFF(CURRENT_DATE, order_date) AS days_since_ordered FROM orders;
```

Example 2:
```sql
-- Find users whose accounts are about to expire in the next 10 days
SELECT * FROM users WHERE DATEDIFF(account_expiration_date, CURRENT_DATE) <=
10;
```

### 2. TIME Functions:

#### c. ADDTIME Function:
The `ADDTIME` function adds a time value to a time or datetime value.

Example 1:
```sql
-- Add 2 hours and 30 minutes to the log_time
SELECT ADDTIME(log_time, '02:30:00') AS updated_time FROM log;
```

Example 2:
```sql
-- Update task end time by adding 1 hour
UPDATE tasks SET end_time = ADDTIME(end_time, '01:00:00') WHERE task_id = 2;
```

#### d. TIME_TO_SEC Function:
The `TIME_TO_SEC` function converts a time value to seconds.

Example 1:
```sql
-- Calculate the total seconds spent on a task
SELECT task_id, TIME_TO_SEC(task_duration) AS total_seconds FROM tasks;
```

Example 2:
```sql
-- Find tasks that took more than 3 hours (10,800 seconds) to complete
SELECT * FROM tasks WHERE TIME_TO_SEC(task_duration) > 10800;
```

### 3. INTERVALs:

#### c. Using INTERVAL with JOIN:
The `INTERVAL` can be used in JOIN conditions to filter records based on time intervals.

Example 1:
```sql
-- Join orders with promotions that were active 30 days ago
SELECT *
FROM orders
JOIN promotions ON orders.order_date BETWEEN promotions.start_date AND
promotions.start_date + INTERVAL 30 DAY;
```

Example 2:
```sql
-- Find users whose accounts were created within the last 6 months
SELECT * FROM users WHERE account_creation_date >= (CURRENT_DATE - INTERVAL 6
MONTH);
```

#### d. TIMESTAMPDIFF Function:
The `TIMESTAMPDIFF` function calculates the difference between two timestamps.

Example 1:
```sql
-- Calculate the age of a product in months
SELECT product_id, TIMESTAMPDIFF(MONTH, production_date, CURRENT_DATE) AS
age_in_months FROM products;
```

Example 2:
```sql
-- Find tasks that took more than 1 day to complete
SELECT * FROM tasks WHERE TIMESTAMPDIFF(DAY, start_time, end_time) > 1;
```

These additional examples demonstrate more advanced scenarios for working with dates, times, and intervals in SQL. As always, adapt the syntax based on the specific database system you are using.