```python
import logging
import sqlite3
import time
import psutil

# Configuration du logging
logging.basicConfig(filename='monitoring.log', level=logging.INFO, format='%(asctime)s - %(levelname)s -
%(message)s')

# Function to create and populate the CPU table
def create_cpu_table(conn):
    """
    Creates a table named 'cpu' in the given SQLite connection if it doesn't exist already,
    with columns for id (primary key), timestamp, and cpu_percent.

    Args:
        conn (sqlite3.Connection): SQLite database connection object.
    """
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS cpu (
                        id INTEGER PRIMARY KEY,
                        timestamp TIMESTAMP,
                        cpu_percent REAL
                    )''')
    conn.commit()

def insert_cpu_data(conn, cpu_percent):
    """
    Inserts CPU data (timestamp and CPU percentage) into the 'cpu' table.

    Args:
        conn (sqlite3.Connection): SQLite database connection object.
        cpu_percent (float): CPU percentage value.
    """
    cursor = conn.cursor()
    cursor.execute('''INSERT INTO cpu (timestamp, cpu_percent) VALUES (?, ?)''', (int(time.time()), cpu_percent))
    conn.commit()

# Function to create and populate the Memory table
def create_memory_table(conn):
```

```
    """
    Creates a table named 'memory' in the given SQLite connection if it doesn't exist already,
    with columns for id (primary key), timestamp, total_memory, available_memory, used_memory, and
free_memory.

    Args:
        conn (sqlite3.Connection): SQLite database connection object.
    """
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS memory (
                        id INTEGER PRIMARY KEY,
                        timestamp TIMESTAMP,
                        total_memory REAL,
                        available_memory REAL,
                        used_memory REAL,
                        free_memory REAL
                    )''')
    conn.commit()

def insert_memory_data(conn):
    """
    Inserts memory data (timestamp, total, available, used, and free memory) into the 'memory' table.

    Args:
        conn (sqlite3.Connection): SQLite database connection object.
    """
    memory = psutil.virtual_memory()
    cursor = conn.cursor()
    cursor.execute('''INSERT INTO memory (timestamp, total_memory, available_memory, used_memory,
free_memory)
                      VALUES (?, ?, ?, ?, ?)''', (int(time.time()), memory.total, memory.available, memory.used,
memory.free))
    conn.commit()

# Function to create and populate the Network table
def create_network_table(conn):
    """
    Creates a table named 'network' in the given SQLite connection if it doesn't exist already,
    with columns for id (primary key), timestamp, bytes_sent, and bytes_recv.
```

```
    Args:
        conn (sqlite3.Connection): SQLite database connection object.
    """
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS network (
                        id INTEGER PRIMARY KEY,
                        timestamp TIMESTAMP,
                        bytes_sent REAL,
                        bytes_recv REAL
                    )''')
    conn.commit()

def insert_network_data(conn):
    """
    Inserts network data (timestamp, bytes_sent, and bytes_recv) into the 'network' table.

    Args:
        conn (sqlite3.Connection): SQLite database connection object.
    """
    network = psutil.net_io_counters()
    cursor = conn.cursor()
    cursor.execute('''INSERT INTO network (timestamp, bytes_sent, bytes_recv) VALUES (?, ?, ?)''',
(int(time.time()), network.bytes_sent, network.bytes_recv))
    conn.commit()

# Establish SQLite database connection
conn = sqlite3.connect('monitoring.db')

# Create tables if they do not exist already
create_cpu_table(conn)
create_memory_table(conn)
create_network_table(conn)

try:
    while True:
        # Data collection
        # CPU
        cpu_percent = psutil.cpu_percent()
        cpu_time = psutil.cpu_times_percent()
        cpu_load_av = psutil.getloadavg
```

```
        # Memory
        virtual_memory = psutil.virtual_memory()
        swap_memory = psutil.swap_memory()

        # Network
        network_info = psutil.net_io_counters()

        # Print collected data
        print(f"CPU Info:\t{cpu_percent}")
        print(f"Memory Info:\t{virtual_memory}")
        print(f"Network Info:\t{network_info}")

        # Insert data into the database
        insert_cpu_data(conn, cpu_percent)
        insert_memory_data(conn)
        insert_network_data(conn)

        # Log the data
        logging.info(f"CPU Percent:\t{cpu_percent}%")
        logging.info(f"Memory Info:\t{virtual_memory}")
        logging.info(f"Network Info:\t{network_info}")

        time.sleep(10)

except KeyboardInterrupt:
    conn.close()
```