```python
import sqlite3

# Підключення до бази
conn = sqlite3.connect("database.db")
cursor = conn.cursor()

# Створення таблиці, якщо вона ще не існує
cursor.execute('''
    CREATE TABLE IF NOT EXISTS medals (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        medal_type TEXT NOT NULL,
        count INTEGER NOT NULL,
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    )
''')

# Збереження змін
conn.commit()
conn.close()

print("✅ Таблиця 'medals' створена успішно!")
```

2-3.

```python
import random
from prefect import task, flow

# Завдання: випадково вибирає тип медалі
@task
def pick_medal():
    medals = ["Bronze", "Silver", "Gold"]
    chosen_medal = random.choice(medals)
    print(f"🎖️ Вибрано медаль: {chosen_medal}")
    return chosen_medal

# Потік Prefect
@flow
def medal_selection_flow():
    medal = pick_medal()
    print(f"🏅 Обрана медаль: {medal}")

if __name__ == "__main__":
    medal_selection_flow()
```

4-5-6.

```python
import sqlite3
import random
import time
from datetime import datetime
from prefect import flow, task
```

```python
DATABASE_NAME = "database.db"


@task
def pick_medal():
    """Випадковий вибір медалі."""
    medal = random.choice(["Gold", "Silver", "Bronze"])
    print(f"🏅 Вибрано медаль: {medal}")
    return medal


@task
def insert_medal(medal):
    """Додає запис у базу даних."""
    conn = sqlite3.connect(DATABASE_NAME)
    cursor = conn.cursor()

    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    cursor.execute("INSERT INTO medals (medal_type, count, created_at) VALUES (?, ?, ?)", (medal, 1, timestamp))
    conn.commit()

    print(f"✅ Додано запис: {medal} ({timestamp})")
    conn.close()


@task
def count_medals(medal):
    """Рахує кількість медалей заданого типу."""
    conn = sqlite3.connect(DATABASE_NAME)
    cursor = conn.cursor()

    cursor.execute("SELECT COUNT(*) FROM medals WHERE medal_type = ?", (medal,))
    count = cursor.fetchone()[0]

    print(f"📊 В таблиці знайдено {count} медалей типу {medal}")
    conn.close()


@task
def delay_execution():
    """Затримка у 35 секунд перед перевіркою."""
    print("⏳ Чекаємо 35 секунд перед перевіркою...")
    time.sleep(35)
    print("✅ Затримка завершена!")


@task
def check_latest_entry():
    """Перевіряє, чи останній запис у базі не старший за 30 секунд."""
    conn = sqlite3.connect(DATABASE_NAME)
```

```
    cursor = conn.cursor()

    cursor.execute("SELECT created_at FROM medals ORDER BY created_at DESC
LIMIT 1")
    last_entry = cursor.fetchone()

    if last_entry:
        last_time = datetime.strptime(last_entry[0], "%Y-%m-%d %H:%M:%S")
        time_diff = (datetime.now() - last_time).total_seconds()

        if time_diff > 30:
            print(f"❌ Помилка: останній запис у базі старіший за 30 секунд
({time_diff:.2f} сек)")
            raise Exception("Занадто старий запис у базі")  # Зупиняє
виконання потоку
        else:
            print(f"✅ Останній запис у базі в межах 30 секунд
({time_diff:.2f} сек)")
    else:
        print("⚠️ Увага: У базі немає жодного запису!")
        raise Exception("Немає записів у базі")  # Зупиняє виконання потоку

    conn.close()


@flow
def medal_branching_flow():
    """Основний потік, що виконує логіку вибору медалі, запису, підрахунку
та перевірки часу."""
    medal = pick_medal()
    insert_medal(medal)
    count_medals(medal)
    delay_execution()
    check_latest_entry()  # Додаємо перевірку часу


if __name__ == "__main__":
    medal_branching_flow()
```