The first step is to create the SQLite database using the following command:

sh

```
sqlite3 loop.db < schema.sql
```

The schema.sql file contains the SQL code to create the necessary tables for our application.

sql

```
CREATE TABLE stores (id INTEGER PRIMARY KEY, timezone_str TEXT DEFAULT 'America/Chicago');

CREATE TABLE business_hours (  id INTEGER PRIMARY KEY,store_id INTEGER,day_of_week INTEGER,
start_time_local TEXT, end_time_local TEXT, FOREIGN KEY(store_id) REFERENCES stores(id));

CREATE TABLE store_status (id INTEGER PRIMARY KEY,store_id INTEGER, timestamp_utc TEXT, status TEXT,

  FOREIGN KEY(store_id) REFERENCES stores(id));
```

Next, we will use SQLAlchemy to interact with the database in our Flask application.

python

```
from flask import Flask, jsonify, request

from flask_sqlalchemy import SQLAlchemy

from datetime import datetime, timedelta

from pytz import timezone

import pandas as pd

app = Flask(__name__)

app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///loop.db'

db = SQLAlchemy(app)

elclass Store(db.Model):

    __tablename__ = 'stores'

    id = db.Column(db.Integer, primary_key=True)

    timezone_str = db.Column(db.String(50), default='America/Chicago')


class BusinessHour(db.Model):
    __tablename__ = 'business_hours'
    id = db.Column(db.Integer, primary_key=True)
    store_id = db.Column(db.Integer, db.ForeignKey('stores.id'))
    day_of_week = db.Column(db.Integer)
    start_time_local = db.Column(db.String(10))
    end_time_local = db.Column(db.String(10))
    store = db.relationship('Store', backref=db.backref('business_hours', lazy=True))

    class StoreStatus(db.Model):
    __tablename__ = 'store_status'
    id = db.Column(db.Integer, primary_key=True)
    store_id = db.Column(db.Integer, db.ForeignKey('stores.id'))
    timestamp_utc = db.Column(db.String(25))
    status = db.Column(db.String(10))
    store = db.relationship('Store', backref=db.backref('store_status', lazy=True))
def get_store_timezone(store_id):
    store = Store.query.filter_by(id=store_id).first()
    if store:
        return timezone(store.timezone_str)
    else:
        return timezone('America/Chicago')
def get_business_hours(store_id):
    business_hours = BusinessHour.query.filter_by(store_id=store_id).all()
    return business_hours
def get_store_status(store_id, start_time, end_time):
    store_status = StoreStatus.query.filter_by(store_id=store_id).\
        filter(StoreStatus.timestamp_utc >= start_time).\
        filter(StoreStatus.timestamp_utc < end_time).\
        all()
    return store_status
def get_time_ranges(start_time, end_time):


    Split time range into hourly intervals
    time_ranges = []
    curr_time = start_time
    while curr_time < end_time:
        next_time = curr_time + timedelta(hours=1)
        if next_time > end_time:
            next_time = end_time
        time_ranges.append((curr_time, next_time))
        curr_time = next_time
    return time_ranges


def calculate_uptime_downtime(store_id
```