```
from pathlib import Path
import os
import sqlite3

import s3fs
import pandas as pd

current_dir = Path(os.getcwd()).absolute()
results_dir = current_dir.joinpath('results')
kv_data_dir = results_dir.joinpath('kvdb')
kv_data_dir.mkdir(parents=True, exist_ok=True)


def read_cluster_csv(file_path, endpoint_url='https://storage.budsc.midwest-datasci
    s3 = s3fs.S3FileSystem(
        anon=True,
        client_kwargs={
            'endpoint_url': endpoint_url
        }
    )
    return pd.read_csv(s3.open(file_path, mode='rb'))

# Define the base path for your S3 data
base_path = 'C:/Users/PN174MM/OneDrive - EY/Desktop/Personal Projects/650/dsc650/da
def read_local_csv(file_path):
    return pd.read_csv(file_path)
```

Create and Load Measurements Table

```
def create_measurements_table(conn):
    sql = """
    CREATE TABLE IF NOT EXISTS measurements (
        visit_id integer NOT NULL,
        person_id text NOT NULL,
        quantity text,
        reading real,
        FOREIGN KEY (visit_id) REFERENCES visits (visit_id),
        FOREIGN KEY (person_id) REFERENCES people (people_id)
        );
    """

    c = conn.cursor()
    c.execute(sql)

def load_measurements_table(conn):
    create_measurements_table(conn)
    df = read_local_csv(base_path + 'measurements.csv')
```

```
    df = read_local_csv(base_path + 'measurements.csv')
    measurements_data = df.values
    c = conn.cursor()
    c.execute('DELETE FROM measurements;')
    c.executemany('INSERT INTO measurements VALUES (?,?,?,?)', measurements_data)
    conn.commit()  # Commit changes to release the lock
```

Create and Load People Table

```
def create_people_table(conn):
    sql = """
    CREATE TABLE IF NOT EXISTS person (
        person_id text PRIMARY KEY,
        personal_name text NOT NULL,
        family_name text NOT NULL
    );
    """
    c = conn.cursor()
    c.execute(sql)


def load_people_table(conn):
    create_people_table(conn)
    df = read_local_csv(base_path + 'person.csv')
    person_data = df.values
    c = conn.cursor()
    c.execute('DELETE FROM person;')
    c.executemany('INSERT INTO person VALUES (?,?,?)', person_data)
    conn.commit()  # Commit changes to release the lock
```

Create and Load Sites Table

```
def create_sites_table(conn):
    sql = """
    CREATE TABLE IF NOT EXISTS sites (
        site_id text PRIMARY KEY,
        latitude double NOT NULL,
        longitude double NOT NULL
    );
    """

    c = conn.cursor()
    c.execute(sql)
```

```
def load_sites_table(conn):
    create_sites_table(conn)
    df = read_local_csv(base_path + 'site.csv')
    site_data = df.values
    c = conn.cursor()
    c.execute('DELETE FROM sites;')
    c.executemany('INSERT INTO sites VALUES (?,?,?)', site_data)
    conn.commit()  # Commit changes to release the lock
```

Create and Load Visits Table

```
def create_visits_table(conn):
    sql = """
    CREATE TABLE IF NOT EXISTS visits (
        visit_id integer PRIMARY KEY,
        site_id text NOT NULL,
        visit_date text,
        FOREIGN KEY (site_id) REFERENCES sites (site_id)
        );
    """

    c = conn.cursor()
    c.execute(sql)

def load_visits_table(conn):
    create_visits_table(conn)
    df = read_local_csv(base_path + 'visited.csv')
    visits_data = df.values
    c = conn.cursor()
    c.execute('DELETE FROM visits;')
    c.executemany('INSERT INTO visits (visit_id, site_id, visit_date) VALUES (?,?,?)', visits
    conn.commit()  # Commit changes to release the lock
```

Create DB and Load Tables

```
db_path = results_dir.joinpath('patient-info.db')
conn = sqlite3.connect(str(db_path))
# TODO: Uncomment once functions completed
load_people_table(conn)
load_sites_table(conn)
load_visits_table(conn)
load_measurements_table(conn)
```

```
conn.commit()
conn.close()
```