```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 5,
   "metadata": {
    "pycharm": {
     "name": "#%%\n"
    }
   },
   "outputs": [],
   "source": [
    "from pathlib import Path\n",
    "import os\n",
    "import sqlite3\n",
    "\n",
    "import s3fs\n",
    "import pandas as pd\n",
    "\n",
    "current_dir = Path(os.getcwd()).absolute()\n",
    "results_dir = current_dir.joinpath('results')\n",
    "kv_data_dir = results_dir.joinpath('kvdb')\n",
    "kv_data_dir.mkdir(parents=True, exist_ok=True)\n",
    "\n",
    "\n",
    "def read_cluster_csv(file_path, endpoint_url='https://storage.budsc.midwest-datascience.com'):\n",
    "    s3 = s3fs.S3FileSystem(\n",
    "        anon=True,\n",
    "        client_kwargs={\n",
    "            'endpoint_url': endpoint_url\n",
    "        }\n",
    "    )\n",
    "    return pd.read_csv(s3.open(file_path, mode='rb'))"
   ]
  },
  {
   "cell_type": "markdown",
```

```
38    "metadata": {
39     "pycharm": {
40      "name": "#%% md\n"
41     }
42    },
43    "source": [
44     "## Create and Load Measurements Table"
45    ]
46   },
47   {
48    "cell_type": "code",
49    "execution_count": 6,
50    "metadata": {
51     "pycharm": {
52      "name": "#%%\n"
53     }
54    },
55    "outputs": [],
56    "source": [
57     "def create_measurements_table(conn):\n",
58     "    sql = \"\"\"\n",
59     "    CREATE TABLE IF NOT EXISTS measurements (\n
   ",
60     "        visit_id integer NOT NULL,\n",
61     "        person_id text NOT NULL,\n",
62     "        quantity text,\n",
63     "        reading real,\n",
64     "        FOREIGN KEY (visit_id) REFERENCES visits
    (visit_id),\n",
65     "        FOREIGN KEY (person_id) REFERENCES
   people (people_id)\n",
66     "        );\n",
67     "    \"\"\"\n",
68     "\n",
69     "    c = conn.cursor()\n",
70     "    c.execute(sql)\n",
71     "    \n",
72     "def load_measurements_table(conn):\n",
73     "    create_measurements_table(conn)\n",
74     "    df = pd.read_csv('C:/Users/taylo/OneDrive/
   Documents/dsc650/data/external/tidynomicon/
```

```
 74 measurements.csv')\n",
 75     "    measurements = df.values\n",
 76     "    c = conn.cursor()\n",
 77     "    c.execute('DELETE FROM measurements;') #
    Delete data if exists\n",
 78     "    c.executemany('INSERT INTO measurements
    VALUES (?,?,?,?)', measurements)"
 79    ]
 80   },
 81   {
 82    "cell_type": "markdown",
 83    "metadata": {
 84     "pycharm": {
 85      "name": "#%% md\n"
 86     }
 87    },
 88    "source": [
 89     "## Create and Load People Table"
 90    ]
 91   },
 92   {
 93    "cell_type": "code",
 94    "execution_count": 7,
 95    "metadata": {
 96     "pycharm": {
 97      "name": "#%%\n"
 98     }
 99    },
100    "outputs": [],
101    "source": [
102     "def create_people_table(conn):\n",
103     "    sql = \"\"\"\n",
104     "        CREATE TABLE IF NOT EXISTS people (\n",
105     "            person_id text PRIMARY KEY,\n",
106     "            personal_name text,\n",
107     "            family_name text\n",
108     "        );\n",
109     "    \"\"\"\n",
110     "    ## TODO: Complete SQL\n",
111     "    c = conn.cursor()\n",
112     "    c.execute(sql)\n",
```

```
113     "    \n",
114     "def load_people_table(conn):\n",
115     "    create_people_table(conn)\n",
116     "    ## TODO: Complete code\n",
117     "    df = pd.read_csv('C:/Users/taylo/OneDrive/
    Documents/dsc650/data/external/tidynomicon/person.
    csv')\n",
118     "    people = df.values\n",
119     "    c = conn.cursor()\n",
120     "    c.execute('DELETE FROM people;') # del if
    data exists\n",
121     "    c.executemany('INSERT INTO people VALUES
     (?,?,?)', people)"
122    ]
123   },
124   {
125    "cell_type": "markdown",
126    "metadata": {
127     "pycharm": {
128      "name": "#%% md\n"
129     }
130    },
131    "source": [
132     "## Create and Load Sites Table"
133    ]
134   },
135   {
136    "cell_type": "code",
137    "execution_count": 8,
138    "metadata": {
139     "pycharm": {
140      "name": "#%%\n"
141     }
142    },
143    "outputs": [],
144    "source": [
145     "def create_sites_table(conn):\n",
146     "    sql = \"\"\"\n",
147     "    CREATE TABLE IF NOT EXISTS sites (\n",
148     "        site_id text PRIMARY KEY,\n",
149     "        latitude double NOT NULL,\n",
```

```
150     "        longitude double NOT NULL\n",
151     "        );\n",
152     "    \"\"\"\n",
153     "\n",
154     "    c = conn.cursor()\n",
155     "    c.execute(sql)\n",
156     "\n",
157     "def load_sites_table(conn):\n",
158     "    create_sites_table(conn)\n",
159     "    ## TODO: Complete code\n",
160     "    df = pd.read_csv('C:/Users/taylo/OneDrive/
    Documents/dsc650/data/external/tidynomicon/site.csv
    ')\n",
161     "    sites = df.values\n",
162     "    c = conn.cursor()\n",
163     "    c.execute('DELETE FROM sites;') # del if
    data exists\n",
164     "    c.executemany('INSERT INTO sites VALUE
     (?, ?, ?', sites)"
165    ]
166   },
167   {
168    "cell_type": "markdown",
169    "metadata": {
170     "pycharm": {
171      "name": "#%% md\n"
172     }
173    },
174    "source": [
175     "## Create and Load Visits Table"
176    ]
177   },
178   {
179    "cell_type": "code",
180    "execution_count": 9,
181    "metadata": {
182     "pycharm": {
183      "name": "#%%\n"
184     }
185    },
186    "outputs": [],
```

```
187    "source": [
188     "def create_visits_table(conn):\n",
189     "    sql = \"\"\"\n",
190     "    CREATE TABLE IF NOT EXISTS visits (\n",
191     "        visit_id integer PRIMARY KEY,\n",
192     "        site_id text NOT NULL,\n",
193     "        visit_date text,\n",
194     "        FOREIGN KEY (site_id) REFERENCES sites
     (site_id)\n",
195     "        );\n",
196     "    \"\"\"\n",
197     "\n",
198     "    c = conn.cursor()\n",
199     "    c.execute(sql)\n",
200     "\n",
201     "def load_visits_table(conn):\n",
202     "    create_visits_table(conn)\n",
203     "    ## TODO: Complete code\n",
204     "    df = pd.read_csv('C:/Users/taylo/OneDrive/
    Documents/dsc650/data/external/tidynomicon/visited.
    csv')\n",
205     "    visits = df.values\n",
206     "    c = conn.cursor()\n",
207     "    c.execute('DELETE FROM visits;') # del if
    data exists\n",
208     "    c.executemany('INSERT INTO sites VALUE
     (?, ?, ?', visits)"
209    ]
210   },
211   {
212    "cell_type": "markdown",
213    "metadata": {
214     "pycharm": {
215      "name": "#%% md\n"
216     }
217    },
218    "source": [
219     "## Create DB and Load Tables"
220    ]
221   },
222   {
```

```
223    "cell_type": "code",
224    "execution_count": 10,
225    "outputs": [],
226    "source": [
227     "db_path = results_dir.joinpath('patient-info.db
    ')\n",
228     "conn = sqlite3.connect(str(db_path))\n",
229     "# TODO: Uncomment once functions completed\n",
230     "load_people_table(conn)\n",
231     "# load_sites_table(conn)\n",
232     "# load_visits_table(conn)\n",
233     "load_measurements_table(conn)\n",
234     "\n",
235     "conn.commit()\n",
236     "conn.close()"
237    ],
238    "metadata": {
239     "collapsed": false,
240     "pycharm": {
241      "name": "#%%\n"
242     }
243    }
244   },
245   {
246    "cell_type": "code",
247    "execution_count": 11,
248    "outputs": [
249     {
250      "name": "stdout",
251      "output_type": "stream",
252      "text": [
253       "('dyer', 'William', 'Dyer')\n",
254       "('pb', 'Frank', 'Pabodie')\n",
255       "('lake', 'Anderson', 'Lake')\n",
256       "('roe', 'Valentina', 'Roerich')\n",
257       "('danforth', 'Frank', 'Danforth')\n"
258      ]
259     }
260    ],
261    "source": [
262     "# check data was entered properly by querying
```

```
data into pandas df\n",
    "conn = sqlite3.connect(str(db_path))\n",
    "c = conn.cursor()\n",
    "for row in c.execute('SELECT * FROM people;'):\n",
    "    print(row)\n",
    "c.close()"
   ],
   "metadata": {
    "collapsed": false,
    "pycharm": {
     "name": "#%%\n"
    }
   }
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.8.3"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 4
}
```