```
import sqlite3
import pandas as pd


_dbfile       = "buslah.sqlite"
_dbfile = "buslah.sqlite"
_csv_routes    = "singbus/bus_routes.csv"
_csv_services  = "singbus/bus_services.csv"
_csv_stops     = "singbus/bus_stops.csv"

class BusLahException(Exception):
    pass


def build_database(db_file, bus_stops, bus_routes, bus_services):
    try:
        con = sqlite3.connect(db_file)
        request1 = "CREATE TABLE IF NOT EXISTS bus_stops ( bus_stop_code INTEGER,\
            road_name varchar(25),\
            latitude NUMERIC(18,14),\
            longitude NUMERIC (18,14),\
            stop_description varchar(50),\
            PRIMARY KEY (bus_stop_code) )"

        request2 = "CREATE TABLE IF NOT EXISTS bus_services ( service_number
varchar(4),\
            direction integer,\
            operator varchar(4),\
            category varchar(15),\
            origin_stop_code integer,\
            destination_stop_code integer,\
            AM_peak_interval varchar(6),\
            AM_offpeak_interval varchar(6),\
            PM_peak_interval varchar(6),\
            PM_offpeak_interval varchar(6),\
            service_description varchar(50),\
            PRIMARY KEY (service_number,direction))"
        request3 = "CREATE TABLE IF NOT EXISTS bus_routes ( route_id integer,\
            service_number varchar(4),\
            direction integer,\
            operator varchar(4),\
            stop_sequence integer,\
            bus_stop_code integer,\
            distance_traveled numeric(3,1),\
            weekday_first_bus varchar(4),\
            weekday_last_bus varchar(4),\
            saturday_first_bus varchar(4),\
            saturday_last_bus varchar(4),\
            sunday_first_bus varchar(4),\
            sunday_last_bus varchar(4),\
            PRIMARY KEY(route_id))"
        cursor = con.cursor()
        cursor.execute(request1)
        cursor.execute(request2)
        cursor.execute(request3)
        cursor.close()
    # for stops
        stops = pd.read_csv(_csv_stops)
        del stops["Unnamed: 0"]
```

```
        connect = stops.to_sql('bus_stops',con,if_exists ='replace',index = False)


    # for services
        services = pd.read_csv(_csv_services)
        del services["Unnamed: 0"]
        services_temp = services.drop_duplicates(subset =
['service_number','direction'], keep = 'first', inplace = False)
        connect = services_temp.to_sql('bus_services',con,if_exists
='replace',index = False)


    # for routes
        routes = pd.read_csv(_csv_routes)
        del routes['Unnamed: 0']
        connect = routes.to_sql('bus_routes',con,if_exists ='replace',index =
False)

        con.commit()
    except:
        raise BusLahException()
    finally:
        con.close()




build_database(db_file = _dbfile, bus_routes = _csv_routes,bus_services =
_csv_services,bus_stops = _csv_stops)




# end of task 1.

# class GenericDB :

#     def __init__ (self, dbfile) :
#         self.conn = sqlite3.connect(dbfile)

#     def get_tables(self):
#         return pd.read_sql_query("SELECT name FROM sqlite_master WHERE\
#                                   type='table' ORDER BY name;", self.conn)

#     def get_table_infos(self,table):
#         return pd.read_sql_query("PRAGMA table_info({});".format(table),
self.conn)

#     def custom_request(self,req):
#         return pd.read_sql_query(req, self.conn)



# def stops_query(dbfile):
#     con = sqlite3.connect(dbfile)
```

```
#     mt = []
#     cursor = con.cursor()
#     cursor.execute("SELECT bus_stop_code FROM bus_stops")
#     result = cursor.fetchall()
#     for i in result:
#         mt.append(i[0])
#     con.close()
#     return mt

# stops_query(_dbfile)

# def line_stops_query(line,direction):
#     con = sqlite3.connect(_dbfile)
#     mt = []
#     cursor = con.cursor()
#     cursor.execute("SELECT bus_stop_code FROM bus_routes WHERE service_number ==
{} and direction == {};".format(line,direction))
#     result = cursor.fetchall()
#     for i in result:
#         mt.append(i[0])
#     con.close()
#     return mt
# line_stops_query(96,1)

# def most_left():
#     con = sqlite3.connect(_dbfile)
#     mt = []
#     cursor = con.cursor()
#     cursor.execute('SELECT bus_stop_code FROM bus_stops WHERE longitude ==
(SELECT MIN(longitude) from bus_stops)')
#     result = cursor.fetchall()
#     for i in result:
#         mt.append(i[0])
#     con.close()
#     return mt
# most_left()

# def area_line(lon_min,lon_max,lat_min,lat_max):
#     con = sqlite3.connect(_dbfile)
#     mt = []
#     cursor = con.cursor()
#     cursor.execute("SELECT DISTINCT service_number FROM bus_routes INNER JOIN
bus_stops ON bus_routes.bus_stop_code = bus_stops.bus_stop_code WHERE latitude
BETWEEN {0} and {1} and longitude BETWEEN {2} and
{3};".format(lat_min,lat_max,lon_min,lon_max))
#     result = cursor.fetchall()
#     for i in result:
#         mt.append(i[0])
#     con.close()
#     return mt
# area_line(103.0, 104.0, 1.2, 1.4)
# sum(['402' == area_line(103.0, 104.0, 1.2, 1.4)]) == 1
# '402' == area_line(103.0, 104.0, 1.2, 1.4)



class GenericDB :
```

```
    def __init__ (self, _dbfile) :
        self.conn = sqlite3.connect(_dbfile)

    def get_tables(self):
        return pd.read_sql_query("SELECT name FROM sqlite_master WHERE\
                                  type='table' ORDER BY name;", self.conn)

    def get_table_infos(self,table):
        return pd.read_sql_query("PRAGMA table_info({});".format(table), self.conn)

    def custom_request(self,req):
        return pd.read_sql_query(req, self.conn)




class SQLahDB(GenericDB):
    def stops_query(self):
        con = sqlite3.connect(_dbfile)
        mt = []
        cursor = con.cursor()
        cursor.execute("SELECT bus_stop_code FROM bus_stops")
        df = pd.read_sql_query("SELECT bus_stop_code FROM bus_stops",con)
        result = cursor.fetchall()
        for i in result:
            mt.append(i[0])
        con.close()
        mt.append("bus_stop_code")
        return df
    def line_stops_query(self,line,direction):
        con = sqlite3.connect(_dbfile)
        mt = []
        cursor = con.cursor()
        cursor.execute("SELECT bus_stop_code FROM bus_routes WHERE service_number
== {} and direction == {};".format(line,direction))
        df = pd.read_sql_query("SELECT DISTINCT bus_stop_code FROM bus_routes WHERE
service_number == {} and direction == {};".format(line,direction),con)
        result = cursor.fetchall()
        for i in result:
            mt.append(i[0])
        con.close()
        mt.append("bus_stop_code")
        return df
    def most_left(self):
        con = sqlite3.connect(_dbfile)
        mt = {}
        cursor = con.cursor()
        cursor.execute('SELECT bus_stop_code FROM bus_stops WHERE longitude ==
(SELECT MIN(longitude) from bus_stops)')
        result = cursor.fetchall()
        df = pd.read_sql_query('SELECT bus_stop_code FROM bus_stops WHERE longitude
== (SELECT MIN(longitude) from bus_stops)',con)
        for i in result:
            mt["bus_stop_code"] = ([i[0]])
        con.close()
        return df
    def area_line(self,lon_min,lon_max,lat_min,lat_max):
        con = sqlite3.connect(_dbfile)
```

```
        mt = []
        cursor = con.cursor()
        cursor.execute("SELECT DISTINCT service_number FROM bus_routes INNER JOIN
bus_stops ON bus_routes.bus_stop_code = bus_stops.bus_stop_code WHERE latitude
BETWEEN {0} and {1} and longitude BETWEEN {2} and
{3};".format(lat_min,lat_max,lon_min,lon_max))
        result = cursor.fetchall()
        for i in result:
            mt.append(i[0])
        df = pd.read_sql_query("SELECT DISTINCT service_number FROM bus_routes
INNER JOIN bus_stops ON bus_routes.bus_stop_code = bus_stops.bus_stop_code WHERE
latitude BETWEEN {0} and {1} and longitude BETWEEN {2} and
{3};".format(lat_min,lat_max,lon_min,lon_max),con)
        con.close()
        mt.append("service_number")
        return df



haha = SQLahDB(_dbfile)
# haha2 = SQLahDB(_dbfile)
# assert(haha2.most_left()["bus_stop_code"][0] == 25751)
```