```python
"""
    File that handles all connections with the Postgresql database.
"""

import psycopg2
import json


class PostgresConnector:
    def __init__(self, config_file):
        """
            Constructor for the class; creates connection with information within the db.config
            file.
        """
        self.__connection = None
        self.db_name = 'student'
        try:
            self.config = json.load(open(config_file))
        except:
            self.config = None
            print("error loading file " + config_file)

        self.__create_connection()

    def __create_connection(self):
        """
        (Private)
        Creates database connection with data from db.config file. Initially run when server starts.

        Sets variable __connection to psycopg2 connection object to be used when editing database.
        """
        if self.config:
            try:
                # Attempts to create connection.
                conn = psycopg2.connect(
                    'dbname=' + self.config['db'] + ' user=' + self.config['user'] + ' password=' + self.config[
                        'password'] + ' host=' + self.config['host'] + ' port=' + self.config['port'])
                conn.autocommit = True
                self.__connection = conn
            except:
                print("Connection Error")

    def query(self, sql_command, query_string):
        """
```

```
            Method for querying the database. Accepts:

                sql_command: String
                query_string: String

            :returns

                A list of all matching results
        """
        cur = self.__connection.cursor()
        # Query string is concatenated with sql command by Psycopg2 to prevent
        # SQL injection attacks.
        cur.execute(sql_command, query_string)
        return cur.fetchall()

    def update(self, sql_command):
        """
            Method for updating the database. Accepts:

                sql_command: String
        """
        cur = self.__connection.cursor()
        cur.execute(sql_command)

    def select(self, sql_command):
        """
            Method for selecting from the database. Accepts:

                sql_command: String

            :returns

                A list of all matching results
        """
        cur = self.__connection.cursor()
        cur.execute(sql_command)
        return cur.fetchall()

    def insert(self, sql_command, args):
        """
            Method for inserting items into the database. Accepts:

                sql_command: String
                args: (String)     (a tuple of strings)
        """
        cur = self.__connection.cursor()
        cur.execute(sql_command, args)

```