```python
from cassandra.auth import PlainTextAuthProvider
from cassandra.cluster import Cluster

from config import database_config
from Logger import Logging

logger_obj = Logging('Advance Image Downloader')  # Creating a custom based logger
logger_obj.initialize_logger()  # Instantiating the logger object


class Cassandra:

    def __init__(self):
        """
        This function will instantiate the session for the Cassandra database
        """
        try:
            cloud_config = {
                'secure_connect_bundle': database_config.cloud_config_path
            }
            auth_provider = PlainTextAuthProvider(database_config.cassandra_uname,
                                                  database_config.cassandra_password)
            cluster = Cluster(cloud=cloud_config, auth_provider=auth_provider)
            self.session = cluster.connect()

        except Exception as e:
            logger_obj.print_log('(Cassandra.py(__init__) - Something went wrong ' + str(e), 'exception')
            raise Exception(e)

    def connect_keyspace(self):
        """
        This function will use the given keyspace as the default method to work on.
        """
        try:
            self.session.set_keyspace(database_config.keyspace_name)
        except Exception as e:
            logger_obj.print_log('(Cassandra.py(connect_keyspace) - Something went wrong ' + str(e), 'exception')
            raise Exception(e)

    def create_table(self):
        """
        This function will create the table if it does not exists in the keyspace
        """
        try:
            self.session.execute('CREATE TABLE IF NOT EXISTS {} '
                                 '(id UUID, email text, url text, PRIMARY KEY (id, email, url));'
                                 .format(database_config.table_name))
        except Exception as e:
            logger_obj.print_log('(Cassandra.py(create_table) - Something went wrong ' + str(e), 'exception')
            raise Exception(e)

    def select_query(self, req_id):
        """
        This function will execute and return the select query on the table
        :param req_id: Unique request id for the request generated by the user
        :return: Results after the selection query
        """
        try:
            return self.session.execute('SELECT id, email, url FROM {} WHERE id={}'.format(database_config.table_name,
                                                                                           req_id))
        except Exception as e:
            logger_obj.print_log('(Cassandra.py(select_query) - Something went wrong ' + str(e), 'exception')
            raise Exception(e)

    def insert_url(self, uuid, email, url):
        """
        This function will insert the data into the table
        :param uuid: It is a unique user id object
        :param email: email of the user
        :param url: url of the search query
        """
        try:
            self.session.execute(
                "INSERT INTO " + database_config.table_name + " (id, email, url) VALUES (%s, %s, %s)",
                (uuid, email, url))

        except Exception as e:
            logger_obj.print_log('(Cassandra.py(insert_url) - Something went wrong ' + str(e), 'exception')
            raise Exception(e)

    def shutdown(self):
        """
        This function will close the cassandra session
        """
        try:
            self.session.shutdown()
        except Exception as e:
            logger_obj.print_log('(Cassandra.py(shutdown) - Something went wrong ' + str(e), 'exception')
            raise Exception(e)

    def delete_url(self, req_id):
        """
        This function will delete the url for given request ID
        """
        try:
            self.session.execute('DELETE FROM {} WHERE id={}'.format(database_config.
table_name, req_id))
        except Exception as e:
            logger_obj.print_log('(Cassandra.py(shutdown) - Something went wrong ' + str(e), '
exception')
            raise Exception(e)

    def drop_table(self):
        """
        This function will drop the given table from the keyspace
        """
        try:
            self.session.execute('DROP TABLE IF EXISTS {}'.format(database_config.table_name))
        except Exception as e:
            logger_obj.print_log('(Cassandra.py(drop_table) - Something went wrong ' + str(e), '
exception')
            raise Exception(e)

```