SQL\_Eval

**Működés:**

A rendszer SQL lekérdezés stringekkel dolgozik, azokat alakítja át pandas dataframekre, majd hasonlítja őket össze. Ezt a compare\_query\_results függvénnyel lehet elérni.

*Compare\_Query\_results:*

Bemenetek:

* query\_gold (str): az elvárt sql lekérdezés egy erősen „parametrizált” változata
* query\_gen (str): a generált és összehasonlításra váró sql lekérdezés

Kimenet:

* bool: azt adja vissza, hogy található-e a részleges egyezést (subset)

Gyakorlatilag egy összesítő függvényről beszélünk, mely legenerálja az összes lehetséges elfogadott lekérdezést a query\_gold átalakításával, majd legenerál az adatbázis szerint pandas dataframeket, melyből az egyik tartalmazza a generált lekérdezés eredményeit, a másik pedig összehasonlításképpen az éppen vizsgált referencia változatot (for ciklus végigmegy a lehetséges query\_goldból generált lekérdezéseken). Ezután a subset\_df-el visszatér az eredménnyel.

*get\_all\_minimal\_queries:*

Bemenetek:

* query (str): lekérdezés parametrizált megvalósítása (pl.: SELECT {uid,name}, likes\_movies FROM users. A parametrizálás a {} jelek között lévő értékek potenciális előfordulását jelentik (egynek biztos benne kell lennie))

Kimenet:

* list[str]: a lista tartalmaz minden lehetséges permutációt, ami a parametrizált queryből előfordulhat (kivéve a triviális üres esetet)

A megfelelő formalizálással végigiterál a lehetséges megoldásokon és berakja őket a kimeneti listába.

*subset\_df:*

Bemenetek:

* df\_sub (pd.Dataframe): az első összhasonlításra váró dataframe
* query\_sub (str): az első dataframe lekérdezése
* df\_super (pd.Dataframe): a második összhasonlításra váró dataframe
* query\_super (str): a második dataframe lekérdezése

Kimenet:

* bool: Van-e részleges egyezés => df\_sub subset-e a df\_superhez képest

Megkeresi, hogy a df\_sub minden oszlopa megtalálható-e a superben, minden esetben az assert\_series\_equal függvényt alkalmazza az oszlopok beazonosítására, az assert\_frame\_equal függvényt pedig, miután lekreálta a redukált listát, mellyel összhasonlítja a df\_subot. Ha nem sikerül, akkor nem található meg benne (vagy nem találja meg az adott oszlopot, akkor egy listát is kiír a különbségekkel, vagy nem egyeznek az értékek)

*normalize\_table:*

Bemenetek:

* df (pd.Dataframe): a normalizálandó dataframe
* quary (str): a dataframe lekérdezése

Kimenet:

* pd.Dataframe: a normalizált dataframe

Rendezi a dataframe oszlopsorrendjét alfabetikus sorba, majd megvizsgálja, hogy a lekérdezésben van-e sorrendkikötés, ha nincs, akkor a sorokat is rendezi alfabetikus sorba. Ezután reseteli a dataframe indexeit és visszatér vele.

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás

Felismerte az első egyedet, a második és harmadik egyednél pedig kiírta, hogy melyik oszlopnál akadt el (nem találta meg)

A képen szöveg, képernyőkép, szoftver látható

Automatikusan generált leírás

Megfordítva az oszlopokat is felismerte, hiszen a normalizálásnál rendezi magának az oszlopokat plusz, ha nincs a lekérdezésben ORDER BY, rendezi magának a sorokat is (balról jobbra). Ezen kívül megint látható a keresés

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás

A hiányos quaryt nem ismeri fel egyik megoldás sem.

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás

Ugyanez igaz akkor is, ha megváltozik az érték.

A képen képernyőkép, szöveg, Multimédiás szoftver, szoftver látható

Automatikusan generált leírás

Amikor megfordítva az oszlopokat, de beraktam az összes oszlopot, felismerte a rendszer, plusz mivel ez volt a legtágabb halmaz, amit meg lehetett adni, ezért a subseteket is felismerte mindegyik esetben

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Automatikusan generált leírás

Az első felállás nevének megváltoztatásával felismerte a rendszer, hiszen az adott oszlop neve nem befolyásolja a felismerést (a nevet nem vizsgálja csak a tartalmat)