問題

次のフィールド定義にしたがう行動履歴ログテーブルが存在します。
この行動履歴テーブルを用いて、一緒に購入されやすい商品を見つけるアソシエーション分析を行います。

行動履歴テーブル:

あるウェブサイトでのユーザの行動履歴に関するテーブルです。
操作を行った時間、ユーザを識別するためのID、行った操作、操作を行った商品が記録されています。

- テーブル定義

フィールド	説明	データベースでの型
action_time	操作を行った時間。yyyy-MM-dd HH:mm:ss	TEXT
uu_id	ユーザを識別するためのID	TEXT
action_type	ユーザが行った操作を示す。ACTIONの種類はPV（閲覧）,MYLIST（お気に入り登録）, CV（購入）のいずれか	TEXT
item_id	操作を行った商品を識別するためのID	TEXT

- サンプル

action_time	uu_id	action_type	item_id
2020-05-02 12:48:35	6208f02039cee35638c4d8a	PV	A0001
2020-05-02 12:50:27	6208f02039cee35638c4d8a	MYLIST	A0001
2020-05-02 12:52:34	6208f02039cee35638c4d8a	PV	B0002
2020-05-02 12:54:10	6208f02039cee35638c4d8a	CV	B0002
2020-05-02 17:01:11	6208f02039cee35638c4d8a	CV	A0001
2020-05-10 10:56:45	6208f02039cee35638c4d8a	CV	A0001
2020-05-03 20:20:31	478f256076a7fc9ad20c859	PV	B0002
2020-05-03 21:34:10	478f256076a7fc9ad20c859	CV	B0002
2020-05-03 14:53:08	478f256076a7fc9ad20c859	PV	C1002
2020-05-03 15:26:05	478f256076a7fc9ad20c859	CV	C1002

- サンプルデータ

`data/database.sqlite`の`access_log`テーブルに行動履歴テーブルのサンプルデータを格納しています。またSQL実行のサンプルクエリは`sample.sql`です。必要な場合は、SQLite3 (==3.28.0)をインストールの上、`sample.sql`を書き直し、クエリの実行を行ってください。

実行方法

```
$ sqlite3 data/database.sqlite < sample.sql
```

注意事項:

- 本問題に出てくるSQLは標準SQL規格に準拠したものです。
- 本問題内で出てくるSQLは簡略化のため、`CREATE TABLE 'テーブル名' AS`という表現を省略しています。
- 各問題内では、同じ選択肢を複数回選択しても良いものとします。

問1

以下は今回のアソシエーション分析に用いる指標に関する説明です。空欄(A)~(D)に当てはまるものとして適切なものを選択肢1~8の中から選んでください。なお、以下の説明文では、商品1を購入したユーザの集合をX、商品2を購入したユーザの集合をY、全体のユーザの集合をZとします。

支持度（Support）は、全ユーザのうち、商品1と商品2の両方を購入したユーザの割合として定義され、数式を用いて下記のように表現される。

$$Support = (A)$$

確信度・信頼度（Confidence）は、商品1を購入したユーザのうち、商品2も購入したユーザの割合として定義され、数式を用いて下記のように表現される。

$$Confidence = (B)$$

リフト（Lift）は、商品1を購入したユーザのうち商品2も購入したユーザの割合を、全体のうち商品2を購入するユーザの割合で割った値として定義され、数式を用いて下記のように表現される。

$$Lift = \frac{(C)}{(D)}$$

1. $\frac{|X|}{|Z|}$
2. $\frac{|Y|}{|Z|}$
3. $\frac{|X \cap Y|}{|Z|}$
4. $\frac{|X \cup Y|}{|Z|}$
5. $\frac{|X \cap Y|}{|X|}$
6. $\frac{|X \cap Y|}{|Y|}$
7. $\frac{|X \cup Y|}{|X|}$
8. $\frac{|X \cup Y|}{|Y|}$

問2

問2-1

アソシエーション分析を行うために、まず`行動履歴テーブル`から、ユーザが過去に購入したことのある商品のみを抽出した、下記のような`購入商品テーブル`を作成します。

購入商品テーブル:

- サンプル

uu_id	item_id
6208f02039cee35638c4d8a	A0001
6208f02039cee35638c4d8a	B0002
478f256076a7fc9ad20c859	B0002
478f256076a7fc9ad20c859	C1002

ただし、同一ユーザが同一商品を複数回購入するケースも存在するので、`uu_id`、`item_id`が重複したレコードはユニークにする必要があります。
このような操作をするSQLとして、**不適切な**ものを下記から1つ選びなさい。

1

```
SELECT
  DISTINCT uu_id,
  item_id
FROM
  `行動履歴テーブル`
WHERE
  action_type = "CV"
```

2

```
SELECT
  DISTINCT uu_id,
  item_id
FROM
  `行動履歴テーブル`
HAVING
  action_type = "CV"
```

3

```
SELECT
  uu_id,
  item_id
FROM
  `行動履歴テーブル`
WHERE
  action_type = "CV"
GROUP BY
  uu_id,
  item_id
```

4

```
SELECT
  uu_id,
  item_id
FROM
  `行動履歴テーブル`
GROUP BY
  uu_id,
  item_id,
  action_type
HAVING
  action_type = "CV"
```

問2-2

問2-1で作成した`購入商品テーブル`に、各商品を購入したユーザのユニーク数を示した`num_cv_uu`、購入商品テーブルに記録されたユーザのユニーク数を示した`num_total_uu`を付与した下記のような`ユニークユーザ数付与済み購入商品テーブル`を作成します。

ユニークユーザ数付与済み購入商品テーブル:

- サンプル

uu_id	item_id	num_cv_uu	num_total_uu
6208f02039cee35638c4d8a	A0001	1	2
6208f02039cee35638c4d8a	B0002	2	2
478f256076a7fc9ad20c859	B0002	2	2
478f256076a7fc9ad20c859	C1002	1	2

この`ユニークユーザ数付与済み購入商品テーブル`を下記のSQLで作成します。(A), (B)に当てはまるものとして、1~6の選択肢からそれぞれ正しいものを選びなさい。

```
SELECT
  uu_id,
  item_id,
  (A) num_cv_uu,
  num_total_uu
FROM
  `購入商品テーブル` table1
INNER JOIN
  (
    SELECT
      (B) num_total_uu
    FROM
      `購入商品テーブル`
  ) table2
```

1. `COUNT(1)`
2. `COUNT(1) OVER()`
3. `COUNT(1) OVER(PARTITION BY item_id)`
4. `COUNT(1) OVER(PARTITION BY uu_id)`
5. `COUNT(DISTINCT uu_id)`
6. `COUNT(DISTINCT item_id)`

問3

問3-1

問2-2で作成した`ユニークユーザ数付与済み購入商品テーブル`を自己結合することで、商品1、商品2の両方を購入したユーザのユニーク数を集計したテーブルである`商品共起テーブル`を作成します（ただし商品1≠商品2）。また同時に問1で回答した支持度（Support）、確信度・信頼度（Confidence）、リフト（Lift）の計算も行います。ただし、支持度、確信度・信頼度、リフトは四捨五入し、小数点以下5桁までの数字とします。

商品共起テーブル:

- テーブル定義

フィールド	説明
item_id1	商品1を識別するためのID
item_id2	商品2を識別するためのID
num_cv_uu1	商品1を購入したユーザのユニーク数
num_cv_uu2	商品2を購入したユーザのユニーク数
num_total_uu	全ユーザのユニーク数
support	支持度
confidence	確信度・信頼度
lift	リフト

この`商品共起テーブル`を下記のSQLで作成します。(A)~(F)に当てはまるものとして、下記の選択肢からそれぞれ正しいものを選びなさい。

```
SELECT
  table1.item_id item_id1,
  table2.item_id item_id2,
  table1.num_cv_uu num_cv_uu1,
  table2.num_cv_uu num_cv_uu2,
  table1.num_total_uu num_total_uu,
  ROUND((A), 5) support,
  ROUND((B), 5) confidence,
  ROUND((C), 5) lift
FROM
  `ユニークユーザ数付与済み購入商品テーブル` table1
INNER JOIN
  `ユニークユーザ数付与済み購入商品テーブル` table2
ON
  (D)
WHERE
  (E)
GROUP BY
  (F)
```

- (A) ~ (C)の選択肢

1. `CAST(COUNT(1) AS REAL) / CAST(table1.num_total_uu AS REAL)`
2. `CAST(COUNT(1) AS REAL) / CAST(table1.num_cv_uu AS REAL)`
3. `CAST(COUNT(1) AS REAL) / CAST(table2.num_cv_uu AS REAL)`

4. `CAST(table1.num_cv_uu AS REAL) / CAST(table1.num_total_uu AS REAL)`
5. `CAST(table2.num_cv_uu AS REAL) / table1.num_total_uu`
6. `CAST(table1.num_cv_uu * table1.num_total_uu AS REAL) / CAST(table2.num_cv_uu * COUNT(1) AS REAL)`
7. `CAST(table2.num_cv_uu * table1.num_total_uu AS REAL) / CAST(table1.num_cv_uu * COUNT(1) AS REAL)`
8. `CAST(COUNT(1) * table1.num_total_uu AS REAL) / CAST(table1.num_cv_uu * table2.num_cv_uu AS REAL)`
9. `CAST(table1.num_cv_uu * table2.num_cv_uu AS REAL) / CAST(table1.num_total_uu * COUNT(1) AS REAL)`

- (D)、(E)の選択肢

1. `table1.item_id = table2.item_id`
2. `table1.item_id <> table2.item_id`
3. `table1.uu_id = table2.uu_id`
4. `table1.uu_id <> table2.uu_id`

- (F)の選択肢

1. `table1.item_id, table2.item_id`
2. `table1.item_id, table2.item_id, table1.num_cv_uu`
3. `table1.item_id, table2.item_id, table1.num_cv_uu, table2.num_cv_uu`
4. `table1.item_id, table2.item_id, table1.num_cv_uu, table2.num_cv_uu, table1.num_total_uu`

問3-2

問3-2で作成した`商品共起テーブル`のLiftの値を元に、item_id1に対してLiftが高い順にitem_id2を上位3件並べたものを1行とする`アソシエーションルールテーブル`を作成します。ただし、Lift値が同率のitem_id2が複数存在した場合は、item_id2の値が小さいものを上位とします。また、item_id1に紐づくitem_id2が3件以上ない場合は、存在しないitem_idをnullとして出力します。さらにレコードはitem_id1の昇順にソートして出力します。

アソシエーションルールテーブル:

- テーブル定義

フィールド	説明
item_id1	商品1を識別するためのID
item_id21	item_id1に紐づくitem_id2のうち、1番Liftが高いitem_id2
item_id22	item_id1に紐づくitem_id2のうち、2番目にLiftが高いitem_id2
item_id23	item_id1に紐づくitem_id2のうち、3番目にLiftが高いitem_id2

この`アソシエーションルールテーブル`を下記のSQLで作成します。(A)~(K)に当てはまるものとして、下記の選択肢からそれぞれ正しいものを選びなさい。

```
SELECT
  item_id1,
  (A)((B) (C) lift_rank = 1 (D) item_id2 (E) null (F)) item_id21,
  (A)((B) (C) lift_rank = 2 (D) item_id2 (E) null (F)) item_id22,
  (A)((B) (C) lift_rank = 3 (D) item_id2 (E) null (F)) item_id23
FROM
  (
      SELECT
        item_id1,
        item_id2,
        ROW_NUMBER() OVER(PARTITION BY (G) ORDER BY (H) DESC, (I) ASC)
lift_rank
      FROM
        `商品共起テーブル`
  )
GROUP BY
  (J)
ORDER BY
  (K)
```

- (A) ~ (F)の選択肢

1. `CASE`
2. `COUNT`
3. `ELSE`
4. `END`
5. `MATCH`
6. `MAX`
7. `THAN`
8. `THEN`
9. `WHEN`

- (G) ~ (K)の選択肢

1. `item_id1`
2. `item_id2`
3. `num_cv_uu1`
4. `num_cv_uu2`
5. `num_total_uu`
6. `support`
7. `confidence`
8. `lift`
9. `lift_rank`