1. [MapReduce] (30) Consider a set R of web pages. Each record r from set R is in the form of <docid, [List of terms]>, which contains the id (docid) of a web page, and a list of terms in the web page. Below are two examples: <1, [Harry, Potter, fantasy, novels, lives, wizard]>; <2, [Harry, young, wizard, student, Hogwarts]>.

Design and describe MR algorithms. You are not required to provide detailed code, but provide (1) Map and Reduce functions following the examples in the lecture notes; and (2) a brief description of the algorithm that invokes the Map and Reduce functions. If needed, you may use multiple rounds of Map and Reduce.

a. Recall the inverted index, which is a set of pairs (term, list(docid)). Each pair records a term and all the documents (docid) that contain the term. Build an inverted index for the set R for a set of terms T.

Map: `<docid, list(term)>` → `list(<term, docid>)`

- Intermediates are hash partitioned on term. Then each partition (`<term, list(docid)>`) is sent to the reducer

Reducer: `<term, list(docid)>` → `list(<term, list(docid)>)`

- Reducer just gets rid of duplicates. Nothing much after the hash partition

b. Top-k frequent words. Search engines often maintain popular web pages and retrieve most frequent keywords to support fast keyword search. Given this set R, describe a MapReduce algorithm to report the top k most frequent terms and their frequency as appeared in the web pages in R. For the above example with input doc1 and doc 2, the output returns top 2 most frequent terms as <Harry, 3> and <wizard, 2>.

We want the global frequencies of all terms, then select the **k** largest.

1. First MR (term counts):
   - Map: same as above but emit `(t, 1)` for each occurrence.
   - Reduce: for each term t, sum all counts → emit `(t, count_t)`.
2. Second MR (top-k):

- Map: identity: receive `(t, count_t)`, emit `("*", (t, count_t))` so all go to one key.

- Reduce : receive key `"*"` and the list `[(t, count_t), …]`. Keep a min-heap (size k) of highest-frequency terms; at end, output the contents of the heap in descending order:

c. Co-occurrence. Given R, describe an MR algorithm that compute how often each pair of terms co-occurs in all the individual documents. The output as a format like ((t1, t2), # of occurrence). For the above example, one such output is: <(Harry, wizard), 2>

For each unordered pair of distinct terms (t , t) that appear in the *same* document, we want the total number of documents in which they co-occur.

Map:

- Input: `docId, [t , t , …, t ]`
- Operation:
  1. Sort terms and remove duplicates within the doc: `[u , u , …, u_m]`.
  2. For every *unordered* pair (u , u) with i < j: `emit(key = (u , u ), value = 1)`

Reduce:

- Input: `((u , u ), [1, 1, …])`
- Operation: sum the list total co-occurrence count `emit((u , u ), total_count)`

2. [Dirty data] (30) Recall the five common types of data quality issues we discussed: inconsistency (violation of data constraints), duplication (redundant information), missing data (incomplete data), data currency (outdated data), and data accuracy (inaccurate data). You are reviewing medical record data (structured data stored in tables) for a hospital, which records patient information, and their pharmacy and insurance providers. Annotate the following descriptions with *all* the type of data quality issues that can describe them and briefly clarify the reason.

a. A patient named "John Carroll" is recorded as "J.Carrol" in the database. His case is confused with another patient named "Joe Carol" who has the same birthdate as John Carroll. The latter patient has a recorded address in "Nimrod St, Solon" but he said he had already moved to "Wintergreen Dr, Solon".

- **Inconsistency:** name format differs ("John Carroll" vs "J.Carrol" vs "Joe Carol")
- **Duplication/Ambiguity:** two similar records lead to confusion which is which
- **Data Currency:** address "Nimrod St" is stale; actual is "Wintergreen Dr"

b. There are multiple medical records which have no required identifiers called "medical record numbers" (MRN).

- **Data Completeness:** mandatory key field (MRN) is absent

c. There are two patient records with different MRNs but the same name, birthdate, SSN, and home address. Later it is confirmed that both refer to the same patient.

- **Duplication:** redundant patient records referring to the same entity

d. A transferred patient has an age "65" in her profile from another healthcare provider and an age "70" in the hospital records.

- **Inconsistency:** conflicting ages across sources • **Data Accuracy:** at least one age is wrong

e. A patient claimed he has an insurance provider A, but A does not exist in any existing provider records; a data constraint states "any insurance provider of a patient must be from a set of registered providers".

- **Inconsistency (Constraint Violation):** FK constraint on insurer not satisfied

f. A patient visited his local pharmacy to get his medicine and was told that he would need to pay the full bill because the pharmacy's record and the insurance provider's record of that patient had "age" mismatched.

- **Inconsistency:** age fields disagree across two systems

g. A patient record shows a patient with an SSN X passed away on "5/1/2022". Another record in a pharmacy shows a medical purchase with the same SSN X on "4/11/2024".

- **Data Accuracy:** impossible timeline; one record is erroneous

3. [Data Error] (40) Consider two databases from a bank with the following schema:

Customer (FN, LN, St, City, CC, Country, tel, gd);

Tran (FN, LN, St, City, CC, Country, phn, when, where)

Customer record specifies a credit card holder identified by first name (FN), last name (LN), street (St), city, country code (CC), country, phone number (tel) and gender (gd). A tran tuple is a record of a purchase paid by a credit card at time when (local time) and place where, by a customer identified by first name (FN), last name (LN), street (St), City, country code (CC), Country and phone number (phn). An inclusion dependency states that for any tuple in Tran with value Tran(FN,LN,phn), there is a tuple in Customer with the same value Customer(FN,LN,Tel). The records below are correct, consistent, and up to date. They log two customers and their matching transaction records.

FN	LN	St	City	CC	Country	Tel	gd
David	Jordan	12 Holywell St	Oxford	44	UK	66700543	Male
Paul	Simon	5 Ratcliffe Terrace	Oxford	44	UK	44944631	Male

FN	LN	St	City	CC	Country	phn	when	where
David	Jordan	12 Holywell St	Oxford	44	UK	66700543	1 pm, 10/18/2019	Netherlands
Paul	Simon	5 Ratcliffe Terrace	Oxford	44	UK	44944631	6 am, 11/25/2019	US

If You are an "attacker" and have gained access to the table. For each of the five data quality issues,

a. give at least one example modification (Insert, delete, update) that will "pollute" the above clean data by causing that data quality issue (no need to give SQL; just describe how you will modify the tuples);

1. Data Consistency

Update the phone number for David to 66666666 in only the Tran table.

2. Entity Resolution

Create a duplicate customer record for Paul Simon:

('Paul','Simon','5 Ratcliffe Terrace','Oxford','44','UK','44944631','Male')

3. Information Completeness

Remove Paul's Telephone number and set it to Null

4. Data Currency

Update the address of Paul to a city he was formerly in

Paul country = Cambridge

5. Data Accuracy

Update the transaction code to something incorrect

Paul CC = 44

b. give a query Q such that the results of Q from the original tables and the polluted tables will be different. You may state the query in SQL, relational algebra, or natural language.

For example, removing "Oxford" from City causes missing data in Tran; a query SELECT City FROM Customer WHERE CC='44' and LN='Simon' will be affected.

1. Data Consistency

```
SELECT C.FN, C.LN, T.when
FROM Customer C
JOIN Tran T
  ON C.FN = T.FN
 AND C.LN = T.LN
 AND C.Tel = T.phn;
```

Clean result: returns David Jordan's transaction on 10/18/2019.

After pollution: returns no rows for David Jordan.

2. Entity Resolution

```
SELECT FN, LN, COUNT(*) AS cnt
FROM Customer
GROUP BY FN, LN
HAVING COUNT(*) > 1;
```

Clean result: no (FN,LN) has cnt>1.

After pollution: returns ('Paul','Simon') with cnt=2.

3. Information Completeness

```
SELECT Tel
FROM Customer
WHERE FN='Paul' AND LN='Simon';
```

Clean result: returns 44944631.

After pollution: returns NULL.

4. Data Currency

```
SELECT City
FROM Customer
WHERE FN='David' AND LN='Jordan';
```

Clean result: Oxford.

After pollution: Cambridge (outdated/incorrect).

5. Data Accuracy

```
SELECT C.FN, C.LN, C.CC AS cust_CC, T.CC AS tran_CC
FROM Customer C
JOIN Tran T
  ON C.FN = T.FN
 AND C.LN = T.LN
WHERE C.CC <> T.CC;
```

Clean result: no mismatches.

After pollution: shows Paul Simon with cust_CC = '44' vs tran_CC = '45'.