**Part 1.  
Question 1**:

**SELECT** SalesRank, Title, Publisher, Pages, RatingsCount, RatingValue **from** Book3 b2 **where** ID **in**

(**SELECT** ID

**FROM** Book3 b

**where** RatingValue != '' **and** RatingValue **IS** **NOT** **NULL**

**ORDER** **BY** RatingValue **DESC**

**LIMIT** 5)

**or**

(

**SELECT** ID

**FROM** Book3 b

**where** RatingValue != '' **and** RatingValue **IS** **NOT** **NULL**

**ORDER** **BY** RatingValue

**LIMIT** 5

)

**order** **by** SalesRank **desc**

**limit** 10

**1.2**

We use table and columns, but we also can use “stored procedure” to do it too.

We use table schema : Book3(SalesRank, Title, Publisher, Pages, RatingsCount, RatingValue)

**Question 2**:

**2.1**

We choose vertically fragmented.

Fragement1:

ID, **Date**, Pages, HardcoverPrice, EbookPrice

**FROM** Book3;

Fragement2:

**SELECT** Title, Author1, Author2, Author3, Publisher, ISBN13,

ProductDimensions, SalesRank, RatingsCount, RatingValue, PaperbackPrice, AudiobookPrice

**FROM** Book3;

**2.2**

not valid, because we miss some query ,for example rating count in [120,125] will not be queried.

And it’s not distributed we’ll.

As ID column is from 0,1,2 …. ID column is step as 1 incrementing column, we can:

Fragement1: ID < max(ID)/3

Fragement2: ID < max(ID)\*2/3 and ID >= max(ID)/3

Fragement3: ID < max(ID) and ID >= max(ID)\*2/3

**Part 2**

**Question 3**:

Sales table, the fact table have most records, dimension table only have summed-up records which less than fact table.

Fact table:

sale\_table(

branch\_key,

time\_key,  
book\_key,

price\_solide

)

dimension tables :

LanguageDim (Langguage\_id, LanguageName)

DayDim (Day, Week, Month, Year)

PublisherDim (Publisher\_id, PublisherName)

Branch:

branch\_key

branch\_name  
branch\_type

**Question 4**

**4.1**

The advantages of using bitmap indexes are：

Significant reduction in space and I/O  
Reduce processing time：

Comparison, join and aggregation operations can be reduced to bit operation ；

Bit operations are very fast

Bitmap index can query on multiple column keys with less space.

Bitmap indexs is not suitable for high cardinality domains

**4.2**

publisher language

AAAI Press | Springer International Publishing | Springer London |IEEE Computer Society Press English |Spanish

1 0 0 0 1 0

0 1 0 0 1 0

0 0 1 0 1 0

0 0 0 1 1 0

1 0 0 0 0 1

0 1 0 0 0 1

0 0 1 0 0 1

0 0 0 1 0 1

then the bitmap index is :

1 0 0 0 1 0

0 1 0 0 1 0

0 0 1 0 1 0

0 0 0 1 1 0

1 0 0 0 0 1

0 1 0 0 0 1

0 0 1 0 0 1

0 0 0 1 0 1

**4.3**

bitmap of “AAAI Pres” and language “english” is: 100010

In order to find the total sales of “English” books published by “AAAI Press”:

we can filter out **T** from fact tables by bitmap index = 100010

then Select SUM(Sales) from **T**

**Part 3**

**Question 5**

**5.1**

create view global\_view\_book

as

select t1.title,t1, t1.authors,

t1.publisher,t1.isbn13, t1.pages from Book1 t1

left outer join Book2 t2 on t2.isbn13=t1.isbn13

left outer join Book3 t3 on t3.ISBN13=t1.isbn13

left outer join Book4 t4 on t4.ISBN13=t1.isbn13

So our global view is:

global\_view\_book (id, title, authors, publisher, isbn13, pages)

**5.2**

1）Different tables from the gobal view may the column is same data, but data type is different.

For example: maybe book1 table’s isbn13 is all digital values, but some isbn13 of book2 is strings.

For example: book3.date and boo4.Publication\_Date ’s format is different.

Handle: first we must make sure we change the tables’s join columns same data type. If the format is different ,we write function /(or use built-in function in orcale/mysql )to format the according column to same format.

2）structural heterogeneity

For example :book2. publication\_year, publication\_month, publication\_day and book3.date  
Handle:we write function /(or use built-in function in orcale/mysql )to combine the different tables’s columns into one, make all columns the same.

3）structural heterogeneity

For example:book1. publisher and book2. publisher\_name

We can use unique naming when import or make join operations.

**Question 6**

**6.1**

select count(\*) from book3

where mod(ID, 100) = 0

**6.2**

declare @col varchar(50), @cmd varchar(max)

DECLARE getinfo cursor for

SELECT c.name FROM sys.tables t JOIN sys.columns c ON t.Object\_ID = c.Object\_ID

WHERE t.Name = ‘book3' and mod(ID,100)=0

OPEN getinfo

FETCH NEXT FROM getinfo into @col

WHILE @@FETCH\_STATUS = 0

BEGIN

SELECT @cmd = 'IF NOT EXISTS (SELECT top 1 \* FROM ADDR\_Address WHERE [' + @col + '] IS NOT NULL) BEGIN print ''' + @col + ''' end'

EXEC(@cmd)

FETCH NEXT FROM getinfo into @col

END

CLOSE getinfo

DEALLOCATE getinfo

**6.3**

code is at:

Set 6.2 result is default\_number

Select 6.1 result is total\_number

DPMO=[default\_number/（ total\_number\*10000）]\*1000000=50

**Question 7**

**7.1**

code is at: q7\_I\_II.py

The edit distance is :39

**7.3**

code is at: q7\_I\_II.py

The Jaccard distance is 0.375

**7.3**

code is at: Q7\_III.py

the linkage-pair file we generated is at: out/b1\_b2\_linkage.csv

tp, fp, fn= 212 937 20

precision= 0.18450826805918188

recall= 0.9137931034482759

f\_measure= 0.3070238957277335