Harsha Joshi
45561623

Final Assignment Report

Part 1

Question 1.1

The following is an implementation of a SQL query that could return the given query in this question:

```
SELECT Rating_Rank, Title, Publisher, Pages
FROM (
  SELECT RANK() OVER (ORDER BY RatingValue DESC) AS Rating_Rank, Title, Publisher, Pages
  FROM Book3
  ORDER BY RatingValue DESC
  LIMIT 15
) AS Highest_Rated
UNION ALL
SELECT Rating_Rank, Title, Publisher, Pages
FROM (
  SELECT RANK() OVER (ORDER BY RatingValue ASC) AS Rating_Rank, Title, Publisher, Pages
  FROM Book3
  ORDER BY RatingValue ASC
  LIMIT 10
) AS Lowest_Rated
ORDER BY Rating_Rank DESC;
```

Though this has not been physically tested via the dataset, this can be logically justified as follows:

1. To simplify the expression of finding the "top 15 books that have the highest rating/10 books that have the lowest rating" Rating_Rank is utilised.
2. The RANK() OVER keywords are utilised together in conjunction with ORDER BY, to add another column to the result set. Essentially this new column is added to sort the result via the specified highest → lowest (when ORDER BY RatingValue DESC is utilised) and lowest → highest (when ORDER BY RatingValue ASC is utilised). (Koiden, 2021)
3. Further, ORDER BY RatingValue DESC is used in terms of getting the top-rated books (limited to 15 by LIMIT 15) as DESC sorts the database from (highest → lowest), and so lowest-rated books (limited to 10 by LIMIT 10) as ASC sorts the database from (lowest → highest). Therefore, the top 15 tuples will be the highest of the highest after ordered and the top 10 tuples will be the lowest of the lowest after ordered accordingly. (W3Schools, 2023)
4. It is to be noted that LIMIT (value) only takes the top number of rows, so it is important to sort the database in the relevant order before utilising the LIMIT keyword.
5. Next, the UNION ALL keyword was utilised to make sure both of these conditions were met, before sorting the overall rank of the two sub-queries (denoted as Highest_Rated and Lowest_Rated with the AS keyword) in descending order (highest → lowest). (W3Schools, 2023)
6. Because the overall Rating_Rank must be sorted in descending order, this is used at the very end of the query as the final step.

Question 1.2

Book3 is the only table schema that is used to answer the query, because the query explicitly refers to the attributes: RatingValue only available in the Book3 schema and none of the other table schemas.

Question 2.1

To handle Book3, vertical fragmentation should be implemented. This is because the queries presented on the distributed database (A) are relevant to specific categories.

For example, the first dot point specifically points to book title and its relevant author information, based on publisher. The attributes Title, Author1, Author2, Author3, Publisher from Book3 specifically refer to the first dot point. This same logic was applicable to the second dot point stated in the question, whereby ID, Date, Pages, HardcoverPrice, EbookPrice are the attributes needed from Book3. It is also evident that book vertical fragments below must include ID it is the primary key for Book3 and to maintain constructiveness in vertical fragmentation (Hardini, 2020).

It is also important to note that all attributes in Book3, even those not directly relevant to the queries presented in the dot points, must be organised onto one of the vertical fragments. This is because the whole dataset in Book3 must be completely fragmented, and its attributes must be available in one of the fragments. Below is the implementation of this to demonstrate this:

- Given Schema:
  Book3[**ID**, Title, Author1, Author2, Author3, Publisher, ISBN13, Date, Pages, ProductDimensions, SalesRank, RatingsCount, RatingValue, PaperbackPrice, HardcoverPrice, EbookPrice, AudiobookPrice]
- Vertical Fragment 1 Schema: relevant to first dot point
  Book3_VF_Schema1[**ID**, Title, Author1, Author2, Author3, Publisher, SalesRank, RatingsCount, RatingValue]
- Vertical Fragment 2 Schema: relevant to second dot point
  Book3_VF_Schema2[**ID**, Date, Pages, HardcoverPrice, EbookPrice, ProductDimensions, PaperbackPrice, AudiobookPrice]

The categories of the first few attributes in each of the vertically fragmented schemas, represent the query relevant to their respective dot point along with the primary key. Though, it is evident that attributes not directly linked to the dot points have been sorted in order of how relevant they are to the existing/query-relevant attributes. For example, Book3_VF_Schema_2 contains ProductDimensions, PaperbackPrice and AudiobookPrice – all three of which are directly relevant to the existing/query-relevant attributes HardcoverPrice and EbookPrice.

Question 2.2

The given fragments and their assigned predicates are as follows:

- Fragment 1: $\text{pages} \leq 200$ → Predicate 1 (P_1)
- Fragment 2: $200 < \text{pages} \leq 600$ → Predicate 2 (P_2)
- Fragment 3: $\text{pages} > 800$ → Predicate 3 (P_3)

It is evident that these predicates are invalid because the predicates do not account for when the value of pages in the database is between $600 < \text{pages} \leq 800$. Therefore, while there is disjointness, there is no completeness or reconstructability for these predicates – which are all three conditions that must be best for successful horizontal fragmentation of a database. Hence, valid predicates must be found via the minterm predicates, using the existing values from above.

Given the fact that there are 3 predicates presented, the way to calculate the number of minterm predicate possibilities is: $2^{\text{number of predicates}}$, which is equal to 8 in this case ($2^{\text{number of predicates}} = 2^3 = 8$). This is because each predicate must undergo the following general equation: $P_1^* \wedge P_2^* \wedge P_3^*$, where $P_1^* = P_1 \vee \neg P_1$, and so forth for the other predicates. The process is shown in the table below:

First, a set of simple predicates must be inputted:

$$P = \{P_1, P_2, P_3\}$$

Next, all possible minterm predicates ($M_i = M_3$) must be enumerated as follows, for this case of 3 predicates:

$$M_3 = P_1^* \wedge P_2^* \wedge P_3^*$$

- Where: $P_1^* = P_1 \vee \neg P_1$, $P_2^* = P_2 \vee \neg P_2$, $P_3^* = P_3 \vee \neg P_3$

Thus, the following combinations are possible, as listed in the table below under the heading minterm predicates and labelled accordingly. It is to be noted that for the $200 <$ pages part of Predicate 2, was written as follows for clarity: $P_2 = \text{pages} > 200$ and $\neg P_2 = \text{pages} \leq 200$.

Minterm Predicate	$\boldsymbol{P_1}$	$\boldsymbol{P_2}$		$\boldsymbol{P_3}$
M1: $P_1 \wedge P_2 \wedge P_3$	$\text{pages} \leq 200$	$\text{pages} > 200$	$\text{pages} \leq 600$	$\text{pages} > 800$
M2: $P_1 \wedge P_2 \wedge \neg P_3$	$\text{pages} \leq 200$	$\text{pages} > 200$	$\text{pages} \leq 600$	$\text{pages} \leq 800$
M3: $P_1 \wedge \neg P_2 \wedge P_3$	$\text{pages} \leq 200$	$\text{pages} \leq 200$	$\text{pages} > 600$	$\text{pages} > 800$
M4: $P_1 \wedge \neg P_2 \wedge \neg P_3$	$\text{pages} \leq 200$	$\text{pages} \leq 200$	$\text{pages} > 600$	$\text{pages} \leq 800$
M5: $\neg P_1 \wedge P_2 \wedge P_3$	$\text{pages} > 200$	$\text{pages} > 200$	$\text{pages} \leq 600$	$\text{pages} > 800$
M6: $\neg P_1 \wedge P_2 \wedge \neg P_3$	$\text{pages} > 200$	$\text{pages} > 200$	$\text{pages} \leq 600$	$\text{pages} \leq 800$
M7: $\neg P_1 \wedge \neg P_2 \wedge P_3$	$\text{pages} > 200$	$\text{pages} \leq 200$	$\text{pages} > 600$	$\text{pages} > 800$
M8: $\neg P_1 \wedge \neg P_2 \wedge \neg P_3$	$\text{pages} > 200$	$\text{pages} \leq 200$	$\text{pages} > 600$	$\text{pages} \leq 800$

It is immediately evident that M1, M2, M7, M8 are all invalid options of minterm predicates, because it is not possible for pages to be greater than 200 while also simultaneously being less than and equal to 200. Looking at M4 a similar logic can be applied; it is not possible for pages to be greater than 600, while also being less than or equal to 800, because 600 does not equal 800 and nor is 600 greater than 800. Further, this is evident for M5 as well, whereby pages cannot be greater than 800, while also being less than or equal to 600. This is also because 800 is not equal to 600 and nor is it less than 600.

Hence, it is evident that M3 and M6 are the only two valid minterm predicates and will be used to generate a new set of predicates for Book3. This is primarily due to the bounds presented via the predicates are all valid and more importantly can co-exist simultaneously at the same time.

- M3: $P_1 \wedge \neg P_2 \wedge P_3 = \text{pages} \leq 200 \wedge (\text{pages} \leq 200 \wedge \text{pages} > 600) \wedge \text{pages} > 800$
- M6: $\neg P_1 \wedge P_2 \wedge \neg P_3 = \text{pages} > 200 \wedge (\text{pages} > 200 \wedge \text{pages} \leq 600) \wedge \text{pages} \leq 800$

Thus, the new horizontal fragmentation bounds for consideration are:

	From M3	**From M6**
P_1	$\text{pages} \leq 200$	$\text{pages} > 200$
P_2	$\text{pages} \leq 200 \land \text{pages} > 600$	$\text{pages} > 200 \land \text{pages} \leq 600$
P_3	$\text{pages} > 800$	$\text{pages} \leq 800$
• NOTE: the highlighted bounds are what is originally presented in the question.		

It is evident in assessing the above, that predicate 1 and 3 from M3 are valid predicates which will consider books with pages less than or equal to 200 and those greater than 800. However, for predicate 2, it becomes evident that a combination of values presented in M6 are required to encapsulate the page values between 201-800 (inclusive). Hence, a valid combination of M6's P_2 and P_3 will ensure that this range is met. M6's P_1 does not need to be considered because this is repeated in P_2. Thus, becomes apparent that to capture this page range $P_2 = \text{pages} > 200 \land \text{pages} \leq 800$, also denoted as $200 < \text{pages} \leq 800$.

Finally, the fragments and predicates are as follows, given the minterm consideration for Book3.

- Fragment 1: $\text{pages} \leq 200$ → Predicate 1 (P_1)
- Fragment 2: $200 < \text{pages} \leq 800$ → Predicate 2 (P_2)
- Fragment 3: $\text{pages} > 800$ → Predicate 3 (P_3)

The new fragments/predicates derived from the relevant minterms can be used explain the insertion process for a new record. To insert a new tuple or record into Book3, it must first be considered that 'pages' attribute is not the primary key and that Book3 was not fragmented via its primary key (ID). After all the fragments are checked to make sure the primary key does not exist, the new tuple's 'pages' value must be checked against the new predicates and correctly inserted into the right fragment.

For example, if a tuple containing a 'pages' value of 700 was to be inserted into Book3, it would be compared against the bounds presented by the predicates. Regarding the newly formed predicates the whole tuple would be inserted into fragment 2 successfully, though the old predicates would not be able to insert this tuple because none of the predicates included the 'pages' attribute bounds for 601-800 (inclusive).

Part 2

Question 3.1

A star schema is ideally utilised as follows to represent the given data warehouse. It is evident that the star schema is based on given values presented in the Question. However, this idea can be expanded out based on if more datasets are added to overall or more attributes need to be included:

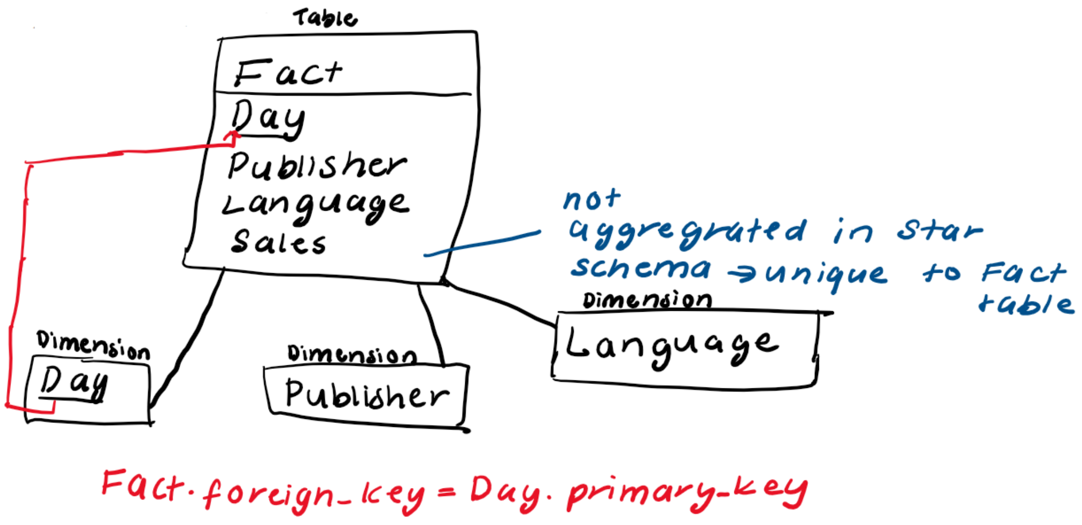


Table
Fact
Day
Publisher
Language
Sales
not aggregrated in star schema ⇒ unique to Fact table
Dimension
Day
Dimension
Publisher
Dimension
Language
Fact.foreign_key = Day.primary_key

A star schema is "a multi-dimensional data model used to organise data in a database so that it is easy to understand and analyse." It's "design is optimised for querying large data sets" (Databricks, 2023). It is implemented in the diagrammatic format above in correlation with it's conventionally drawing. This is specifically useful for this dataset because the current Book1-4 dataset can be expanded upon and catalogued in a very specific way and order. The dimensions mentioned can be added and expanded upon to include edition, series, etc., while the sales value in the Fact table remains the only one that does not have its own dimension – it is uniquely aggregated only the Fact table across all datasets. Further, it is to be noted that the Fact table includes a foreign_key to the Day Dimension's primary_key (which itself is linked to the Book1-4 datasets). Therefore, the Fact table in accordance with the star schema and will be able to successfully aggregate sales data for these booksets, provided the Day, Publisher and Language information.

Question 4.1

Bitmap indexes are "widely used in data warehousing applications, which have large amounts of data and ad hoc queries but a low level of concurrent transactions" (Oracle, 2023). Hence the advantages are as follows from the Oracle docs:

- Reduced response time for large classes of ad hoc queries
- A substantial reduction of space usage compared to other indexing techniques
- Dramatic performance gains even on hardware with a relatively small number of CPUs or small amount of memory
- Very efficient maintenance during parallel DML and loads

However, it is evident that the disadvantages of bitmap indexes are that they are not a reliable method of indexes for databases which undergo high rates of concurrent transactions and have large volumes of data. The response time for queries will increase, despite less storage space used by bitmap indexes and the small CPUs/memory would not undergo dramatic performance gains, if given a highly concurrent-transaction database. Hence, this will inadvertently affect the bitmap indexes ability to efficiently maintain the database during parallel DML and loads. A good example of a database system which would not be reliable for bitmap indexes would be that of a banking/financial institution. This is primarily because various concurrent transactions occur at high frequency over large volumes of financial data.

Question 4.2

A bitmap index assigns 0 to all the values that are not the same as the target value that is to be acquired from the column, which is Publisher and Language in this case, otherwise 1 is assigned to the values which are the same as the target value. The number of distinct values in each column directly correlates to the number of possible bitmap indexes possible for that column – i.e. Publisher containing 4 distinct values will allow it to contain 4 variations of possible bitmap indexes and similarly, Language containing 2 distinct values will allow this column to have 2 variations of possible bitmap indexes. This is shown in the table below, with the second row referring to the target value to be achieved via its respective bitmap index. It is important to note that the extract bitmap indexes below must reflect the accurate order and positioning of values from the original schema.

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

Question 4.3

From Question 4.2, it is evident that the bitmap indexes for all their target values have been specified in the second row in the table. Thereby, the relevant columns representing the bitmap indexes for "English" (from the Language column) and "AAAI Press" (from the Publisher column) need to be selected such that the total sales of "English" books published by "AAAI Press" can be found. This will be expanded upon in the steps below and the process of calculating the total sales values has been detailed below utilising the two relevant bitmap indexes.

1. Select relevant bitmap indexes for "English" (from the Language column) and "AAAI Press" (from the Publisher column) from Question 4.2

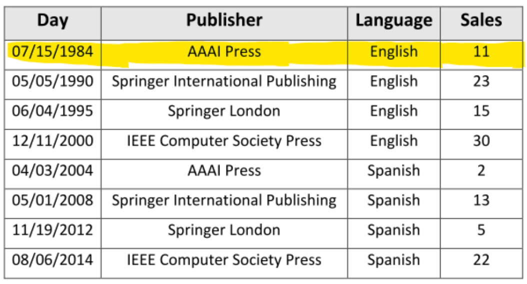
Language	**Publisher**
English	AAAI Press
1	1
1	0
1	0
1	0
0	1
0	0
0	0
0	0

2. Hence, matching all the relevant '1' values from the bitmap index for English along with the relevant '1' values from the bitmap index for AAAI Press, gives us the following. For the '1's to be relevant they must appear in the same row for both bitmap indices for English (Language) and AAAI Press (Publisher).

Language	**Publisher**

English	AAAI Press
1	1
1	0
1	0
1	0
0	1
0	0
0	0
0	0

3. It is found that only the very first row of the bitmap indexes for English and AAAI Press are relevant. Thus, utilising this knowledge, the original schema is consulted to find 'sales' value stated in the very first row as this is the only sales value that matches with the intersection of the bitmap indexes for English and AAAI Press respectively (highlighted in yellow). It is important that the '1' values for "English" and "AAAI Press" overlap and intersect to find the appropriate 'sales' values – otherwise, it is possible that multiple, incorrect sales values will be returned because both these parameters have not been met. For this reason, it was important that the original bitmap indexes constructed in Question 4.2 was accurate and in positional order of the original schema.



Day	Publisher	Language	Sales
07/15/1984	AAAI Press	English	11
05/05/1990	Springer International Publishing	English	23
06/04/1995	Springer London	English	15
12/11/2000	IEEE Computer Society Press	English	30
04/03/2004	AAAI Press	Spanish	2
05/01/2008	Springer International Publishing	Spanish	13
11/19/2012	Springer London	Spanish	5
08/06/2014	IEEE Computer Society Press	Spanish	22

4. In correlating the bitmap indexes for both English and AAAI Press and the fact that only the very first row of the provided schema reflected the intersection successfully – the total sales value for "English" books published by the "AAAI Press" should be 11, as demonstrated by the highlighted row. It is important that once the relevant row/tuple has been identified (the first row in this case), only then is it possible for the correct sales value to be returned by the bitmap indexes.
5. Therefore, the total sales value for "English" books published by the "AAAI Press" is 11.

Part 3

Question 5.1

The Global as a View approach (GAV) is "in which the global schema is described in terms of the local schemas". In these types of systems, "the contents of each relation R in the target schema G are specified through a query (view) V over the combined schemas of the sources" (Katsis et. al, 2009). It is also evident that duplicate attributes must be avoided when designing a global schema to ensure the same data is not repeated in the global schema – this will likely cause structural and semantic heterogeneity issues which will be discussed further.

In this case, the global schema is designed as follows:

- Given Schemas:
  - Book1[<u>**id**</u>, title, authors, pubyear, pubmonth, pubday, edition, publisher, isbn13, language, series, pages]
  - Book2[<u>**id**</u>, book_title, authors, publication_year, publication_month, publication_day, edition, publisher_name, isbn13, language, series, pages]

- Book3[**<u>ID</u>**, Title, Author1, Author2, Author3, Publisher, ISBN13, Date, Pages, ProductDimensions, SalesRank, RatingsCount, RatingValue, PaperbackPrice, HardcoverPrice, EbookPrice, AudiobookPrice]
- Book4[**<u>ID</u>**, Title, UsedPrice, NewPrice, Author, ISBN10, ISBN13, Publisher, Publication_Date, Pages, Dimensions]

- Combining All Schemas (including duplicate primary keys):
  - AllSchemas[**<u>id</u>**, title, authors, pubyear, pubmonth, pubday, edition, publisher, isbn13, language, series, pages, **<u>id</u>**, book_title, authors, publication_year, publication_month, publication_day, edition, publisher_name, isbn13, language, series, pages, **<u>ID</u>**, Title, Author1, Author2, Author3, Publisher, ISBN13, Date, Pages, ProductDimensions, SalesRank, RatingsCount, RatingValue, PaperbackPrice, HardcoverPrice, EbookPrice, AudiobookPrice, **<u>ID</u>**, Title, UsedPrice, NewPrice, Author, ISBN10, ISBN13, Publisher, Publication_Date, Pages, Dimensions]
- Removing Potential Duplicate Attributes:
  - AllSchemas_DuplicatesIdentified[**<u>id</u>**, title, authors, pubyear, pubmonth, pubday, edition, publisher, isbn13, language, series, pages, **<u>id</u>**, book_title, authors, publication_year, publication_month, publication_day, edition, publisher_name, isbn13, language, series, pages, **<u>ID</u>**, Title, Author1, Author2, Author3, Publisher, ISBN13, Date, Pages, ProductDimensions, SalesRank, RatingsCount, RatingValue, PaperbackPrice, HardcoverPrice, EbookPrice, AudiobookPrice, **<u>ID</u>**, Title, UsedPrice, NewPrice, Author, ISBN10, ISBN13, Publisher, Publication_Date, Pages, Dimensions]
  - AllSchemas_DuplicatesRemoved[**<u>id</u>**, title, authors, pubyear, pubmonth, pubday, edition, publisher, isbn13, language, series, pages, Date, ProductDimensions, SalesRank, RatingsCount, RatingValue, PaperbackPrice, HardcoverPrice, EbookPrice, AudiobookPrice, UsedPrice, NewPrice, Author, ISBN10]
    - NOTE: the attributes 'Date' (Book3) has been kept as it is not clear exactly what they refer to in terms of the given database system. This is most likely the publication date, but this could refer to any date.
- Final Global Schema:
  - GlobalSchemaAllBooks[**<u>id</u>**, title, authors, pubyear, pubmonth, pubday, edition, publisher, isbn13, language, series, pages, Date, ProductDimensions, SalesRank, RatingsCount, RatingValue, PaperbackPrice, HardcoverPrice, EbookPrice, AudiobookPrice, UsedPrice, NewPrice, Author, ISBN10]
- Final Global Schema (as a View): in accordance with the Global as View approach defined above
  - GlobalSchemaAllBooks_View[**<u>id</u>**, title, authors, pubyear, pubmonth, pubday, edition, publisher, isbn13, language, series, pages, Date, ProductDimensions, SalesRank, RatingsCount, RatingValue, PaperbackPrice, HardcoverPrice, EbookPrice, AudiobookPrice, UsedPrice, NewPrice, Author, ISBN10]

Question 5.2

Structural heterogeneity is when a database system loses valuable attributes when it is part of a global schema amongst other databases. An example of this occurring above is in Book3: whereby the three attributes Author1, Author2, Author3 are condensed down to authors from Book1. It is also to be noted that along with causing data integration issues, this can also be expanded into causing issues for primarily query processing (Harder et. al, 1999).

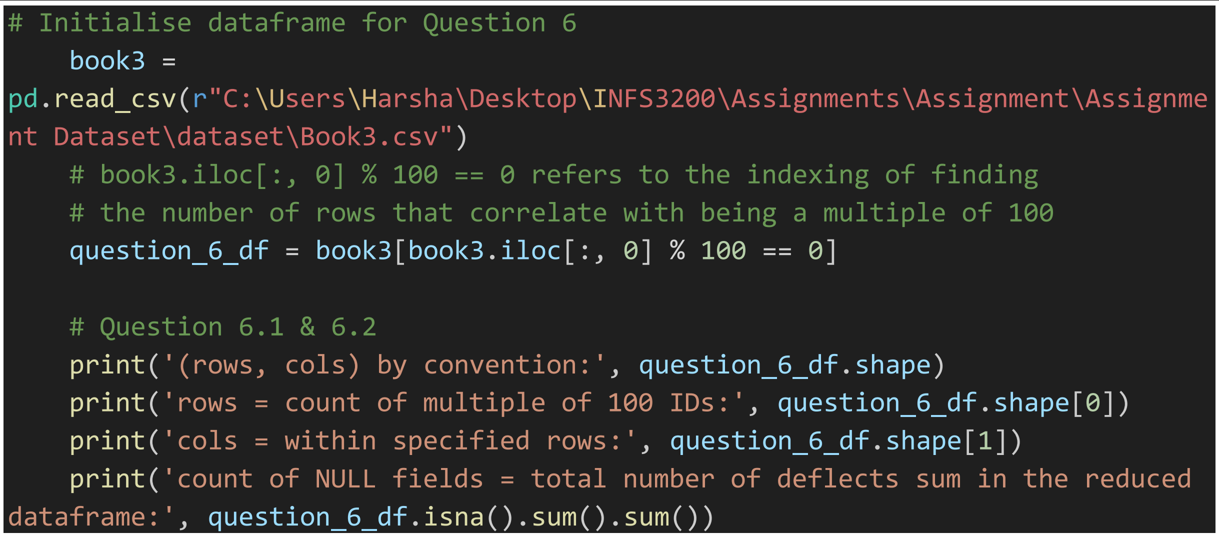
Question 5.3

Semantic heterogeneity is "when schema or data set for same domain is developed by independent parties which leads to differences in meaning, interpretation and intended use of the same or related data" (Kumar, 2023). An example of this occurring above is in terms of edition from Book1 and Book2: whereby though this attribute looks the same structurally in both schema, the ways this are represented could be different – for example, one of the table schemas may have considered limited edition book release data, while the other table schema may not have done so. If merged together in a global schema, it difficult to identify which table schema's edition data was excluded from the final available dataset from the schema. It is also to be noted that along with causing data integration issues, this can be expanded into causing issues for primarily application development (Kumar, 2023).

Part 4

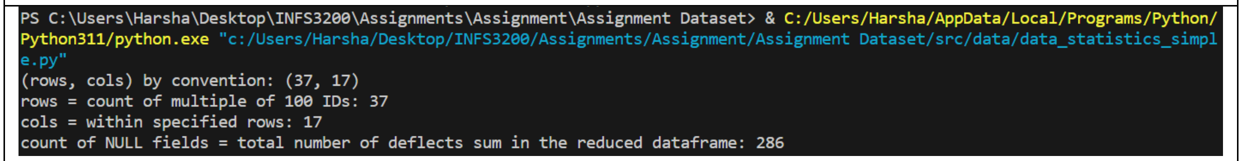
Question 6.1 & Question 6.2

The code snapshot below demonstrates the relevant number of records whose ID is a multiple of 100 and also those fields that contain NULL values (from the data_statistics_simple.py() file). It is important to note that the python pandas package was utilised (`import pandas as pd`) in order to achieve these values. Further, the values found will directly be utilised to calculate the EMPO (error per million opportunities).



```
# Initialise dataframe for Question 6
    book3 = 
pd.read_csv(r"C:\Users\Harsha\Desktop\INFS3200\Assignments\Assignment\Assignme
nt Dataset\dataset\Book3.csv")
    # book3.iloc[:, 0] % 100 == 0 refers to the indexing of finding
    # the number of rows that correlate with being a multiple of 100
    question_6_df = book3[book3.iloc[:, 0] % 100 == 0]

    # Question 6.1 & 6.2
    print('(rows, cols) by convention:', question_6_df.shape)
    print('rows = count of multiple of 100 IDs:', question_6_df.shape[0])
    print('cols = within specified rows:', question_6_df.shape[1])
    print('count of NULL fields = total number of deflects sum in the reduced 
dataframe:', question_6_df.isna().sum().sum())
```



```
PS C:\Users\Harsha\Desktop\INFS3200\Assignments\Assignment\Assignment Dataset> & C:/Users/Harsha/AppData/Local/Programs/Python/
Python311/python.exe "c:/Users/Harsha/Desktop/INFS3200/Assignments/Assignment/Assignment Dataset/src/data/data_statistics_simpl
e.py"
(rows, cols) by convention: (37, 17)
rows = count of multiple of 100 IDs: 37
cols = within specified rows: 17
count of NULL fields = total number of deflects sum in the reduced dataframe: 286
```

Therefore, it is evident that, the number of records whose ID is a multiple of 100 is 37, while the number of values with NULL fields are 286 within the Question 6 formulated data frame.

This is due to the commands presented via the implementation of pandas. The read_csv command is an in-bult pandas command which loads the relevant csv file date and converts this into a dataframe to be utilised for further calculation. The .iloc[; 0] is directly indexing into the 1st column (ID column) of Book3 and finding the relevant rows which are multiples of 100 – this aspect is demonstrated by the use of the modulo function. Similarly, the command question_6_df.shape[0] was utilised to find the number of rows which are multiples of 100. 0 is used to index into the first value of the (37, 17) tuple because by convention the tuple is returned as (row value, column value).

Further, this whole command is encapsulated within Book3 itself to return a new reduced Question 6 formulated data frame which can be utilised for further calculation to find the number of NULL field values. Finally, another command .isna().sum().sum() counts the number of NULL field values for both rows and columns. For this reason .sum() is utilised twice in this implementation.

Question 6.3

Hence, it is evident that with the values found from above the EMPO can be calculated. According to SixSigmaDaily – an online data science resource, EMPO can also be referred to as DPMO (defects per million opportunities), for which the formula is (SixSigmaDaily, 2020):

$$\begin{aligned} DPMO \;\&\; EMPO(\text{NULL values}) &= \left(\frac{total\ number\ of\ defects\ found\ in\ a\ sample}{total\ number\ of\ defect\ opportunities\ per\ unit\ in\ the\ sample}\right) \times 1000000 \\ &= \left(\frac{total\ number\ of\ defects\ found\ in\ a\ sample}{sample\ size\ \times number\ of\ defect\ opportunities\ per\ unit\ in\ the\ sample}\right) \\ &\times 1000000 \end{aligned}$$

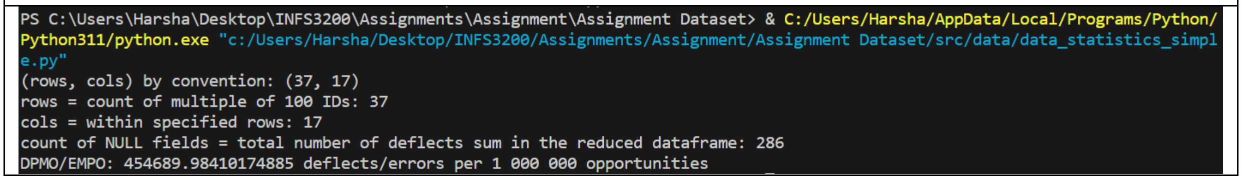
- Where (from code):
  - $total\ number\ of\ defects\ found\ in\ a\ sample = \text{count of NULL fields}$
  - $sample\ size = \text{count of multiple of 100 IDs}$
  - $number\ of\ defect\ opportunities\ per\ unit\ in\ the\ sample = \text{columns within specified rows}$

This formula was utilised directly to calculate the DPMO which is the same as the EMPO for NULL values. Hence, the EMPO for the NULL field values is also demonstrated by applying the same formula. The code demonstrates this below and reassigns values directly from the question_6_df for clarity when it is used in the formula.

```
 # Initialise dataframe for Question 6
    book3 = 
pd.read_csv(r"C:\Users\Harsha\Desktop\INFS3200\Assignments\Assignment\Assignme
nt Dataset\dataset\Book3.csv")
    # book3.iloc[:, 0] % 100 == 0 refers to the indexing of finding
    # the number of rows that correlate with being a multiple of 100
    question_6_df = book3[book3.iloc[:, 0] % 100 == 0]

    # Question 6.1 & 6.2
    print('(rows, cols) by convention:', question_6_df.shape)
    print('rows = count of multiple of 100 IDs:', question_6_df.shape[0])
    print('cols = within specified rows:', question_6_df.shape[1])
    print('count of NULL fields = total number of deflects sum in the reduced 
dataframe:', question_6_df.isna().sum().sum())

    #Question 6.3
    total_number_deflects = question_6_df.isna().sum().sum()
    sample_size = question_6_df.shape[0]
    number_of_deflect_opportunities = question_6_df.shape[1]
    dpmo_empo_null = (total_number_deflects / (sample_size *
number_of_deflect_opportunities)) * 1000000
    print('DPMO/EMPO:', dpmo_empo_null, 'deflects/errors per 1 000 000 
opportunities')
```



```
PS C:\Users\Harsha\Desktop\INFS3200\Assignments\Assignment\Assignment Dataset> & C:/Users/Harsha/AppData/Local/Programs/Python/
Python311/python.exe "c:/Users/Harsha/Desktop/INFS3200/Assignments/Assignment/Assignment Dataset/src/data/data_statistics_simpl
e.py"
(rows, cols) by convention: (37, 17)
rows = count of multiple of 100 IDs: 37
cols = within specified rows: 17
count of NULL fields = total number of deflects sum in the reduced dataframe: 286
DPMO/EMPO: 454689.98410174885 deflects/errors per 1 000 000 opportunities
```

Hence, the EMPO (or DPMO) for the given sample size values within 37 rows by 17 columns, correlating to the IDs which are multiples of 100 is: 454689.98410174885 errors/deflects per 1 000 000 opportunities, which approximately equals to 454690 errors per 1 000 000 opportunities. This value is extremely important in terms of data quality issues because it demonstrates a quantitative measurement of how many values are to be updated if the database is being maintained, or values to be discarded if the database is not in use anymore.

Question 7.1

Between the two functions provided, it is evident that Jaccard distance is most likely to regard them as similar. This is because edit distance takes the spaces, commas etc. as part of the string itself, while typically Jaccard distance will compare string-by-string in groups of 3 for example. In this case the code below demonstrates this (from ed_calculation_detail.py()):

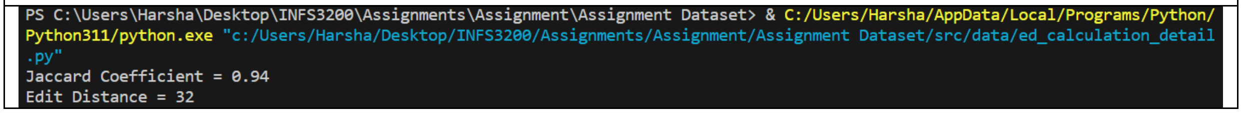
```
def calc_ed_sim(str1, str2):
    if str1 == str2:
        return 1
    ed = calc_ed(str1, str2)
    print('Edit Distance =', ed)
    return 1 - (ed / max(len(str1), len(str2)))


def calc_ed(str1, str2):
    ed = 0

    if len(str1) > len(str2):
        str1, str2 = str2, str1

    distances = range(len(str1) + 1)
    for i2, c2 in enumerate(str2): #longer string
        longer_string_distances_ = [i2+1]
        for i1, c1 in enumerate(str1): #shorter string
            if c1 == c2:
                longer_string_distances_.append(distances[i1])
            else:
                longer_string_distances_.append(1 + min((distances[i1], 
distances[i1 + 1], longer_string_distances_[-1])))
        distances = longer_string_distances_
        ed = distances[-1]
    out2 = calc_jaccard(str1, str2, 2)
    print("Jaccard Coefficient =", out2)

    return ed
```



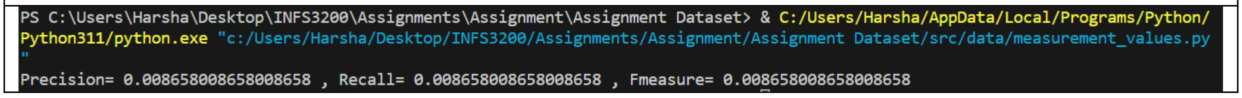
```
PS C:\Users\Harsha\Desktop\INFS3200\Assignments\Assignment\Assignment Dataset> & C:/Users/Harsha/AppData/Local/Programs/Python/
Python311/python.exe "c:/Users/Harsha/Desktop/INFS3200/Assignments/Assignment/Assignment Dataset/src/data/ed_calculation_detail
.py"
Jaccard Coefficient = 0.94
Edit Distance = 32
```

Hence, it is evident that the Jaccard coefficient = 0.94, while the Edit Distance = 32 for the given string. Thereby, the Jaccard coefficient regarded these given strings are more similar in comparison to edit distance calculation.

Question 7.2

The data linkage between Book1 and Book2 data is performed as follows, with a similarity (or threshold value) of 0.75 from measurement_values.py():

```
def calc_measure(results):
    benchmark = load_benchmark()
    if len(results) == 0:
        print('Precision = 0, Recall = 0, Fmeasure = 0')
        return
    count = 0
    for pair in results:
        if pair in benchmark:
            count = count + 1
    if count == 0:
        print('Precision=0, Recall=0, Fmeasure=0')
        return
    precision = count / len(results)
    recall = count / len(benchmark)
    f_measure = 2 * precision * recall / (precision + recall)
    print("Precision=", precision, ", Recall=", recall, ", Fmeasure=",
f_measure)
    return
```



```
PS C:\Users\Harsha\Desktop\INFS3200\Assignments\Assignment\Assignment Dataset> & C:/Users/Harsha/AppData/Local/Programs/Python/
Python311/python.exe "c:/Users/Harsha/Desktop/INFS3200/Assignments/Assignment/Assignment Dataset/src/data/measurement_values.py
"
Precision= 0.008658008658008658 , Recall= 0.008658008658008658 , Fmeasure= 0.008658008658008658
```

Hence, the values are completely precise, recall and measure to 0.008658. Thus, all the provided book pairs do not match the given book pairs in the benchmark well.

References

Databricks (2023). Retrieved from https://www.databricks.com/glossary/star-schema#:~:text=A%20star%20schema%20is%20a,for%20querying%20large%20data%20sets

Hardini, D. P. S. (2020). Types of Fragmentation. Retrieved from https://sis.binus.ac.id/2020/12/11/types-of-fragmentation/#:~:text=In%20order%20to%20maintain%20constructiveness,of%20data%20or%20important%20transactions

Härder, T., Sauter, G. & Thomas, J. The intrinsic problems of structural heterogeneity and an approach to their solution. The VLDB Journal 8, 25–43 (1999). https://doi.org/10.1007/s007780050072

Katsis, Y., Papakonstantinou, Y. (2009). View-based Data Integration. In: LIU, L., ÖZSU, M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_1072

Koidan, K. (2021). Retrieved from https://learnsql.com/blog/how-to-rank-rows-sql/

Kumar, A. (2023). Retrieved from https://www.geeksforgeeks.org/semantic-heterogeneity-in-dbms/

Oracle (2023). Retrieved from https://docs.oracle.com/cd/A84870_01/doc/server.816/a76994/indexes.htm#97322

SixSigmaDaily (2020). Retrieved from https://www.sixsigmadaily.com/dpu-dpmo-ppm-and-rty/

W3Schools (2023). Retrieved from https://www.w3schools.com/sql/sql_ref_order_by.asp

W3Schools (2023). Retrieved from https://www.w3schools.com/sql/sql_ref_union_all.asp#:~:text=The%20UNION%20ALL%20command%20combines,statements%20(allows%20duplicate%20values