This report describes the process of creating a local and global data sources over a relational database and subsequently querying those. A section also examines optimizing query performance.

Note: The program code is provided individually for each question in the root folder. The program can be executed by following the "README.md" instructions. The questions from the root directory contain method calls from the other files in the "sql" subfolder to make the code more manageable and readable. The global settings are provided in "globals.py" inside the root folder.

Question 1: Implement data sources in relational database using both non-materialized and materialized views. Provide your code.

The program provided in "q1.py" implements creating data sources based on existing (populated) dataset from assignment 2 using both materialized and non-materialized views (materialized view names are appended with the word "Materialized". The program executes the queries provided in "create_schema_basic.sql" in the root folder.

Question 2: Using GAV mappings, describe the global schema based solely on the sources.

The program provided in "q2.py" implements creating global schema based on sources from question 1 using both materialized and non-materialized views (materialized view names are appended with the word "Materialized". The program executes the queries provided in "create_schema_gav.sql" in the root folder. Non-materialized views are created using non-materialized sources, while materialized views are created using the materialized sources from the question 1.

Question 3: Provide queries for answering over global schema.

The program provided in "q3.py" implements queries overall global schema using both materialized and non-materialized views. The queries are provided in "queries.py" in the "sql" sub-folder. Queries are named with question number, query number and type (e.g., q3_query1_materialized refers to first query in question 3 run using the materialized views). The same naming convention will be followed in the subsequent questions. The results from queries in this question are shown in the following table.

Sr	Materialized (time)	Non-Materialized (time)
Query 1	0.74 seconds	2.70 seconds
Query 2	0.42 seconds	1.65 seconds

Query 1. Users with reputation more than 100 who have commented on at least 10 posts.

Sample output: Sorted on reputation (last value in displayed row).

```
++++++++++++++++++++++++++++++++++++++++++++++
Running Non-Materialized Version
++++++++++++++++++++++++++++++++++++++++++++++
(15811, 'Rinzwind', 12154, 296459)
(449, 'Oli', 2487, 291283)
(7035, 'Luis Alvarado', 1912, 210043)
(158442, 'muru', 13191, 195356)
(6969, 'Lekensteyn', 1882, 172793)
(14356, 'fossfreedom', 3126, 171996)
(147044, 'Radu Rădeanu', 1436, 167920)
(3940, 'Takkat', 2913, 141544)
(58612, 'ish', 1276, 139186)
(178692, 'steeldriver', 10072, 133434)
++++++++++++++++++++++++++++++++++++++++++++++
Total running time: 2.702834367752075 seconds
++++++++++++++++++++++++++++++++++++++++++++++
```

Query 2. Users whose display name starts with "john-" and who have never commented on any post with the tag "networking".

Sample output: Sorted on reputation (last value in displayed row).

```
++++++++++++++++++++++++++++++++++++++++++++
Running Non-Materialized Version
++++++++++++++++++++++++++++++++++++++++++++
(680049, 'john-g', 186)
(443695, 'john-aziz57', 101)
(977113, 'john-raymon', 101)
(1179002, 'john-hen', 101)
(898797, 'john-d0e', 21)
(1636601, 'john-hawthorn', 11)
(1625421, 'john-1990-', 1)
(965726, 'john-a', 1)
++++++++++++++++++++++++++++++++++++++++++++
Total running time: 1.6527774333953857 seconds
++++++++++++++++++++++++++++++++++++++++++++
```

Question 4: Provide the resulting queries after expanding queries using the GAV mappings over the sources.

The program provided in "q4.py" implements queries expansion to using GAV mapping over the sources using both materialized and non-materialized views. The queries are provided in "queries.py" in the "sql" sub-folder. The results from queries in this question are shown in the following table.

Sr	Materialized (time)	Non-Materialized (time)
Query 1	1.30 seconds	2.55 seconds
Query 2	0.57 seconds	1.63 seconds

Question 5: Optimize the queries from Q4 by eliminating redundant joins and sources where possible using views from Q1.

The program provided in "q5.py" implements queries by eliminating redundant joins and sources where possible using views from Q1 using both materialized and non-materialized views. The queries are provided in "queries.py" in the "sql" sub-folder. The results from queries in this question are shown in the following table. Eliminated sources are explained below. No joins look redundant.

Sr	Materialized (time)	Non-Materialized (time)
Query 1	1.30 seconds	2.69 seconds
Query 2	0.73 seconds	1.56 seconds

Query 1:

1. Changed from AllCommentsMaterialized GAV to CommentedOnMaterialized source as both contained the equivalent definition. This changes selection from GAV to source.
2. Shifted from AllComments GAV to CommentedOn source as both contained the equivalent definition. This changes selection from GAV to source.

Query 2:

1. Shifted from AllCommentsMaterialized GAV to CommentedOnMaterialized source as both contained the equivalent definition. This changes selection from GAV to source.
2. Shifted from AllComments GAV to CommentedOn source as both contained the equivalent definition. This changes selection from GAV to source.

Note: Instructions to run the code are provided in the README.md file in the root folder.