Advanced Database Architects

Group – 3

Group Members

Athul Rohan Anil Kumar Rajitha
Zaw Win Htet Naing
Srushti Deepak Borkar
Lakshmi Sahiti Varada

Submission Date: 20.02.2026

Table of Contents

ADVANCED DATABASE ARCHITECTS **0**

1. INTRODUCTION **2**

2. DATASET DESCRIPTION **2**

3. SYSTEM ARCHITECTURE OVERVIEW **2**

4. RELATIONAL DATABASE DESIGN (SQL SCHEMA) **3**

5. DATA CLEANING AND PREPROCESSING **4**

6. INEFFICIENT QUERY VS OPTIMIZED QUERY **5**

7. MQL/JAVASCRIPT DESIGN FOR UNSTRUCTURED DATA **6**

8. QUERY OPTIMIZATION AND PERFORMANCE BENCHMARKING **8**

Approach 1: Denormalized Table *8*

Approach 2: Normalized Schema with Indexing *8*

9. CROSS-DATABASE INTEGRATION **9**

10. VISUALIZATION AND ANALYTICAL INSIGHTS **9**

11. ETHICAL CONSIDERATIONS AND CONCLUSION **10**

12. REFERENCES **11**

1. Introduction

This project presents the design and implementation of a scalable meeting intelligence data pipeline based on the MeetingBank dataset. The primary objective was to transform complex semi-structured and unstructured meeting data into a structured, queryable, and analytically usable system.

The pipeline integrates both relational and non-relational database technologies to simulate a real-world data engineering workflow. Structured numerical metadata is stored in MySQL, while transcript-based unstructured data is stored in MongoDB. The final analytical layer integrates both systems into a unified dataset for visualization.

As Group 3, Advanced Database Architects, our focus was on schema normalization, indexing strategies, and performance benchmarking to evaluate database optimization techniques and scalability.

2. Dataset Description

The MeetingBank dataset (Hu et al., 2023) is a large-scale collection of over 1,300 city council meetings from six major U.S. cities. The dataset includes:

- Full-length meeting transcripts
- Human-written summaries
- Metadata (meeting ID, city, date, agenda information)
- Multimedia references

For this project, **Seattle** and **Long Beach City Council** meetings were selected as representative subset cities.

The dataset contains both structured and semi-structured formats. The transcript text represents unstructured data, requiring parsing and feature engineering before storage and analysis.

3. System Architecture Overview

The implemented pipeline follows a modular end-to-end architecture:

1. Data ingestion from MeetingBank repository
2. Data cleaning and preprocessing using Python
3. Structured data storage in MySQL
4. Unstructured transcript storage in MongoDB
5. Query optimization benchmarking
6. Cross-database integration

7. Visualization using Power BI

The architecture separates operational storage (SQL) from document storage (JavaScript), while maintaining the ability to merge both systems during the analytical stage.

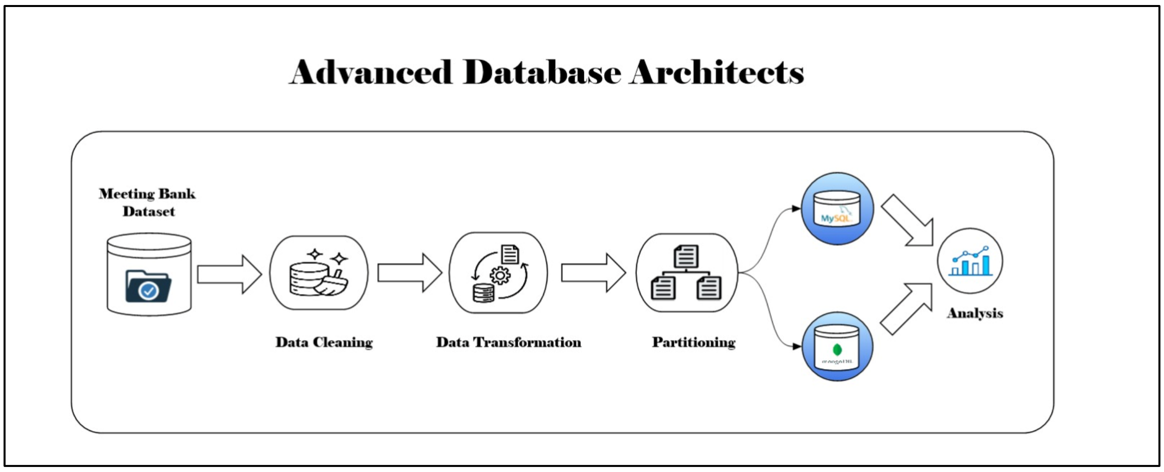


Figure 1: Advanced Database Architecture Diagram

4. Relational Database Design (SQL Schema)

The relational schema was designed following Third Normal Form (3NF) principles to reduce redundancy and ensure data integrity.

Three core tables were implemented:

- **cities** (city_id, city)
- **meetings** (pk_id, city_id, meeting_id)
- **meeting_metrics** (metric_id, pk_id, video_duration_sec, item_count, segment_count)

Primary and foreign key constraints enforce referential integrity.

The schema supports:

- One-to-many relationship: city → meetings
- One-to-one relationship: meeting → meeting_metrics

Normalization reduces duplication and improves maintainability while supporting efficient join operations.

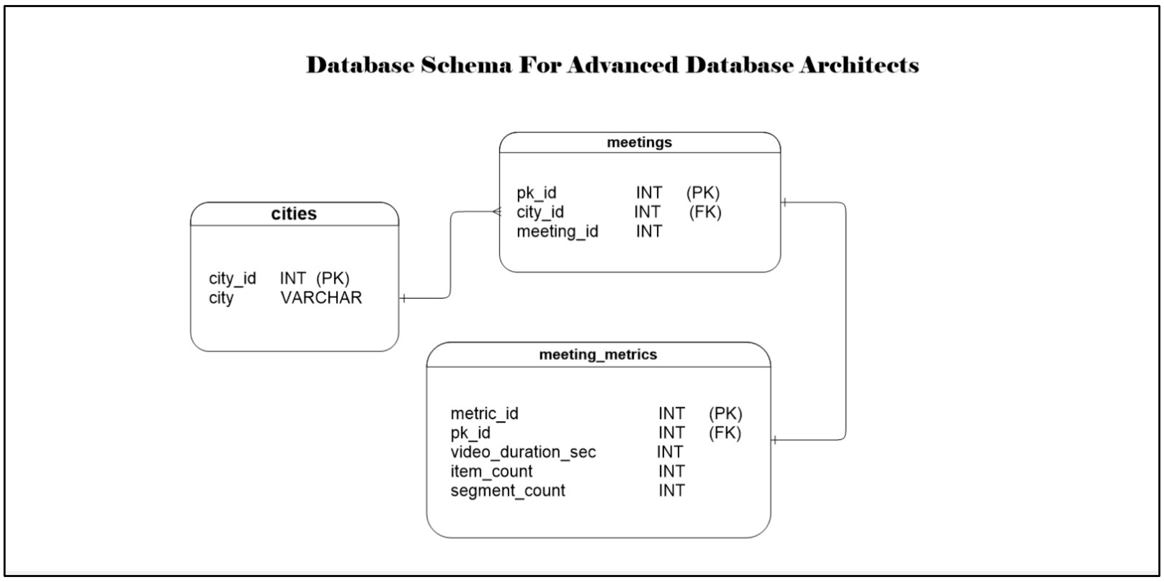


Figure 2: Database Schema for Advanced Database Architects

5. Data Cleaning and Preprocessing

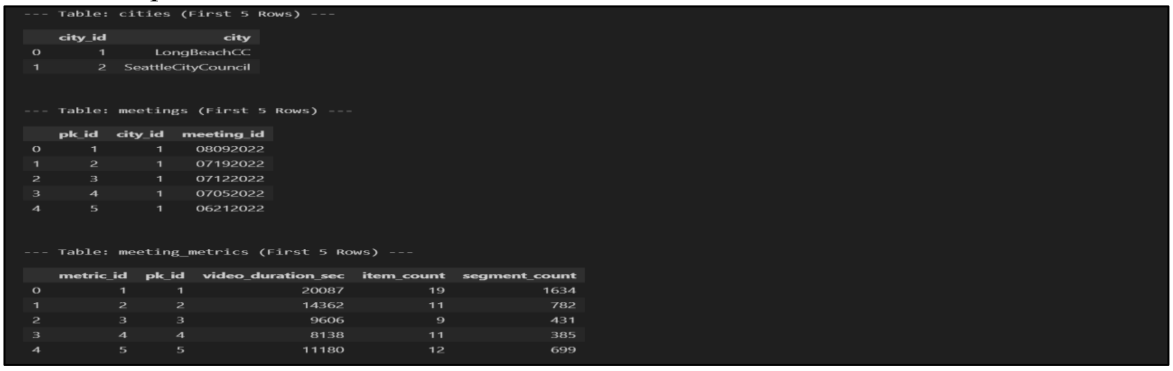
A Python-based preprocessing pipeline was developed to parse transcripts and metadata.

The following transformations were performed:

- Computed transcript_word_count from full-length transcripts
- Identified number of distinct speakers as speaker_count
- Extracted city information from the meeting_id field
- Generated a new sequential primary key index starting from 1

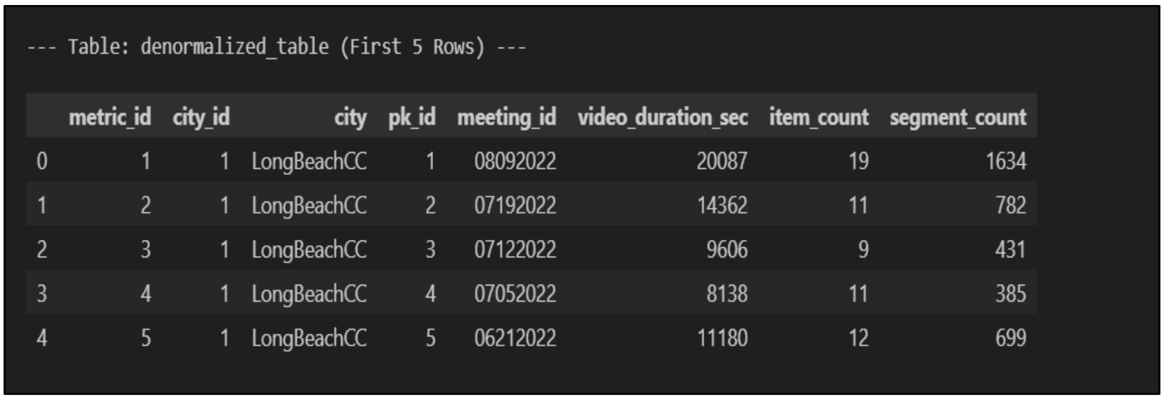
Structured numerical metadata was saved in Parquet format to improve storage efficiency and loading speed into MySQL. Transcript-related data was stored separately for MongoDB ingestion.

Parquet was selected due to its columnar format, efficient compression, and optimized read/write performance.



```
--- Table: cities (First 5 Rows) ---

   city_id                city
0        1         LongBeachCC
1        2  SeattleCityCouncil


--- Table: meetings (First 5 Rows) ---

   pk_id  city_id  meeting_id
0      1        1    08092022
1      2        1    07192022
2      3        1    07122022
3      4        1    07052022
4      5        1    06212022


--- Table: meeting_metrics (First 5 Rows) ---

   metric_id  pk_id  video_duration_sec  item_count  segment_count
0          1      1               20087          19           1634
1          2      2               14362          11            782
2          3      3                9606           9            431
3          4      4                8138          11            385
4          5      5               11180          12            699
```

Figure 3: Creating Sample Tables



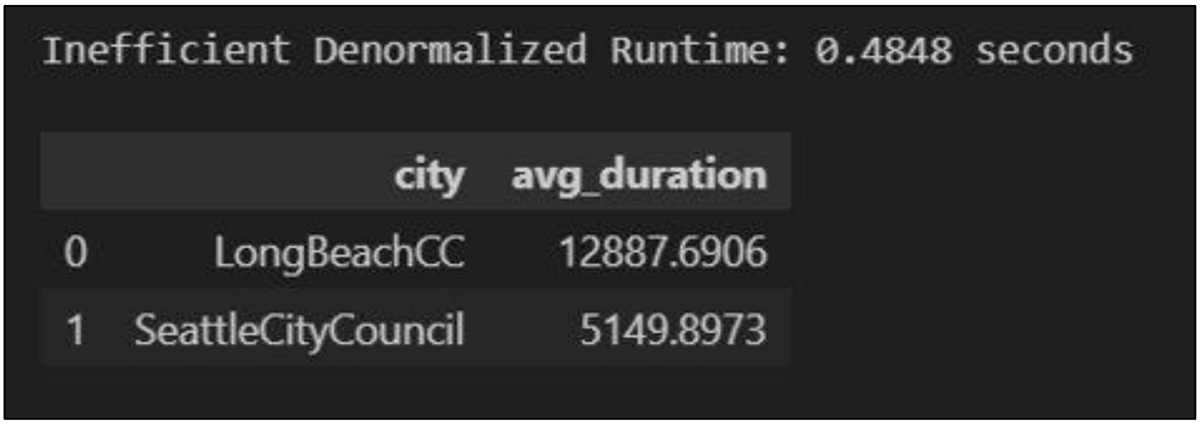
```
--- Table: denormalized_table (First 5 Rows) ---

   metric_id  city_id         city  pk_id  meeting_id  video_duration_sec  item_count  segment_count
0          1        1  LongBeachCC      1    08092022               20087          19           1634
1          2        1  LongBeachCC      2    07192022               14362          11            782
2          3        1  LongBeachCC      3    07122022                9606           9            431
3          4        1  LongBeachCC      4    07052022                8138          11            385
4          5        1  LongBeachCC      5    06212022               11180          12            699
```

Figure 4: Denormalize the table

6. Inefficient Query Vs Optimized Query

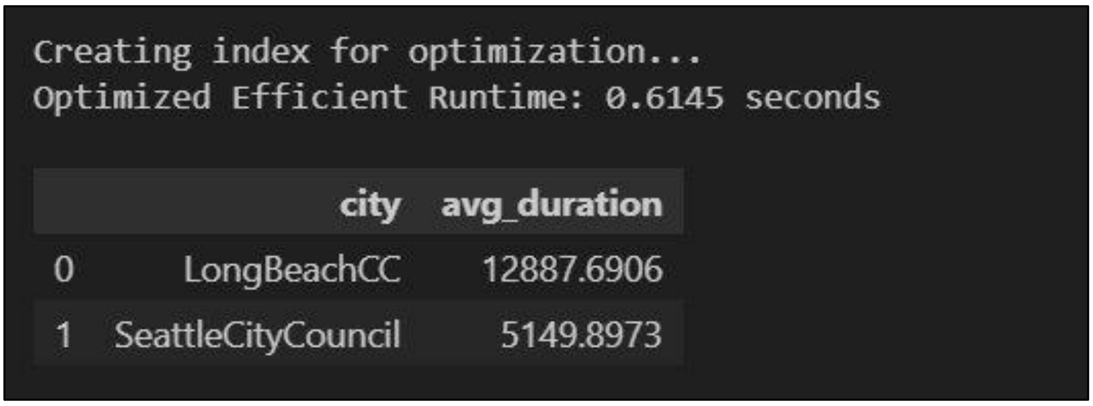
The inefficient query uses one large denormalized table and groups of data using a text column (city) without an index. Because there is no index, the database must scan the entire table to calculate the average. This makes the query slower and less efficient, especially when the dataset becomes large.

On the other hand, the optimized query uses a normalized database structure with separate tables. It groups data using an integer column (city_id), which is faster than grouping text. It also uses indexing, which helps the database find and process data more efficiently. This method is better for large datasets and is more scalable in real-world applications.



```
Inefficient Denormalized Runtime: 0.4848 seconds

                  city  avg_duration
0          LongBeachCC    12887.6906
1  SeattleCityCouncil     5149.8973
```

Figure 5: Inefficient Denormalize Runtime



```
Creating index for optimization...
Optimized Efficient Runtime: 0.6145 seconds

                 city   avg_duration
0         LongBeachCC     12887.6906
1  SeattleCityCouncil      5149.8973
```

Figure 6: Optimized Efficient Runtime

7. MQL/JavaScript Design for Unstructured Data

Full-length transcripts were stored in MongoDB using a document-based schema.

Each document includes:

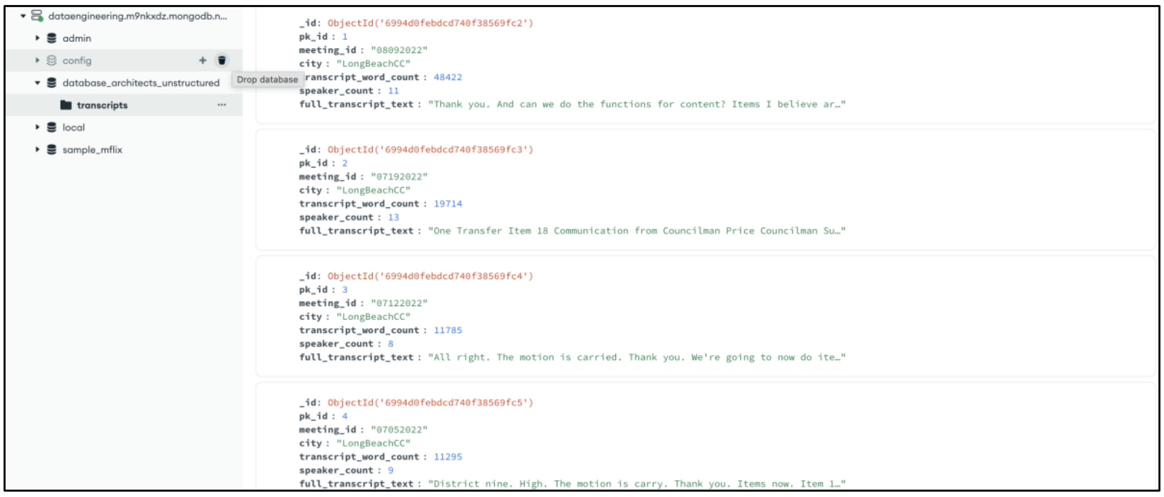
- Meeting identifiers
- Full transcript text
- transcript_word_count
- speaker_count

Although transcript_word_count and speaker_count are numeric attributes, they were retained in MongoDB to logically group transcript-related features within the same document structure.

MongoDB's aggregation framework was used to perform:

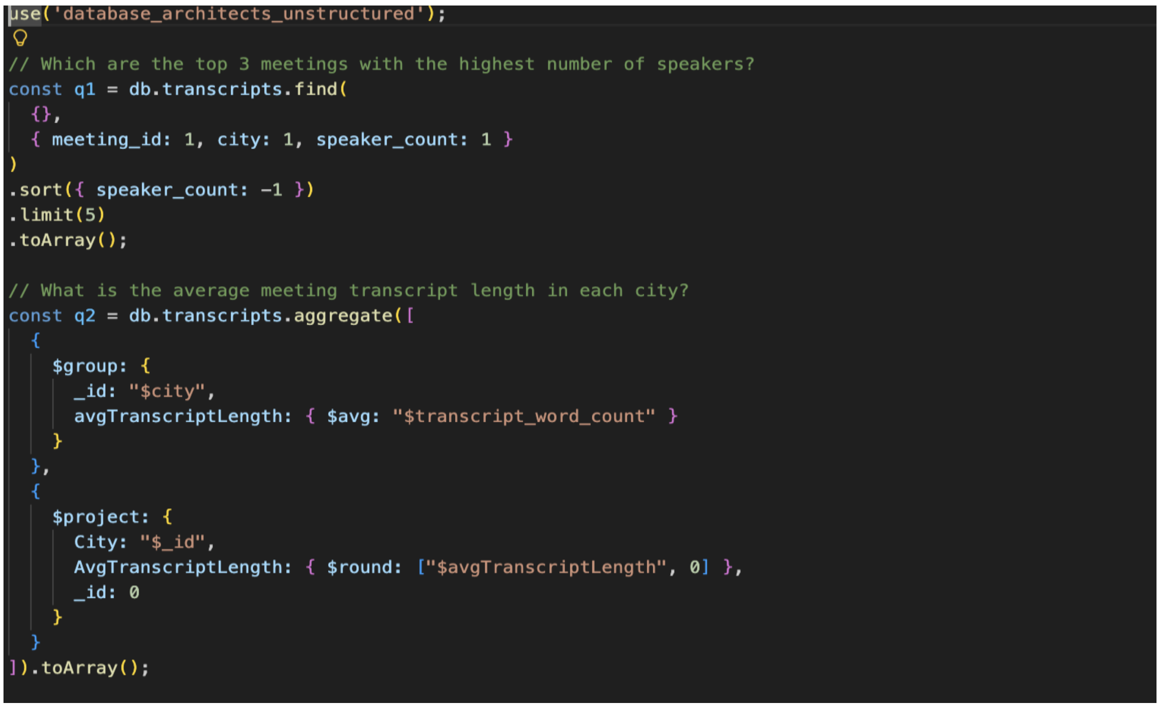
- Top meetings by speaker count
- Average speakers per city
- Longest meetings by transcript length
- Topic frequency analysis (Budget vs Housing mentions)

This approach demonstrates effective JavaScript for querying and document-based analytics.


dataengineering.m9nkxdz.mongodb.n...
admin
config
database_architects_unstructured
Drop database
transcripts
local
sample_mflix

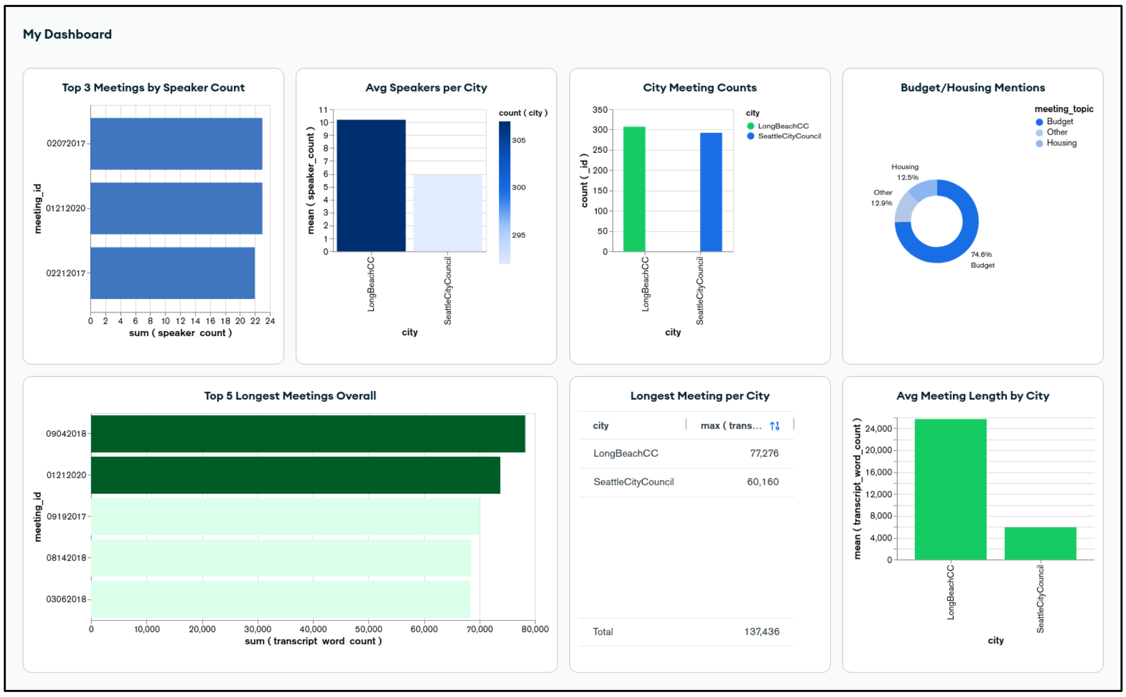
```
_id: ObjectId('6994d0febdcd740f38569fc2')
pk_id : 1
meeting_id : "08092022"
city : "LongBeachCC"
transcript_word_count : 48422
speaker_count : 11
full_transcript_text : "Thank you. And can we do the functions for content? Items I believe ar…"

_id: ObjectId('6994d0febdcd740f38569fc3')
pk_id : 2
meeting_id : "07192022"
city : "LongBeachCC"
transcript_word_count : 19714
speaker_count : 13
full_transcript_text : "One Transfer Item 18 Communication from Councilman Price Councilman Su…"

_id: ObjectId('6994d0febdcd740f38569fc4')
pk_id : 3
meeting_id : "07122022"
city : "LongBeachCC"
transcript_word_count : 11785
speaker_count : 8
full_transcript_text : "All right. The motion is carried. Thank you. We're going to now do ite…"

_id: ObjectId('6994d0febdcd740f38569fc5')
pk_id : 4
meeting_id : "07052022"
city : "LongBeachCC"
transcript_word_count : 11295
speaker_count : 9
full_transcript_text : "District nine. High. The motion is carry. Thank you. Items now. Item 1…"
```

Figure 7: Sample Data from MongoDB



```
use('database_architects_unstructured');

// Which are the top 3 meetings with the highest number of speakers?
const q1 = db.transcripts.find(
  {},
  { meeting_id: 1, city: 1, speaker_count: 1 }
)
.sort({ speaker_count: -1 })
.limit(5)
.toArray();

// What is the average meeting transcript length in each city?
const q2 = db.transcripts.aggregate([
  {
    $group: {
      _id: "$city",
      avgTranscriptLength: { $avg: "$transcript_word_count" }
    }
  },
  {
    $project: {
      City: "$_id",
      AvgTranscriptLength: { $round: ["$avgTranscriptLength", 0] },
      _id: 0
    }
  }
]).toArray();
```

Figure 8: Sample MongoDB Query



city	max (trans...
LongBeachCC	77,276
SeattleCityCouncil	60,160
Total	137,436

Figure 9: MongoDB Dashboard

8. Query Optimization and Performance Benchmarking

To demonstrate the impact of normalization and indexing, two approaches were benchmarked.

Approach 1: Denormalized Table

- Merged cities, meetings, and meeting_metrics into a single table
- Executed query without indexing
- Recorded runtime

Approach 2: Normalized Schema with Indexing

- Used 3NF schema
- Created index on frequently queried columns
- Executed identical query
- Recorded runtime

Results showed that for small datasets, full table scans may perform slightly faster due to lower indexing overhead. However, this method does not scale efficiently.

As dataset size increases, indexed normalized schemas significantly outperform denormalized full scans. Indexing ensures stable query performance and scalability for large-scale systems.

9. Cross-Database Integration

After independent processing of SQL and MongoDB data, both outputs were merged into a unified analytical dataframe.

The integration process included:

- Establishing connections to MySQL and MongoDB
- Extracting relevant analytical attributes
- Matching records using meeting identifiers
- Merging into a consolidated dataset

SQLAlchemy was used to manage MySQL interactions programmatically. The MySQL server was hosted using **Aiven** cloud services, enabling cross-system accessibility and deployment flexibility.

10. Visualization and Analytical Insights

The unified dataset was imported into Power BI for interactive visualization.

The dashboard includes:

- City-wise meeting comparison
- Transcript length distribution
- Speaker participation analysis
- Topic frequency distribution

Comparative analysis between Seattle and Long Beach highlighted variations in meeting duration, participation levels, and topic emphasis.

The visualization layer transforms raw transcript and metadata into actionable insights.

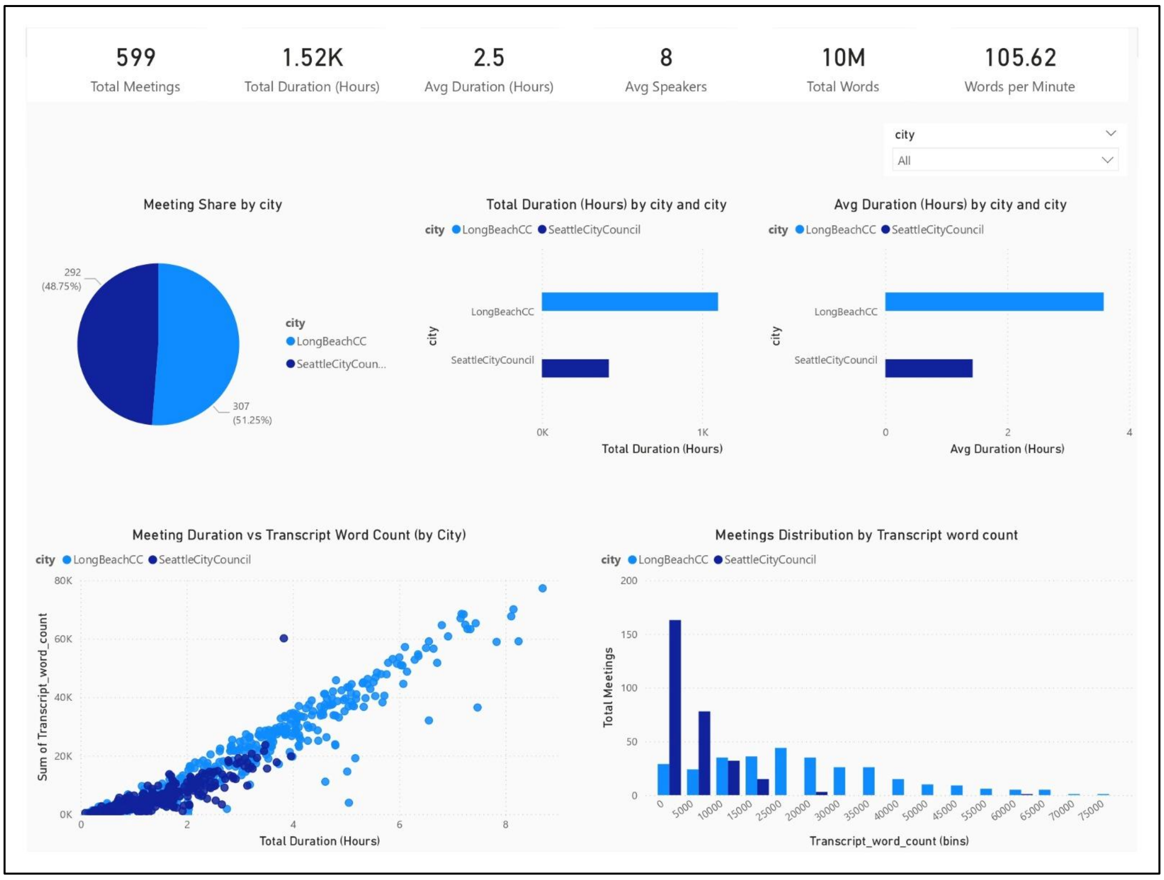


Figure 10: Power BI Visualization

11. Ethical Considerations and Conclusion

Although the MeetingBank dataset is publicly available, ethical considerations remain critical. The project avoided re-identification of individuals and focused exclusively on aggregated analytical insights.

This project successfully implemented:

- A hybrid SQL–MQL architecture
- Schema normalization and indexing strategies
- Performance benchmarking
- Cross-database integration
- Interactive visualization
- Server User ID and Password stored in a separate file for security reasons

Despite using a subset dataset, the architecture is scalable and adaptable to significantly larger data volumes.

The pipeline demonstrates a complete, modular, and reproducible data engineering workflow aligned with industry practices.

12. References

Hu, Y., Ganter, T., Deilamsalehy, H., Dernoncourt, F., Foroosh, H., & Liu, F. (2023). *MeetingBank: A Benchmark Dataset for Meeting Summarization*. Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL).

Gemini 3 Flash: Utilized as a technical collaborator for syntax optimization, debugging Python/SQL integration, and refining documentation structure.

SQLAlchemy & Pandas: Core libraries used for the ETL and benchmarking logic.

MongoDB Documentation. https://www.mongodb.com/docs/

MySQL Documentation. https://dev.mysql.com/doc/

SQLAlchemy Documentation. https://docs.sqlalchemy.org/

Microsoft Power BI Documentation. https://learn.microsoft.com/power-bi/