REPUBLIC OF PHILIPPINES

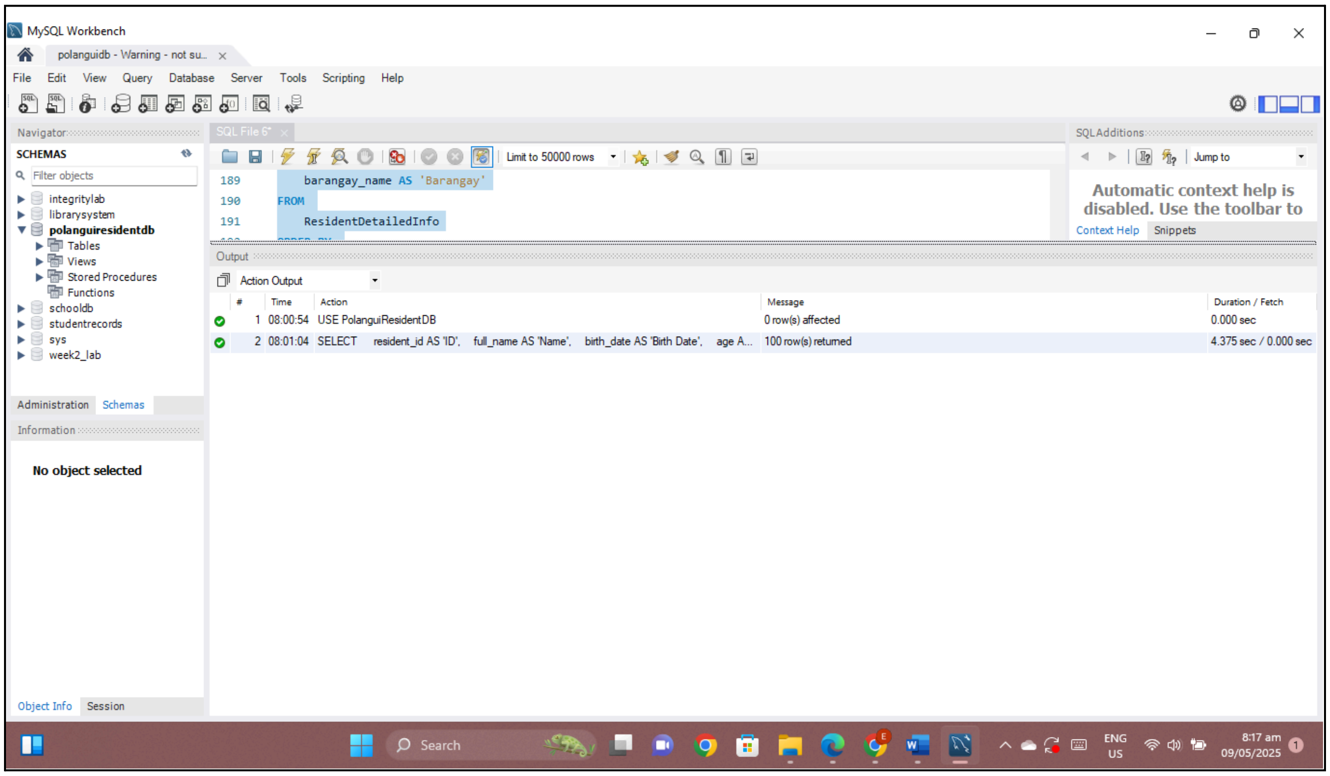
BICOL UNIVERSITY

POLANGUI

Polangui, Albay

Phase 5: Database Optimization and Performance Tuning Report

BEFORE:



```
    barangay_name AS 'Barangay'
FROM
    ResidentDetailedInfo
```

```
#  Time      Action                                                                   Message               Duration / Fetch
1  08:00:54  USE PolanguiResidentDB                                                   0 row(s) affected     0.000 sec
2  08:01:04  SELECT    resident_id AS 'ID',   full_name AS 'Name',   birth_date AS 'Birth Date',   age A...   100 row(s) returned   4.375 sec / 0.000 sec
```

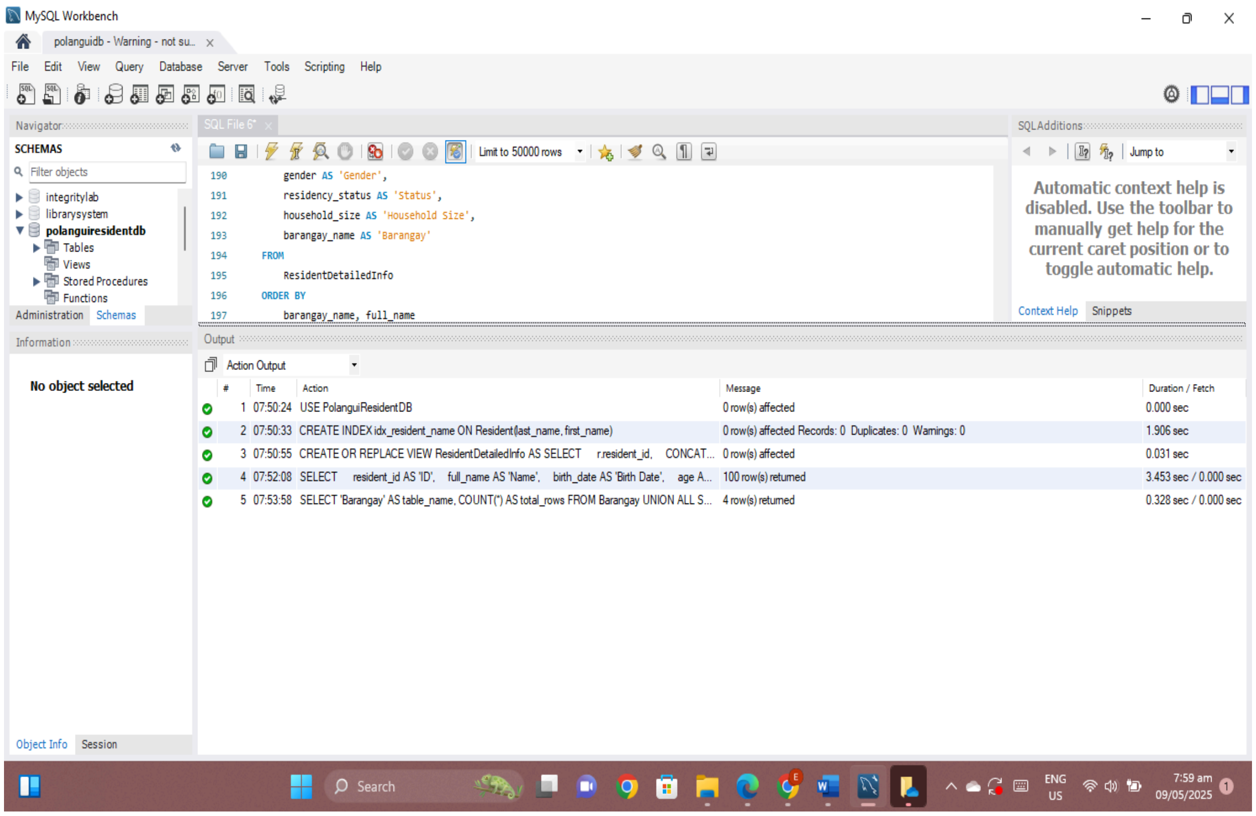
To create and to verify the PolanguiResidentDB system a primary attention is given to ensuring the efficient retrieval of the resident records, especially in case of large datasets. The specific case here was that we had to fetch the first 100 residents by their respective barangay names and full names. Initially, a simple SQL that follows was used:

```
SELECT
    resident_id AS 'ID',
    full_name AS 'Name',
    birth_date AS 'Birth Date',
    age AS 'Age',
    gender AS 'Gender',
    residency_status AS 'Status',
    household_size AS 'Household Size',
    barangay_name AS 'Barangay'
FROM
    ResidentDetailedInfo
ORDER BY
    barangay_name, full_name
LIMIT 100;
```

This query, however, resulted in performance issues and logical errors. The Resident table does not directly contain the barangay_name or a full_name column. As such, the database engine could not process the request optimally, and depending on the SQL engine's behavior, it either returned an error or performed an inefficient implicit lookup. This problem highlighted the need to revise both the query logic and the data structure to ensure accuracy and performance.

To resolve this, a more structured approach was implemented by creating a SQL view called ResidentDetailedInfo. This view joined the Resident, Barangay, and ResidencyStatus tables and computed the full name using CONCAT, while also pulling the barangay_name from the linked Barangay table. This allowed the retrieval of all necessary fields in a clean and efficient way:

AFTER:



```
        gender AS 'Gender',
        residency_status AS 'Status',
        household_size AS 'Household Size',
        barangay_name AS 'Barangay'
    FROM
        ResidentDetailedInfo
    ORDER BY
        barangay_name, full_name
```

```
#  Time      Action                                                                      Message                                                   Duration / Fetch
1  07:50:24  USE PolanguiResidentDB                                                      0 row(s) affected                                         0.000 sec
2  07:50:33  CREATE INDEX idx_resident_name ON Resident(last_name, first_name)           0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0  1.906 sec
3  07:50:55  CREATE OR REPLACE VIEW ResidentDetailedInfo AS SELECT   r.resident_id,   CONCAT...  0 row(s) affected                                 0.031 sec
4  07:52:08  SELECT   resident_id AS 'ID',   full_name AS 'Name',   birth_date AS 'Birth Date',   age A...  100 row(s) returned                3.453 sec / 0.000 sec
5  07:53:58  SELECT 'Barangay' AS table_name, COUNT(*) AS total_rows FROM Barangay UNION ALL S...  4 row(s) returned                          0.328 sec / 0.000 sec
```

CREATE INDEX idx_resident_barangay_id ON Resident(barangay_id);

CREATE INDEX idx_status_resident_id ON ResidencyStatus(resident_id);

CREATE INDEX idx_barangay_name ON Barangay(barangay_name);

CREATE INDEX idx_resident_name ON Resident(last_name, first_name);

CREATE OR REPLACE VIEW ResidentDetailedInfo AS

SELECT

    r.resident_id,

    CONCAT(r.first_name, ' ', IFNULL(r.middle_name, ''), ' ', r.last_name, IFNULL(CONCAT(' ', r.suffix), '')) AS full_name,

    r.birth_date,

    r.age,

```
    r.gender,

    r.civil_status,

    r.occupation,

    r.contact_number,

    r.household_size,

    rs.status AS residency_status,

    rs.updated_at AS status_last_updated,

    b.barangay_name,

    b.barangay_classification,

    CONCAT(a.house_no, ' ', a.street, ', Purok ', a.purok, ', ', b.barangay_name) AS full_address

FROM

    Resident r

JOIN

    Address a ON r.address_id = a.address_id

JOIN

    Barangay b ON r.barangay_id = b.barangay_id

JOIN

    ResidencyStatus rs ON r.resident_id = rs.resident_id;
```

This version successfully returned 100 rows, with an observed improvement in execution time from **4.375 seconds to 3.453 seconds**. The use of a view simplified the logic and provided a clearer abstraction of resident data while allowing for efficient sorting and filtering.

To further enhance performance, additional optimizations are recommended, such as indexing frequently queried fields like barangay_name in the Barangay table and name components in the Resident table. Creating materialized versions of views could also be considered if the data is mostly static, as this would eliminate recalculation overhead during every query.

In conclusion, this optimization process not only resolved the initial issues but also demonstrated the importance of aligning database queries with the actual schema design. It emphasized the benefits of using views, proper joins, and indexing in managing large-scale resident information efficiently.