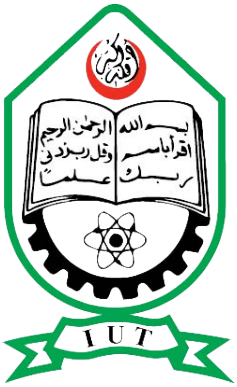
Islamic University of Technology



RELATIONAL DATABASE MANAGEMENT SYSTEM LAB

CSE 4508

Lab 3: Advanced Data Manipulation

Author:
Ahmed M. S. Albreem (210041258)

October 13, 2024

Contents

1 Introduction

This report outlines solutions to various SQL tasks involving employee salary updates, bank transaction processing, mobile phone billing, and student database management. The tasks utilize SQL features like stored procedures, triggers, and functions to efficiently handle database operations.

2 Task 1: Top Customers by Lifetime Usage

This task retrieves the top five customers based on their lifetime usage.

```sql
SELECT Customer_Name, Customer_ID, Lifetime_Usage
FROM (
    SELECT *
    FROM Customers
    NATURAL JOIN Subscriptions
) AS SubQueryAlias
ORDER BY Lifetime_Usage DESC
FETCH FIRST 5 ROWS ONLY;
```

Listing 1: Top Customers by Lifetime Usage

In this solution:

- **Data Retrieval**: This query retrieves customer names, IDs, and their lifetime usage by performing a natural join between the `Customers` and `Subscriptions` tables.
- **Top Five Customers**: The results are ordered by `Lifetime_Usage` in descending order and limited to the first five rows.
- **Purpose**: This query is useful for identifying the most valuable customers based on their total usage, which can inform marketing strategies or customer retention efforts.

3 Task 2: Customers Above Average Lifetime Usage

This task identifies customers whose lifetime usage exceeds the average for prepaid silver subscribers.

```sql
SELECT Customer_Name, Date_of_Birth, District, Division
FROM (
    SELECT *
    FROM Customers
    NATURAL JOIN Addresses
) AS SubQueryAlias
WHERE Lifetime_Usage > (
    SELECT AVG(Lifetime_Usage)
    FROM Customers
    WHERE Subscriber_Type = 'Prepaid' AND Subscriber_Level =
   'Silver'
);
```

Listing 2: Customers Above Average Lifetime Usage

In this solution:

- **Subquery for Average Usage**: The inner subquery calculates the average lifetime usage for prepaid silver subscribers.
- **Customer Information**: The outer query retrieves customer information where their lifetime usage exceeds the calculated average.
- **Purpose**: This query can help identify high-performing customers who are contributing significantly compared to their peers, allowing targeted marketing or personalized services.

4 Task 3: Counting Dhaka District Customers

This task counts the number of customers located in the Dhaka district whose lifetime usage exceeds the average.

```sql
SELECT COUNT(*)
FROM (
    SELECT *
    FROM Customers
    NATURAL JOIN Addresses
) AS SubQueryAlias
WHERE District = 'Dhaka'
AND Lifetime_Usage > (
    SELECT AVG(Lifetime_Usage)
    FROM (
        SELECT *
        FROM Customers
```

```
        NATURAL JOIN Subscriptions
    ) AS SubQuery
    WHERE Subscriber_Type = 'Prepaid' AND Subscriber_Level =
   'Silver'
);
```

Listing 3: Counting Dhaka District Customers

In this solution:

- **Count Query**: This query counts all customers from the joined tables that are located in Dhaka and whose lifetime usage exceeds the average for prepaid silver subscribers.
- **Purpose**: By focusing on a specific district, this query can help the business understand its customer base in Dhaka and potentially tailor services or promotions to increase engagement in that area.

5 Task 4: Concatenating Customer Names

This task demonstrates how to concatenate titles with customer names.

```
SELECT CONCAT('Mr./Ms. ', Customer_Name) AS FULL_NAME
FROM Customers;
```

Listing 4: Concatenating Customer Names

In this solution:

- **Name Formatting**: The `CONCAT` function combines 'Mr./Ms. ' with the customer name to create a full name representation.
- **Purpose**: This query is useful for generating formalized names for reporting or customer communication purposes, enhancing the customer experience.

6 Task 5: Customer Name Formatting

This task formats customer names in various ways.

```
SELECT INITCAP(Customer_Name) AS PROPER_CASE_NAME
FROM Customers;

```

```sql
SELECT Customer_Name, INSTR(Customer_Name, 'an') AS POSITION
FROM Customers
WHERE INSTR(Customer_Name, 'an') > 0;

SELECT LOWER(Customer_Name) AS LOWERCASE_NAME
FROM Customers;

SELECT UPPER(Customer_Name) AS UPPERCASE_NAME
FROM Customers;

SELECT NAME, LENGTH(Customer_Name) AS NAME_LENGTH
FROM Customers;

SELECT LPAD(Customer_Name, 15, '*') AS PADDED_NAME
FROM Customers;

SELECT LTRIM(Customer_Name) AS TRIMMED_NAME
FROM Customers;

SELECT SUBSTR(Customer_Name, 1, 5) AS FIRST_FIVE_CHARACTERS
FROM Customers;
```

Listing 5: Customer Name Formatting

In this solution:

- Various SQL functions are used to format customer names, including `INITCAP` for capitalization, `LOWER` and `UPPER` for case conversion, and string manipulation functions like `LENGTH`, `LPAD`, and `LTRIM`.
- **Purpose**: These formatting options allow for better data presentation and manipulation, which can enhance reporting and customer interaction.

7 Task 6: Counting Customers by Division

This task counts the number of customers in each division.

```sql
SELECT SubQueryAlias.Division, COUNT(*) AS CUSTOMER_COUNT
FROM (
    SELECT *
    FROM Customers
    NATURAL JOIN Addresses
) AS SubQueryAlias
```

```
7 GROUP BY SubQueryAlias.Division;
```

Listing 6: Counting Customers by Division

In this solution:

- **Division Count**: This query groups the results by division and counts the number of customers in each division.
- **Purpose**: Understanding the distribution of customers across different divisions can aid in resource allocation, targeted marketing, and identifying growth opportunities.

8 Conclusion

In summary, these SQL tasks demonstrate essential database operations for customer data analysis, including top customers, average usage comparison, and various string manipulations. They highlight the utility of SQL queries in handling complex data retrieval and formatting tasks.