SQL Lab Report

DengFeng Qiao Yi He MingYi Wang

October 13, 2024

Introduction

This report summarizes the SQL syntax used to solve the tasks assigned during Week 05 of the database lab session. Each task focuses on using specific SQL commands and functions to manipulate and query a bank database.

Basic Tasks

Task 1: Total Number of Transactions

To calculate the total number of transactions, we used the `COUNT()` function. This function allows us to count the number of rows in a specified column, in this case, the transaction table.

Task 2: Number of Accounts of Type 'CHK'

To count the number of accounts with a specific type, the `COUNT()` function is used again, combined with a `WHERE` clause to filter the results based on the account type ('CHK').

Task 3: Job Titles and Employee Count

In this task, the `GROUP BY` clause is used to group employees based on their job title, while the `COUNT()` function counts how many employees hold each title.

Task 4: Customer and Number of Accounts

To list the customers and the number of accounts they have, we used `JOIN` to combine the `customers` and `accounts` tables. The `COUNT()` function is applied to count the number of accounts per customer, and `GROUP BY` is used to group the results by customer.

Task 5: Total Balance for James Hadley

To find the total available balance for a specific customer, we applied the `SUM()` function, which calculates the total sum of balances. A `WHERE` clause was used to filter the results for the customer with `cust_id = 1`.

Task 6: Total Available Balance for All Customers

This task involved calculating the total available balance for each customer. The `SUM()` function is applied to the balance amounts, and a `JOIN` is used to connect customers with their accounts. `GROUP BY` was used to aggregate the results for each customer.

Task 7: Average Available Balance for Each Account Type

The `AVG()` function was used in this task to calculate the average available balance for each account type. The `GROUP BY` clause is applied to group the results by account type.

Medium Tasks

Task 8: Total Balance in Woburn Branch Accounts

This task required finding the total balance for accounts opened at a specific branch. A `JOIN` was used to connect the `accounts` and `branch` tables, while the `SUM()` function calculated the total balance. A `WHERE` clause was applied to filter by the branch name.

Task 9: Highest Available Balance by Account Type

To find the highest balance for each account type, the `MAX()` function was used. This function returns the highest value from a group of values. The results were grouped by account type using `GROUP BY`.

Task 10: Minimum Available Balance

The `MIN()` function was used to find the smallest available balance in the accounts. This function returns the lowest value from a set of data.

Task 11: Total Balance Per Customer (Rounded Down)

For this task, the `FLOOR()` function was used to round down the total balance for each customer. The `SUM()` function calculated the balance, and `GROUP BY` aggregated the results by customer.

Task 12: Employee Details in Specific Formats

This task involved formatting output with string concatenation functions such as `CONCAT()`. This allowed us to display employee names and positions in custom formats.

Advanced Tasks

Task 13: Text Replacement in a String

In this task, the `REPLACE()` function was used to substitute specific words within a string. It replaces all occurrences of a target word with a new word.

Task 14: Standardizing FED_ID Format

To standardize the FED_ID format, the `REPLACE()` function was used again. This time, it was applied to remove specific characters (like hyphens) from the data, ensuring consistency in the format.

Task 15: Transactions Count by Year

The `YEAR()` function was used to extract the year portion from the transaction date. The `COUNT()` function then tallied the number of transactions per year, and `GROUP BY` was used to group the results by year.

Task 16: Updating Job Titles to Uppercase and Counting 'Teller' Roles

The `UPPER()` function was used to convert job titles to uppercase, ensuring consistency. To count specific job titles such as 'Teller', a `CASE` statement was used to display 'Cashier' instead of 'Teller'. The results were grouped by job title.

Task 17: Customers with Less Than £5000 Balance

The `HAVING` clause was used in this task to filter customers based on their total balance. After calculating the total balance using `SUM()`, `HAVING` was applied to show only customers whose balance was less than £5000.

Task 18: Total Number of Staff Per Branch

To count the number of employees per branch, a `JOIN` was performed between the `employees` and `branch` tables. The `COUNT()` function was used to tally the number of employees, and `GROUP BY` was applied to aggregate the results by branch.

Task 19: Count of Accounts by Product Type (CHK and SAV)

In this task, the `COUNT()` function was used again to count the number of accounts for each product type ('CHK' and 'SAV'). A `CASE` statement was used to display customized labels for each product type, and `GROUP BY` was used to group the results.

Conclusion

Throughout the tasks, we utilized various SQL commands and functions such as `COUNT()`, `SUM()`, `AVG()`, `MAX()`, `MIN()`, `GROUP BY`, `JOIN`, `REPLACE()`, and `CASE` to manipulate and retrieve data from the bank database. These functions allowed us to perform complex queries, including data aggregation, filtering, and formatting, all of which are essential for database management.