Name: Prashant Suresh Shirgave
Roll No: 3 **Batch:** T1
Class: TY(CSE-AIML)

Experiment No. 6

Title: Aggregate functions and Group by, having, between, Order by clauses

Objective: To understand the use of Aggregate functions and various clauses like group by, order by, having, between clause, etc.

Theory:

Aggregate functions An aggregate function is a function that performs a calculation on a set of values, and returns a single value.

The most commonly used SQL aggregate functions are:

- MIN() - returns the smallest value within the selected column
- MAX() - returns the largest value within the selected column
- COUNT() - returns the number of rows in a set
- SUM() - returns the total sum of a numerical column
- AVG() - returns the average value of a numerical column

Syntax:

select groupfunction (column)

from tablename

[where condition(s)]

Group by clause: The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions
(COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

Syntax:

select column , groupfunction (column)

from tablename

[where condition(s)] [group by column/exp]

Having clause: To apply condition on group, having clause is used.

Syntax:

select column , groupfunction (column)

from tablename [where condition(s)]

[group by column/exp]

[having groupcondition]

Between clause: The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates. The BETWEEN operator is inclusive: begin and end values are included.

Syntax: SELECT column_name(s)

FROM table_name

WHERE column_name

BETWEEN value1 AND value2;

Order by clause: The ORDER BY keyword is used to sort the result-set in ascending or descending order.

Syntax:

select columnlist

from tablename

[where condition(s)]

[order by column/exp[asc/desc]];

Consider the following schema.

account (account-number, branch-name, balance)

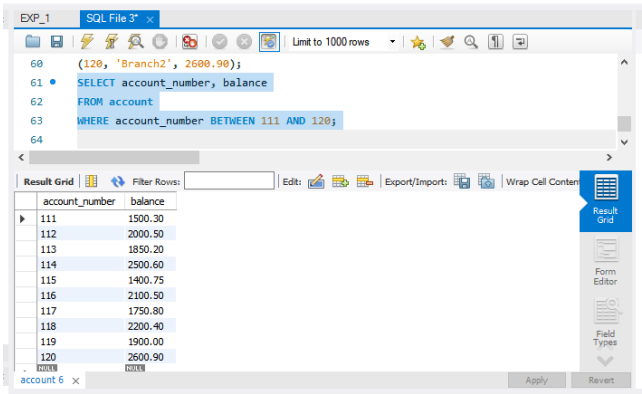
Execute the following queries:

1. Find the balance of account numbers in between '111' and '120'.

SELECT account_number, balance

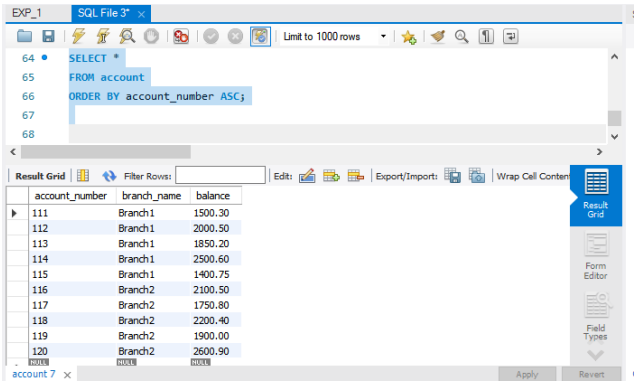
FROM account

WHERE account_number BETWEEN 111 AND 120;



2. List entire account relation in ascending order by loan number.

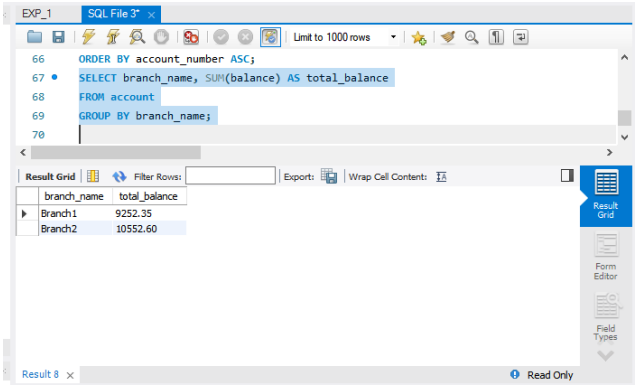
SELECT * FROM account
ORDER BY account_number ASC;



3. Find the total account balance at each branch.

SELECT branch_name, SUM(balance) AS total_balance

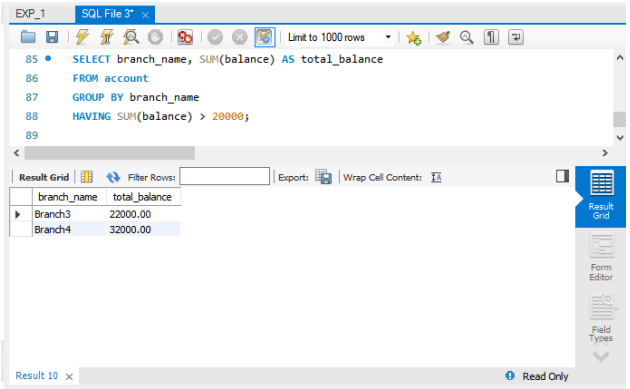
FROM account GROUP BY branch_name;



4. Find those branches having total balance greater than 20,000

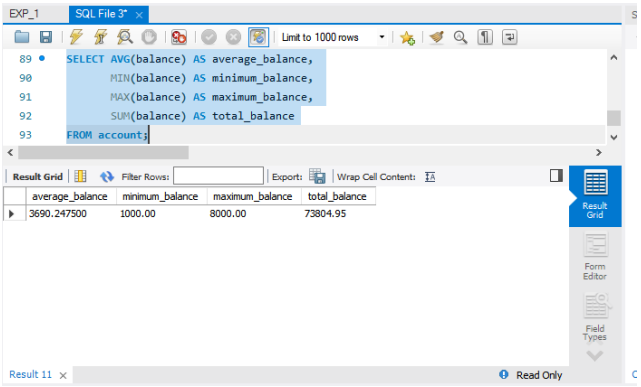
SELECT branch_name, SUM(balance) AS total_balance FROM account

GROUP BY branch_name HAVING SUM(balance) > 20000;



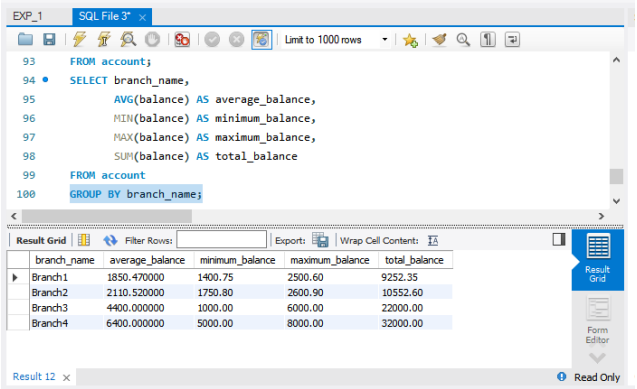
5. Find the average, minimum, maximum, total account balance.

```
SELECT AVG(balance) AS average_balance,
       MIN(balance) AS minimum_balance,
       MAX(balance) AS maximum_balance,
       SUM(balance) AS total_balance
FROM account;
```



6. Find the average, minimum, maximum, total account balance at each branch.

```
SELECT branch_name,
       AVG(balance) AS average_balance,
       MIN(balance) AS minimum_balance,
       MAX(balance) AS maximum_balance,
       SUM(balance) AS total_balance
FROM account
GROUP BY branch_name;
```

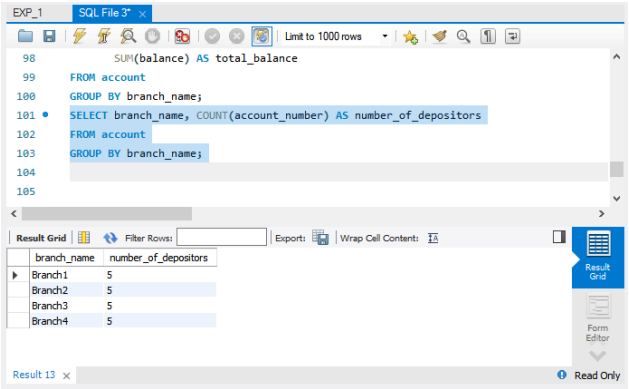


7. Find the number of depositors at each branch.

SELECT branch_name, COUNT(account_number) AS number_of_depositors

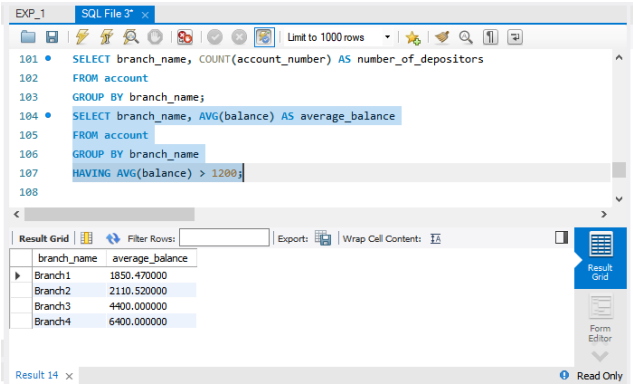
FROM account

GROUP BY branch_name;



8. Find those branches where average account balance is more than 1200.

SELECT branch_name, AVG(balance) AS average_balance
FROM account
GROUP BY branch_name
HAVING AVG(balance) > 1200;



Outcome: Students will be able to use aggregate functions and above clauses in their miniprojects & day-to-day problems solving.