program.sql

```sql
/*
Query 1: Daily Net Balance Change Per Customer
Problem:
Write a SQL query to display the daily net transaction amount
(deposits - withdrawals) for each customer.
Only include customers who had at least one transaction on that day.

Expected Output Columns:
customer_id
transaction_date (date only)
net_amount


---------------
Database Name: fs
---------------

TABLE: transactions

Field   Type    Null    Key     Default Extra
id      int     NO      PRI     NULL
customer_id     int     YES             NULL
amount  decimal(10,2)   YES             NULL
transaction_type        varchar(10)     YES             NULL
transaction_date        timestamp       YES             NULL
-------------------




Sample Output:
--------------
customer_id     transaction_date        net_amount
101     2023-10-01      5000.00
101     2023-10-02      -2000.00
101     2023-10-03      2500.00
101     2023-10-04      7000.00
101     2023-10-05      4000.00
101     2023-10-06      6000.00
101     2023-10-07      3000.00
101     2023-10-08      -1500.00
101     2023-10-09      8000.00
102     2023-10-03      1000.00



*/
use fs;
select customer_id, date(transaction_date) as transaction_date,SUM(
    CASE
        WHEN transaction_type='deposit' then amount
        when transaction_type='withdraw' then -amount
        else 0
        end
        ) as net_amount
from transactions group by customer_id, date(transaction_date) order by customer_id, date(transaction_date)

```

Proposed grade: 100 / 100

Comments

Summary of tests

1 test run/ 1 test p

program.sql

```
/*
Query 2: Customers with At Least 3 Deposits Over Rs. 5000 in a Month
Problem:
Write a SQL query to find customers who made at least 3 high-value deposits
(amount > 5000) within the same calendar month.

Your result should include:

The customer_id

The year and month (e.g., '2023-10')

The total number of such high-value deposits in that month

Sort of result:

Each row in the output should represent a customer-month combination where
the condition was met.


---------------
Database Name: fs
---------------

TABLE: transactions

Field   Type    Null    Key     Default Extra
id      int     NO      PRI     NULL
customer_id     int     YES             NULL
amount  decimal(10,2)   YES             NULL
transaction_type        varchar(10)     YES             NULL
transaction_date        timestamp       YES             NULL
-------------------



Sample Output:
--------------
customer_id month_label high_value_deposit_count
101 2023-10 3


*/
```

program.sql

```
/*
Query 3:

Write a SQL query to compute the running total of deposit amounts for each
customer in chronological order of transactions.

Expected Output Columns:

customer_id
transaction_date
amount
running_total

Hint: Use a window function like SUM(...) OVER (PARTITION BY ... ORDER BY ...).


---------------
Database Name: fs
---------------

TABLE: transactions

Field   Type    Null    Key     Default Extra
id      int     NO      PRI     NULL
customer_id     int     YES             NULL
amount  decimal(10,2)   YES             NULL
transaction_type        varchar(10)     YES             NULL
transaction_date        timestamp       YES             NULL
-------------------




Sample Output:
--------------
customer_id transaction_date    amount  running_total
101 2023-10-01 10:00:00 5000.00 5000.00
101 2023-10-03 09:00:00 2500.00 7500.00
101 2023-10-04 12:00:00 7000.00 14500.00
101 2023-10-05 14:30:00 4000.00 18500.00
101 2023-10-06 16:00:00 6000.00 24500.00
101 2023-10-07 13:15:00 3000.00 27500.00
101 2023-10-09 09:30:00 8000.00 35500.00
102 2023-10-03 15:00:00 1000.00 1000.00




*/
```

program.sql

```
/*
Query 4:

Write a SQL query to find the students who have scored more than the average
marks of each subject in that respective subject.

Your output should include:

student_id

student_name

This query should only return students who have outperformed the average mark
in every subject they appeared in.


---------------
Database Name: fs
---------------

TABLE: student_marks

Field   Type    Null    Key Default Extra
student_id  int YES     NULL
student_name    varchar(50) YES     NULL
subject varchar(30) YES     NULL
marks   int YES     NULL

--------------------




Sample Output:
--------------
student_id      student_name
2       Bob
3       Charlie
5       Eva




*/
```

program.sql

```
/*
Query 5:

Write a SQL query to compute the rank of each student per subject based on their marks.

Your output should include:

student_id
student_name
subject
marks

subject_rank (1 for highest marks in that subject, 2 for second highest, etc.)

Use a window function like RANK() or DENSE_RANK() to assign ranks



---------------
Database Name: fs
---------------

TABLE: student_marks

Field   Type    Null    Key Default Extra
student_id  int YES     NULL
student_name    varchar(50) YES     NULL
subject varchar(30) YES     NULL
marks   int YES     NULL

-------------------




Sample Output:
--------------
student_id  student_name    subject marks   subject_rank
3   Charlie Math    95  1
5   Eva Math    92  2
2   Bob Math    90  3
1   Alice   Math    80  4
4   David   Math    70  5
5   Eva Science 89  1
3   Charlie Science 88  2
2   Bob Science 85  3
1   Alice   Science 75  4
4   David   Science 68  5




*/
```

program.sql

```
/*
Query 6:

Write a SQL query to find students who have scored at least 85 marks in every
subject they have appeared for.

Your output should include:

student_id
student_name

These are consistent top performers — you should eliminate any student who has
even one subject below 85.




---------------
Database Name: fs
---------------

TABLE: student_marks

Field   Type    Null    Key Default Extra
student_id  int YES     NULL
student_name    varchar(50) YES     NULL
subject varchar(30) YES     NULL
marks   int YES     NULL

-------------------




Sample Output:
--------------
student_id  student_name
2   Bob
3   Charlie
5   Eva




*/
use fs;
select * from student_marks
-- select student_id, student_name from student_marks
-- where
```

program.sql

```
/*
Query 7:

Write a SQL query to find the top-performing student (by total marks) in each
combination of class and section.

Expected Output Columns:

class
section
name
total_marks

This should return 1 student per class-section pair who has the highest total
across all subjects.









---------------
Database Name: fs
---------------

TABLE: students

Field   Type    Null    Key     Default Extra
roll_no int     YES             NULL
name    varchar(50)     YES             NULL
class   varchar(10)     YES             NULL
section char(1) YES             NULL
subject varchar(30)     YES             NULL
marks   int     YES             NULL

--------------------



Sample Output:
--------------
class   section name    total_marks
10  A   Eva 181
10  B   Charlie 183









*/
```

program.sql

```
/*
Query 8:


Write a SQL query to find the average marks scored in each subject for every
class.

The result should include:

The class
The subject
The average_marks (rounded to 2 decimal places)

Sort the output by average_marks in ascending order so that the
lowest-performing class-subject combinations appear first.









---------------
Database Name: fs
---------------

TABLE: students

Field   Type    Null    Key     Default Extra
roll_no int     YES             NULL
name    varchar(50)     YES             NULL
class   varchar(10)     YES             NULL
section char(1) YES             NULL
subject varchar(30)     YES             NULL
marks   int     YES             NULL

-------------------



Sample Output:
--------------
class   subject average_marks
10  Science 81.00
10  Math    85.40









*/


```