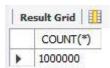
BankLoan dataset analysis using MySQL

SELECT COUNT(*) FROM bankloan;



SELECT * FROM bankloan;

1. Find top 5 loan officers by total provided loan amount.

SELECT

officer_id,

SUM(loan_amount) AS total_loan_provided

FROM

bankloan

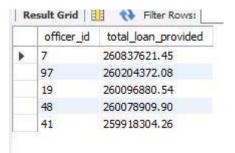
GROUP BY

officer id

ORDER BY

total_loan_provided **DESC**

LIMIT 5;



2. List the customers with more than 10 loans

SELECT

customer_id,

COUNT(customer_id) AS loan_count

FROM

bankloan

GROUP BY

```
customer_id
```

HAVING loan_count > 10;

customer_id	d loan_count	
4842	96	
9535	102	
1560	103	
6119	89	
2419	98	
851	90	
7427	99	
8411	93	

3. Find Month-Over-Month loan growth.

```
WITH MonthlyLoan AS (
      SELECT
             DATE_FORMAT(start_date, '%Y-%m') AS month,
             COUNT(customer_id) AS loan_count
       FROM
             bankloan
      GROUP BY
             month
)
SELECT
      month,
      loan_count,
      loan_count - LAG (loan_count) OVER (ORDER BY month) AS loan_growth
FROM
       MonthlyLoan
ORDER BY
      month;
```



4. Rank loan officers by region.

```
WITH RankOfficers AS (

SELECT

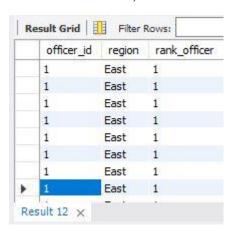
officer_id,
region,
DENSE_RANK() OVER(PARTITION BY region ORDER BY officer_id) AS rank_officer
FROM
bankloan
)

SELECT

officer_id,
region,
rank_officer
```

FROM

RankOfficers;



```
5. Find average loan amount by collateral type.
```

SELECT

```
collateral,
```

AVG(loan_amount) AS avg_loan_amnt

FROM

bankloan

GROUP BY

collateral

ORDER BY

avg_loan_amnt DESC;



6. List customers in top 10% of loan amount distribution.

```
WITH CustomerLoan AS (

SELECT

customer_id,
loan_amount

FROM

bankloan
```

AmountPercentage AS (

),

```
SELECT
```

customer_id,

loan_amount,

PERCENT_RANK() OVER(ORDER BY loan_amount DESC) AS loan_percent

FROM

CustomerLoan

```
SELECT

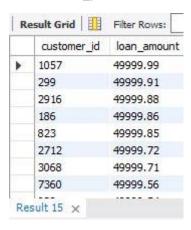
customer_id,
loan_amount

FROM
```

AmountPercentage

WHERE

loan_percent <= 0.10;</pre>



7. Calculate total loan amount by application channel per region.

SELECT

```
region,
application_channel,
SUM(loan_amount) AS total_loan_amnt
```

FROM

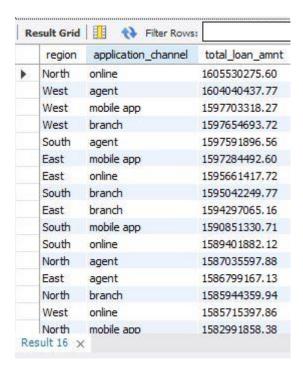
bankloan

GROUP BY

region,
application_channel

ORDER BY

total_loan_amnt DESC;



8. Identify loan officers with average interest rate greater than 5%.

SELECT

officer_id,

AVG(interest_rate) AS avg_interest_rate

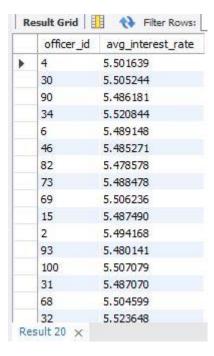
FROM

bankloan

GROUP BY

officer_id

HAVING avg_interest_rate > 5.0;



9. Cumulative loan amount per customer using windows function.

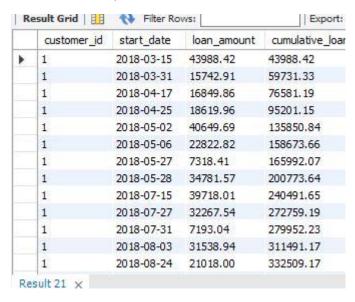
SELECT

customer_id,
start_date,
loan_amount,

SUM(loan_amount) **OVER** (PARTITION BY customer_id **ORDER** BY start_date) **AS** cumulative_loan

FROM

bankloan;



10. Regions where auto loans not exceed 50% of total loans.

```
WITH RegionLoans AS (
  SELECT
    region,
    COUNT(customer_id) AS loan_count,
    SUM(loan_amount) AS total_loan_amnt,
    SUM(CASE
               WHEN loan_type = 'auto' THEN loan_amount ELSE 0 END) AS total_auto_loan_amnt
  FROM
    bankloan
  GROUP BY
    region
)
SELECT
  region,
  loan_count,
  total_loan_amnt
FROM
  RegionLoans
WHERE
  (total_auto_loan_amnt * 1.0 / total_loan_amnt) < 0.50;
Result Grid Filter Rows:
     region loan_count total_loan_amnt
     North
            249665
                       6361502091.80
     East
            250016
                       6374042142.61
            249935
                       6372887359.16
     South
     West
            250384
                       6385113847.62
  Result 31 ×
```

11. Calculate yearly average loan amount per loan type.

SELECT

loan_type,

AVG(loan_amount) AS avg_loan_amnt,

EXTRACT(YEAR FROM start date) AS year

FROM

bankloan

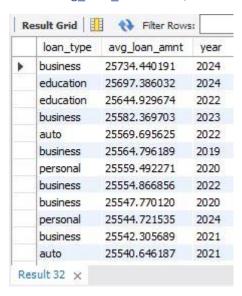
GROUP BY

loan_type,

year

ORDER BY

avg loan amnt DESC;



12. Count the loans by status and payment frequency.

SELECT

status,

payment frequency,

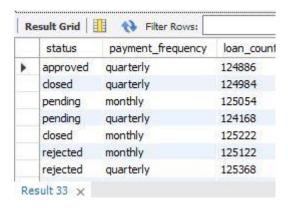
COUNT(*) AS loan count

FROM

bankloan

GROUP BY

payment_frequency;



13. List the customers with highest average interest rate.

SELECT

customer_id,

AVG(interest_rate) AS avg_interest_rate

FROM

bankloan

GROUP BY

customer_id

ORDER BY

avg interest rate DESC

LIMIT 1;



14. Differ each customers loan and their average loan amount.

WITH AvgLoanAmnt AS (

SELECT

customer id,

AVG(loan_amount) AS avg_loan_amnt

FROM

```
bankloan
```

GROUP BY

```
customer_id
```

)

SELECT

```
b.customer_id,
```

b.loan amount,

av.avg_loan_amnt,

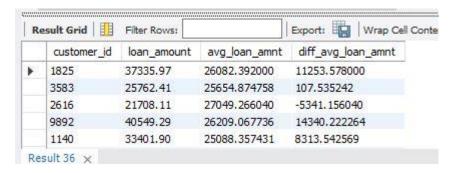
(b.loan_amount - av.avg_loan_amnt) AS diff_avg_loan_amnt

FROM

bankloan b

JOIN

AvgLoanAmnt av **ON** b.customer id = av.customer id;



15. Rank customers by total loan amount.

WITH RankCustomer AS (

SELECT

customer_id,

SUM(loan_amount) AS total_loan_amount,

RANK() OVER(ORDER BY SUM(loan_amount) DESC) AS rank_customers

FROM

bankloan

GROUP BY

customer_id

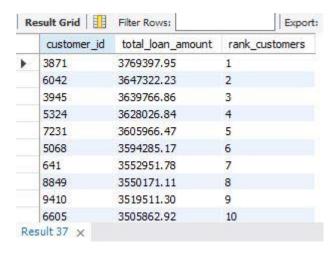
)

SELECT

customer_id,
total_loan_amount,
rank customers

FROM

RankCustomer;



16. What is the range and average interest rate for different loan types?

SELECT

loan_type,

MIN(interest_rate) AS min_int_rate,

MAX(interest_rate) AS max_int_rate,

AVG(interest_rate) AS avg_int_rate

FROM

bankloan

GROUP BY

loan_type;



17. Calculate the average interest rate for different loan term categories, where short-term is 12 months or less, medium-term is between 13 and 36 months, and long-term is greater than 36 months.

SELECT

CASE

WHEN term months <= 12 THEN 'Short-term'

WHEN term months > 12 AND term months <= 36 THEN 'Medium-term'

ELSE 'Long-term'

END AS loan_term_category,

AVG(interest_rate) AS avg_interest_rate

FROM

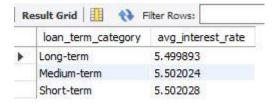
bankloan

GROUP BY

loan_term_category

ORDER BY

loan_term_category;



18. Find Loan Performance Analysis Over Time.

SELECT

YEAR(start_date) AS loan_year,

MONTH(start_date) AS loan_month,

AVG(loan_amount) AS avg_loan_amount,

status,

COUNT(*) AS loan_count

FROM

bankloan

GROUP BY

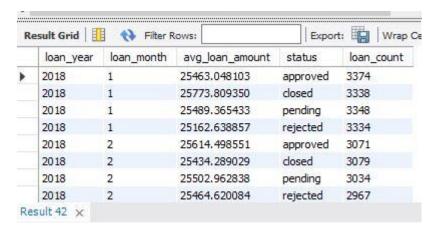
loan_year,

```
loan_month,
status
ORDER BY
```

loan_year,

loan_month,

status;



19. Track the trend of loan applications and their status (e.g., 'Approved', 'Rejected', 'Funded') over the 3 years. Calculate the number of loans and the average loan amount for each status on a monthly basis.

This provides insights into application success rates and loan volume fluctuations.

SELECT

```
YEAR(start_date) AS loan_year,

MONTH(start_date) AS loan_month,

status,

COUNT(*) AS num_of_loans,

AVG(loan_amount) AS avg_loan_amnt
```

FROM

bankloan

WHERE

start_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 3 YEAR)

GROUP BY

loan_year,

loan_month,

status

ORDER BY

loan_year,
loan_month,

status;



20. Evaluate the performance of loan officers based on the number of loans they have processed and the average loan amount they have handled.

SELECT

officer_id,

COUNT(customer_id) AS num_of_loans,

AVG(loan_amount) AS avg_loan_amnt

FROM

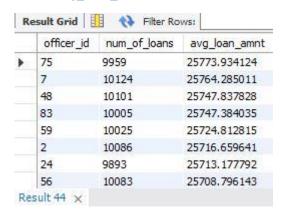
bankloan

GROUP BY

officer id

ORDER BY

avg loan amnt DESC;



```
-- creating stored procedures--
```

DELIMITER //

CREATE PROCEDURE GetLoanSummaryByRegion (IN loan region VARCHAR(100))

BEGIN

SELECT

COUNT(*) AS total_loans,

SUM(loan_amount) **AS** total_loan_amount

FROM

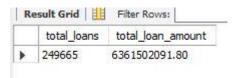
bankloan

WHERE

region = loan_region;

END //

CALL GetLoanSummaryByRegion ('North');



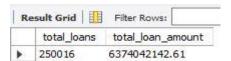
CALL GetLoanSummaryByRegion ('South');



CALL GetLoanSummaryByRegion ('West');



CALL GetLoanSummaryByRegion ('East');



```
DELIMITER //
```

CREATE PROCEDURE GetLoanSummaryBypayment frequency (IN payment freq VARCHAR(100))

BEGIN

```
payment_frequency,

COUNT(*) AS loan_count,

SUM(loan_amount) AS total_loan_amnt

FROM

bankloan

WHERE

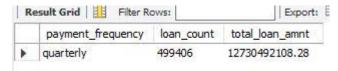
payment_frequency = payment_freq

GROUP BY

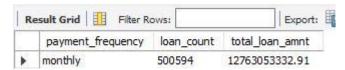
payment_frequency;
```

END //

CALL GetLoanSummaryBypayment_frequency ('quarterly');



CALL GetLoanSummaryBypayment_frequency ('monthly');



```
DELIMITER //
```

CREATE PROCEDURE GetLoanSummaryBystatus (IN Loan Status VARCHAR (100))

BEGIN

SELECT

COUNT(*) AS loan_count,

SUM(loan_amount) AS total_loan_amnt,

status

FROM

bankloan

WHERE

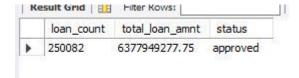
status = Loan Status

GROUP BY

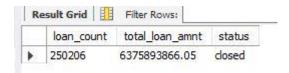
status;

END //

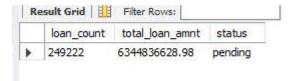
CALL GetLoanSummaryBystatus ('approved');



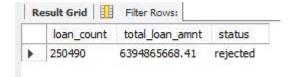
CALL GetLoanSummaryBystatus ('closed');



CALL GetLoanSummaryBystatus ('pending');



CALL GetLoanSummaryBystatus ('rejected');



DELIMITER //

CREATE PROCEDURE GetLoanSummaryByloan_type (IN loan_type VARCHAR (100))

BEGIN

SELECT

COUNT(*) AS loan_count,
SUM(loan_amount) AS total_loan_amnt,

loan_type

FROM

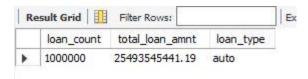
bankloan

GROUP BY

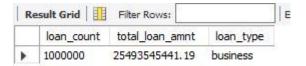
loan_type;

END //

CALL GetLoanSummaryByloan_type ('auto');



CALL GetLoanSummaryByloan type ('business');



-- creating Indexes for bankloan--

CREATE INDEX idx_application_channel **ON**

Bankloan (application channel);

SELECT

application_channel,

COUNT(*) AS loan_count,

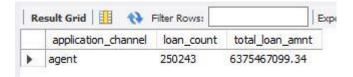
SUM(loan_amount) **AS** total_loan_amnt

FROM

bankloan

WHERE

application_channel = 'agent';



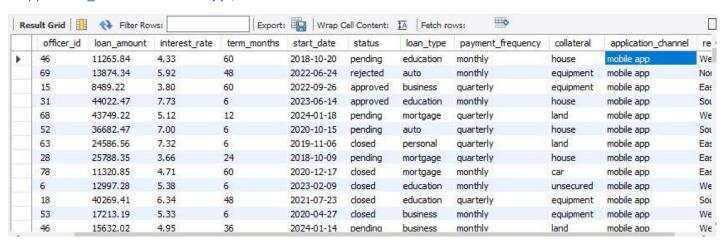
SELECT *

FROM

bankloan

WHERE

application channel = 'mobile app';



```
-- Creating Views for bankloan--
```

```
CREATE VIEW North AS
```

SELECT * FROM bankloan

WHERE region = 'North';

SELECT

```
customer_id,
```

officer_id,

SUM(loan_amount) AS total_loan_amnt,

AVG(interest_rate) AS avg_int_rate,

status

FROM

North

WHERE

status = 'approved'

GROUP BY

customer_id,

officer_id;

	customer_id	officer_id	total_loan_amnt	avg_int_rate	status
•	1825	4	37335.97	4.220000	approved
	5139	8	3456.97	5.380000	approved
	4559	97	47609.00	5.110000	approved
	9434	87	45493.28	4.880000	approved
	8851	33	40127.18	3.550000	approved
	6133	20	81363.66	6.465000	approved

```
CREATE VIEW quarterly AS
SELECT * FROM bankloan
WHERE payment frequency = 'quarterly';
SELECT
              payment_frequency,
              COUNT(*) AS loan_count,
              SUM(loan_amount) AS total_loan_amnt
FROM
              quarterly
GROUP BY
              payment frequency;
 Export:
    payment_frequency
                     loan_count
                                total_loan_amnt
                     499406
                               12730492108.28
   quarterly
CREATE VIEW unsecured AS
SELECT * FROM bankloan
WHERE collateral = 'unsecured';
SELECT
       COUNT(*) AS loan_count,
       SUM(loan amount) AS total loan amnt,
      AVG(interest_rate) AS avg_int_rate,
       collateral
FROM
       unsecured
GROUP BY
```

collateral;

