BankLoan dataset analysis using MySQL

```
SELECT COUNT(*) FROM bankloan;
```

Result Grid

	COUNT(*)
▶	1000000

```
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;
```

Result Grid | Filter Rows:

	officer_id	total_loan_provided
▶	7	260837621.45
	97	260204372.08
	19	260096880.54
	48	260078909.90
	41	259918304.26

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;
```

Result Grid | Filter Ro

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

Result 5

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;
```

Result Grid | Filter Rows:

month	loan_count	loan_growth
2023-09	12875	-271
2023-10	13383	508
2023-11	13249	-134
2023-12	13571	322
2024-01	13496	-75
2024-02	12759	-737
2024-03	13379	620
2024-04	8312	-5067

Result 7

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;
```

Result Grid | Filter Rows:

officer_id	region	rank_officer
1	East	1
1	East	1
1	East	1
1	East	1
1	East	1
1	East	1
1	East	1
1	East	1

Result 12

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;
```

Result Grid | Filter Rows:

collateral	avg_loan_amnt
unsecured	25558.017220
equipment	25517.010678
car	25487.552257
house	25457.714385
land	25447.399344

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;
```

Result Grid | Filter Rows:

customer_id	loan_amount
1057	49999.99
299	49999.91
2916	49999.88
186	49999.86
823	49999.85
2712	49999.72
3068	49999.71
7360	49999.56

Result 15

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;
```

Result Grid | Filter Rows:

region	application_channel	total_loan_amnt
North	online	1605530275.60
West	agent	1604040437.77
West	mobile app	1597703318.27
West	branch	1597654693.72
South	agent	1597591896.56
East	mobile app	1597284492.60
East	online	1595661417.72
South	branch	1595042249.77
East	branch	1594297065.16
South	mobile app	1590851330.71
South	online	1589401882.12
North	agent	1587035597.88
East	agent	1586799167.13
North	branch	1585944359.94
West	online	1585715397.86
North	mobile app	1582991858.38

Result 16

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;
```

Result Grid | Filter Rows:

officer_id	avg_interest_rate
4	5.501639
30	5.505244
90	5.486181
34	5.520844
6	5.489148
46	5.485271
82	5.478578
73	5.488478
69	5.506236
15	5.487490
2	5.494168
93	5.480141
100	5.507079
31	5.487070
68	5.504599
32	5.523648

Result 20

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;
```

Result Grid | Filter Rows: | Export:

customer_id	start_date	loan_amount	cumulative_loan
1	2018-03-15	43988.42	43988.42
1	2018-03-31	15742.91	59731.33
1	2018-04-17	16849.86	76581.19
1	2018-04-25	18619.96	95201.15
1	2018-05-02	40649.69	135850.84
1	2018-05-06	22822.82	158673.66
1	2018-05-27	7318.41	165992.07
1	2018-05-28	34781.57	200773.64
1	2018-07-15	39718.01	240491.65
1	2018-07-27	32267.54	272759.19
1	2018-07-31	7193.04	279952.23
1	2018-08-03	31538.94	311491.17
1	2018-08-24	21018.00	332509.17

Result 21

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
South	249935	6372887359.16
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;
```

Result Grid | Filter Rows:

loan_type	avg_loan_amnt	year
business	25734.440191	2024
education	25697.386032	2024
education	25644.929674	2022
business	25582.369703	2023
auto	25569.695625	2022
business	25564.796189	2019
personal	25559.492271	2020
business	25554.866856	2022
business	25547.770120	2020
personal	25544.721535	2024
business	25542.305689	2021
auto	25540.646187	2021

Result 32

12. Count the loans by status and payment frequency.

```
SELECT
    status,
    payment_frequency,
    COUNT(*) AS loan_count
FROM
    bankloan
GROUP BY
```

```
    status,
    payment_frequency;
```

Result Grid | Filter Rows:

status	payment_frequency	loan_count
approved	quarterly	124886
closed	quarterly	124984
pending	monthly	125054
pending	quarterly	124168
closed	monthly	125222
rejected	monthly	125122
rejected	quarterly	125368

Result 33

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;
```

Result Grid | Filter Rows:

customer_id	avg_interest_rate
4245	6.050521

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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conte

customer_id	loan_amount	avg_loan_amnt	diff_avg_loan_amnt
1825	37335.97	26082.392000	11253.578000
3583	25762.41	25654.874758	107.535242
2616	21708.11	27049.266040	-5341.156040
9892	40549.29	26209.067736	14340.222264
1140	33401.90	25088.357431	8313.542569

Result 36

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;
```

Result Grid | Filter Rows: | Export:

customer_id	total_loan_amount	rank_customers
3871	3769397.95	1
6042	3647322.23	2
3945	3639766.86	3
5324	3628026.84	4
7231	3605966.47	5
5068	3594285.17	6
641	3552951.78	7
8849	3550171.11	8
9410	3519511.30	9
6605	3505862.92	10

Result 37

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;
```

Result Grid | Filter Rows: | Export:

loan_type	min_int_rate	max_int_rate	avg_int_rate
education	3.00	8.00	5.508188
auto	3.00	8.00	5.499464
personal	3.00	8.00	5.501422
business	3.00	8.00	5.497018
mortgage	3.00	8.00	5.500470

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;
```

Result Grid | Filter Rows:

loan_term_category	avg_interest_rate
Long-term	5.499893
Medium-term	5.502024
Short-term	5.502028

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;
```

Result Grid | Filter Rows: | Export: | Wrap Ce

loan_year	loan_month	avg_loan_amount	status	loan_count
2018	1	25463.048103	approved	3374
2018	1	25773.809350	closed	3338
2018	1	25489.365433	pending	3348
2018	1	25162.638857	rejected	3334
2018	2	25614.498551	approved	3071
2018	2	25434.289029	closed	3079
2018	2	25502.962838	pending	3034
2018	2	25464.620084	rejected	2967

Result 42

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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell C

loan_year	loan_month	status	num_of_loans	avg_loan_amnt
2022	4	approved	2047	25417.920151
2022	4	closed	2061	25183.884212
2022	4	pending	2087	25671.377638
2022	4	rejected	2166	25252.230314
2022	5	approved	3373	25616.888787
2022	5	closed	3310	25559.416891
2022	5	pending	3310	25398.408961
2022	5	rejected	3361	25507.483713

Result 43

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;
```

Result Grid | Filter Rows:

officer_id	num_of_loans	avg_loan_amnt
75	9959	25773.934124
7	10124	25764.285011
48	10101	25747.837828
83	10005	25747.384035
59	10025	25724.812815
2	10086	25716.659641
24	9893	25713.177792
56	10083	25708.796143

Result 44

-- 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');
```

Result Grid | Filter Rows:

	total_loans	total_loan_amount
▶	249665	6361502091.80

```
CALL GetLoanSummaryByRegion ('South');
```

Result Grid | Filter Rows:

	total_loans	total_loan_amount
▶	249935	6372887359.16

```
CALL GetLoanSummaryByRegion ('West');
```

Result Grid | Filter Rows:

	total_loans	total_loan_amount
▶	250384	6385113847.62

```
CALL GetLoanSummaryByRegion ('East');
```

Result Grid | Filter Rows:

	total_loans	total_loan_amount
▶	250016	6374042142.61

```
DELIMITER //

CREATE PROCEDURE GetLoanSummaryBypayment_frequency ( IN payment_freq VARCHAR(100))
BEGIN
	SELECT
		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');
```

Result Grid | Filter Rows: | Export:

payment_frequency	loan_count	total_loan_amnt
quarterly	499406	12730492108.28

```
CALL GetLoanSummaryBypayment_frequency ('monthly');
```

Result Grid | Filter Rows: | Export:

payment_frequency	loan_count	total_loan_amnt
monthly	500594	12763053332.91

```
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');
```

Result Grid | Filter Rows:

	loan_count	total_loan_amnt	status
▶	250082	6377949277.75	approved

```
CALL GetLoanSummaryBystatus ('closed');
```

Result Grid | Filter Rows:

	loan_count	total_loan_amnt	status
▶	250206	6375893866.05	closed

```
CALL GetLoanSummaryBystatus ('pending');
```

Result Grid | Filter Rows:

	loan_count	total_loan_amnt	status
▶	249222	6344836628.98	pending

```
CALL GetLoanSummaryBystatus ('rejected');
```

Result Grid | Filter Rows:

loan_count	total_loan_amnt	status
250490	6394865668.41	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');
```

Result Grid | Filter Rows: | Ex

loan_count	total_loan_amnt	loan_type
1000000	25493545441.19	auto

```
CALL GetLoanSummaryByloan_type ('business');
```

Result Grid | Filter Rows: | E

loan_count	total_loan_amnt	loan_type
1000000	25493545441.19	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';
```

Result Grid | Filter Rows: | Exp

application_channel	loan_count	total_loan_amnt
agent	250243	6375467099.34

```
SELECT *
FROM
  bankloan
WHERE
  application_channel = 'mobile app';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

officer_id	loan_amount	interest_rate	term_months	start_date	status	loan_type	payment_frequency	collateral	application_channel	re
46	11265.84	4.33	60	2018-10-20	pending	education	monthly	house	mobile app	We
69	13874.34	5.92	48	2022-06-24	rejected	auto	monthly	equipment	mobile app	No
15	8489.22	3.80	60	2022-09-26	approved	business	quarterly	equipment	mobile app	Ea
31	44022.47	7.73	6	2023-06-14	approved	education	monthly	house	mobile app	So
68	43749.22	5.12	12	2024-01-18	pending	mortgage	quarterly	land	mobile app	We
52	36682.47	7.00	6	2020-10-15	pending	auto	quarterly	house	mobile app	So
63	24586.56	7.32	6	2019-11-06	closed	personal	quarterly	land	mobile app	Ea
28	25788.35	3.66	24	2018-10-09	pending	mortgage	quarterly	house	mobile app	Ea
78	11320.85	4.71	60	2020-12-17	closed	mortgage	monthly	car	mobile app	Ea
6	12997.28	5.38	6	2023-02-09	closed	education	monthly	unsecured	mobile app	We
18	40269.41	6.34	48	2021-07-23	closed	education	quarterly	equipment	mobile app	So
53	17213.19	5.33	6	2020-04-27	closed	business	monthly	equipment	mobile app	We
46	15632.02	4.95	36	2024-01-14	pending	business	monthly	land	mobile app	We

-- 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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell

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

Result 63

```
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;
```

Result Grid | Filter Rows: | Export:

payment_frequency	loan_count	total_loan_amnt
quarterly	499406	12730492108.28

```
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;
```

Result Grid | Filter Rows: | Export:

loan_count	total_loan_amnt	avg_int_rate	collateral
199948	5110274427.03	5.501060	unsecured