```sql
CREATE DATABASE FINANCIAL_LOAN;

USE FINANCIAL_LOAN;

SELECT * FROM BANK_DATA;

-- 1. TOTAL NUMBER OF APPLICATIONS

SELECT COUNT(ID) AS TOTAL_APPLICATIONS FROM BANK_DATA;



-- 2. TOTAL FUNDED AMOUNT

SELECT SUM(LOAN_AMOUNT)/1000000 AS TOTAL_FUNDED_AMOUNT_IN_MILIONS FROM BANK_DATA;



-- 3. TOTAL PAYMENT RECORD

SELECT SUM(TOTAL_PAYMENT)/1000000 AS TOTAL_PAYMENT_RECORD_IN_MILIONS FROM
  BANK_DATA;



-- 4. AVERAGE INTREST FOR EACH PURPOSE OF LOAN

SELECT DISTINCT PURPOSE FROM BANK_DATA;

SELECT PURPOSE, ROUND((AVG(INT_RATE)),2) * 100 AS AVG_INT_RATE_OF_EACH_PURPOSE
FROM BANK_DATA
GROUP BY PURPOSE
ORDER BY AVG_INT_RATE_OF_EACH_PURPOSE DESC;



--5. AVERAGE  DTI WHICH WILL BE GROUP BY ON MONTHLY BASIS AND I NEED THE MONTHS
  ONLY FOR YEAR 2021 WITH THAT I NEED 3 COLUMNS YEAR, MONTH, AVG DTI COLUMN

SELECT YEAR(ISSUE_DATE) AS 'YEAR',
MONTH(ISSUE_DATE) AS 'MONTH',
ROUND(AVG(DTI),2) * 100 AS 'AVG_DTI'
FROM BANK_DATA
GROUP BY YEAR(ISSUE_DATE), MONTH(ISSUE_DATE)
HAVING YEAR(ISSUE_DATE) = 2021
ORDER BY 'MONTH';
```

```sql
--6. GOOD LOAN V/S BAD LOAN
-- GOOD LOAN APPLICATION IN %

SELECT DISTINCT LOAN_STATUS FROM BANK_DATA;

SELECT COUNT(CASE WHEN LOAN_STATUS = 'FULLY PAID' OR LOAN_STATUS = 'CURRENT' THEN
  ID END) * 100 / COUNT(ID) AS 'GOOD_LOAN_APP.-IN%' FROM BANK_DATA;
SELECT COUNT(CASE WHEN LOAN_STATUS IN ('FULLY PAID','CURRENT') THEN ID END) * 100 /
   COUNT(ID) AS 'GOOD_LOAN_APP.-IN%' FROM BANK_DATA;


-- GOOD LOAN APPLICATIONS

SELECT COUNT(CASE WHEN LOAN_STATUS IN ('FULLY PAID','CURRENT') THEN ID END) AS
  GOOD_LOAN_APPLICATIONS,
COUNT(ID) AS TOTAL_LOAN_APPLIACATIONS FROM BANK_DATA;



-- 7. TOTAL AMOUNT RECORD IN GOOD LOAN APPLICATIONS

SELECT ROUND(SUM(TOTAL_PAYMENT),2) / 1000000 AS TOTAL_AMOUNT_RECORD_APPLICATION
  FROM BANK_DATA
WHERE LOAN_STATUS IN ('FULLY PAID','CURRENT');



-- 8. BAD LOAN APPLICATION %

SELECT COUNT(CASE WHEN LOAN_STATUS = 'CHARGED OFF' THEN ID END) * 100/ COUNT(ID) AS
   'BAD LOAN APPLICATION %' FROM BANK_DATA;

-- BAD LOAN APPLICATION

SELECT COUNT(CASE WHEN LOAN_STATUS = 'CHARGED OFF' THEN ID END) AS 'BAD LOAN
  APPLICATION ',
COUNT(ID) AS 'TOTAL APPLICATIONS'
FROM BANK_DATA



-- 9. COMPARE TOTAL AMOUNT RECORD TO TOTAL FUNDED AMOUNT IN BAD LOAN APPLICATIONS

SELECT ROUND(SUM(TOTAL_PAYMENT),2) / 1000000 AS TOTAL_BAD_LOAN_AMOUNT_RECORD FROM
  BANK_DATA
WHERE LOAN_STATUS = 'CHARGED OFF';

SELECT ROUND(SUM(LOAN_AMOUNT),2) / 1000000 AS TOTAL_BAD_LOAN_FUNDED_AMOUNT FROM
  BANK_DATA
```

```sql
WHERE LOAN_STATUS = 'CHARGED OFF';



-- 10. WHAT IS THE TOTAL LOAN AMOUNT ISSUED FOR EACH STATE?

SELECT
    ADDRESS_STATE,
    COUNT(ADDRESS_STATE) AS LOAN_ISSUED_OF_EACH_STATE,
    SUM(LOAN_AMOUNT) AS TOTAL_LOAN_AMOUNT_OF_EACH_STATE
FROM BANK_DATA
GROUP BY
    ADDRESS_STATE;



-- 11. HOW MANY LOANS ARE IN EACH LOAN STATUS CATEGORY (E.G., FULLY PAID, CHARGED
  OFF)?

SELECT
    LOAN_STATUS,
    COUNT(*) AS NO_OF_LOAN_STATUS
FROM BANK_DATA
GROUP BY
    LOAN_STATUS;



-- 12.WHAT IS THE AVERAGE INTEREST RATE (INT_RATE) FOR LOANS BASED ON GRADE?

SELECT
    GRADE,
    ROUND(AVG(INT_RATE),3) AS AVG_INT_RATE
FROM BANK_DATA
GROUP BY
    GRADE
ORDER BY
    GRADE;



-- 13. COUNT THE NUMBER OF LOANS FOR EACH APPLICATION_TYPE (E.G., INDIVIDUAL,
  JOINT) ?

SELECT
    APPLICATION_TYPE,
    COUNT(APPLICATION_TYPE) AS NO_OF_EACH_APP
FROM BANK_DATA
GROUP BY
```

```sql
    APPLICATION_TYPE;



-- 14. FIND THE AVERAGE ANNUAL_INCOME OF APPLICANTS FOR EACH HOME_OWNERSHIP TYPE.

SELECT
    HOME_OWNERSHIP AS HOME_OWNERSHIP_TYPE,
    ROUND(AVG(ANNUAL_INCOME),3) AS AVG_ANNUAL_INCOME
FROM BANK_DATA
GROUP BY
    HOME_OWNERSHIP;



--15. DETERMINE THE TOTAL PAYMENT MADE (TOTAL_PAYMENT) PER LOAN STATUS.

SELECT
    LOAN_STATUS,
    COUNT(LOAN_STATUS) AS NO_OF_LOAN_STATUS,
    SUM(TOTAL_PAYMENT) AS TOTAL_PAYMENT_OF_EACH_LOAN
FROM BANK_DATA
GROUP BY
    LOAN_STATUS;



-- 16. WHAT IS THE AVERAGE AND MAXIMUM INT_RATE FOR LOANS WITH DIFFERENT TERMS
--     (E.G., 36 MONTHS VS. 60 MONTHS)?

SELECT
    TERM AS TYPE_OF_TERM,
    ROUND(AVG(INT_RATE),3) AS AVG_INT_RATE,
    ROUND(MAX(INT_RATE),3) AS MAX_INT_RATE
FROM BANK_DATA
GROUP BY
    TERM;



--17. FIND THE TOP 5 STATES WITH THE HIGHEST AVERAGE LOAN_AMOUNT.

SELECT TOP 5
    ADDRESS_STATE AS TOP_5_STATES,
    ROUND(AVG(LOAN_AMOUNT),3) AS AVG_LOAN_AMNT
FROM BANK_DATA
GROUP BY
    ADDRESS_STATE
ORDER BY
```

```
    AVG_LOAN_AMNT DESC;



-- 18. CALCULATE THE AVERAGE DEBT-TO-INCOME RATIO (DTI) GROUPED BY
  VERIFICATION_STATUS.

-- THE DTI RATIO IS CALCULATED BY DIVIDING A PERSON'S TOTAL MONTHLY DEBT PAYMENTS
  BY THEIR
-- GROSS MONTHLY INCOME AND THEN MULTIPLYING BY 100 TO GET A PERCENTAGE.

SELECT
    VERIFICATION_STATUS,
    ROUND(AVG(DTI),3) AS AVG_DTI
FROM BANK_DATA
GROUP BY
    VERIFICATION_STATUS;



-- 19. CALCULATE THE MONTH-OVER-MONTH GROWTH IN TOTAL_PAYMENT FOR THE YEAR 2021.

WITH MONTHLYTOTAL AS (
SELECT
    YEAR(ISSUE_DATE) AS ISSUE_YEAR,
    MONTH(ISSUE_DATE) AS ISSUE_MONTH,
    SUM(TOTAL_PAYMENT) AS MONTHLY_TOTAL_PAYMENT_RECORD
FROM BANK_DATA
WHERE YEAR(ISSUE_DATE) = 2021
GROUP BY YEAR(ISSUE_DATE), MONTH(ISSUE_DATE)
),
MONTH_OVER_MONTH AS (
SELECT
    T1.ISSUE_YEAR,
    T1.ISSUE_MONTH,
    T1.MONTHLY_TOTAL_PAYMENT_RECORD AS CURRENT_MONTH_PAYMENT,
    T2.MONTHLY_TOTAL_PAYMENT_RECORD AS PREVIOUS_MONTH_PAYMENT,
    T1.MONTHLY_TOTAL_PAYMENT_RECORD - T2.MONTHLY_TOTAL_PAYMENT_RECORD AS
      MONTH_OVER_MONTH_AMOUNT
FROM MONTHLYTOTAL T1
LEFT JOIN
MONTHLYTOTAL T2 ON T1.ISSUE_YEAR = T2.ISSUE_YEAR AND T1.ISSUE_MONTH =
  T2.ISSUE_MONTH + 1
)
SELECT
    ISSUE_YEAR,
    ISSUE_MONTH,
    CURRENT_MONTH_PAYMENT,
    PREVIOUS_MONTH_PAYMENT,
```

```sql
    MONTH_OVER_MONTH_AMOUNT
FROM MONTH_OVER_MONTH
ORDER BY ISSUE_MONTH;




-- 20. FIND THE TOP 10 OCCUPATIONS (EMP_TITLE) WITH THE HIGHEST DEFAULT RATE.

SELECT TOP 10
    EMP_TITLE,
    COUNT(*) AS TOTAL_LOANS,
    SUM(CASE WHEN LOAN_STATUS = 'CHARGED OFF' THEN 1 ELSE 0 END) AS DEFAULT_LOANS,
    (SUM(CASE WHEN LOAN_STATUS = 'CHARGED OFF' THEN 1 ELSE 0 END) * 100 / COUNT(*)) ↵
       AS DEFAULT_RATE
FROM BANK_DATA
GROUP BY
    EMP_TITLE
ORDER BY
    DEFAULT_RATE DESC;



-- 21. IDENTIFY LOANS WITH AN UNUSUALLY HIGH INT_RATE FOR THEIR RESPECTIVE GRADE     ↵
  AND SUB_GRADE.

-- "UNUSUALLY HIGH INT_RATE" REFERS TO AN INTEREST RATE SIGNIFICANTLY HIGHER THAN    ↵
  THE TYPICAL
--  OR EXPECTED RATE FOR LOANS WITHIN THE SAME CATEGORY.

WITH AVG_INT_RATE AS (
   SELECT
       GRADE,
       SUB_GRADE,
       AVG(INT_RATE) AS AVG_INT_RATE
   FROM BANK_DATA
   GROUP BY
          GRADE,
       SUB_GRADE
),
GRADE_SUBGRADE_STATS AS (
   SELECT
       B.GRADE,
       B.SUB_GRADE,
       A.AVG_INT_RATE,
       SQRT(SUM(POWER(B.INT_RATE - A.AVG_INT_RATE,2)) / COUNT(*)) AS                 ↵
         STDDEV_INT_RATE
   FROM BANK_DATA B
   JOIN
```

```
        AVG_INT_RATE A ON B.GRADE = A.GRADE AND B.SUB_GRADE = A.SUB_GRADE
    GROUP BY
            B.GRADE,
        B.SUB_GRADE,
        A.AVG_INT_RATE
),
HIGH_INTERSET_LOANS AS (
    SELECT
        B.ID,
        B.GRADE,
        B.SUB_GRADE,
        B.INT_RATE,
        G.AVG_INT_RATE,
        G.STDDEV_INT_RATE
    FROM BANK_DATA B
    JOIN
        GRADE_SUBGRADE_STATS G ON B.GRADE = G.GRADE AND B.SUB_GRADE = G.SUB_GRADE
    WHERE
        B.INT_RATE > G.AVG_INT_RATE + G.STDDEV_INT_RATE
)
SELECT
    ID,
    GRADE,
    SUB_GRADE,
    ROUND(INT_RATE,4) AS INT_RATE,
    ROUND(AVG_INT_RATE,4) AS AVG_INT_RATE,
    ROUND(STDDEV_INT_RATE,4) AS STDDEV_INT_RATE
FROM HIGH_INTERSET_LOANS
ORDER BY
    GRADE, SUB_GRADE;




-- 22. CALCULATE THE AVERAGE INSTALLMENT AMOUNT FOR APPLICANTS IN EACH
  ADDRESS_STATE WITH AN
--      ANNUAL_INCOME ABOVE $100,000.

SELECT
    ADDRESS_STATE,
    AVG(TOTAL_PAYMENT) AS AVG_INSTALLMENT_AMNT
FROM BANK_DATA
WHERE ANNUAL_INCOME > 100000
GROUP BY
    ADDRESS_STATE
ORDER BY
    ADDRESS_STATE;
```

```sql
-- 23. FOR EACH ADDRESS_STATE, CALCULATE THE PERCENTAGE OF LOANS WITH LOAN_STATUS
  MARKED AS CHARGED OFF.

SELECT
    ADDRESS_STATE,
    COUNT(CASE WHEN LOAN_STATUS = 'CHARGED OFF' THEN ID END) * 100 / COUNT(ID) AS
      CHARGED_OFF_LOAN_PERCENT
FROM BANK_DATA
GROUP BY
    ADDRESS_STATE;



-- 24. CALCULATE THE WEIGHTED AVERAGE INT_RATE FOR EACH GRADE, USING LOAN_AMOUNT AS
   THE WEIGHT.

SELECT
    GRADE,
    ROUND(SUM(LOAN_AMOUNT * INT_RATE) / SUM(LOAN_AMOUNT),4) AS
      WEIGHTED_AVG_INT_RATE
FROM BANK_DATA
GROUP BY
    GRADE
ORDER BY
    GRADE;



-- 25. MONTH OVER MONTH CHECK AVERAGE INTEREST RATE AND ROUND OFF THE INETREST RATE
   UPTO 5 DECIMAL PLACES.


WITH MONTHLYTOTAL AS (
    SELECT
        YEAR(ISSUE_DATE) AS ISSUE_YEAR,
        MONTH(ISSUE_DATE) AS ISSUE_MONTH,
        SUM(TOTAL_PAYMENT) AS MONTHLY_TOTAL_PAYMENT_RECORD,
        ROUND(AVG(INT_RATE),5) AS AVERAGE_INTEREST_RATE
    FROM BANK_DATA
    WHERE YEAR(ISSUE_DATE) = 2021
    GROUP BY YEAR(ISSUE_DATE), MONTH(ISSUE_DATE)
),
MONTH_OVER_MONTH AS (
    SELECT
        T1.ISSUE_YEAR,
        T1.ISSUE_MONTH,
        T1.MONTHLY_TOTAL_PAYMENT_RECORD AS CURRENT_MONTH_PAYMENT,
        T2.MONTHLY_TOTAL_PAYMENT_RECORD AS PREVIOUS_MONTH_PAYMENT,
```

```
        T1.AVERAGE_INTEREST_RATE AS CURRENT_MONTH_INTEREST_RATE,
        T2.AVERAGE_INTEREST_RATE AS PREVIOUS_MONTH_INTEREST_RATE,
        T1.MONTHLY_TOTAL_PAYMENT_RECORD - T2.MONTHLY_TOTAL_PAYMENT_RECORD AS
          MONTH_OVER_MONTH_AMOUNT,
        ROUND((T1.AVERAGE_INTEREST_RATE - T2.AVERAGE_INTEREST_RATE),5) AS
          MONTH_OVER_MONTH_INTEREST_RATE_CHANGE
    FROM MONTHLYTOTAL T1
    LEFT JOIN
    MONTHLYTOTAL T2 ON T1.ISSUE_YEAR = T2.ISSUE_YEAR AND T1.ISSUE_MONTH =
      T2.ISSUE_MONTH + 1
)
SELECT
    ISSUE_YEAR,
    ISSUE_MONTH,
    MONTH_OVER_MONTH_AMOUNT,
    CURRENT_MONTH_INTEREST_RATE,
    PREVIOUS_MONTH_INTEREST_RATE,
    MONTH_OVER_MONTH_INTEREST_RATE_CHANGE
FROM MONTH_OVER_MONTH
ORDER BY ISSUE_MONTH;
```