**-- Creating Table Loan\_stats**

create table Loan\_stats

(

    id int,

    member\_id int,

    loan\_amnt int,

    funded\_amnt int,

    funded\_amnt\_inv int,

    term string,

    init\_rate string,

    installment float,

    grade string,

    subgrade string,

    emp\_title string,

    emp\_length string,

    home\_ownership string,

    annual\_inc bigint,

    verification\_status string,

    issue\_d string,

    loan\_status string,

    pymt\_plan string,

    url string,

    desc string,

    purpose string,

    title string,

    zip\_code string,

    addr\_state string,

    dti float,

    delinq\_2yrs int,

    earliest\_cr\_line string,

    inq\_last\_6mths int,

    mths\_since\_last\_delinq string,

    mths\_since\_last\_record string,

    open\_acc int,

    pub\_rec int,

    revol\_bal int,

    revol\_util string,

    total\_acc int,

    initial\_list\_status string,

    out\_prncp float,

    out\_prncp\_inv double,

    total\_pymnt float,

    total\_pymnt\_inv float,

    total\_rec\_int float,

    total\_rec\_late\_fee float,

    recoveries float,

    collection\_recovery\_fee float,

    last\_pymnt\_d string,

    last\_pymnt\_amnt float,

    next\_pymnt\_d string,

    last\_credit\_pull\_d string,

    collections\_12\_mths\_ex\_med float,

    mths\_since\_last\_major\_derog string,

    policy\_code int, application\_type string,

    annual\_inc\_joint string,

    dti\_joint string,

    verification\_status\_joint string,

    acc\_now\_delinq int,

    tot\_coll\_amt string,

    tot\_cur\_bal string,

    open\_acc\_6m string,

    open\_il\_6m string,

    open\_il\_12m string,

    open\_il\_24m string,

    mths\_since\_rcnt\_il string,

    total\_bal\_il string,

    il\_util string,

    open\_rv\_12m string,

    open\_rv\_24m string,

    max\_bal\_bc string,

    all\_util string,

    total\_credit\_rv string,

    inq\_fi string,

    total\_fi\_tl string,

    inq\_last\_12m string

)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

WITH SERDEPROPERTIES

(

    "separatorChar" = ",",

    "quoteChar"     = "\""

) tblproperties('serialization.null.format'='','skip.header.line.count'='2');

**-- Creating Reject status**

create external table reject\_stats

(

    amount\_requested string,

    Application\_date string,

    loan\_title string,

    risk\_score string,

    Debt\_To\_Income string,

    zipcode string,

    state string,

    employment\_length string,

    policy\_code string

)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

WITH SERDEPROPERTIES

(

"separatorChar" = ",",

"quoteChar" = "\""

) STORED AS TEXTFILE

tblproperties('serialization.null.format'='','skip.header.line.count'='2');

**-- Q1. Total Loan issuance by yearly & Quarterly and calculate growth rate by quarter**

**-- on quarter and year on year.**

create table quarter\_table as

select

    id as id,

    case

        when lower(month) in ('jan','feb','mar') then 1

        when lower(month) in('apr','may','jun') then 2

        when lower(month) in ('jul','aug','sep') then 3

        when lower(month) in ('oct','nov','dec') then 4

        else 0 end as year\_quarter,

    loan\_amnt as total\_amnt,

    loan\_year as loan\_year

from loan\_timeline;

select

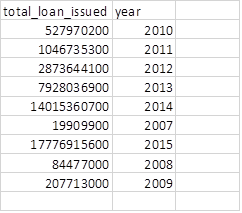
    sum(loan\_amnt) total\_loan\_issued,

    substr(issue\_d,5)  Year

from Loan\_stats

where isnotnull(year) and isnotnull(total\_loan\_issued)

group by substr(issue\_d,5) ;



**-- Total Loan Issuance by Quarter**

select

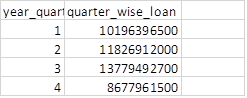
    year\_quarter,

sum(total\_amnt) quarter\_wise\_loan

from quarter\_table

where year\_quarter > 0

group by year\_quarter;



select

    Year,

    round((diff/previous)\*100) Percentage\_Growth

from (

    select

        substr(issue\_d,5)  Year,

        sum(loan\_amnt) - lag(sum(loan\_amnt)) over( order by substr(issue\_d,5)) diff,

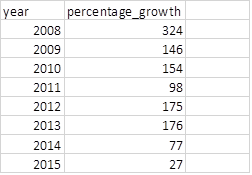
        lag(sum(loan\_amnt)) over(order by substr(issue\_d,5)) previous

    from loan\_stats

    group by substr(issue\_d,5)) t1

where isnotnull(round((diff/previous)\*100))

order by Year;



**-- calculating growth rate by quarter on quarter and year on year**

select

    loan\_year,

    year\_quarter,

    round((diff/previous)\*100) Percentage\_Growth

from

(

    select

        loan\_year ,

        year\_quarter,

        sum(total\_amnt) - lag(sum(total\_amnt)) over(order by loan\_year,year\_quarter) diff,

        lag(sum(total\_amnt)) over(order by loan\_year,year\_quarter) previous

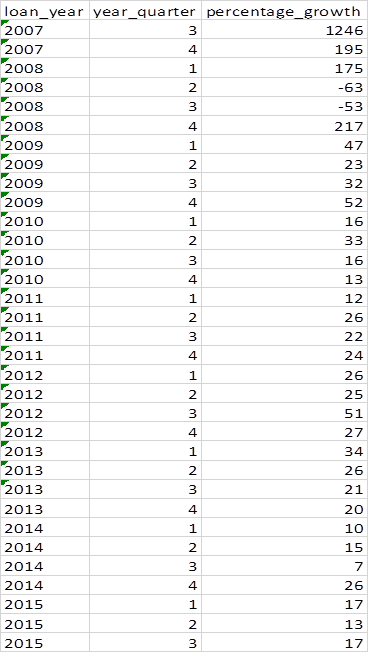
    from quarter\_table

    group by loan\_year, year\_quarter

) t2

where year\_quarter <> 0 and not isnull((diff/previous)\*100)

order by loan\_year,year\_quarter;

;

**--Q2. Percentage of loans based on the reported loan purpose. (Note: Loan purpose describes**

**--  the reported intent of borrowers from the most recent completed quarter and may not**

**--  reflect actual usage. Investors should rely on loan grades rather than loan purpose)**

select

    purpose ,

    100\*(sum(loan\_amnt)/sum(distinct total\_amt)) Percentage\_of\_Loans

from (

        select

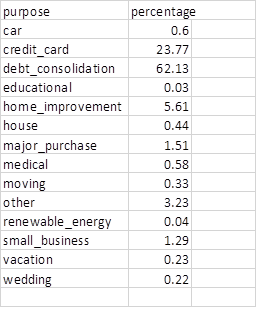
            \*,

            sum(loan\_amnt) over() as total\_amt

            from loan\_stats

    ) t1

group by purpose;



**-- Q3. Loan Issuance by state – classify the states based on loan issuance**

**--      by $50+ MM, $25-50 MM, $10-25 MM and $0-10 MM**

Select addr\_state, sum(loan\_amnt) total\_Amount

From Loan\_stats

Group by addr\_state;

Select addr\_state, case

When total\_Amount <1000000 then 'LOW'

When total\_Amount <2500000 then 'MEDIUM'

When total\_Amount <5000000 then 'HIGH'

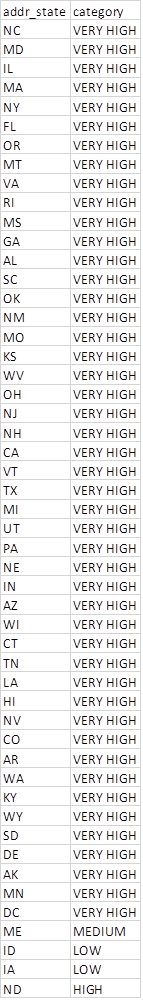
When total\_Amount >5000000 then 'VERY HIGH'

End as Category

from state\_class

WHERE isnotnull(addr\_state)

Order by Category DESC;



**-- Q4. Find the last quarter average interest rates by different term loans and overall**

SELECT

    avg(intrest\_rate) as average\_intrest\_rate\_byterm,

    t\_term as term

FROM interest\_term

GROUP BY t\_term;

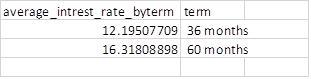
SELECT

    avg(intrest\_rate) as average\_intrest\_rate\_byterm,

    t\_term as term

FROM interest\_term

GROUP BY t\_term;



SELECT

    avg(intrest\_rate) as average\_intrest\_rate\_overall

FROM interest\_term;



**-- Q5. Find the historical returns by loan grade (Historical performance by grade**

**--      for all issued loans) and overall**

select

    grade,

    sum(loan\_amnt) as total\_loan,

    sum (total\_pymnt) as total\_pymnt

from loan\_stats

group by grade;



select

    sum(total\_pymnt - loan\_amnt) as return

from loan\_stats;



**-- Q6. Find the historical average interest rates by loan terms and loan grades (also for overall)**

select

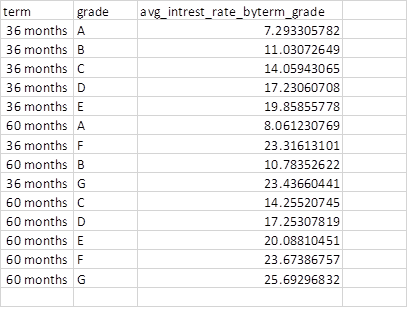
    term,

    grade,

    avg(regexp\_replace(int\_rate,'[^0-9.]','')) avg\_intrest\_rate\_byterm\_grade

from loan\_stats

group by term, grade;



SELECT

     avg(regexp\_replace(init\_rate,'[^0-9.]','')) Avg\_interest\_rate

from loan\_stats;



**-- Q7. What is percentage of loans by different loan grades by each year**

**--     and loan term level (also for overall**

select

    grade,

    term,

    substr(issue\_d,5) as year ,

    100\*(count(\*)/sum(distinct ctr))

from (

    select

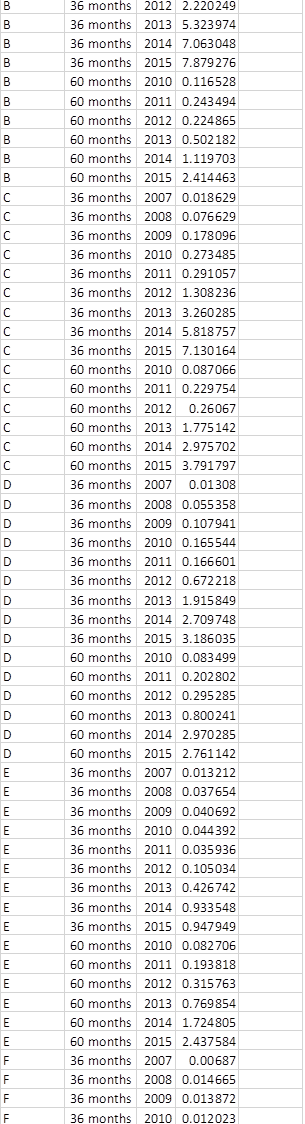
        \*,

        count(\*) over() as ctr

    from loan\_stats

    ) t1

group by grade, term , substr(issue\_d,5);



select

    substr(issue\_d,5) as year ,

    100\*(count(\*)/sum(distinct ctr))

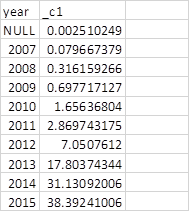
from (

        select

            \*,

                count(\*) over() as ctr from loan\_stats) t1

group by  substr(issue\_d,5);



**-- Q8. What is the loan performance details by different loan grades and overall**

select

    loan\_status,

    grade,

    100\*(count(\*)/sum(distinct ctr))

from (

        select

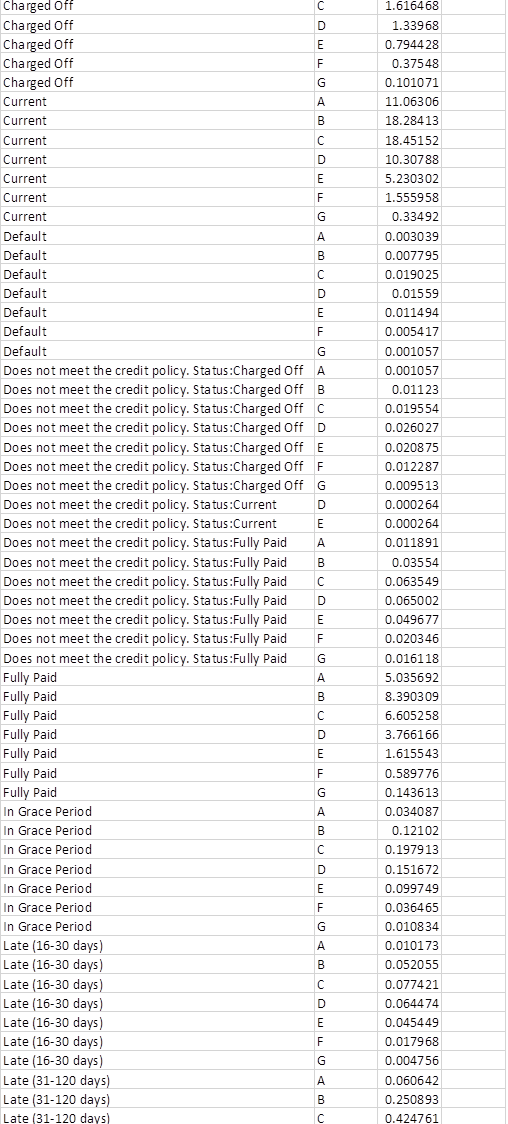
            \*,

            count(\*) over() as ctr

        from loan\_stats

    ) t2

group by loan\_status,grade;



**-- Q9. Find Net Annualized returns by vintage by different loan grades and different loan terms (also for overall)**

select

    grade,

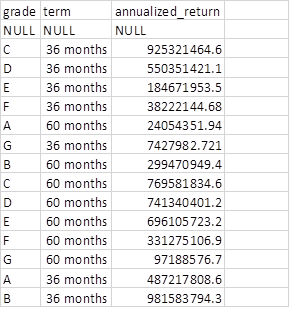
    term,

    sum((cast(regexp\_replace(int\_rate,'[^0-9.]','') as float)\*loan\_amnt)/100) annualized\_return

from loan\_stats

where isnotnull(grade)

group by grade ,term ;



**--Q10 What is loan status migration over 9 months (Net Charge offs: 120+days delinquency)**

create table loan\_migrate\_status as

select

    loan\_status as status,

    count(loan\_status) as cnt

from loan\_stats

group by loan\_status

having loan\_status in ('In Grace Period', 'Late (16-30 days)', 'Late (31-120 days)', 'Default');

Create table charged\_off as

Select

    count(loan\_status) as loan\_count

from loan\_stats

GROUP BY loan\_status

Having loan\_status='Charged Off';

Create table migrate\_status as

select

    loan\_migrate\_status.status as status,

    loan\_migrate\_status.cnt as count,

    charged\_off.loan\_count as chargeed\_count

from loan\_migrate\_status

cross join charged\_off;

Select

    status,

    100\*(count/chargeed\_count) as Migrtn\_Status

from migrate\_status;

****