LOAN MANAGEMENT SYSTEM DATABASE

SQL Implementation and Analysis

- Prathik Lathish babu

MySQL

- **MySQL** is an open-source relational database management system (RDBMS) developed by **MySQL AB**, now owned by **Oracle Corporation**.
- It is one of the most popular databases used globally, especially for web applications.
- Known for its **speed, reliability**, and **ease of use**, MySQL supports large-scale databases and is highly customizable.

Key Features:

- ✓ **Open-source**: Free to use and distribute, with a vast developer community.
- ✓ **Cross-platform**: Available on various operating systems (Windows, macOS, Linux).

Basic Concepts

- **Database**: Organized collection of data.
- **Table**: Collection of rows and columns (records and fields).
- **Schema**: The structure that defines the database (tables, views, etc.).

SQL Commands Overview

Data Definition Language (DDL): Commands that define the structure of a database.

- CREATE: Create new databases, tables.
- ALTER: Modify structure of existing tables.
- DROP: Delete tables or databases.

Data Manipulation Language (DML): Used to manipulate the data inside tables.

- INSERT: Add new records to a table.
- UPDATE: Modify existing data.
- DELETE: Remove records.
- SELECT: Retrieve data from a table.

Basic Query Example

SELECT Statement:

```
SELECT column1, column2 FROM table_name WHERE condition;
```

Example: Fetch Names from Customer Table.

```
SELECT name FROM customers WHERE city = 'New York';
```

Filtering with WHERE Clause:

Operators: =, >, <, >=, <=, <>
Combine filters with AND, OR, NOT.

Aggregate Functions:

Used to summarize data.
COUNT(), SUM(), AVG(), MAX(), MIN()

Constraints

- **PRIMARY KEY:** Unique identifier for each record.
- **FOREIGN KEY:** Links two tables.
- **NOT NULL:** Ensures a column cannot have a NULL value.
- **UNIQUE:** Ensures all values in a column are different.

Sorting Results: ORDER BY

Syntax:

```
SELECT column1, column2 FROM table_name ORDER BY column1 ASC|DESC;
```

Default Sorting is Ascending (ASC).

Loan Management System Project

Project Summary:

The Loan Management System project is designed to manage and automate various aspects of loan processing and customer data management. By working with key datasets and applying criteria-based classifications, triggers, and stored procedures, this system efficiently handles loan applications, customer statuses, and interest calculations.

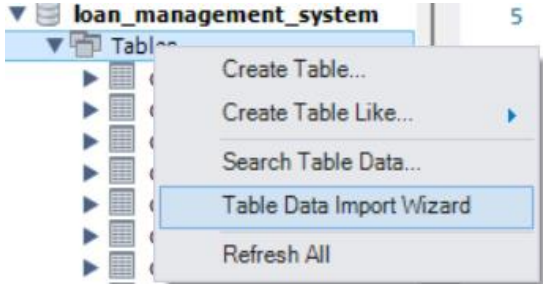
Objective:

The project aims to analyze customer income status to categorize customers based on their financial standing.

- Calculate loan amounts, monthly and annual interest, and update customer CIBIL scores.
- Automate the loan approval process by using triggers for real-time status updates and by filtering customers based on defined criteria.

Customer Income Status Analysis

- Import the Customer Income table containing applicant income data to the database.



Customers are categorized into different grades based on the following criteria:

- Grade A: Applicant income > 15,000
- Grade B: Applicant income > 9,000
- Middle Class: Applicant income > 5,000
- Low Class: Applicant income ≤ 5,000

After categorization, create a new table called Customer Income Status which stores the categorized income data. This table is then used for further calculations and analysis in the project.

Customer Income Status Analysis

```
create table customer_income_status
select *, case when applicantincome > 15000 then "Grade A Customer"
when applicantincome > 9000 then "Grade B Customer"
when applicantincome > 5000 then "Middle Class Customer"
else "Low Class Customer"end as Grade from customer_income;
```

Loan_ID	Customer ID	ApplicantIncome	CoapplicantIncome	Property_Area	Loan_Status	Grade
LP001002	IP43001	5849	0	Urban	Y	Middle Class Customer
LP001003	IP43002	4583	1508	Rural	N	Low Class Customer
LP001005	IP43003	3000	0	Urban	Y	Low Class Customer
LP001006	IP43004	2583	2358	Urban	Y	Low Class Customer
LP001008	IP43005	6000	0	Urban	Y	Middle Class Customer
LP001011	IP43006	5417	4196	Urban	Y	Middle Class Customer
LP001013	IP43007	2333	1516	Urban	Y	Low Class Customer
LP001014	IP43008	3036	2504	Semiurban	N	Low Class Customer
LP001018	IP43009	4006	1526	Urban	Y	Low Class Customer
LP001020	IP43010	12841	10968	Semiurban	N	Grade B Customer

Monthly Interest Percentage Calculation

Based on the applicant's income and their property area (urban, rural, semi-rural, etc.), the system applies different interest rates:

- Rural → 3%
- Semi-rural → 3.5%
- Urban → 5%
- Semi-urban → 2.5%
- Otherwise → 7%

```
create table monthly_interest_percentage_table select *,
case
when applicantincome < 5000 and property_area = 'rural' then 3
when applicantincome < 5000 and property_area = 'semirural' then 3.5
when applicantincome < 5000 and property_area = 'urban' then 5
when applicantincome < 5000 and property_area = 'semiurban' then 2.5
else 7
end as Monthly_interest_percentage from customer_income_status;
```

Loan_ID	Customer_ID	ApplicantIncome	CoapplicantIncome	Property_Area	Loan_Status	Grade	Monthly_interest_percentage
LP001002	IP43001	5849	0	Urban	Y	Middle Class Customer	7.0
LP001003	IP43002	4583	1508	Rural	N	Low Class Customer	3.0
LP001005	IP43003	3000	0	Urban	Y	Low Class Customer	5.0
LP001006	IP43004	2583	2358	Urban	Y	Low Class Customer	5.0
LP001008	IP43005	6000	0	Urban	Y	Middle Class Customer	7.0
LP001011	IP43006	5417	4196	Urban	Y	Middle Class Customer	7.0
LP001013	IP43007	2333	1516	Urban	Y	Low Class Customer	5.0
LP001014	IP43008	3036	2504	Semiurban	N	Low Class Customer	2.5
LP001018	IP43009	4006	1526	Urban	Y	Low Class Customer	5.0
LP001020	IP43010	12841	10968	Semiurban	N	Grade B Customer	7.0
LP001024	IP43011	3200	700	Urban	Y	Low Class Customer	5.0

Trigger

What is a Trigger?

A trigger is a special type of stored procedure in a database that automatically executes in response to specific events on a table, such as inserts, updates, or deletes.

Types of Triggers:

Row-Level Triggers: Executes once for each row affected by the event (e.g., updating each row's loan status).
Statement-Level Triggers: Executes once per statement, regardless of the number of rows affected (e.g., updating the CIBIL score status).

Why Use Triggers?

Automation: Automatically perform actions, such as validating or modifying data when certain conditions are met.
Data Integrity: Enforce business rules and ensure data consistency within the database.

Loan Status Updates and CIBIL Score Management

The system manages loan processing and CIBIL score status through a set of triggers:
A row-level trigger ensures that if the loan amount is not yet assigned (null), the status is automatically set to "Loan Still Processing."

Note: The table should be created manually for triggers.

```
delimiter //
create trigger loan_amt_null
before insert on loan_status for each row
begin
if new.loan_amount is null
then set new.loan_amount = 'Loan still processing';
end if;
end //

delimiter ;
```

A statement-level trigger updates customers' CIBIL scores and categorizes them:

- **High CIBIL score: Above 900.**
- **No penalty: CIBIL score above 750.**
- **Penalty customers: CIBIL score above 0 but below 750.**
- **Reject customers: CIBIL score 0 or less; these customers are not eligible to apply for a loan.**

```
delimiter &&
create trigger update_cibil_score_status
after insert on loan_status for each row
begin
if new.cibil_score > 900 then
insert into cibil_score_status_details (loan_id,loan_amount,cibil_score,cibil_score_status) values
(new.loan_id,new.loan_amount,new.cibil_score,'High Cibil Score');
elseif new.cibil_score > 750 then
insert into cibil_score_status_details (loan_id,loan_amount,cibil_score,cibil_score_status) values
(new.loan_id,new.loan_amount,new.cibil_score,'No Penalty');
elseif new.cibil_score > 0 then
insert into cibil_score_status_details (loan_id,loan_amount,cibil_score,cibil_score_status) values
(new.loan_id,new.loan_amount,new.cibil_score,'Penalty Customers');
else insert into cibil_score_status_details (loan_id,loan_amount,cibil_score,cibil_score_status) values
(new.loan_id,new.loan_amount,new.cibil_score,'Reject Customers');
end if;
end &&
```

After classification, customers with rejected loan applications or those still in the "processing" stage are deleted to ensure data cleanliness.

The final CIBIL score updates, including the loan amount (converted to integers), are stored in a new table named Loan CIBIL Score Status Details.

```
delete from cibil_score_status_details where cibil_score_status = 'reject customers';
delete from cibil_score_status_details where loan_amount = 'loan still processing';
alter table cibil_score_status_details modify loan_amount int;
```

loan_id	loan_amount	cibil_score	cibil_score_status
LP001003	128	920	High Cibil Score
LP001005	66	606	Penalty Customers
LP001006	120	851	No Penalty
LP001008	141	420	Penalty Customers
LP001011	267	173	Penalty Customers
LP001013	95	650	Penalty Customers
LP001014	158	471	Penalty Customers
LP001018	168	863	No Penalty
LP001020	349	730	Penalty Customers
LP001024	70	143	Penalty Customers
LP001027	109	384	Penalty Customers

Customer Interest Analysis

Calculate Monthly and Annual Interest Amount

Use the monthly interest percentage and the loan amount to calculate:
Monthly Interest Amount = (Loan Amount * Monthly Interest Percentage) / 100
Annual Interest Amount = Monthly Interest Amount * 12

Join the Customer Interest Analysis table with the CIBIL Score Status Details table to bring together loan, interest, and CIBIL score data for further analysis.

```
create table customer_interest_analysis
select c.loan_id,m.customer_id,m.applicantincome,m.coapplicantincome,m.property_area,m.loan_status,
m.grade,m.monthly_interest_percentage,c.loan_amount,c.cibil_score,c.cibil_score_status,
c.loan_amount*(m.Monthly_interest_percentage/100) as Monthly_interest_amount,
c.loan_amount*(m.Monthly_interest_percentage/100)*12 as Annual_interest_amount
from monthly_interest_percentage_table m inner join cibil_score_status_details c
on c.loan_id = m.Loan_id;
```

loan_id	customer_id	applicantincome	coapplicantincome	property_area	loan_status	grade
LP001003	IP43002	4583	1508	Rural	N	Low Class Customer
LP001005	IP43003	3000	0	Urban	Y	Low Class Customer
LP001006	IP43004	2583	2358	Urban	Y	Low Class Customer
LP001008	IP43005	6000	0	Urban	Y	Middle Class Customer
LP001011	IP43006	5417	4196	Urban	Y	Middle Class Customer
LP001013	IP43007	2333	1516	Urban	Y	Low Class Customer
LP001014	IP43008	3036	2504	Semiurban	N	Low Class Customer
LP001018	IP43009	4006	1526	Urban	Y	Low Class Customer
LP001020	IP43010	12841	10968	Semiurban	N	Grade B Customer
LP001024	IP43011	3200	700	Urban	Y	Low Class Customer

monthly_interest_percentage	loan_amount	cibil_score	cibil_score_status	Monthly_interest_amount	Annual_interest_amount
3.0	128	920	High Cibil Score	3.84000	46.08000
5.0	66	606	Penalty Customers	3.30000	39.60000
5.0	120	851	No Penalty	6.00000	72.00000
7.0	141	420	Penalty Customers	9.87000	118.44000
7.0	267	173	Penalty Customers	18.69000	224.28000
5.0	95	650	Penalty Customers	4.75000	57.00000
2.5	158	471	Penalty Customers	3.95000	47.40000
5.0	168	863	No Penalty	8.40000	100.80000
7.0	349	730	Penalty Customers	24.43000	293.16000
5.0	70	143	Penalty Customers	3.50000	42.00000

Customer Information Update

The Customer Info dataset is imported to the system, where the project updates the gender and age of customers based on their customer_id. This ensures that the data remains up-to-date and accurate for analysis and reporting.

```
update customer_det set gender = 'female'
where customer_id in ('IP43006','IP43016','IP43508','IP43577','IP43589','IP43593');
update customer_det set gender = 'male'
where customer_id in ('IP43018','IP43038');
update customer_det set age=45 where customer_id ="IP43007";
update customer_det set age=32 where customer_id ="IP43009";
```

Customer_ID	Customer_name	Gender	Age	Married	Education	Self_Employed	Loan_Id	Region_id
IP43001	Claire Gute	Male	50	No	Graduate	No	LP001002	13.2
IP43002	Darrin Van Huff	Male	66	Yes	Graduate	No	LP001003	13.2
IP43003	Sean O'Donnell	Male	20	Yes	Graduate	Yes	LP001005	13.2
IP43004	Brosina Hoffman	Male	46	Yes	Not Graduate	No	LP001006	13.2
IP43005	Andrew Allen	Male	18	No	Graduate	No	LP001008	13.2
IP43006	Irene Maddox	female	66	Yes	Graduate	Yes	LP001011	13.2
IP43007	Harold Pawlan	Male	45	Yes	Not Graduate	No	LP001013	13.3
IP43008	Pete Kriz	Male	41	Yes	Graduate	No	LP001014	13.3
IP43009	Alejandro Grove	Male	32	Yes	Graduate	No	LP001018	13.2
IP43010	Zuschuss Donatelli	Male	21	Yes	Graduate	No	LP001020	13.2
IP43011	Ken Black	Male	48	Yes	Graduate	No	LP001024	13.3

Joins in SQL

A join is an MySQL operation that combines rows from two or more tables based on a related column between them. Joins are essential for retrieving related data across multiple tables in a relational database.

Types of Joins:

Inner Join: Returns rows with matching values in both tables.

Left Join: Returns all rows from the left table and matched rows from the right. Unmatched rows return NULL for the right table.

Right Join: Returns all rows from the right table and matched rows from the left. Unmatched rows return NULL for the left table.

Full Join: Returns all rows when there is a match in either table. Unmatched rows return NULLs.

Joining all 5 tables

The project joins all datasets (Customer_interest_analysis, cibil_score_Status, Customer_det, Country_state, and Region tables) without repeating any fields. This creates a unified view of the data and allows the system to perform additional analysis.

- Joining 3 tables using INNER JOIN and creating it as a new table – (customer_info).

```
create table customer_info
select c.customer_id,c.customer_name,c.Gender,c.Age,c.Married,c.Education,c.Self_Employed,c.Loan_Id,c.Region_id,
s.Postal_Code,s.Segment,s.State,
r.region
from customer_det c inner join country_state s on c.Customer_ID = s.Customer_id
inner join region_info r on c.Region_id = r.Region_Id;
```

- Joining 2 tables using INNER JOIN and creating it as a new table – (customer_loan_info).

```
create table customer_loan_info
select c.loan_id,c.customer_id,c.applicantincome,c.coapplicantincome,c.property_area,c.loan_status,c.grade,
c.monthly_interest_percentage,c.loan_amount,c.cibil_score,c.cibil_score_status,c.Monthly_interest_amount,
c.Annual_interest_amount,l.loan_amount_term
from customer_interest_analysis c inner join loan_status l on c.customer_id = l.customer_id;
```

- Joining customer_info and customer_loan_info using INNER JOIN and creating it as a new table – (result_table).

```
create table result_table
select i.customer_id,i.loan_id,i.customer_name,i.gender,i.age,i.married,i.education,i.self_employed,
i.region_id,i.postal_code,i.segment,i.state,i.region,
l.applicantincome,l.coapplicantincome,l.property_area,l.loan_status,l.grade,l.monthly_interest_percentage,
l.loan_amount,l.cibil_score,l.cibil_score_status,l.monthly_interest_amount,l.annual_interest_amount,l.loan_amount_term
from customer_info i inner join customer_loan_info l on i.customer_id = l.customer_id;
```

- **Finding the Mismatched details from country_state and region tables using RIGHT JOIN:**

```
create table mismatch_table1
select c.Customer_id,c.Load_Id,c.Customer_name,c.Region_id,c.Postal_Code,c.Segment,c.State,
r.region from country_state c right join region_info r on c.Region_id = r.Region_id;
```

Stored Procedure

What is a Stored Procedure?

A stored procedure is a precompiled collection of one or more SQL statements stored in the database. It can be executed as a single call, encapsulating complex logic and enhancing performance.

Key Benefits:

Reusability: Stored procedures can be reused across multiple applications and queries.
Maintainability: Changes to the logic can be made in one place, without altering application code.
Performance: Precompilation can improve execution speed, as the database server optimizes the execution plan.

Syntax:

```
CREATE PROCEDURE procedure_name
    [ (parameter1 datatype, parameter2 datatype, ...) ]
AS
BEGIN
    -- SQL statements
    [ RETURN; ]
END;
```

Final Outputs

All the analysis outputs (such as mismatch detection, high CIBIL score, etc.) are stored as procedures in the database. This allows the system to perform quick lookups and automated processing for future queries and loan approvals.

```
delimiter $$

create procedure final_outputs()
begin
select * from result_table;
select * from mismatch_table1;
select * from result_table where cibil_score_status = 'High Cibil Score';
select * from result_table where segment in ('Home Office', 'Corporate');
end $$
delimiter ;
```

Output 1:

customer_id	loan_id	customer_name	gender	age	married	education	self_employed	region_id	postal_code	segment	state	region
IP43002	LP001003	Darrin Van Huff	Male	66	Yes	Graduate	No	13.2	90036	Corporate	California	West
IP43003	LP001005	Sean O'Donnell	Male	20	Yes	Graduate	Yes	13.2	33311	Consumer	Florida	West
IP43004	LP001006	Brosina Hoffman	Male	46	Yes	Not Graduate	No	13.2	90032	Consumer	California	West
IP43005	LP001008	Andrew Allen	Male	18	No	Graduate	No	13.2	28027	Consumer	North Carolina	West
IP43006	LP001011	Irene Maddox	female	66	Yes	Graduate	Yes	13.2	98103	Consumer	Washington	West
IP43007	LP001013	Harold Pawlan	Male	45	Yes	Not Graduate	No	13.3	76106	Home Office	Texas	North
IP43008	LP001014	Pete Kriz	Male	41	Yes	Graduate	No	13.3	53711	Consumer	Wisconsin	North
IP43009	LP001018	Alejandro Grove	Male	32	Yes	Graduate	No	13.2	84084	Consumer	Utah	West
IP43010	LP001020	Zuschuss Donatelli	Male	21	Yes	Graduate	No	13.2	94109	Consumer	California	West
IP43011	LP001024	Ken Black	Male	48	Yes	Graduate	No	13.3	68025	Corporate	Nebraska	North

monthly_interest_amount	annual_interest_amount	loan_amount_term
3.84000	46.08000	360
3.30000	39.60000	360
6.00000	72.00000	360
9.87000	118.44000	360
18.69000	224.28000	360
4.75000	57.00000	360
3.95000	47.40000	360
8.40000	100.80000	360
24.43000	293.16000	360
3.50000	42.00000	360

applicantincome	coapplicantincome	property_area	loan_status	grade	monthly_interest_percentage	loan_amount	cibil_score	cibil_score_status
4583	1508	Rural	N	Low Class Customer	3.0%	128	920	High Cibil Score
3000	0	Urban	Y	Low Class Customer	5.0%	66	606	Penalty Customers
2583	2358	Urban	Y	Low Class Customer	5.0%	120	851	No Penalty
6000	0	Urban	Y	Middle Class Customer	7.0%	141	420	Penalty Customers
5417	4196	Urban	Y	Middle Class Customer	7.0%	267	173	Penalty Customers
2333	1516	Urban	Y	Low Class Customer	5.0%	95	650	Penalty Customers
3036	2504	Semiurban	N	Low Class Customer	2.5%	158	471	Penalty Customers
4006	1526	Urban	Y	Low Class Customer	5.0%	168	863	No Penalty
12841	10968	Semiurban	N	Grade B Customer	7.0%	349	730	Penalty Customers
3200	700	Urban	Y	Low Class Customer	5.0%	70	143	Penalty Customers

Output 2:

Customer_id	Load_Id	Customer_name	Region_id	Postal_Code	Segment	State	region
NULL	NULL	NULL	NULL	NULL	NULL	NULL	South
IP43614	LP002990	Eleni McCrary	13.2	90036	Corporate	California	West
IP43613	LP002984	Tamara Manning	13.2	94122	Consumer	California	West
IP43611	LP002979	Christina VanderZanden	13.2	93727	Consumer	California	West
IP43610	LP002978	Scott Cohen	13.2	94122	Corporate	California	West
IP43609	LP002974	Kelly Williams	13.2	37042	Consumer	Tennessee	West
IP43608	LP002964	Kristina Nunn	13.2	80525	Home Office	Colorado	West
IP43605	LP002959	Becky Castell	13.2	85345	Home Office	Arizona	West
IP43603	LP002953	Ricardo Sperren	13.2	98115	Corporate	Washington	West
IP43602	LP002950	Patrick Jones	13.2	37130	Corporate	Tennessee	West
IP43601	LP002949	Jim Karlsson	13.2	98115	Consumer	Washington	West

Output 3:

monthly_interest_percentage	loan_amount	cibil_score	cibil_score_status	monthly_interest_amount	annual_interest_amount	loan_amount_term
3.0%	128	920	High Cibil Score	3.84000	46.08000	360
5.0%	200	928	High Cibil Score	10.00000	120.00000	360
7.0%	315	903	High Cibil Score	22.05000	264.60000	360
2.5%	122	999	High Cibil Score	3.05000	36.60000	360
5.0%	201	972	High Cibil Score	10.05000	120.60000	360
5.0%	144	949	High Cibil Score	7.20000	86.40000	360
2.5%	116	924	High Cibil Score	2.90000	34.80000	360
2.5%	130	951	High Cibil Score	3.25000	39.00000	360
5.0%	50	933	High Cibil Score	2.50000	30.00000	240
2.5%	99	985	High Cibil Score	2.47500	29.70000	360

Output 4:

customer_id	loan_id	customer_name	gender	age	married	education	self_employed	region_id	postal_code	segment	state	region	applicantincome
IP43002	LP001003	Darrin Van Huff	Male	66	Yes	Graduate	No	13.2	90036	Corporate	California	West	4583
IP43007	LP001013	Harold Pawlan	Male	45	Yes	Not Graduate	No	13.3	76106	Home Office	Texas	North	2333
IP43011	LP001024	Ken Black	Male	48	Yes	Graduate	No	13.3	68025	Corporate	Nebraska	North	3200
IP43016	LP001032	Matt Abelman	female	51	No	Graduate	No	13.3	77095	Home Office	Texas	North	4950
IP43017	LP001034	Gene Hale	Male	20	No	Not Graduate	No	13.3	75080	Corporate	Texas	North	3596
IP43018	LP001036	Steve Nguyen	male	27	No	Graduate	No	13.3	77041	Home Office	Texas	North	3510
IP43019	LP001038	Linda Cazamias	Male	64	Yes	Not Graduate	No	13.3	60540	Corporate	Illinois	North	4887
IP43020	LP001041	Ruben Ausman	Male	66	Yes	Graduate	NULL	13.2	90049	Corporate	California	West	2600
IP43021	LP001043	Erin Smith	Male	40	Yes	Not Graduate	No	13.2	32935	Corporate	Florida	West	7660
IP43022	LP001046	Odella Nelson	Male	23	Yes	Graduate	No	13.3	55122	Corporate	Minnesota	North	5955

Conclusion:

This Loan Management System project brings together different datasets and business rules to streamline loan management. By automating the classification of customers, updating CIBIL score statuses, and calculating loan interest amounts, the system provides a comprehensive approach to managing loans and customer data efficiently.

Thank you!