LOAN MANAGEMENT SYSTEM

My SQL Project

FAZIL HUSSAIN Z

INTRODUCTION TO MY SQL

MySQL is an open-source relational database management system (RDBMS) that uses SQL(Structured Query Language) for managing data. It is known for its speed, reliability, and ease of use, making it popular for web applications, enterprise solutions, and data-driven applications. MySQL supports features like transactions, indexing, replication, and stored procedures, and it integrates well with programming languages like Python, Java, and PHP. It is widely used in platforms such as Wordpress, e-commerce systems, and cloud applications.

KEY FEATURES

High Performance, Scalability, Replication & Clustering, Security Features, Cross-Platform Support, Integration

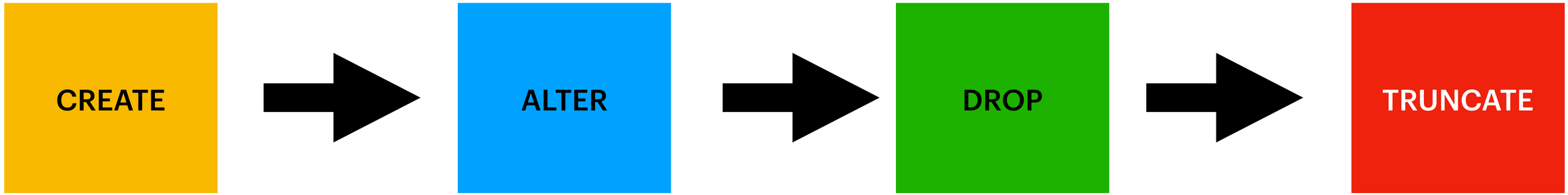
SQL STATEMENTS

An SQL statement is a command used to interact with a relational database to perform tasks like retrieving, inserting, updating, deleting, or managing data structures.

- DDL - Data Definition Language
- DML - Data Manipulation Language
- DCL - Data Control Language
- DQL - Data Query Language
- TCL - Transaction Control Language

Data Definition Language

DDL (Data Definition Language) statements are used in SQL to define and manage database structures such as tables, schemas, indexes, and constraints. These statements do not manipulate data but rather define its structure.

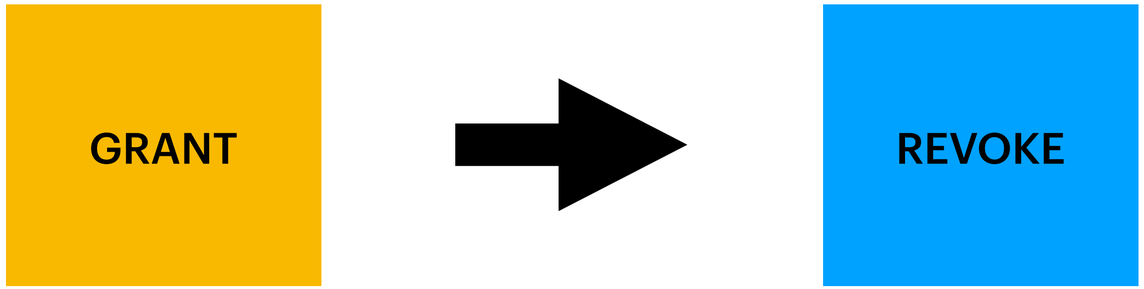


Data Manipulation Language

DML statements in SQL are used to manipulate and manage data stored in database tables. These statements allow users to insert, update, delete, and retrieve data.

Data Control Language

DCL statements in SQL are used to control access and permissions for database users.These statements help in securing data by granting or revoking privileges.



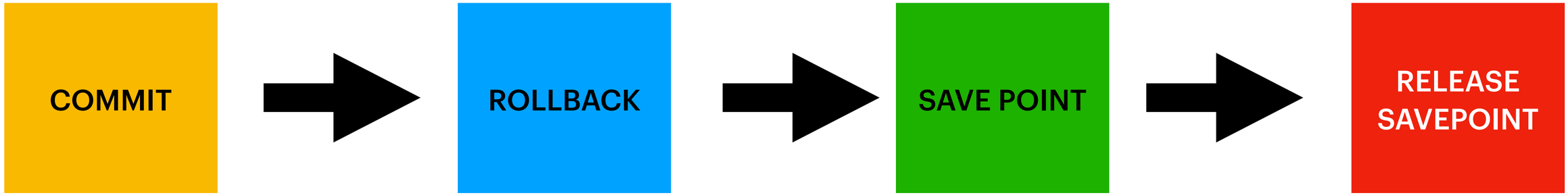
Data Query Language

DQL is used to retrieve data from a database using the SELECT statement. It allows users to perform queries to fetch specific data based on conditions.

SELECT

Transaction Control Language

TCL statements in SQL are used to manage transactions in a database. Transactions are sequences of SQL operations that should be executed as a single unit. TCL ensures data integrity by committing or rolling back changes based on success or failure.



PROJECT INTRODUCTION

LOAN MANAGEMENT SYSTEM

- The Loan Management System (LMS) is a database-driven application designed to manage the complete loan lifecycle efficiently. It allows users to apply for loans, process approvals, track repayments, and generate financial reports. The system ensures accuracy in interest calculations, EMI tracking, and penalty management for delayed payments.

- By utilising SQL for data management, the system provides a structured approach to handling loan applications, borrower details, and repayment schedules. It also enhances security through role-based access control, ensuring that only authorised users can manage loan records. This project aims to streamline loan processing, minimise manual errors, and improve financial decision-making for lenders and borrowers alike.

DATA SET

The Loan Management System dataset is a structured collection of data that stores and manages loan-related information, including borrower details, loan applications, approvals, repayments, and financial transactions. It provides a well organised database schema that helps in efficient data handling, loan tracking, and financial reporting.

- Customer income status
- Loan status
- Customer info
- Country state
- Region info

CREATING DATABASE

Creating database as "Project", to use the database and set the auto commit "off" to start the transaction.

```
create database project;
use project;
set autocommit = off;
start transaction;
```

TABLE IMPORT AND SET CRITERIA

After importing customer income data ,set the criteria based on the applicant income, in this project we implement "case-end statement" achieve the criteria.

```
-- Import table from sheet 1- customer income status
select * from customer_income;
select count(*) from customer_income;

-- set customer criteria based on applicant income
/* • Applicant income >15,000 = grade a
   • Applicant income >9,000 = grade b
   • Applicant income >5000 = middle class customer
   •  Otherwise low class
    (Create this as new table)*/

create table Pro
select *, case
when Applicant_income > 15000 then "Grade A"
when Applicant_income > 9000 then "Grade B"
when Applicant_income > 5000 then "Middle_class_customer"
else "Low class"
end as Grades
from customer_income;
```

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

Loan_ID	Customer_ID	Applicant_income	Coapplican_income	Property_Area	Loan_Status	Grades
LP001002	IP43001	5849	0	Urban	Y	Middle_class_customer
LP001003	IP43002	4583	1508	Rural	N	Low class
LP001005	IP43003	3000	0	Urban	Y	Low class
LP001006	IP43004	2583	2358	Urban	Y	Low class
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
LP001014	IP43008	3036	2504	Semiurban	N	Low class
LP001018	IP43009	4006	1526	Urban	Y	Low class
LP001020	IP43010	12841	10968	Semiurban	N	Grade B
LP001024	IP43011	3200	700	Urban	Y	Low class
LP001027	IP43012	2500	1840	Urban	Y	Low class
LP001028	IP43013	3073	8106	Urban	Y	Low class
LP001029	IP43014	1853	2840	Rural	N	Low class
LP001030	IP43015	1299	1086	Urban	Y	Low class

Pro 2

SET CRITERIA BASED ON INCOME AND AREA

Then Calculate the monthly interest percentage based on income and area by using "case-end statement".

```
-- Monthly interest percentage
/* • Applicant income <5000 rural=3%
   • Applicant income <5000 semi rural=3.5%
   • Applicant income <5000 urban=5%
   • Applicant income <5000 semi urban= 2.5%
    Otherwise =7% */

create table Pro1
select *,case
when Applicant_income <5000 then
case
when Property_Area = "rural" then 3
when Property_Area = "semi_rural" then 3.5
when Property_Area = "urban" then 5
when Property_Area = "semi urban" then 2.5
else 7
end
end as int_per
from customer_income;
```

CRITERIA BASED ON INCOME AND AREA

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

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

Pro1 3

TRIGGER

A trigger in SQL is a special stored procedure that automatically executes in response to certain events (INSERT, UPDATE, DELETE) on a table. Triggers help enforce business rules, maintain data integrity, and automate processes in a Loan Management System (LMS.

- ROW LEVEL TRIGGER
- STATEMENT LEVEL TRIGGER

In SQL, triggers are classified based on when and why they execute. They help automate tasks, maintain data consistency, and enforce business rules in a Loan Management System (LMS)

- BEFORE TRIGGER
- AFTER TRIGGER

SET TRIGGERS

- Create a table to get the status of loan and Cibil score. Before insert the value into the table, initialise the "before insertion trigger" and set the criteria condition inside the trigger.

- Once the value is inserted, trigger automatically get fired and then it check for condition depends on that it will change the value on the specified column.

```sql
-- Primary Table
create table loan_status_copy
(loan_id varchar(50), customer_id varchar(50), loan_amount varchar(50),
loan_amount_term int, cibil_score int,
primary key (loan_id));

-- Secondary Table
create table loan_remark (
Loan_id varchar(50),Loan_amount varchar(50),
Cibil_Score int,Cibil_Score_status varchar (50),
primary key (loan_id));
```

ROW AND STATEMENT LEVEL TRIGGERS

- Create row level trigger for loan amt
- Create statement level trigger for cibil score

```
delimiter //
create trigger Cibil_score before insert on
loan_status_copy for each row
begin
if new.Loan_amount is null then set new.Loan_amount = "Loan is still processing";
end if ;
insert into loan_remark (loan_id,loan_amount,cibil_score,cibil_score_status)
values (new.loan_id,new.loan_amount,new.cibil_score,
case
when new.cibil_score > 900 then "High cibil score"
when new.cibil_score > 750 then "no penalty"
when new.cibil_score > 0 then "Penalty customers"
else "Loan cannot apply"
end );
end //
delimiter ;
```

TRIGGER OUTPUT

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Loan_id	Loan_amount	Cibil_Score	Cibil_Score_status
LP001002	Loan is still processing	303	Penalty customers
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

loan_remark 16

ALTER AND CREATE NEW TABLE WITH BELOW DATA

After execution and insertion, we modify datatype of loan. amount into int and join two tables, remove customers with "rejected" and "loan still processing" statuses, modify the data type of the loan amount column, and create a new table based on the loan status table.

```sql
-- Then delete the reject and loan still processing customers
delete from loan_remark where loan_amount = "Loan is still processing" or
Cibil_Score_status = "loan cannot apply";

-- Update loan as integers
alter table loan_remark modify loan_amount int;

create table customer_interest_analysis as
select loan_id, loan_amount, int_per,
ROUND((int_per / 100) * loan_amount, 0) as monthly_int,
ROUND(((int_per / 100) * loan_amount) * 12, 0) as annual_int
from Amount;
```

CUSTOMER ANALYSIS TABLE

Rejected and Loan still processing status are deleted and monthly and annual interest field has been added.

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

Loan_id	loan_amount	Cibil_Score	Cibil_Score_status	int_per	monthly_int	annual_int
LP001005	66	606	Penalty customers	5.0	3	40
LP001006	120	851	no penalty	5.0	6	72
LP001008	141	420	Penalty customers	7.0	10	118
LP001011	267	173	Penalty customers	7.0	19	224
LP001013	95	650	Penalty customers	5.0	5	57
LP001014	158	471	Penalty customers	7.0	11	133
LP001018	168	863	no penalty	5.0	8	101

PL 22

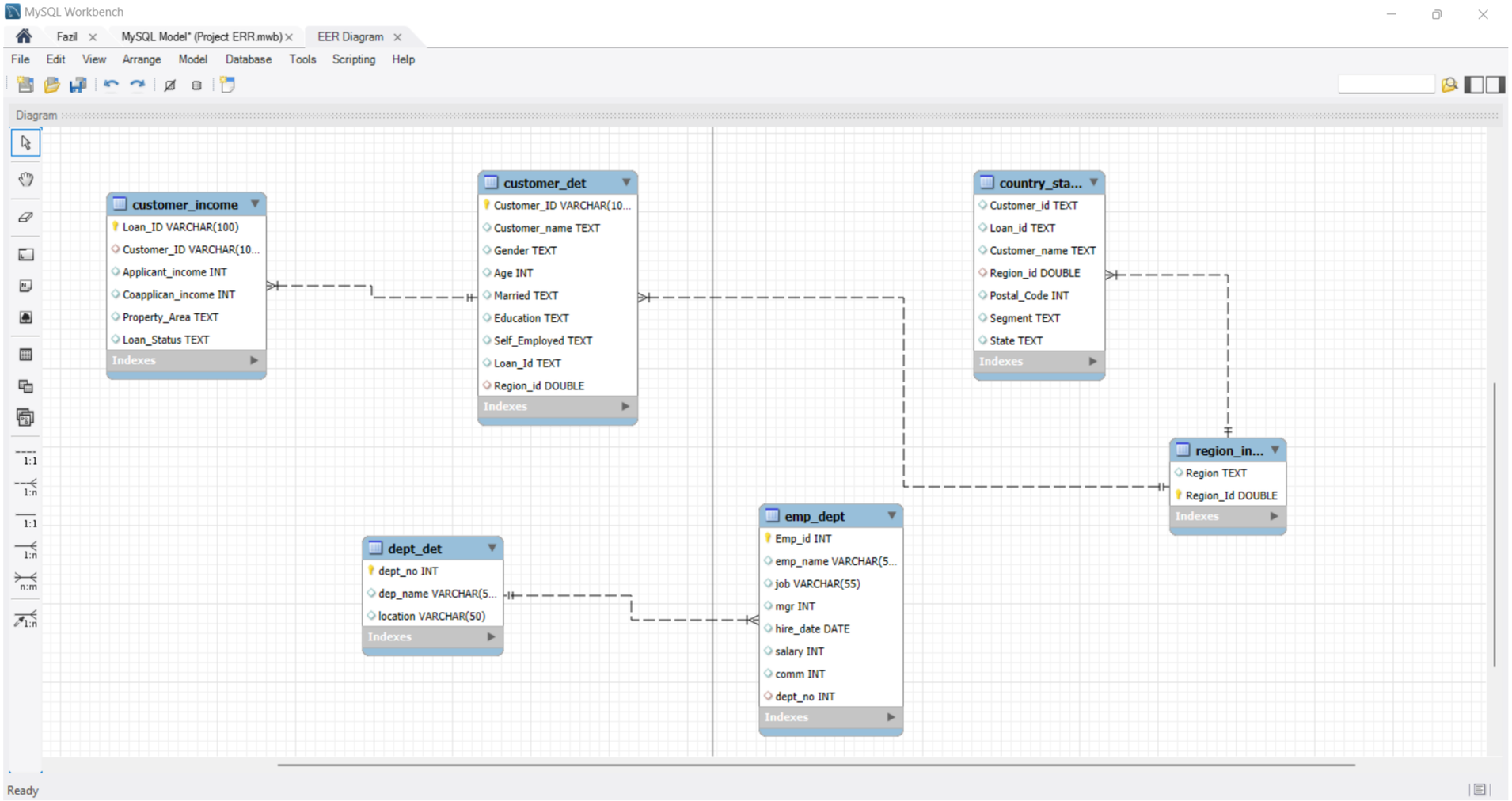
FOREIGN KEY

A foreign key in SQL is used to link two tables. It creates a relationship between the column(s) in one table (the child) and the primary key in another table (the parent). This helps maintain referential integrity.

```
-- foreign key
alter table customer_income add constraint customer_foreign_key foreign key(customer_id)
references customer_det(customer_id) on delete cascade;
alter table country_state add constraint country_region_fk foreign key(region_id)
references region_info(region_id) on update cascade;
alter table customer_det add constraint customer_region_fk foreign key (region_id)
references region_info(region_id) on update cascade;
```

ENTITY-RELATIONSHIP DIAGRAM

An ER (Entity-Relationship) diagram describes the structure of a database in terms of entities, attributes, and relationships. When translating an ER diagram into SQL, you create CREATE TABLE statements that reflect those entities, their attributes (columns), and the relationships (foreign keys).



UPDATE CUSTOMER INFO

Update some customers info like gender and age using case end statement.

```
update customer_det set gender = case
when customer_id = "IP43006" then "Female"
when customer_id = "IP43016" then "Female"
when customer_id = "IP43018" then "Male"
when customer_id = "IP43038" then "Male"
when customer_id = "IP43508" then "Female"
when customer_id = "IP43577" then "Female"
when customer_id = "IP43589" then "Female"
when customer_id = "IP43593" then "Female"
else gender
end,
age = case
when customer_id = "IP43007" then 45
when customer_id = "IP43009" then 32
else age
end;
```

OUTPUT FOR UPDATED DATA

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

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

customer_det 23

PROCEDURE

A Stored Procedure in SQL is a predefined set of SQL statements that can be executed as a single unit. It helps automate processes like loan approvals, repayments, and financial calculations in a Loan Management System (LMS).

Benefits of Using Stored Procedures

- Automation: Reduces manual effort in loan processing
- Efficiency: Optimised execution for complex operations
- Security: Limits direct database access to users
- Reusability: Can be used multiple times without rewriting SQL queries

PROCEDURE

INPUT - 1

```
-- Sheet 4 and 5- country state and region
-- Join all the 5 tables without repeating the fields - output 1
select c.*,
A.applicant_income, A.coapplican_income, A.property_area, A.loan_status,
A.int_per, A.loan_amount, A.cibil_score, A.cibil_score_status,
s.postal_code, s.segment, s.state,
i.monthly_int, i.annual_int,
r.region
from customer_det c
inner join Amount A on c.loan_id = A.loan_id
inner join country_state s on c.loan_id = s.loan_id
inner join customer_interest_analysis i on c.loan_id = i.loan_id
inner join region_info r on c.region_id = r.region_id;

select*from customer_det;
```

PROCEDURE

Output - 1

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

Customer_ID	Customer_name	Gender	Age	Married	Education	Self_Employed	Loan_Id	Region_id	applicant_income	coapplican_income	property_area	loan_status	int_per	loan_amount	cibil_score	cibil_score_status	postal_code	segment	state
IP43002	Darrin Van Huff	Male	66	Yes	Graduate	No	LP001003	13.2	4583	1508	Rural	N	3.0	128	920	High cibil score	90036	Corporate	Califor
IP43003	Sean O'Donnell	Male	20	Yes	Graduate	Yes	LP001005	13.2	3000	0	Urban	Y	5.0	66	606	Penalty customers	33311	Consumer	Florida
IP43004	Brosina Hoffman	Male	46	Yes	Not Graduate	No	LP001006	13.2	2583	2358	Urban	Y	5.0	120	851	no penalty	90032	Consumer	Califor
IP43005	Andrew Allen	Male	18	No	Graduate	No	LP001008	13.2	6000	0	Urban	Y	7.0	141	420	Penalty customers	28027	Consumer	North
IP43006	Irene Maddox	Female	66	Yes	Graduate	Yes	LP001011	13.2	5417	4196	Urban	Y	7.0	267	173	Penalty customers	98103	Consumer	Washi
IP43007	Harold Pawlan	Male	45	Yes	Not Graduate	No	LP001013	13.3	2333	1516	Urban	Y	5.0	95	650	Penalty customers	76106	Home Office	Texas
IP43008	Pete Kriz	Male	41	Yes	Graduate	No	LP001014	13.3	3036	2504	Semiurban	N	7.0	158	471	Penalty customers	53711	Consumer	Wisco
IP43009	Alejandro Grove	Male	32	Yes	Graduate	No	LP001018	13.2	4006	1526	Urban	Y	5.0	168	863	no penalty	84084	Consumer	Utah

Result 24 | Read Only

PROCEDURE

INPUT - 2

```sql
-- Find the mismatch details using joins - output 2
select c.*, s.postal_code, s.segment, s.state, r.region
from region_info r
left join customer_det c on r.region_id = c.region_id
left join country_state s on r.region_id = s.region_id
where c.region_id is null;
```

OUTPUT - 2

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

Customer_ID	Customer_name	Gender	Age	Married	Education	Self_Employed	Loan_Id	Region_id	postal_code	segment	state	region
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	South

Result 32

PROCEDURE

INPUT - 3

```sql
 -- Filter high cibil score - output 3
select c.*,
A.applicant_income, A.coapplican_income, A.property_area, A.loan_status,
A.int_per, A.loan_amount, A.cibil_score, A.cibil_score_status,
s.postal_code, s.segment, s.state,
i.monthly_int, i.annual_int,
r.region
from customer_det c
inner join Amount A on c.loan_id = A.loan_id
inner join country_state s on c.loan_id = s.loan_id
inner join customer_interest_analysis i on c.loan_id = i.loan_id
inner join region_info r ON c.region_id = r.region_id
where A.cibil_score_status = 'High cibil score';
```

PROCEDURE

Output - 3

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

Customer_ID	Customer_name	Gender	Age	Married	Education	Self_Employed	Loan_Id	Region_id	applicant_income	coapplican_income	property_area	loan_status	int_per	loan_amount	cibil_score	cibil_score_status	postal_code	segment	state
IP43002	Darrin Van Huff	Male	66	Yes	Graduate	No	LP001003	13.2	4583	1508	Rural	N	3.0	128	920	High cibil score	90036	Corporate	California
IP43013	Emily Burns	Male	28	Yes	Graduate	No	LP001028	13.2	3073	8106	Urban	Y	5.0	200	928	High cibil score	84057	Consumer	Utah
IP43022	Odella Nelson	Male	23	Yes	Graduate	No	LP001046	13.3	5955	5625	Urban	Y	7.0	315	903	High cibil score	55122	Corporate	Minnesot
IP43027	Ted Butterfield	Male	59	Yes	Graduate	No	LP001068	13.4	2799	2253	Semiurban	Y	7.0	122	999	High cibil score	12180	Consumer	New York
IP43031	Karen Daniels	Male	18	Yes	Graduate	NULL	LP001091	13.2	4166	3369	Urban	N	5.0	201	972	High cibil score	22153	Consumer	Virginia
IP43060	Troy Staebel	Male	57	Yes	Not Graduate	No	LP001199	13.2	3357	2859	Urban	Y	5.0	144	949	High cibil score	85023	Consumer	Arizona
IP43065	Sally Hughsby	Female	49	No	Graduate	No	LP001222	13.2	4166	0	Semiurban	N	7.0	116	924	High cibil score	94122	Corporate	California
IP43080	Chad Sievert	Male	20	Yes	Not Graduate	Yes	LP001264	13.2	3333	2166	Semiurban	Y	7.0	130	951	High cibil score	90004	Consumer	California
IP43085	Robert Marley	Male	61	Yes	Graduate	No	LP001275	13.2	3988	0	Urban	Y	5.0	50	933	High cibil score	71203	Home Office	Louisiana
IP43087	Frank Merwin	Male	43	Yes	Not Graduate	No	LP001280	13.2	3333	2000	Semiurban	Y	7.0	99	985	High cibil score	90032	Home Office	California
IP43098	Laurel Elliston	Male	40	Yes	Graduate	No	LP001333	13.2	1977	997	Semiurban	Y	7.0	50	994	High cibil score	90604	Consumer	California
IP43117	Roy Collins	Female	36	Yes	Graduate	No	LP001404	13.3	3167	2283	Semiurban	Y	7.0	154	919	High cibil score	60610	Consumer	Illinois

Result 5 ×

Read Only

PROCEDURE

INPUT - 4

```sql
-- Filter home office and corporate - output 4
SELECT c.*,
A.applicant_income, A.coapplican_income, A.property_area, A.loan_status,
A.int_per, A.loan_amount, A.cibil_score, A.cibil_score_status,
s.postal_code, s.segment, s.state,
i.monthly_int, i.annual_int,
r.region
from customer_det c
inner join Amount A on c.loan_id = A.loan_id
inner join country_state s on c.loan_id = s.loan_id
inner join customer_interest_analysis i on c.loan_id = i.loan_id
inner join region_info r on c.region_id = r.region_id
where s.segment in ('home office','corporate');
```

PROCEDURE

Output - 4

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

Customer_ID	Customer_name	Gender	Age	Married	Education	Self_Employed	Loan_Id	Region_id	applicant_income	coapplican_income	property_area	loan_status	int_per	loan_amount	cibil_score	cibil_score_status	postal_code	segment	state
IP43002	Darrin Van Huff	Male	66	Yes	Graduate	No	LP001003	13.2	4583	1508	Rural	N	3.0	128	920	High cibil score	90036	Corporate	Californi
IP43007	Harold Pawlan	Male	45	Yes	Not Graduate	No	LP001013	13.3	2333	1516	Urban	Y	5.0	95	650	Penalty customers	76106	Home Office	Texas
IP43011	Ken Black	Male	48	Yes	Graduate	No	LP001024	13.3	3200	700	Urban	Y	5.0	70	143	Penalty customers	68025	Corporate	Nebrask
IP43016	Matt Abelman	Female	51	No	Graduate	No	LP001032	13.3	4950	0	Urban	Y	5.0	125	477	Penalty customers	77095	Home Office	Texas
IP43017	Gene Hale	Male	20	No	Not Graduate	No	LP001034	13.3	3596	0	Urban	Y	5.0	100	888	no penalty	75080	Corporate	Texas
IP43018	Steve Nguyen	Male	27	No	Graduate	No	LP001036	13.3	3510	0	Urban	N	5.0	76	387	Penalty customers	77041	Home Office	Texas
IP43019	Linda Cazamias	Male	64	Yes	Not Graduate	No	LP001038	13.3	4887	0	Rural	N	3.0	133	371	Penalty customers	60540	Corporate	Illinois
IP43020	Ruben Ausman	Male	66	Yes	Graduate	NULL	LP001041	13.2	2600	3500	Urban	Y	5.0	115	537	Penalty customers	90049	Corporate	Californi
IP43021	Erin Smith	Male	40	Yes	Not Graduate	No	LP001043	13.2	7660	0	Urban	N	7.0	104	534	Penalty customers	32935	Corporate	Florida
IP43022	Odella Nelson	Male	23	Yes	Graduate	No	LP001046	13.3	5955	5625	Urban	Y	7.0	315	903	High cibil score	55122	Corporate	Minneso
IP43026	Janet Molinari	Male	45	Yes	Graduate	Yes	LP001066	13.4	9560	0	Semiurban	Y	7.0	191	293	Penalty customers	10024	Corporate	New Yor
IP43029	Paul Stevenson	Male	25	No	Not Graduate	No	LP001086	13.3	1442	0	Urban	N	5.0	35	521	Penalty customers	60610	Home Office	Illinois
IP43030	Brendan Sweed	Female	29	No	Graduate	NULL	LP001087	13.2	3750	2083	Semiurban	Y	7.0	120	620	Penalty customers	85234	Corporate	Arizona

Result 6 ×

Read Only

PROCEDURE

Store all the outputs as procedure

Final Output

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

Customer_ID	Customer_name	Gender	Age	Married	Education	Self_Employed	Loan_Id	Region_id	applicant_income	coapplican_income	property_area	loan_status	int_per	loan_amount	cibil_score	cibil_score_status	postal_code	segment	state
IP43002	Darrin Van Huff	Male	66	Yes	Graduate	No	LP001003	13.2	4583	1508	Rural	N	3.0	128	920	High cibil score	90036	Corporate	Californi
IP43007	Harold Pawlan	Male	45	Yes	Not Graduate	No	LP001013	13.3	2333	1516	Urban	Y	5.0	95	650	Penalty customers	76106	Home Office	Texas
IP43011	Ken Black	Male	48	Yes	Graduate	No	LP001024	13.3	3200	700	Urban	Y	5.0	70	143	Penalty customers	68025	Corporate	Nebrask
IP43016	Matt Abelman	Female	51	No	Graduate	No	LP001032	13.3	4950	0	Urban	Y	5.0	125	477	Penalty customers	77095	Home Office	Texas
IP43017	Gene Hale	Male	20	No	Not Graduate	No	LP001034	13.3	3596	0	Urban	Y	5.0	100	888	no penalty	75080	Corporate	Texas
IP43018	Steve Nguyen	Male	27	No	Graduate	No	LP001036	13.3	3510	0	Urban	N	5.0	76	387	Penalty customers	77041	Home Office	Texas
IP43019	Linda Cazamias	Male	64	Yes	Not Graduate	No	LP001038	13.3	4887	0	Rural	N	3.0	133	371	Penalty customers	60540	Corporate	Illinois
IP43020	Ruben Ausman	Male	66	Yes	Graduate	NULL	LP001041	13.2	2600	3500	Urban	Y	5.0	115	537	Penalty customers	90049	Corporate	Californi
IP43021	Erin Smith	Male	40	Yes	Not Graduate	No	LP001043	13.2	7660	0	Urban	N	7.0	104	534	Penalty customers	32935	Corporate	Florida
IP43022	Odella Nelson	Male	23	Yes	Graduate	No	LP001046	13.3	5955	5625	Urban	Y	7.0	315	903	High cibil score	55122	Corporate	Minneso

Result 14 | Result 15 | Result 16 | Result 17 ×

Read Only

THANK YOU