

The background features several abstract, organic shapes in shades of purple and blue. A large, irregular shape is in the top right, a smaller circle is in the top center, and another shape is in the bottom right. The shapes have a gradient effect, with some areas appearing lighter and others darker.

# MY SQL

(Structured Query Language)



## SQL Presentation Index

- Introduction to SQL
- SQL Data Types
- SQL Commands (DDL, DML, DCL, TCL)
- SQL Constraints
- SQL Joins
- Aggregate Functions & Grouping
- Subqueries
- SQL Constraints
- Views and Indexes
- Transactions
- PROJECT SUMMARY

# Introduction to SQL

SQL (Structured Query Language) is a programming language used to manage, manipulate, and retrieve data from relational databases. It is widely used in database management systems such as MySQL, PostgreSQL, SQL Server, Oracle, and SQLite.

- ✓ Data Querying – Retrieve specific data using SELECT statements.
- ✓ Data Manipulation – Insert, update, and delete records (INSERT, UPDATE, DELETE).
- ✓ Data Definition – Create and modify database structures (CREATE, ALTER, DROP).
- ✓ Data Control – Manage access permissions (GRANT, REVOKE).
- ✓ Transaction Control – Ensure data integrity with transactions (COMMIT, ROLLBACK).

SQL is not case-sensitive (but keywords are usually written in uppercase for readability).

#### 1. Numeric Data Types

- INT
- TINYINT
- SMALLINT
- DECIMAL
- FLOAT



# SQL Data Types



#### 2. String (Character) Data Types

- VARCHAR
- TEXT

#### 4. Boolean Data Type

- BOOLEAN – Stores TRUE (1) or FALSE (0)



#### 3. Date & Time Data Types

- DATE (YYYY-MM-DD)
- TIME (HH:MM:SS)
- DATETIME

# SQL Commands

- » DDL
- » DML
- » DCL
- » TCL

# DDL

- DDL (Data Definition Language) in SQL is used to define and manage database structures like tables, schemas, and indexes. DDL commands don't manipulate data but rather deal with the schema of the database. The main DDL

## 1. CREATE

```
CREATE TABLE employees (  
  emp_id INT PRIMARY KEY,  
  name VARCHAR(50),  
  age INT,  
  department VARCHAR(50)  
);
```

## 2. ALTER

```
ALTER TABLE employees ADD salary DECIMAL(10,2);  
ALTER TABLE employees MODIFY COLUMN name VARCHAR(100);  
ALTER TABLE employees DROP COLUMN age;
```

## 3. DROP

```
DROP TABLE employees;  
DROP DATABASE company;
```

## 4. TRUNCATE

```
TRUNCATE TABLE employees;
```

## 5. RENAME

```
RENAME TABLE employees TO staff;  
ALTER TABLE staff RENAME COLUMN department TO dept;
```

# DML

DML (Data Manipulation Language) in SQL  
DML is used to manipulate and retrieve data in database tables. It includes commands for inserting, updating, deleting, and selecting data.

## 1. INSERT

```
INSERT INTO employees (emp_id, name, age, department, salary)  
VALUES (101, 'John Doe', 30, 'IT', 50000);
```

### Insert multiple records

```
INSERT INTO employees (emp_id, name, age, department, salary)  
VALUES  
(102, 'Alice Smith', 28, 'HR', 45000),  
(103, 'Bob Johnson', 35, 'Finance', 55000);
```

## 2. UPDATE

### Update multiple columns

```
UPDATE employees  
SET salary = 60000  
WHERE emp_id = 101;
```

```
UPDATE employees  
SET age = 32, department = 'Admin'  
WHERE emp_id = 101;
```



### 3. DELETE

```
DELETE FROM employees  
WHERE emp_id = 102;
```

### Delete all records from a table

```
DELETE FROM employees;
```

### 4. SELECT

```
SELECT * FROM employees;
```

### Select specific columns

```
SELECT name, department, salary  
FROM employees  
WHERE age > 30;
```

### Using LIMIT to fetch a specific number of records

```
SELECT * FROM employees  
LIMIT 5;
```

### Using ORDER BY to sort data

```
SELECT * FROM employees  
ORDER BY salary DESC;
```



# DCL

- DCL (Data Control Language) is used to control access and permissions in a database. It mainly includes two

## 1.GRANT

```
GRANT SELECT, INSERT ON Employees TO user1;
```

## 2.REVOKE

```
REVOKE INSERT ON Employees FROM user1;
```

# TCL

TCL (Transaction Control Language) in SQL  
TCL commands in SQL are used to manage transactions in a database. A transaction is a sequence of operations that are executed as a single unit of work. TCL ensures data integrity by controlling when changes are saved or undone.

## 1. COMMIT

```
BEGIN TRANSACTION;  
UPDATE employees SET salary = 55000 WHERE emp_id = 101;  
COMMIT; -- Changes are saved permanently
```

## 2. ROLLBACK

```
BEGIN TRANSACTION;  
UPDATE employees SET salary = 60000 WHERE emp_id = 102;  
ROLLBACK; -- Changes are undone, and salary remains unchanged
```

## 3. SAVEPOINT

```
BEGIN TRANSACTION;  
UPDATE employees SET salary = 60000 WHERE emp_id = 103;  
SAVEPOINT sp1;  
  
UPDATE employees SET salary = 70000 WHERE emp_id = 104;  
ROLLBACK TO sp1; -- Only the second update is undone, first update remains  
  
COMMIT; -- Saves changes permanently
```

## 4. SET TRANSACTION

```
SET TRANSACTION READ ONLY;
```

# SQL Constraints

## 1. NOT NULL

```
CREATE TABLE Employees (  
  ID INT NOT NULL,  
  Name VARCHAR(100) NOT NULL  
);
```

## 2. UNIQUE

```
CREATE TABLE Employees (  
  Email VARCHAR(255) UNIQUE  
);
```

## 3. PRIMARY KEY

```
CREATE TABLE Employees (  
  ID INT PRIMARY KEY,  
  Name VARCHAR(100)  
);
```

## 4. FOREIGN KEY

```
CREATE TABLE Orders (  
  OrderID INT PRIMARY KEY,  
  CustomerID INT,  
  FOREIGN KEY (CustomerID) REFERENCES Customers(ID)  
);
```

## 5. CHECK

```
CREATE TABLE Employees (  
  Age INT CHECK (Age >= 18)  
);
```

## 6. DEFAULT

```
CREATE TABLE Employees (  
  Status VARCHAR(20) DEFAULT 'Active'  
);
```

## 7. INDEX

```
CREATE INDEX idx_lastname ON Employees (LastName);
```

# JOINS IN SQL



## SQL Joins

SQL JOINS are used to combine rows from two or more tables based on a related column between them.

### 1. INNER JOIN

Returns only matching records from both tables.

### 2. LEFT JOIN (LEFT OUTER JOIN)

Returns all records from the left table and matching records from the right table. If no match is found, NULL is returned from the right table.

### 3. RIGHT JOIN (RIGHT OUTER JOIN)

Returns all records from the right table and matching records from the left table. If no match is found, NULL is returned from the left table.

### 4. FULL JOIN (FULL OUTER JOIN)

Returns all records from both tables, with NULLs where there's no match.

## Aggregate Functions & Grouping in SQL

Aggregate functions perform a calculation on a set of values and return a single value. They're commonly used with the GROUP BY clause.

```
SELECT department, COUNT(*) AS total_employees
FROM employees
GROUP BY department;
```

### GROUP BY Clause

Used to group rows that have the same values in specified columns. Aggregate functions are then applied per group.

```
SELECT column, AGG_FUNC(column)
FROM table
GROUP BY column;
```

### HAVING Clause (Filter After Grouping)

Use HAVING to filter groups, similar to how WHERE filters rows.

```
SELECT department, COUNT(*) AS total
FROM employees
GROUP BY department
HAVING COUNT(*) > 5;
```

# Subqueries in SQL

A Subquery (also called an inner query or nested query) is a query inside another query. It is used to retrieve data that will be used in the main (outer) query.

## Single-row Subquery

```
SELECT name, salary
FROM employees
WHERE salary = (SELECT MAX(salary) FROM employees);
```

## Multi-row Subquery

```
SELECT name
FROM employees
WHERE department_id IN (SELECT department_id FROM departments WHERE location = 'New York');
```

## Correlated Subquery Example

```
SELECT name
FROM employees e
WHERE salary > (
    SELECT AVG(salary)
    FROM employees
    WHERE department_id = e.department_id
);
```

NOT NULL



UNIQUE



# SQL Constraints



FOREIGN KEY



PRIMARY KEY



## NOT NULL

```
CREATE TABLE Students (  
  ID INT NOT NULL,  
  Name VARCHAR(100) NOT NULL  
);
```

## UNIQUE

```
CREATE TABLE Employees (  
  Email VARCHAR(100) UNIQUE  
);
```

## PRIMARY KEY

```
CREATE TABLE Customers (  
  CustomerID INT PRIMARY KEY,  
  Name VARCHAR(100)  
);
```

## FOREIGN KEY

```
CREATE TABLE Orders (  
  OrderID INT PRIMARY KEY,  
  CustomerID INT,  
  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)  
);
```

# Views and Indexes in SQL

A View is a virtual table based on the result of a SQL query.

It does not store data physically, but pulls data from one or more tables.

Used to simplify complex queries, enhance security, and present specific data to users.

## Index

An Index is a database object that improves the speed of data retrieval operations.  
Works like an index in a book — helps find data faster without scanning every row.

Speeds up SELECT queries.

Optimizes performance for JOIN, WHERE, ORDER BY.

```
CREATE INDEX index_name  
ON table_name (column1, column2);
```



```
CREATE INDEX idx_customer_name  
ON Customers (Name);
```

# Transactions

A transaction is a sequence of one or more SQL operations (queries) that are executed as a single unit of work.

A transaction must be completed entirely or not at all.

Ensures data integrity in situations like money transfers, order processing, etc.

Command	Description
<code>BEGIN</code> or <code>START TRANSACTION</code>	Starts a transaction
<code>COMMIT</code>	Saves all changes made during the transaction
<code>ROLLBACK</code>	Undoes all changes if there's an error
<code>SAVEPOINT</code>	Sets a point within a transaction for partial rollback
<code>SET TRANSACTION</code>	Sets properties like isolation level



## Bank Transfer

```
START TRANSACTION;

UPDATE Accounts
SET Balance = Balance + 1000
WHERE AccountID = 1;

UPDATE Accounts
SET Balance = Balance + 1000
WHERE AccountID = 2;

COMMIT;
```



## ROLLBACK

```
START TRANSACTION;

UPDATE Products SET Stock = Stock - 10 WHERE ProductID = 101;

-- Oops! Price update failed or gave wrong result
UPDATE Products SET Price = 'Free' WHERE ProductID = 101;

ROLLBACK; -- Undo both changes
```

# PROJECT SUMMARY

Create Database Project: Design and analyze customer loan data.

Import Data: Add datasets into the database and set a primary key for unique values.

Customer Criteria: Create a table ("Applicant Income Grades") based on income and property area.

Triggers: Use row-level and statement-level triggers to filter and remove unapproved customers, creating a new table for approved customers.

Interest Calculation: Join customer and loan tables to compute monthly and annual interest, saving results in a new table.

Project View Link : [github](#)



Update Customer Info: Add gender and age data using customer ID.

Regional Data: Use joins to map customers to regions (country, state, and region info).

Outputs:

Output 1: Full customer details without duplicate columns.

Output 2: Extract mismatching or null values.

High CIBIL Score: Retrieve high-score customers.

Home/Corporate Customers: Filter customers from specific groups.

Stored Procedures: Save outputs in procedures for future retrieval.

## REQUIRED DATASET

Data set

- customer income status
- loan status
- customer info
- country state
- region info

## Creating database and using the database

```
create database loan_management_systems;  
use loan_management_systems;
```

#	Time	Action	Message
✓ 1	19:59:04	create database loan_management_systems	1 row(s) affected
✓ 2	19:59:10	use loan_management_systems	0 row(s) affected



## Import table customer income

```
select * from customer_income;
```

	Loan_ID	Customer ID	ApplicantIncome	CoapplicantIncome	Property_Area	Loan_Status
▶	LP001002	SP43001	5849	0	Urban	Y
	LP001003	SP43002	4583	1508	Rural	N
	LP001005	SP43003	3050	0	Urban	Y
	LP001006	SP43004	2583	2358	Urban	Y
	LP001008	SP43005	6050	0	Urban	Y
	LP001011	SP43006	5417	4196	Urban	Y
	LP001013	SP43007	2333	1516	Urban	Y
	LP001014	SP43008	3036	2504	Semiurban	N
	LP001018	SP43009	4056	1526	Urban	Y
	LP001020	SP43010	12841	10968	Semiurban	N
	LP001024	SP43011	3250	700	Urban	Y
	LP001027	SP43012	2500	1840	Urban	Y
	LP001028	SP43013	3073	8106	Urban	Y
	LP001029	SP43014	1853	2840	Rural	N
	LP001030	SP43015	1299	1086	Urban	Y
	LP001032	SP43016	4950	0	Urban	Y
	LP001034	SP43017	3596	0	Urban	Y
	LP001036	SP43018	3510	0	Urban	N
	LP001038	SP43019	4887	0	Rural	N
	LP001041	SP43020	2650	3503	Urban	Y
	LP001043	SP43021	7660	0	Urban	N
	LP001046	SP43022	5955	5625	Urban	Y
	LP001047	SP43023	2650	1911	Semiurban	N
	LP001050	SP43024	3365	1917	Rural	N

## set criteria and create a table

### Criteria

- Applicant income >15,000 = grade a
- Applicant income >9,000 = grade b
- Applicant income >5000 = middle class customer
- Otherwise low class

Created this as new table 'CUSTOMER\_GRADE

```
create table Customer_Grade as select *,
case
    when applicantincome > 15000 then
        'Grade A'
    when applicantincome > 9000 then
        'Grade B'
    when applicantincome > 5000 then
        'Middle Class Customer'
    else
        'Low Class'
end as customer_grade from Customer_income;
```

## Output for the above query

	Loan_ID	Customer ID	ApplicantIncome	CoapplicantIncome	Property_Area	Loan_Status	customer_grade
▶	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
	LP001032	IP43016	4950	0	Urban	Y	Low Class
	LP001034	IP43017	3596	0	Urban	Y	Low Class
	LP001036	IP43018	3510	0	Urban	N	Low Class
	LP001038	IP43019	4887	0	Rural	N	Low Class

## Monthly interest percentage

### Criteria

- 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%

## Query for the above criteria

```
create table monthly_interest as 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;
```

## Output for the above Query

	Loan_ID	Customer ID	ApplicantIncome	CoapplicantIncome	Property_Area	Loan_Status	Monthly_interest_percentage
▶	LP001002	IP43001	5849	0	Urban	Y	7%
	LP001003	IP43002	4583	1508	Rural	N	3%
	LP001005	IP43003	3000	0	Urban	Y	5%
	LP001006	IP43004	2583	2358	Urban	Y	5%
	LP001008	IP43005	6000	0	Urban	Y	7%
	LP001011	IP43006	5417	4196	Urban	Y	7%
	LP001013	IP43007	2333	1516	Urban	Y	5%
	LP001014	IP43008	3036	2504	Semiurban	N	2.5%
	LP001018	IP43009	4006	1526	Urban	Y	5%
	LP001020	IP43010	12841	10968	Semiurban	N	7%
	LP001024	IP43011	3200	700	Urban	Y	5%
	LP001027	IP43012	2500	1840	Urban	Y	5%
	LP001028	IP43013	3073	8106	Urban	Y	5%
	LP001029	IP43014	1853	2840	Rural	N	3%
	LP001030	IP43015	1299	1086	Urban	Y	5%
	LP001032	IP43016	4950	0	Urban	Y	5%
	LP001034	IP43017	3596	0	Urban	Y	5%
	LP001036	IP43018	3510	0	Urban	N	5%
	LP001038	IP43019	4887	0	Rural	N	3%

## loan status - Create row level trigger for loan amt and inserting Values into Dummy-table

```
create table dummy_table (loan_id text, customer_id text, loanamount text, loan_amount_term int, cibil_score int);

delimiter //
create trigger loan_amt before insert on dummy_table for each row
begin
if new.loanamount is null then set new.loanamount = "loan still processing";
end if;
end //
delimiter ;
```

```
select * from dummy_table;
```

	loan_id	customer_id	loanamount	loan_amount_term	cibil_score
▶	LPO01002	IP43001	Loan still processing	360	303
	LPO01003	IP43002	128	360	920
	LPO01005	IP43003	66	360	606
	LPO01006	IP43004	120	360	851
	LPO01008	IP43005	141	360	420
	LPO01011	IP43006	267	360	173
	LPO01013	IP43007	95	360	650
	LPO01014	IP43008	158	360	471
	LPO01018	IP43009	168	360	863
	LPO01020	IP43010	349	360	730
	LPO01024	IP43011	70	360	143
	LPO01027	IP43012	109	360	384
	LPO01028	IP43013	200	360	928
	LPO01029	IP43014	114	360	455
	LPO01030	IP43015	17	120	564
	LPO01032	IP43016	125	360	477
	LPO01034	IP43017	100	240	888
	LPO01036	IP43018	76	360	387
	LPO01038	IP43019	133	360	371
	LPO01041	IP43020	115	0	537



## Create statement level trigger for cibil score

```
-- Primary table
select * from dummy_table;

-- Secondary table
create table cibil_score (loan_id text, loan_amount text, cibil_score int, cibil_score_status varchar(30));

create trigger cibil after insert on dummy_table for each row
begin
insert into cibil_score (loan_id, loan_amount, cibil_score, cibil_score_status)
values (new.loan_id, new.loanamount, 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"
when new.cibil_score <=0 then "reject customers"
end );
end //
delimiter ;
```

## Output for above Query

```
select * from cibil_score_status;
```

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
LP001028	200	928	high cibil score
LP001029	114	455	penalty customers

## Delete the reject and loan still processing customers

```
delete from cibil_score where cibil_score_status = "reject customers";  
delete from cibil_score where loan_amount = "Loan still processing";
```

## **New field creation based on interest**

- Calculate monthly interest amt and annual interest amt based on loan amt
- Create all the above fields as a table
- Table name – customer interest analysis
- (create this into a new table and connect with sheet 2 (loan status) bring the output)

## Query for above criteria

```
create table customer_interest select m.*,  
l.loan_amount, l.cibil_score,l.cibil_score_status,  
case  
when applicantincome < 3000 and property_area = "rural"  
then (loan_amount * 3 /100)  
when applicantincome < 5000 and property_area = "semirural"  
then (loan_amount * 3.5 / 100)  
when applicantincome < 5000 and property_area = "urban"  
then (loan_amount * 5 /100)  
when applicantincome < 5000 and property_area = "semiurban"  
then (loan_amount * 2.5 /100)  
else (loan_amount * 7 /100)  
end as Monthly_interest,
```

```
case  
when applicantincome < 5000 and property_area = "rural"  
then (loan_amount * 3 /100) * 12  
when applicantincome < 5000 and property_area = "semirural"  
then (loan_amount * 3.5 / 100) * 12  
when applicantincome < 5000 and property_area = "urban"  
then (loan_amount * 5 /100) * 12  
when applicantincome < 5000 and property_area = "semiurban"  
then (loan_amount * 2.5 /100) * 12  
else (loan_amount * 7 /100) * 12  
end as Annual_interest
```

```
from monthly_interest m inner join cibil_score l on m.loan_id = l.loan_id;  
select * from customer_interest;
```

## Output for above Query

con_ID	CustomerID	Applcardnum	Debitcardnum	Priority_Area	loan_Status	paid	north_yrnet_percent	lansum	dbscore	dbscore_status	north_yrnet	annual_yrnet
P0001	P4001	493	158	Fixed	N	low debt	1	128	92	high db score	1.9400	4.0000
P0001	P4001	388	0	Other	Y	low debt	1	86	64	penalty customer	1.3000	3.6000
P0006	P4004	283	238	Other	Y	low debt	1	128	81	no penalty	6.0000	71.0000
P0008	P4005	608	0	Other	Y	middle debt customer	7	141	92	penalty customer	5.8700	118.4400
P0011	P4006	547	436	Other	Y	middle debt customer	7	267	175	penalty customer	31.4900	124.2800
P0011	P4007	233	128	Other	Y	low debt	1	95	60	penalty customer	4.7000	17.0000
P0014	P4008	326	284	Secured	N	low debt	7	118	47	penalty customer	11.3600	112.7200
P0018	P4009	406	126	Other	Y	low debt	1	168	80	no penalty	8.4000	101.8000
P0020	P4010	124	1088	Secured	N	good	7	248	70	penalty customer	24.4200	203.1600
P0024	P4011	118	78	Other	Y	low debt	1	71	140	penalty customer	1.5000	41.0000
P0027	P4012	258	184	Other	Y	low debt	1	108	34	penalty customer	1.4000	61.4000
P0028	P4013	373	818	Other	Y	low debt	1	208	128	high db score	10.0000	110.0000
P0029	P4014	183	284	Fixed	N	low debt	1	114	43	penalty customer	2.4000	41.0000
P0030	P4015	126	108	Other	Y	low debt	1	17	34	penalty customer	1.8000	11.2000
P0032	P4016	493	0	Other	Y	low debt	1	123	47	penalty customer	6.2000	71.0000
P0034	P4017	196	0	Other	Y	low debt	1	108	88	no penalty	1.0000	61.0000

## Import Customer Info Table

- Import the table
- Update gender and age based on customer id

```
update customer_det set gender = "Female" where `customer ID` = "IP43006";
update customer_det set gender = "Female" where `customer ID` = "IP43016";
update customer_det set gender = "Male" where `customer ID` = "IP43018";
update customer_det set gender = "Male" where `customer ID` = "IP43036";
update customer_det set gender = "Female" where `customer ID` = "IP43508";
update customer_det set gender = "Female" where `customer ID` = "IP43577";
update customer_det set gender = "Female" where `customer ID` = "IP43589";
update customer_det set gender = "Female" where `customer ID` = "IP43593";
update customer_det set age = 45 where `customer ID` = "IP43007";
update customer_det set age = 32 where `customer ID` = "IP43009";
```



## Output for the above Query

CustomerID	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

## Import country state and region Table

```
select * from country_state;  
select * from region_info;
```

	Customer_id	Load Id	Customer_name	Region_id	Postal_Code	Segment	State
▶	IP43001	LP001002	Claire Gute	13.2	42420	Consumer	Kentucky
	IP43002	LP001003	Darrin Van Huff	13.2	90036	Corporate	California
	IP43003	LP001005	Sean O'Donnell	13.2	33311	Consumer	Florida
	IP43004	LP001006	Brosina Hoffman	13.2	90032	Consumer	California
	IP43005	LP001008	Andrew Allen	13.2	28027	Consumer	North Carolina
	IP43006	LP001011	Irene Maddox	13.2	98103	Consumer	Washington
	IP43007	LP001013	Harold Pavilan	13.3	76106	Home Office	Texas
	IP43008	LP001014	Pete Kriz	13.3	53711	Consumer	Wisconsin
	IP43009	LP001018	Alejandro Grove	13.2	84084	Consumer	Utah
	IP43010	LP001020	Zuschuss Donatelli	13.2	94109	Consumer	California
	IP43011	LP001024	Ken Black	13.3	68025	Corporate	Nebraska
	IP43012	LP001027	Sandra Flanagan	13.4	19140	Consumer	Pennsylvania
	IP43013	LP001028	Emily Burns	13.2	84057	Consumer	Utah
	IP43014	LP001029	Eric Hoffmann	13.2	90049	Consumer	California
	IP43015	LP001030	Tracy Blumstein	13.4	19140	Consumer	Pennsylvania
	IP43016	LP001032	Matt Abelman	13.3	77095	Home Office	Texas
	IP43017	LP001034	Gene Hale	13.3	75080	Corporate	Texas
	IP43018	LP001036	Steve Nguyen	13.3	77041	Home Office	Texas
	IP43019	LP001038	Linda Cazamas	13.3	60540	Corporate	Illinois

	Region	Region_Id
▶	South	13.1
	West	13.2
	North	13.3
	East	13.4

## Join all the 5 tables without repeating the fields

```
create table output_1 as select a.*,  
c.loan_amount,c.cibil_score,c.cibilscore_status,c.monthly_interest,c.annual_interest,  
d.customer_name, d.gender,d.age,d.married,d.education,d.self_employed,d.region_id,r.postal_code,r.segment,r.state  
from customer_income a  
inner join customer_interest_analysis c on a.loan_id = c.loan_id  
inner join customer_det d on c.loan_id = d.loan_id  
inner join country_state r on r.customer_id=d.'customer ID' ;  
select * from output_1;
```

loan_ID	Customer ID	Applicantname	Capplicantname	Property_Area	Loan_Status	grade	monthly_interest_percentage	loanamount	cibiscor	cibiscor_status	monthly_intrest	annual_intrest	cu
UP01001	PH0002	4993	1008	Rural	N	low-dea	5	128	939	high cib score	3,9400	46,0800	De
UP01005	PH0003	3000	0	Urban	Y	low-dea	5	66	886	penalty customers	1,9800	23,7600	Se
UP01006	PH0004	2503	2258	Urban	Y	low-dea	5	120	851	no penalty	3,6000	43,2000	Br
UP01008	PH0005	6000	0	Urban	Y	middle-dea customer	7	140	420	penalty customers	4,2000	50,7600	An
UP01011	PH0006	5407	4036	Urban	Y	middle-dea customer	7	267	173	penalty customers	8,0100	96,1200	Br
UP01012	PH0007	2103	1316	Urban	Y	low-dea	5	95	690	penalty customers	2,8500	34,2000	Na
UP01014	PH0008	3036	2504	Semurban	N	low-dea	7	138	471	penalty customers	4,7400	56,8800	Pe
UP01018	PH0009	4008	1326	Urban	Y	low-dea	5	168	863	no penalty	5,0400	60,4800	Al
UP01020	PH0010	12841	10968	Semurban	N	grade B	7	148	738	penalty customers	10,4700	125,6400	Zu
UP01024	PH0011	3380	780	Urban	Y	low-dea	5	70	140	penalty customers	2,1000	25,2000	Ka
UP01027	PH0012	2360	1840	Urban	Y	low-dea	5	109	384	penalty customers	3,2700	39,2400	Se
UP01028	PH0013	3673	8106	Urban	Y	low-dea	5	200	938	high cib score	6,0000	72,0000	Br
UP01029	PH0014	1853	1840	Rural	N	low-dea	5	114	485	penalty customers	3,4200	41,0400	Br
UP01030	PH0015	1289	1086	Urban	Y	low-dea	5	17	384	penalty customers	5,1000	6,1200	Tr
UP01032	PH0016	4993	0	Urban	Y	low-dea	5	125	477	penalty customers	3,7500	45,0000	Na

## Find the mismatch details using joins

```
create table output_2 as select c.*,
s.postal_code,s.segment,s.state,r.region from customer_det c right join country_state s on c.`customer ID` = s.customer_id
right join region_info r on r.region_id = s.region_id where c.`customer ID` is null and s.segment is null ;
select * from output_2;
```

[illegible]

## Filter high cibil score

```
create table output_3 as select a.*,  
c.loan_amount,c.cibil_score,c.cibilscore_status,c.monthly_interest,c.annual_interest,  
d.customer_name, d.gender,d.age,d.married,d.education,d.self_employed,d.region_id,r.postal_code,r.segment,r.state  
from customer_income a  
inner join customer_interest_analysis c on a.loan_id = c.loan_id  
inner join customer_det d on c.loan_id = d.loan_id  
inner join country_state r on d.'customer ID' = r.customer_id where c.cibil_score_status = ' high cibil score';  
select * from output_3;
```

loan_id	Customer ID	ApplicantIncome	CopapplicantIncome	Property_Area	Loan_Status	grade	monthly_intrest_percentage	loanamount	cibilscore	cibilscore_status	monthly_intrest	annual_intrest
LP001003	SP43002	4082	1509	Rural	N	low-class	3	128	920	high cibil score	3.8400	46.0800
LP001038	SP43013	3672	8106	Urban	Y	low-class	5	209	928	high cibil score	6.6900	72.0000
LP001046	SP43022	5955	5625	Urban	Y	middle class customer	7	315	993	high cibil score	5.4900	113.4000
LP001068	SP43027	2799	2253	Semurban	Y	low-class	3	122	999	high cibil score	3.8400	40.3200
LP001091	SP43031	4186	3186	Urban	N	low-class	5	281	972	high cibil score	6.5700	71.3600
LP001199	SP43066	3387	2659	Urban	Y	low-class	5	144	946	high cibil score	4.5200	51.8400
LP001232	SP43083	4186	0	Semurban	N	low-class	7	138	924	high cibil score	3.4800	41.7600
LP001264	SP43086	3155	2186	Semurban	Y	low-class	7	136	951	high cibil score	3.9000	46.8000
LP001276	SP43093	3686	0	Urban	Y	low-class	6	90	933	high cibil score	3.5000	38.0000
LP001280	SP43097	3332	2000	Semurban	Y	low-class	7	99	985	high cibil score	2.9700	30.6400
LP001333	SP43098	2977	957	Semurban	Y	low-class	7	50	994	high cibil score	1.5000	16.0000
LP001604	SP43117	3167	2283	Semurban	Y	low-class	7	154	919	high cibil score	4.6200	50.4000
LP001422	SP43120	10408	0	Urban	Y	grade B	7	239	997	high cibil score	7.7700	93.2400
LP001451	SP43126	10513	3850	Urban	N	grade B	7	180	929	high cibil score	4.8000	57.6000
LP001482	SP43134	3459	0	Semurban	Y	low-class	7	25	926	high cibil score	0.7500	8.0000
LP001514	SP43146	2152	4496	Semurban	Y	low-class	7	100	921	high cibil score	1.0000	36.0000

## Filter home office and corporate

```
create table output_4 as select a.*,  
c.loan_amount,c.cibil_score,c.cibilscore_status,c.monthly_interest,c.annual_interest,  
d.customer_name, d.gender,d.age,d.married,d.education,d.self_employed,d.region_id,r.postal_code,r.segment,r.state  
from customer_income a  
inner join customer_interest_analysis c on a.loan_id = c.loan_id  
inner join customer_det d on c.loan_id = d.loan_id  
inner join country_state r on d.`customer ID`= r.customer_id where segment in ('home-office', 'corporate');  
select * from output_4;
```

loanamount	ciblscore	ciblscore_status	monthly_intrest	annual_intrest	customer_name	gender	age	married	education	self_employed	region_id	postal_code	segment	state
128	920	high cibil score	3.84000	46.08000	Dennis van Huff	Male	66	Yes	Graduate	No	13.2	90036	Corporate	California
91	690	penalty customers	4.75000	57.00000	Harold Pavlan	Male	45	Yes	Not Graduate	No	13.3	76106	Home Office	Texas
76	943	penalty customers	3.50000	42.00000	Kim Black	Male	48	Yes	Graduate	No	13.3	68025	Corporate	Nebraska
125	477	penalty customers	6.25000	75.00000	Matt Abelman	Female	51	No	Graduate	No	13.3	77095	Home Office	Texas
100	888	no penalty	5.00000	60.00000	Gene Hale	Male	20	No	Not Graduate	No	13.3	75080	Corporate	Texas
76	387	penalty customers	3.80000	45.60000	Steve Nguyen	male	27	No	Graduate	No	13.3	77041	Home Office	Texas
133	371	penalty customers	3.99000	47.88000	Linda Cazamas	Male	64	Yes	Not Graduate	No	13.3	60540	Corporate	Illinois
115	537	penalty customers	5.75000	69.00000	Ruben Auzman	Male	66	Yes	Graduate	Yes	13.2	90046	Corporate	California
104	514	penalty customers	7.28000	87.36000	Erin Smith	Male	40	Yes	Not Graduate	No	13.2	32905	Corporate	Florida
115	963	high cibil score	22.05000	264.60000	Odella Nelson	Male	23	Yes	Graduate	No	13.3	95122	Corporate	Minnesota

## Store all the outputs as procedure

```
delimiter //  
create procedure projectoutput ()  
begin  
select * from output_1;  
select * from output_2;  
select * from output_3;  
select * from output_4;  
end //  
delimiter ;  
  
call projectoutput();
```



**THANK YOU**

**–Affan Ahmed P**