LOAN MANAGEMENT SYSTEM PROJECT BY USING SQL

SUBTITLE : A COMPREHENSIVE APPROACH TO LOAN AND CUSTOMER ANALYSIS

PRESENTED BY :
R.P.RAJESH KANNAN

ABSTRACT

*This project titled **Loan Management System** aims to streamline loan and customer data management using **SQL**. It integrates multiple datasets such as customer income status, loan status, customer information, country-state mapping, and regional data. Core functionalities include classifying customers based on income, calculating monthly and annual interest rates, and automating updates using triggers. Additionally, the project identifies mismatches and filters high-priority data, ensuring accuracy and efficiency. By employing advanced SQL techniques like **joins, triggers, and stored procedures**, the system delivers a comprehensive solution for financial data management, providing actionable insights for improved decision-making.*

Dataset Overview

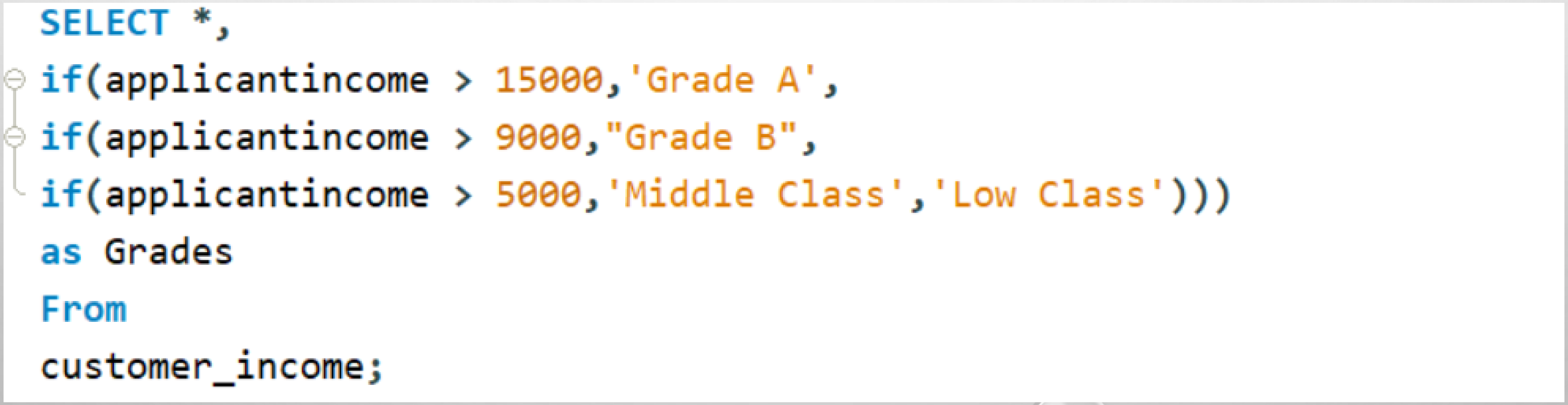
DATASETS USED:

1.Customer Income Status
2.Loan Status
3.Customer Info
4.Country State
5.Region Info

Customer Income Status

Task: Import table from Sheet 1
Steps:
Imported the table "customer income status."
Set criteria based on applicant income:

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



```
SELECT *,
if(applicantincome > 15000,'Grade A',
if(applicantincome > 9000,"Grade B",
if(applicantincome > 5000,'Middle Class','Low Class')))
as Grades
From
customer_income;
```

Monthly Interest Percentage

Criteria:

Applicant income < 5000 and location-based:

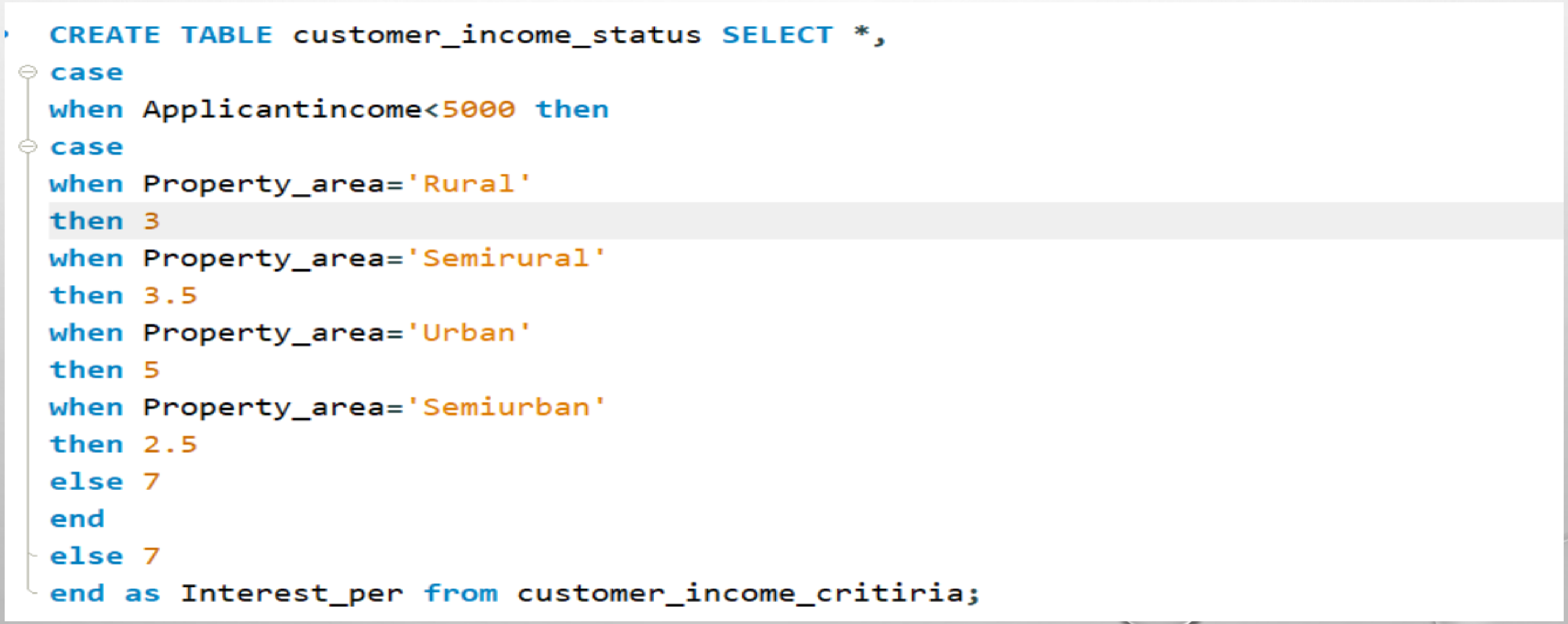
Rural: 3%

Semi-Rural: 3.5%

Urban: 5%

Semi-Urban: 2.5%

Otherwise: 7%

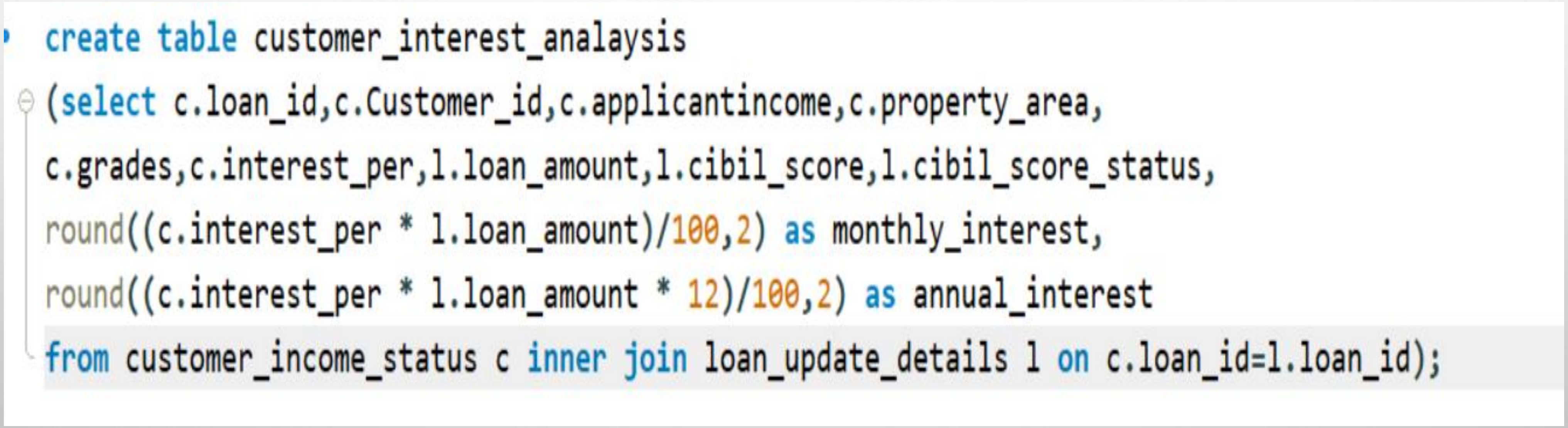


```
CREATE TABLE customer_income_status SELECT *,
case
when Applicantincome<5000 then
case
when Property_area='Rural'
then 3
when Property_area='Semirural'
then 3.5
when Property_area='Urban'
then 5
when Property_area='Semiurban'
then 2.5
else 7
end
else 7
end as Interest_per from customer_income_critiria;
```

Customer Interest Analysis

New Fields:

1. Monthly Interest Amount
2. Annual Interest Amount

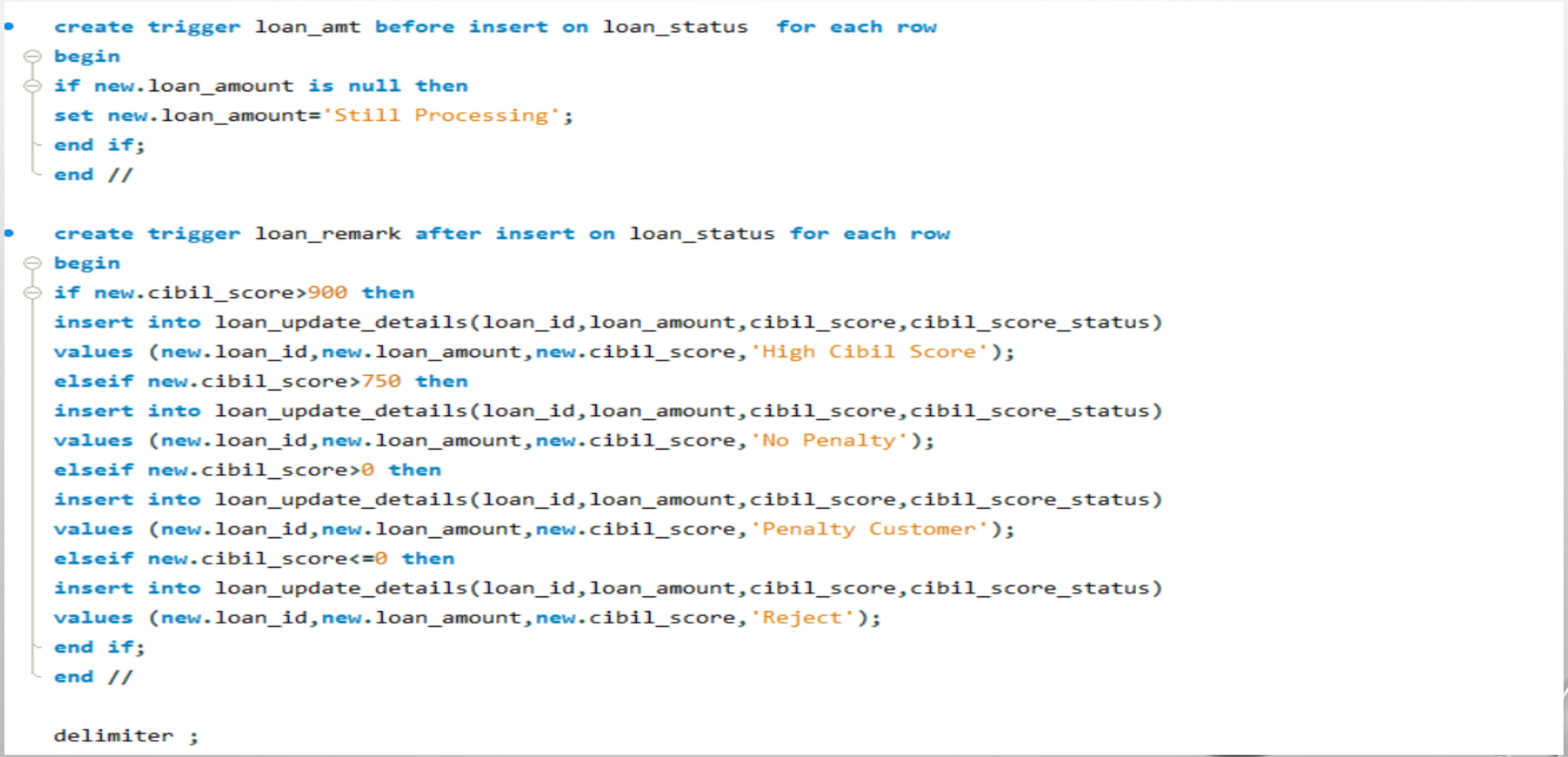


```
create table customer_interest_analaysis
(select c.loan_id,c.Customer_id,c.applicantincome,c.property_area,
c.grades,c.interest_per,l.loan_amount,l.cibil_score,l.cibil_score_status,
round((c.interest_per * l.loan_amount)/100,2) as monthly_interest,
round((c.interest_per * l.loan_amount * 12)/100,2) as annual_interest
from customer_income_status c inner join loan_update_details l on c.loan_id=l.loan_id);
```

Loan Status

Row-Level Trigger for Loan Amount:
Loan amount NULL: **Loan Still Processing**

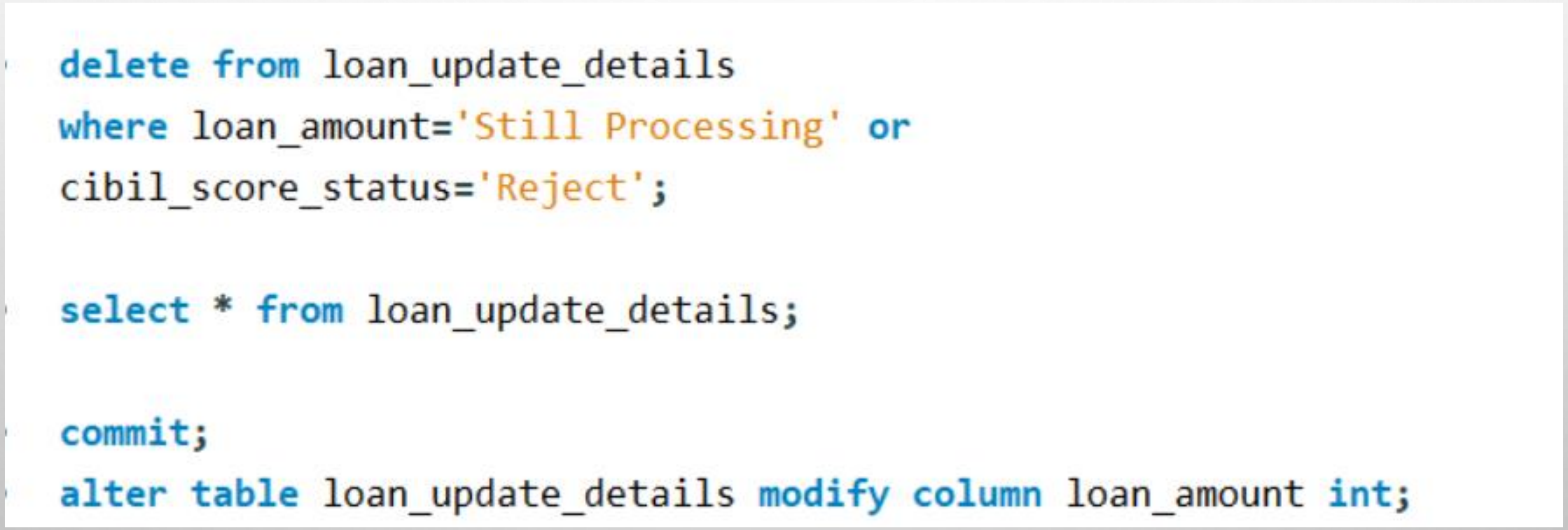
Statement-Level Trigger for CIBIL Score:
CIBIL Score > 900: **High CIBIL Score**
CIBIL Score > 750: **No Penalty**
CIBIL Score > 0: **Penalty Customers**
CIBIL Score <= 0: **Rejected Customers**



```
create trigger loan_amt before insert on loan_status  for each row
begin
if new.loan_amount is null then
set new.loan_amount='Still Processing';
end if;
end //

create trigger loan_remark after insert on loan_status for each row
begin
if new.cibil_score>900 then
insert into loan_update_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 loan_update_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 loan_update_details(loan_id,loan_amount,cibil_score,cibil_score_status)
values (new.loan_id,new.loan_amount,new.cibil_score,'Penalty Customer');
elseif new.cibil_score<=0 then
insert into loan_update_details(loan_id,loan_amount,cibil_score,cibil_score_status)
values (new.loan_id,new.loan_amount,new.cibil_score,'Reject');
end if;
end //

delimiter ;
```

Data Cleaning and Updates

Steps:
Deleted "Rejected Customers" and "Loan Still Processing" customers.
Updated loan amounts to integers.



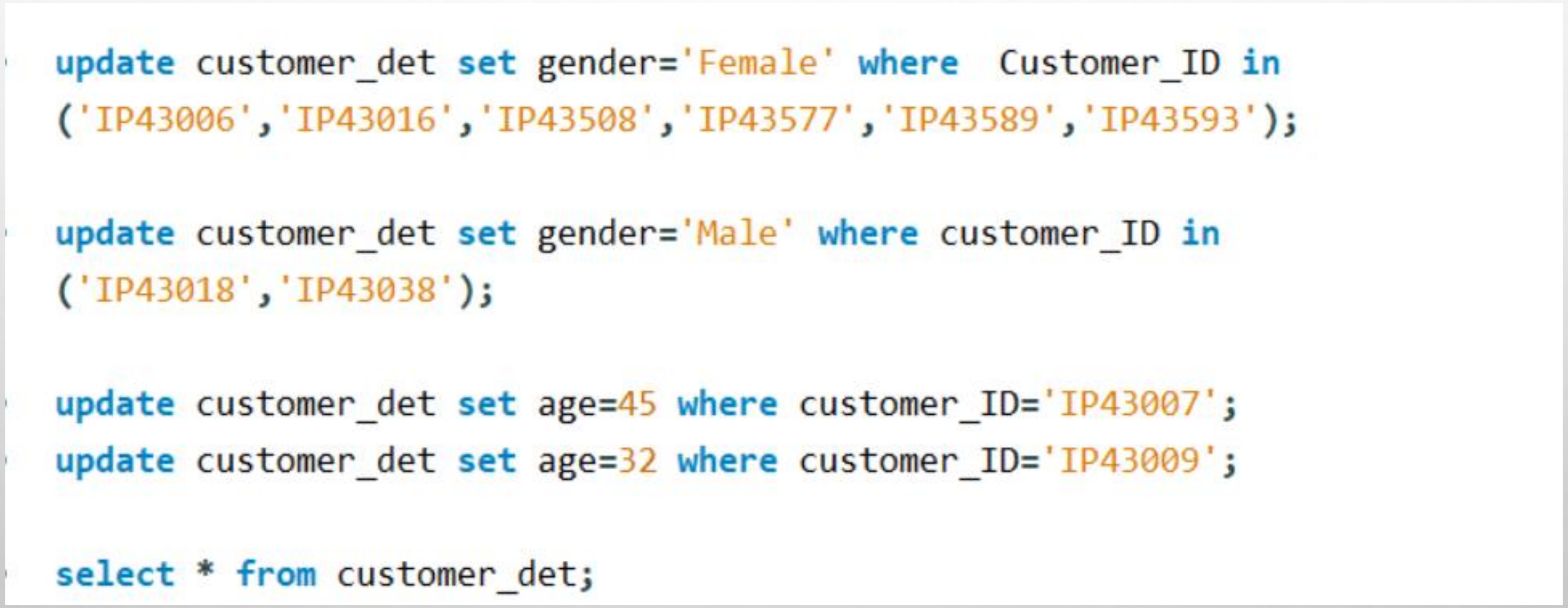
```
delete from loan_update_details
where loan_amount='Still Processing' or
cibil_score_status='Reject';

select * from loan_update_details;

commit;
alter table loan_update_details modify column loan_amount int;
```

Customer Info

Steps:

Imported the table.

Updated gender and age based on customer ID.



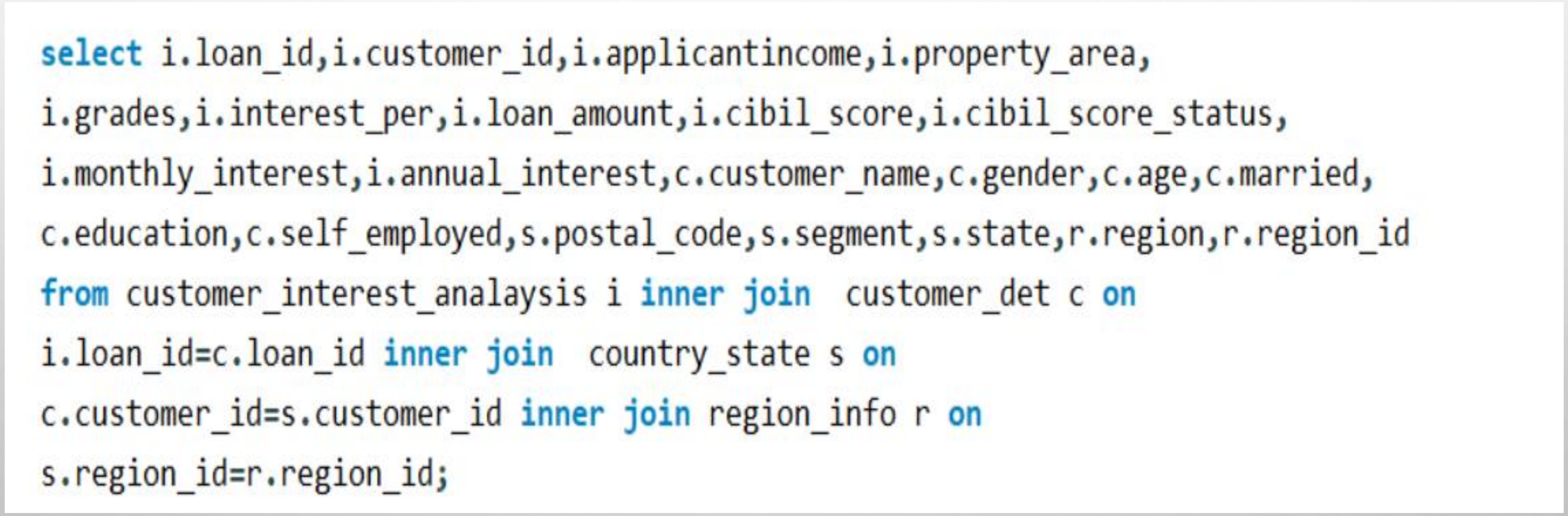
```
update customer_det set gender='Female' where  Customer_ID in
('IP43006','IP43016','IP43508','IP43577','IP43589','IP43593');

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';

select * from customer_det;
```

Country State and Region

Steps:

Imported tables "Country State" and "Region Info."

Joined all 5 tables without repeating fields.

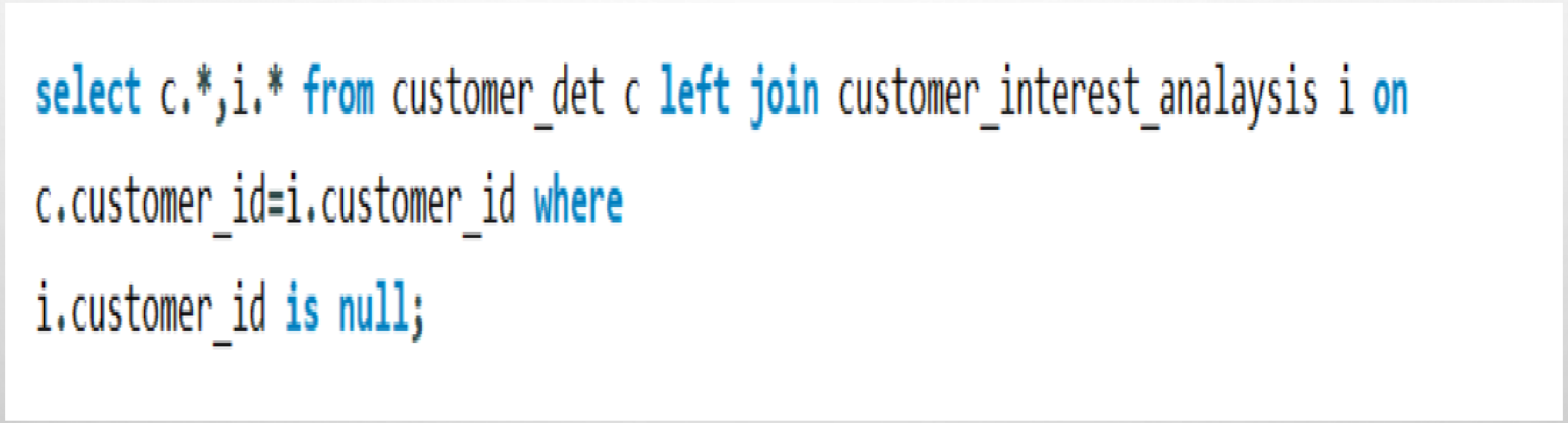


```
select i.loan_id,i.customer_id,i.applicantincome,i.property_area,
i.grades,i.interest_per,i.loan_amount,i.cibil_score,i.cibil_score_status,
i.monthly_interest,i.annual_interest,c.customer_name,c.gender,c.age,c.married,
c.education,c.self_employed,s.postal_code,s.segment,s.state,r.region,r.region_id
from customer_interest_analaysis i inner join  customer_det c on
i.loan_id=c.loan_id inner join  country_state s on
c.customer_id=s.customer_id inner join region_info r on
s.region_id=r.region_id;
```

Mismatch Details

Steps:

Found mismatched details using joins.

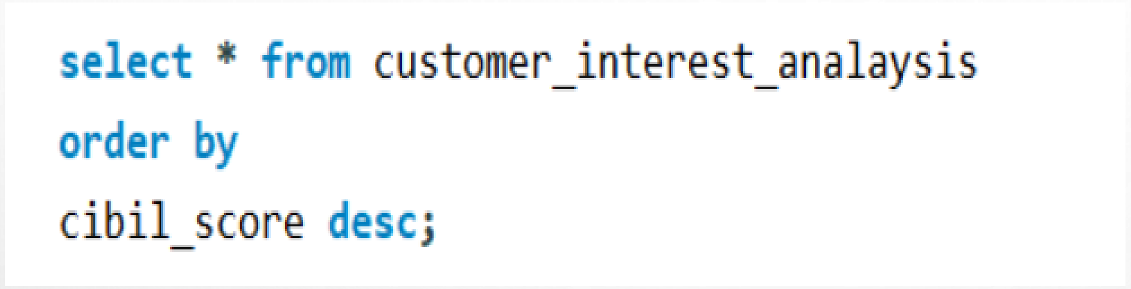


```
select c.*,i.* from customer_det c left join customer_interest_analaysis i on
c.customer_id=i.customer_id where
i.customer_id is null;
```

Filtered Information

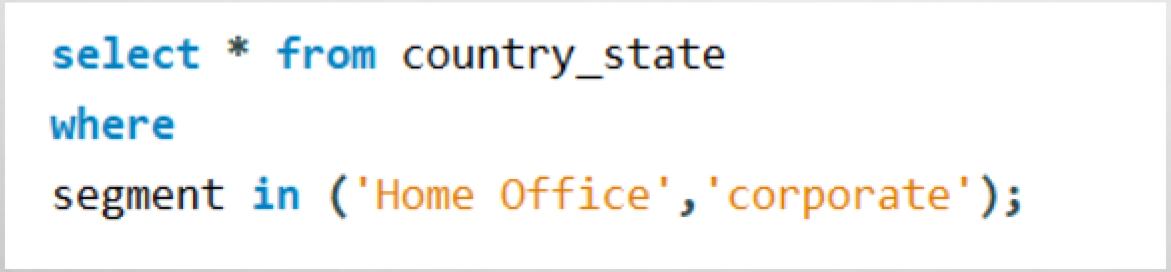
Filters Applied:

1. High CIBIL Score (Query for Output 3) :



```
select * from customer_interest_analaysis
order by
cibil_score desc;
```

2. Home, Office, and Corporate (Query for Output 4) :

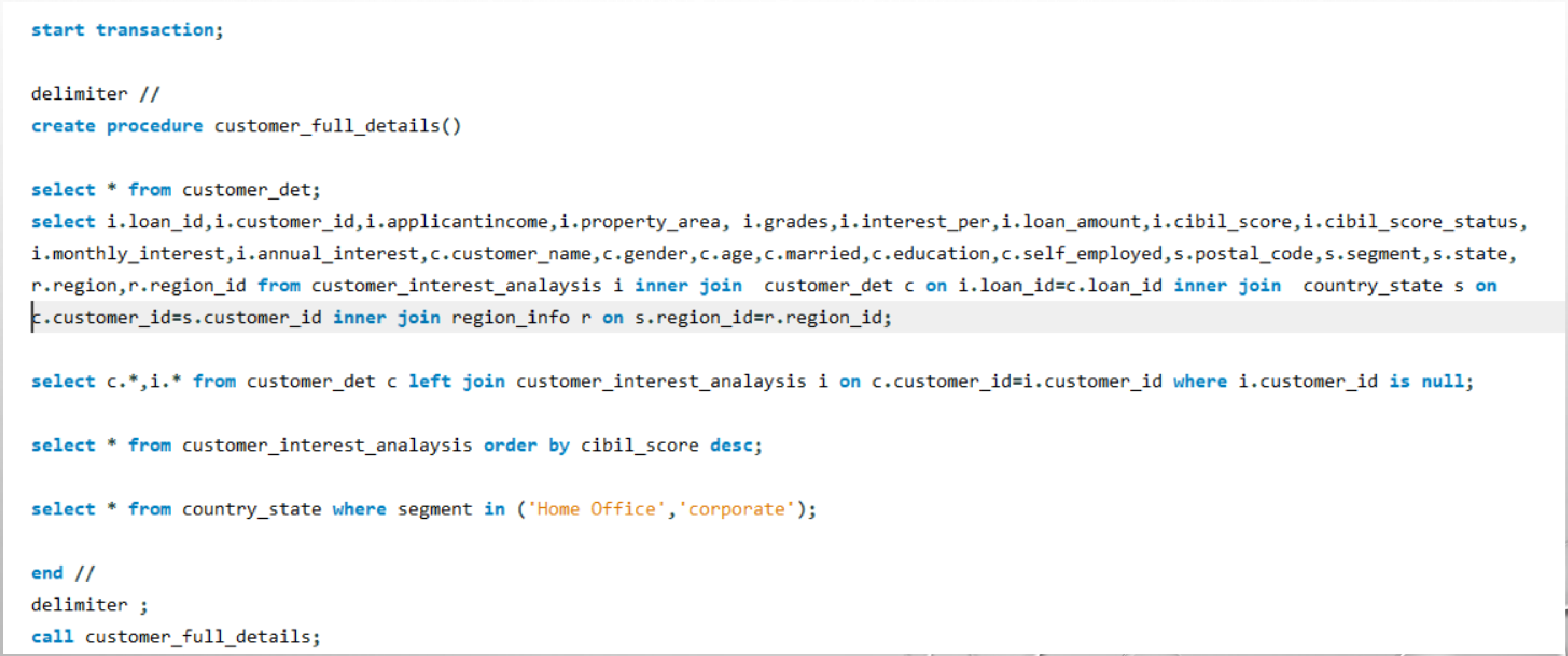


```
select * from country_state
where
segment in ('Home Office','corporate');
```

Stored Procedures

Steps:

Stored all outputs as procedures.



```
start transaction;

delimiter //
create procedure customer_full_details()

select * from customer_det;
select i.loan_id,i.customer_id,i.applicantincome,i.property_area, i.grades,i.interest_per,i.loan_amount,i.cibil_score,i.cibil_score_status,
i.monthly_interest,i.annual_interest,c.customer_name,c.gender,c.age,c.married,c.education,c.self_employed,s.postal_code,s.segment,s.state,
r.region,r.region_id from customer_interest_analaysis i inner join  customer_det c on i.loan_id=c.loan_id inner join  country_state s on
c.customer_id=s.customer_id inner join region_info r on s.region_id=r.region_id;

select c.*,i.* from customer_det c left join customer_interest_analaysis i on c.customer_id=i.customer_id where i.customer_id is null;

select * from customer_interest_analaysis order by cibil_score desc;

select * from country_state where segment in ('Home Office','corporate');

end //
delimiter ;
call customer_full_details;
```

CONCLUSION

The **Loan Management System** effectively addresses complexities in managing large-scale loan data by automating interest calculations, enabling dynamic updates, and eliminating inconsistencies. The integration of datasets ensures seamless functionality, while the use of **SQL procedures and triggers** enhances the system's adaptability. Key features like CIBIL score analysis and customer categorization add value to the decision-making process. Overall, the project showcases the efficiency of **SQL** in handling intricate financial data, emphasizing its role in delivering robust, scalable, and accurate management systems.

THANK YOU…!