**Scenario 1:Handle exceptions during fund transfers between accounts.**

create or replace procedure SafeTransferFunds(

amt in Customers.Balance%type,

cid1 in Customers.CustomerID%type,

cid2 in Customers.CustomerID%type

)

is

camt1 Customers.Balance%type;

begin

savepoint starting\_transfer;

select Balance into camt1 from Customers where CustomerID = cid1;

if camt1 < amt then

raise\_application\_error(-20001,'Insufficient Balance!!!');

else

update Customers set Balance = Balance + amt, LastModified = SYSDATE where CustomerID = cid2;

update Customers set Balance = Balance - amt, LastModified = SYSDATE where CustomerID = cid1;

update Accounts set Balance = Balance + amt, LastModified = SYSDATE where CustomerID = cid2;

update Accounts set Balance = Balance - amt, LastModified = SYSDATE where CustomerID = cid1;

commit;

end if;

exception

when OTHERS then

rollback to starting\_transfer;

raise\_application\_error(-20001,'Error: ' || SQLERRM);

end;

/

**Scenario 2: Manage errors when updating employee salaries.**

create or replace procedure UpdateSalary(

bonus in Number,

empid in Employees.EmployeeID%type

)

is

begin

update Employees set Salary = Salary + (Salary \* (bonus/100))

where EmployeeID = empid;

if SQL%ROWCOUNT = 0 then

raise NO\_DATA\_FOUND;

end if;

exception

when NO\_DATA\_FOUND then

raise\_application\_error(-20001,'Employee ID Not Found!!!');

end;

/  
  
**Scenario 3: Ensure data integrity when adding a new customer.**

create or replace procedure AddNewCustomer(

cid in Customers.CustomerID%type,

cname in Customers.Name%type,

birth in Customers.DOB%type,

bal in Customers.Balance%type

)

is

begin

insert into Customers(CustomerID, Name, DOB, Balance, LastModified)

values(cid, cname, birth, bal, SYSDATE);

exception

when DUP\_VAL\_ON\_INDEX then

raise\_application\_error(-20001,'Customer ID Already Present!!!');

end;

/