SIMPLE BANKING APPLICATION

Efficient SQL Solutions for Simple Banking Applications

INTRODUCTION

In the digital age, efficient financial management systems are essential. This case study explores a Simple Banking Application, highlighting the use of SQL for managing banking data. It covers the design and implementation of a basic banking system, focusing on core functionalities like account management, transactions, and balance tracking. We will examine the database schema, essential SQL queries, and practices for maintaining data integrity and security. This case study aims to demonstrate how SQL can effectively support the development of practical and reliable banking applications

DATA DEFINITION LANGUAGE(CREATE)

- Write SQL statements to create all the tables with the specified columns and foreign key references.

```sql
Create table Customers(
CustomerID int primary key,
FirstName varchar(20),
LastName varchar(20),
Email varchar (50),
Phone varchar(50),
AccountCreationDate date);

desc  customers;
```

1:11

Filter Rows: Search

Field	Type	Null	Key	Default
CustomerID	int	NO	PRI	NULL
FirstName	varchar(20)	YES		NULL
LastName	varchar(20)	YES		NULL
Email	varchar(50)	YES		NULL
Phone	varchar(50)	YES		NULL
AccountCreationDate	date	YES		NULL

- Write SQL statements to create all the tables with the specified columns and foreign key references.

```sql
Create table Accounts(
AccountID int primary key,
CustomerID int,
Foreign key (CustomerID) references Customers(CustomerID),
AccountType varchar(50),
Balance decimal);
desc accounts;
```

15:38

Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
AccountID	int	NO	PRI	NULL	
CustomerID	int	YES	MUL	NULL	
AccountType	varchar(50)	YES		NULL	
Balance	decimal(10,0)	YES		NULL	

- Write SQL statements to create all the tables with the specified columns and foreign key references.

```sql
Create table Transactions(
TransactionID int primary key,
AccountID int,
foreign key (AccountID) references Accounts(AccountID),
TransactionDate Date,
Amount decimal(10,5),
TransactionType varchar(50));
desc Transactions;
```

22:62

Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
AccountID	int	YES	MUL	NULL	
Amount	decimal(10,2)	YES		NULL	
TransactionDate	date	YES		NULL	
TransactionID	int	NO	PRI	NULL	
TransactionType	varchar(50)	YES		NULL	

DATA DEFINITION LANGUAGE(CREATE)

- Write SQL statements to create all the tables with the specified columns and foreign key references.

```
Create table Branches(
BranchID int primary key,
BranchName varchar(30),
BranchAddress varchar(50),
BranchPhone varchar(30));

desc branches;
```

1:90

Filter Rows: Search

Field	Type	Null	Key	Default	Ex
BranchID	int	NO	PRI	NULL	
BranchName	varchar(30)	YES		NULL	
BranchAddress	varchar(50)	YES		NULL	
BranchPhone	varchar(30)	YES		NULL	

- Write SQL statements to create all the tables with the specified columns and foreign key references.

```
Create table Loans(
LoanID int primary key,
CustomerID int,
Foreign key(CustomerID) references Customers(CustomerID),
LoanAmount decimal(10,1),
InterestRate Decimal(3,2),
StartDate date,
EndDate date);
```

12:153

Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
LoanID	int	NO	PRI	NULL	
CustomerID	int	YES	MUL	NULL	
LoanAmount	decimal(10,1)	YES		NULL	
InterestRate	decimal(10,0)	YES		NULL	
StartDate	date	YES		NULL	
EndDate	date	YES		NULL	

- Write SQL statements to create all the tables with the specified columns and foreign key references.

```
create table AccountBranches(
AccountID int ,
foreign key (AccountID) references Accounts(AccountID),
BranchID int ,
Foreign key(BranchID) references Branches(BranchID),
AssignmentDate date);
```

22:125

Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
AccountID	int	YES	MUL	NULL	
BranchID	int	YES	MUL	NULL	
AssignmentDate	date	YES		NULL	

DATA MANUPULATION LANGUAGE (INSERT)

- Insert at least 10 records into each table to populate the database with sample data.

```
insert into customers
values(1,'Gautam','Krishnan',"Gautamkrishnan82@gmail.com",'9833652505',"2010-05-19"),
      (2,'Anmol','Dubey',"anmol286@gmail.com",'7678208596',"2009-05-28"),
      (3,'Ananya','Bansode',"ananya12@gmail.com",'9757158690',"2013-07-12"),
      (4,'Usha','Krishnan',"cgusha62@gmail.com",'9892347075',"1989-12-02"),
      (5,'Rahul','Nayak',"raulnayak93@gmail.com",'9234678990',"2003-05-17"),
      (6,'Priyanka','Krishnan',"priyankakrishnan89@gmail.com",'9892357079',"2009-09-17"),
      (7,'Vrashank','Shetty',"svrashank96@gmail.com",'8651233489',"2007-10-10"),
      (8,'Harsh','Mehta',"harshmehta56@gmail.com",'8888901934',"2002-09-17"),
      (9,'Ravindra','Bansode',"ravindrabansode4@gmail.com",'9004500789',"2004-08-23"),
      (10,'Tim','Cook',"timcook90@gmail.com",'8000978945',"2000-02-29"),
      (11,'Rishil','Kritikar',"rishil78@gmail.com",'9833762506',"1999-09-14"),
      (12,'Sagar','Pawar',"sagarupawar12@gmail.com",'7772444489',"2005-07-26"),
      (13,'Rahul','Jaiswal',"Rahuljaiswal89@gmail.com",'7773895601',"2001-09-23"),
      (14,'Fenil' ,'Aiya',"Fenilaiya89@gmail.com",'7774895601',"2005-09-23");
```

80:28

Result Grid Filter Rows: Search Edit: Export/Import:

CustomerID	FirstName	LastName	Email	Phone	AccountCreationD...
1	Gautam	Krishnan	Gautamkrishnan82@gmail.com	9833652505	2010-05-19
2	Anmol	Dubey	anmol286@gmail.com	7678208596	2009-05-28
3	Ananya	Bansode	ananya12@gmail.com	9757158690	2013-07-12
4	Usha	Krishnan	cgusha62@gmail.com	9892347075	1989-12-02
5	Rahul	Nayak	raulnayak93@gmail.com	9234678990	2003-05-17
6	Priyanka	Krishnan	priyankakrishnan89@gmail.com	9892357079	2009-09-17
7	Vrashank	Shetty	svrashank96@gmail.com	8651233489	2007-10-10
8	Harsh	Mehta	harshmehta56@gmail.com	8888901934	2002-09-17
9	Ravindra	Bansode	ravindrabansode4@gmail.com	9004500789	2004-08-23
10	Tim	Cook	timcook90@gmail.com	8000978945	2000-02-29
11	Rishil	Kritikar	rishil78@gmail.com	9833762506	1999-09-14
12	Sagar	Pawar	sagarupawar12@gmail.com	7772444489	2005-07-26
13	Rahul	Jaiswal	Rahuljaiswal89@gmail.com	7773895601	2001-09-23
14	Fenil	Aiya	Fenilaiya89@gmail.com	7774895601	2005-09-23

- Insert at least 10 records into each table to populate the database with sample data.

```
insert into Branches
values(15,'HDFC Fort','Nariman point, near kala ghoda, CSMT, Mumbai','02225632375'),
      (16,'HDFC Gawanpada','Gawanpada, Mulund east, Mumbai','02224632498'),
      (17,'HDFC Talaopali','Talaopali, Khopat, Thane','02225635589'),
      (18,'HDFC Thrissur','Wadkancheri, Thrissur, Kerala','0666732398'),
      (19,'HDFC Satra Plaza','Satra plaza 4th floor, Vashi, Navi Mumbai','02223457890'),
      (20,'HDFC Shalimar','Old Delhi, Red Fort, New Delhi','0444578921'),
      (21,'HDFC Dapoli','Navsheva,Dapoli beach, Ratnagiri','0234567821'),
      (22,'HDFC Gondia','New Horizon lane, Nagpur','9222679890'),
      (23,'HDFC Noida','Smart city, Noida, Greater Noida','9757168790'),
      (24,'HDFC Dolphins Nose','Guntur Palace road,Vizag','8930456789'),
      (25,'HDFC Howrah','High Street Road,Kolkata','9892556065'),
      (26,'HDFC Gandhinagar','Gandhinagar market,Churu,Ahemadabad','0959234567'),
      (27,'HDFC Bhopal','Pheonix MarketCity, Bhopal','9000489012');
```

24:125

Result Grid Filter Rows: Search Edit: Export/Import:

BranchID	BranchName	BranchAddress	BranchPhone
15	HDFC Fort	Nariman point, near kala ghoda, CSMT, Mumbai	02225632375
16	HDFC Gawanpada	Gawanpada, Mulund east, Mumbai	02224632498
17	HDFC Talaopali	Talaopali, Khopat, Thane	02225635589
18	HDFC Thrissur	Wadkancheri, Thrissur, Kerala	0666732398
19	HDFC Satra Plaza	Satra plaza 4th floor, Vashi, Navi Mumbai	02223457890
20	HDFC Shalimar	Old Delhi, Red Fort, New Delhi	0444578921
21	HDFC Dapoli	Navsheva,Dapoli beach, Ratnagiri	0234567821
22	HDFC Gondia	New Horizon lane, Nagpur	9222679890
23	HDFC Noida	Smart city, Noida, Greater Noida	9757168790
24	HDFC Dolphins Nose	Guntur Palace road,Vizag	8930456789
25	HDFC Howrah	High Street Road,Kolkata	9892556065
26	HDFC Gandhinagar	Gandhinagar market,Churu,Ahemadabad	0959234567
27	HDFC Bhopal	Pheonix MarketCity, Bhopal	9000489012

SELECT & WHERE CLAUSE

- Write a query to select all transactions from the Transactions table where the Amount is greater than $500.

```
select * from transactions
where amount>500;
```

19:181

sult Grid Filter Rows: Search Edit:

TransactionID	AccountID	TransactionDate	Amount	TransactionType
20002	123	2016-10-10	3000.10	Withdrawal
20004	129	2021-11-25	900.00	Withdrawal
20006	127	2016-02-13	750.90	Deposit
20007	128	2017-11-06	2000.15	Withdrawal
20010	130	2023-08-19	800.18	Deposit
20013	124	2021-11-12	4000.00	Deposit
20014	124	2021-12-10	515.00	Withdrawal
20015	125	2015-09-17	890.16	Deposit
20016	130	2016-04-01	789.00	Withdrawal
20017	131	2013-12-07	989.00	Deposit
20018	128	2021-03-03	1000.00	Withdrawal
20019	133	2017-09-17	2000.00	Withdrawal
NULL	NULL	NULL	NULL	NULL

- Write a query to select all Accounts where the Balance is between $1000 and $5000 and the AccountType is 'Checking'

```
185
186 select* from accounts
187 where balance between 1000 and 5000;
188
```

00% 37:187

Result Grid Filter Rows: Search Edi

AccountID	CustomerID	AccountType	Balance
121	1	Checking	4000
122	2	Savings	5000
123	3	Savings	1000
126	6	Checking	4500
132	12	Savings	1100
135	8	Checkings	1800
138	9	Savings	1500
139	8	Savings	1500
NULL	NULL	NULL	NULL

USING LIKE OPERATOR AND CASE STATEMENTS

- Write a query to select all Customers whose LastName starts with 'J'.

```
select * from customers
where LastName like "J%";
```

26:191

sult Grid Filter Rows: Search Edit: Export/Import:

CustomerID	FirstName	LastName	Email	Phone	AccountCreationDate
13	Rahul	Jaiswal	Rahuljaiswal89@gmail.com	7773895601	2001-09-23
NULL	NULL	NULL	NULL	NULL	NULL

- Write a query to select AccountID and a new column AccountStatus from the Accounts table. If Balance is greater than $1000, set AccountStatus to 'Active', otherwise 'Inactive'.

```
select AccountID, Balance,
case
    when Balance >1000 then 'Active'
    else 'Inactive'
    end as AccountStatus
  from Accounts;
```

16:201

sult Grid Filter Rows: Search Export:

AccountID	Balance	AccountStat...
121	4000	Active
122	5000	Active
123	1000	Inactive
124	100	Inactive
125	7000	Active
126	4500	Active
127	50	Inactive
128	890	Inactive
129	789	Inactive
130	300	Inactive
131	800	Inactive
132	1100	Active
133	500	Inactive
134	700	Inactive
135	1800	Active
136	7000	Active
137	7500	Active
138	1500	Active
139	1500	Active
140	50000	Active
141	50000	Active

SUBQUERY AND GROUP BY

- Write a query to find all Customers who have a balance in their accounts greater than the average balance of all accounts. Use a subquery to find these CustomerIDs.

```
select CustomerID,Balance
 from Accounts
 where Balance> (select avg(Balance)
                 from Accounts);
```

17:212

esult Grid Filter Rows: Search Export:

CustomerID	Balance
5	7000
7	7000
9	7500
8	50000
8	50000

- Write a query to get the total Balance for each AccountType. Group the results by AccountType.

```
select* from Accounts;

select AccountType,sum(Balance) Total_Balance
from Accounts
Group by AccountType;
```

22:224

Result Grid Filter Rows: Search Export:

AccountType	Total_Balance
Checking	77039
Savings	68990

HAVING CLAUSE AND LIMIT

- Write a query to get the total number of accounts for each Customer, but only include customers who have more than 2 accounts. Use the HAVING clause.

```
select CustomerID,Count(CustomerId)
from Accounts
Group by CustomerID
having Count(customerID)>2;
```

28:228

Result Grid Filter Rows: Search Export:

CustomerID	Count(CustomerI...
7	3
8	5
9	3

- Write a query to select the top 5 customers with the highest LoanAmount.

```
select customerID,LoanAmount
from loans
order by LoanAmount desc
limit 5;
```

1:238

Result Grid Filter Rows: Search

customerID	LoanAmount
7	55000.0
2	50000.0
12	34000.0
1	30000.0
8	25000.0

JOINS

- Write a query to join Transactions with Accounts to get a list of all transactions with AccountID, TransactionDate, and Amount

```
select AccountID, TransactionDate, Amount
from Accounts join Transactions
using(AccountID);
```

18:244

esult Grid Filter Rows: Search Export:

AccountID	TransactionDate	Amount
121	2015-09-21	200.01
123	2016-10-10	3000.10
127	2014-02-05	300.00
129	2021-11-25	900.00
121	2015-10-02	500.00
127	2016-02-13	750.90
128	2017-11-06	2000.15
125	2018-12-09	300.15
126	2022-05-12	400.01
130	2023-08-19	800.18
131	2020-06-10	90.00
132	2019-10-06	100.15
124	2021-11-12	4000.00
124	2021-12-10	515.00
125	2015-09-17	890.16
130	2016-04-01	789.00
131	2013-12-07	989.00
128	2021-03-03	1000.00
133	2017-09-17	2000.00

- Write a query to get a list of all Accounts and any associated Transactions. Include accounts that might not have any transactions.

```
select AccountID,TransactionID,Amount,TransactionType
from Accounts left outer join Transactions
using(AccountID);
```

18:254

sult Grid Filter Rows: Search Export:

AccountID	TransactionID	Amount	TransactionType
121	20001	200.01	Deposit
121	20005	500.00	Withdrawal
122	NULL	NULL	NULL
123	20002	3000.10	Withdrawal
124	20013	4000.00	Deposit
124	20014	515.00	Withdrawal
125	20008	300.15	Deposit
125	20015	890.16	Deposit
126	20009	400.01	Withdrawal
127	20003	300.00	Deposit
127	20006	750.90	Deposit
134	NULL	NULL	NULL
136	NULL	NULL	NULL
128	20007	2000.15	Withdrawal
128	20018	1000.00	Withdrawal
135	NULL	NULL	NULL
139	NULL	NULL	NULL
140	NULL	NULL	NULL
141	NULL	NULL	NULL
129	20004	900.00	Withdrawal
137	NULL	NULL	NULL
138	NULL	NULL	NULL
130	20010	800.18	Deposit
130	20016	789.00	Withdrawal
131	20011	90.00	Deposit
131	20017	989.00	Deposit
132	20012	100.15	Withdrawal

SUBQUERY USING JOIN & JOIN WITH AGGREGATION

- Write a query to get the total number of accounts for each branch. Use an INNER JOIN between AccountBranches and Branches, and group by BranchID.

```
select b.BranchID,b.BranchName, Count(a.BranchID)
from AccountBranches  a join Branches  b
on a.BranchID=b.BranchID
group by a.branchID;
```

21:266

Result Grid Filter Rows: Search Export:

BranchID	BranchName	Count(a.Branchl...
15	HDFC Fort	2
16	HDFC Gawanpada	2
17	HDFC Talaopali	1
18	HDFC Thrissur	2
19	HDFC Satra Plaza	1
20	HDFC Shalimar	3
21	HDFC Dapoli	2
22	HDFC Gondia	2
23	HDFC Noida	1
24	HDFC Dolphins Nose	1
25	HDFC Howrah	2
26	HDFC Gandhinagar	1
27	HDFC Bhopal	1

- Write a query to find all Branches that manage accounts with a total balance of more than $100,000. Use a subquery in the WHERE clause to find these BranchIDs.

```
select branchID
from Branches
where BranchID=(select branchID
                from AccountBranches join Accounts
                using(AccountID)
                group by BranchID
                having sum(balance)>100000);
```

22:283

Result Grid Filter Rows: Search Export:

branchID
20

ADVANCED JOIN

- Write a query to list FirstName, LastName, AccountID, and TransactionDate for all transactions. Use INNER JOIN and LEFT JOIN as necessary to get all required details.

```
select Firstname,Lastname,AccountID, TransactionDate
from Transactions left outer join accounts
using(AccountID)
join customers
using (customerID);
```

20:295

esult Grid Filter Rows: Search Export:

Firstname	Lastname	AccountID	TransactionDate
Gautam	Krishnan	121	2015-09-21
Ananya	Bansode	123	2016-10-10
Vrashank	Shetty	127	2014-02-05
Ravindra	Bansode	129	2021-11-25
Gautam	Krishnan	121	2015-10-02
Vrashank	Shetty	127	2016-02-13
Harsh	Mehta	128	2017-11-06
Rahul	Nayak	125	2018-12-09
Priyanka	Krishnan	126	2022-05-12
Tim	Cook	130	2023-08-19
Rishil	Kritikar	131	2020-06-10
Sagar	Pawar	132	2019-10-06
Usha	Krishnan	124	2021-11-12
Usha	Krishnan	124	2021-12-10
Rahul	Nayak	125	2015-09-17
Tim	Cook	130	2016-04-01
Rishil	Kritikar	131	2013-12-07
Harsh	Mehta	128	2021-03-03
Rahul	Jaiswal	133	2017-09-17

THANKYOU