set2. CREATE

1. CREATE TABLE Customers(
CustomerID INT Primary key Auto Increment,
FirstName VARCHAR(50),
LastName VARCHAR(50),
email VARCHAR(100) UNIQUE,
PhoneNumber VARCHAR(15),
Address VARCHAR(200)
);

2. CREATE TABLE Accounts(
AccountNumber INT primary key,
CustomerID INT Foreignkey,
Accounttype VARCHAR(20),
Balance DECIMAL(10,2),
Datecreated DATE
);

Inserting Data.

1. INSERT INTO Customer(CustomerID, FirstName, LastName, email, phoneNumber, Address)
Values
('John', 'Doe', 'john.doe@email.com', 1234567890, '123Main St, city'),
('Jane', 'Smith', 'janesmith@email.com', 0987654321, '456 Elm St, Townville'),
('Mike', 'Johnson', 'mike.johnson@email.com', 1122334455,
'789 Oak St, Villageville');

2. INSERT INTO Accounts(AccountNumber, CustomerID, Accounttype, Balancetype, Balance, Datecreated)
Values

(1001, 1, 'saving', 5000.00, 2023-01-15),
(1002, 1, 'Checking', 1500.00, 2023-02-20),
(1003, 2, 'Saving', 2000.00, 2023-03-01),
(1004, 3, 'Checking', 2000.00, 2023-03-10)
;

3. Update Data.

5. UPDATE Accounts SET AccountNumber Balance = 5500.00 WHERE AccountNumber = 1001;

6. UPDATE ~~Accounts~~ Customers SET email = 'jane.smith@new domain.com' WHERE customer = 2;

7. UPDATE ~~Customers~~ Accounts SET Balance = Balance * 1.10 WHERE Accounttype = 'saving';

4. Select Queries

8. SELECT (customerID, FirstName, lastName, Balance FROM Customer WHERE Accounttype = 'saving';

9. SELECT * FROM Customers WHERE Balance 73000.00 AND Accountype = 'checking';

10. SElect *~~FROM~~ (customerID, AccountNumber Accountype FROM Customers WHERE Balance 2000.00;

8. List CustomerID, FirstName, LastName, Balance for saving Account Holders.

⇒ SELECT Customers.CustomerID, Customers.FirstName, Customer LastName, Accounts.Balance
FROM Customers
JOIN Accounts ON Customers.CustomerID = Accounts.CustomerID
WHERE Accounts.AccountType = 'saving';

9. SELECT Customers.CustomerID, Customers.FirstName, Customers. lastName, Accounts.Balance
FROM customers
JOIN Accounts ON Customers.CustomerID = Accounts.CustomerID
WHERE Accounts.Balance > 3000 AND Accounts.AccountType = 'checking';

10. List All Accounts with Balance less than 2000.
SELECT CustomerID, AccountNumber, Accounttype, Balance
FROM Accounts
WHERE Balance < 2000;

5. Deleting Data
11. Delete Account with Account Number 1002
⇒ DELETE FROM ACCOUNTS WHERE AccountNumber = 1002;

12. Delete Customers whose phonenumber starts with '123'
⇒ DELETE FROM Customer WHERE AccountNumber LIKE '123%';

13. Delete Accounts Created Before '2023-02-01'
⇒ DELETE FROM Accounts WHERE Datacreated < '2023-02-01'

6. Join queries.

14. Find FirstName, lastName and Accounttype of customer with Balance Greater than ₹2000.

⇒ SELECT customers.FirstName, customers.LastName, Account.Accounttype
FROM Customers
JOIN Accounts ON customers.ID = Accounts.customerID
WHERE Accounts.Balance > 2000;

15. Get total Balance of saving Accounts.

⇒ SELECT Accounttype SUM(Balance) AS TotalBalance
FROM Accounts WHERE Accounttype = 'saving'
GROUP BY Accounttype;

16. Show customer's FirstName, LastName, AccountNumber, and Balance, including customers without Accounts.

⇒ SELECT customers.FirstName, customers.lastName, Accounts.AccountNumber, Account.Balance
FROM ~~Accounts~~ customers
LEFT JOIN Accounts ON customer.customer.ID = Account.cu
~~WHERE~~

7. Constraints and validation.

17. Ensure Balance Cannot Be Negative.

⇒ ALTER TABLE Accounts ADD Constraints check_Balance
CHECK (Balance >= 0);

18. Ensure email is unique and Not NULL
⇒ ALTER TABLE customer ~~ADD~~ MODIFY COLUMN email VARCHAR (100) NOT NULL UNIQUE;

19. Add Foreign key constraints to Accounts:
⇒ ALTER TABLE Accounts Add CONSTRANT fk_customers FOREIGN KEY (CustomerID) REFERENCES customers(customerID);

20. Create check constraints for Accounttype.
⇒ ALTER TABLE Accounts ~~check~~ column ~~Accounttype~~ CHECK(CONSTRANT) ADD CONSTRANT check_account_type CHECK (Accounttype IN ('saving', 'checking'));

8. Complex queries.
21. Find the customer with the highest Balance across All Account types.
⇒ SELECT customers.firstName, customers.lastName, Accounts.AccountNumber, MAX(Accounts.Balance) As highest Balance FROM customers
JOIN Accounts ON customer.customerID = Accounts.Customer
ORDER BY HighestBalance DESC
LIMIT 1;

22. Transfer 1000 from Accounts 1002 to Account 1001
⇒ BEGIN TRANSACTION;
UPDATE Accounts
SET Balance = Balance - 1000
WHERE AccountNumber = 1002;

UPDATE Accounts SET
Balance = Balance + 1000.
WHERE AccountNumber = 1001;
COMMIT;

22. Show total Balance for each customer.
=> SELECT customer.(customerFirstName, customers.lastName
SUM(Accounts.Balance) AS totalBalance
FROM customers
JOIN Accounts ON customers.customerID = Accounts.customerID
GROUP BY customers.customerID;

9. Aggregation.
24. Find the average balance of All Accounts.
=> SELECT ~~*~~ COUNT(*) ~~Accounts AVG(Ba AS totalsavingaccounts~~
FROM Accounts
WHE
=> SELECT AVG(Balance) AS averageBalance FROM Accounts.

25. Count total Number of saving Accounts.
=> SELECT COUNT(*) AS totalsaving FROM Accounts WHERE
AccountType = 'saving',

10. Transactions (withdraw from one Account & deposit into Another)
BEGIN transactions
UPDATE Accounts SET Balance = Balance - 500 WHERE Account
Number = 1002;
UPDATE Accounts SET Balance = Balance + 500 WHERE
AccountNumber = 1003;
COMMIT;