Consider following bank schema and solve following questions using SQL.

Branch-schema = (branch-name, branch-city, assets)

Customer-schema = (customer-name, customer-street, customer-city)

Loan-schema = (loan-number, branch-name, amount)

Borrower-schema = (customer-name, loan-number)

Account-schema = (account-number, branch-name, balance)

Depositor-schema = (customer-name, account-number)

1. Find the names of all branches in the loan relation

SQL
```
SELECT DISTINCT branch-name
FROM Loan;
```
2. Find all loan numbers for loans made at the Perryridge branch with loan amounts greater than 1200.

SQL
```
SELECT loan-number
FROM Loan
WHERE branch-name = 'Perryridge' AND amount > 1200;
```
3. Find the loan number of those loans with loan amounts between 9000 and 10000.

SQL
```
SELECT loan-number
FROM Loan
WHERE amount BETWEEN 9000 AND 10000;
```
4. For all customers who have a loan from the bank, find their names, loan numbers and loan amount.

SQL
```
SELECT customer-name, loan-number, amount
FROM Borrower b
JOIN Loan l ON b.loan-number = l.loan-number;
```
5. Find the customer names, loan numbers, and loan amounts for all loans at the Perryridge branch.

SQL
```
SELECT customer-name, loan-number, amount
FROM Borrower b
JOIN Loan l ON b.loan-number = l.loan-number
WHERE branch-name = 'Perryridge';
```
6. For all customers who have a loan from the bank, find their names, loan numbers, and loan amount.

SQL
```
SELECT customer-name, loan-number, amount
FROM Borrower b
JOIN Loan l ON b.loan-number = l.loan-number;
```

7. Find the names of all customers whose street address includes the substring ‘Main’.

SQL
```sql
SELECT customer-name
FROM Customer
WHERE customer-street LIKE '%Main%';
```

8. List in alphabetic order all customers who have a loan at the Perryridge branch.

SQL
```sql
SELECT DISTINCT customer-name
FROM Borrower b
JOIN Loan l ON b.loan-number = l.loan-number
WHERE branch-name = 'Perryridge'
ORDER BY customer-name;
```

9. List the entire loan relation in descending order of amount.

SQL
```sql
SELECT *
FROM Loan
ORDER BY amount DESC;
```

10. Find all customers having a loan, an account, or both at the bank.

SQL
```sql
SELECT DISTINCT customer-name
FROM (
    SELECT customer-name
    FROM Borrower
    UNION
    SELECT customer-name
    FROM Depositor
) AS combined_customers;
```

11. Find all customers who have both a loan and an account at the bank.

SQL
```sql
SELECT customer-name
FROM Borrower b
JOIN Depositor d ON b.customer-name = d.customer-name;
```

12. Find all customers who have an account but no loan at the bank.

SQL
```sql
SELECT DISTINCT customer-name
FROM Depositor d
WHERE customer-name NOT IN (
    SELECT customer-name
    FROM Borrower
);
```

13. Find the average account balance at the Perryridge branch.

SQL
```sql
SELECT AVG(balance)
FROM Account
WHERE branch-name = 'Perryridge';
```

14. Find the average account balance at each branch.

SQL
```sql
SELECT branch-name, AVG(balance)
FROM Account
GROUP BY branch-name;
```
15. Find the number of depositors for each branch.

SQL
```sql
SELECT branch-name, COUNT(*) AS num_depositors
FROM Depositor
GROUP BY branch-name;
```
16. Find those branches where the average account balance is more than 1200.

SQL
```sql
SELECT branch-name
FROM Account
GROUP BY branch-name
HAVING AVG(balance) > 1200;
```
17. Find the number of tuples in the customer relation.

SQL
```sql
SELECT COUNT(*) AS num_customers
FROM Customer;
```
18. Find the average balance for each customer who lives in Kolhapur and has at least three accounts.

SQL
```sql
SELECT customer-name, AVG(balance)
FROM Depositor d
JOIN Account a ON d.account-number = a.account-number
WHERE customer-city = 'Kolhapur'
GROUP BY customer-name
HAVING COUNT(*) >= 3;
```
19. Find all loan numbers that appear in the loan relation with null values for amount.

SQL
```sql
SELECT loan-number
FROM Loan
WHERE amount IS NULL;
```

20 – 22 (Using Subquery)

23. Customers with loan (not Smith or Jones):

SQL
```sql
SELECT DISTINCT customer-name
FROM Borrower
WHERE customer-name NOT IN ('Smith', 'Jones');
```
24. Branches with assets > some Kolhapur branch:

SQL
```sql
SELECT DISTINCT T.branch-name
FROM Branch T, Branch S
WHERE T.assets > S.assets AND S.branch-city = 'Kolhapur';
```

25. Branches with assets > all Kolhapur branches:

SQL
```
SELECT branch-name
FROM Branch T
WHERE NOT EXISTS (
    SELECT *
    FROM Branch S
    WHERE S.branch-city = 'Kolhapur' AND T.assets <= S.assets
);
```
26. Branch with highest average balance:

SQL
```
SELECT branch-name
FROM Account
GROUP BY branch-name
ORDER BY AVG(balance) DESC
LIMIT 1;
```
27. Customers with at most one account at Perryridge:

SQL
```
SELECT customer-name
FROM Depositor d
WHERE branch-name = 'Perryridge'
GROUP BY customer-name
HAVING COUNT(*) <= 1;
```
28. View: all-customer (customers with account/loan):

SQL
```
CREATE VIEW all_customer AS
SELECT DISTINCT branch-name, customer-name
FROM Borrower
UNION
SELECT DISTINCT branch-name, customer-name
FROM Depositor;
```
29. Perryridge customers from all-customer view:

SQL
```
SELECT customer-name
FROM all_customer
WHERE branch-name = 'Perryridge';
```
30. View: total-loan (sum of loan amounts per branch):

SQL
```
CREATE VIEW total_loan AS
SELECT branch-name, SUM(amount) AS total_amount
FROM Loan
GROUP BY branch-name;
```
31. Maximum total balance across all branches:

SQL
```
SELECT MAX(total_amount)
FROM total_loan;
```
32. Branches with total deposit below average:

SQL

```sql
SELECT branch-name
FROM Account
GROUP BY branch-name
HAVING SUM(balance) < (
    SELECT AVG(SUM(balance))
    FROM Account
    GROUP BY branch-name
);
```

33. Delete accounts in Perryridge:

SQL
```sql
DELETE FROM Account
WHERE branch-name = 'Perryridge';
```

34. Delete loans between 1300 and 1500:

SQL
```sql
DELETE FROM Loan
WHERE amount BETWEEN 1300 AND 1500;
```

35. Delete accounts in Kolhapur branches:

SQL
```sql
DELETE FROM Account
WHERE branch-name IN (
    SELECT branch-name
    FROM Branch
    WHERE branch-city = 'Kolhapur'
);
```

36. Delete account of 'Smith':

SQL
```sql
DELETE FROM Account
WHERE customer-name = 'Smith';
```

37. Delete accounts below average balance:

SQL
```sql
DELETE FROM Account
WHERE balance < (
    SELECT AVG(balance)
    FROM Account
);
```

38. Update all balances by 5%:

SQL
```sql
UPDATE Account
SET balance = balance * 1.05;
```

39. Update balances > 30000 by 5%:

SQL
```sql
UPDATE Account
SET balance = balance * 1.05
WHERE balance > 30000;
```

40. Pay 5% interest on accounts with balance > average:

SQL
```sql
UPDATE Account
SET balance = balance + balance * 0.05
```

```
WHERE balance > (
    SELECT AVG(balance)
    FROM Account
);
```

41. Update accounts based on balance:

SQL

```
UPDATE Account
SET balance = CASE WHEN balance > 10000 THEN balance * 1.06 ELSE
balance * 1.05 END;
```

42. Customers with either account or loan (not both):

SQL

```
SELECT DISTINCT customer-name
FROM (
    SELECT customer-name
    FROM Borrower
    WHERE customer-name NOT IN (
        SELECT customer-name
        FROM Depositor
    )
) AS account_only
UNION
SELECT DISTINCT customer-name
FROM (
    SELECT customer-name
    FROM Depositor
    WHERE customer-name NOT IN (
        SELECT customer-name
        FROM Borrower
    )
) AS loan_only;
```

These queries address all provided