DBMS Assignment Solutions

Question 1: Schema Diagram

Below is the schema diagram created from the given relations:

[Schema Diagram Image Here - See Attached File Earlier]

Question 2: SQL Queries and Relational Algebra

Q1. Find the average loan amount from each branch.
SQL:
SELECT branch_name, AVG(amount) AS avg_loan FROM loan GROUP BY branch_name;

Relational Algebra:
γ branch_name, AVG(amount) (loan)

Q2. Show details of customers whose street name has two consecutive 's'.
SQL:
SELECT * FROM customer WHERE customer_street LIKE '%ss%';

Relational Algebra:
σ customer_street LIKE '%ss%' (customer)

Q3. Find all customers who have a loan, list their names & loan numbers.
SQL:
SELECT c.customer_name, b.loan_number FROM customer c JOIN borrower b ON c.customer_name = b.customer_name;

Relational Algebra:
π customer_name, loan_number (customer ■ borrower)

Q4. List customers in alphabetical order who have a loan in 'Perryridge' branch.
SQL:
SELECT DISTINCT c.customer_name FROM customer c JOIN borrower b ON c.customer_name = b.customer_name JOIN loan l ON b.loan_number = l.loan_number WHERE l.branch_name = 'Perryridge' ORDER BY c.customer_name;

Relational Algebra:
π customer_name (σ branch_name='Perryridge' (customer ■ borrower ■ loan))

Q5. Find all customers having a loan, an account, or both.
SQL:
SELECT customer_name FROM borrower UNION SELECT customer_name FROM depositor;

Relational Algebra:
π customer_name (borrower) ∪ π customer_name (depositor)

Q6. Find customers whose street includes substring 'Main'.
SQL:
SELECT customer_name FROM customer WHERE customer_street LIKE '%Main%';

Relational Algebra:

π customer_name (σ customer_street LIKE '%Main%' (customer))

Q7. Find average loan amount from each branch where avg > 1500.
SQL:

```
SELECT branch_name, AVG(amount) AS avg_loan FROM loan GROUP BY branch_name
HAVING AVG(amount) > 1500;
```

Relational Algebra:
σ AVG(amount) > 1500 (γ branch_name, AVG(amount) (loan))

Q8. Count number of tuples in customer.
SQL:

```
SELECT COUNT(*) AS total_customers FROM customer;
```

Relational Algebra:
COUNT(customer)

Q9. Find average & max account balance at each branch.
SQL:

```
SELECT branch_name, AVG(balance) AS avg_balance, MAX(balance) AS max_balance FROM
account GROUP BY branch_name;
```

Relational Algebra:
γ branch_name, AVG(balance), MAX(balance) (account)

Q10. Names of customers who have loan at Perryridge branch.
SQL:

```
SELECT DISTINCT c.customer_name FROM customer c JOIN borrower b ON c.customer_name =
b.customer_name JOIN loan l ON b.loan_number = l.loan_number WHERE l.branch_name =
'Perryridge';
```

Relational Algebra:
π customer_name (σ branch_name='Perryridge' (customer ■ borrower ■ loan))

Q11. Delete accounts with balance below average.
SQL:

```
DELETE FROM account WHERE balance < (SELECT AVG(balance) FROM account);
```

Q12. Perform RIGHT, LEFT, INNER, NATURAL JOIN on loan & borrower.
INNER JOIN:

```
SELECT * FROM loan l INNER JOIN borrower b ON l.loan_number = b.loan_number;
```

LEFT JOIN:

```
SELECT * FROM loan l LEFT JOIN borrower b ON l.loan_number = b.loan_number;
```

RIGHT JOIN:

```
SELECT * FROM loan l RIGHT JOIN borrower b ON l.loan_number = b.loan_number;
```

NATURAL JOIN:

```
SELECT * FROM loan NATURAL JOIN borrower;
```

Question 3: Demonstrate Outputs

The outputs depend on the dataset in your database. Below is an example format of how to present results after running SQL queries.

branch_name	avg_loan
Perryridge	2000
Downtown	1800

Question 4: Superkeys and Candidate Keys

Relation: Employee(EmpID, Name, Email, Phone, Department)

- Superkeys: {EmpID}, {Email}, {Phone}, and all supersets containing them.
- Candidate Keys: {EmpID}, {Email}, {Phone}.
- Primary Key: EmpID (recommended, as IDs are unique).