1.Table Structure

Table	Key Constraints	Mandatory Fields (NOT NULL)
`Customer`	PK: `cid` (unique customer ID)	`cid`, `cname` (customer name)
`Company`	PK: `cid`; FK: `cid` → `Customer.cid` (ON DELETE CASCADE)	`cid`, `street`, `city`
`Individual`	PK: `cid`; FK: `cid` → `Customer.cid` (ON DELETE CASCADE)	`cid`, `gender`, `age`
`Account`	PK: `aid`; FK: `cid` → `Customer.cid` (ON DELETE CASCADE)	`aid`, `overdraft_limit`, `start_date`, `pin`, `cid`
`Branch`	PK: `branch_number` (unique branch ID)	`branch_number`, `city`, `street`
`Loan`	PK: `loan_number`; FK: `branch_number` → `Branch.branch_number` (ON DELETE CASCADE)	`loan_number`, `loan_type`, `amount`, `branch_number`
`LoanPayment`	Composite PK: `(loan_number, payment_number)`; FK: `loan_number` → `Loan.loan_number`	`loan_number`, `payment_number`, `date`, `amount`
`LoanCustomer`	Composite PK: `(loan_number, cid)`; FKs: `loan_number` → `Loan`, `cid` → `Customer`	`loan_number`, `cid`

2.Basic Constraints

Table	Column	Type	Constraints
`Customer`	`cid`	`VARCHAR(10)`	PRIMARY KEY (unique customer identifier), NOT NULL
`Customer`	`cname`	`VARCHAR(50)`	NOT NULL (customer name cannot be empty)
`Company`	`cid`	`VARCHAR(10)`	PRIMARY KEY, FOREIGN KEY references `Customer.cid`

Table	Column	Type	Constraints
			(ON DELETE CASCADE);
`Company`	`street`	`VARCHAR(100)`	NOT NULL (company street address cannot be empty)
`Company`	`city`	`VARCHAR(50)`	NOT NULL (company city cannot be empty)
`Individual`	`cid`	`VARCHAR(10)`	PRIMARY KEY, FOREIGN KEY references `Customer.cid` (ON DELETE CASCADE);
`Individual`	`gender`	`VARCHAR(10)`	NOT NULL (gender cannot be empty)
`Individual`	`age`	`INTEGER`	NOT NULL, must be a positive integer (enforced by trigger: `age > 0`)
`Account`	`aid`	`VARCHAR(10)`	PRIMARY KEY (unique account identifier)
`Account`	`overdraft_limit`	`DECIMAL(10,2)`	NOT NULL (overdraft limit cannot be empty)
`Account`	`start_date`	`DATE`	NOT NULL (account opening date cannot be empty)
`Account`	`pin`	`VARCHAR(20)`	NOT NULL (account PIN cannot be empty)
`Account`	`cid`	`VARCHAR(10)`	NOT NULL, FOREIGN KEY references `Customer.cid` (ON DELETE CASCADE)
`Branch`	`branch_number`	`VARCHAR(10)`	PRIMARY KEY (unique branch identifier)
`Branch`	`city`	`VARCHAR(50)`	NOT NULL (branch city cannot be empty)
`Branch`	`street`	`VARCHAR(100)`	NOT NULL (branch street address cannot be empty)
`Loan`	`loan_number`	`VARCHAR(10)`	PRIMARY KEY (unique loan identifier)
`Loan`	`loan_type`	`VARCHAR(20)`	NOT NULL (loan type cannot be empty)

Table	Column	Type	Constraints
`Loan`	`amount`	`DECIMAL(10,2)`	NOT NULL (loan amount cannot be empty)
`Loan`	`branch_number`	`VARCHAR(10)`	NOT NULL, FOREIGN KEY references `Branch.branch_number` (ON DELETE CASCADE)
`LoanPayment`	`loan_number`	`VARCHAR(10)`	Part of composite PRIMARY KEY, FOREIGN KEY references `Loan.loan_number` (ON DELETE CASCADE)
`LoanPayment`	`payment_number`	`INTEGER`	Part of composite PRIMARY KEY (payment sequence number)
`LoanPayment`	`date`	`DATE`	NOT NULL (payment date cannot be empty)
`LoanPayment`	`amount`	`DECIMAL(10,2)`	NOT NULL (payment amount cannot be empty)
`LoanCustomer`	`loan_number`	`VARCHAR(10)`	Part of composite PRIMARY KEY, FOREIGN KEY references `Loan.loan_number` (ON DELETE CASCADE)
`LoanCustomer`	`cid`	`VARCHAR(10)`	Part of composite PRIMARY KEY, FOREIGN KEY references `Customer.cid` (ON DELETE CASCADE)
`LoanCustomer`	`role`	`VARCHAR(20)`	Optional (can be NULL) (customer’ s role in the loan)

2.1 Type-Based Constraints

domains (e.g., integer for dno), key constraints, foreign key constraints, and participation constraints (using NOT NULL) achieved

2.2 Trigger-Based Constraints

Positive age check for individuals and mutual exclusion of customer types

3. Key Technical Issue: SQLite Version Compatibility

In SQLite versions **below 3.38.0**, dynamic string concatenation (e.g., `||`, `printf()`) is not supported in `RAISE()`—only fixed string literals work.

4. Summary

Core Goals Achieved: Entity/referential integrity is ensured; critical business rules (positive age, customer type exclusion) are enforced.

Key Issue: Low SQLite versions limit dynamic error messages, but workarounds maintain functionality.

Next Step: Adjust error messages based on the target SQLite version to improve usability.