Banking Database Management

Course: Database Management 1
Instructor: Prof. Dr. Melike ŞahDirekoğlu

Group Members:
- Farhaneh Aryanejad - 22101015
- Amanuel Galema - 22316708
- Nathan Ngoyi - 22116175
- Joshua Omotosho - 22210133

Introduction

Banking database management is essential for efficient handling of customer and transaction data in the system.

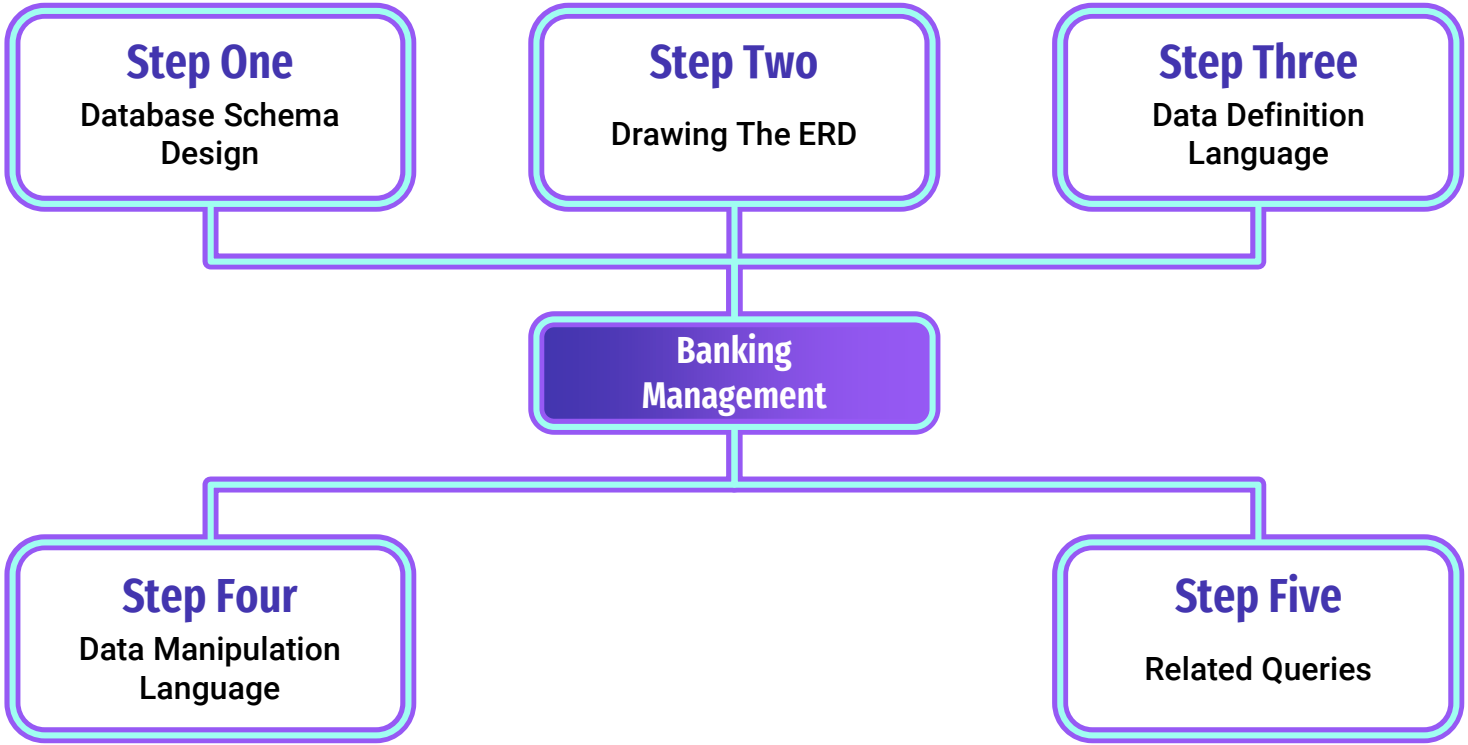
We have designed the database to efficiently manage essential operations and ensure data integrity.

It connects core entities such as accounts, customers, employees, branches, loans, and transactions.

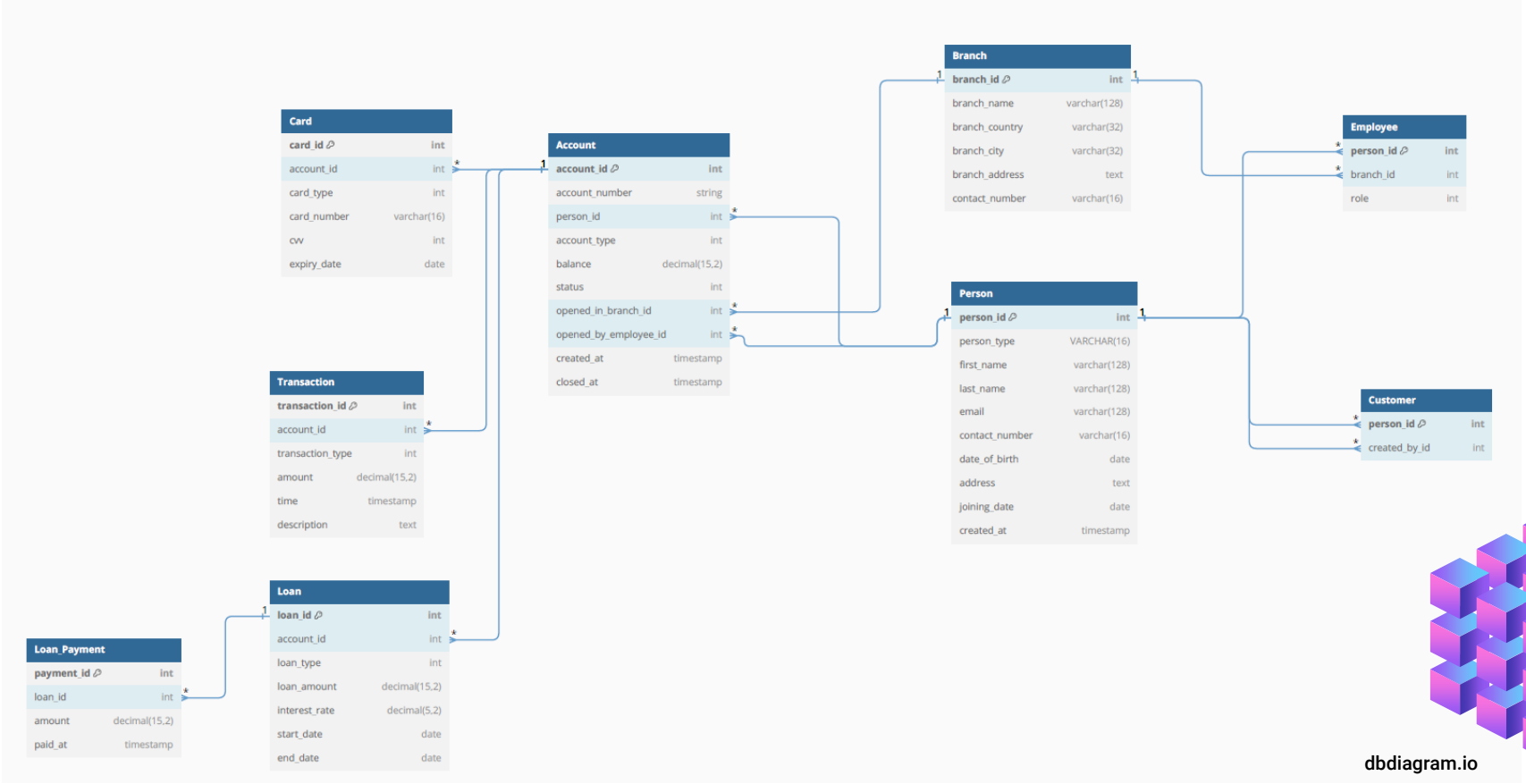
The schema supports critical functionalities like account management, loan processing, transaction tracking, and card services.

We have organized the structure to ensure seamless collaboration between various banking operations, enhancing overall efficiency and customer satisfaction.

The Database Design and Implementation Steps



Step One : Database Schema Design



dbdiagram.io

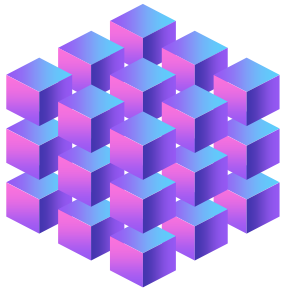
Step Two : ERD (Entity Relationship Diagram)

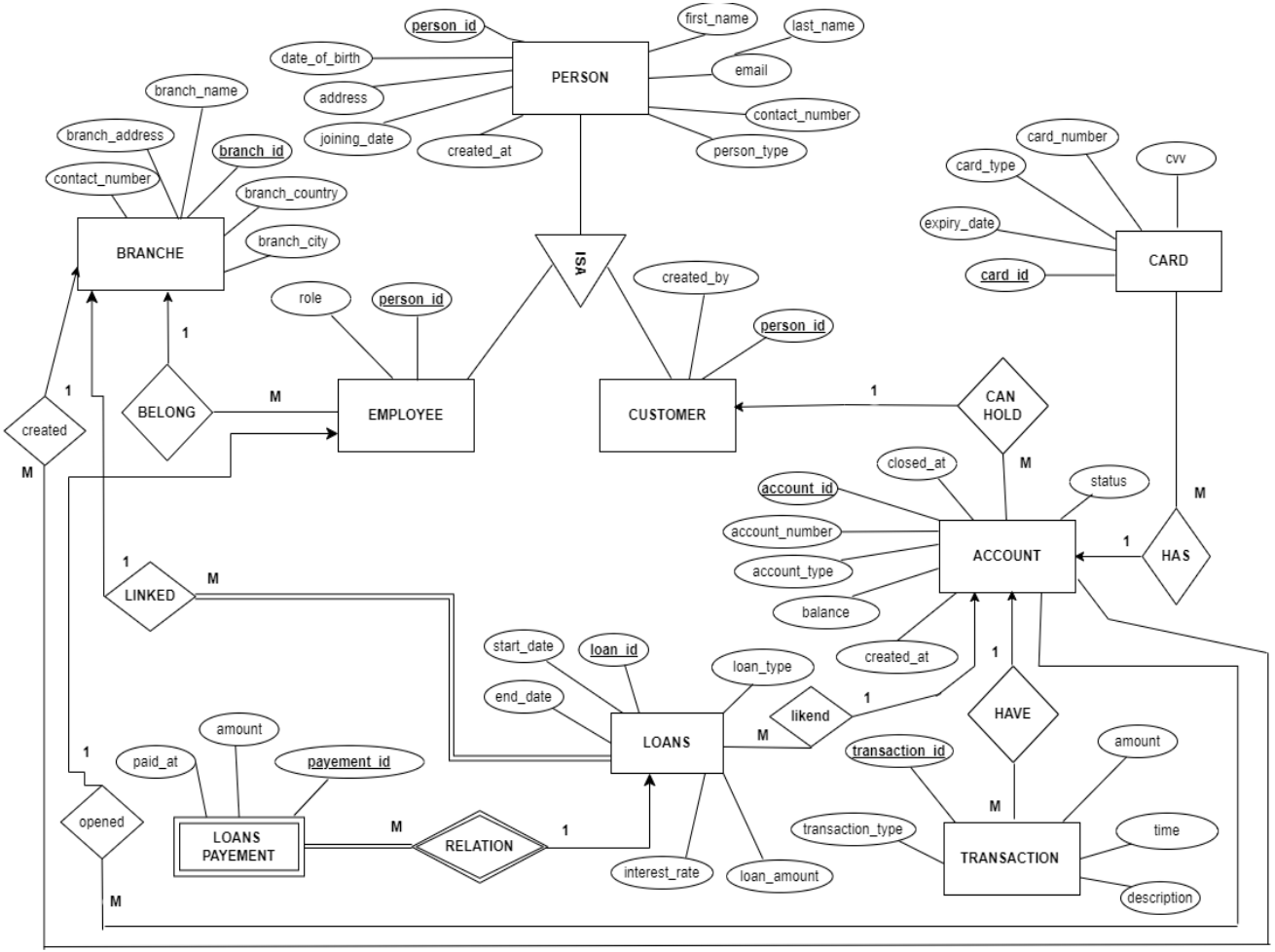
ENTITIES :

- PERSON, CUSTOMERS , ACCOUNTS, TRANSACTIONS, LOANS, LOANS PAYMENT , BRANCHES, EMPLOYEES, CARD .

RELATIONSHIPS :

- PERSON (PARENT) ARE LINKED TO CUSTOMER AND EMPLEYEE
- CUSTOMERS CAN HAVE MULTIPLE ACCOUNTS IN DIFFERENT BRANCHES
- AN ACCOUNT HAS A CARDS
- EACH ACCOUNT CAN HAVE MULTIPLE TRANSACTIONS
- LOANS ARE LINKED TO ACCOUNT AND BRANCHES
- LOAN PAYMENT HAS LINKED TO LOANS
- EMPLOYEES BELONG TO BRANCHES



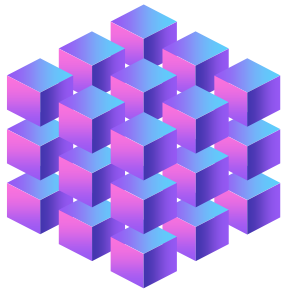

person_id
first_name
last_name
date_of_birth
email
PERSON
address
contact_number
joining_date
created_at
person_type
branch_name
branch_address
branch_id
contact_number
branch_country
BRANCHE
branch_city
ISA
created_by
role
person_id
person_id
1
BELONG
M
EMPLOYEE
CUSTOMER
1
CAN HOLD
M
created
1
M
card_number
card_type
cvv
expiry_date
CARD
card_id
M
closed_at
status
account_id
account_number
account_type
balance
ACCOUNT
1
HAS
created_at
1
1
LINKED
M
start_date
loan_id
loan_type
end_date
likend
1
HAVE
LOANS
M
transaction_id
amount
1
paid_at
amount
payement_id
M
transaction_type
time
opened
LOANS PAYEMENT
M
RELATION
1
TRANSACTION
interest_rate
loan_amount
description
M

Step Three : Data Definition Language

```sql
CREATE TABLE `Branch` (
  `branch_id` int PRIMARY KEY AUTO_INCREMENT,
  `branch_name` varchar(128),
  `branch_country` varchar(32),
  `branch_city` varchar(32),
  `branch_address` text,
  `contact_number` varchar(16)
);
```

```sql
CREATE TABLE `Person` (
  `person_id` int PRIMARY KEY AUTO_INCREMENT,
  `person_type` int,
  `first_name` varchar(128),
  `last_name` varchar(128),
  `email` varchar(128),
  `contact_number` varchar(16),
  `date_of_birth` date,
  `address` text,
  `joining_date` date,
  `created_at` timestamp
);
```

```sql
CREATE TABLE `Employee` (
  `person_id` int PRIMARY KEY AUTO_INCREMENT,
  `branch_id` int,
  `role` int
);

CREATE TABLE `Customer` (
  `person_id` int PRIMARY KEY AUTO_INCREMENT,
  `created_by_id` int
);
```

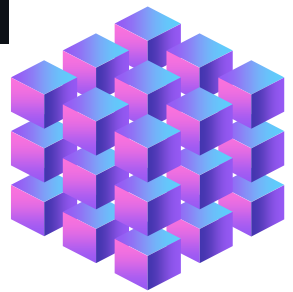


Step Three : Data Definition Language

```
CREATE TABLE `Account` (
  `account_id` int PRIMARY KEY AUTO_INCREMENT,
  `account_number` varchar(32),
  `person_id` int,
  `account_type` int,
  `balance` decimal(15,2),
  `status` int,
  `opened_in_branch_id` int,
  `opened_by_employee_id` int,
  `created_at` timestamp,
  `closed_at` timestamp
);
```

```
CREATE TABLE `Transaction` (
  `transaction_id` int PRIMARY KEY AUTO_INCREMENT,
  `account_id` int,
  `transaction_type` int,
  `amount` decimal(15,2),
  `time` timestamp,
  `description` text
);
```

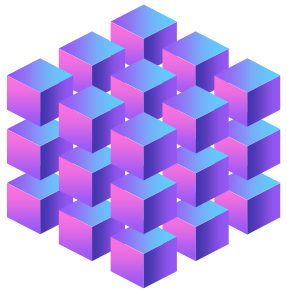
```
CREATE TABLE `Card` (
  `card_id` int PRIMARY KEY AUTO_INCREMENT,
  `account_id` int,
  `card_type` int,
  `card_number` varchar(16),
  `cvv` int,
  `expiry_date` date
);
```



Step Three : Data Definition Language

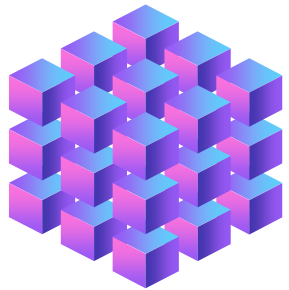
```
CREATE TABLE `Loan` (
  `loan_id` int PRIMARY KEY AUTO_INCREMENT,
  `account_id` int,
  `loan_type` int,
  `loan_amount` decimal(15,2),
  `interest_rate` decimal(5,2),
  `start_date` date,
  `end_date` date
);
```

```
CREATE TABLE `Loan_Payment` (
  `payment_id` int PRIMARY KEY AUTO_INCREMENT,
  `loan_id` int,
  `amount` decimal(15,2),
  `paid_at` timestamp
);
```



Step Three : Data Definition Language

```
ALTER TABLE `Employee` ADD FOREIGN KEY (`person_id`) REFERENCES `Person` (`person_id`);

ALTER TABLE `Customer` ADD FOREIGN KEY (`person_id`) REFERENCES `Person` (`person_id`);

ALTER TABLE `Customer` ADD FOREIGN KEY (`created_by_id`) REFERENCES `Person` (`person_id`);

ALTER TABLE `Employee` ADD FOREIGN KEY (`branch_id`) REFERENCES `Branch` (`branch_id`);

ALTER TABLE `Account` ADD FOREIGN KEY (`person_id`) REFERENCES `Person` (`person_id`);

ALTER TABLE `Account` ADD FOREIGN KEY (`opened_by_employee_id`) REFERENCES `Person` (`person_id`);

ALTER TABLE `Account` ADD FOREIGN KEY (`opened_in_branch_id`) REFERENCES `Branch` (`branch_id`);

ALTER TABLE `Card` ADD FOREIGN KEY (`account_id`) REFERENCES `Account` (`account_id`);

ALTER TABLE `Transaction` ADD FOREIGN KEY (`account_id`) REFERENCES `Account` (`account_id`);

ALTER TABLE `Loan` ADD FOREIGN KEY (`account_id`) REFERENCES `Account` (`account_id`);

ALTER TABLE `Loan_Payment` ADD FOREIGN KEY (`loan_id`) REFERENCES `Loan` (`loan_id`);
```



Step Four : Data Manipulation Language

INSERT DATA FOR TABLE BRANCH

INSERT INTO "Branch" (branch_id, branch_name, branch_country, branch_city, branch_address, contact_number)

VALUES

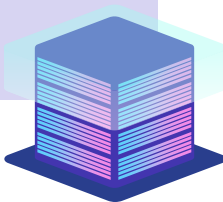
- (1, 'Central Branch', 'USA', 'New York', '100 Main Street', '+1-212-555-0101'),
- (2, 'West Branch', 'USA', 'Los Angeles', '200 West Sunset Blvd', '+1-310-555-0102'),
- (3, 'East Branch', 'USA', 'Boston', '300 Commonwealth Ave', '+1-617-555-0103'),
- (4, 'North Branch', 'Canada', 'Toronto', '400 Maple Street', '+1-416-555-0104'),
- (5, 'South Branch', 'USA', 'Miami', '500 Ocean Drive', '+1-305-555-0105'),
- (6, 'London Branch', 'UK', 'London', '600 High Street', '+44-20-555-0106'),
- (7, 'Paris Branch', 'France', 'Paris', '700 Rue de Rivoli', '+33-1-555-0107'),
- (8, 'Berlin Branch', 'Germany', 'Berlin', '800 Unter den Linden', '+49-30-555-0108'),
- (9, 'Tokyo Branch', 'Japan', 'Tokyo', '900 Shibuya Crossing', '+81-3-555-0109'),
- (10, 'Sydney Branch', 'Australia', 'Sydney', '1000 George Street', '+61-2-555-0110');

INSERT DATA FOR TABLE ACCOUNT

INSERT INTO "Account" (account_number, person_id, account_type, balance, status, opened_in_branch_id, opened_by_employee_id, created_at, closed_at)

VALUES

- ('ACC-1001', 10001, 1, 1000.20, 1, 1, 10005, '2018-12-31', NULL),
- ('ACC-1002', 10009, 2, 2500.50, 1, 9, 10507, '2020-10-07', NULL),
- ('ACC-1003', 10011, 1, 501.00, 1, 3, 10015, '2019-05-10', NULL),
- ('ACC-1004', 10027, 2, 10000.00, 1, 4, 10036, '2021-11-02', NULL),
- ('ACC-1005', 10028, 2, 750.75, 1, 7, 10237, '2024-06-17', NULL),
- ('ACC-1006', 10051, 1, 220.00, 1, 6, 10171, '2023-02-24', NULL),
- ('ACC-1007', 10101, 2, 3000.00, 1, 5, 10109, '2022-09-30', NULL),
- ('ACC-1008', 10189, 1, 400.00, 1, 8, 10251, '2020-08-28', NULL),
- ('ACC-1009', 10194, 2, 9900.99, 1, 2, 10008, '2023-01-13', NULL),
- ('ACC-1010', 10266, 1, 50.00, 1, 10, 10532, '2018-04-16', NULL);



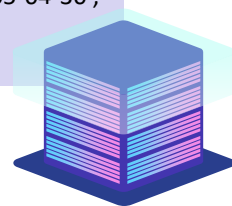
Step Four : Data Manipulation Language

INSERT DATA FOR TABLE PERSON

INSERT INTO "Person" (person_ID, first_name, last_name, email, contact_number, date_of_birth, address, person_type, joining_date, created_at)

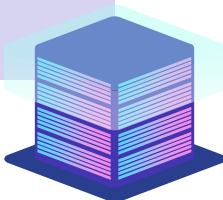
VALUES

- (10001, 'John', 'Peters', 'john.peters@bankexample.com', '+1-212-555-1001', '1979-04-15', '101 Park Ave, New York, USA', 'Customer', '2008-03-15', NOW()),
- (10005, 'Jane', 'Smith', 'jane.smith@bankexample.com', '+1-310-555-1002', '1982-08-21', '202 West Blvd, Los Angeles, USA', 'Employee', '2009-07-10', NOW()),
- (10008, 'Robert', 'Johnson', 'robert.johnson@bankexample.com', '+1-617-555-1003', '1975-11-30', '303 Beacon St, Boston, USA', 'Employee', '2007-05-25', NOW()),
- (10009, 'Emily', 'Davis', 'emily.davis@bankexample.com', '+1-416-555-1004', '1985-01-05', '404 Maple Rd, Toronto, Canada', 'Customer', '2010-02-14', NOW()),
- (10011, 'Michael', 'Wilson', 'michael.wilson@bankexample.com', '+1-305-555-1005', '1978-06-11', '505 Ocean Dr, Miami, USA', 'Customer', '2006-11-01', NOW()),
- (10015, 'Karen', 'Brown', 'karen.brown@bankexample.com', '+44-20-555-1006', '1980-09-12', '606 Regent St, London, UK', 'Employee', '2011-09-10', NOW()),
- (10027, 'David', 'Jones', 'david.jones@bankexample.com', '+33-1-555-1007', '1977-12-02', '707 Rue Cler, Paris, France', 'Customer', '2005-04-30', NOW()),



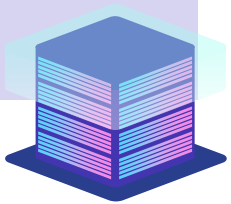
Step Four : Data Manipulation Language

- (10028, 'Linda', 'Garcia', 'linda.garcia@bankexample.com', '+49-30-555-1008', '1983-03-19', '808 Alexanderplatz, Berlin, Germany', 'customer', '2012-12-12', NOW()),
- (10036, 'Mark', 'Miller', 'mark.miller@bankexample.com', '+81-3-555-1009', '1976-10-27', '909 Shinjuku, Tokyo, Japan', 'Employee', '2008-08-08', NOW()),
- (10051, 'Patricia', 'Taylor', 'patricia.taylor@bankexample.com', '+61-2-555-1010', '1981-07-16', '1010 Darling St, Sydney, Australia', 'Customer', '2009-04-04', NOW()),
- (10101, 'Alice', 'Anderson', 'alice.anderson@example.com', '+1-212-555-2001', '1990-03-14', '1101 Broadway, New York, USA', 'Customer', '2020-01-10', NOW()),
- (10109, 'Peter', 'Thomas', 'peter.thomas@example.com', '+1-310-555-2002', '1992-05-29', '1202 Vine St, Los Angeles, USA', 'Employee', '2020-02-15', NOW()),
- (10171, 'Susan', 'Jackson', 'susan.jackson@example.com', '+1-617-555-2003', '1988-11-03', '1303 Commonwealth Ave, Boston, USA', 'Employee', '2020-03-20', NOW()),
- (10189, 'Daniel', 'White', 'daniel.white@example.com', '+1-416-555-2004', '1991-09-07', '1404 King St, Toronto, Canada', 'Customer', '2020-04-25', NOW()),
- (10194, 'Sarah', 'Harris', 'sarah.harris@example.com', '+1-305-555-2005', '1989-01-12', '1505 Collins Ave, Miami, USA', 'Customer', '2020-05-30', NOW()),
- (10237, 'Jessica', 'Martin', 'jessica.martin@example.com', '+44-20-555-2006', '1993-07-08', '1606 Baker St, London, UK', 'Employee', '2020-06-05', NOW()),



Step Four : Data Manipulation Language

- (10251, 'Thomas', 'Thompson', 'thomas.thompson@example.com', '+33-1-555-2007', '1994-12-19', '1707 Montparnasse, Paris, France', 'Employee', '2020-07-10', NOW()),
- (10266, 'Maria', 'Moore', 'maria.moore@example.com', '+49-30-555-2008', '1996-04-01', '1808 Potsdamer Platz, Berlin, Germany', 'Customer', '2020-08-15', NOW()),
- (10507, 'James', 'Clark', 'james.clark@example.com', '+81-3-555-2009', '1995-02-23', '1909 Ginza, Tokyo, Japan', 'Employee', '2020-09-20', NOW()),
- (10532, 'Laura', 'Lee', 'laura.lee@example.com', '+61-2-555-2010', '1993-06-14', '2010 Circular Quay, Sydney, Australia', 'Employee', '2020-10-10', NOW());



Step Four : Data Manipulation Language

ATTRIBUTE PERSON_TYPE CHANGE FROM INTEGER TO VARCHAR

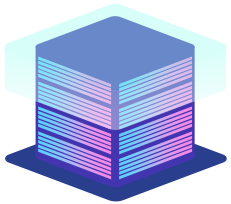
```
ALTER TABLE "Person"
ALTER COLUMN person_type
SET DATA TYPE VARCHAR(16);
```

ENSURE NO DUPLICATE EMAIL IN THE PERSON TABLE

```
ALTER TABLE "Person"
ADD CONSTRAINT unqiue_email UNIQUE(email);
```

ENSURE NO DUPLICATE CONTACT_NUMBER IN THE PERSON TABLE

```
ALTER TABLE "Person"
ADD CONSTRAINT unqiue_contact UNIQUE(contact_number);
```



Step Four : Data Manipulation Language

INSERT DATA FOR TABLE EMPLOYEE

INSERT INTO "Employee" (person_id, branch_id, role)

VALUES

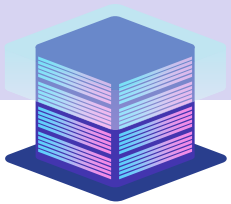
- (10005, 1, 01),
- (10008, 2, 02),
- (10015, 3, 01),
- (10036, 1, 02),
- (10109, 5, 01),
- (10171, 6, 01),
- (10237, 1, 02),
- (10251, 8, 01),
- (10507, 9, 02),
- (10532, 10, 01);

INSERT DATA FOR TABLE CUSTOMER

INSERT INTO "Customer" (person_id, created_by_id)

VALUES

- (10001, 10532),
- (10009, 10532),
- (10011, 10532),
- (10027, 10109),
- (10028, 10251),
- (10051, 10532),
- (10101, 10532),
- (10189, 10507),
- (10194, 10015),
- (10266, 10036);



Step Four : Data Manipulation Language

INSERT DATA FOR TABLE CARD

INSERT INTO "Card" (account_id, card_type, card_number, cvv, expiry_date)

VALUES

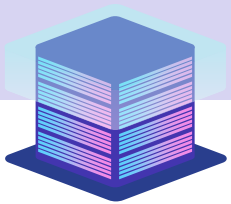
- ('1', 1, '4111111111111111', 123, '2025-12-31'),
- ('2', 1, '5500000000000004', 456, '2025-11-30'),
- ('3', 2, '4111111111111112', 234, '2026-10-31'),
- ('4', 2, '5500000000000005', 567, '2025-09-30'),
- ('5', 1, '4111111111111113', 345, '2026-08-31'),
- ('6', 1, '4111111111111114', 678, '2025-07-31'),
- ('7', 2, '5500000000000006', 789, '2026-06-30'),
- ('8', 1, '4111111111111115', 890, '2025-05-31'),
- ('9', 2, '5500000000000007', 901, '2026-04-30'),
- ('10', 1, '4111111111111116', 321, '2026-03-31');

INSERT DATA FOR TABLE TRANSACTION

INSERT INTO "Transaction" (account_id, transaction_type, amount, time, description)

VALUES

- ('8', 1, 100.00, ('2024-02-21 09:10:05'), 'Initial Deposit'),
- ('7', 2, 50.00, ('2022-07-18 12:22:41'), 'ATM Withdrawal'),
- ('1', 1, 500.00, ('2024-12-31 11:56:26'), 'ACH Deposit'),
- ('9', 1, 200.00, ('2023-10-27 08:15:03'), 'Check Deposit'),
- ('3', 2, 100.00, ('2024-05-10 07:27:15'), 'Online Purchase'),
- ('2', 1, 300.00, ('2021-12-12 10:48:00'), 'Cash Deposit'),
- ('4', 2, 150.00, ('2022-08-14 11:07:52'), 'Bill Payment'),
- ('5', 3, 1000.00, ('2024-09-09 00:12:38'), 'Wire Transfer In'),
- ('10', 2, 200.00, ('2020-11-01 12:55:11'), 'Debit Card Purchase'),
- ('6',1, 400.00, ('2023-12-07 10:31:22'), 'Mobile Deposit');



Step Four : Data Manipulation Language

INSERT DATA FOR TABLE LOAN

INSERT INTO "Loan" (account_id, loan_id, loan_type, loan_amount, interest_rate, start_date, end_date)

VALUES

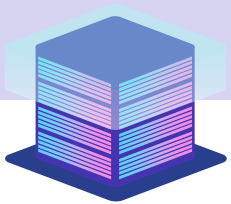
- ('2', 0290, 1, 5000.00, 5.00, '2024-01-30', '2026-01-29'),
- ('7', 1059, 1, 10000.00, 4.50, '2024-02-19', '2027-02-18'),
- ('4', 0001, 2, 250000.00, 3.25, '2024-03-09', '2030-03-08'),
- ('6', 2832, 1, 2000.00, 6.00, '2024-04-15', '2025-04-14'),
- ('1', 2010, 2, 150000.00, 3.50, '2024-05-08', '2028-05-07'),
- ('3', 4372, 1, 7500.00, 5.25, '2024-06-20', '2027-06-19'),
- ('10', 3904, 2, 300000.00, 3.00, '2024-07-16', '2029-07-15'),
- ('5', 8392, 1, 4000.00, 5.75, '2024-08-31', '2026-08-01'),
- ('8', 2013, 1, 12000.00, 4.75, '2024-09-18', '2026-09-17'),
- ('9', 2387, 2, 200000.00, 3.20, '2024-10-11', '2031-10-10');

INSERT DATA FOR TABLE LOAN_PAYMENT

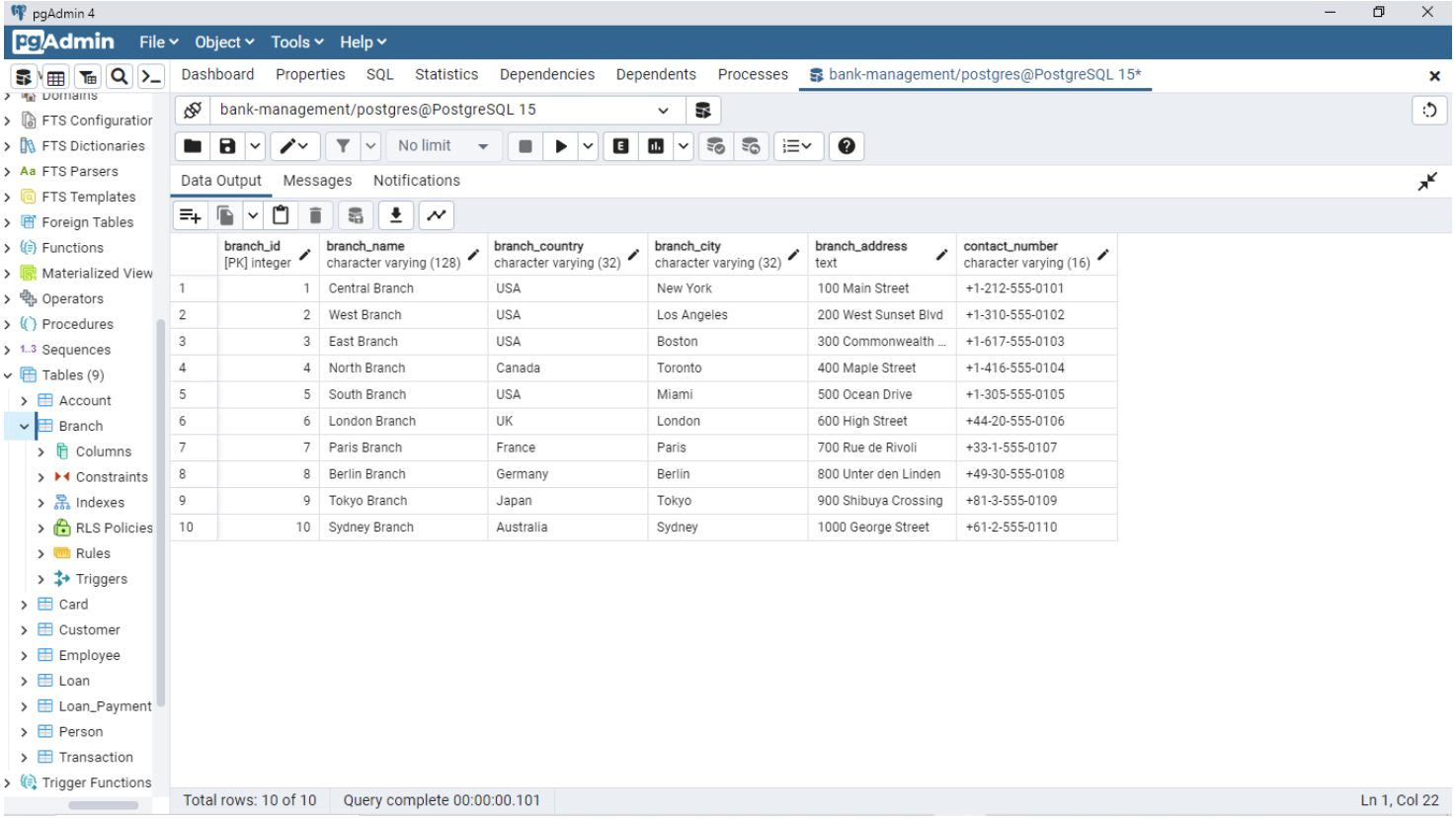
INSERT INTO "Loan_Payment" (payment_id, amount, paid_at)

VALUES

- (0290, 500.00, '2024-04-29'),
- (2832, 600.00, '2024-06-14'),
- (2387, 2000.00, '2024-11-10'),
- (1059, 250.00, '2025-12-18'),
- (2010, 1500.00, '2027-12-07'),
- (8392, 750.00, '2026-10-01'),
- (0001, 30000.00, '2028-05-08'),
- (3904, 400.00, '2029-02-15'),
- (2013, 1200.00, '2025-03-17'),
- (4372,2000.00, '2025-01-19');

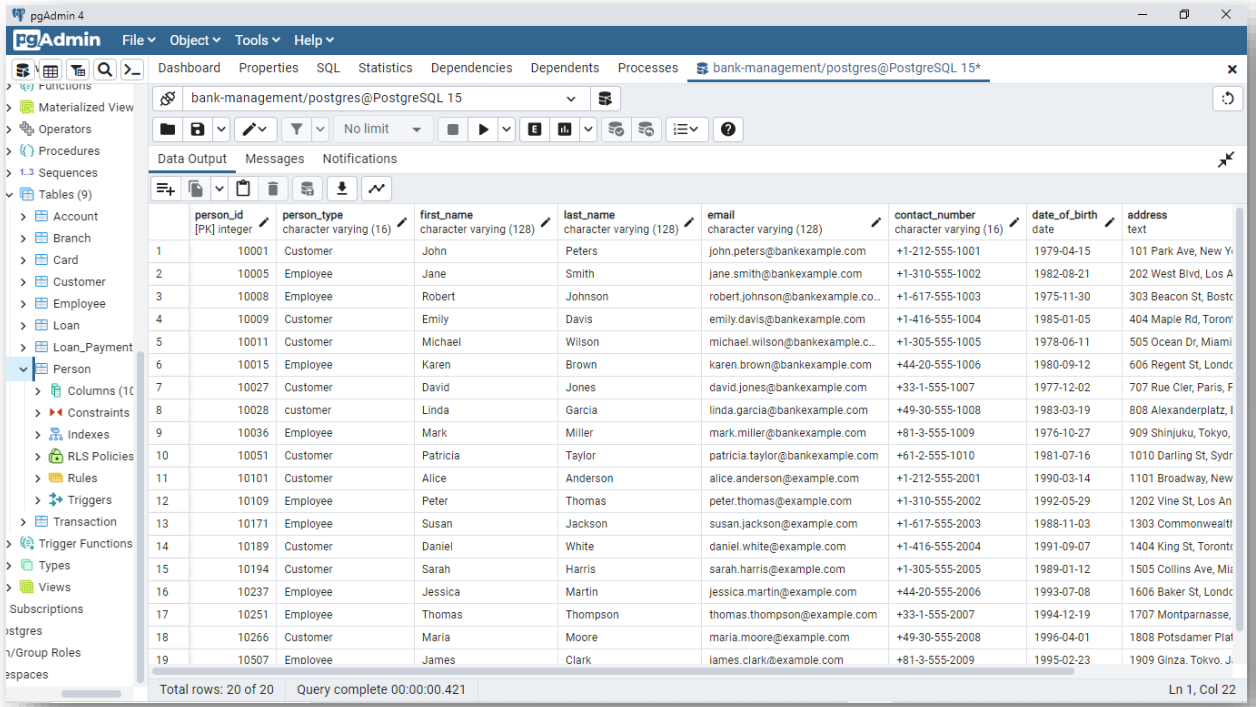


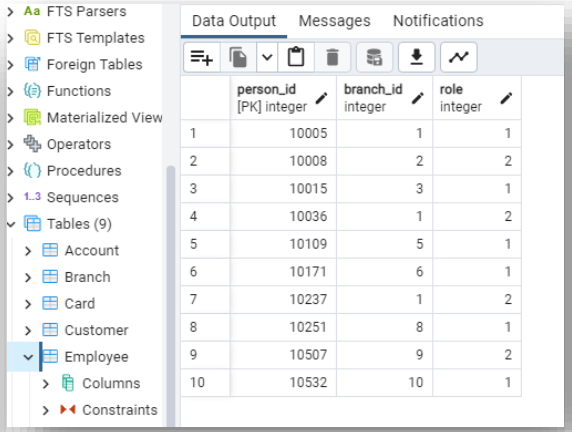
Sample Tables

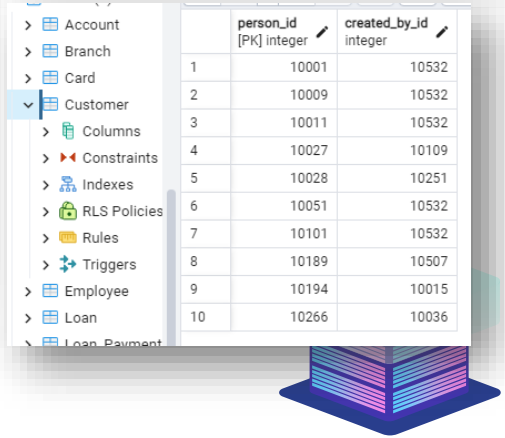


	branch_id [PK] integer	branch_name character varying (128)	branch_country character varying (32)	branch_city character varying (32)	branch_address text	contact_number character varying (16)
1	1	Central Branch	USA	New York	100 Main Street	+1-212-555-0101
2	2	West Branch	USA	Los Angeles	200 West Sunset Blvd	+1-310-555-0102
3	3	East Branch	USA	Boston	300 Commonwealth ...	+1-617-555-0103
4	4	North Branch	Canada	Toronto	400 Maple Street	+1-416-555-0104
5	5	South Branch	USA	Miami	500 Ocean Drive	+1-305-555-0105
6	6	London Branch	UK	London	600 High Street	+44-20-555-0106
7	7	Paris Branch	France	Paris	700 Rue de Rivoli	+33-1-555-0107
8	8	Berlin Branch	Germany	Berlin	800 Unter den Linden	+49-30-555-0108
9	9	Tokyo Branch	Japan	Tokyo	900 Shibuya Crossing	+81-3-555-0109
10	10	Sydney Branch	Australia	Sydney	1000 George Street	+61-2-555-0110

Sample Tables







Step Five : Related Queries

Account number of all accounts opened by Laura Lee

bank-management/postgres@PostgreSQL 15

No limit

Query | Query History

```
1  SELECT account_number
2  FROM "Account" a
3  JOIN "Person" p ON a.person_id = p.person_id
4  JOIN "Customer" c ON c.person_id = p.person_id
5  WHERE c.created_by_id =
6      (SELECT person_id
7       FROM "Person" p
8       WHERE p.first_name = 'Laura'
9       AND p.last_name = 'Lee');
```

Data Output | Messages | Notifications

	account_number character varying (32)
1	ACC-1001
2	ACC-1002
3	ACC-1003
4	ACC-1006
5	ACC-1007

Step Five : Related Queries cont.

Find a person who created an account very close to or on the same day as their birthday

bank-management/postgres@PostgreSQL 15

No limit

Query | Query History

```
1  SELECT
2      p.first_name || ' ' || p.last_name name,
3      p.date_of_birth,
4      a.created_at AS account_creation_date,
5      ABS(
6              (EXTRACT(DOY FROM a.created_at) -
7               EXTRACT(DOY FROM p.date_of_birth))
8          )
9      AS days_to_birthday
10 FROM
11     "Person" p
12 JOIN
13     "Account" a ON p.person_id = a.person_id
14 ORDER BY
15     days_to_birthday ASC
16 LIMIT 1;
17
```

Data Output | Messages | Notifications

	name text	date_of_birth date	account_creation_date timestamp without time zone	days_to_birthday numeric
1	Sarah Harris	1989-01-12	2023-01-13 00:00:00	1

Step Five : Related Queries cont.

Find out which country has the highest total account balance by grouping accounts by countries

bank-management/postgres@PostgreSQL 15

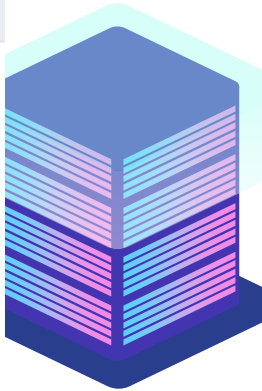
No limit

Query | Query History

```
1  SELECT
2      b.branch_country,
3      SUM(a.balance) AS total_balance
4  FROM
5      "Branch" b
6  JOIN
7      "Account" a ON b.branch_id = a.opened_in_branch_id
8  GROUP BY
9      b.branch_country
10 ORDER BY
11     total_balance DESC;
```

Data Output | Messages | Notifications

	branch_country character varying (32)	total_balance numeric
1	USA	14402.19
2	Canada	10000.00
3	Japan	2500.50
4	France	750.75
5	Germany	400.00
6	UK	220.00
7	Australia	50.00



Step Five : Related Queries cont.

Find the number of loans taken and the total loan amount by loan type

Query | Query History

```
1  SELECT
2      "loan_type",
3      COUNT("loan_id") AS loan_count,
4      SUM("loan_amount") AS total_loan_amount
5  FROM
6      "Loan"
7  GROUP BY
8      "loan_type";
```

Data Output | Messages | Notifications

	loan_type integer	loan_count bigint	total_loan_amount numeric
1	2	4	900000.00
2	1	6	40500.00

Step Five : Related Queries cont.

List the shortened version of the account number (first 5 characters), and the formatted balance of all active accounts.

bank-management/postgres@PostgreSQL 15

No limit

Query | Query History

```
1  SELECT
2      account_id,
3      SUBSTRING(account_number FROM 1 FOR 5)
4      AS short_account_number,
5
6      TO_CHAR(ROUND(balance, 2), '$99,999.00')
7      AS formatted_balance
8
9  FROM
10     "Account"
11 WHERE
12     status = 1;
13
```

Data Output | Messages | Notifications

	account_id [PK] integer	short_account_number text	formatted_balance text
1	1	ACC-1	$ 1,000.20
2	2	ACC-1	$ 2,500.50
3	3	ACC-1	$ 501.00
4	4	ACC-1	$ 10,000.00
5	5	ACC-1	$ 750.75
6	6	ACC-1	$ 220.00
7	7	ACC-1	$ 3,000.00
8	8	ACC-1	$ 400.00
9	9	ACC-1	$ 9,900.99
10	10	ACC-1	$ 50.00

Step Five : Related Queries cont.

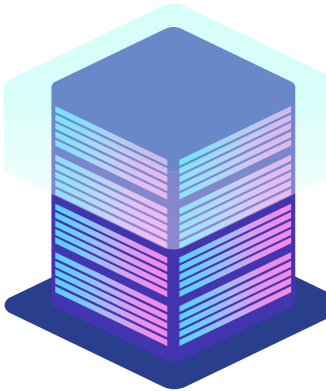
List the name of all customers who haven't opened account in New York

Query | Query History

```
1  SELECT first_name || ' ' || last_name name
2  FROM "Person" p
3  JOIN "Customer" c
4      ON c.person_id = p.person_id
5  JOIN "Employee" e
6      ON e.person_id = c.created_by_id
7  JOIN "Branch" b
8      ON b.branch_id = e.branch_id
9  WHERE b.branch_city <> 'New York';
10
```

Data Output | Messages | Notifications

	name text
1	John Peters
2	Emily Davis
3	Michael Wilson
4	David Jones
5	Linda Garcia
6	Patricia Taylor
7	Alice Anderson
8	Daniel White
9	Sarah Harris



Step Five : Related Queries cont.

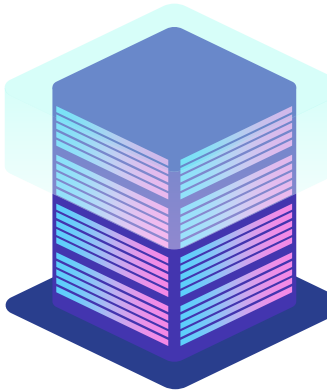
List all branches with the number of employees in each branch

Query | Query History

```
1  SELECT
2      branch_name,
3      COUNT(e.person_id) AS employee_count
4  FROM
5      "Branch" b
6  LEFT JOIN
7      "Employee" e ON b.branch_id = e.branch_id
8  GROUP BY
9      branch_name;
10
```

Data Output | Messages | Notifications

	branch_name character varying (128)	employee_count bigint
1	Paris Branch	0
2	Tokyo Branch	1
3	North Branch	0
4	Sydney Branch	1
5	Central Branch	3
6	London Branch	1
7	South Branch	1
8	West Branch	1
9	East Branch	1
10	Berlin Branch	1



Step Five : Related Queries cont.

Calculate the average account balance by account type

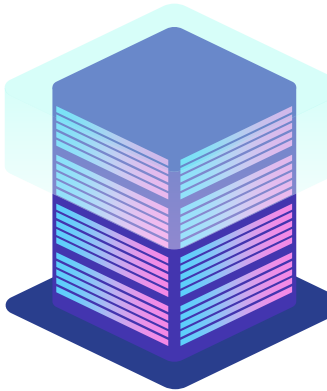
No limit

Query | Query History

```
1  SELECT
2      account_type,
3      AVG(balance) AS average_balance
4  FROM
5      "Account" a
6  GROUP BY
7      account_type;
8
9
```

Data Output | Messages | Notifications

	account_type integer	average_balance numeric
1	2	5230.448
2	1	434.24



Step Five : Related Queries cont.

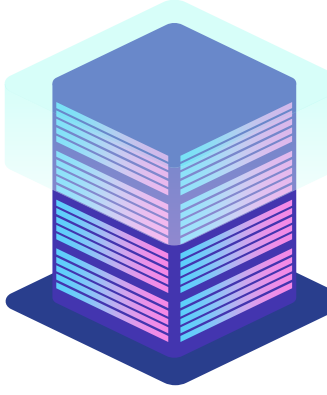
List card information of all customers with an outstanding loan above 10000

Query | Query History

```
1  SELECT c.*
2  FROM "Card" c
3  LEFT JOIN "Account" a ON a.account_id = c.account_id
4  WHERE a.account_id IN (
5      SELECT account_id
6      FROM "Loan"
7      WHERE loan_amount > 10000);
```

Data Output | Messages | Notifications

	card_id [PK] integer	account_id integer	card_type integer	card_number character varying (16)	cvv integer	expiry_date date
1	1	1	1	4111111111111111	123	2025-12-31
2	4	4	2	5500000000000005	567	2025-09-30
3	8	8	1	4111111111111115	890	2025-05-31
4	9	9	2	5500000000000007	901	2026-04-30
5	10	10	1	4111111111111116	321	2026-03-31



Step Five : Related Queries cont.

Find the top 5 customers with the highest account balances

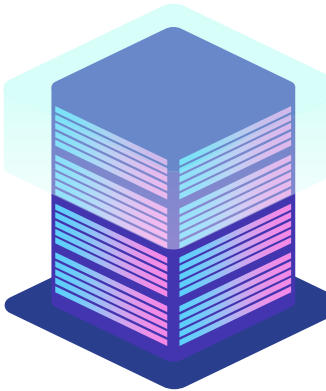
No limit

Query | Query History

```
1  SELECT
2      first_name || ' ' || last_name name,
3      balance
4  FROM "Person" p
5  INNER JOIN
6      "Account" a ON p.person_id = a.person_id
7  ORDER BY balance DESC
8  LIMIT 5;
```

Data Output | Messages | Notifications

	name text	balance numeric (15,2)
1	David Jones	10000
2	Sarah Harris	9900.99
3	Alice Anderson	3000
4	Emily Davis	2500.5
5	John Peters	1000.2



Step Five : Related Queries cont.

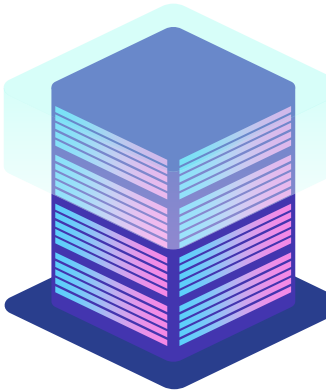
Display the total loan payments made for each loan.

Query | Query History

```
1  SELECT
2      l.loan_id,
3      SUM(lp.amount) AS "Total Payment"
4  FROM
5      "Loan" l
6  JOIN
7      "Loan_Payment" lp ON l.loan_id = lp.loan_id
8  GROUP BY
9      l.loan_id;
10
```

Data Output | Messages | Notifications

	loan_id [PK] integer	Total Payment numeric
1	8392	750
2	2010	1500
3	2387	2000
4	3904	400
5	4372	2000
6	1059	250
7	290	500
8	1	30000
9	2013	1200
10	2832	600



Step Five : Related Queries cont.

Find the last transaction of youngest person who withdrew money from an ATM

```sql
SELECT
    p.first_name,
    p.last_name,
    p.date_of_birth,
    t.time AS last_withdrawal_time
FROM
    "Person" p
JOIN
    "Account" a ON p.person_id = a.person_id
JOIN
    "Transaction" t ON a.account_id = t.account_id
WHERE
    t.description = 'ATM Withdrawal'
ORDER BY
    t.time DESC,
    p.date_of_birth DESC
LIMIT 1;
```

	first_name character varying (128)	last_name character varying (128)	date_of_birth date
1	Alice	Anderson	1990-03-14

Step Five : Related Queries cont.

Combine Customers and Employees' Emails

Query | Query History

```
1  SELECT email
2  FROM "Person"
3  WHERE person_id IN (SELECT person_id FROM "Customer")
4  UNION
5  SELECT email
6  FROM "Person"
7  WHERE person_id IN (SELECT person_id FROM "Employee");
8
9
```

Data Output | Messages | Notifications

	email character varying (128)
1	patricia.taylor@bankexample.com
2	maria.moore@example.com
3	peter.thomas@example.com
4	jane.smith@bankexample.com
5	michael.wilson@bankexample.c...
6	jessica.martin@example.com
7	mark.miller@bankexample.com
8	susan.jackson@example.com
9	james.clark@example.com
10	thomas.thompson@example.com

Step Five : Related Queries cont.

Find the name, age, and address of cardholders for the three cards expiring soonest

No limit

Query | Query History

```
1  SELECT
2      p.first_name || ' ' || p.last_name name,
3      EXTRACT(YEAR FROM
4              AGE(CURRENT_DATE, p.date_of_birth))
5              AS age,
6      p.address,
7      c.card_number,
8      c.expiry_date
9  FROM
10     "Card" c
11 JOIN
12     "Account" a ON c.account_id = a.account_id
13 JOIN
14     "Person" p ON a.person_id = p.person_id
15 ORDER BY
16     c.expiry_date ASC
17 LIMIT 3;
```

Data Output | Messages | Notifications

	name text	age numeric	address text
1	Daniel White	33	1404 King St, Toronto, Canada
2	Patricia Taylor	43	1010 Darling St, Sydney, Australia
3	David Jones	47	707 Rue Cler, Paris, France

Step Five : Related Queries cont.

Group cards by their type and finds the most common transaction type for each card type

Query | Query History

```
1  SELECT
2      c.card_type,
3      t.description,
4      COUNT(t.transaction_id) AS transaction_count
5  FROM
6      "Card" c
7  JOIN
8      "Account" a ON c.account_id = a.account_id
9  JOIN
10     "Transaction" t ON a.account_id = t.account_id
11 GROUP BY
12     c.card_type, t.description
13 ORDER BY
14     c.card_type, transaction_count DESC;
```

Data Output | Messages | Notifications

	card_type integer	description text	transaction_count bigint
1	1	Debit Card Purchase	1
2	1	Mobile Deposit	1
3	1	Wire Transfer In	1
4	1	Initial Deposit	1
5	1	Cash Deposit	1
6	1	ACH Deposit	1
7	2	Online Purchase	1
8	2	ATM Withdrawal	1
9	2	Bill Payment	1
10	2	Check Deposit	1

Thank You For Your Attention.