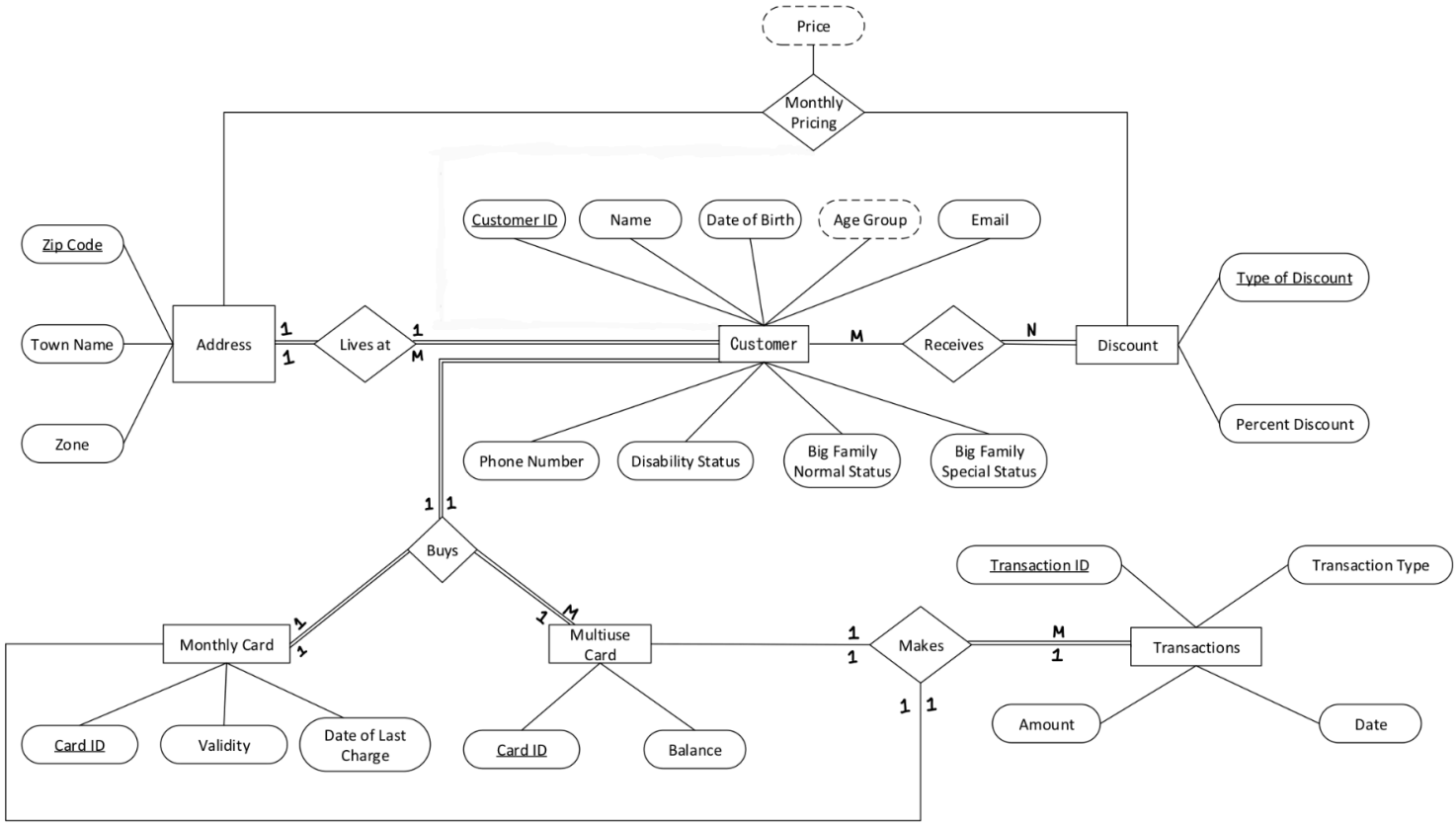
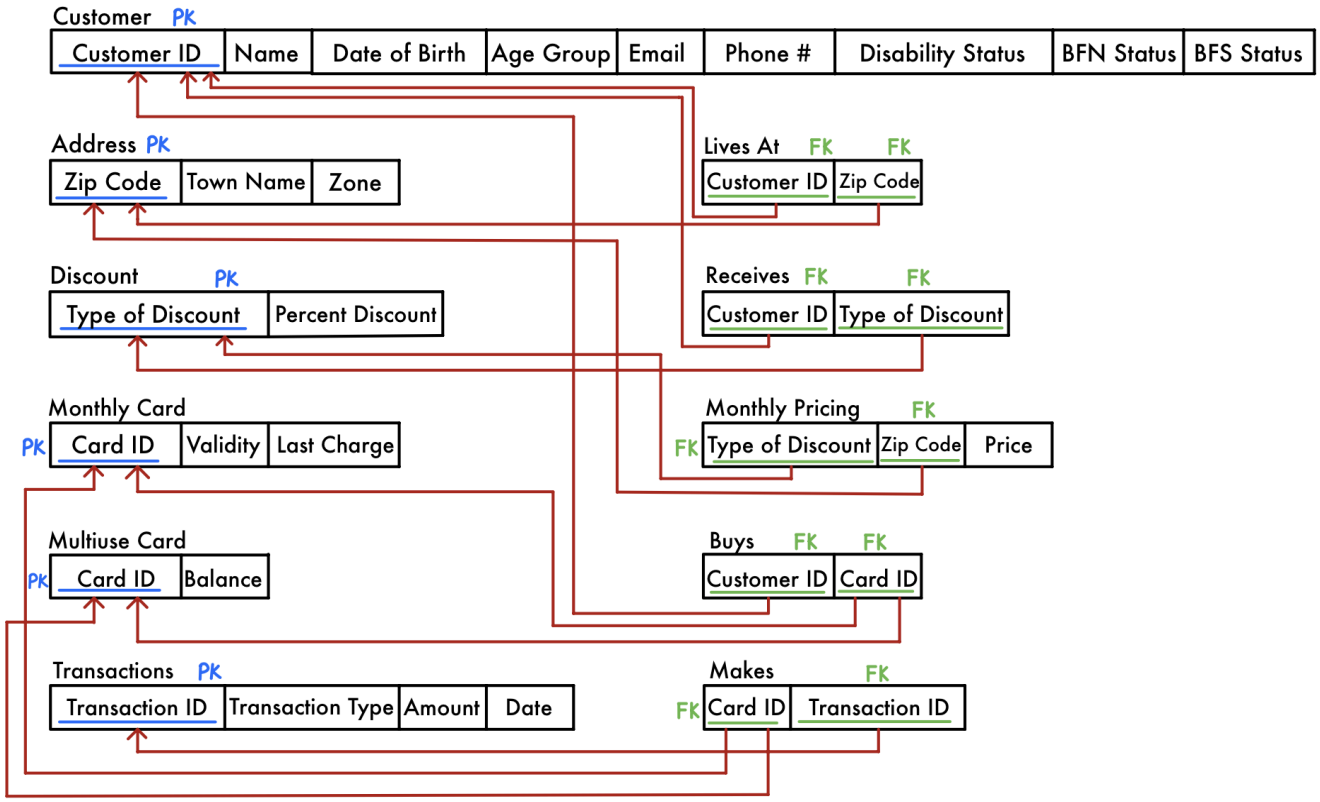
Jack Becker
Madrid Metro Database

ER Model, Relational Schema, & Metadata





Metadata Description:

Relation Name	Number of Columns
Customer	9
Address	3
Discount	2
Monthly Card	3
Multi-use Card	2
Transaction	4

Column Name	Data Type	Belongs-to-Relation
Customer ID	int(10)	Customer
Name	varchar(50)	Customer
Date of birth	date	Customer
Age Group	varchat(10)	Customer (derived)
Email	varchar(50)	Customer
Phone Number	int(15)	Customer
Disability Status	boolean	Customer
Big Family Normal Status	boolean	Customer
Big Family Special Status	boolean	Customer
Zip code	int(5)	Address
Town Name	varchar(100)	Address
Zone	varchar(2)	Address
Type of Discount	varchar(15)	Discount
Percent Discount	float	Discount
Card ID	int(10)	Monthly Card
Validity	boolean	Monthly Card
Date of last charge	date	Monthly Card
Card ID	int(10)	Multi-use Card
Balance	float	Multi-use Card

Transaction ID	int(15)	Transactions
Transaction Type	varchar(10)	Transactions
Amount	float	Transactions
Date	date	Transactions

Queries:

All of my queries were created using SQL procedures that carry out specific tasks and update the database accordingly. In the real-world, a metrocard machine could call these simple, single-line procedures any time it needs to complete a task.

1. **Register a new client**

Before ("customer" table):

customer_id	full_name	date_of_bir...	age_group	zip_code	email	phone_number	student_stat...	disability_stat...	family_normal_stat...	family_special_stat...
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

NOTE: this could also be done using "insert into customer" but I preferred a procedural approach so that new customers can be added later on without having to modify existing code.

```
-- Procedure to register a new customer and add them into the database
DROP PROCEDURE IF EXISTS RegisterNewCustomer;
DELIMITER $$
CREATE PROCEDURE `RegisterNewCustomer`(
  IN p_full_name VARCHAR(100),
  IN p_date_of_birth DATE,
  IN p_zip_code INT,
  IN p_email VARCHAR(100),
  IN p_phone_number VARCHAR(100),
  IN p_student_status BOOLEAN,
  IN p_disability_status BOOLEAN,
  IN p_family_normal_status BOOLEAN,
  IN p_family_special_status BOOLEAN
)
BEGIN
  INSERT INTO customer (
    full_name,
    date_of_birth,
    zip_code,
    email,
    phone_number,
    student_status,
    disability_status,
    family_normal_status,
```

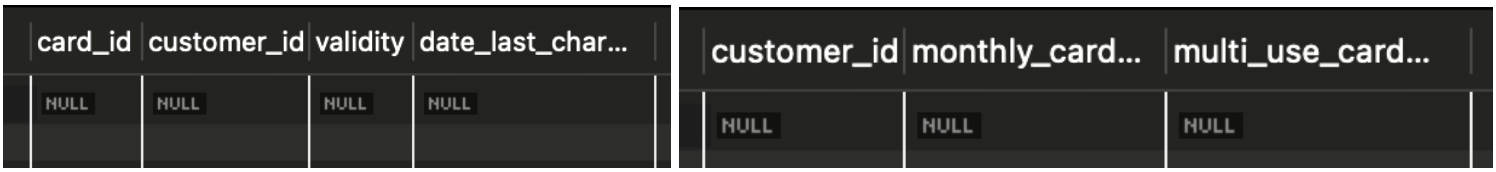
```
    family_special_status
  )
  VALUES (
    p_full_name,
    p_date_of_birth,
    p_zip_code,
    p_email,
    p_phone_number,
    p_student_status,
    p_disability_status,
    p_family_normal_status,
    p_family_special_status
  );
END$$
DELIMITER ;

CALL RegisterNewCustomer('John Doe', '1985-05-15', 28001, 'john.doe@email.com', '555-1234', true, false, true, false);
CALL RegisterNewCustomer('Jane Smith', '1998-07-22', 28100, 'jane.smith@email.com', '555-5678', false, false, false, true);
CALL RegisterNewCustomer('Carlos Ruiz', '2004-02-01', 28200, 'carlos.ruiz@email.com', '555-8765', true, true, false, true);
CALL RegisterNewCustomer('Aisha Lee', '1947-11-30', 28300, 'aisha.lee@email.com', '555-4321', false, false, false, false);
CALL RegisterNewCustomer('Xavier Hernández', '1990-03-15', 28205, 'xavier.hernandez@email.com', '555-1122', true, true, false, false);
CALL RegisterNewCustomer('Liu Wei', '2001-12-12', 28105, 'liu.wei@email.com', '555-3344', false, false, true, false);
CALL RegisterNewCustomer('Sara Alba', '1955-06-09', 28301, 'sara.alba@email.com', '555-7788', true, true, false, false);
CALL RegisterNewCustomer('Mohamed Al-Fayed', '1979-04-08', 28107, 'mohamed.alfayed@email.com', '555-9900', false, false, false, true);
CALL RegisterNewCustomer('Chloe Darcy', '2003-08-19', 28400, 'chloe.darcy@email.com', '555-2211', true, false, false, false);
CALL RegisterNewCustomer('Ivan Petrov', '1968-01-25', 28325, 'ivan.petrov@email.com', '555-5656', false, true, true, false)
```

After (“customer” table):

customer_id	full_name	date_of_bir...	age_group	zip_code	email	phone_number	student_stat...	disability_stat...	family_normal_stat...	family_special_stat.
1	John Doe	1985-05-15	Youth/Student	28001	john.doe@email.com	555-1234	1	0	1	0
2	Jane Smith	1998-07-22	Youth/Student	28100	jane.smith@email.com	555-5678	0	0	0	1
3	Carlos Ruiz	2004-02-01	Youth/Student	28200	carlos.ruiz@email.com	555-8765	1	1	0	1
4	Aisha Lee	1947-11-30	Senior	28300	aisha.lee@email.com	555-4321	0	0	0	0
5	Xavier Hernández	1990-03-15	Youth/Student	28205	xavier.hernandez@email.com	555-1122	1	1	0	0
6	Liu Wei	2001-12-12	Youth/Student	28105	liu.wei@email.com	555-3344	0	0	1	0
7	Sara Alba	1955-06-09	Youth/Student	28301	sara.alba@email.com	555-7788	1	1	0	0
8	Mohamed Al-Fayed	1979-04-08	Normal	28107	mohamed.alfayed@email.com	555-9900	0	0	0	1
9	Chloe Darcy	2003-08-19	Youth/Student	28400	chloe.darcy@email.com	555-2211	1	0	0	0
10	Ivan Petrov	1968-01-25	Normal	28325	ivan.petrov@email.com	555-5656	0	1	1	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. **Buy a new monthly metrocard**

Before (“monthly_card” table on left, “buys” table on right):

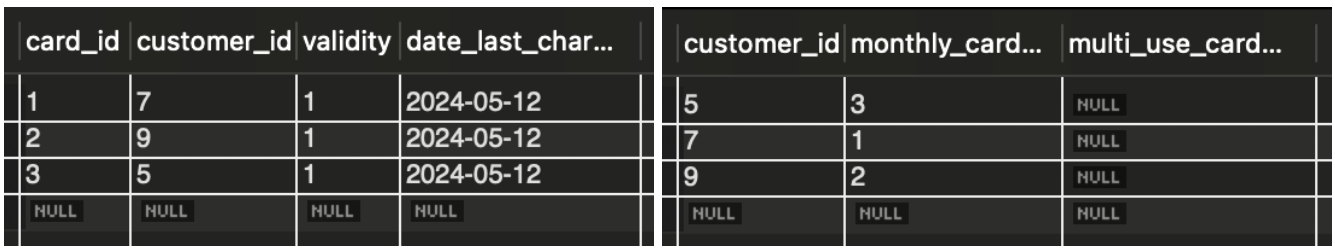


card_id	customer_id	validity	date_last_char...
NULL	NULL	NULL	NULL

customer_id	monthly_card...	multi_use_card...
NULL	NULL	NULL

```
-- Query/procedure to purchase a new monthly card
DROP PROCEDURE IF EXISTS IssueMonthlyCard;
DELIMITER $$
CREATE PROCEDURE `IssueMonthlyCard`(IN customer_id_param INT)
BEGIN
  DECLARE new_card_id INT;
  DECLARE existing_card_count INT;
  SELECT COUNT(*) INTO existing_card_count FROM monthly_card WHERE customer_id = customer_id_param;
  IF existing_card_count = 0 THEN
    INSERT INTO monthly_card (customer_id, validity, date_last_charge)
    VALUES (customer_id_param, TRUE, CURDATE());
    SET new_card_id = LAST_INSERT_ID();
    IF EXISTS (SELECT 1 FROM buys WHERE customer_id = customer_id_param) THEN
      UPDATE buys SET monthly_card_id = new_card_id WHERE customer_id = customer_id_param;
    ELSE
      INSERT INTO buys (customer_id, monthly_card_id) VALUES (customer_id_param, new_card_id);
    END IF;
  ELSE
    SELECT 'Customer already has a monthly card.' AS Message;
  END IF;
END$$
DELIMITER ;

Call IssueMonthlyCard(7);
Call IssueMonthlyCard(9);
Call IssueMonthlyCard(5);
```

After (“monthly_card” table on left, “buys” table on right):



card_id	customer_id	validity	date_last_char...
1	7	1	2024-05-12
2	9	1	2024-05-12
3	5	1	2024-05-12
NULL	NULL	NULL	NULL

customer_id	monthly_card...	multi_use_card...
5	3	NULL
7	1	NULL
9	2	NULL
NULL	NULL	NULL

3. **Buy a new multi-use metrocard**

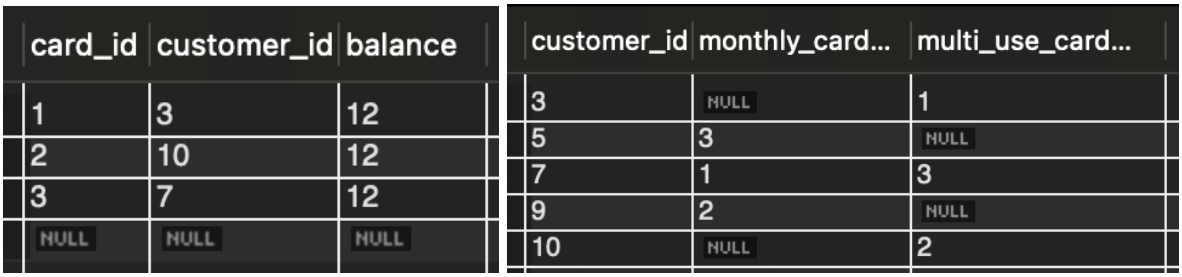
Before ("multi_use_card" table on left, "buys" table on right):

card_id	customer_id	balance
NULL	NULL	NULL

customer_id	monthly_card...	multi_use_card...
5	3	NULL
7	1	NULL
9	2	NULL
NULL	NULL	NULL

```
-- Query/procedure to purchase a new multi use card
DROP PROCEDURE IF EXISTS IssueMultiUseCard;
DELIMITER $$
CREATE PROCEDURE `IssueMultiUseCard`(IN customer_id_param INT)
BEGIN
  DECLARE new_card_id INT;
  INSERT INTO multi_use_card (customer_id, balance) VALUES (customer_id_param, 12.00);
  SET new_card_id = LAST_INSERT_ID();
  IF EXISTS (SELECT 1 FROM buys WHERE customer_id = customer_id_param) THEN
    UPDATE buys SET multi_use_card_id = new_card_id WHERE customer_id = customer_id_param;
  ELSE
    INSERT INTO buys (customer_id, multi_use_card_id) VALUES (customer_id_param, new_card_id);
  END IF;
END$$
DELIMITER ;

Call IssueMultiUseCard(3);
Call IssueMultiUseCard(10);
Call IssueMultiUseCard(7);
```

After ("multi_use_card" table on left, "buys" table on right):

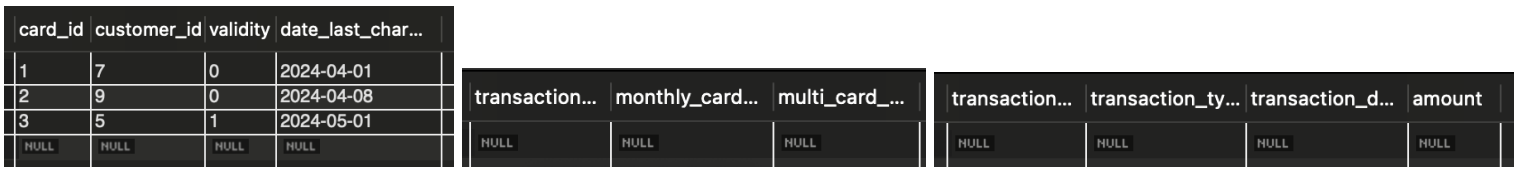


card_id	customer_id	balance
1	3	12
2	10	12
3	7	12
NULL	NULL	NULL

customer_id	monthly_card...	multi_use_card...
3	NULL	1
5	3	NULL
7	1	3
9	2	NULL
10	NULL	2

4. **Reload a monthly metrocard**

Before ("monthly_card" table on left, "makes" table in middle, "transactions" table on right):



card_id	customer_id	validity	date_last_char...
1	7	0	2024-04-01
2	9	0	2024-04-08
3	5	1	2024-05-01
NULL	NULL	NULL	NULL

transaction...	monthly_card...	multi_card_...
NULL	NULL	NULL

transaction...	transaction_ty...	transaction_d...	amount
NULL	NULL	NULL	NULL

```
-- Query/procedure to reload a monthly metrocard
DELIMITER $$
DROP PROCEDURE IF EXISTS ReloadMonthlyCard;
CREATE PROCEDURE `ReloadMonthlyCard`(IN card_id_param INT)
BEGIN
  DECLARE customer_var INT;
  DECLARE price DECIMAL(10,2);
  DECLARE transaction_date DATE;
  DECLARE new_transaction_id INT;
  SET transaction_date = CURDATE();
  SELECT customer_id INTO customer_var FROM monthly_card WHERE card_id = card_id_param;
  CALL CalculateMonthlyPrice(customer_var, price);
  UPDATE monthly_card
  SET validity = TRUE, date_last_charge = transaction_date
  WHERE card_id = card_id_param;
  INSERT INTO transactions(transaction_type, transaction_date, amount)
  VALUES ('Reload Monthly', transaction_date, price);
  SET new_transaction_id = LAST_INSERT_ID();
  INSERT INTO makes(transaction_id, monthly_card_id)
  VALUES (new_transaction_id, card_id_param);
END$$
DELIMITER ;

Call ReloadMonthlyCard(3);
Call ReloadMonthlyCard(1);
Call ReloadMonthlyCard(2);
```

After (“monthly_card” table on left, “makes” table in middle, “transactions” table on right):

card_id	customer_id	validity	date_last_char...
1	7	1	2024-05-12
2	9	1	2024-05-12
3	5	1	2024-05-12
NULL	NULL	NULL	NULL

transaction...	monthly_card...	multi_card_...
1	3	NULL
2	1	NULL
3	2	NULL
NULL	NULL	NULL

transaction...	transaction_ty...	transaction_d...	amount
1	Reload Monthly	2024-05-13	7
2	Reload Monthly	2024-05-13	7
3	Reload Monthly	2024-05-13	20
NULL	NULL	NULL	NULL

5. **Reload a multi-use metrocard**

Before (“multi_use_card” table on left, “makes” table in middle, “transactions table on right):

card_id	customer_id	balance
1	3	12
2	10	12
3	7	12
NULL	NULL	NULL

transaction...	monthly_card...	multi_card_...
1	3	NULL
2	1	NULL
3	2	NULL
NULL	NULL	NULL

transaction...	transaction_ty...	transaction_d...	amount
1	Reload Monthly	2024-05-13	7
2	Reload Monthly	2024-05-13	7
3	Reload Monthly	2024-05-13	20
NULL	NULL	NULL	NULL

```
-- Query/procedure to reload a multi-use metrocard
DROP PROCEDURE IF EXISTS ReloadMultiUseCard;
DELIMITER $$
CREATE PROCEDURE `ReloadMultiUseCard`(IN card_id_param INT, IN amount DECIMAL(10,2))
BEGIN
  DECLARE current_balance DECIMAL(10,2);
  DECLARE new_balance DECIMAL(10,2);
  DECLARE new_transaction_id INT;
  DECLARE transaction_date DATE;
  SET transaction_date = CURDATE();
  IF amount < 1.7 THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The minimum reload amount is 1.7 euros.';
  END IF;
  SELECT balance INTO current_balance
  FROM multi_use_card
  WHERE card_id = card_id_param;
  SET new_balance = current_balance + amount;
  UPDATE multi_use_card
  SET balance = new_balance
  WHERE card_id = card_id_param;
          INSERT INTO transactions(transaction_type, amount, transaction_date)
          VALUES('Reload Multi', amount, CURDATE());
  SET new_transaction_id = LAST_INSERT_ID();
  INSERT INTO makes(transaction_id, multi_card_id)
  VALUES(new_transaction_id, card_id_param);
END$$
DELIMITER ;

Call ReloadMultiUseCard(3, 10.25);
Call ReloadMultiUseCard(1, 15.50);
Call ReloadMultiUseCard(2, 20.75);
```

After (“multi_use_card” table on left, “makes” table in middle, “transactions” table on right):

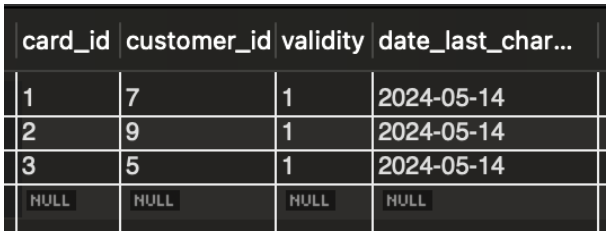
card_id	customer_id	balance
1	3	27.5
2	10	32.75
3	7	22.25
NULL	NULL	NULL

transaction...	monthly_card...	multi_card_...
1	3	NULL
2	1	NULL
3	2	NULL
4	NULL	3
5	NULL	1
6	NULL	2
NULL	NULL	NULL

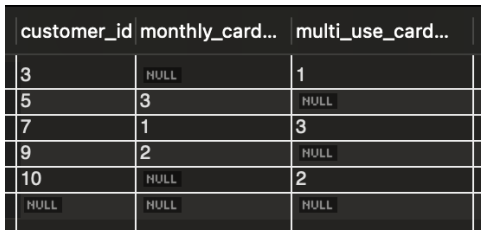
transaction...	transaction_ty...	transaction_d...	amount
1	Reload Monthly	2024-05-13	7
2	Reload Monthly	2024-05-13	7
3	Reload Monthly	2024-05-13	20
4	Reload Multi	2024-05-13	10.25
5	Reload Multi	2024-05-13	15.5
6	Reload Multi	2024-05-13	20.75
NULL	NULL	NULL	NULL

6. **Replace a missing monthly metrocard**

Before (“monthly_card” table on left, “buys” table on right):



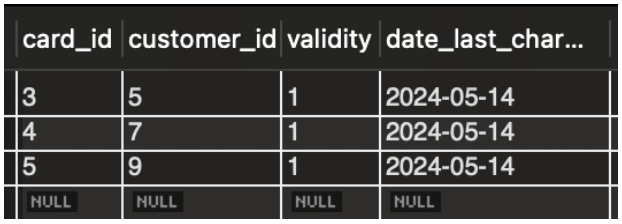
card_id	customer_id	validity	date_last_char...
1	7	1	2024-05-14
2	9	1	2024-05-14
3	5	1	2024-05-14
NULL	NULL	NULL	NULL



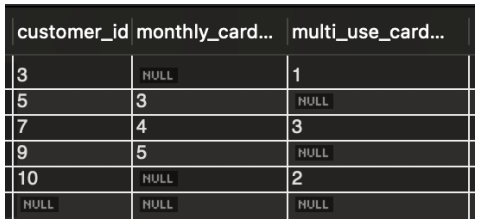
customer_id	monthly_card...	multi_use_card...
3	NULL	1
5	3	NULL
7	1	3
9	2	NULL
10	NULL	2
NULL	NULL	NULL

```
-- Query/procedure to replace a monthly metrocard
DROP PROCEDURE IF EXISTS ReplaceMonthlyCard;
DELIMITER $$
CREATE PROCEDURE `ReplaceMonthlyCard`(IN customer_id_param INT)
BEGIN
  DECLARE old_card_id INT;
  SELECT card_id INTO old_card_id FROM monthly_card WHERE customer_id = customer_id_param LIMIT 1;
  IF old_card_id IS NOT NULL THEN
    DELETE FROM makes WHERE monthly_card_id = old_card_id;
    UPDATE buys SET monthly_card_id = NULL WHERE customer_id = customer_id_param;
    DELETE FROM monthly_card WHERE card_id = old_card_id;
    CALL IssueMonthlyCard(customer_id_param);
    SELECT 'Monthly card replaced successfully.' AS Message;
  ELSE
    SELECT 'No monthly card found for this customer.' AS Message;
  END IF;
END$$
DELIMITER ;

Call ReplaceMonthlyCard(7);
Call ReplaceMonthlyCard(9);
```

After (“monthly_card” table on left, “buys” table on right):



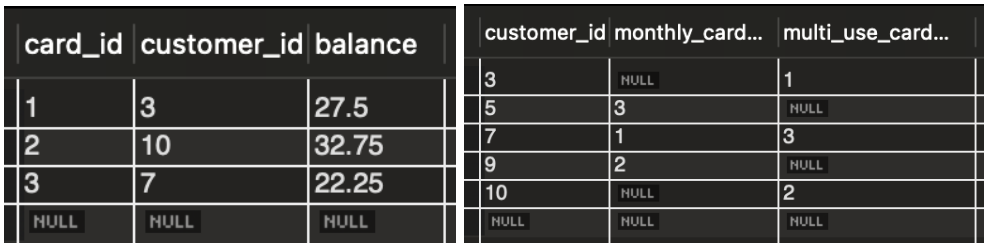
card_id	customer_id	validity	date_last_char...
3	5	1	2024-05-14
4	7	1	2024-05-14
5	9	1	2024-05-14
NULL	NULL	NULL	NULL



customer_id	monthly_card...	multi_use_card...
3	NULL	1
5	3	NULL
7	4	3
9	5	NULL
10	NULL	2
NULL	NULL	NULL

7. **Replace a missing multi-use metrocard:**

Before (“multi_use_card” table on left, “buys” table on right):



card_id	customer_id	balance
1	3	27.5
2	10	32.75
3	7	22.25
NULL	NULL	NULL

customer_id	monthly_card...	multi_use_card...
3	NULL	1
5	3	NULL
7	1	3
9	2	NULL
10	NULL	2
NULL	NULL	NULL

```
-- Query/procedure to replace a multi-use metrocard
DROP PROCEDURE IF EXISTS ReplaceMultiUseCard;
DELIMITER $$
CREATE PROCEDURE `ReplaceMultiUseCard`(IN customer_id_param INT)
BEGIN
  DECLARE old_card_id INT;
  DECLARE old_balance FLOAT;
  DECLARE new_card_id INT;
  SELECT card_id, balance INTO old_card_id, old_balance FROM multi_use_card
  WHERE customer_id = customer_id_param ORDER BY card_id DESC LIMIT 1;
    IF old_card_id IS NOT NULL THEN
    DELETE FROM makes WHERE multi_card_id = old_card_id;
    DELETE FROM buys WHERE multi_use_card_id = old_card_id;
    DELETE FROM multi_use_card WHERE card_id = old_card_id;
    INSERT INTO multi_use_card (customer_id, balance)
    VALUES (customer_id_param, old_balance);
    SET new_card_id = LAST_INSERT_ID();
    INSERT INTO buys (customer_id, multi_use_card_id)
    VALUES (customer_id_param, new_card_id);
    SELECT 'Multi-use card replaced successfully, balance transferred.' AS Message;
  ELSE
    SELECT 'No multi-use card found for this customer.' AS Message;
  END IF;
END$$
DELIMITER ;

Call ReplaceMultiUseCard(10);
Call ReplaceMultiUseCard(3);
```

After (“multi_use_card” table on left, “buys table on right):

card_id	customer_id	balance
3	7	22.25
4	10	32.75
5	3	27.5
NULL	NULL	NULL

customer_id	monthly_card...	multi_use_card...
3	NULL	5
5	3	NULL
7	1	3
9	2	NULL
10	NULL	4
NULL	NULL	NULL

8. Cancel customer registration

Before:

"customer" table

customer_id	full_name	date_of_bir...	age_group	zip_code	email	phone_number	student_stat...	disability_stat...	family_normal_stat...	family_special_stat.
1	John Doe	1985-05-15	Youth/Student	28001	john.doe@email.com	555-1234	1	0	1	0
2	Jane Smith	1998-07-22	Youth/Student	28100	jane.smith@email.com	555-5678	0	0	0	1
3	Carlos Ruiz	2004-02-01	Youth/Student	28200	carlos.ruiz@email.com	555-8765	1	1	0	1
4	Aisha Lee	1947-11-30	Senior	28300	aisha.lee@email.com	555-4321	0	0	0	0
5	Xavier Hernández	1990-03-15	Youth/Student	28205	xavier.hernandez@email.com	555-1122	1	1	0	0
6	Liu Wei	2001-12-12	Youth/Student	28105	liu.wei@email.com	555-3344	0	0	1	0
7	Sara Alba	1955-06-09	Youth/Student	28301	sara.alba@email.com	555-7788	1	1	0	0
8	Mohamed Al-Fayed	1979-04-08	Normal	28107	mohamed.alfayed@email.com	555-9900	0	0	0	1
9	Chloe Darcy	2003-08-19	Youth/Student	28400	chloe.darcy@email.com	555-2211	1	0	0	0
10	Ivan Petrov	1968-01-25	Normal	28325	ivan.petrov@email.com	555-5656	0	1	1	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

"buys" table

customer_id	monthly_card...	multi_use_card...
3	NULL	1
5	3	NULL
7	1	3
9	2	NULL
10	NULL	2
NULL	NULL	NULL

"monthly_card" table

card_id	customer_id	validity	date_last_char...
1	7	1	2024-05-14
2	9	1	2024-05-14
3	5	1	2024-05-14
NULL	NULL	NULL	NULL

"multi_use_card" table

card_id	customer_id	balance
1	3	27.5
2	10	12
3	7	22.25
NULL	NULL	NULL

"makes" table

transaction...	monthly_card...	multi_card_...
1	3	NULL
2	1	NULL
3	2	NULL
4	NULL	3
5	NULL	1
NULL	NULL	NULL

"transactions" table

transaction...	transaction_ty...	transaction_d...	amount
1	Reload Monthly	2024-05-14	7
2	Reload Monthly	2024-05-14	7
3	Reload Monthly	2024-05-14	20
4	Reload Multi	2024-05-14	10.25
5	Reload Multi	2024-05-14	15.5
NULL	NULL	NULL	NULL

"receives" table

customer_id	type_of_disco...
3	Disability
5	Disability
7	Disability
10	Disability
1	Family Normal
6	Family Normal
10	Family Normal
2	Family Special
3	Family Special
8	Family Special
NULL	NULL

```
-- Query/procedure to remove a customer from the database
DROP PROCEDURE IF EXISTS RemoveCustomer;
DELIMITER $$
CREATE PROCEDURE `RemoveCustomer`(IN customer_id_param INT)
BEGIN
  CREATE TEMPORARY TABLE IF NOT EXISTS temp_transactions (
    transaction_id INT
  );
  INSERT INTO temp_transactions (transaction_id)
  SELECT t.transaction_id
  FROM transactions t
  JOIN makes m ON t.transaction_id = m.transaction_id
  WHERE m.monthly_card_id IN (SELECT mc.card_id FROM monthly_card mc WHERE mc.customer_id =
customer_id_param)
    OR m.multi_card_id IN (SELECT muc.card_id FROM multi_use_card muc WHERE muc.customer_id =
customer_id_param);
  DELETE FROM makes WHERE transaction_id IN (SELECT transaction_id FROM temp_transactions);
  DELETE FROM transactions WHERE transaction_id IN (SELECT transaction_id FROM temp_transactions);
```

```
    DROP TEMPORARY TABLE IF EXISTS temp_transactions;
    DELETE FROM buys WHERE customer_id = customer_id_param;
    DELETE FROM monthly_card WHERE customer_id = customer_id_param;
    DELETE FROM multi_use_card WHERE customer_id = customer_id_param;
    DELETE FROM receives WHERE customer_id = customer_id_param;
    DELETE FROM customer WHERE customer_id = customer_id_param;
END$$
DELIMITER ;

Call RemoveCustomer(1);
Call RemoveCustomer(3);
Call RemoveCustomer(5);
Call RemoveCustomer(7);
```

After:

"customer" table

customer_id	full_name	date_of_bir...	age_group	zip_code	email	phone_number	student_stat...	disability_stat...	family_normal_stat...	family_special_stat...
2	Jane Smith	1998-07-22	Youth/Student	28100	jane.smith@email.com	555-5678	0	0	0	1
4	Aisha Lee	1947-11-30	Senior	28300	aisha.lee@email.com	555-4321	0	0	0	0
6	Liu Wei	2001-12-12	Youth/Student	28105	liu.wei@email.com	555-3344	0	0	1	0
8	Mohamed Al-Fayed	1979-04-08	Normal	28107	mohamed.alfayed@email.com	555-9900	0	0	0	1
9	Chloe Darcy	2003-08-19	Youth/Student	28400	chloe.darcy@email.com	555-2211	1	0	0	0
10	Ivan Petrov	1968-01-25	Normal	28325	ivan.petrov@email.com	555-5656	0	1	1	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

"buys" table

customer_id	monthly_card...	multi_use_card...
9	2	NULL
10	NULL	2
NULL	NULL	NULL

"monthly_card" table

card_id	customer_id	validity	date_last_char...
2	9	1	2024-05-14
NULL	NULL	NULL	NULL

"multi_use_card" table

card_id	customer_id	balance
2	10	32.75
NULL	NULL	NULL

"makes" table

transaction...	monthly_card...	multi_card_...
3	2	NULL
6	NULL	2
NULL	NULL	NULL

"transactions" table

transaction...	transaction_ty...	transaction_d...	amount
3	Reload Monthly	2024-05-14	20
6	Reload Multi	2024-05-14	20.75
NULL	NULL	NULL	NULL

"receives" table

customer_id	type_of_disco...
10	Disability
6	Family Normal
10	Family Normal
2	Family Special
8	Family Special
NULL	NULL

Triggers:

1. **Use triggers to check card charging date and change monthly card state to “0” if at least 30 days from the last charge.**

Before Trigger (“monthly_card” table):

card_id	customer_id	validity	date_last_char...
1	1	1	2024-04-13
2	2	1	2024-04-24
NULL	NULL	NULL	NULL

After Trigger (“monthly_card” table):

card_id	customer_id	validity	date_last_char...
1	1	0	2024-04-13
2	2	1	2024-04-24
NULL	NULL	NULL	NULL

```
DELIMITER $$
CREATE TRIGGER check_card_validity_insert
BEFORE INSERT ON monthly_card
FOR EACH ROW
BEGIN
  IF DATEDIFF(CURDATE(), NEW.date_last_charge) > 30 THEN
    SET NEW.validity = FALSE;
  ELSE
    SET NEW.validity = TRUE;
  END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER check_card_validity_update
BEFORE update ON monthly_card
FOR EACH ROW
BEGIN
  IF DATEDIFF(CURDATE(), NEW.date_last_charge) > 30 THEN
    SET NEW.validity = FALSE;
  ELSE
    SET NEW.validity = TRUE;
  END IF;
END$$
DELIMITER ;
```

2. **Use triggers to check the date of the client birthdate and current date, once the client age status changed the card cost should be updated automatically to the new price that suits that age.**

Before Trigger (first 4 columns of the “customer” table):

customer_id	full_name	date_of_bir...	age_group
1	Alice Johnson	1998-03-22	Youth/Student
2	Bob Smith	1957-08-12	Normal
3	Charlie Brown	1996-01-01	Youth/Student
4	Diana Prince	1940-05-25	Normal
5	Edward Cullen	1995-11-15	Youth/Student
NULL	NULL	NULL	NULL

After Trigger (first 4 columns of the “customer” table):

customer_id	full_name	date_of_bir...	age_group
1	Alice Johnson	1998-03-22	Normal
2	Bob Smith	1957-08-12	Senior
3	Charlie Brown	1996-01-01	Normal
4	Diana Prince	1940-05-25	Senior
5	Edward Cullen	1995-11-15	Normal
NULL	NULL	NULL	NULL

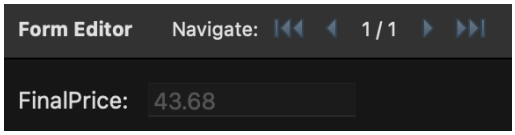
```
DELIMITER $$
CREATE TRIGGER age_group_insert
BEFORE INSERT ON customer
FOR EACH ROW
BEGIN
  DECLARE new_age INT;
  SET new_age = TIMESTAMPDIFF(YEAR, NEW.date_of_birth, CURDATE());
  IF new_age < 26 OR NEW.student_status = TRUE THEN
    SET NEW.age_group = 'Youth/Student';
  ELSEIF new_age >= 65 THEN
    SET NEW.age_group = 'Senior';
  ELSE
    SET NEW.age_group = 'Normal';
  END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER age_group_update
AFTER UPDATE ON customer
FOR EACH ROW
BEGIN
  DECLARE new_age INT;
  SET new_age = TIMESTAMPDIFF(YEAR, NEW.date_of_birth, CURDATE());
  IF new_age < 26 OR NEW.student_status = TRUE THEN
    SET NEW.age_group = 'Youth/Student';
```

```
  ELSEIF new_age >= 65 THEN
    SET NEW.age_group = 'Senior';
  ELSE
    SET NEW.age_group = 'Normal';
  END IF;
END$$
DELIMITER ;
```

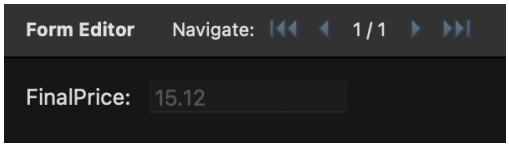
3. **Use a procedure to determine the card fee according to the city (zone) when the client registers or updates his address.**

```
INSERT INTO customer
(full_name, date_of_birth, zip_code, email, phone_number, student_status, disability_status, family_normal_status, family_special_status)
VALUES
('John Doe', '1997-05-15', 28001, 'john.doe@email.com', '555-1234', FALSE, FALSE, TRUE, FALSE),
('Xavier Hernández', '1989-03-15', 28205, 'xavier.hernandez@email.com', '555-1122', FALSE, TRUE, FALSE, TRUE);
```

```
Call CalculateMonthlyPrice(1);
```

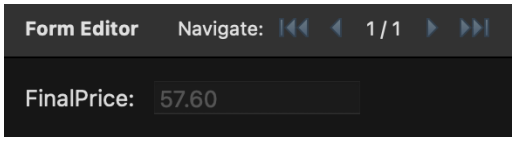



```
Call CalculateMonthlyPrice(2);
```




```
UPDATE customer SET zip_code = 28200 WHERE customer_id = 1;
UPDATE customer SET zip_code = 28300 WHERE customer_id = 2;
```

```
Call CalculateMonthlyPrice(1);
```




```
Call CalculateMonthlyPrice(2);
```