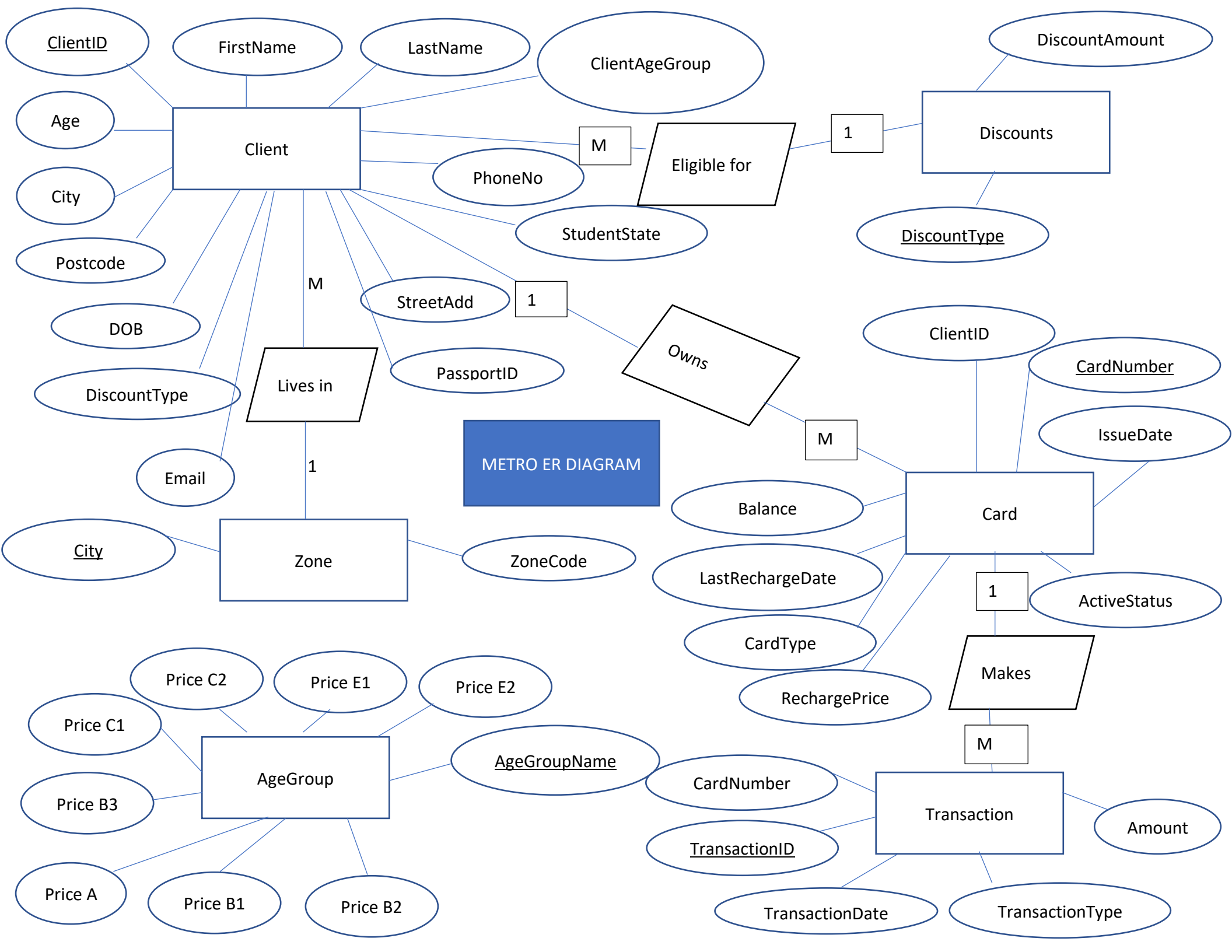
ClientID
FirstName
LastName
ClientAgeGroup
DiscountAmount
Age
Client
M
Eligible for
1
Discounts
City
PhoneNo
StudentState
DiscountType
Postcode
M
StreetAdd
1
DOB
Owns
ClientID
CardNumber
Lives in
PassportID
DiscountType
METRO ER DIAGRAM
M
IssueDate
Email
1
Balance
Card
City
Zone
ZoneCode
LastRechargeDate
1
ActiveStatus
CardType
Makes
RechargePrice
Price C2
Price E1
Price E2
Price C1
AgeGroup
AgeGroupName
M
Price B3
CardNumber
Transaction
Amount
TransactionID
Price A
Price B1
Price B2
TransactionDate
TransactionType

RELATIONAL SCHEMA

CLIENT[1]

ClientID	FirstName	LastName	Email	Age	PhoneNo	DOB	Postcode	StreetAddress	City
StudentState	DiscountType	PassportID	ClientAgeGroup						

CARD

CardNumber	ClientID	CardType	ActiveStatus	Balance	IssueDate	LastRechargeDate	RechargePrice

Transaction

TransactionID	CardNumber	TransactionType	Amount	TransactionDate

ZONE

City	ZoneCode

AGEGROUP

AgeGroupName	PriceA	PriceB1	PriceB2	PriceB3	PriceC1	PriceC2	PriceE1	PriceE2

DISCOUNTS

DiscountType	DiscountAmount

[1] Used two rows here since all the fields could not fit on one

METADATA

Table: Client

- ClientID: Unique ID of client (INT, PRIMARY KEY)
- FirstName: First name of client (VARCHAR(100))
- LastName: Last name of client (VARCHAR(100))
- Email: Email address of client (VARCHAR(100))
- PhoneNumber: Phone number of client (VARCHAR(20))
- City: City where client resides (VARCHAR(20), FOREIGN KEY)
- Postcode: Postal code of client's address (INT)
- StreetAddress: Street address of client (VARCHAR(30))
- DOB: Date of birth of client (DATE)
- ClientAgeGroup: Age group of client (VARCHAR(30))
- StudentState: Whether client is a student (BOOLEAN)
- DiscountType: Type of discount applicable to client (VARCHAR(30), FOREIGN KEY)
- PassportID: Passport ID of client (VARCHAR(30))

Table: Card

- CardNumber: Unique card number (INT, PRIMARY KEY)
- ClientID: ID of the client associated with the card (INT, FOREIGN KEY)
- CardType: Type of card (VARCHAR(50))
- ActiveStatus: Whether the card is active (BOOLEAN)
- Balance: Current balance of the card (DECIMAL(10, 2))
- IssueDate: Date the card was issued (DATE)
- LastRechargeDate: Date the card was last recharged (DATE)
- RechargePrice: Price for recharging the card (DECIMAL(10, 2))

Table: Discounts

- DiscountType: Type of discount (DECIMAL(10, 2), PRIMARY KEY)
- DiscountAmount: Amount of discount (INT)

Table: AgeGroup

- AgeGroupName: Name of the age group (VARCHAR(20), PRIMARY KEY)
- PriceA: Price category A (DECIMAL(10, 2))
- PriceB1: Price category B1 (DECIMAL(10, 2))
- PriceB2: Price category B2 (DECIMAL(10, 2))
- PriceB3: Price category B3 (DECIMAL(10, 2))
- PriceC1: Price category C1 (DECIMAL(10, 2))
- PriceC2: Price category C2 (DECIMAL(10, 2))
- PriceE1: Price category E1 (DECIMAL(10, 2))
- PriceE2: Price category E2 (DECIMAL(10, 2))

Table: Transaction

- TransactionID: Unique transaction ID (INT, PRIMARY KEY, AUTO_INCREMENT)
- CardNumber: Card number associated with the transaction (INT, FOREIGN KEY)
- TransactionType: Type of transaction (VARCHAR(50))
- Amount: Amount involved in the transaction (DECIMAL(10, 2))
- TransactionDate: Date of the transaction (DATE)

Table: Zone

- City: Name of the city (VARCHAR(50), PRIMARY KEY)
- ZoneCode: Code of the zone (VARCHAR(20))

Database Management System

Madrid Metro

Course Project – Spring 2024

Kabir Pamnani

SECTION 1: SQL Code for Creating the MadridMetro DB

MadridMetro Schema

```
1  create SCHEMA MadridMetro;
```

Client Table

```
create TABLE MadridMetro.Client (
    ClientID INT PRIMARY KEY,
    FirstName VARCHAR(100),
    LastName VARCHAR(100),
    Email VARCHAR(100),
    PhoneNumber VARCHAR(20),
    City VARCHAR(20),
    Postcode INT,
    StreetAddress VARCHAR(30),
    DOB DATE,
    ClientAgeGroup VARCHAR(30),
    StudentState BOOLEAN,
    DiscountType VARCHAR(30),
    PassportID VARCHAR(30),
    FOREIGN KEY (City) REFERENCES MadridMetro.Zone(City),
    FOREIGN KEY (DiscountType) REFERENCES MadridMetro.Discounts(DiscountType)
);
```

Card Table

```
CREATE TABLE MadridMetro.Card (
    CardNumber INT PRIMARY KEY,
    ClientID INT,
    CardType VARCHAR(50),
    ActiveStatus BOOLEAN,
    Balance DECIMAL(10, 2),
    IssueDate DATE,
    LastRechargeDate DATE,
    RechargePrice DECIMAL (10, 2),
    FOREIGN KEY (ClientID) REFERENCES MadridMetro.Client(ClientID)
);
```

Discounts Table

```
76  CREATE TABLE MadridMetro.Discounts (
77      DiscountType DECIMAL(10,2) PRIMARY KEY,
78      DiscountAmount INT
79  );
```

AgeGroup Table

```
91  CREATE TABLE MadridMetro.AgeGroup (
92      AgeGroupName VARCHAR(20) PRIMARY KEY, -- 'Abono Joven', 'Normal', 'Senior'
93      PriceA DECIMAL(10, 2),
94      PriceB1 DECIMAL(10, 2),
95      PriceB2 DECIMAL(10, 2),
96      PriceB3 DECIMAL(10, 2),
97      PriceC1 DECIMAL(10, 2),
98      PriceC2 DECIMAL(10, 2),
99      PriceE1 DECIMAL(10, 2),
100     PriceE2 DECIMAL(10, 2)
101 );
```

Transaction Table

```
104 CREATE TABLE MadridMetro.Transaction (
105     TransactionID INT PRIMARY KEY AUTO_INCREMENT,
106     CardNumber INT,
107     TransactionType VARCHAR(50),
108     Amount DECIMAL(10, 2),
109     TransactionDate DATE,
110     FOREIGN KEY (CardNumber) REFERENCES MadridMetro.Card(CardNumber)
111 );
```

Zone Table

```
235 CREATE TABLE MadridMetro.Zone (
236     City VARCHAR(50) PRIMARY KEY,
237     ZoneCode VARCHAR (20)
238
239 );
```

Code for populating Discounts table

```
INSERT INTO MadridMetro.Discounts (DiscountType, DiscountAmount) VALUES
('Big Family Normal', 0.2),
('Big Family Special', 0.4),
('Disability', 0.65),
('Senior', 0.65);
```

Figure 1: Discount Table populated with available discounts

DiscountType	DiscountAmount
Big Family Normal	0.20
Big Family Special	0.40
Disability	0.65
Senior	0.65

Code for populating AgeGroup table

```
INSERT INTO MadridMetro.AgeGroup (AgeGroupName, PriceA, PriceB1, PriceB2, PriceB3, PriceC1, PriceC2, PriceE1, PriceE2) VALUES
('Normal', 54.60, 63.70, 72.00, 82.00, 89.50, 99.30, 110.60, 131.80),
('Abono Joven', 20.00, 20.00, 20.00, 20.00, 20.00, 20.00, 20.00, 20.00),
('Senior', 6.30, 6.30, 6.30, 6.30, 6.30, 6.30, NULL, NULL);
```

Figure 2: AgeGroup Table populated with Age Groups and associated zone costs

AgeGroupName	PriceA	PriceB1	PriceB2	PriceB3	PriceC1	PriceC2	PriceE1	PriceE2
Abono Joven	20.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00
Normal	54.60	63.70	72.00	82.00	89.50	99.30	110.60	131.80
Senior	6.30	6.30	6.30	6.30	6.30	6.30	NULL	NULL

Code for populating Zone table (Only included some, but this continues for all given cities)

```
-- Insert data for Zone A
INSERT INTO Zones (City, ZoneCode) VALUES ('Madrid centre', 'A');

-- Insert data for Zone B1
INSERT INTO Zones (City, ZoneCode) VALUES ('Alcobendas', 'B1');
INSERT INTO Zones (City, ZoneCode) VALUES ('Alcorcón', 'B1');
INSERT INTO Zones (City, ZoneCode) VALUES ('Cantoblanco', 'B1');
INSERT INTO Zones (City, ZoneCode) VALUES ('Coslada', 'B1');
INSERT INTO Zones (City, ZoneCode) VALUES ('Facultad de Informática', 'B1');
INSERT INTO Zones (City, ZoneCode) VALUES ('Getafe', 'B1');
INSERT INTO Zones (City, ZoneCode) VALUES ('Leganés', 'B1');
```

Figure 3: Zone Table populated with each city and associated zone code

City	ZoneCode
Madrid centre	A
Alcobendas	B1
Alcorcón	B1
Cantoblanco	B1
Coslada	B1
Facultad de Infor…	B1
Getafe	B1
Leganés	B1
Paracuellos del J…	B1
Pozuelo de Alarcón	B1
Rivas Vaciamadrid	B1
San Fernando d…	B1
San Sebastián d…	B1
Ajalvir	B2
Belvis y Los Berr…	B2
Boadilla del Monte	B2
Fuenlabrada	B2
Fuente del Fresn…	B2
Las Matas	B2
Las Rozas de M…	B2
Majadahonda	B2
Mejorada del Ca…	B2
Móstoles	B2
Parla	B2
Pinto	B2
Torrejón de Ardoz	B2
Tres Cantos	B2
Velilla de San An…	B2
Villaviciosa de O…	B2
Alcalá de Henares	B3
Algete	B3
Arganda	B3
Arroyomolinos	B3
Brunete	B3
Ciempozuelos	B3
Ciudalcampo	B3
Cobeña	B3
Collado Villalba	B3
Colmenar Viejo	B3
Colmenarejo	B3
Daganzo de Arriba	B3
Galapagar	B3
Griñón	B3
Hoyo de Manzan…	B3
Humanes de Ma…	B3
Loeches	B3
Moraleja de Enm…	B3
Navalcarnero	B3
San Agustín de…	B3
San Martín de la…	B3
Torrejón de la Ca…	B3
Torrejón de Vela…	B3
Torrelodones	B3
Valdemoro	B3
Villanueva de la…	B3
Villanueva del P…	B3
Alpedrete	C1
Anchuelo	C1
Aranjuez	C1
Batres	C1
Becerril de la Sie…	C1
Camarma de Est…	C1
Campo Real	C1
Casarrubuelos	C1
Chinchón	C1
Collado-Mediano	C1
Cubas de la Sagra	C1
El Álamo	C1
El Boalo y entida…	C1
El Escorial	C1
El Molar	C1
Fresno de Torote	C1
Fuente el Saz de…	C1
Guadarrama	C1
Los Santos de la…	C1
Manzanares El…	C1
Meco	C1
Moralzarzal	C1
Morata de Tajuña	C1
Pedrezuela	C1
Perales de Tajuña	C1
Pozuelo del Rey	C1
Quijorna	C1
Ribatejada	C1
San Lorenzo de…	C1
Serranillos del V…	C1
Sevilla la Nueva	C1
Soto del Real	C1
Titulcia	C1
Torres de la Ala…	C1
Valdeavero	C1
Valdemorillo	C1
Valdeolmos-Alal…	C1
Valdetorres de J…	C1
Valverde de Alcalá	C1
Villaconejos	C1
Villalbilla	C1

SECTION 2: SQL Code for Queries

Query 1: Register a new client[1]

```
INSERT INTO MadridMetro.Client (ClientID, FirstName, LastName, Email, PhoneNumber, City, Postcode, StreetAddress, DOB, StudentState, DiscountType) VALUES
(1, 'Kabir', 'Pamnani', 'kabir.pamnani@gmail.com', '655277718', 'Madrid Centre', 28020, 'Calle de Isturiz', '2001-06-16', TRUE, 'Big Family Normal'),
(2, 'Shahmeer', 'Madni', 'shahmeer.madni@gmail.com', '612345869', 'Leganes', 28912, 'Pz Fuente de Honda 8', '1985-01-19', FALSE, 'Disability'),
(3, 'Jonas', 'Lund', 'jonas.lund@gmail.com', '123893231', 'Anchuelo', 28818, 'Pz Matur 1', '1960-10-10', FALSE, 'Senior');
```

Figure 4: Client Registration for 3 different age groups

ClientID	FirstName	LastName	Email	PhoneNumber	City	Postcode	StreetAddress	DOB	S...	DiscountType	ClientAgeGroup
1	Kabir	Pamnani	kabir.pamnani@gmail.com	655277718	Madrid Centre	28020	Calle de Isturiz	2001-06-16	1	Big Family Normal	Abono Joven
2	Shahmeer	Madni	shahmeer.madni@gmail.com	612345869	Leganes	28912	Pz Fuente de Honda 8	1985-01-19	0	Disability	Normal
3	Jonas	Lund	jonas.lund@gmail.com	123893231	Anchuelo	28818	Pz Matur 1	1950-10-10	0	Senior	Senior

Query 2: Registering a new Card

2a) Registering a ‘Monthly’ card for the first time

```
INSERT INTO MadridMetro.Card (CardNumber, ClientID, CardType, ActiveStatus, Balance, IssueDate) VALUES
(10001, 1, 'Monthly', FALSE, NULL, CURDATE());
```

In the code snippet, I show how a first-time registration of a monthly card (which has not yet been topped up) will be recorded in the Card DB. The ClientID is linked to the ID of the client whose card it is, the CardType is set to ‘Monthly’ to show that this is a personal monthly card as opposed to a Multi-Use card, and the ActiveStatus is set to False since it has no balance yet.

2b) Registering a ‘Multi-Use’ card for the first time

```
INSERT INTO MadridMetro.Card (CardNumber, ClientID, CardType, ActiveStatus, Balance, IssueDate, LastRechargeDate) VALUES
(10002, 2, 'Multi-Use', True, 12, CURDATE(), NULL);
```

In the code snippet, I show how a first-time registration of a Multi-use card is recorded. Since the first charge needs to be at least 12 euro, I have added a balance of 12 euro to the card, and changed its ActiveStatus = True.

[1] Note that the DB shown here accidentally excludes the passport ID field. This was something I realized later and so I am not able to amend all the screenshots I have taken. Do note however that I have amended the DB to include this field.

Since this counts as a transaction, the Transaction table will be updated to reflect the fact that a ‘recharge’ purchase was made to this multi-use card.

```
399    INSERT INTO MadridMetro.Transaction (CardNumber, TransactionType, Amount, TransactionDate)
400    VALUES (10002, 'Recharge Multi-Use', 12, CURDATE());
```

Figure 5: States of the Card and Transaction Tables after 2 card registrations

CardNumber	ClientID	CardType	ActiveStatus	Balance	IssueDate	LastRechargeDate	RechargePrice
10001	1	Monthly	0	NULL	2024-05-13	NULL	16.00
10002	2	Multi-Use	1	12.00	2024-05-13	NULL	22.30

16	10002	Recharge Multi-Use	12.00	2024-05-15

Query 3: Recharge the card

When a monthly card is registered for the first time, its ActiveStatus is set to False before any money has been added to it. When a Multi-Use card is registered for the first time, its ActiveStatus is set to False unless at least 12 Euros has been added to it.

3a) Recharging a monthly card

To recharge a monthly card, a user will enter the details of their transaction into the Transaction table. Only if the “Amount” they recharge their card by is >= their individual RechargePrice (this computation will be shown later in the triggers section), will their card be activated.

SQL Code for a monthly card recharge

```
402    INSERT INTO MadridMetro.Transaction (CardNumber, TransactionType, Amount, TransactionDate)
403    VALUES (10001, 'Recharge Monthly', 16, CURDATE());
```

Figure 6: Transaction table after recharge entry

TransactionID	CardNumber	TransactionType	Amount	TransactionDate
8	10001	Recharge Monthly	16.00	2024-05-13

As shown in the previous Card table (Figure 5), the recharge price for the client with card 10001 = 16 euros. As seen from this transaction (Figure 6), the amount the client recharged their card was also = 16 euros. Therefore, the card 10001 will get activated (because 16 >= 16).

Before recharging (ActiveStatus = 0)

CardNumber	ClientID	CardType	ActiveStatus	Balance	IssueDate	LastRechargeDate	RechargePrice
10001	1	Monthly	0	NULL	2024-05-13	NULL	16.00

After recharging (ActiveStatus = 1, LastRechargeDate = Date of transaction)

CardNumber	ClientID	CardType	ActiveStatus	Balance	IssueDate	LastRechargeDate	RechargePrice
10001	1	Monthly	1	NULL	2024-05-13	2024-05-13	16.00

The trigger to automatically activate the card, and to update the LastRechargeDate is shown here:

```
368     -- Trigger to activate Card
369     DELIMITER $$
370     CREATE TRIGGER AfterTransactionInsert
371     AFTER INSERT ON MadridMetro.Transaction
372     FOR EACH ROW
373     BEGIN
374         -- Declare variables at the beginning of the trigger
375         DECLARE rechargePrice DECIMAL(10, 2);
376
377         -- Check if the transaction is a 'Recharge Monthly'
378         IF NEW.TransactionType = 'Recharge Monthly' THEN
379
380             -- Retrieve the RechargePrice for the card involved in the transaction
381             SELECT RechargePrice INTO rechargePrice
382             FROM MadridMetro.Card
383             WHERE CardNumber = NEW.CardNumber;
384
385             -- Check if the amount is sufficient to recharge the card
386             IF NEW.Amount >= rechargePrice THEN
387                 -- Update the card's ActiveStatus and LastRechargeDate
388                 UPDATE MadridMetro.Card
389                 SET ActiveStatus = true,
390                     LastRechargeDate = NEW.TransactionDate
391                 WHERE CardNumber = NEW.CardNumber;
392             END IF;
393         END IF;
394     END$$
395     DELIMITER ;
```

3b) Recharging a multi-use card

For a multi-use card, there are certain conditions that need to be accounted for. Every usage will deduct 1.70 euros from the current balance. Once the value falls below 1.70, the card's ActiveStatus will change to False, and it will no longer be able to be used until the value rises above 1.7 again.

SQL Code Snippet: User pays for a single-fare transaction, 1.70 euros is automatically deducted from current balance of the linked card

Trigger to automatically deduct 1.70 euros

```
DELIMITER $$
CREATE TRIGGER AfterTransactionInsertForSingleFare
AFTER INSERT ON MadridMetro.Transaction
FOR EACH ROW
BEGIN
    -- Declare a variable to hold the current balance of the card
    DECLARE currentBalance DECIMAL(10, 2);
    -- Check if the transaction is 'Single-fare'
    IF NEW.TransactionType = 'Single-fare' THEN


        -- Retrieve the current balance for the card involved in the transaction
        SELECT Balance INTO currentBalance
        FROM MadridMetro.Card
        WHERE CardNumber = NEW.CardNumber;

        -- Check if the current balance is sufficient
        IF currentBalance >= 1.7 THEN
            -- Reduce the balance by 1.7
            UPDATE MadridMetro.Card
            SET Balance = currentBalance - 1.7
            WHERE CardNumber = NEW.CardNumber;
        END IF;
    END IF;
END$$
DELIMITER ;
```

User records a single-fare transaction in transaction table

```
INSERT INTO MadridMetro.Transaction (CardNumber, TransactionType, Amount, TransactionDate)
VALUES (10002, 'Single-Fare', 1.7, CURDATE());
```

TransactionID	CardNumber	TransactionType	Amount	TransactionDate
8	10001	Recharge Monthly	16.00	2024-05-13
9	10002	Single-Fare	1.70	2024-05-14

Before transaction recorded: Balance = 12 euros

10002	2	Multi-Use	1	12.00	2024-05-13	NULL	22.30

After trigger: Balance = 12 – 1.70 = 10.30 euros

CardNumber	ClientID	CardType	ActiveStatus	Balance	IssueDate	LastRechargeDate	RechargePrice
10001	1	Monthly	1	NULL	2024-05-13	2024-05-13	16.00
10002	2	Multi-Use	1	10.30	2024-05-13	NULL	22.30

Trigger to deactivate card when value falls < 1.7

```
475     DELIMITER $$
476     CREATE TRIGGER UpdateActiveStatusAfterBalanceChange
477     AFTER UPDATE ON MadridMetro.Card
478     FOR EACH ROW
479     BEGIN
480         -- Check if the balance is not NULL, is an integer, and has fallen below 1.7
481         IF NEW.Balance IS NOT NULL AND NEW.Balance < 1.7 AND NEW.Balance = FLOOR(NEW.Balance) THEN
482             -- Update the ActiveStatus to 0
483             UPDATE MadridMetro.Card
484             SET ActiveStatus = 0
485             WHERE CardNumber = NEW.CardNumber;
486         END IF;
487     END$$
488     DELIMITER ;
```

Query 4: Replace a missing card

To replace a missing card, a new card will be issued to the client (with same ClientID), and replace the previous, lost card. As seen below, the data from the old card is transferred completely to the old client since it is only the CardNumber that changes. So if the old card was previously active, the new one will be too. Since this overwrites the CardNumber of the old card, the old card effectively gets cancelled as a result.

SQL Code:

```
427     -- Overwrite the old CardNumber with the new one and change IssueDate
428     UPDATE MadridMetro.Card
429     SET CardNumber = 10004, IssueDate = curdate()
430     WHERE CardNumber = 10003;
```

Before new card issued to same client

CardNumber	ClientID	CardType	ActiveStatus	Balance	IssueDate	LastRechargeDate	RechargePrice
10001	1	Monthly	1	NULL	2024-05-13	2024-05-13	16.00
10002	2	Multi-Use	1	0.10	2024-05-13	NULL	22.30
10003	3	Monthly	1	NULL	2023-10-10	NULL	3.78

Old card number updated to new one and IssueDate changed to reflect current date

CardNumber	ClientID	CardType	ActiveStatus	Balance	IssueDate	LastRechargeDate	RechargePrice
10001	1	Monthly	1	NULL	2024-05-13	2024-05-13	16.00
10002	2	Multi-Use	1	0.10	2024-05-13	NULL	22.30
10004	3	Monthly	1	NULL	2024-05-14	NULL	3.78

Query 5: Cancel client registration

SQL Code to cancel a registered client:

Pre-Cancellation state of Client Table

ClientID	FirstName	LastName	Email	PhoneNumber	City	Postcode	StreetAddress	DOB	S...	DiscountType	ClientAg
1	Kabir	Pamnani	kabir.pamnani@gmail.com	655277718	Madrid Centre	28020	Calle de Isturiz	2001-06-16	1	Big Family Normal	Abono Jo
2	Shahmeer	Madni	shahmeer.madni@gmail.com	612345869	Leganes	28912	Pz Fuente de Honda 8	1985-01-19	0	Disability	Normal
3	Jonas	Lund	jonas.lund@gmail.com	123893231	Leganes	28818	Pz Matur 1	1950-10-10	0	Big Family Special	Senior

```
432     -- Remove a client by ClientID
433     DELETE FROM MadridMetro.Client
434     WHERE ClientID = 1;
```

This code deletes all corresponding card records of cancelled client:

Foreign key amended to include "ON DELETE CASCADE"

FOREIGN KEY (ClientID) REFERENCES MadridMetro.Client(ClientID) ON DELETE CASCADE

Post-Cancellation state of Client Table

ClientID	FirstName	LastName	Email	PhoneNumber	City	Postcode	StreetAddress	DOB	S...	DiscountType	ClientAge
2	Shahmeer	Madni	shahmeer.madni@gmail.com	612345869	Leganes	28912	Pz Fuente de Honda 8	1985-01-19	0	Disability	Normal
3	Jonas	Lund	jonas.lund@gmail.com	123893231	Leganes	28818	Pz Matur 1	1950-10-10	0	Big Family Special	Senior

SECTION 3: Triggers

Trigger 1: Change card state to “0” if it has been more than 30 days from the last charge

```
508     -- Procedure for Expired Card
509     DELIMITER $$
510     CREATE PROCEDURE UpdateCardStatusIfExpired()
511     BEGIN
512         UPDATE MadridMetro.Card
513         SET ActiveStatus = 0
514         WHERE LastRechargeDate <= CURDATE() - INTERVAL 30 DAY;
515     END$$
516     DELIMITER ;
517
518     -- Call Procedure to check which cards are expired
519     CALL UpdateCardStatusIfExpired();
```

Currently, this works by manually calling the procedure. An alternative method is to have a scheduled procedure that is automatically called every 24 hours.

The two tables below dictate, for CardNumber = 10006, what happens when this procedure is called and there is a record whose LastRechargeDate > 30 days (31 in this case):

Pre procedure call (ActiveStatus = 1)

CardNumber	ClientID	CardType	ActiveStatus	Balance	IssueDate	LastRechargeDate	RechargePrice
10001	1	Monthly	1	NULL	2024-05-13	2024-05-13	16.00
10002	2	Multi-Use	1	0.10	2024-05-13	NULL	22.30
10003	3	Monthly	1	NULL	2023-10-10	NULL	3.78
10006	4	Monthly	1	NULL	2023-05-14	2024-04-13	NULL

Post procedure call (ActiveStatus changed to 0)

CardNumber	ClientID	CardType	ActiveStatus	Balance	IssueDate	LastRechargeDate	RechargePrice
10001	1	Monthly	1	NULL	2024-05-13	2024-05-13	16.00
10002	2	Multi-Use	1	0.10	2024-05-13	NULL	22.30
10003	3	Monthly	1	NULL	2023-10-10	NULL	3.78
10006	4	Monthly	0	NULL	2023-05-14	2024-04-13	NULL

Trigger 2: Check date of client birthdate and current date, and automatically update to new price that suits that age

```
DELIMITER $$
CREATE TRIGGER MadridMetro.UpdateClientAgeGroup
BEFORE INSERT ON MadridMetro.Client
FOR EACH ROW
BEGIN
    SET NEW.ClientAgeGroup = CASE
        WHEN TIMESTAMPDIFF(YEAR, NEW.DOB, CURDATE()) < 26 OR NEW.StudentState = TRUE THEN 'Abono Joven'
        WHEN TIMESTAMPDIFF(YEAR, NEW.DOB, CURDATE()) >= 26 AND TIMESTAMPDIFF(YEAR, NEW.DOB, CURDATE()) < 65 THEN 'Normal'
        ELSE 'Senior'
    END;
END$$

CREATE TRIGGER MadridMetro.MadridMetroUpdateClientAgeGroupOnUpdate
BEFORE UPDATE ON MadridMetro.Client
FOR EACH ROW
BEGIN
    SET NEW.ClientAgeGroup = CASE
        WHEN TIMESTAMPDIFF(YEAR, NEW.DOB, CURDATE()) < 26 OR NEW.StudentState = TRUE THEN 'Abono Joven'
        WHEN TIMESTAMPDIFF(YEAR, NEW.DOB, CURDATE()) >= 26 AND TIMESTAMPDIFF(YEAR, NEW.DOB, CURDATE()) < 65 THEN 'Normal'
        ELSE 'Senior'
    END;
END$$
DELIMITER ;
```

As shown below, the ClientAgeGroup is computed automatically due to the above trigger, based on the DOB entered by client. The edge case of where a client is older than 26 years, but are still a student is shown in the ClientID = 4 record. In this case they are assigned to the ClientAgeGroup = “Abono Joven” since they are still eligible for that pricing.

ClientID	FirstName	LastName	Email	PhoneNumber	City	Postcode	StreetAddress	DOB	S...	DiscountType	ClientAgeGroup
1	Kabir	Pamnani	kabir.pamnani@gmail.com	655277718	Madrid Centre	28020	Calle de Isturiz	2001-06-16	1	Big Family Normal	Abono Joven
2	Shahmeer	Madni	shahmeer.madni@gmail.com	612345869	Leganes	28912	Pz Fuente de Honda 8	1985-01-19	0	Disability	Normal
3	Jonas	Lund	jonas.lund@gmail.com	123893231	Leganes	28818	Pz Matur 1	1950-10-10	0	Big Family Special	Senior
4	Pietro	Cannizzo	pietch.can@gmail.com	123456789	Leganes	28938	Leganes Renfe	1963-10-10	1	Senior	Abono Joven

The way that this impacts pricing will be shown in the next trigger.

Trigger 3: Determine card fee according to city (zone) when the client registers or updates address, according to the current age group of the client, and according to whatever discounts they are eligible for.

SQL Trigger code to calculate the client's personal card fee if it's a new card entry

```sql
-- Trigger to update card for new card entries
DELIMITER $$

CREATE TRIGGER CalculateRechargePrice
BEFORE INSERT ON MadridMetro.Card
FOR EACH ROW
BEGIN
    DECLARE zoneCode VARCHAR(20);
    DECLARE basePrice DECIMAL(10,2);
    DECLARE discountRate DECIMAL(10,2);
    DECLARE finalPrice DECIMAL(10,2);

    -- Retrieve the client's city and determine the zone code
    SELECT z.ZoneCode INTO zoneCode
    FROM MadridMetro.Zone z
    JOIN MadridMetro.Client c ON z.City = c.City
    WHERE c.ClientID = NEW.ClientID;

    -- Retrieve the client's age group and determine the base price
    SELECT CASE
        WHEN zoneCode = 'A' THEN ag.PriceA
        WHEN zoneCode = 'B1' THEN ag.PriceB1
        WHEN zoneCode = 'B2' THEN ag.PriceB2
        WHEN zoneCode = 'B3' THEN ag.PriceB3
        WHEN zoneCode = 'C1' THEN ag.PriceC1
        WHEN zoneCode = 'C2' THEN ag.PriceC2
        ELSE NULL
    END INTO basePrice
    FROM MadridMetro.AgeGroup ag
    JOIN MadridMetro.Client c ON ag.AgeGroupName = c.ClientAgeGroup
    WHERE c.ClientID = NEW.ClientID;
```

```sql
    -- Get the discount rate; if no discount type is set, assume 0
    SELECT COALESCE(d.DiscountAmount, 0) INTO discountRate
    FROM MadridMetro.Discounts d
    WHERE d.DiscountType = (SELECT c.DiscountType FROM MadridMetro.Client c WHERE c.ClientID = NEW.ClientID);
    -- Calculate the final recharge price
    SET finalPrice = basePrice * (1 - discountRate);
    -- Set the RechargePrice in the new card record
    SET NEW.RechargePrice = finalPrice;
END$$
-- Trigger to update card price for existing cards
DELIMITER $$
CREATE TRIGGER UpdateCardPriceAfterClientUpdate
AFTER UPDATE ON MadridMetro.Client
FOR EACH ROW
BEGIN
    -- Declare variables to hold dynamic data for price calculation
    DECLARE newZoneCode VARCHAR(20);
    DECLARE newBasePrice DECIMAL(10,2);
    DECLARE newDiscountRate DECIMAL(10,2);
    DECLARE newFinalPrice DECIMAL(10,2);
    DECLARE done INT DEFAULT FALSE; -- Handler flag
    DECLARE card_number INT; -- To hold CardNumber in loop
    -- Cursor to iterate through each card that the client owns
    DECLARE card_cursor CURSOR FOR
        SELECT CardNumber FROM MadridMetro.Card WHERE ClientID = NEW.ClientID;
    -- Handler for the end of the cursor loop
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    -- Check if the relevant columns are updated
    IF OLD.City <> NEW.City OR OLD.ClientAgeGroup <> NEW.ClientAgeGroup OR OLD.DiscountType <> NEW.DiscountType THEN
        -- Determine the new ZoneCode based on the updated city
        SELECT ZoneCode INTO newZoneCode
```

71:339

```sql
        WHERE DiscountType = NEW.DiscountType;
        -- Open the cursor to process each card
        OPEN card_cursor;
        card_loop: LOOP
            FETCH card_cursor INTO card_number;
            IF done THEN
                LEAVE card_loop;
            END IF;
            -- Calculate the base price for each card using the potentially new zone and age group
            SELECT CASE
                WHEN newZoneCode = 'A' THEN ag.PriceA
                WHEN newZoneCode = 'B1' THEN ag.PriceB1
                WHEN newZoneCode = 'B2' THEN ag.PriceB2
                WHEN newZoneCode = 'B3' THEN ag.PriceB3
                WHEN newZoneCode = 'C1' THEN ag.PriceC1
                WHEN newZoneCode = 'C2' THEN ag.PriceC2
                ELSE NULL
            END INTO newBasePrice
            FROM MadridMetro.AgeGroup ag
            WHERE ag.AgeGroupName = NEW.ClientAgeGroup;
            -- Calculate the final price considering the base price and the discount rate
            SET newFinalPrice = newBasePrice * (1 - newDiscountRate );
            -- Update the RechargePrice for the card
            UPDATE MadridMetro.Card
            SET RechargePrice = newFinalPrice
            WHERE CardNumber = card_number;
        END LOOP;
        CLOSE card_cursor;
    END IF;
END$$
DELIMITER ;
```

SQL Trigger code to update price according if there are any changes to Client's info (ie updates to address, age, discounts)

```sql
-- Trigger to update card price for existing cards
DELIMITER $$

CREATE TRIGGER UpdateCardPriceAfterClientUpdate
AFTER UPDATE ON MadridMetro.Client
FOR EACH ROW
BEGIN
    -- Declare variables to hold dynamic data for price calculation
    DECLARE newZoneCode VARCHAR(20);
    DECLARE newBasePrice DECIMAL(10,2);
    DECLARE newDiscountRate DECIMAL(10,2);
    DECLARE newFinalPrice DECIMAL(10,2);
    DECLARE done INT DEFAULT FALSE; -- Handler flag
    DECLARE card_number INT; -- To hold CardNumber in loop
    -- Cursor to iterate through each card that the client owns
    DECLARE card_cursor CURSOR FOR
        SELECT CardNumber FROM MadridMetro.Card WHERE ClientID = NEW.ClientID;
    -- Handler for the end of the cursor loop
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    -- Check if the relevant columns are updated
    IF OLD.City <> NEW.City OR OLD.ClientAgeGroup <> NEW.ClientAgeGroup OR OLD.DiscountType <> NEW.DiscountType THEN
        -- Determine the new ZoneCode based on the updated city
        SELECT ZoneCode INTO newZoneCode
        FROM MadridMetro.Zone
        WHERE City = NEW.City;
        -- Get the current discount rate from the Discounts table
        SELECT COALESCE(DiscountAmount, 0) INTO newDiscountRate
        FROM MadridMetro.Discounts
        WHERE DiscountType = NEW.DiscountType;

        -- Open the cursor to process each card
        OPEN card_cursor;
        card_loop: LOOP
            FETCH card_cursor INTO card_number;
            IF done THEN
                LEAVE card_loop;
            END IF;
            -- Calculate the base price for each card using the potentially new zone and age group
            SELECT CASE
                WHEN newZoneCode = 'A' THEN ag.PriceA
                WHEN newZoneCode = 'B1' THEN ag.PriceB1
                WHEN newZoneCode = 'B2' THEN ag.PriceB2
                WHEN newZoneCode = 'B3' THEN ag.PriceB3
                WHEN newZoneCode = 'C1' THEN ag.PriceC1
                WHEN newZoneCode = 'C2' THEN ag.PriceC2
                ELSE NULL
            END INTO newBasePrice
            FROM MadridMetro.AgeGroup ag
            WHERE ag.AgeGroupName = NEW.ClientAgeGroup;

            -- Calculate the final price considering the base price and the discount rate
            SET newFinalPrice = newBasePrice * (1 - newDiscountRate );

            -- Update the RechargePrice for the card
            UPDATE MadridMetro.Card
            SET RechargePrice = newFinalPrice
            WHERE CardNumber = card_number;
        END LOOP;
        CLOSE card_cursor;
    END IF;
END$$
```

Trigger 3 Explained: The first trigger (for new card entries) is triggered when an insert occurs in the card table. To illustrate its functionality, let's use an example. Suppose there is a client who is born in 1950, whom lives in Anchuelo, and is eligible for the Senior discount. Trigger 2 would have already determined and assigned an accurate age group category, which is "Senior" in this case. This trigger will then find the appropriate zone code for Anchuelo (B2 in this case), by searching up Anchuelo in the Zone table and storing the value of its corresponding zone code. Using the age group and zone code, the trigger will then search for the pricing of this combination in the AgeGroup table. Finally, it will search for the discount price that corresponds to the client's discount eligibility. In this case it is "Senior" which is a 65% discount. Finally, the trigger will multiple the base price from the AgeGroup table with the discount amount. The computed value is the card fee for a monthly card for that particular individual. When this client registers for a new card, the field titled "RechargePrice" will automatically be updated with the value computed by this trigger.

The value that should be computed = 6.30 (Senior + Anchuelo) * 0.35 (1 – senior discount) = **2.21 Euros.**

This exact example is shown below:

```
230  INSERT INTO MadridMetro.Client (ClientID, FirstName, LastName, Email, PhoneNumber, City, Postcode, StreetAddress, DOB, StudentState, DiscountType) VALUES
231  (3, 'Jonas', 'Lund', 'jonas.lund@gmail.com', '123893231', 'Anchuelo', 28818, 'Pz Matur 1', '1950-10-10', FALSE, 'Senior');
```

Figure: Recharge Price is calculated automatically and correctly for ClientID = 3

CardNumber	ClientID	CardType	ActiveStatus	Balance	IssueDate	LastRechargeDate	RechargePrice
10001	1	Monthly	1	NULL	2024-05-13	2024-05-13	16.00
10002	2	Multi-Use	1	0.10	2024-05-13	NULL	22.30
10003	3	Monthly	1	NULL	2023-10-10	NULL	2.21

To show that the trigger works for updates, let's now change the data of the client. Since we can't change the current date, I'll change the DOB of the client to show the case if the client were to move into another Age Group. I'll also change the city the client lives in (in case client address changes), and the discount that they are eligible for.

Updates:

1) DOB change from 1960 -> 1985 (Senior -> Normal age group)
2) City change from Anchuelo to Madrid Centre (B2 -> A zone code)
3) Discount change from Senior to Big Family Normal (0.65 -> 0.2)

The SQL code for this looks like:

```
233    UPDATE MadridMetro.Client
234    SET DOB = '1985-5-12', City = 'Madrid Centre', DiscountType = 'Big Family Normal'
235    WHERE ClientID = 3;
```

The new value that should be computed = 54.6 (Normal + Madrid Centre) * 0.8 (1 – BFN discount) = **43.68 Euros.**

Figure: Recharge Price for ClientID = 3 is updated automatically to reflect Client info changes

CardNumber	ClientID	CardType	ActiveStatus	Balance	IssueDate	LastRechargeDate	RechargePrice
10001	1	Monthly	1	NULL	2024-05-13	2024-05-13	16.00
10002	2	Multi-Use	1	0.10	2024-05-13	NULL	22.30
10003	3	Monthly	1	NULL	2023-10-10	NULL	43.68

This shows that the triggers are able to dynamically calculate the card fee or “RechargePrice” for each individual based on their current client information and any changes.

SECTION 4: Relational Algebra + SQL Queries

1) Show the First Name, Last Name, Email, Zone Code, and Card Fee for all clients that live in ‘Leganes’.

Relational Algebra:

$$\pi_{C.FirstName,C.LastName,C.Email,Z.ZoneCode,Cd.RechargePrice}$$

$$(\sigma_{C.City='\text{Leganes}'}((C \bowtie_{C.ClientID=Cd.ClientID} Cd) \bowtie_{C.City=Z.City} Z))$$

SQL Query:

```
513 •   SELECT c.FirstName, c.LastName, c.Email, z.ZoneCode, cd.RechargePrice
514     FROM MadridMetro.Client c
515     JOIN MadridMetro.Card cd ON c.ClientID = cd.ClientID
516     JOIN MadridMetro.Zone z ON z.City = c.City
517     WHERE c.City = 'Leganes';
```

Screenshot of Query Result:

FirstName	LastName	Email	ZoneCode	RechargePrice
Shahmeer	Madni	shahmeer.madni@gmail.com	B1	22.30
Pietro	Cannizzo	pietch.can@gmail.com	B1	NULL
Declan	Tierney	d.j@gmail.com	B1	12.00

2) Display the card status for a given client ID no. (I amended it to show the CardNumber too)

Relational Algebra:

$$\pi_{\text{CardNumber},\text{ActiveStatus}}(\sigma_{\text{ClientID}=1}(\text{Card}))$$

SQL Query (? can be any given ID number):

```
522 •   SELECT CardNumber, ActiveStatus
523     FROM MadridMetro.Card
524     WHERE ClientID = ?;
```

Screenshot of Query Result (for ClientID = 1):

CardNumber	ActiveStatus
10001	1