

U-Mobile

System Request

Customers over the last 10 years have indicated that the process of editing their phone/data lines, including adding and activating a new one on their existing plans can be cumbersome; especially since the latter requires a mandatory in-person visit to one of our customer service branches. Although management understood the validity of this, the technology available at the time would not allow for this to be feasible in a way that would not require a wider systems overhaul that would prove very costly, especially given that this process requires complex transactions such as ID validation and the installation of a physical SIM card on the new device.

In the last couple of years, the process has evolved. Smartphone manufacturing companies have begun to incorporate newer technology on their hand-held devices that eliminate the need for a physical SIM card, replacing it with what is known as an embedded SIM, or eSIM, for short. The trend is for this new technology to become the standard, and major firms such as Apple and Samsung already include it in their products.

We now need this process to be deployed as new modules within our website and mobile apps, so that any phone that currently supports SIM (83% of the devices that work within our network currently do) for any customer with an open account with us, will be eligible to carry out these tasks without having to appear in person at one of our branches.

Company Information & Business Problem

- a) **Organization Name:** U-Mobile
- b) **Industry:** Telecommunications (Phone Service Provider)
- c) **Business Problem:** Currently with phone services, customers aren't able to add a line without rigorous in-person identification verification. U-mobile allows for online identification verification of existing plans. Now existing members can add members to their plan online if they have an e-sim. We chose this organization to allow ease for

customers to add lines to existing phone plans without having to undergo long wait times and a service break.

Entity-Attribute Chart

Entity	Attributes
Customer	<u>CustomerID</u> , Name, Address, Email, Phone
Device	<u>IMEI</u> , Manufacturer, SIMType (eSIM? SIM?)
Account	<u>AccountNo</u> , Name, BillingAddress(Street, City, Zip, State), AccountSince
Invoice	<u>InvoiceID</u> , BillingAmount, MinPaymentDue, TotalAcctBalance,
Payment	<u>PaymentID</u> , PaymentAmount, PaymentType(AutoPayment, ManualPayment)
Line	<u>IMEI</u> , LineActiveDate
Plan	<u>PlanID</u> , PlanType, PlanPrice, PlanStartDate

Entity and Attribute Description

ENTITIES:

Consumer: Any physical person requiring the service

Device: Phone device or any device requiring a data connection

Account: Already Existing Lifetime account, directly connected to the consumer's main owner

Invoice: Payment request sent to the account owner

Payment: The actual amount paid by the consumer (the actual transaction)

Line: Single data line per device

Plan: Collection of lines and type of promotion, or deal the consumer is locked into

ATTRIBUTES:

Customer

Customer: 13-digit Identification Number of Individual Person in System.

Phone Number: 10-Digit Number

Account

Account ID: 8 Digit Identification of Account

IMEI: International Mobile Equipment Identity: ID of SIM in Device.

Billing Address: Address Invoices are sent to.

AccountSince: Account Creation Date

Invoice

Billing Amount: Amount Added for Last Month onto the Account Balance

Account Balance: Total Amount Due on Account

Minimum Payment: Minimum Amount of Payment Amount due to keeping the Account Active.

Device

IMEI (International Mobile Equipment Identity): ID of SIM in Device.

Manufacturer: Maker of the Phone

Payment

PaymentID: 9 Digit Unique VARCHAR.

Payment Amount: Must be \geq Minimum Amount on Invoice.

Line

Line ID: 7 VARCHAR that identifies the Line of a Cellular Connection.

LineActiveDate: Date Line was initially activated

Plan

PlanID: Unique 13 VARCHAR that identifies an account plan.

PlanType: Type of Plan (Single or Family)

Plan Price: Monthly Cost of Plan

Plan Start: Date account enrolled into the plan

Plan Duration: Derived by subtracting the current date from Plan Start

Business Rules

Each Customer must have at least one device

A device must belong to only one customer

Each device must have at least one line, but can also have many lines

Each line must have only one device

Each Plan must have at least one line

Each Line must have only one plan

Each Account must have only one plan

Each plan must have only one account

Each account must have at least one or many invoices

Each invoice must have only one account

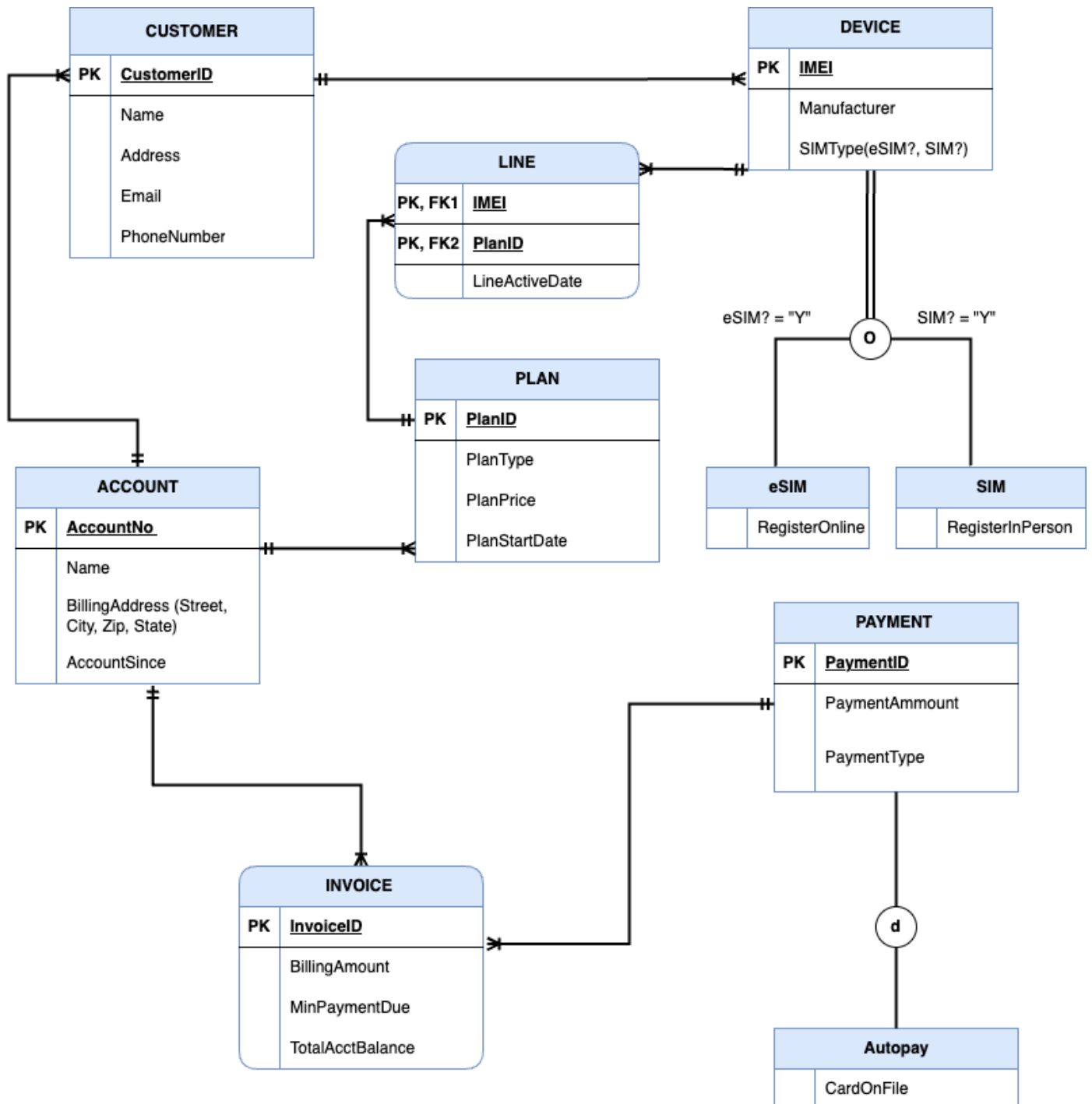
Each invoice must have only one means of payment

Each Payment must have at least one or many invoices

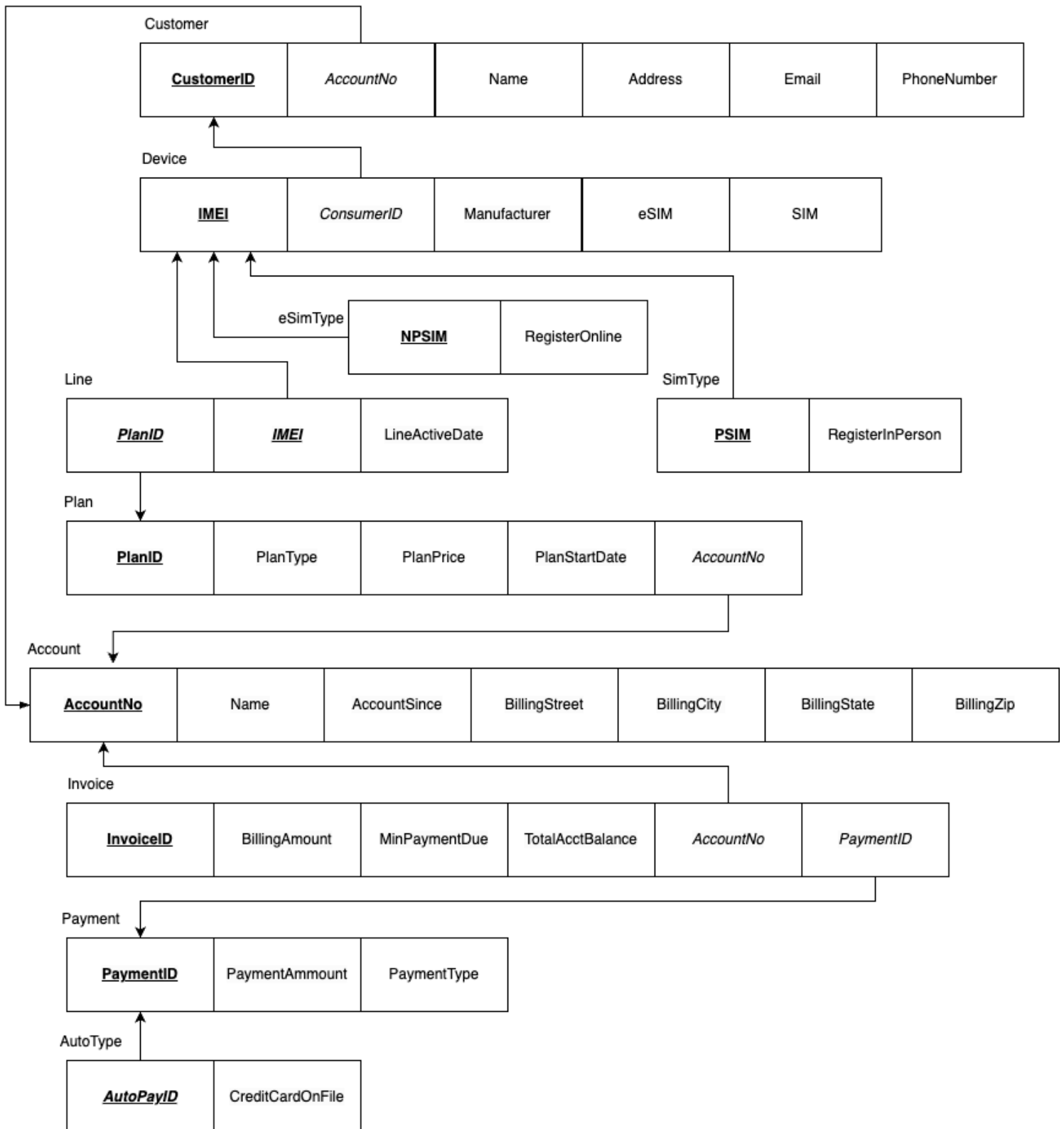
A device may be an eSim device and have a physical SIM at the same time

A payment method may be an autopay method.

Entity Relation Diagram



RELATION MODEL 3NF



Database Creation and Data Entry (SQL Code)

```
CREATE SCHEMA IF NOT EXISTS U_Mobile;
```

```
USE U_Mobile;
```

```
CREATE TABLE Account
```

```
(
    AccountNo          INT          NOT NULL,
    Name                VARCHAR(15)  NOT NULL,
    AccountSince        INT,
    BillingStreet        VARCHAR(15),
    BillingCity          VARCHAR(15),
    BillingState         VARCHAR(2),
    BillingZip           INT          NOT NULL,

    CONSTRAINT Account_PK PRIMARY KEY (AccountNo),
);
```

```
INSERT INTO Account (AccountNo, Name, AccountSince, BillingStreet, BillingCity, BillingState, BillingZip)
VALUES
(778294, 'John Smith', 2012, '789 Oak St', 'Sonoma', 12345),
(219912, 'Allison Moore', 2009, '456 Elm St', 'San Francisco', 56789),
(673354, 'Joe Taylor', 2017, '123 Main St', 'San Bruno', 98765);
```

```
CREATE TABLE Customer
```

```
(
    CustomerID INT NOT NULL,
    AccountNo  INT NOT NULL,
    Name       VARCHAR(20),
    Address    VARCHAR(50),
    Email      VARCHAR(20),

    CONSTRAINT Cust_PK PRIMARY KEY (CustomerID)
    CONSTRAINT Customer_FK FOREIGN KEY(AccountNo) REFERENCES Account(AccountNo)
);
```

```
INSERT INTO Customer (CustomerID, Name, Address, Email, AccountNo)
VALUES
(2356, 'John Smith', '123 Francisco Way', 'jsmith@gmail.com', 778294),
(4567, 'Allison Moore', '442 Charter Ln', 'amooore@yahoo.com', 219912),
(6859, 'Joe Taylor', '5539 Sadowa Ct', 'jtaylor@hotmail.com', 673354);
```

```
CREATE TABLE Device
```

```
(
    IMEI          VARCHAR(15) NOT NULL,
    CustomerID    INT NOT NULL,
    Manufacturer   VARCHAR(15),
    eSIM          VARCHAR(25),
    SIM           VARCHAR(25),
);
```

```
CONSTRAINT Device_PK PRIMARY KEY (IMEI),
CONSTRAINT Device_FK FOREIGN KEY(CustomerID) REFERENCES Customer (CustomerID)
);
```

```
INSERT INTO Device (IMEI, CustomerID, Manufacturer, eSIM, SIM)
VALUES
('479503849411123', 2356, 'Apple', '479503844657830', NULL),
('129305786940322', 4567, 'Samsung', NULL, '129305787810093'),
('745960385628591', 6859, 'Apple', '745960385628591', NULL),
('745960381122456', 2356, 'Motorola', NULL, '745960381122456'),
('129305783345196', 4567, 'LG', NULL, '129305783345196'),
('479503845748294', 6859, 'Samsung', '479503845748294', NULL);
```

```
CREATE TABLE eSimType
(
    NPSIM                VARCHAR(15) NOT NULL,
    RegisterOnline       VARCHAR(1)  CHECK (RegisterOnline IN ('Y'),

    CONSTRAINT Device_PK PRIMARY KEY (NPSIM),
    CONSTRAINT Device_FK FOREIGN KEY(NPSIM) REFERENCES Device (IMEI),
);
```

```
INSERT INTO eSimType (NPSIM, RegisterOnline)
VALUES
('479503849411123', 'Y'),
('745960385628591', 'Y'),
('479503845748294', 'Y');
```

```
CREATE TABLE SimType
(
    PSIM                VARCHAR(15),
    RegisterOnline       VARCHAR(1)  CHECK (RegisterOnline IN ('N'),

    CONSTRAINT Device_PK PRIMARY KEY (PSIM),
    CONSTRAINT Device_FK FOREIGN KEY(PSIM) REFERENCES Device (IMEI),
);
```

```
INSERT INTO SimType (PSIM, RegisterOnline)
VALUES
('129305786940322', 'N'),
('745960381122456', 'N'),
('129305783345196', 'N');
```

```
CREATE TABLE Line
(
    IMEI                VARCHAR(15) NOT NULL,
    PlanID              INT NOT NULL,
    LineActiveDate      DATE NOT NULL,

    CONSTRAINT Line_PK PRIMARY KEY (PlanID, IMEI),
    CONSTRAINT Line_FK FOREIGN KEY(PlanID) REFERENCES Plan (PlanID),
```



```

        CONSTRAINT Line_FK FOREIGN KEY(IMEI) REFERENCES Device (IMEI)
    );
INSERT INTO Line (IMEI, PlanID, LineActiveDate)
VALUES
('479503849411123', 472, '2012-03-12'),
('129305783345196', 003, '2012-02-04'),
('129305786940322', 003, '2009-10-06'),
('745960381122456', 278, '2020-08-19'),
('745960385628591', 278, '2017-06-27'),
('479503845748294', 278, '2018-05-12');

CREATE TABLE Plan
(
    PlanID                INT    NOT NULL,
    AccountNo             INT    NOT NULL,
    PlanType              VARCHAR(15),
    PlanPrice             DECIMAL(2),
    PlanStartDate         DATE   NOT NULL

    CONSTRAINT Plan_PK PRIMARY KEY (PlanID),
    CONSTRAINT Plan_FK FOREIGN KEY(AccountNo) REFERENCES Account (AccountNo),
);

INSERT INTO Plan (PlanID, AccountNo, PlanType, PlanPrice, PlanStartDate)
VALUES
(472, 778294, 'Basic', 79.99, 2012-03-12),
(003, 219912, 'Extra', 99.99, 2010-01-01),
(278, 673354, 'Premium', 119.99, 2017-09-01);

CREATE TABLE Invoice
(
    InvoiceID             INT    NOT NULL,
    AccountNo            INT    NOT NULL,
    BillingAmount         DECIMAL(2),
    MinPaymentDue        DECIMAL(2),
    TotalAccntBalance    DECIMAL(2),

    CONSTRAINT Invoice_PK PRIMARY KEY (InvoiceID),
    CONSTRAINT Invoice_FK FOREIGN KEY(AccountNo) REFERENCES Account (AccountNo),
);

INSERT INTO INVOICE(InvoiceID, AccountNo, BillingAmount, MinPaymentDue, TotalAccntBalance)
VALUES
(14780976 , 778294, 200.50 ,100,1000),
(47370448, 219912, 300.75, 300, 500),
(1859065,673354,500.75 ,250, 2000);

```

```
CREATE TABLE Payment
(
    PaymentID          INT NOT NULL,
    InvoiceID           INT NOT NULL,
    PaymentAmount       DECIMAL(2),
    PaymentType         VARCHAR(10),

    CONSTRAINT Payment_PK PRIMARY KEY (PaymentID),
    CONSTRAINT Payment_FK FOREIGN KEY(InvoiceID) REFERENCES Invoice (InvoiceID),
);
```

```
INSERT INTO Payment(PaymentID, InvoiceID, PaymentAmount, PaymentType)
VALUES
(22861, 14780976, 150, 'CreditCard'),
(75925, 47370448, 175, 'MailCheck'),
(11107, 1859065, 250, 'PayPal');
```

```
CREATE TABLE AutoType
(
    AutoPayID          INT NOT NULL,
    CreditCardOnFile    VARCHAR(50),

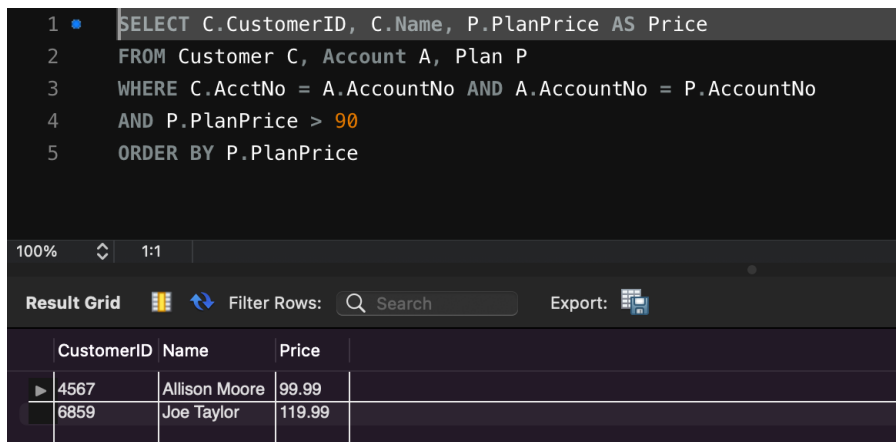
    CONSTRAINT Payment_PK PRIMARY KEY (AutoPayID),
    CONSTRAINT Payment_FK FOREIGN KEY(AutoPayID) REFERENCES Payment (PaymentID),
);
```

```
INSERT INTO AutoType (AutoPayID, CreditCardOnFile)
VALUES
(22861, '2345-1029-8593-1938'),
(75925, '4756-1029-8478-9982'),
(11107, '6628-9984-1829-3744');
```

Querying Database

- 1) Show customerIDs and customer names of customers who have plan prices above \$90. Show the prices from lowest to highest.

```
SELECT C.CustomerID, C.Name, P.PlanPrice AS Price
FROM Customer C, Account A, Plan P
WHERE C.AccountNo = A.AccountNo AND A.AccountNo = P.AccountNo
AND P.PlanPrice > 90
ORDER BY P.PlanPrice
```



The screenshot shows a SQL query editor with a dark theme. The query is entered in a text area with line numbers 1 through 5. Below the query area, there is a toolbar with options for 'Result Grid', 'Filter Rows', and 'Export'. The 'Result Grid' is selected, and it displays the results of the query in a table format. The table has three columns: 'CustomerID', 'Name', and 'Price'. There are two rows of data: one for Allison Moore with a price of 99.99, and one for Joe Taylor with a price of 119.99.

CustomerID	Name	Price
4567	Allison Moore	99.99
6859	Joe Taylor	119.99

- 2) Show customer ID, AccountNo, and customer name for customer(s) who have the same dates for both the line activation and Plan start date

```
SELECT C.CustomerID, C.Name, A.AccountNo, L.LineActiveDate, P.PlanStartDate
FROM Customer C, Plan P, Account A, Line L
WHERE C.AccountNo = A.AccountNo
AND A.AccountNo = P.AccountNo
AND L.PlanID = P.PlanID
AND L.LineActiveDate = P.PlanStartDate
```

```

1
2 SELECT C.CustomerID, C.Name, A.AccountNo, L.LineActiveDate, P.PlanStartDate
3 FROM Customer C, Plan P, Account A, Line L
4 WHERE C.AcctNo = A.AccountNo
5 AND A.AccountNo = P.AccountNo
6 AND L.PlanID = P.PlanID
7 AND L.LineActiveDate = P.PlanStartDate

```

100% 1:2

Result Grid Filter Rows: Search Export:

	CustomerID	Name	AccountNo	LineActiveDate	PlanStartDate
▶	2356	John Smith	778294	2012-03-12	2012-03-12

- 3) Display the Average Minimum Amount and Payment Amount for Each Account Type.
USE U_Mobile;

```

SELECT PL.PlanType, AVG(P.PaymentAmount) AS AveragePayment, AVG(I.MinPaymentDue) AS
AverageAmountDue
FROM Payment P, Invoice I, Account A, Plan PL
AND A.AccountNo = PL.AccountNo
GROUP BY(PL.PlanType);

```

```

1 SELECT PL. PlanType, AVG(P. PaymentAmount) AS AveragePayment, AVG(I.MinPaymentDue) AS AverageAmountDue
2 FROM Payment P, Invoice I, Account A, Plan PL
3 WHERE P. InvoiceID = I.InvoiceID
4 AND I.AccountNo = A.AccountNo
5 AND A.AccountNo = PL.AccountNo
6 GROUP BY (PL. PlanType)
7

```

100% 1:1

Result Grid Filter Rows: Search Export:

	PlanType	AveragePayment	AverageAmountDue
▶	Premium	250.000000	250.000000
	Basic	150.000000	100.000000
	Extra	175.000000	300.000000

- 4) Display Customers with either Apple or Samsung, their Average: Account Balance, Minimum Payment, and PaymentDue

```

SELECT D.Manufacturer AS Brand, AVG (I.TotalAcctBalance) AS AverageBalance, AVG(I.MinPaymentDue) AS
MinimumDue,
AVG(P.PaymentAmount) AveragePayment
FROM Customer C, Device D, Invoice I, Account A, Payment P
WHERE P.InvoiceID = I.InvoiceID AND I.AccountNo = A.AccountNo
AND A.AccountNo = C.AccountNo AND C.CustomerID = D.CustomerID
GROUP BY D.Manufacturer;

```

```

1 SELECT D.Manufacturer AS Brand, AVG (I.TotalAcctBalance) AS AverageBalance, AVG(I.MinPaymentDue) AS MinimumDue,
2     AVG(P.PaymentAmount) AveragePayment
3 FROM Customer C, Device D, Invoice I, Account A, Payment P
4 WHERE P.InvoiceID = I.InvoiceID AND I.AccountNo = A.AccountNo
5     AND A.AccountNo = C.AcctNo AND C.CustomerID = D.CustomerID
6 GROUP BY D.Manufacturer;
7

```

100% 1:1

Result Grid Filter Rows: Search Export:

Brand	AverageBalance	MinimumDue	AveragePayment
Apple	1500.000000	175.000000	200.000000
Motorola	1000.000000	100.000000	150.000000
LG	500.000000	300.000000	175.000000
Samsung	1250.000000	275.000000	212.500000

Result Grid Form Editor

- 5) Show the Device manufacturer, plan ID, plan type, and plan prices for each plan that has a price above \$80.

```

SELECT D.CustomerID, D.Manufacturer AS Brand, P.PlanType, P.PlanPrice, P.PlanID
FROM Device D
INNER JOIN Line L ON D.IMEI = L.IMEI
INNER JOIN Plan P ON L.PlanID = P.PlanID
WHERE P.PlanPrice > 80

```

```

1
2 SELECT D.CustomerID, D.Manufacturer AS Brand, P.PlanType, P.PlanPrice, P.PlanID
3 FROM Device D
4 INNER JOIN Line L ON D.IMEI = L.IMEI
5 INNER JOIN Plan P ON L.PlanID = P.PlanID
6 WHERE P.PlanPrice > 80

```

100% 23:6

Result Grid Filter Rows: Search Export:

CustomerID	Brand	PlanType	PlanPrice	PlanID
4567	LG	Extra	99.99	3
4567	Samsung	Extra	99.99	3
6859	Samsung	Premium	119.99	278
2356	Motorola	Premium	119.99	278
6859	Apple	Premium	119.99	278

Conclusion

For the implementation of our new remote line-adding feature into our company's system, we created a database for the ingestion of customer data. Firstly we defined our entities and their respective attributes within the database, through the creation of the entity-attribute chart. This allowed us to accurately depict the necessary data infrastructure for incoming customer data. Furthermore, we provided descriptions for the entities and attributes. Lastly, we developed our business rules which we used to derive the relationships between the entities within the database.

Moving to our data models, we created both an entity relation diagram and a third normal form relational model to accurately represent the entities and attributes within the schema. Normalization is a tool to validate and improve a logical design so that it satisfies certain constraints that avoid unnecessary data anomalies. These anomalies include insertion, deletion, and modification anomalies.

Next, we created our database in MySQL using SQL data definition language. We wrote SQL statements to create our schema and then populated it with our entity tables. We made sure we incorporated the correct data types to allow for proper data insertion, ensuring integrity and consistency. We populated the tables with sample customer data and queried the database to ensure proper functionality.

Our newly created database will catalyze the enhancement of customer user experience within our telecommunications company, allowing customers to conveniently add additional lines to their existing accounts from the comfort of their homes.